DMA Locker: New Ransomware, But No Reason To Panic

blog.malwarebytes.com/threat-analysis/2016/02/dma-locker-a-new-ransomware-but-no-reason-to-panic/

hasherezade

February 3, 2016

All your personal files are LOCKED!

WHAT'S HAPPENED?

- * All your important files(including hard disks, network disks, flash, USB) are encrypted.
- * All of files are locked with asymetric algorithm using AES-256 and then RSA-2048 cipher.
- * You are not possible to unlock your files because all your backups are removed.
- * Only way to unlock your files is to pay us 536 GBP in Bitcoin currency (2.0 BTC). After payment we will send you decryption key automatically, which allow you to unlock files .

DMA Locker is another <u>ransomware</u> that appeared at the beginning of this year. For now it has been observed to be active only on a small scale (<u>source</u>) – but we just want to warn you that it exists.

[UPDATE] READ ABOUT THE LATEST VERSION OF DMA LOCKER: 4.0

UPDATE [4 Feb 2016]: I apologize to everyone misguided by my rush conclusions about the crypto. After further analysis and consultation with other analysts (special thanks to <u>@fwosar</u> and <u>@maciekkotowicz</u>) I confirmed that in reality it is AES in ECB mode. Low entropy was just caused by the fact, that it encrypts separately 16 byte chunks, that are small enough to give this effect. Authors of the malware told many lies in their ransom note, but this one was true, just my mistake. The only way to recover the key is to find the original sample with key included. My goal is always to provide best quality analysis – this time I failed, but I tried to fix it as soon as possible and not let the false information spreading.

Analyzed samples

- <u>d35344b1f48764ba083e51438121e6a9</u> Polish version type 2 (from Jan 2016) <main focus of this analysis
- <u>4190df2af81ece296c465e245fc0caea</u> English version type 2 (from Jan 2016)
- <u>6fbd3cdcafd6695c384a1119873786aa</u> Polish version type 1 (from Dec 2015)

// Special thanks to malware hunters: <u>@PhysicalDrive0</u>, <u>@JAMESWT_MHT</u> and <u>@siri_urz</u> for their respective help in collecting the samples!

Behavioral analysis

When deployed, the ransomware moves itself into C:\ProgramData (or C:\Documents and Settings\All Users\Dokumenty\), renamed to fakturax.exe and drops another, modified copy: ntserver.exe. File faktura.exe is removed after execution. Depending on its version, it may also drop some other files in the same location.

Local Disk (C:) 🕨 ProgramData 🕨		Search ProgramD	ata
New folder			
Name	Date modified	Туре	Size
🐴 Templates	2009-07-14 06:53	File folder	
Cryptinfo.txt	2016-02-02 15:02	Text Document	1 KB
date_1.txt	2016-02-02 15:02	Text Document	1 KB
fakturax.exe	2016-01-28 15:16	Application	96 KB
ntserver.exe	2016-02-02 14:59	Application	96 KB

Symptoms of this ransomware can be recognized by a red window popping up on the screen. So far, it has been observed in two language versions – Polish or English. An example of the English is below:

DMA Locker	
	All your personal files are LOCKED!
	WHAT'S HAPPENED? * All your important files(including hard disks, network disks, flash, USB) are encrypted. * All of files are locked with asymetric algorithm using AES-256 and then RSA-2048 cipher. * You are not possible to unlock your files because all your backups are removed. * Only way to unlock your files is to pay us 536 GBP in Bitcoin currency (2.0 BTC). After payment we will send you decryption key automatically, which allow you to unlock files .
	HOW TO PAY US AND UNLOCK YOUR FILES? 1. To pay us, you have to use Bitcoin currency. You can easily buy Bitcoins at following sites: * https://www.coinfloor.co.uk * https://www.coinfloor.co.uk * https://www.coinfloor.co.uk * https://www.coinfloor.co.uk * https://www.coinfloor.co.uk * https://www.coinfloor.co.uk * https://www.bitstamp.net/ 2. If you already have Bitcoins, pay us 2.0 BTC (536 GBP) on following Bitcoin address:
	3. After payment, necessarily contact with us to get your decryption key: january0030@gmx.com . In mail title write your unique ID:
 You have 96 hours to pay us! After this time all your files will be lost! Your decryption key will destroy on: 	4. We will automatically send you decryption key after bitcoin transfer . When you receive your decryption key, copy and paste it to "DECRYPTION KEY" field Then, press the DECRYPT button to UNLOCK ALL YOUR FILES.
3/2/2016 1:57	IF FILES UNLOCKING PROCEDURE IS ALREADY WORKING, YOU CAN EASILY TURN OFF YOUR COMPUTER AND CONTINUE FILES UNLOCKING AFTER NEXT STARTUP. TO CONTINUE HEALING YOUR FILES, COPY AND PASTE THE SAME DECRYPTION KEY TO THE "DECRYPTION KEY" FIELD AND PRESS "DECRYPT" BUTTON. THE FILES RECOVERING WILL BE CONTINUED!
	DECRYPTION KEY: DECRYPT

Earlier version comes with a bit different GUI (also Polish or English variant):



In contrast to other ransomware that are offering a separate decrypter, **DMA Locker** comes with a decrypting feature built-in. It is available from the GUI with ransom note. If the user enters a key (32 characters long) in the text field and clicks the button, the program switches to the decryption mode (using supplied key):



The program is not very stable and may crash during encryption. An older version has been observed to sometimes crash after finishing encryption – but before displaying any info about what happened, which may be very confusing for the victim. What makes things worse is the fact that it does not change file extensions. So, in such a case the only visible symptom will be that the attacked person cannot open some of his/her files.

Newer versions also add keys to the autorun. One is to deploy a dropped copy of the program, and the other to display a ransom note in TXT format (via notepad). However, the copy of the program (DMALOCK 41:55:16:13:51:76:67:99*ntserver.exe*) – is not always dropped successfully and then only the TXT note may be displayed.



Detection

It is detected by Malwarebytes Anti-Malware as Ransom.DMALocker:



Experiment

In the ransom note, the authors mention that the data is encrypted by **AES and RSA**. Let's look at the files.

After the first look at encrypted content we can see repetitive patterns and entropy is relatively low.

Left – raw bytes of original BMP, right – the same BMP encrypted by DMA Locker:



Let's compare some more files and see how they changed after being encrypted by DMA Locker.

Example 1 – HTML files:

comparison of original files:

C:\Users	\test	er\D	eskt	op\raw_samples\index.html								>>	C:\Users	\test	er\D	eskt	op\r	amples\modules.html										
С	ompa	are			Next difference Previous							us dif	ference			Fo	nt		🗎 🏥 🔽 Binary									
Ec	dit ma	de				С	opy ·	>			(Сору	<-			Un	do					ANSI<->ANSI						
00000:	3C	21	44	4F	43	54	59	50)</th <th>DOCT</th> <th>YP</th> <th>] 🔺</th> <th>00000:</th> <th>3C</th> <th>21</th> <th>44</th> <th>4F</th> <th>43</th> <th>54</th> <th>59</th> <th>50</th> <th><pre> <!--DOCTYP</pre--></pre></th> <th></th>	DOCT	YP] 🔺	00000:	3C	21	44	4F	43	54	59	50	<pre> <!--DOCTYP</pre--></pre>						
00008:	45	20	48	54	4D	4C	20	50	E]	HTMI	P		00008:	45	20	48	54	4D	4C	20	50	E HTML P						
00010:	55	42	4C	49	43	20	22	2D	UB:	LIC	"-	_	00010:	55	42	4C	49	43	20	22	2D	UBLIC "-						
00018:	2F	2F	57	33	43	2F	2F	44	177	ИЗС/	'/D		00018:	2F	2F	57	33	43	2F	2F	44	//W3C//D						
00020:	54	44	20	48	54	4D	4C	20	TD	HTM	ſL		00020:	54	44	20	48	54	4D	4C	20	TD HTML						
00028:	34	2E	30	31	20	54	72	61	4.	01 I	ra		00028:	34	2E	30	31	20	54	72	61	4.01 Tra						
00030:	6E	73	69	74	69	6F	6E	61	ns:	itic	na		00030:	6E	73	69	74	69	6F	6E	61	nsitiona						
00038:	6C	2F	2F	45	4E	22	ЗE	0A	11/	/EN"	'>.		00038:	6C	2F	2 F	45	4E	22	3E	0A	1//EN">.						
00040:	3C	68	74	6D	6C	ЗE	3C	68	<h< td=""><td>tml></td><td>∽h</td><td></td><td>00040:</td><td>3C</td><td>68</td><td>74</td><td>6D</td><td>6C</td><td>ЗE</td><td>3C</td><td>68</td><td> <html≻<h< td=""><td></td></html≻<h<></td></h<>	tml>	∽h		00040:	3C	68	74	6D	6C	ЗE	3C	68	<html≻<h< td=""><td></td></html≻<h<>						
00048:	65	61	64	ЗE	3C	6D	65	74	ea	d≻ <n< td=""><td>net</td><td></td><td>00048:</td><td>65</td><td>61</td><td>64</td><td>ЗE</td><td>3C</td><td>6D</td><td>65</td><td>74</td><td> ead><met< td=""><td></td></met<></td></n<>	net		00048:	65	61	64	ЗE	3C	6D	65	74	ead> <met< td=""><td></td></met<>						
00050:	61	20	68	74	74	70	2D	65	a]	http)-е		00050:	61	20	68	74	74	70	2D	65	a http-e						
00058:	71	75	69	76	ЗD	22	43	6F	qu	iv="	'Co		00058:	71	75	69	76	ЗD	22	43	6F	quiv="Co						
00060:	6E	74	65	6E	74	2D	54	79	nt	ent-	ту		00060:	6E	74	65	6E	74	2D	54	79	ntent-Ty						
00068:	70	65	22	20	63	6F	6E	74	lpe	" co	nt		00068:	70	65	22	20	63	6F	6E	74	pe" cont						
00070:	65	6E	74	ЗD	22	74	65	78	en	t="t	ex		00070:	65	6E	74	ЗD	22	74	65	78	ent="tex						
00078:	74	2F	68	74	6D	6C	ЗB	63	[t/]	html	;c		00078:	74	2F	68	74	6D	6C	ЗB	63	<pre>lt/html;c</pre>						
00080:	68	61	72	73	65	74	ЗD	69	ha	rset	;=i		00080:	68	61	72	73	65	74	ЗD	69	harset=i						
00088:	73	6F	2D	38	38	35	39	2D	so	-885	i9-		00088:	73	6F	2D	38	38	35	39	2D	so-8859-						
00090:	31	22	3E	0A	3C	74	69	74	1"	>. <t< td=""><td>it</td><td></td><td>00090:</td><td>31</td><td>22</td><td>3E</td><td>0A</td><td>3C</td><td>74</td><td>69</td><td>74</td><td> 1"≻.<tit< td=""><td></td></tit<></td></t<>	it		00090:	31	22	3E	0A	3C	74	69	74	1"≻. <tit< td=""><td></td></tit<>						
00098:	6C	65	ЗE	50	69	6E	ЗA	20	le	>Pin	11		00098:	6C	65	3E	50	69	6E	ЗA	20	le>Pin:						
000A0:	50	69	6E	20	32	2E	31	33	Pi	n 2.	13		000A0:	4D	6F	64	75	6C	65	20	49	Module I						
000A8:	20	55	73	65	72	20	47	75	ΙU	ser	Gu		000A8:	6E	64	65	78	зC	2 F	74	69	ndex <td></td>						
000B0:	69	64	65	зc	2 F	74	69	74	lid	e≺/t	it		000B0:	74	6C	65	ЗE	0A	зc	6C	69	tle>. <li< td=""><td></td></li<>						
000B8:	6C	65	ЗE	0A	зc	6C	69	6E	le	>.<1	in		000B8:	6E	6B	20	68	72	65	66	ЗD	nk href=						
000C0:	6B	20	68	72	65	66	ЗD	22	k	href	="		00000:	22	64	6F	78	79	67	65	6E	doxygen						
000C8:	64	6F	78	79	67	65	6E	2E	do:	xyge	en.		000C8:	2E	63	73	73	22	20	72	65	.css" re						

comparison of the same files encrypted:

C:\Users\tester\Desktop\encrypted_samples\index.html Compare Next difference Previo Edit mode Copy -> (00000: 04 90 4E E3 E7 38 A3 30 .Năç82.0 00008: BF DE 4A 71 3B 7E 4A EB żŢJq;~JĔ 00010: 0E E8 AE 91 44 44 34 76 .č@`DD4v 00018: 28 61 C5 A6 A6 E5 D1 53 (aĹ; ;ĺŃS 00020: E3 F9 85 08 A6 EC FE 55 ăů; ěţU 00028: B2 FE 12 71 56 7A 32 1A ,ţ.qV22. 00030: 38 16 4C 69 16 0F D7 1A 8.Li*. 00038: D4 D7 48 E3 0D 6C 8B 70 Ô×Hă.l 00048: 31 08 D5 2E 51 02 D2 B4 1.Õ.Q.Ň' 00058: E7 66 3E C7 69 1C 9D CF ‡f>Ç1.ťĎ 00058: CE AC EC 58 D2 2E 2C 77 άěXŇ.,w 00060: D0 B9 90 1B 71 B3 E4 29 Đạ.q2ă) 00068: BB 3A 36 38 D5 B1 99 90 »:68Ô± ^{m4} 00070: 16 EA AA 44 F3 EC BB 37 .e§D6± ^{m4} 00078: 17 D5 7B 59 0D F1 51 14 .Õ{Y.ŕQ. 00080: 15 7C 31 16 E3 21 1A F7 . 1.ǎ!.+ 00081: 56 4C 21 BD 7C E8 74 DD vL!" čtÝ		t.html >> C:\Users\	tester\Desktop\encryp	ted_samples\modules.html
Compare	Next difference	Previous difference	Font	🕅 🏨 🗸 Binary
Edit mode	Copy ->	Copy <-	Undo	ANSI<->ANSI
00000: 04 90 4E E	3 E7 38 A3 30 .Nă	<u>ç8Ł0 🔺 00000:</u>	04 90 4E E3 E7	38 A3 30 .Năç8Ł0
00008: BF DE 4A 7	1 3B 7E 4A EB źŢJ	Iq;∼Jë 🗌 00008:	BF DE 4A 71 3B	7E 4A EB żŢJq;∼Jë
00010: 0E E8 AE 9	91 44 44 34 76 .č@	DD4v 00010:	OE E8 AE 91 44	44 34 76 .č®`DD4∨
00018: 28 61 C5 A	A6 A6 E5 D1 53 (aÍ	.;;íŃS 00018:	28 61 C5 A6 A6	E5 D1 53 (aŦ¦ĺŃS
00020: E3 F9 85 0)8 A6 EC FE 55 ăů	¦ěţU 00020:	E3 F9 85 08 A6	EC FE 55 ăů¦ěţU
00028: B2 FE 12 7	1 56 7A 32 1A ţ.	qVz2. 00028:	B2 FE 12 71 56	7A 32 1A ˌţ.qVz2.
00030: 38 16 4C 6	59 16 OF D7 1A 8.1	i×. 00030:	38 16 4C 69 16	OF D7 1A 8.Li×.
00038: D4 D7 48 E	13 OD 6C 8B 70 Ö×H	lă.l <p 00038:<="" td=""><td>D4 D7 48 E3 0D</td><td>6C 8B 70 Ö×Hă.l<p< td=""></p<></td></p>	D4 D7 48 E3 0D	6C 8B 70 Ö×Hă.l <p< td=""></p<>
00040: 83 A4 8C 2	6 AO 82 A9 39 ¤Ša	,©9 00040:	83 A4 8C 26 A0	82 A9 39 ¤Š& ,©9
00048: 31 08 D5 2	E 51 02 D2 B4 1.Ö	00048:	31 08 D5 2E 51	02 D2 B4 1.Ö.Q.N
00050: 87 66 3E C	7 69 1C 9D CF ‡f>	Çi.tD 00050:	87 66 3E C7 69	1C 9D CF ‡f≻Çi.ťĎ
00058: CE AC EC 5	58 D2 2E 2C 77 άĕ	XŇ.,w 00058:	CE AC EC 58 D2	2E 2C 77 άĕXŇ.,₩
00060: D0 B9 90 1	.B 71 B3 E4 29 Đą.	qłä) 00060:	D0 B9 90 1B 71	B3 E4 29 Đą.qłä)
00068: BB 3A 36 3	88 D5 B1 99 90 »:6	58Ő±™ 00068:	BB 3A 36 38 D5 1	B1 99 90 ∣»:68Ő±™
00070: 16 EA AA 4	4 F3 EC BB 37 .ęş	Dóě»7 00070:	16 EA AA 44 F3	EC BB 37 .ęŞDóě»7
00078: 17 D5 7B 5	69 OD F1 51 14 .Ő{	Y.ńQ. 00078:	17 D5 7B 59 0D	F1 51 14 .Ő{Y.ńQ.
00080: 15 7C 31 1	.6 E3 21 1A F7 . 1	ă!.÷ 00080:	15 7C 31 16 E3	21 1A F7 . 1.ă!.÷
00088: 76 4C 21 B	3D 7C E8 74 DD vL!	" čtÝ 00088:	76 4C 21 BD 7C	E8 74 DD vL!" čtÝ
00090: 09 F9 E0 9	94 27 03 27 F8 .ůź	"'.'ž 00090:	09 F9 E0 94 27	03 27 F8 .ůŕ″'.'ř
00098: 1D 80 DD 4	F 86 71 CC 6D .€Ϋ	00098:	1D 80 DD 4F 86	71 CC 6D .€ÝO†qĚm
000A0: 61 DC BF 6	A 23 AE D5 B7 aÜż	;j ‡®Ö ∙ 000A0:	8B CB D3 11 51	37 75 3D ≮ËÓ.Q7u≕
000A8: 77 E8 AB E	:3 6F 4C 3B 13 ∣wč⊗	(ăoL;. 000A8:	41 OC E2 3B 26	2E 43 8D A.â;&.CŤ
000B0: 8B 70 83 A	LD 72 8F C6 72 <p-< td=""><td>rŹĆr 000B0:</td><td>A4 94 99 55 92 3</td><td>3D AF 68 ¤‴™U'=Żh</td></p-<>	rŹĆr 000B0:	A4 94 99 55 92 3	3D AF 68 ¤‴™U'=Żh
000B8: 49 03 6E 3	6 A1 49 D0 EF I.m	16*IÐď 000B8:	C5 E6 53 67 1F	23 01 37 ĹćSg.‡.7
000C0: 66 C1 EB E	18 F5 36 E8 4F fÁë	čő6č0 000C0:	OF 94 1B OF CE	7F 8E 45 .″ÎOŽE
000C8: D0 A6 5B 6	6 55 C4 D6 6B Ц[fUÄÖk 000C8:	D9 8A A5 DF 1F	27 E2 A2 ŬŠĄß.'â~

Example 2 – PNG files:

comparison of original files:

C:\\	Jsers	\tes	ter\l	Desk	top'	vaw	sam	ples	\stac	k-debugger-c	ommands.png	:	»>	C:\Use	rs\te	ster\	Desk	top\	raw_	_sam	ples	\doxy	/gen.png
	С	òmp	are				Next	t diff	erend	e Pre	vious difference			Font			A	A		Bina	ary		Case sei
	E	dit m	ode				(Сору	/->		Copy <-			Undo						AN	SI<⇒	ANS	🖳 🔽 Ignore fr
000	0:	89	50	4E	47	0D	0A	1A	0A	%PNG			*	0000:	89	50	4E	47	0D	0A	1A	0A	*PNG
000	8:	00	00	00	0D	49	48	44	52	IHDR				0008:	00	00	00	0D	49	48	44	52	IHDR
001	.0:	00	00	01	EB	00	00	00	C0	Ŕ			_	0010:	00	00	00	64	00	00	00	2D	d
001	8:	08	02	00	00	00	AB	8D	DC	׍Ü				0018:	08	03	00	00	00	6F	6B	3E	ok>
002	0:	B7	00	00	00	01	73	52	47	· sRG				0020:	C2	00	00	00	04	67	41	4D	ÂgAM
002	8:	42	00	AE	CE	1C	Ε9	00	00	B.®Î.é				0028:	41	00	00	D6	D8	D4	4 F	58	AÖŘÔOX
003	0:	00	04	67	41	4D	41	00	00	gAMA				0030:	32	00	00	00	19	74	45	58	2tEX
003	8:	B1	8F	0B	FC	61	05	00	00	±Ź.üa				0038:	74	53	6F	66	74	77	61	72	tSoftwar
004	0:	00	09	70	48	59	73	00	00	pHYs				0040:	65	00	41	64	6F	62	65	20	e.Adobe
004	8:	12	74	00	00	12	74	01	DE	.tt.Ţ				0048:	49	6D	61	67	65	52	65	61	ImageRea
005	0 :	66	1F	78	00	00	1B	3B	49	f.x;I				0050:	64	79	71	C9	65	зc	00	00	dyqÉe<
005	8:	44	41	54	78	5E	ED	9D	6D	DATx^itm				0058:	00	5D	50	4C	54	45	00	00	.]PLTE

comparison of the same files encrypted:

C:\Use	C:\Users\tester\Desktop\encrypted_samples\stack-debubger-commands.png										>>	C:\User	s∖tes	ster∖	Desk	top\	enc	rypte	ed_sa	ample	s\doxyge	en.png		
	Com	pare				Next	t diffe	erend	e	Prev	vious difference			Font			A	Ä		Bina	ary			Case sen Ignore rej
	Edit n	node				(Сору	/ ->			Copy <-			Undo						AN	SI<->	ANS	i 🗸	Ignore fre
0000:	42	4A	A 5	10	Α6	B2	39	06	BJĄ	.:.9.				0000:	42	4A	A5	10	A6	B2	39	06	BJĄ.;	<mark>,</mark> ۹.
0008:	A2	зc	17	7B	В4	C8	DE	EF	*<	{´ČŢď				0008:	A2	зc	17	7B	В4	C8	DE	EF	1*<.{*	ČŢď
0010:	C4	90	в8	7 F	9E	99	D8	A 8	Ä,Ūž	™Ř			_	0010:	BF	EF	CA	95	04	13	89	A 8	żdĘ•.	.¥"
0018:	ED	41	DA	BC	8C	40	25	60	íAÚ	CŚ@ზ`				0018:	22	5F	0B	4E	24	BB	66	26	"N\$	»f&
0020:	8B	45	AC	AE	05	ВD	7 F	9C	< E⇒0	0.″lś				0020:	62	18	E8	B2	AЗ	AF	EΒ	CA	b.č,Ł	ŻëĘ
0028:	9B	19	89	9E	19	0A	03	DE	1>.8	žŢ				0028:	43	ЗB	C7	C0	D6	31	8E	C0	C;ÇŔÖ	1ŽŔ
0030:	00	62	53	00	62	C3	F5	2D	.bS	.bĂő-				0030:	F5	Α4	46	4B	41	7B	94	5E	ő¤FKA	{ " ^
0038:	4A	cc	9B	52	ЗF	42	8B	70	JĚ ≻	R?Bkp				0038:	B2	C8	19	C7	5A	2 F	95	82	T,Č.ÇZ	1.
0040:	6B	75	Ε7	46	ΕO	F6	C9	84	kuçl	FŕöÉ"				0040:	CB	0F	D5	1B	13	В9	ЗD	D2	Ë.Ö	ą=Ň
0048:	5C	D8	DB	29	A2	BA	E5	61	\ŘŰ))ĭşĺa				0048:	A2	Α9	4D	CE	4A	61	88	СВ	∣~©MÎJ	laË
0050:	AA	EΒ	46	в0	45	43	22	9F	ļŞēF	°EC"ź				0050:	82	C1	89	99	F2	F5	ED	1B	,Á≿™ň	őí.
0058:	35	60	EE	10	4E	ED	F1	98	5°î	.Níń				0058:	4C	EΕ	34	50	4E	2 F	C3	11	Lî4PN	/Ă.

As we can see, when the beginnings of original files are identical, the beginnings of encrypted outputs also are. But it seems that encryption is done in some chunks – possibly 8 or 16 bytes at once. Look at the comparison of PNG files – from 0x10 they have been encrypted differently – although they both have zeros at positions 0x10, 0x11...

Inside

This ransomware is distributed without any packing and no defense against analysis has been observed. All the used strings and called API functions are in plain text. In fact, the malware even "helps" the analyst by providing a lot of debug strings describing all it's activities (original + translation):

[+] Plik jest aktualnie zaszyfrowany, pomijanie.. //The file is already encrypted, skipping.. [*] Rozmiar pliku = %I64d bajtow.. //File size = %I64d bytes.. [+] Rozpoczeto szyfrowanie pliku: %s //Started encrypting the file: %s [+] Zakonczono szyfrowanie pliku: %s //Finished encrypting the file: %s [+] Rozpoczeto zapisywanie z pamieci do pliku: %s //Started dumping from memory to a file: %s [+] Zakonczono zapisywanie z pamieci do pliku: %s //Finished dumping from memory to a file: %s [*] Plik jest aktualnie odszyfrowany, pomijanie.. //The file is already decrypted, skipping.. [+] Rozpoczeto deszyfrowanie pliku: %s //Started decrypting file: %s [+] Zakonczono deszyfrowanie pliku: %s //Finished decrypting file: %s Alokacja, error: %d //Allocation error: %d DMA Locker Otwieranie pliku: %d //Opening file: %d

Thanks to the logs, finding important part of the code is trivial!

At the beginning of the execution a new thread is deployed – whose role is to check for the presence of following processes:

- rstrui.exe
- ShadowExplorer.exe
- sesvc.exe
- cbengine.exe

If any of them is detected, malware tries to terminate it. Just after deploying this thread malware logs (in Polish):

"[+] Blocking processes of system recovery"

004043AD 004043B2 004043B7 004043B7 004043B9 004043B8 004043B8 004043B0 004043C1 004043C1	call call xor push push push push push	<pre>init_filenames drop_ransom_note_txt ebx, ebx ebx</pre>
004043C9 004043CB	test jz	eax, eax short skip_logging
	- 6 6	
004043CD push 004043D2 call	offse prin	t ablokowanleproc ; " [+] blokowanle procesow przywracanla s" tf

Instead of a list of attacked extensions, this malware contains two blacklists. One for directories:

004025A3	sub	esp, 2Ch
004025A6	push	esi
004025A7	mov	<pre>[ebp+var_2C], offset aWindows ; "\\Windows\\"</pre>
004025AE	mov	<pre>[ebp+var_28], offset aWindows_0 ; "\\WINDOWS\\"</pre>
004025B5	mov	<pre>[ebp+var_24], offset aProgramFiles ; "\\Program Files\\"</pre>
004025BC	mov	[ebp+var_20], offset aProgramFilesX8 ; "\\Program Files (x86)\\"
004025C3	mov	[ebp+var_1C], offset aGames ; "Games"
004025CA	mov	[ebp+var_18], offset aTemp ; "\\Temp"
004025D1	mov	[ebp+var_14], offset aSamplePictures ; "\\Sample Pictures"
004025D8	mov	[ebp+var_10], offset aSampleMusic ; "\\Sample Music"
004025DF	mov	[ebp+var C], offset aCache ; "\\cache"
004025E6	mov	[ebp+var 8], offset aCache 0 ; "\\Cache"
004025ED	xor	esi, esi

and another for file extensions:

00402627 mov	<pre>[ebp+var_30], offset a_exe ; ".exe"</pre>
0040262E mov	<pre>[ebp+var_2C], offset a_msi ; ".msi"</pre>
00402635 mov	<pre>[ebp+var_28], offset a_dl1 ; ".dl1"</pre>
0040263C mov	<pre>[ebp+var_24], offset a_pif ; ".pif"</pre>
00402643 mov	<pre>[ebp+var_20], offset a_scr ; ".scr"</pre>
0040264A mov	<pre>[ebp+var_1C], offset a_sys ; ".sys"</pre>
00402651 mov	<pre>[ebp+var_18], offset a_msp ; ".msp"</pre>
00402658 mov	<pre>[ebp+var_14], offset a_com ; ".com"</pre>
0040265F mov	<pre>[ebp+var_10], offset a_lnk ; ".lnk"</pre>
00402666 mov	<pre>[ebp+var_C], offset a_hta ; ".hta"</pre>
0040266D mov	<pre>[ebp+var_8], offset a_cpl ; ".cpl"</pre>
00402674 mov	<pre>[ebp+var_4], offset a msc ; ".msc"</pre>

Files that contain in their path blacklisted substrings are skipped.

Malware enumerates all the files – browsing first logical drives, after that network resources – trying to encrypt each and every file (except the blacklisted)

00404640	add	esp, 0Ch	
00404643	lea	edx, [esp+90h+key	_buf]
00404647	push	ebx	
00404648	push	edx	
08484649	call	enc_dec_logical_d	rives
0040464E	add	esp, 8	
00404651	push	ebx ;	int
00404652	lea	eax, [esp+94h+key	_buf]
00404656	push	eax ;	int
00404657	push	ebx ;	1pNetResource
00404658	call	enc_dec_net_resou	rces
0040465D	nov	ebx, ds:Sleep	
00404663	nov	esi, OAh	
0828748	imn	short loc 484678	

A single flag decides whether the malware is in encryption or decryption mode:

60440286 60440286 60440286 60440286 60440286 60440286 80040286	0 nov edx, edi 2 push edx ; char * 3 call _puts 8 add esp, 4 8 cmp [ebp+mode_flag], 0
80482BAF	F jnz short decrypt node
) 💶 🚅 🖼
00402BB1 push offset aFlaga0Szyfrowa ; "[+] Flaga = 0, s	szyfrowanie\n" 00402BCF
00402BB6 call _printf	00402BCF decrypt_mode: ; "[+] Flaga = 1, deszyfrowanie\n"
00402BBB mov eax, [ebp+cryptoKey]	00402BCF push offset aFlaga1Deszyfro
00402BC1 add esp, 4	00402BD4 call _printf
00402BC4 push eax ; int	00402BD9 mov edx, [ebp+cryptoKey]
004028C5 nov ecx, ed1 ; filename	BUHUZEDF add esp, 4
eckeepee sall eck (d) the second sall eckeepee	ees ; int
00402868 Call encrypt_file	eevense eest eest eest eest teelense eest teeleest teelense eest teest teelense eest teest teelense eest teelense eest teelense
00402860 Jub 20016 TOC 402602	ae4e2867 push edax ; Chdr *
	Behazer call uterspitting
	100402bcr Jup Shore 100_402602

Encryption (as well as decryption) is deployed in a new thread:



Encryption key

The encryption key is 32 byte long. In newer version of the malware it is hard-coded at the end of the original file, and then read. However, there is a twist.

During execution, two copies of the original file are dropped: **fakturax.exe** and **ntserver.exe** – but only **fakturax.exe** contains the key – **ntserver.exe** have it cleaned. After reading the key, **fakturax.exe** is removed and the key is lost along with it. That's why, we can easily

recover the key if, by any means, we managed to persist the original copy of the malware sample (it is not a problem if we know the source of infection, i.e in case if the malware arrived as an e-mail attachment).

In the examined variant of the malware (referred as the type 2, i.e

<u>4190df2af81ece296c465e245fc0caea</u>) – it was enough to find the key at the end of the original sample (*WARNING: this is not the original key of this sample. It has been used just to present how it works and where the real key can be found. Before trying to recover files, make sure that you made their backup, just in case if in some other editions the algorithm would be different.)

XVI32 - f	aktı	ırax	.exe	1																						_][•		Σ	3
File Edit	Se	arch	1	Add	res	5	Boo	kma	arks	Т	ool	s .	XVIs	crip	ot	Hel	р														
ഥേളി		X	Х	Ê	a (â	Q	ď	÷ [f	ŝ	N	?																		
17DA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Γ														▲
17DB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00															
17DC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00															
17DD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00															
17DE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00															
17DF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00															
17E00	31	32	33	34	35	36	37	38	39	31	32	33	34	35	36	37	1	2	3	4 5	6	7	8	9	1 2	3	4	5	6	7	
17E10	38	39	31	32	33	34	35	36	37	38	39	31	32	33	34	35	8	9	1	2 3	4	5	6	7 8	3 9) 1	2	3	4 9	5	Ŧ
Adr. dec: 97	792		0	Cha	r de	c: 4	9 (Ove	rwri	te																					//

and enter it to the text field:



in order to get all the files back.

Encryption algorithm

Authors claimed that they used AES and RSA. How it looks from the side of code?

File is encrypted chunk by chunk – single unit have 16 bytes (4 DWORDs). The key is 32 bytes long, and is preprocessed before the encryption. Both elements – the preprocessed key and a chunk of the input file – are copied to a buffer, that is supplied to the encrypting procedure.

Below – a sample file: **square.png** processed by the encrypting function. Used key: "11111...". (The copied chunk of the file has been selected on the picture)

C *G.P.U* - thread 00000AE8, module fakturax	
004019CA . MOV DWORD PTR SS:[ESP+0xC],EAX 004019CE . MOV EDI,EDI 004019D2 . MOV ECX,DWORD PTR DS:[EBX] 004019D2 . MOV ECX,DWORD PTR DS:[EAX+EDI] 004019D5 . MOV ECX,DWORD PTR DS:[EAX+EDI] 004019D5 . MOV ECX,DWORD PTR DS:[EAX+EDI]	DWORD* input DWORD input[x] DWORD input[x+1]
004019DB . MOV DWORD PTR SS: LESP+0x74J, ECX ECX <th>DWORD input[x] DWORD input[x+1] DWORD input[x]</th>	DWORD input[x] DWORD input[x+1] DWORD input[x]
004019ED LEA EAX, DWORD PTR DS:[EBX+0x10] DS:[DS:000000000000000000000000000000000	DWORD input[x+2] preprocess
E32431-D2 . POSH EHX 00401A03 . LEA ESI, DWORD PTR SS:[ESP+0x78] 00401A07 . CALL fakturax.004014D0 00401A07 . ADD ESP,0x4 00401A07 . LEA EAX.DWORD PTR SS:[ESP+0x30] 00401A07 . LEA EAX.DWORD PTR SS:[ESP+0x30] 00401A07 . LEA EAX.0x00 PTR SS:[ESP+0x30] 00401A13 . MOV ECX.0x20 00401A13 . JMPE SHORT fakturax.00401020	enc_fragment
Address Hex dump ASC	II
0224FF10 00 <	111111111111111 original key
0224FF30 107 F9 15 83 01 HH F8 E6 BC 38 D9 55 HH FF 5C 2E - 3 0224FF60 13 38 82 76 B1 60 EC 0A 6E 11 79 58 67 5A 63 24 0224FF70 00 00 00 00 00 89 50 4E 47 00 0H 1H 0H 00 00 00 00 00 0224FF30 49 48 44 52 D5 0A FE A3 94 FF 24 02 45 3C 03 77 IHD	RN. uč sec vy

after encryption (output marked gray):

004019CA 004019CE 004019D2 004019D2 004019D5 004019D9 004019DB 004019DB 004019DF 004019E2 004019E6 004019E0 004019F0 004019F0 004019F6	<pre>MOV DWORD PTR SS:LESP+0xCl,EAX MOV EDI.EDI > MOV EAX,DWORD PTR DS:LEBX1 MOV ECX,DWORD PTR DS:LEAX+EDI1 . MOV ECX,DWORD PTR DS:LEAX+EDI+0x41 . ADD EAX,EDI . MOV DWORD PTR SS:LESP+0x741,ECX . MOV ECX,DWORD PTR DS:LEAX+0x81 . MOV DWORD PTR SS:LESP+0x781,EDX . MOV EDX,DWORD PTR SS:LESP+0x781,EDX . MOV EDX,DWORD PTR SS:LESP+0x781,EDX . MOV EDX,DWORD PTR SS:LESP+0x781,EDX . LEA EAX,DWORD PTR SS:LESP+0x101 . LEA ECX,DWORD PTR SS:LESP+0x801,EDX . MOV DWORD PTR SS:LESP+0x801,EDX . LEA ECX,DWORD PTR SS:LESP+0x801,EDX</pre>	DWORD* input DWORD input[x] DWORD input[x+1] DWORD input[x+1] DWORD input[x+1] DWORD input[x+1] DWORD input[x] DWORD input[x+2]												
00401A00 00401A02 00401A03 00401A03	. MOV EAX,ECX . PUSH EAX . LEA ESI,DWORD PTR SS:LESP+0x781 . CALL fakturax.004014D0	prepared_buffer												
00401A0C 00401A0F 00401A13 00401A13	. ADD ESP,0×4 . LEA EAX,DWORD PTR SS:[ESP+0×30] . MOV ECX,0×20 . ✓ JMP SHORT fakturax.00401A20													
Address H	Hex dump ASC	п												
0224FF10 0 0224FF20 4 0224FF30 3 0224FF50 0 0224FF50 0 0224FF50 0 0224FF60 4 0224FF80 0 0224FF80 0 0224FF80 0 0224FF80 0	37 F9 15 B3 01 AA F8 E6 BC 38 D9 55 AA FF 5C 22 •3 43 38 82 76 B1 60 EC 0A 6E 11 79 58 67 5A 63 24 •3 31	6 °5"8"U \. . v ± €												

output is then copied back to the original buffer, containing the full file. Every encrypted file has a content prefixed by "ABCXYZ11" – a magic value, used by the ransomware to recognize encrypted files (it has been introduced in the newer version). Below, we can see the sample file after being dumped on the disk.

XVI32 - s	qua	re.p	ng																								_	-				Σ	3
File Edit	Sea	arch	1	Add	ress	; [Boo	kma	arks	Т	ool	s .	XVIs	scrip	ot	He	р																
රිල්ද්		X	Х	Ê	≙ (ì	୍ଦ୍	ď	¢ [f	ŝ	Ņ	?																				
0	41	42	43	58	59	5A	31	31	1A	9F	F4	DA	EE	1B	во	AA	EA	A	в	c	(Y	z	1	1	ź	ô	Ú	î	۰	۰	şe	2	
11	СВ	CF	DF	91	7 F	81	A6	24	00	60	C4	C2	46	19	63	E3	B5	č	Ď	ß	۱ ا		; ;	\$	•	Ä	Â	F	ŀ	с	ă 1	1	
22	AC	FB	89	48	8D	41	84	00	02	5D	2A	9E	во	C8	26	25	FE	-	ű	2 I	ΙŤ	A	"	٦	1	*	ž	۰	č	£	8 1	;	
33	23	10	зс	43	F3	CD	зс	C8	в1	46	E3	6C	B6	4B	50	FO	02	#	ł	< (ó	Í	<	čι	E	ă	1	I	K	P	đ,		
44	oc	A2	E6	61	B2	34	70	48	67	C0	5D	22	E8	00	77	4F	F6		•	ćł	1.	4	p	H g	ŗŔ	1	"	č		w	O à	5	
55	E6	DC	2E	89	FD	18	49	F5	91	E7	7E	CF	5E	E3	5B	6C	49	ć	Ü	. 1	:ý	t	I	ő	ç	~	Ď	^	ă	[1]	[
66	44	A6	AF	05	7E	C2	44	D1	5D	50	0E	сс	F3	74	FC	05	8A	D	1	ż	~	Â	D	ŃJ	P	ß	Ě	ó	t	ü	3	5	

16 byte long chunks of file are encrypted by AES in ECB mode.

Conclusion

First of all, not all what malware authors tell is true. In this case the key was neither RSA encrypted, nor randomly generated – just stored in the original file.

Second – immediately removing the malware is not always the best solution – sometimes we may need it to recover the data.

If you encountered a ransomware, it is better to try to gather information about it before taking any steps. In case you cannot find any information, the best way is to make a topic on the forum of your favorite vendor or contact some known analyst. We are in a constant search of samples of new threats, trying to describe and solve the problems.

And remember: only some families are really nasty. Other, like i.e <u>LeChiffre</u> have implementation flaws allowing to recover files.

Appendix

<u>https://forum.4programmers.net/Hardware_Software/264028-dma_locker_-</u> <u>_zaszyfrowane_pliki</u> – a thread on a Polish forum, created by a user infected by DMA Locker