

Hands-on Cybersecurity Studies: Ransomware Key Recovery

by Jaime C Acosta, Adrian J Belmontes, and Salamah Salamah

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.





Hands-on Cybersecurity Studies: Ransomware Key Recovery

Jaime C Acosta Computational and Information Sciences Directorate, CCDC Army Research Laboratory

Adrian J Belmontes and Salamah Salamah University of Texas at El Paso

Approved for public release; distribution is unlimited.

	REPORT D	OCUMENTATIO		Form Approved OMB No. 0704-0188			
Public reporting burden data needed, and comple burden, to Department o Respondents should be a valid OMB control num PLEASE DO NOT	for this collection of informa ting and reviewing the collect f Defense, Washington Head ware that notwithstanding an ber. RETURN YOUR FOR !	tion is estimated to average 1 h tion information. Send comme quarters Services, Directorate f y other provision of law, no pe M TO THE ABOVE ADI	our per response, including ti nts regarding this burden esti or Information Operations ar rrson shall be subject to any p DRESS.	he time for reviewing i imate or any other aspe d Reports (0704-0188) enalty for failing to co	nstructions, searching existing data sources, gathering and maintaining the ct of this collection of information, including suggestions for reducing the b, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, mply with a collection of information if it does not display a currently		
1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE			3. DATES COVERED (From - To)		
April 2020		Technical Report	t		August 2019 – March 2020		
4. TITLE AND SUB	TITLE ersecurity Studies	: Ransomware Key	v Recovery		5a. CONTRACT NUMBER		
Thunds on Cyc	erseeunty Studies		y need very		5b. GRANT NUMBER		
					5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S)					5d. PROJECT NUMBER		
Jaime C Acost	a, Adrian J Belmo	ontes, and Salamah	Salamah				
					5e. TASK NUMBER		
					5f. WORK UNIT NUMBER		
7. PERFORMING C	DRGANIZATION NAM	E(S) AND ADDRESS(ES)			8. PERFORMING ORGANIZATION REPORT NUMBER		
CCDC Army I	Research Laborato	ory			ADI TD 9027		
White Sands M	-KLC-ND Jissile Range-NN	4 88002		AKL-1K-8937			
ti inte Sunus it	instite itunge, i ti	100002					
9. SPONSORING/I	MONITORING AGENC	Y NAME(S) AND ADDR	ESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)		
					11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION	I/AVAILABILITY STAT	EMENT					
Approved for J	public release; dis	tribution is unlimit	ted.				
13. SUPPLEMENT	ARY NOTES						
14. ABSTRACT							
Ransomware in the form of cry downloaded w victim's device and guides par	s a type of malicic ptocurrency. It is ithout the user kn es. This report pre ticipants through	ous software that d typically spread th owing; it can also esents a hands-on e a set of steps that y	enies access to a onrough phishing e spread by taking a xercise that demo will regenerate the	computer syst mails or throu advantage of v onstrates the e e key required	em or files until a ransom is paid, usually in igh websites where the software is rulnerabilities in software running on the ffects of ransomware on vulnerable machines to decrypt the ransomed data.		
15. SUBJECT TERM	ſS						
ransomware, V	VannaCry, key re	covery, encryption	, hands-on cybers	ecurity, Cybe	rRIG		
			17. LIMITATION	18. NUMBER	19a. NAME OF RESPONSIBLE PERSON		
			OF	OF	Jaime C Acosta		
a. REPORT	b. ABSTRACT	c. THIS PAGE		PAUES 26	19b. TELEPHONE NUMBER (Include area code)		
Unclassified	Unclassified	Unclassified	20	(575) 993-2375			

Standard Form 298 (Rev. 8/98) Prescribed by ANSI Std. Z39.18

Contents

List	of Fig	gures	iv					
List	of Ta	bles	iv					
1.	Intr	Introduction						
2.	Setup and Configuration							
3.	. Learning Objectives							
4.	Me	hodology	3					
5.	Exe	rcise	4					
	5.1	Activity 1: Dynamic Analysis of the Process Memory	4					
	5.2	Activity 2: Extracting Encryption Parameters from the Memory Dump	9					
	5.3	Activity 3: Calculating the Missing Encryption Values	12					
	5.4	Activity 4: Regenerate the Private Key	14					
	5.5	Activity 5: Decrypting Files Using the Regenerated Private Key	16					
6.	Con	clusion	17					
7.	Ref	erences	18					
List	of Sy	mbols, Abbreviations, and Acronyms	19					
Dist	ribut	ion List	20					

List of Figures

Fig. 1	WannaCry ransomware indicator	5
Fig. 2	Processes window	5
Fig. 3	Dumping Process window	6
Fig. 4	Path to dump file	6
Fig. 5	Public key (color-coded) consists of header (in red), exponent (in yellow), and modulus (in blue)	7
Fig. 6	Debugger attach to process	8
Fig. 7	Windows Restart screenshot	8
Fig. 8	Format with color-coded values	10
Fig. 9	Encryption numbers in DMP file	11
Fig. 10	Prime 1 and prime 2 placement	13
Fig. 11	Public exponent placement	13
Fig. 12	Special header entry	14
Fig. 13	Key entry	15
Fig. 14	Key structure	16
Fig. 15	Decrypt button	16

List of Tables

Table 1	Key structure	. 9
Table 2	Key structure values found up to this point	12
Table 3	Key generation status	15

1. Introduction

Ransomware is a type of malicious software that denies access to a computer system or files until a ransom is paid, usually in the form of cryptocurrency. It is typically spread through phishing emails or through websites where the software is downloaded without the user knowing; it can also spread by taking advantage of vulnerabilities in software running on the victim's devices. This report presents a hands-on exercise that demonstrates the effects of ransomware on vulnerable machines and guides participants through the steps that will regenerate the key required to decrypt the ransomed data.

When the WannaCry ransomware crypto worm spread worldwide in 2017, consumers and large corporations felt the impact.¹ Critical data were locked away and made inaccessible to owners until a ransom was paid in the form of anonymous currency known as bitcoin. Over 200,000 victims were affected by this attack, including the National Health Service hospitals in England and Scotland. Even though a kill switch was found and implemented, variants of this malware continue to arise.

The variant that we reference in this report uses the Rivest, Shamir, and Adleman (RSA) encryption method² to lock (ransom) files. This method uses a pair of keys, known as public and private keys.³ The former is used to encrypt while the latter is used to decrypt data. The malware removes the private key from the victim's machine and provides it only after a ransom is paid.

This report demonstrates how a machine becomes infected, allowing participants to experience the effects of the malware. Participants are then guided through the process of identifying and recovering data that is inadvertently left behind by the malware after encrypting files. The data are used to regenerate the private key, and decrypt and regain access to the ransomed files. This hands-on exercise is related to a previous exercise that we developed, showing how to analyze and implement a kill switch for the WannaCry ransomware.⁴ It is based on the program logic from the wanakiwi⁵ software, developed by Benjamin Delpy and made available on GitHub.

2. Setup and Configuration

The hands-on exercise consists of two virtual machines: one is used to infect a machine and create a memory dump, and the other is used to analyze the memory dump and public key to look for prime numbers and construct a private key. The setup configuration consists of the following software elements:

- VirtualBox (Version 6.0)
- Two Windows 7 Home Basic 32-bit virtual machines
- LibreOffice 6.4.a
- IDA Pro Free (Version 5.0)
- HxD Hex Editor (Version 2.3.0.0)
- WannaCry malware variant with MD5 hash: ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
- RSA Key Generator⁶
- Little/Big Endian Converter

The first machine, named Ransomed, was set up with IDA Pro Free, HxD Hex Editor, and the WannaCry variant. The second machine, named MemoryAnalysis, was set up with LibreOffice, HxD Hex Editor, and the RSA Key Generator. In addition, the MemoryAnalysis machine was updated to include the WannaCry patch to prevent the malware from infecting the machine. Finally, a shared folder was set up to allow file transfers between the two machines.

The entire exercise runs on the US Army Combat Capabilities Development Command (CCDC) Army Research Laboratory (ARL) South Cyber Rapid Innovation Group (CyberRIG) Collaborative Innovation Testbed (CIT), which provides an isolated environment, ensuring that all the environmental artifacts are segregated from any real systems.

3. Learning Objectives

This exercise will teach participants the following points:

- Participants will have a better understating of how ransomware works, specifically the WannaCry crypto worm. The effects of the ransomware on the sandbox environment should emphasize the importance of having files and other data backed up.
- Participants will gain experience in creating memory dumps and extracting necessary data by examining memory using the IDA Pro software. Working in a timely manner to avoid losing data in memory is important.
- Participants will gain a better understand of RSA encryption. They will learn how a private key is constructed as they calculate all of the necessary values and constructs using the HxD Hex Editor software and RSA Key Generator.

4. Methodology

In creating this exercise, we started by downloading Benjamin Delpy's wanakiwi software onto a virtual machine with the Windows 7 32-bit Operating System. According to the software documentation⁵, this version of Windows is the most recent platform on which wanakiwi was written to work correctly. We found that the software works on both Windows 7 32-bit Pro and Home versions. We later realized that the wanakiwi software does not work if additional updates are installed (including those required to recompile the wanakiwi software). Therefore, we created two separate virtual machines: MemoryAnalysis and Ransomed.

To fully understand the inner workings of the software, we traced through the source code. We installed the latest software updates and patches on the MemoryAnalysis machine as well as the Visual Studio integrated development environment. The Ransomed machine was left untouched, except for the inclusion of the wanakiwi executable binary.

We modified the source code to skip some of the longer procedures and to narrow down where, in memory, the prime numbers are placed when the ransomware infects a machine. The Visual Studio debugger was extremely helpful in tracing through the statements that traversed memory.

To support this interactive exercise, we chose IDA Pro because it is an ideal tool to help participants replicate the logical steps in wanakiwi. Using the prime numbers found with wanakiwi, IDA Pro was able to attach to the ransomware process and show their location in the virtual address space.

We encountered slight differences when comparing the wanakiwi logic and the standard RSA algorithm: the key formatting within the file that stores the key values was not the same. The ransomware used an altered form, with parameters in a different order and an additional value—a *magic* number that seemed to indicate a signature for the malware key that existed as a precursor to the standard values. To calculate the key values, we used an RSA Key Generator website. Values in memory are represented in little-endian format, which is expected by the ransomware. However, the RSA Key Generator expected values in big-endian format, so we developed a small application to handle these conversions.

The final step was to understand how the prime numbers could be found in memory without relying on the wanakiwi application. Wanakiwi uses a brute force approach—it iterates through memory and attempts to compute sequences of bytes and then determines if the values are prime. This was computationally expensive and time consuming, so we looked for another way. We first attempted to find the entire private key in memory using IDA Pro, but this did not work. We noticed that

after infection, a public key is placed on the user's desktop. Looking deeper into key structure, we searched memory for the individual values. Using the modulus, which is a product of the two prime numbers, yielded success. Consequently, the prime numbers were always only a few bytes away from the modulus.

To address the issue of timing and overwriting key in-memory values, participants must create a nonvolatile image immediately after infection. We task participants with creating a memory dump of the ransomware process. To demonstrate the importance of this task, participants must then reboot their machines; they will find that the values required for recreating the keys are no longer resident in memory or anywhere on their machine.

5. Exercise

This exercise is separated into five main activities:

- 1) Infect one machine and take a memory dump of the malware process.
- 2) Analyze the memory dump to recover the two prime numbers.
- 3) Calculate the remaining variables needed to create a private key.
- 4) Combine all variables to create a private key.
- 5) Decrypt the infected machine using the generated key.

The exercise requires about 1.5-2 h to complete.

5.1 Activity 1: Dynamic Analysis of the Process Memory

Run the WannaCry malware in your sandbox and observe its behavior on the machine.

1) Start the Ransomed computer by clicking the corresponding link in your browser. You should see a Windows desktop. If prompted, log in using the following credentials:

Username: Reversing

Password: malware



Open WannaCry by double-clicking this file edotebtes. on the desktop. Click No on any prompts that may come up.

3) The program will execute and finish once you receive the message shown in Fig. 1.



Fig. 1 WannaCry ransomware indicator

Be sure to create a memory dump of the malware process to analyze later. This has to be done as soon as possible to ensure the ransomware and the operating system do not remove important artifacts that are essential for undoing the chaos.

- Open Task Manager by clicking the Windows start key and searching for Task Manager. Select View running processes with Task Manager.
- 5) Click the **Processes** tab and then **Show processes from all users.** Find the WannaCry process (it starts with **ed01ebfbc...**). See Fig. 2.

uwiii.exe	USEL1	00	190 K	Desktop wi
ed01ebfbc9eb5	User1	00	14,064 K	DiskPart
explorer exe	Liser1	00	23 916 K	Windows F

Fig. 2 Processes window

6) Right-click and select **Create Dump File**. You should see the screen shown in Fig. 3.



Fig. 3 Dumping Process window

100

7) Open the File Explorer icon and enter the path in the path bar as shown in Fig. 4.



Fig. 4 Path to dump file

- 8) Drag and drop the memory dump file to the desktop.
- 9) Copy a few files to a shared folder in order to analyze them on a separate machine. Double-click the folder on the desktop named **Shared Folder**.
- 10) Locate the public key file named **0000000.pky** on the desktop. Copy both this **public key file** and the **memory dump file** to the shared folder.
- 11) Open **HxD Hex Editor** by double-clicking the **I** icon on the desktop. Then click **File->Open** and select the **public key** located on the desktop.

The next step is to determine the attacker's private key, but in order to do so, we need to look at a few values from the public key. The public key is shown in Fig. 5 and consists of a **special header** (red), the public **exponent** (yellow), and the **modulus** (blue). Figure 5 is colored to identify components easier. When using HxD, the components will not be colored.

	•					0000	0000.pky		
0	06020000	00A40000	52534131	00080000	01000100	576DAFDB	B506E706	63CB58EA	§ RSA1 WmØ∈µÁcÀXÍ
32	340B1D4E	EEC9A099	6C47BD76	50D269AC	A19A4246	846A1AC4	E46B352F	95E5FB1C	4 NÓ†ôlGΩ∨P"i¨°öBFÑj f‰k5/ïÅ'
64	F4A4CA58	1E5331B0	ØA9AC548	D6DCEB87	32B3026F	997C9C76	7CFC20C7	43CE6205	Ù§ X S1∞ ö≈H÷<Îá2≥ oô∣úv∣, «CŒb
96	8DA1B7B2	88D0CFD8	3DE84DC4	7D0971F3	8C603BD9	40403464	B6184496	47020BEE	ç°∑≤à-œÿ=ËMf} qÛå`;Ÿ@@4d∂ DñG Ó
128	9FF06C82	0005208C	94206FC2	12465588	2F5BBEB4	68A5F7B5	7B075F97	2A39EEA2	ü∉lÇ åî o⊣ FUà/[œ¥h•~µ{ _ó*9Ó¢
160	E62A1AA6	88DDA5C3	D07F9566	AA9AE025	5A9882B9	2B78D21C	57D04990	629D59C4	Ê* ¶à>•√- ïf™ö‡%ZòÇπ+x" W-IêbùYf
192	EF5263E1	059AE998	41D27225	9643ED91	41A4A8DD	895F7C26	7F201A8D	E9C79D64	ÔRc• öÈòA"r%ñCÌëA§®>â_ & çÈ«ùd
224	B52A4F8A	440E3FC2	FØA87E56	5CCADA29	9F92B4AD	2CFB74D4	2C59B76B	C0335F7E	µ*0äD ?¬∉®~V\ /)üí¥≠, 't',Y∑k¿3_~
256	AB3628D9	54F3400B	4A738D5F	4DAFD468	D89B76BA				′6(ŸTÛ@ Jsç_MØ'hÿõv∫

Fig. 5 Public key (color-coded) consists of header (in red), exponent (in yellow), and modulus (in blue)

Note an issue with the values: they are in reverse order. For example, if we encounter the eight digits 12 34 56 78, we would need to convert them to the following: 78 56 34 12. This is a known issue in computing called little versus big endianness. Consider the public exponent bytes in little endian: 01 00 01 00. The actual value of these bytes would be 00 01 00 01. The bytes are reversed. Follow the steps below to notate the correct order of the bytes.

- 12) Look at your hex editor and identify your public exponent. Write down your public exponent **as displayed in the hex editor**.
- 13) **Reverse** the bytes from number 12 as described previously and write them here. (Hint: it should match the previous example.)
- 14) Look at your hex editor and identify your modulus. What are the **first eight characters (or four bytes)** of the modulus, **as displayed in the hex editor**?
- 15) Look at your hex editor and identify your modulus. What are the **last eight** characters (or four bytes) of the modulus, as displayed in the hex editor?
- 16) **Reverse** the bytes from number 15 as described previously and write them here.

Use IDA Pro to search for a few more numbers that were present in the memory dump. You will then simulate the effects of restarting the machine in the hopes that everything will go back to normal.



17) Open up **IDA Pro Free** by double-clicking the ^{IDA Pro Free} icon on the desktop. Then click **GO**. 18) On the toolbar, click Debugger -> Attach -> Local Windows Debugger. Then look for the WannaCry process (it starts with ed01ebfbc...) and click OK (Fig. 6).

Choose	e process to attach to
ID	Name
1020	dlihost.exe
1716	msdtc.exe
1668	SearchIndexer.exe
3356	svchost.exe
3396	sppsvc.exe
3452	svchost.exe
2752	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5b
3852	SearchProtocolHost.exe
2476	taskhsvc.exe
2248	conhost.exe
2832	@WanaDecryptor@.exe
4000	taskdi exe
•	↓
	OK Cancel Help Search
Line 32 of	39 4

Fig. 6 Debugger attach to process

- 19) When the main IDA Pro window opens, click the **Play** icon just below the toolbar and then click the **Pause** icon.
- 20) Now open the search window by clicking **Search -> sequence of bytes**. The search window will take a few seconds to open.
- 21) In the String box, type in your answer for number 16. Make sure **Find all occurrences** is selected. How many occurrences did it find?
- 22) Click the **Windows** button and select **Restart** (Fig. 7). Force quit any processes that are active.



Fig. 7 Windows Restart screenshot

23) After restarting, run the WannaCry icon on the desktop again.

24) Repeat steps 17–21.

25) How many occurrences did it find this time?

26) Did the number of occurrences change? Write down a few reasons why you think this is the case.

5.2 Activity 2: Extracting Encryption Parameters from the Memory Dump

Recall that in the previous activity, you froze a copy of the process data before you restarted. Now you can look in that image to find all of the values needed to recreate the decryption key.

Before building the private key, we will take a closer look at the full structure of the RSA keys. The RSA structure contains several items in addition to the ones you captured previously; you will need many of these to recreate the private key. The memory dump will provide us with the two prime numbers, and the rest can be obtained with a calculator. Table 1 shows the values that make up a private key structure, and Fig. 8 shows a sample public key with values.

Name	Size	Variable name in RSA equation	Color-code
Heading	16 bytes (32 characters)	N/A	Red
Public Exponent	4 bytes (8 characters)	Е	Yellow
Modulus	256 bytes (512 characters)	Ν	Blue
Prime Number 1	128 bytes (256 characters)	Р	Green
Prime Number 2	128 bytes (256 characters)	Q	Orange
CRT Exponent 1	128 bytes (256 characters)	dP	Purple
CRT Exponent 2	128 bytes (256 characters)	dQ	Light Blue
CRT Exponent Coefficient	128 bytes (256 characters)	qInv	Black
Private Exponent	256 bytes (512 characters)	D	Pink

Table 1Key structure

•						00000	0000.dky		
0	07020000	00A40000	52534132	00080000	01000100	576DAFDB	B506E706	63CB58EA	§ RSA2 WmØ∈µÁcÀXÍ
32	340B1D4E	EEC9A099	6C47BD76	50D269AC	A19A4246	846A1AC4	E46B352F	95E5FB1C	4 NÓ…†ôlGΩ∨P"i¨°öBFÑj f‰k5/ïÅ [*]
64	F4A4CA58	1E5331B0	ØA9AC548	D6DCEB87	32B3026F	997C9C76	7CFC20C7	43CE6205	Ù§ X S1∞ ö≈H÷<Îá2≥ oôlúv∣, «CŒb
96	8DA1B7B2	88DØCFD8	3DE84DC4	7D0971F3	8C603BD9	40403464	B6184496	47020BEE	ç°∑≤à-œÿ=ËMf} qÛå`;Ÿ@@4d∂ DñG Ó
128	9FF06C82	0005208C	94206FC2	12465588	2F5BBEB4	68A5F7B5	7B075F97	2A39EEA2	ü∉lÇ åî o¬ FUà/[œ¥h•~µ{ _ó*9Ó¢
160	E62A1AA6	88DDA5C3	D07F9566	AA9AE025	5A9882B9	2B78D21C	57D04990	629D59C4	Ê* ¶à>•√- ïf™ö‡%ZòÇπ+x" W-IêbùYf
192	EF5263E1	059AE998	41D27225	9643ED91	41A4A8DD	895F7C26	7F201A8D	E9C79D64	ÔRc∙ öÈòA"r%ñCÌëA§®>â_ & çÈ«ùd
224	B52A4F8A	440E3FC2	FØA87E56	5CCADA29	9F92B4AD	2CFB74D4	2C59B76B	CØ335F7E	μ*0äD ?¬ s ®~V\ /)üí¥≠, t',Y∑k¿3_~
256	AB3628D9	54F3400B	4A738D5F	4DAFD468	D89B76BA	695B9755	2A768982	53F30D31	í6(ŸTÛ@ Jsç_MØ'hÿõv∫i[óU*vâÇSÛ 1
288	67683627	8E606A57	7460FA6B	56AF7FD5	9DB136C1	ØA66E518	40EAEAE5	D8F4A7B2	gh6'é`jWt``kVØ 'ù±6¦ f @ÍÍÂÿÙß≤
320	F6662337	3BB59EAE	E279A083	5C270FDA	8E62DC18	FF81F727	BØ3E875B	E30759FB	^f#7;µûÆ,y†É∖' ∕éb< čÅ~'∞>á[" Y`
352	E2CEB1D0	ØBB83214	A87F20C6	2C87BFAC	5F8DFC20	BCAB5853	ØBBØABEB	A3487AD0	,Œ±- ∏2 ® Δ,áø"_ç, °´XS ∞´Î£Hz-
384	5B3E8258	BDA46204	ØF9BBE6C	6B13A22E	59C888F2	BF2A4BEB	B4792312	74CB9895	[>ÇXΩ§b õœlk ¢.Y»àÚø*KÎ¥y# tÀòï
416	426091C8	0069E2EA	1FB1C18A	8A2F0953	A7D42EAB	4324852A	64AC26DA	6381C262	B`ë» i,Í ±¡ää/ SB'.´C\$Ö*d¨&/cŬb
448	A80ACB59	36F25133	B7038E72	278B30A1	B1AE3461	2BE19B13	B5D5F35F	05C487FA	® ÀY6ÚQ3∑ ér'ā0°±Æ4a+∙õ µ'Û_ fá'
480	952BEAA2	6A17EE84	EA8C2DE6	AD3ACE43	C9C6EC16	496B8206	C40F3E34	780E5558	ï+Í¢j ÓÑÍå-Ê≠:ŒC…ΔΪ IkÇ ƒ >4x UX
512	86437659	64A86441	01397042	640E7DDF	0CDFD0C4	09F3099A	C9CE1C68	35ED7DC3	ÜCvYd®dA 9pBd }fl fl-f Û ö…Œ h5Ì}√
544	086E1291	CDFDCBA4	8F9ED1AE	0F42842F	693E395A	9098F397	4C130D24	0E54A49F	n ëÕ″À§èû–Æ BÑ∕i>9ZêòÛóL \$ T§ü
576	1326720D	2DC3A623	6152EA8E	2EAF82F1	D66021EA	96D5C237	851A03E3	9510F032	&r -√¶#aRÍé.ØÇÒ÷`!Íñ'¬7Ö "ï ∉2
608	C04471EF	9674083C	BDØ2E9CC	9EF94CEF	D1D088F8	BC932B4A	94C915BA	955BA400	¿DqÔñt <Ω ÈÃûĭLÔà⁻°ì+Jî… ∫ï[§
640	E236325C	30B12A16	4984C1D8	07CCDFAF	19E1F04D	652B7A35	E27262BD	9EE3892C	,62∖0±* IÑįÿ ÃflØ ∙∉Me+z5,rbΩû"â,
672	04D8886B	E3EFDF12	C7C9422C	A438532A	7DF938BF	ØD260AAE	AA6BB32C	370760DD	ÿàk"Ôfl «…B,§8S*}`8ø & Æ™k≥,7 `>
704	4C62DDC4	427033DF	D5AA7966	F67C13F0	BFA2B8E5	63459E22	A8937BE3	5D9E45F8	Lb>fBp3fl'™yf^∣ ∉ø¢∏ÂcEû"®ì{"]ûE⁻
736	20E20F54	A212D112	7BFC2646	93684C02	2458ED97	ED7AEBFA	C018F3D9	468CFEBD	, T¢ – {ˌ&FìhL \$XÌóÌzÎ ¿ ÛŸFå,Ω
768	4A211196	8652A57F	804DFCC4	3AB43756	393FDF1C	ØADD2423	2EC9EØBE	EDØ4B769	J! ñÜR● ÄM,f:¥7V9?fl >\$#‡œÌ ∑i
800	95580A1D	438723C3	3D4BB052	69049A5F	B8596B8E	7F180BA2	FD42DF22	884E7E2E	ïX Cá#√=K∞Ri ö_∏Yké ¢″Bfl"àN~.
832	7DD0406A	49F32A7F	170E3EDD	DE99A181	784C5730	72E77676	659E75F0	54977AC9	}-@jIÛ* >>fiô°ÅxLW0rÁvveûu ∉ Tóz…
864	BE89DFFØ	1CA46252	4FA4C128	10EAB3E1	6CDF935E	3DA49276	1F404B42	712713E8	œâfl ∉ §bRO§¡(Í≥·lflì^=§í∨ @KBq' Ë
896	E299B3C7	C666526A	FC99B2F6	427E3E19	AF767E1B	413A2CC6	9050B71F	21EE4469	,ô≥«∆fRj ô≤^B~> Øv~ A:,∆êPΣ !ÓDi
928	397DD4FB	6F09AA4D	F0409547	457D7D04	EEAE26A6	13942AEA	883D5100	C76AD43D	9}''o ™M e @ïGE}} ÓÆ&¶ î*Íà=Q «j'=
960	3DDDBØBC	6EA835FE	EA05F1E6	CAD6EA94	825DE489	B045B0C0	A65B1882	8D331409	=>∞°n®5,Í ÒÊ ÷ÍîÇ]‰â∞E∞¿¶[Çç3
992	B1E17081	B55437FB	1BFDE112	955DDF5E	1642B730	61C1B57F	ED6CF669	7300FC45	±·pÅμT7° ″· ï]fl^ BΣ0a;μ Ìl^is .E
1024	87FFB30D	ØDF316EC	17EC9E8C	40B1C161	4308A4F2	49EA6A6F	C6A6398C	FØA3E76A	á`≥ Û Ï Ïûå@±;aC §ÚIÍjo∆¶9å∉£Áj
1056	2AAC7423	1B2EFD4A	F7E41542	958A10E8	57F06006	AØDDAA2E	5CF39B52	ØF373575	*"t# .″J~% Bïä ËW∉` †>™.\ÛõR 75u
1088	C287AC87	9EC7EA45	41F93A1A	CD71D6A4	41EFFD7A	D8F70F1D	1969F075	E7054A2F	¬á¨áû«ÍEAĭ: Õq÷§AÔ″zÿ″ i∉uÁ J∕
1120	47357EA4	A2B6DAA9	FB4ADC0B	909B5177	D301376A	6249B931	F37A2B92	4ADA3680	G5~§¢∂/©ʻJ< êõQw" 7jbIπ1Ûz+íJ/6Ä
1152	BB173B8C	637AD3E7	195CØEAB	67EØC5B5	8A282852				° ;åcz"Á ∖ ´g‡≈µä((R
Signe	ed Int 🗘	little 🗘 (s	elect some c	lata)					-+
	0 out of 1172 bytes								

Fig. 8 Format with color-coded values

- 1) Exit the current Windows machine by clicking the Back button in your browser. Start the MemoryAnalysis by clicking on the corresponding link.
- Open HxD Hex Editor by double-clicking the icon on the desktop. Then open the public key that you stored in the shared folder (00000000.pky).

When WannaCry creates the public key, it uses two prime numbers to calculate a key value. The primes are stored in memory until they are overwritten or cleared. Creating the dump file in a timely manner decreases the chance of losing that data. (Note: newer machines clear the memory right away.)

- Click File -> Open and select the DMP file on the shared folder (it starts with ed01ebfbc...).
- On the toolbar click Search -> Find. Then click the Hex-values tab and select All for the search directions.
- 5) In the search bar, enter your **answer from Step 1, number 15** and click **Search all**.
- 6) Double-click through the occurrences and find the one that contains the following characters right after the value you searched: **11000110**.

Now we need to extract a few numbers from the rest of this memory dump. They are mixed with other numbers, so we will have to skip a few numbers and paste only what we need into a temporary spreadsheet file named **Calculations Template**.

- 7) Open **Calculations Template** by double-clicking the ^{Calculations} icon on the desktop. In the next steps, you will fill in the green and blue cells.
- 8) Enter you answer from **Step 1**, **number 12** under **Public Exponent (little endian)**.
- 9) Go back to the HxD application (Fig. 9; your values). From the end of modulus (the number you just found), look eight rows up to find the value from Step 1, number 15. This is the start of the modulus.
- 10) Highlight the bytes corresponding with the entire modulus (the length of bytes you select can be seen at the bottom of the application labeled "Length(d):" make sure it is 256 bytes). Copy and paste the modulus into the Calculations Template spreadsheet under Modulus (little endian). See Fig. 9.

		\frown						
	7E8A0008	CD11CECA	A746BB62	DC2AEE06	A47C923E	5AC1D975	797920C3	9607CE2E
start of modulus	0EEF0178	91170508	5797810F	A0E6F5DC	1E2F7C19	F6618BC7	D6B568FC	948D6CDB
(same value as	3C31ECE3	CD2EE245	35B0DADD	0B9A410B	549594C1	B5F66032	E17C6BA8	3D02ED06
Ctan 1 must an	A688670A	E2DA5BC7	55A6FD0E	10A4414B	EDC5E04D	BE67A16E	C4A2CCEC	E4D7AF7E
Step 1 number	5A8CDABA	0832C102	BDAAE113	A8A2DA1A	04767FE9	68838E79	62E4750A	3F2DC333
<i>14</i>)	7C10E73A	EC84EF1F	4DFE95A4	C619CDBA	40E7527F	5F24D14C	1CF307EB	088E45E1
	8F229EB8	BE1C6D32	2A1309CB	B4A290AB	4980DF67	72978443	9BD04720	D85393B8
	CD607206	7934973C	D5E15425	DC27E55A	0D1B8162	AF74AD67	E065013C	03BC55D7
	1F5C82E2	11000110	5F8A0008	31A5B4B7	F0A2E968	21E6269B	F549ADEF	2C5795C1
				🔍 star	t of fir	*		
end of modulus 🖊		skin 16	characters	star	t of III	si		
(from Step 1		5KIP 10	cnuraciers	prin	ne number			
(nom Step 1,				-				
number 15)								

Fig. 9 Encryption numbers in DMP file

- 11) After the end of the modulus, skip 16 characters (8 bytes); the next 256 characters (128 bytes) make up the first prime number (Fig. 9). Highlight the bytes corresponding with the first prime number (the length of bytes you select can be seen at the bottom of the application—make sure it is 128 bytes) and then copy and paste them into the spreadsheet under Prime 1 (little endian).
- 12) After the first prime number, skip another 16 characters (8 bytes); the next 256 characters (128 bytes) make up the second prime number. Highlight and copy the second prime number and place it under Prime 2 (little endian). The length of bytes you select can be seen at the bottom of the HxD application—make sure it is 128.

We have found all of the values we need from our memory dump. The next step is to get the reverse of some of the numbers (or the **big endian** version) for the modulus and primes.



13) Open the Endian Converter by double-clicking the Shored icon on the desktop. From the Calculations Template copy over the values under Modulus (little endian), click Convert and then paste the result back into the Calculations Template under Modulus (big endian).

If the converter prompts you that your number is not even, make sure there is no space at the end. If there is no space, add a 0 (zero) to the start of your number and try again.

14) Repeat the conversion process from number 13 for all of the little-endian values in your spreadsheet (primes, public exponent, special header). (Hint: Public Exponent [big endian] should match your answer from Step 1, number 13.)

5.3 Activity 3: Calculating the Missing Encryption Values

Table 2 shows the mapping between values and their variable name in RSA equations.

Name	Variable name in RSA equation	Status
Heading	N/A	Found
Public Exponent	e	Found
Modulus	n	Found
Prime Number 1	р	Found
Prime Number 2	q	Found
CRT Exponent 1	dP	Need to Calc
CRT Exponent 2	dQ	Need to Calc
CRT Exponent Coefficient	qInv	Need to Calc
Private Exponent	d	Need to Calc

Table 2Key structure values found up to this point

You will need to calculate a few more numbers using the script located on your desktop.



- 1) Open the **RSA key generation** script by double-clicking the RSA key generation on the desktop. Scroll down until you see the **Input online RSA key generation form**.
- 2) Copy and paste the **PRIME 1 (big endian)** and **PRIME 2 (big endian)** from **Calculations Template** into the boxes (order does not matter; see Fig. 10).



Fig. 10 Prime 1 and prime 2 placement

3) Scroll down further until you see Step 2: Enter public exponent. Enter your value for the public exponent (big endian) from the Calculations Template, and then click on the Generate Keys button (Fig. 11).

	Ster	p 2: Enter public exponent	
	Public exponent (e) is a *: 🕜 hex	xadecimal 🔻	
Enter public	Public exponent (e)*:	010001	Press here to generate all encryption
(big endian)		Demo 1 Clear	parameters (including private key!)
	Step 3: Generate public / pri	ivate keys based on prime numbers and	exponent
	Convert generated keys to *: () hex	xadecimal 🔹	Generate keys

Fig. 11 Public exponent placement

- 4) Scroll down and verify that the **modulus** matches what you have in your spreadsheet.
- 5) Look through the form and copy the private exponent (d) into the private exponent (big endian) on your spreadsheet. Continue through and also copy into the spreadsheet the big endian values for CRT exponent 1 (dP), CRT exponent 2 (dQ), and CRT coefficient (qInv).
- 6) You should now have all of the values in your spreadsheet under the **big** endian column filled. Use the Endian Converter to calculate and then fill in the missing values in the little-endian column.

Once you are done, close all other windows except for the **Calculations Template** and the **HxD application**.

The values you just added are part of the creation of RSA keys; however, the formatting and layout may differ.

5.4 Activity 4: Regenerate the Private Key

You should now have all of the values you need to regenerate the private key. Now you need to put them in the correct order (required by the ransomware) using the hex editor.

 Create a new file by clicking File->New to begin creating the private key. Click on File->Save As... and name the file 00000000.dky (eight zeroes) on the desktop. You can switch between the two files using the tabs at the top.

The key that WannaCry reads uses the **little-endian** format; you will need to create the key using the values obtained in the previous steps.

2) First, create the special header (Fig. 12). Make sure that your cursor is at the beginning of the file. Copy the value from your spreadsheet under the Special Header, Little Endian column. If a warning appears, select the don't ask again box and continue.

1	🐱 HxD - [C:\Users\Reversing\Desktop\Shared Folder\0000000.dky]									
	File Edit Search View Analysis Tools Window Help									
	🗋 🚵 👻 💭 🔳 😃 🔄 🔹 32 🔹 Windows (ANSI) 💿 dec 💿									
ſ	📓 0000000.dky									
	Offset(d)	00	04	08	12	16	20	24	28	Decoded text
	00000000	07020000	00A40000	52534132	00080000					¤RSA2[]

Fig. 12 Special header entry

3) Next, copy over the Public Exponent, Little Endian Column and the Modulus, Little Endian Column. At this point, your file should have values through eight full rows and five columns in the ninth row. See Fig. 13.

€ HxD - [C:\Users\User\\Desktop\:0000000.dky]									
🔝 File Edit Search View Analysis Tools Window Help									
🗋 👌 🖌 📙 🗌	🗋 🚵 🔻 🔄 🔝 🔹 32 💿 Windows (ANSI) 💿 dec 💿								
20000000.dky									
Offset(d)	00	04	08	12	16	20	24	28	Decoded text
00000000	07020000	00A40000	52534132	00080000	01000100	97ACE489	082270AA	750B0F03	¤RSA2a‰."pªu
00000032	E6A0C92F	74197206	C3FEE3EA	ECEFD7F7	D03AC015	E2C311EF	A9F48EA3	A307936D	æ É/t.r.Äþãêìï×÷Ð:À.âÃ.ï©ôŽ££."m
00000064	D389B72D	70B84EB5	22DC8E93	56B53A5A	95DE4110	0975E4F2	F9C3A525	88C9207A	Ó‰·−p,Nµ"ÜŽ"Vµ:Z•ÞAuäòùÃ¥%^É z
00000096	2757E92E	F53AEF8B	B7FFEF0A	406F480C	F262DB0E	20B37908	9A003DF7	369C2606	'Wé.õ:ï< ∵ÿï.@oH.òbÛ. ³y.š.=÷6œ&.
00000128	8556061A	8B638F2D	E6D4EB56	1FBE2844	858C59CA	908058E2	FB0AF2C2	675C49CA	…V <cæôëv.¾(d…œyê.€xâû.òâg\iê< th=""></cæôëv.¾(d…œyê.€xâû.òâg\iê<>
00000160	A5319E1A	ED9058EE	21DBD40C	F7245AA8	BA842A1E	69DC78C6	F1CAD26C	05613EE3	¥1ž.í.Xî!ÛÔ.÷\$Z¨°"*.iÜxÆñÊÒl.a>ã
00000192	3C227BFD	045A0343	784DF9B5	B3CA7F9A	3EF9895E	3EA84570	272067B0	822529E7	<"{ý.Z.CxMùµ³Ê.š>ù‰^>¨Ep' g°,%)ç
00000224	D328B513	97D03B26	26AF84D4	7C6F321F	40C6B1B0	8FE8B992	CC040D09	90D9B6CD	Ó(µ.—Ð;&&¯"Ô∣o2.@Ʊ°.è¹'ÌÙ¶Í
00000256	1DD4BBE5	EFF91523	4DE3CF6E	7ABD762D	298E5FA6				.Ô≫åïù.#MãÏnz¾v-)Ž_¦

Fig. 13 Key entry

4) Continue by filling in the rest of the values starting with Prime Number 1 (use little endian from here on) to create the key structure as shown in Table 3 and Fig. 14.

Name	Size	Variable name in RSA equation	Color-code
Heading	16 bytes (32 digits)	N/A	Red
Public Exponent	4 bytes (8 digits)	e	Yellow
Modulus	256 bytes (512 digits)	n	Blue
Prime Number 1	128 bytes (256 digits)	р	Green
Prime Number 2	128 bytes (256 digits)	q	Orange
CRT Exponent 1	128 bytes (256 digits)	dP	Purple
CRT Exponent 2	128 bytes (256 digits)	dQ	Light Blue
CRT Exponent Coefficient	128 bytes (256 digits)	qInv	Black
Private Exponent	256 bytes (512 digits)	d	Pink

Table 3	Key generation status

						00000	0000.dky		
0	07020000	00A40000	52534132	00080000	01000100	576DAFDB	B506E706	63CB58EA	§ RSA2 WmØ€µÁcÀXÍ
32	340B1D4E	EEC9A099	6C47BD76	50D269AC	A19A4246	846A1AC4	E46B352F	95E5FB1C	4 NÓ…†ôlGΩ∨P"i¨°öBFÑj <i>f</i> ‰k5/ï°
64	F4A4CA58	1E5331B0	ØA9AC548	D6DCEB87	32B3026F	997C9C76	7CFC20C7	43CE6205	Ù§ X S1∞ ö≈H÷<Îá2≥ oôlúvl, «CŒb
96	8DA1B7B2	88DØCFD8	3DE84DC4	7D0971F3	8C603BD9	40403464	B6184496	47020BEE	ç°∑≤à-œÿ=ËMf} qÛå`;Ÿ@@4d∂ DñG Ó
128	9FF06C82	0005208C	94206FC2	12465588	2F5BBEB4	68A5F7B5	7B075F97	2A39EEA2	ü∉lÇ åî o¬ FUà/[œ¥h•~µ{ _ó*9Ó¢
160	E62A1AA6	88DDA5C3	D07F9566	AA9AE025	5A9882B9	2B78D21C	57D04990	629D59C4	Ê* ¶à>•√- ïf™ö‡%ZòÇπ+x" W-IêbùYf
192	EF5263E1	059AE998	41D27225	9643ED91	41A4A8DD	895F7C26	7F201A8D	E9C79D64	ÔRc• öÈòA"r%ñCÌëA§®>â_ & çÈ«ùd
224	B52A4F8A	440E3FC2	FØA87E56	5CCADA29	9F92B4AD	2CFB74D4	2C59B76B	CØ335F7E	μ*OäD ?¬ « ®~V\ /)üí¥≠, t',Y∑k¿3_~
256	AB3628D9	54F3400B	4A738D5F	4DAFD468	D89B76BA	695B9755	2A768982	53F30D31	´6(ŸTÛ@ Jsç_MØ'hÿõv∫i[óU*vâÇSÛ 1
288	67683627	8E606A57	7460FA6B	56AF7FD5	9DB136C1	ØA66E518	40EAEAE5	D8F4A7B2	gh6'é`jWt``kVØ 'ù±6; f @ÍÍÂÿÙß≤
320	F6662337	3BB59EAE	E279A083	5C270FDA	8E62DC18	FF81F727	B03E875B	E30759FB	^f#7;µûÆ,y†É∖' ∕éb< čÅ~'∞>á[" Y'
352	E2CEB1D0	ØBB83214	A87F20C6	2C87BFAC	5F8DFC20	BCAB5853	ØBBØABEB	A3487AD0	,Œ±- ∏2 ® Δ,άø"_ç, °´XS ∞´Î£Hz-
384	5B3E8258	BDA46204	ØF9BBE6C	6B13A22E	59C888F2	BF2A4BEB	B4792312	74CB9895	[>ÇXΩ§b õœlk ¢.Y»àÚø*KÎ¥y# tÀòï
416	426091C8	0069E2EA	1FB1C18A	8A2F0953	A7D42EAB	4324852A	64AC26DA	6381C262	B`ë» i,Í ±¡ää/ SB'.´C\$Ö*d¨&/cŬb
448	A80ACB59	36F25133	B7038E72	278B30A1	B1AE3461	2BE19B13	B5D5F35F	05C487FA	® ÀY6ÚQ3∑ ér'ã0°±Æ4a+∙õ µ'Û_ fá'
480	952BEAA2	6A17EE84	EA8C2DE6	AD3ACE43	C9C6EC16	496B8206	C40F3E34	780E5558	ï+Í¢j ÓÑÍå-Ê≠:ŒC…ΔΪ IkÇ f >4x UX
512	86437659	64A86441	01397042	640E7DDF	ØCDFDØC4	09F3099A	C9CE1C68	35ED7DC3	ÜCvYd®dA 9pBd }fl fl-f Û ö…Œ h5Ì}√
544	Ø86E1291	CDFDCBA4	8F9ED1AE	0F42842F	693E395A	9098F397	4C130D24	0E54A49F	n ëÕ″À§èû–Æ BÑ/i>9ZêòÛóL \$ T§ü
576	1326720D	2DC3A623	6152EA8E	2EAF82F1	D66021EA	96D5C237	851A03E3	9510F032	&r -√¶#aRÍé.ØÇÒ÷`!Íñ'¬7Ö "ï €2
608	C04471EF	9674083C	BDØ2E9CC	9EF94CEF	D1D088F8	BC932B4A	94C915BA	955BA400	¿DqÔñt <Ω ÈÃû LÔà⁻°ì+Jî… ∫ï[§
640	E236325C	30B12A16	4984C1D8	07CCDFAF	19E1F04D	652B7A35	E27262BD	9EE3892C	,62∖0±* IÑįÿ ĀflØ ∙∎Me+z5,rbΩû"â,
672	04D8886B	E3EFDF12	C7C9422C	A438532A	7DF938BF	0D260AAE	AA6BB32C	370760DD	ÿàk"Ôfl «…B,§8S*}`8ø & Æ™k≥,7 `>
704	4C62DDC4	427033DF	D5AA7966	F67C13F0	BFA2B8E5	63459E22	A8937BE3	5D9E45F8	Lb>fBp3fl'™yf^∣ ∉ø¢∏ÂcEû"®ì{"]ûE⁻
736	20E20F54	A212D112	7BFC2646	93684C02	2458ED97	ED7AEBFA	C018F3D9	468CFEBD	, T¢ - { &FìhL \$XÌóÌzÎ ¿ ÛŸFå,Ω
768	4A211196	8652A57F	804DFCC4	3AB43756	393FDF1C	ØADD2423	2EC9EØBE	ED04B769	J! ñÜR● ÄM,f:¥7V9?fl >\$#‡œÌ ∑i
800	95580A1D	438723C3	3D4BB052	69049A5F	B8596B8E	7F180BA2	FD42DF22	884E7E2E	ïX Cá#√=K∞Ri ö_∏Yké ¢″Bfl"àN~.
832	7DD0406A	49F32A7F	170E3EDD	DE99A181	784C5730	72E77676	659E75F0	54977AC9	}-@jIÛ* >>fiô°ÅxLW0rÁvveûu « Tóz…
864	BE89DFF0	1CA46252	4FA4C128	10EAB3E1	6CDF935E	3DA49276	1F404B42	712713E8	œâfl e §bRO§¡(Í≥·lflì^=§ív @KBq' Ë
896	E299B3C7	C666526A	FC99B2F6	427E3E19	AF767E1B	413A2CC6	9050B71F	21EE4469	,ô≥«∆fRj ô≤^B~> Øv~ A:,∆êP∑ !ÓDi
928	397DD4FB	6F09AA4D	F0409547	457D7D04	EEAE26A6	13942AEA	883D5100	C76AD43D	9}''o ™M∉@ïGE}} ÓÆ&¶ î*Íà=Q «j'=
960	3DDDBØBC	6EA835FE	EA05F1E6	CAD6EA94	825DE489	B045B0C0	A65B1882	8D331409	=>∞°n®5,Í ÒÊ ÷ÍîÇ]‰â∞E∞¿¶[Çç3
992	B1E17081	B55437FB	1BFDE112	955DDF5E	1642B730	61C1B57F	ED6CF669	7300FC45	±·pÅμT7° ´· ï]fl^ BΣ0a;μ Ìl^is .E
1024	87FFB30D	ØDF316EC	17EC9E8C	40B1C161	4308A4F2	49EA6A6F	C6A6398C	FØA3E76A	á`≥ Û Ï Ïûå@±¡aC §ÚIÍjo∆¶9å∉£Áj
1056	2AAC7423	1B2EFD4A	F7E41542	958A10E8	57F06006	A0DDAA2E	5CF39B52	ØF373575	*"t# .″J~‰ Bïä ËW ♠` †>™.\ÛõR 75u
1088	C287AC87	9EC7EA45	41F93A1A	CD71D6A4	41EFFD7A	D8F70F1D	1969F075	E7054A2F	¬á¨áû«ÍEAĭ: Õq÷§AÔ″zÿ″ i∉uÁ J∕
1120	47357EA4	A2B6DAA9	FB4ADC0B	909B5177	D301376A	6249B931	F37A2B92	4ADA3680	G5~§¢∂/©'J< êõQw" 7jbIπ1Ûz+íJ/6Ä
1152	BB173B8C	637AD3E7	195CØEAB	67EØC5B5	8A282852				° ;åcz"Á ∖ ´g‡≈µä((R
Signe	ed Int 🔍	little 🗘 (s	elect some o	lata)					-+
						0 out of 1	172 bytes		

Fig. 14 Key structure

- 5) Once done, save the key to your desktop.
- 6) Move the key you just created (0000000.dky) to the shared folder.

5.5 Activity 5: Decrypting Files Using the Regenerated Private Key

You created your own private key. Now it is time to test it on the infected machine.

- 1) Exit the current Windows machine by clicking the Back button in your browser. Click again on the link corresponding with the Ransomed machine.
- 2) Move the private key (0000000.dky) from the shared folder to the desktop.
- 3) Click **Decrypt** on the message prompting you to pay (Fig. 15).



Fig. 15 Decrypt button

- 4) Click Start.
- 5) As a test to make sure it worked, open **Important.txt** to make sure you can read the data.

Great job! You have successfully analyzed a binary and decrypted your ransomed files.

6. Conclusion

After completing this exercise, participants should have a better understanding of how ransomware works, how RSA encryption works, and the steps needed to develop a new decryption key. This exercise will be shared with collaborators and partners (including professionals, faculty, and students) to help establish a common ground for studying ransomware that is similar to WannaCry, analysis tools, and research in binary analysis, both to secure systems and to develop ways to recover after compromise.

More specifically, we envision this report being the first of many studies to uncover the inner workings of ransomware. We hope the information found herein will enlighten researchers and practitioners in the cybersecurity field to understand ransomware and develop detection and quarantine mechanisms rather than using simple signature-based techniques. In addition, this report demonstrates a way to recover data from devices that have fallen victim to WannaCry. While the wanakiwi code is currently not capable of decrypting files on recent systems, this report details the steps associated with analyzing memory to recover the private key. Still left to future work is whether these or similar techniques can be used to recover keys on other systems to include Linux, Mac, and newer versions of Windows.

7. References

- 1. Trautman LJ, Ormerod PC. WannaCry, ransomware, and the emerging threat to corporations. Tenn L Rev. 2018;86:503.
- 2. Rivest RL, Shamir A, Adleman LM, inventors. Cryptographic communications system and method. United States patent US 4,405,829. 1983.
- 3. Jonsson J, Kaliski, B. Public-key cryptography standards (PKCS) #1: RSA cryptography specification version 2.1. 2003 Feb [accessed 2020 Apr 6]. https://tools.ietf.org/html/rfc3447#appendix-A.1.2.
- Acosta JC, Escobar de la Torre A, Salamah S. Hands-on cybersecurity studies: multi-perspective analysis of the WannaCry ransomware. Aberdeen Proving Ground (MD); Army Research Laboratory (US); 2019 Jan. Report No.: ARL-TR-8627.
- 5. Delpy B. Wanakiwi [code]. 2017 May [accessed 2020 Apr 6]. https://github.com/gentilkiwi/wanakiwi/releases
- Mobilefish. Online RSA key generation. 2019 Dec [accessed 2020 Apr 6]. https://www.mobilefish.com/services/rsa_key_generation/rsa_key_generation.php

List of Symbols, Abbreviations, and Acronyms

ARL	Army Research Laboratory
CCDC	US Army Combat Capabilities Development Command
CIT	Collaborative Innovation Testbed
CyberRIG	Cyber Rapid Innovation Group
RSA	Rivest, Shamir, and Adleman

1	DEFENSE TECHNICAL
(PDF)	INFORMATION CTR
	DTIC OCA

CCDC ARL 1

(PDF)	FCDD RLD CL
	TECH LIB

- 2 CCDC ARL
- (PDF) RDRL CIN D J CLARKE
 - J ACOSTA