Explained: Sage ransomware

blog.malwarebytes.com/threat-analysis/2017/03/explained-sage-ransomware/

Malwarebytes Labs

March 29, 2017

роово:	7c a7 43 da		6e 42 6b		7 43 da			
000c0:			67 35 69 da		6e 42 7b da			
:0b000			7c / Se da		6e 42 6 da			
000e0:			7	56 Sale8 da 7	700			
000f0:			75 da	66 da	67 3a VI da			
00100:			00 00 00	00 00	52.69		00 00 00 00	
10110-			00.00	00 30	00.00		00 00 00 00	
10120-								
10130			0		10	60 00 22 20	00 07 0= 00	
10140.		00002220	00		00	00 00 00 00	0 40 02 00	
0140.	00 00 02 00	00 00 00 00	00 10		00.54	6.07.06.00	00 48 02 00	
0150:	00 00 09 14	0000	00 10 5 00 0		0 09 14	ne un route	00 10 00 00	
0160:	05 00 02 00	00 00 00 000	05 00 0 000		0 0 2 00		05 30 02 00	
00170:	00 a0 06 du	000000000	54.0		a0 06 Par	00 00		
00180:		00 00 00	111 10-00	a classic states and	0000	00 00 00 00		
00190:	00 00 10 00	00 00 00 00	OT-	00 00 00 00	00	00 00 00 00	00 10 00 00	
001a0:		10 00 00 00	70 5a 0	51 m 1 m	0 00 00			
00160:				70 00 00 -	60 36 05 00			
001c0:			00 00 00 00		50.05.00			
001d0:			00 00 0	00 00 00	02 11 12 20			
)01e0:			00-00-00	00 00 00 00	00.00	00 00 00 00		00 00 0
001f0:		00 00 00 00	00.00 00.00		00 00 00 00	00 00 00 00	00 00 00 00	
00200:	00 00 00 00	00 00 00 00	0 10 03 00	a0 08 00 00		00 00 00 00	00 f0 03 00	
00210:	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00220:	00 00 00 00	00 00 00 00		74 00 00 00	00 00 00 00	00 00 00 00	20 74 65 78	
10230:	f9 d6 03 00	10 10 10 00	00 48 03 00	00 04 00 00		00 10 00 00	00 d8 03 00	

Sage is yet another <u>ransomware</u> that has become a common threat nowadays. Similarly to <u>Spora</u>, it has capabilities to encrypt files offline. The malware is actively developed and currently, we are facing an outbreak of version 2.2. of this product.

Analyzed samples

3686b6642cf6a3d97e368590557ac3f2 - JS downloader

Distribution method

Most often, Sage is dropped by downloader scripts distributed via phishing e-mails (office documents with malicious macros or standalone JS files). In the analyzed case, the sample was dropped via a JavaScript file.

Behavioral analysis

After being deployed, Sage deletes the original sample and runs another copy, dropped in %APPDATA% (names of the dropped files are different for different machines – probably generated basing on GUID):

AppData 🕨 Roaming 🕨			
Share with 🔻 New folder			
Name	Date modified	Туре	Size
📑 FkGtk5ju.exe	2017-03-25 19:19	Application	84 KB
NSba8HDh.tmp	2017-03-25 19:19	TMP File	1 KB

The dropped copy deploys itself once again, with a parameter 'g'. Example:

"C:\Users\tester\AppData\Roaming\FkGtk5ju.exe" g

After finishing its work, that dropped copy is also being deleted with the help of a batch script dropped in the %TEMP% folder.

🖃 🔝 FkGtk5ju.exe	0.12	3 372 K	8 540 K	3676
FkGtk5ju.exe		968 K	3 368 K	2508
🖃 📶 cmd.exe	0.69	2 704 K	2 524 K	2548 Windows Command Processor Microsoft Corporation
PING.EXE	1.36	652 K	2 556 K	3988 TCP/IP Ping Command Microsoft Corporation
PING.EXE	1.06	656 K	2 564 K	2364 TCP/IP Ping Command Microsoft Corporation

The content dropped in %TEMP% is shown on the below picture. We can see the batch scripts and the BMP that is being set as a wallpaper:

AppData 🕨 Local 🕨 Temp		✓ Search	Temp
Share with 🔻 New folder			
Name	Date modified	Туре	Size
config252888.bat	2017-03-24 21:03	Windows Batch File	1 KB
	2017-03-24 21:12	Windows Batch File	1 KB
FXSAPIDebugLogFile.txt	2015-06-18 22:27	Text Document	0 KB
🛃 xNK.bmp	2017-03-24 21:07	Bitmap image	4 571 KB

Sample contents of the batch scripts is given below. As we can see, the ping command is used to delay operations.

	onfig252888.bat
0:0	abx
o pi	ing 127.0.0.1 -n 2 > nul
o de	el /A /F /Q "C:\Users\tester\Desktop\sage.exe"
11	f exist "C:\Users\tester\Desktop\sage.exe" goto abx
∘ de	el /A /F /Q "C:\Users\tester\Desktop\sage.exe"
[.config252888.bat ● ∫config16184093.bat ●
۰:	abx
٥p	ping 127.0.0.1 -n 2 > nul
0 0	del /A /F /O "C:\Users\tester\AppData\Roaming\FkGtk5ju.exe"
∘i	if exist "C:\Users\tester\AppData\Roaming\FkGtk5ju.exe" goto abx
∘i ∘d	if exist "C:\Users\tester\AppData\Roaming\FkGtk5ju.exe" goto abx del /A /F /Q "C:\Users\tester\AppData\Roaming\FkGtk5ju.exe"

Just in case the system gets restarted before the encryption finished, Sage sets a link in the Startup folder, so that it can continue after the reboot:

C:\Users	s\tester\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Star	tup
V 💷 N	N5q9RdQF.Ink	c:\users\tester\appdata\roaming\fkgtk5ju.exe

However, if the ransomware successfully completed encryption process and deleted itself, the link is left abandoned.

After finishing, the wallpaper is changed. In version 2.2 the wallpaper looks very similar to 2.0, except the font is green instead of red:



At the end of the execution, the ransom note !HELP_SOS.hta opens automatically:

Decryption Instruc	tions	×
Englist	h Deutsch Italiano Português Español Français 한국어 Nederlands المريبة 中文	*
	File recovery instructions	
	You probably noticed that you can not open your files and that some software stopped working correctly.	
	This is expected. Your files content is still there, but it was encrypted by "SAGE 2.2 Ransomware".	
	Your files are not lost, it is possible to revert them back to normal state by decrypting.	
	The only way you can do that is by getting "SAGE Decrypter" software and your personal decryption key.	
	Using any other software which claims to be able to restore your files will result in files being damaged or destroyed.	Ŧ

In addition to the written information, Sage 2.2 plays a voice message informing about the infection. It is deployed via WScript running the default Microsoft voice-to-speech service – just like in the case of <u>Cerber</u>.

Some content is left in %APPDATA%:

AppData 🕨	Ro	ami	ng	۲																•	4	•		Sec	rc	h ƙ	200	mi	ing			
Share v	vith	Ŧ		Ne	w fo	olde	r																								:	≡ •
Name												Dat	te m	nodi	fied			Т	ype	2						1	Siz	e				
📗 Mozi	lla											201	5-0	6-19	9 00	:38		Fi	le f	olo	ler											
📗 Sun												201	6-0	5-31	L 23	:40		Fi	le f	olo	ler											
📗 zyna	mics											201	6-0	8-11	L 01	:21		Fi	le f	olo	ler											
💋 f1.ht	а											201	7-0	3-24	4 21	:07		Н	ΤN	1L /	٩p	olic	ati	ior	ı				6	0 K	В	
📄 NSba	8HD	h.tı	mp									201	7-0	3-24	1 21	:03		Т	MP	Fil	e									1 K	В	
45 XVI32 - 1	VSba	8H	Dh.t	tmp	,																							_		▣		83
File Edit	Sea	irch		Add	ress	; [Bool	kma	irks	Т	ool	s)	XVIs	scrip	ot	Hel	р															
0៤	2)	ĸ	Ж	ß	ð (ì	୍ଦ୍	q	:	f	å	N	?																			
0	BF	83	51	CE	C5	4F	91	FC	OF	зD	68	в1	AB	69	AA	в4	5C	ż		Q í	Ĺ	0	٦	ü	X	= 1	h d	. «	i	ş	1	
11	06	5A	56	53	78	20	E9	2F	FO	46	ЗF	27	89	16	10	ЗA	17	-	z	v٤	х		é	7	đ	F	? '	ž	т	ł	: 1	
22	41	9F	C2	12	AF	96	6C	5D	B5	CE	FO	30	39	BF	69	FF	3E	A	ź	â 1	Ż	-	1	1	μ	Î	3 0	9	ż	i	• >	
33	B2	DC	C2	0A	5B	C9	3F	74	BD	B2	06	22	04	01	00	00	00	•	Ü.	Â	1	É	?	t	"	• -	- '	L '				
44	00	00	00	00																												

Encrypted files are added to the "sage" extension and their icons are changed:

Name	Date modified	Туре	Size
🖉 !HELP_SOS.hta	2017-03-24 21:07	HTML Application	60 KB
🔒 square1 (copy).bmp.sage	2017-03-24 21:07	SAGE File	140 KB
🔒 square1.bmp.sage	2017-03-24 21:07	SAGE File	140 KB
🔒 readme.txt.sage	2017-03-24 21:06	SAGE File	8 KB
🔒 _01BE0000.mem.sage	2017-03-24 21:03	SAGE File	1 279 KB
PORTMON.CNT.sage	2017-03-24 21:07	SAGE File	1 KB

Visualization of a file – before and after encryption:



Files with the same plaintext produce different ciphertexts, that leads to the conclusion that each file is encrypted with a new key.

Sage can work well without internet connection, however, if connected it sends data via UDP (similarly to <u>Cerber</u>):

📙 FkGtk5ju.e	ke:3956 Properties			
Image	Performance	Performar	nce Graph	Threads
TCP/IP	Security	Environment	Job	Strings
Resolve	addresses			
Proto	Local Address	Remote Addr	State	
UDP te	estmachine:64480	10 A		
UDP te	estmachine:64481	11 A		

The traffic is encrypted:

😣 🖻 🗊 🛛 Wireshark • F	Follow UDP Stream (udp.stream eq 19355) · wireshark_p	ca
X 8 .IBdwR 4:cMh.EI2 (esj.Y&>"	7."k4 1k.z9bk PF{5+k..@unhh	

Page for the victim

The ransom note contains a link to the page for the victim. Encrypted and Base64 encoded key of the victim is passed via URL to the server of attackers. Example:

http://7gie6ffnkrjykggd.onion/login/AQAAAAAAAAAAAAV4NRzsVPkfwPPWixq2mqtFwGWIZTeCDpL_BGPyeJFhDA

The key can be also pasted via field on the website:

User Login
Enter your personal ID
AQAAAAAAAAAAAv4NRzsVPkfwPPWixq2mqtFwGWlZTeCDpL_BGPyeJFhDA
Log In

Keep in mind that the first login on the page for the victim triggers the timer to start. From this moment, the countdown to the price increment is running.

The website is protected by a simple captcha and allows for a simple customization – the victim can choose one of the supported languages (currently 17):



The page contains typical information, such as the amount of ransom to be paid and further instructions:

<u>S</u>	🌢 🔻 🗲 🗓 7gie	6ffnkrjy	kggd.onio	n /en/cabinet				C Search					
Sag	ge 2.0 User Area	Home	Payment	Test decryption	Instructions	Support		O Special price time remaining: 6d	23h 55m 48s				
	Important Info	rmat	ion! F	lease rea	ad very	caref	ully!		# He	ome			
					ΑΤΤΕ	NT	ION!						
		SA	GE 2	2.0 ENG	CRYP	TED	ALL YOU	JR FILES!					
	All your files, images, videos and databases where have been encrypted and no longer accessible by software known as Sage 2.0!												
	TO RESTORE ALL YOUR FILES <u>YOU NEED TO PAY</u> \$99 (≈₿0.09895) FOR THE DECRYPTION. AFTER FULL PAYMENT, YOU WILL BE ABLE TO DOWNLOAD THE SOFTWARE TO RESTORE YOUR DATA.												
	0.09895 Amount total					F	0.0989 Remains to pay	5		3			
		Mo	ore info (Э				More info ᢒ					
	99						99						
	Amount total				Ψ	F	Remains to pay		$\mathbf{\Psi}$				
		Mc	ore info (Э				More info 오					
	6d 23h 55m 48s Special price time remaining												

The malware allows to test decryption capabilities by permitting the victim to upload some encrypted files (the size of the file must be lesser than 15 KB):

est decryption			
readme.txt.sage			
(7.81 KB)			

However, the result is not available instantly:

Test decryption Upload file to test decryption	🖨 Home 🤉	Test decryption
You can upload one file up to 15 KB to test decryption		
▲ Test decryption		
 Uploading completed successfully! 		
Link to the decrypted file will appear on this page after some time. If the link does not appear within 10 m page.	inutes, please	refresh this

After some hours, the decrypted version of the uploaded file is indeed available to download:

Sage 2.0 User Area	Home	Payment	Test decryption	Instructions	Support		Special price time remaining: 4d 07h 45m 02s		•					
Test decryptio	Test decryption Upload file to test decryption													
You can upload o	You can upload one file up to 15 KB to test decryption													
▲ Test decryptio	n							-						
	ODownload file													

Inside

Sage is delivered packed by various crypters. After defeating the first layer we obtain second PE file – the malicious core, that is not further obfuscated.

At the beginning of the execution, Sage generates the Victim ID/key and saves it in the .tmp file dropped in %APPDATA% folder. Then, it removes backups from the system:

AGOFFIND IN MOULEON FOR	
	FRGTKSJU.000EE670
000ES460 . PUSH FOX	
008E54C1 . PUSH Ek6tk5 ju, 008EE768	UNICODE "delete shadows Zall Zquiet"
PUSH FkGtk5.ju.008EE74C	UNICODE "Vssadmin.exe"
008E54CB CALL FkGtk5ju 008E5250	
008E54D0 PUSH ESI	FkGtk5ju.008EE678
008E54D1 . PUSH FkGtk5ju.008EE708	UNICODÉ "/set (default) recoveryenabled no"
008E54D6 PUSH FkGtk5ju.008EE6EC	UNICODE "bodedit.exe"
008E54DB	
008E54E0 PUSH ESI	FkGtk5ju.008EE678
PUSH FKGtK5ju 008EE688	UNICODE "Set (default) bootstatuspolicy ignoreallfailures"
	ONICODE "Dededit.exe"
008E54E3	02EDE92C
POP ECX	AZEDE92C
008E54F5 . RETN	

Executed commands:

```
vssadmin.exe delete shadows /all /quiet
bcdedit.exe /set {default} recoveryenabled no
bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures
```

Sage enumerates through the files, and if they matched the defined criteria, they are getting encrypted. First, the malware creates a file with the same name as the attacked one, but with three dots at the end.



Both files coexist in the system until the encrypting is finished.

Name .	Date modified	Туре	Size	
📄 xed-format-options_8h.html	2017-03-27 15:48	File		4 KB
xed-format-options_8h.html	2015-01-20 23:23	Firefox HTML Doc		4 KB

Then, the original file is deleted and the newly created one - renamed with the extension .sage:

008F8129		PUSH EST	
00000120	11 É -	PUCH FROND IN GOOFFORD	OSCIL "YS cope"
00050105	. I I		HOULT NOT SARE
008E812F	. · ·	CHLL FRGTRSJU.008EB620	
008E8134		ADD ESP.0x8	
008E8137		PUSH 0x1	Flags = REPLACE EXISTING
008E8139		MOV EDI.EAX	
009E913B	II .	PUSH ENT	NeuName - NILL
00000100	H •		Terminate - House Photo October into an end in Contract and Contract Contract of Contract - Photo October into the Contract - Photo October into October in Contract - Photo October into O
008E813C	11 · · ·	PUSH EBP	ExistingName = "\\\\?\\C:\\pin\\extras\\xed-la32\\doc\\ref-manual\\ntml\\xed-format-options_8n.ntml"
008E813D	II	CALL DWORD PTR DS:[<&KERNEL32.MoveFileExW>]	▲MoveFileExW
008E8143	II	PUSH EDI	
008F8144		COLL EVENNES IN MAREADIA	
00000140	H		
000E0142	11 · · ·	HDD ESF, 084	
008E814C	H •	PUSH ESI	 FileName = "\\\\?\\C:\\pin\\extras\\xed-ia32\\doc\\ref-manual\\html\\xed-format-options_8h.html"
008E814D		CALL DWORD PTR DS:[<&KERNEL32.DeleteFileW>]	L De leteFileW
008E8153	11.1	PUSH FBP	
AGOEO1E4	п.	COLL EVENUE IN GOVERN10	
00000104	11 ·	CHEE INGUNOJULOOCEDIO	
008E8159	11 · · ·	HUD ESP,084	
008E815C	н.	POP FBP	Ek6tk5 ju, 008EE930

At the end, only the .sage file is left:

Name .	Date modified	Туре	Size
xed-format-options_8h.html.sage	2017-03-27 15:48	SAGE File	4 KB

What is attacked?

Sage comes with a long list of the attacked extensions, that is hard-coded in the binary:

dat mx0 cd pdb xqx old cnt rtp qss qst fx0 fx1 ipg ert pic img cur fxr slk m4u mpe mov wmv mpg vob mpeg 3g2 m4v avi mp4 flv mkv 3gp asf m3u m3u8 wav mp3 m4a m rm flac mp2 mpa aac wma djv pdf djvu jpeg jpg bmp png jp2 lz rz zipx gz bz2 s7z tar 7z tgz rar ziparc paq bak set back std vmx vmdk vdi qcow ini accd db sqli sdf mdf myd frm odb myi dbf indb mdb ibd sql cgn dcr fpx pcx rif tga wpg wi wmf tif xcf tiff xpm nef orf ra bay pcd dng ptx r3d raf rw2 rwl kdc yuv sr2 srf dip x3f mef raw log odg uop potx potm pptx rss pptm aaf xla sxd pot eps as3 pns wpd wps msg pps xlam xll ost sti sxi otp odp wks vcf xltx xltm xlsx xlsm xlsb cntk xlw xlt xlm xlc dif sxc vsd ots prn ods hwp dotm dotx docm docx dot cal shw sldm txt csv mac met wk3 wk4 uot rtf sldx xls ppt stw sxw dtd eml ott odt doc odm ppsm xlr odc xlk ppsx obi ppam text docb wb2 mda wk1 sxm otg oab cmd bat h asx lua pl as hpp clas js fla py rb jsp cs c jar java asp vb vbs asm pas cpp xml php plb asc lay6 pp4 pp5 ppf pat sct ms11 lay iff ldf tbk swf brd css dxf dds efx sch dch ses mml fon gif psd html ico ipe dwg jng cdr aep aepx 123 prel prpr aet fim pfb ppj indd mhtm cmx cpt csl indl dsf ds4 drw indt pdd per lcd pct prf pst inx plt idml pmd psp ttf 3dm ai 3ds ps cpx str cgm clk cdx xhtm cdt fmv aes gem max svg mid iif nd 2017 tt20 qsm 2015 2014 2013 aif qbw qbb qbm ptb qbi qbr 2012 des v30 qbo stc lgb qwc qbp qba tlg qbx qby 1pa ach qpd gdb tax qif t14 qdf ofx qfx t13 ebc ebq 2016 tax2 mye myox ets tt14 epb 500 txf t15 t11 gpc qtx itf tt13 t10 qsd iban ofc bc9 mny 13t qxf amj m14 _vc tbp qbk aci npc qbmb sba cfp nv2 tfx n43 let tt12 210 dac slp qb20 saj zdb tt15 ssg t09 epa qch pd6 rdy sic ta1 lmr pr5 op sdy brw vnd esv kd3 vmb qph t08 qel m12 pvc q43 etq u12 hsr ati t00 mmw bd2 ac2 qpb tt11 zix ec8 nv lid qmtf hif lld quic mbsb nl2 qml wac cf8 vbpf m10 qix t04 qpg quo ptdb gto pr0 vdf q01 fcr gnc ldc t05 t06 tom tt10 qb1 t01 rpf t02 tax1 1pe skg pls t03 xaa dgc mnp qdt mn8 ptk t07 chg #vc qfi acc m11 kb7 q09 esk 09i cpw sbf mql dxi kmo md u11 oet ta8 efs h12 mne ebd fef qpi mn5 exp m16 09t 00c qmt cfdi u10 s12 qme int? cf9 ta5 u08 mmb qnx q07 tb2 say ab4 pma defx tkr q06 tpl ta2 qob m15 fca eqb q00 mn4 lhr t99 mn9 qem scd mwi mrq q98 i2b mn6 q08 kmy bk2 stm mn1 bc8 pfd bgt hts tax0 cb resx mn7 08i mn3 ch meta 07i rcs dtl ta9 mem seam btif 11t efsl \$ac emp imp fxw sbc bpw mlb 10t fa1 saf trm fa2 pr2 xeq sbd fcpa ta6 tdr acm lin dsb vyp emd pr1 mn2 bpf mws h11 pr3 gsb mlc nni cus ldr ta4 inv omf reb qdfx pg coa rec rda ffd ml2 ddd ess qbmd afm d07 vyr acr dtau ml9 bd3 pcif cat h10 ent fyc p08 jsd zka hbk bkf mone pr4 qw5 cdf gfi cht por qbz ens 3pe pxa intu trn 3me 07g jsda 2011 fcpr qwmo t12 pfx p7b der nap p12 p7c crt csr pem gpg key

In order to access all the files without any interference, Sage searches and terminates any associated processes. Processes are identified by their names:

msftesql.exe sqlagent.exe sqlbrowser.exe sqlservr.exe sqlwriter.exe oracle.exe ocssd.exe dbsnmp.exe synctime.exe mydesktopqos.exe agntsvc.exe isqlplussvc.exe xfssvccon.exe mydesktopservice.exe ocautoupds.exe encsvc.exe firefoxconfig.exe tbirdconfig.exe ocomm.exe mysqld.exe mysqld-nt.exe mysqld-opt.exe dbeng50.exe sqbcoreservice.exe

As it is common in ransomware, some paths are excluded from the attack. In this case, blacklisted are not only system directories, but also others, related to popular games like "League of Legends", "steamapps", "GOG Games", and etc.

tmp Temp winnt 'Application Data' AppData ProgramData
'Program Files (x86)' 'Program Files' '\$Recycle Bin'
'\$RECYCLE BIN' Windows.old \$WINDOWS.~BT DRIVER DRIVERS
'System Volume Information' Boot Windows WinSxS DriverStore
'League of Legends' steamapps cache2 httpcache GAC_MSIL
GAC_32 'GOG Games' Games 'My Games' Cookies History IE5
Content.IE5 node_modules All Users AppData ApplicationData
nvidia intel Microsoft System32 'Sample Music'
'Sample Pictures' 'Sample Videos' 'Sample Media' Templates

Some countries (recognized by keyboard layouts) are also excluded from the attack. Below is the function checking if the selected keyboard layout is present in the system:

```
signed int usercall has keyboard layout@<eax>( int16 searched id@<si>)
  int v1; // eax@1
  signed int result; // eax@2
  int index; // ecx@3
  int16 kbds list[40]; // [sp+0h] [bp-50h]@1
  v1 = GetKeyboardLayoutList(20, (HKL *)kbds_list);
  if ( \cup 1 > 0 )
  {
    index = 0;
    if ( U1 <= 0 )
LABEL_6:
      result = 0;
    >
    else
    {
      while ( (kbds list[2 * index] & 1023) != searched id )
      {
        if (++index >= v1)
          goto LABEL_6;
      3
      result = 1;
    }
  }
  else
  Ł
    result = 0;
  - 3
  return result;
b
```

Systems with the following <u>keyboard layouts</u> are omitted by Sage 2.2: Belarusian, Kazak, Ukrainian, Uzbek, Sakha, Russian, Latvian.

```
B00L check_keyboard_layouts()
{
    return has_keyboard_layout(0x23) // Belarusian
    || has_keyboard_layout(0x3F) // Kazak
    || has_keyboard_layout(0x22) // Ukrainian
    || has_keyboard_layout(0x43) // Uzbek
    || has_keyboard_layout(0x85) // Sakha
    || has_keyboard_layout(0x19) && !has_keyboard_layout(0x26);// Russian, Latvian
}
```

How does the encryption works?

Sage uses two cryptographic algorithms: Elliptic Curves and ChaCha20. ChaCha20 is used to encrypt content of each file, while ECC is used to protect the randomly generated keys.

Each random key is retrieved using a cryptographically secure generator (<u>SystemFunction036</u>). The filled buffer is preprocessed by a simple algorithm:

Victim ID

At the beginning of the execution, Sage creates a random buffer and encrypts it using ECC. The buffer created in the first round of encryption we will refer as a Victim ID and the output of the next rounds – as Encrypted Victim ID.

```
int __cdecl make_victim_keys(int victim_id_buffer, int h_key)
{
    char random_val; // [sp+4h] [bp-40h]@1
    char key2; // [sp+24h] [bp-20h]@1
    *(_BYTE *)(victim_id_buffer + 64) = 1;
    make_random_key((int)&random_val);
    ecc_make_key1(victim_id_buffer, &random_val); // Victim_ID
    ecc_make_key2((int)&key2, &random_val, (const void *)h_key);
    process_buffer((int)&key2);
    ecc_make_key1(victim_id_buffer + 32, &key2); // Encrypted Victim_ID
    return 0;
}
```

In the first round, the random value is encrypted using ECC, producing the Victim ID.

In the second round, the same random value is encrypted using ECC along with another buffer, that is hardcoded in the binary. The output is processed in the similar way like the random buffer:

```
BYTE *_cdecl process_buffer(int buffer)
{
    BYTE *result; // eax@1
    char v2; // cl@1
    result = (_BYTE *)buffer;
    v2 = *(_BYTE *)(buffer + 31);
    *result &= 0xF8u;
    *(_BYTE *)(buffer + 31) = v2 & 0x3F | 0x40;
    return result;
}
```

In the third round, the resulting buffer is again encrypted by ECC – producing the Encrypted Victim ID.

Both output buffers are kept in the memory of the application and used further (also they are saved in the TMP file dropped in %APPDATA% folder).

<mark>00406660 call</mark> 00406665 test 00406667 jns	<pre>read_temp_file eax, eax short loc_406691</pre>
	¥
🗾 🚄 🖼	
00406669 mov	eax, 1pBuffer
0040666E push	72
00406670 push	0
00406672 push	eax
00406673 <mark>call</mark>	clear_buffer
00406678 mov	ecx, 1pBuffer
0040667E push	offset hardcoded_key
00406683 push	ecx
00406684 <mark>call</mark>	make_victim_keys
00406689 add	esp, 14h
0040668C <mark>call</mark>	write_tmp_file

The part highlighted on the screenshot is the Victim ID (after that, next 32 bytes are the Encrypted Victim ID):

AppData 🕨 Roaming 🕨																	
Name		Ť						Dat	te m	odifi	ed		Тур)e			Size
NSba8HDh.tmp 2017-03-27 15:22 TMP File 1 KB																	
HxD - [C:\Users\tester\AppData\Roaming\NSba8HDh.tmp]																	
🔝 File Edit Search View Analysis Extras Window ?																	
🗋 👌 🕶 🕞	and	U	++	16		•	AN	SI		•	he	x	•	·			
NSba8HDh.t	tmp																
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	OF	
00000000	1B	0B	19	F8	80	0E	E7	FA	67	В7	32	37	52	70	BC	57	ř€.çúg 27RpLW
00000010	07	92	65	76	8A	43	48	13	27	D3	D1	4B	C0	24	88	09	.'evŠCH.'ÓŃKŔ\$
00000020	5D	89	7E	5D	B5	A6	B6	DF	56	4A	FO	66	0E	90	19	ЗD]‰~]µ¦¶ßVJdf=
00000030	1F	40	10	5A	40	4E	OF	6B	D3	22	5B	5F	84	58	0E	66	.@.Z@N.kO"[_"X.f
00000040	01	00	00	00	00	00	00	00									

The victim ID is also saved in the ransom note, in Base64* version:

If you are asked for your personal key, copy it to the form on the site. This is your personal key:

AQAAAAAAAAAGwsZ-IAO5_pntzI3UnC8VweSZXaKQ0gTJ9PRS8AkiAnA

*The character set is slightly modified in comparison to the classic Base64. In order to decode it as Base64 we must replace '-' with '+' and '_' with '/' for example the ID: AQAAAAAAAAAAAGwsZ-IAO5_pntzI3UnC8VweSZXaKQ0gTJ9PRS8AkiAnA is Base64: AQAAAAAAAAAAGwsZ+IAO5/pntzI3UnC8VweSZXaKQ0gTJ9PRS8AkiAnA

In addition, the Victim ID is also saved in each and every encrypted file:

😰 NSba8HDh.t	tmp	FD	xed	-isa-	set_8	h-so	urce	htm	nl.sag	e							
Offset(h)	00	01	02	03	04	05	06	07	08	09	OA	0B	oc	OD	0E	OF	
000019F0	72	C0	05	28	2B	2D	77	8B	9D	52	9E	E1	7E	8B	5B	A1	rŔ.(+-w <tržá~<[`< td=""></tržá~<[`<>
00001A00	F7	F9	57	FA	DB	AD	DE	9E	5A	1B	0B	19	F8	80	0E	E7	÷ůWúŰ.ŢžZ <mark>ř€.</mark> ç
00001A10	FA	67	Β7	32	37	52	70	BC	57	07	92	65	76	8A	43	48	úg 27RpLW.'evŠCH
00001A20	13	27	DЗ	D1	4B	C0	24	88	09	00	60	CE	5C	E2	4B	03	.'ÓŃKŔ\$ `î\âK.
00001A30	18	66	02	78	AA	Α4	DB	F8	A2	9D	47	D8	A 8	60	AD	50	.f.xޤŰř~ťGŘ~`.P
00001A40	40	AF	D7	08	F9	BE	C3	FF	18	CB	F5	0C	76	05	1A	00	@Ż×.ůIĂ [·] .Ëő.v
00001A50	00	00	00	00	00	00	00	00	00	01	00	00	00	BE	BA	9E	Işž
00001A60	5A	00	00	00	00												Ζ

The Encrypted Victim ID takes part in encrypting file's content (as a key unique per victim).

File encryption

At the beginning of the file encrypting function, a new 32 bytes long key is generated (unique per each file).

The random number is encrypted with the help of ECC twice:

- Individually to make the key1 that is stored in the file
- Along with the Encrypted Victim's ID to make the key2, used by ChaCha20

```
make_random_key((int)&random_val);
sub_404EE0((int)&chacha_key1, &random_val);
sub_404F20((int)&chacha_key2, &random_val, (const void *)(victim_id + 32));
v17 = 0;
v18 = 0;
GetFileSizeEx(hFile, &FileSize);
v6 = a5;
if ( a5 == 2 && FileSize.QuadPart <= 0x400000ui64 )</pre>
Ł
  a5 = 1;
 v6 = 1;
>
U7 = 0;
v12 = FileSize.QuadPart;
if ( V6 == 2 )
Ł
 v12 = 0x400000i64;
}
else if ( v6 == 3 )
Ł
  if ( FileSize.s.LowPart < 0x400000 )</pre>
    HIDWORD(v12) = FileSize.s.HighPart - 1;
  LODWORD(v12) = FileSize.s.LowPart - 0x400000;
  v7 = 0x400000;
3
v11 = 0:
lpBuffer = sub_409D70(1, 0x20000);
chacha20_init(&chacha_context, &chacha_key2, &v17);
```

As we can see, the *key2* is used to initialize the cryptographic function's context. ChaCha20 can be recognized by <u>typical constants</u> used in the initialization function:

```
00401820 chacha20_init proc near
00401820
00401820 arg_0= dword ptr
                           4
00401820 arg_4= dword ptr
                           8
00401820 arg 8= dword ptr
                           ØCh
00401820
00401820 mov
                 eax, [esp+arg_8]
00401824 mov
                 ecx, [esp+arg_0]
                 dword ptr [ecx], 61707865h
00401828 mov
0040182E mov
                 dword ptr [ecx+4], 3320646Eh
00401835 mov
                 dword ptr [ecx+8], 79622D32h
0040183C mov
                 dword ptr [ecx+0Ch], 6B206574h
00401843 movzx
                 edx, byte ptr [eax+3]
```

The file is encrypted chunk by chunk (the maximal chunk size is 0x20000) with the help of ChaCha20:



At the end of the file, the first derived key (key1) and some additional data is appended:

```
if ( !SetFilePointerEx(file, FileSize, 0, 0) )
goto LABEL_33;
append_encrypted_data(file, victim_id, (int)&FileSize, (int)&chacha_key1, a5);
```

Appended data is separated from the encrypted file's content by two hard-coded markers: 0x5A9E**DEAD** and 0x5A9E**BABE**



Markers at the end of the encrypted file:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	OD	0E	OF	
00000C40	89	C4	87	FA	54	0E	FO	08	24	B5	6B	DD	C6	32	Α5	B7	‰Ä‡úT.đ.\$µkÝĆ2Ą∙
00000C50	6B	FA	DB	26	BD	BE	ΕA	FA	ЗE	1F	D1	OF	26	2C	E2	28	kúŰ&″Ięú≻.Ń.&,â(
000000060	20	1A	64	17	FA	AD	DE	9E	5A	1B	0B	19	F8	80	0E	E7	.d.ú.ŢžZř€.ç
00000070	FA	67	В7	32	37	52	70	BC	57	07	92	65	76	8A	43	48	úg·27RpLW.'evŠCH
00000C80	13	27	DЗ	D1	4B	C0	24	88	09	80	11	5F	B2	EB	ЗC	1C	.'ÓŃKŔ\$€įë<.
00000C90	зв	ΕO	35	F9	7D	63	6C	E5	4A	8B	ЗC	0E	8B	EΒ	D8	02	;ŕ5ů}clĺJ<<.<ëŘ.
00000CA0	72	10	43	43	09	8F	C4	ЗD	66	СВ	F5	0C	76	65	0C	00	r.CC.ŹÄ=fËő.ve
00000CB0	00	00	00	00	00	00	00	00	00	01	00	00	00	BE	BA	9E	Işž
000000000	5A	00	00	00	00												z

After the first marker Sage stores the following information: Victim ID, Key1, size of the original file.

Network communication

Sage does not need any data from the CnC in order to work. However, as mentioned before, it may generate some UDP traffic. It is because it has capabilities to send some data about the attacked system. Depending on the configuration, the data may be sent either via UDP or via HTTP POST request. The data is encrypted before being sent – also with the help of ChaCha20 algorithm. In the observed case, the ChaCha20 key was a buffer filled with 0 bytes.

```
v6 = 0;
chacha20_init(&v12, dword_415E2C + 8, &v6);
v2 = *(_DWORD *)(a1 + 4);
v3 = *(_BYTE **)a1;
v4 = sub_40A9B0((int)&buf, v2);
chacha20_encrypt((int)&v12, v4, v3, v2);
res = to_post_request((int)&buf);
if ( res < 0 )
    send_via_udp(buf, len);
result = res;
```

Examples of the data sent to the CnC

Sage sends the generated keys to the CnC, i.e.:

NSba8HDh.tmp																	
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	OF	
00000000	СВ	3B	94	D9	65	A3	89	97	8A	16	03	5E	D7	00	C8	7A	Ë;″Úeى—Ś^×.Ċz
00000010	78	00	88	73	09	89	C2	4C	58	13	25	34	0A	86	6C	4B	xs.%ÂLX.%4.†1K
00000020	2B	B7	BD	53	94	B8	45	62	9C	90	BB	2B	43	D9	65	5D	+·″S″,Ebś.»+CŪe]
00000030	C9	C8	63	47	C4	C6	95	AB	18	15	0D	70	31	В9	E4	1F	ÉČcGÄĆ•«pląä.
00000040	01	00	00	00	00	00	00	00									

Compare with the buffer before encryption:



The same data is also formatted into a human-readable form, like shown below. However, so far we didn't observed any use of this data. It may be some unfinished feature, that will be developed further in new versions of this product. Formatted equivalent of the above buffer:

```
[bin(33) 01CB3B94D965A389978A16035ED700C87A780088730989C24C581325340A866C4B, 4, {
    "v": 1,
    "gpk": bin(32) CB3B94D965A389978A16035ED700C87A780088730989C24C581325340A866C4B,
    "pk": bin(32) 2BB7BD5394B845629C90BB2B43D9655DC9C86347C4C695AB18150D7031B9E41F,
}]
```

Other examples – collected information about the attacked machine:

```
[bin(33) 01CB3B94D965A389978A16035ED700C87A780088730989C24C581325340A866C4B, 3, {
```

```
"s": {
    "w": {
      "v": [
        6,
        1,
        false,
        false,
        7601,
        1,
        Θ,
      ],
      "u": "tester",
      "p": "TESTMACHINE",
    },
    "c": "
                  Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz",
    "m": 232,
    "k": [68486165, 4026598409, 4026991637],
  },
  "i": 12288,
  "w": null,
31
```

Adding icons

Interesting and uncommon feature deployed by Sage is the change of icons for the used datatypes. Padlock icon is added to the encrypted files with the *.sage* extension and the key icon is added to the files with *.hta* extensions (that are used for the ransom notes). Icon change is implemented via setting appropriate registry keys:

```
LSTATUS cdecl add sage icons(int a1, char a2)
  HKEY v2; // ecx@0
   LSTATUS result; // eax@1
   const WCHAR *v4; // edi@2
   HKEY phkResult; // [sp+0h] [bp-4h]@1
   phkResult = v2;
   result = RegCreateKeyExA(HKEY CURRENT USER, "Software\\Classes", 0, 0, 0, 0xF003Fu, 0, &phkResult, 0);
  if ( !result )
   {
      v4 = (const WCHAR *)sub 40B620((int)"mshta.exe \"%s\" \"%%1\"", a2);
      RegSetValueW(phkResult, L".sage.notice\\DefaultIcon", 1u, L"%WinDir%\\system32\\shell32.dll,47", 0);
RegSetValueW(phkResult, L"sage.notice\\DefaultIcon", 1u, L"%WinDir%\\system32\\shell32.dll,47", 0);
RegSetValueW(phkResult, L"sage.notice\\FriendlyTypeName", 1u, L"encrypted by SAGE", 0);
      RegSetValueW(phkResult, L'sage.notice\\shell\\open\\command", 1u, v4, 0);
RegSetValueW(phkResult, L'sage.notice\\DefaultIcon", 1u, L'%WinDir%\\system32\\shell32.dll,47", 0);
RegSetValueW(phkResult, L'htafile\\DefaultIcon", 1u, L'%WinDir%\\system32\\shell32.dll,44", 0);
      get_proc_heap((LPV0ID)v4);
      result = RegCloseKey(phkResult);
   return result;
13
```

Conclusion

Sage, similar to Spora, uses a complex way of deriving keys. So far, there is no solution that would allow recovering files without paying the ransom – that's why we recommend focusing on prevention instead. <u>Malwarebytes 3.0 Premium</u> users are protected from Sage ransomware as long as it is installed prior to being infected.

Appendix

<u>https://blog.fortinet.com/2017/02/02/a-closer-look-at-sage-2-0-ransomware-along-with-wise-mitigations</u> – Fortinet about Sage 2.0

This was a guest post written by Hasherezade, an independent researcher and programmer with a strong interest in InfoSec. She loves going in details about malware and sharing threat information with the community. Check her out on Twitter @hasherezade and her personal blog: <u>https://hshrzd.wordpress.com</u>.