Looking Into Jaff Ransomware

By Raul Alvarez

Published: 2019-01-31 · Archived: 2026-04-05 23:34:35 UTC

FortiGuard Labs Threat Analysis Blog

Jaff ransomware was originally released in the spring of 2017, but it was largely neglected because that was the same time that WannaCry was the lead story for news agencies around the world. Since that time, Jaff ransomware has lurked in the shadows while infecting machines worldwide. In this FortiGuard Labs analysis, we will look into some of the common ransomware techniques used by this malware, and how it represents the ransomware's infection routine in general.

Entry Point

Like many ransomware variants, Jaff ransomware commonly arrives as a pdf attachment. Once you open the attachment, it displays a one-line document along with a pop-up message asking whether you want to open an embedded (See figure 1).

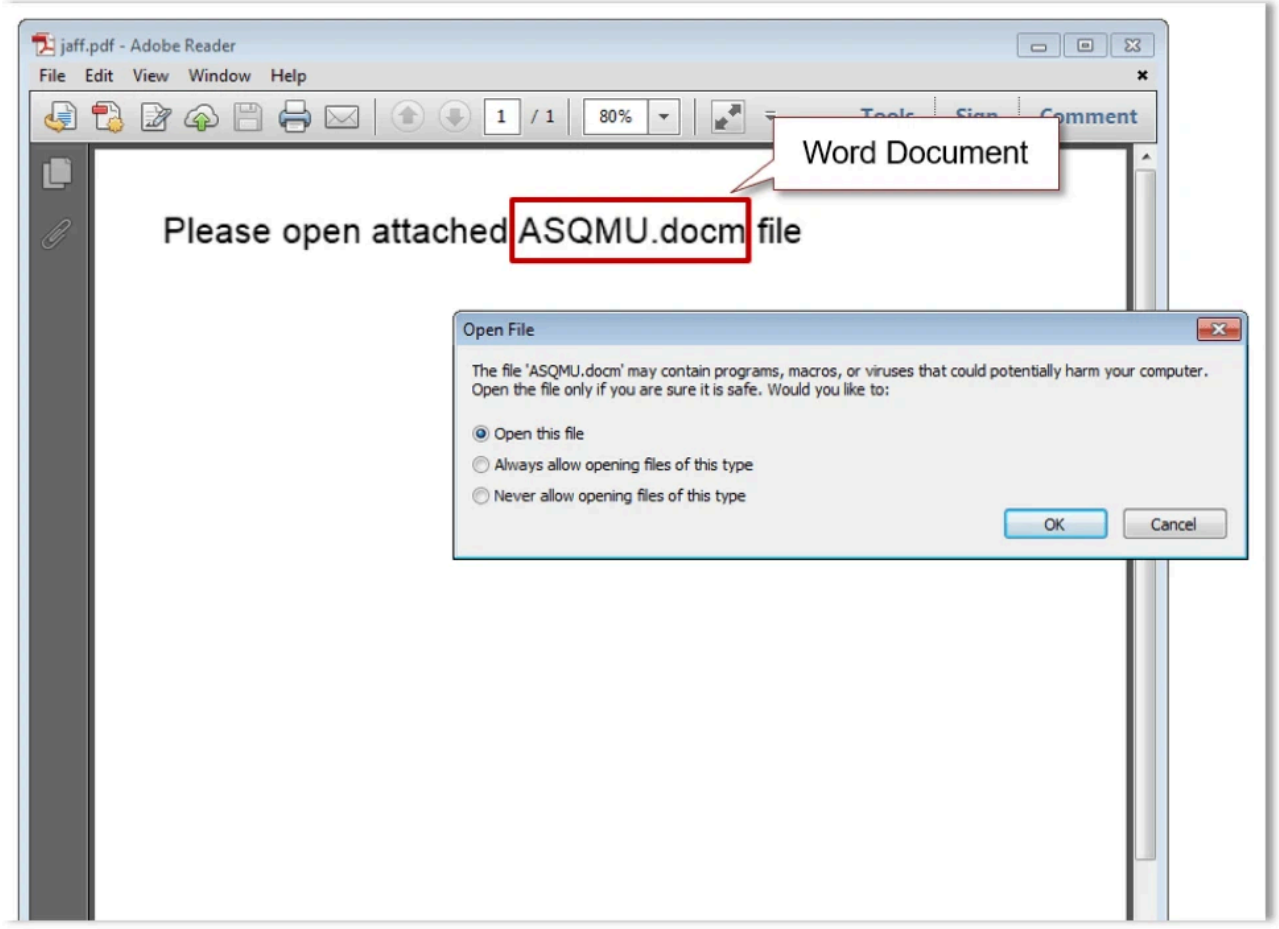

Figure 1

If you choose to open the file, that's where the fun begins. It then launches an embedded document that contains instructions on how to remove Macro protection from your document (See Figure 2). The yellow strip at the top of the document includes the button "Enable Content," which enables any macro within the document to execute. And of course, we all already know that this document contains macros.

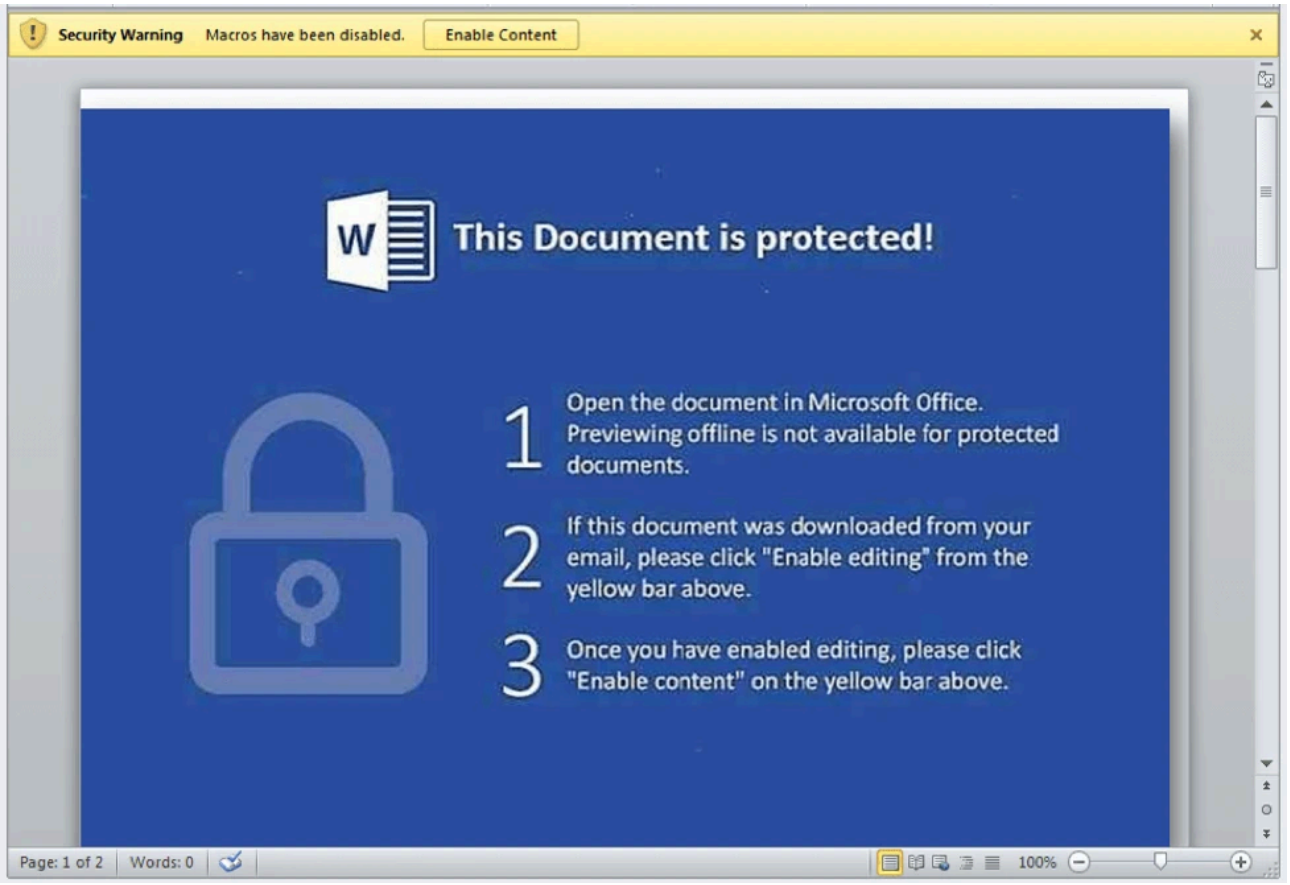

Figure 2

Macros Everywhere

In fact, this document contains lots of macros (See figure 3), only one of which downloads the Jaff binary file. The following is a list of macros found in this variant:

- autoopen()
- Document_Open()
- setAsMainTarget()
- Challenge(sender As String, e As Integer)
- Subfunc(MethodParam2() As Byte, MethodParam As String)
- Lipochanko(a, b)
- Synomati(Comps)

- Vgux(strComputer As Integer)
- enumMembers(objDomain)
- Assimptota4(FullPath As String, NumHoja As Integer)
- Assimptota6(FullPath As String, NumHoja As Integer)
- WidthA(Dbbb As String, bbbJ As String, Optional system_ofADown_Sexote As String)
- Function system_ofADown_ProjectSpeed()
- privateProbe()
- SaveDataCSVToolStripMenuItem_Click(e As Integer)
- RepackOK(sheetToMove As String, sheetAnchor As String, Assimptota6OrAfter As String)
- CheckRectsAd()

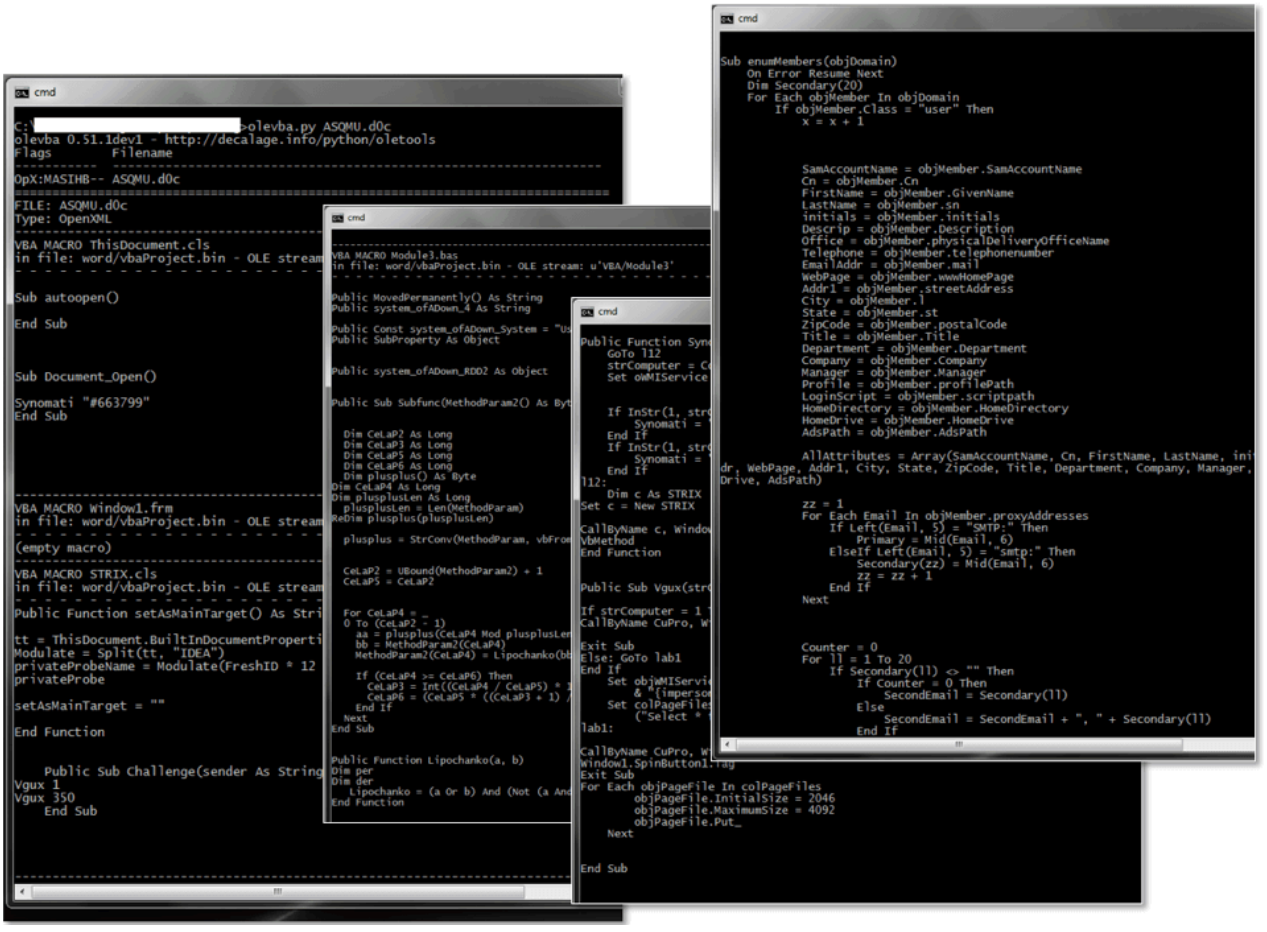

Figure 3

privateProbe()

The privateProbe() macro contains the code that downloads the Jaff binary file (See figure 4).

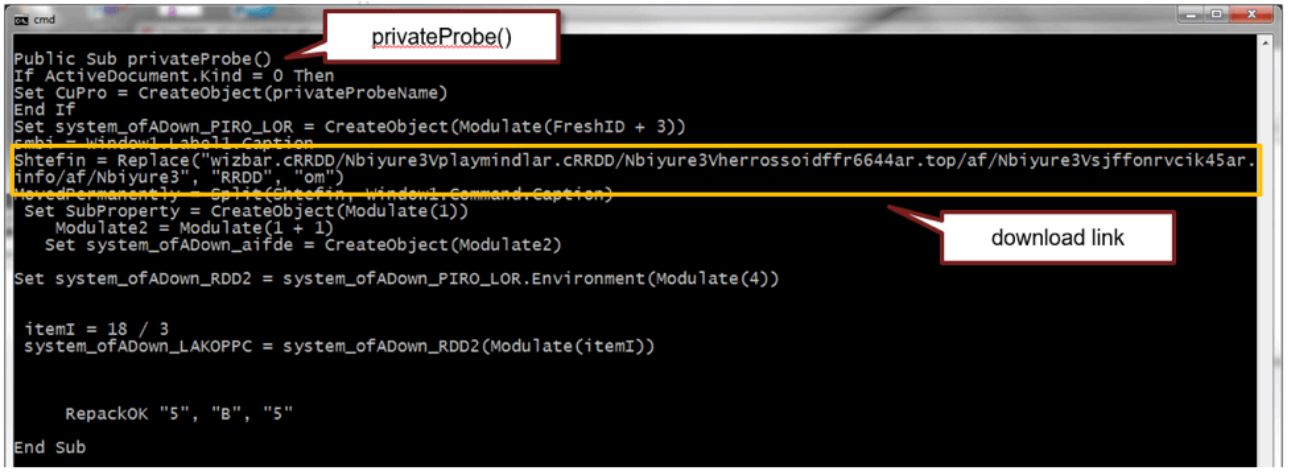

Figure 4

We can do a simple substitution to manually generate the download link.

From the encoded links, we can replace the letters “RRDD” with “om”, and splitting the links from every occurrence of the word “Nbiyure3”(See figure 5).

Figure 5

Decryption, Redirection, and Garbage Code

After downloading the binary file, Jaff ransomware starts decrypting part of the malware code. It uses a simple code redirection routine as an anti-analysis trick to stretch the time it requires to analyze the actual malicious code. In between code execution, it uses garbage code that is not relevant to the malware execution.

Figure 6 shows different blocks of code executed in a random fashion. Each pass from this group of codes decrypts a DWORD value, and then continues until it decrypts the rest of the malware. It also shows the numbered

directions of code execution for the decryption routine.

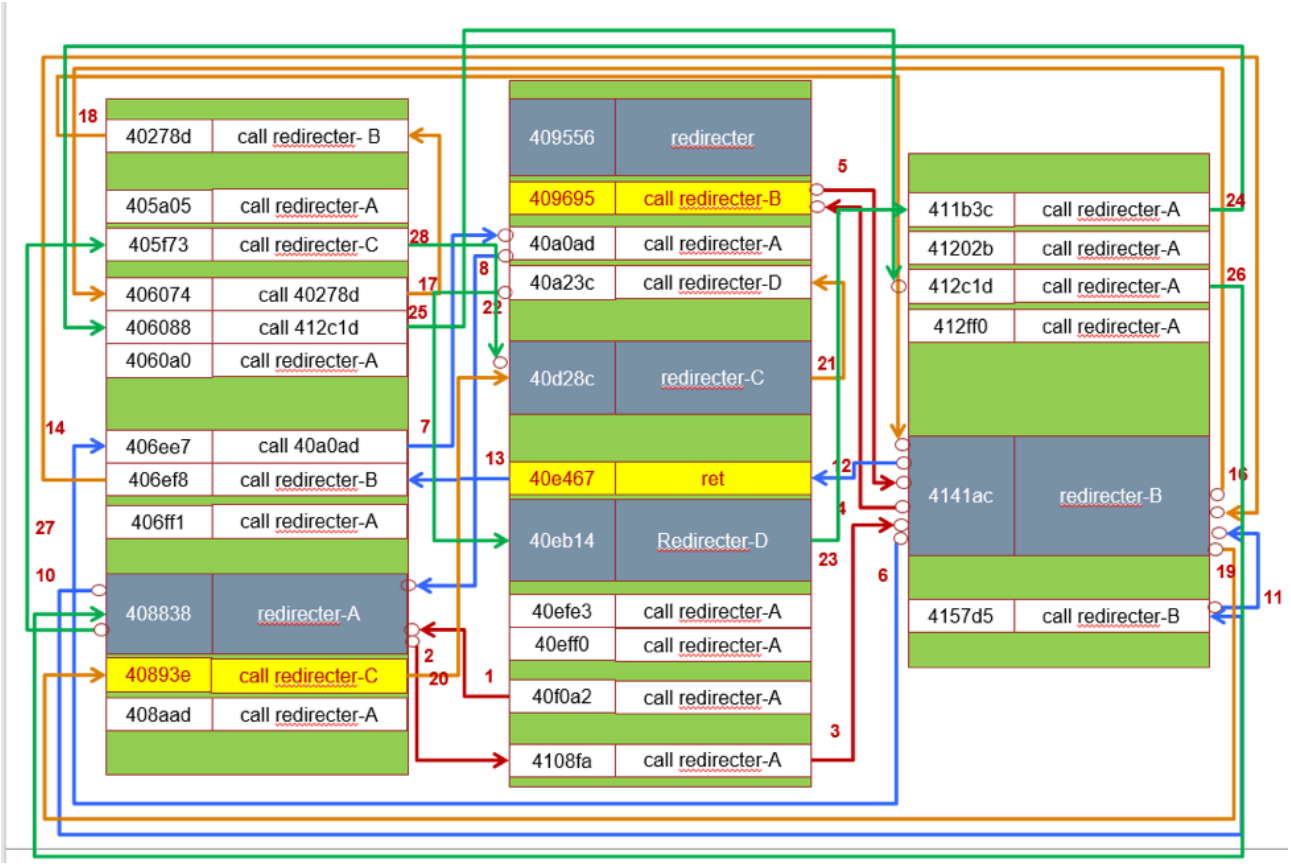

Figure 6

Once we remove the garbage code and irrelevant blocks, we can see that only three blocks are used for the actual decryption. Figure 7 shows the same group of blocks highlighting the actual relevant code used for the decryption routine. It turns out that the actual decryption routine is just a simple XOR.

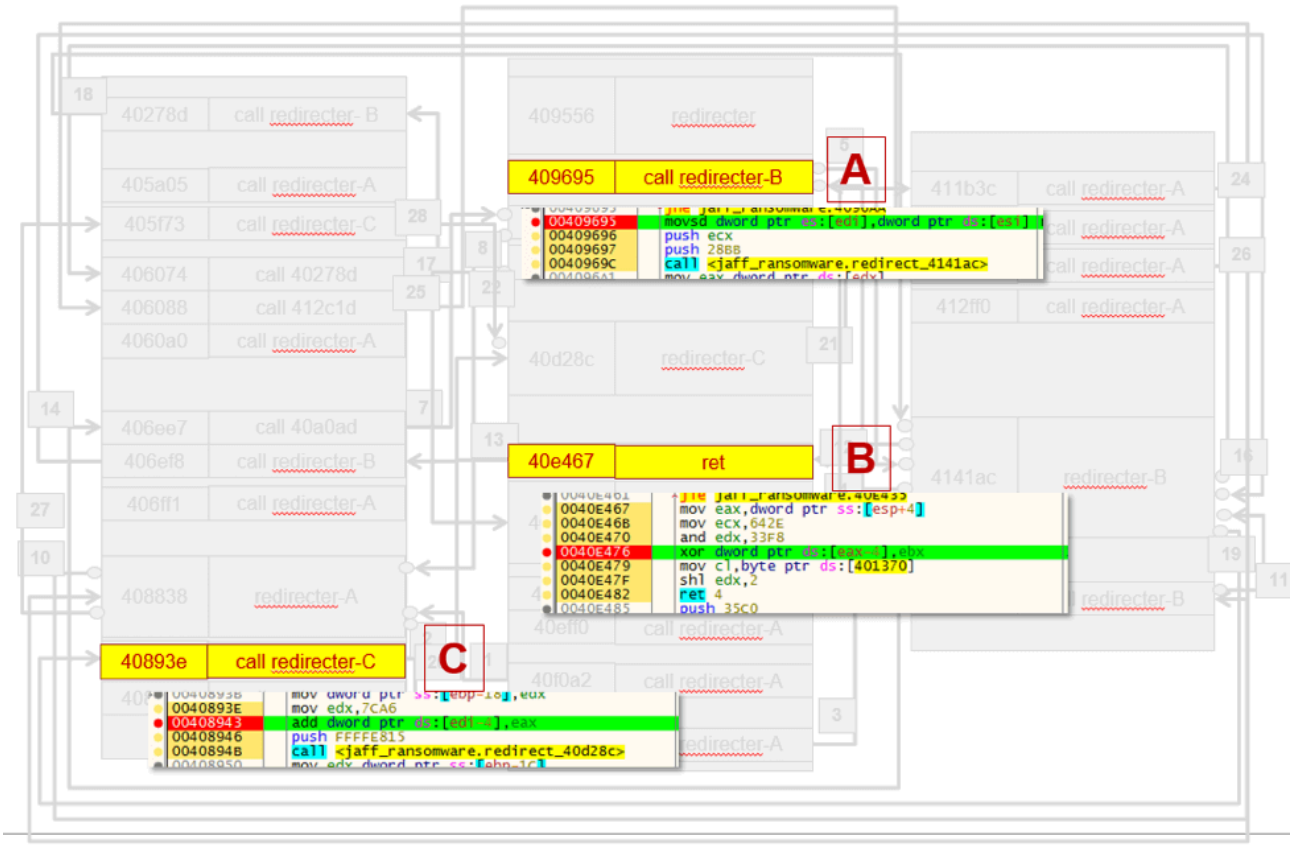

Figure 7

Resolving the APIs

After decrypting the malware code, most of the API names the malware uses are still hidden.

Hiding API names is a malware feature designed to conceal them from an Antivirus scanner. It helps to avoid being detected based on a combination of known APIs used by malware. There are different ways of hiding the APIs—some malware uses encryption, and some uses hashing. The latter is used by Jaff. Following are the steps necessary to resolve the APIs.

Initially, it looks for the “kernel32.dll” string by parsing the PEB (Process Environment Block) structure. It computes the hash of every module name found in PEB and compares it to the hash for “kernel32.dll”. Once it finds a match, it then grabs the location for kernel32.dll and starts resolving the rest of the needed APIs in a similar fashion.

Process Hollowing

After resolving all the needed APIs, Jaff performs process hollowing. This is a malware feature that instead of dropping another executable file and executing it, overwrites part of the original malware code in memory with its new executable code.

In order for Jaff to do process hollowing, it clears the memory blocks of the current process using UnmapViewOfFile API. It then re-allocates the same memory blocks using VirtualAlloc API, and changes its

protection to PAGE_EXECUTE_READWRITE by calling the VirtualProtect API. A series of REPE MOVSB instructions are used to copy the contents of the malicious code to the newly allocated memory blocks.

Wrapper Code

As we have seen so far; the decryption, code redirection, API resolution, and process hollowing are just part of the wrapper code designed to hide the actual malicious binaries. After executing all those codes, the malware now is ready to show its true nature.

Interestingly enough, using the wrapping technique allows you to basically upgrade the wrapper code without the need to upgrade the malicious executable. In this way, you can quickly deploy a new version of the malware that avoids previously used detection parameters.

Let's now look at where the different parts and features of the embedded executable code are located.

Resource Section

The resource section of the malware contains the key block. It also contains the encrypted list of extension names, a download URL link, and the ransom note (See Figure 8).

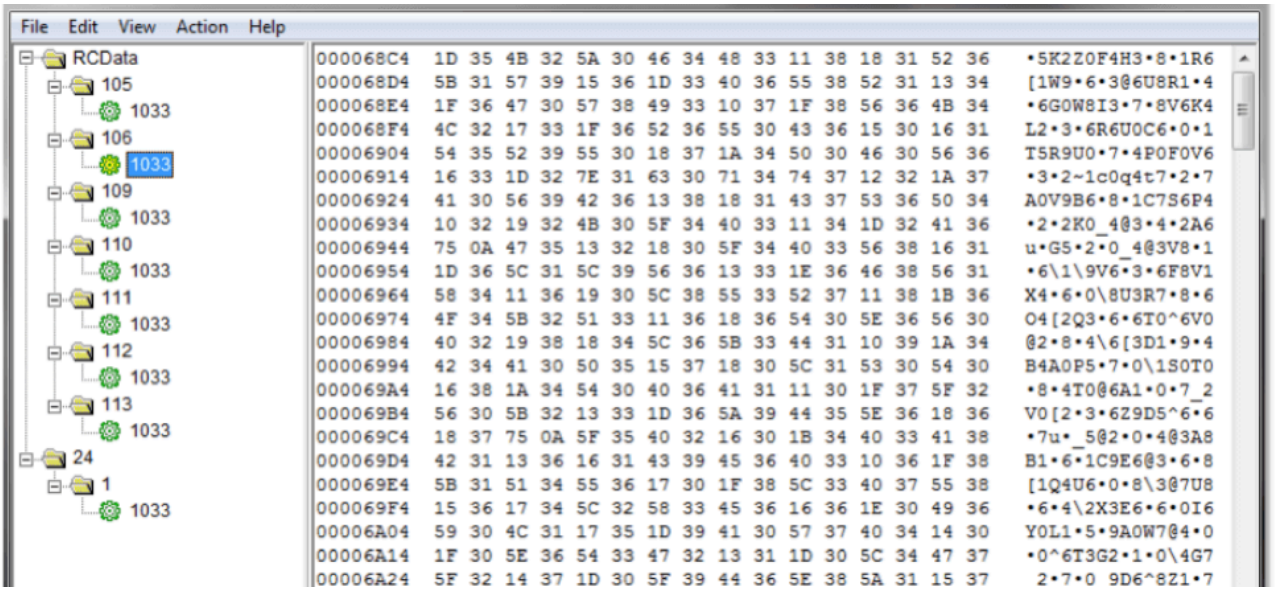

Figure 8

Key Block

The key block is a 260-byte key found in one of the resources. It is used to decrypt the contents of different resources within the section.

Figure 9 shows a snapshot of the code that fetches a resource, the resource that contains the key-block, and the 260 bytes key.

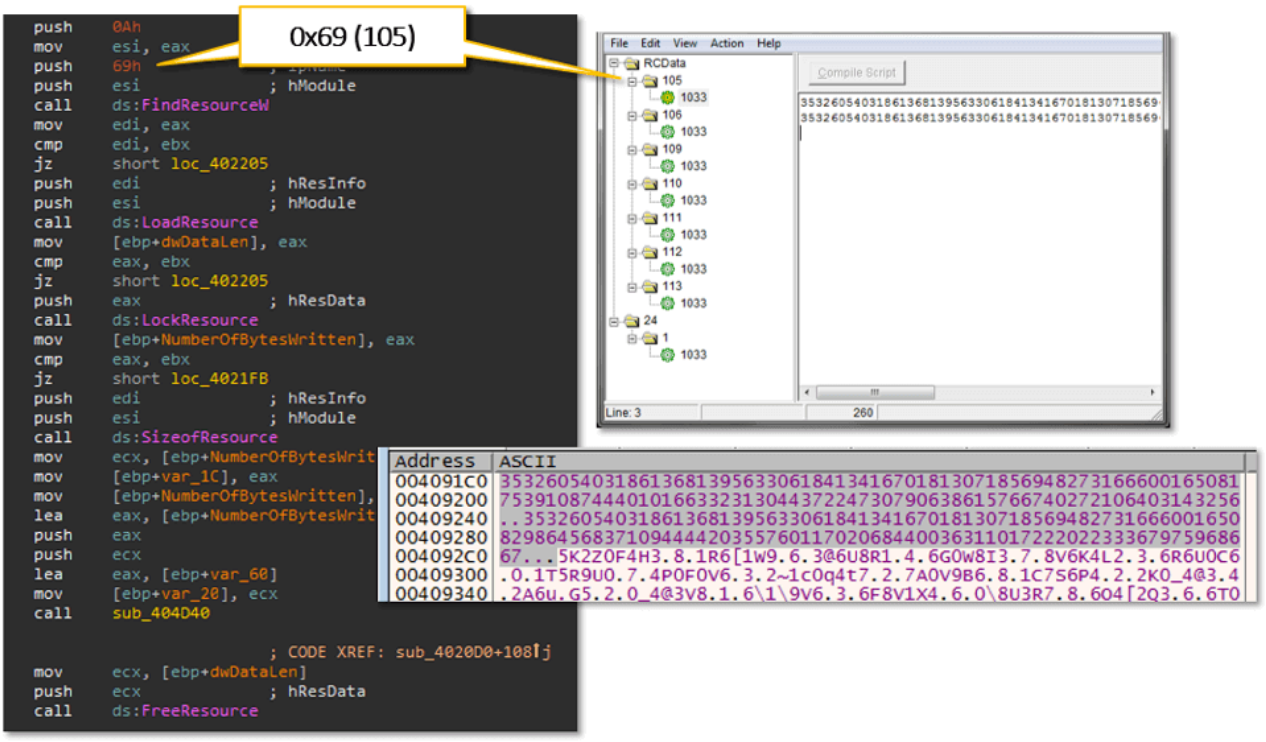

Figure 9

Extension Names

One of the resources contains the decrypted list of extension names. Figure 10 shows the encrypted and decrypted list of extension names of the files that the malware will try to search for and encrypt (See also Figure 11).

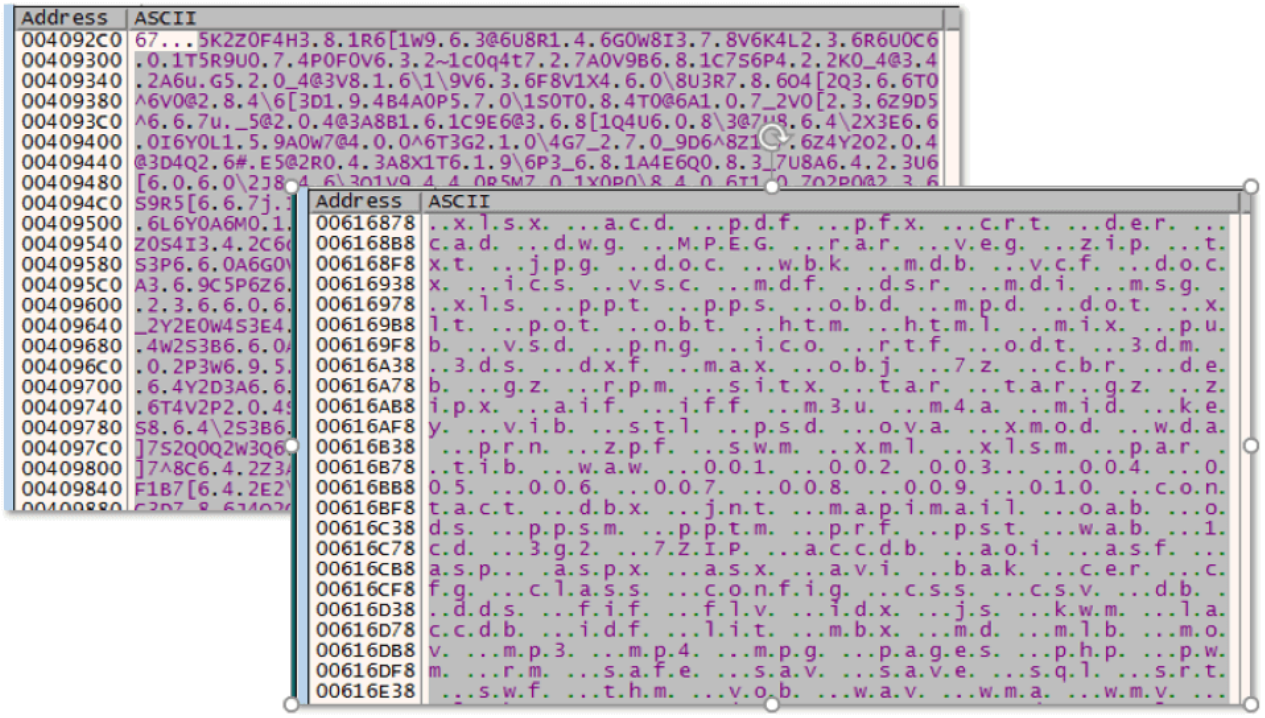

Figure 10

xlsx	deb	cer	dotm	fpx	pdb	rwl	kc2	apj	xls	par	pwm	tga	sxw	st5	qba	gray
acd	gz	cfg	dotx	h	pif	rwz	kdbx	asm	ppt	tib	rm	wps	tlg	st8	qbr	grey
pdf	rpm	class	drw	indd	qed	s3db	kdc	awg	pps	waw	safe	xla	wad	std	myd	gry
pfx	sitx	config	dxb	jpeg	qcow	sd0	kpdx	back	obd	contact	sav	xlam	xlk	sti	ndd	hbk
crt	tar	css	eps	mos	qcow2	sda	lua	backup	mpd	dbx	save	xlm	aiff	stw	nef	ibank
der	tar	csv	fla	nd	rvt	sdf	mdc	backupdb	dot	jnt	sql	xltm	bin	stx	nk	ibd
cad	gz	db	flac	nsd	st7	sqlite	mef	bank	xlt	mapimail	srt	xltx	bmp	sxd	nop	cdr4
dwg	zipx	dds	fxg	nsf	stm	sqlite3	mfw	bay	pot	oab	swf	xlw	cmt	sxg	nrw	cdr5
MPEG	aif	fif	java	nsg	vbox	sqlitedb	mmw	bdb	obt	ods	thm	act	dat	sxi	ns2	cdr6
rar	iff	flv	m	nsh	vdi	sr	mny	bgt	htm	ppsm	vob	adp	dit	sxm	ns3	cdrw
veg	m3u	idx	m4v	odc	vhd	srf	moneywell	bik	html	pptm	wav	al	edb	tex	ns4	ce1
zip	m4a	js	pcd	odp	vhdx	oth	mrw	bpw	mix	prf	wma	bkp	flvv	wallet	nwb	ce2
txt	mid	kwm	pct	oil	vmdk	otp	des	cdr3	pub	pst	wmv	blend	gif	wb2	nx2	cib
jpg	key	laccdb	pl	pas	vmsd	ots	dgc	as4	vsd	wab	xlsb	cdf	groups	wpd	nxl	craw
doc	vib	idf	potm	pat	vmx	ott	djvu	tif	png	1cd	aac	cdx	hdd	x11	nyf	crw
wbk	stl	lit	potx	pef	vmxf	p12	dng	asp	ico	3g2	ai	cgm	hpp	x3f	odb	csh
mdb	psd	mbx	ppam	ptx	3fr	p7b	drf	hdr	rtf	7ZIP	arw	cr2	log	xis	odf	csl
vcf	ova	md	ppsx	qbb	3pr	p7c	dxg	db_journal	odt	accdb	c	dac	m2ts	ycbcra	odg	
docx	xmod	mlb	ps	qbm	ab4	pdd	eml	dc2	3dm	aoi	cdr	dbf	m4p	qbw	odm	
ics	wda	mov	pspimage	sas7bdat	accde	pem	erbsql	dcs	3ds	asf	cls	dcr	mkv	qbx	ord	
vsc	prn	mp3	r3d	say	accdt	plus_muhd	erd	ddoc	dxf	asp	cpi	ddd	ndf	qby	otg	
mdf	zpf	mp4	rw2	st4	ach	plc	exf	ddrw	max	aspx	cpp	design	nvram	raf	ibz	
dsr	swm	mpg	sldm	st6	acr	pptx	ffd	ads	obj	asx	cs	dtd	ogg	rat	iiq	
mdi	xml	pages	sldx	stc	adb	psafe3	fh	agdl	7z	avi	db3	fdb	ost	raw	incpas	
msg	xlsm	php	svg	sxc	srw	py	fhd	ait	cbr	bak	docm	fff	pab	rdb	jpe	

Figure 11

Ransom Notes

Jaff's ransom note is stored in three different formats; html, regular text, and image (bmp). The text and html versions are found in the resource section, while the bmp version is generated using the same text. Figure 12 shows the html version of the ransom note in encrypted and decrypted form, and the location in the resource section where it can be found.

Figure 12

To generate the ransom note in image form, Jaff uses the following combinations of APIs.

CreateStreamOnHGlobal

CreateDCW(DISPLAY)

GetDeviceCaps

SetRect

CreateSolidBrush

FillRect

OleDraw

Figure 13

Figure 13 shows a sample of the ransom note in image form. The decrypt ID is dynamically generated and added to the image. In this particular variant of Jaff ransomware, this image is set as the desktop's wallpaper after the infection.

File Encryption Routine

After all the complex code wrapping and initialization, the main malicious payload that encrypts files is the simplest routine.

To encrypt the file, Jaff searches for files in a given directory, followed by checking if the extension name of the file is found in the list (see Figure 11). Next, it renames the file with a .jaff extension and opens it for encryption. It then encrypts the file using a call to the CryptEncrypt API (see Figure 14).

After all possible files are encrypted, the malware drops the ReadMe.bmp, ReadMe.html, and ReadMe.txt versions of the ransom note in the given directory.

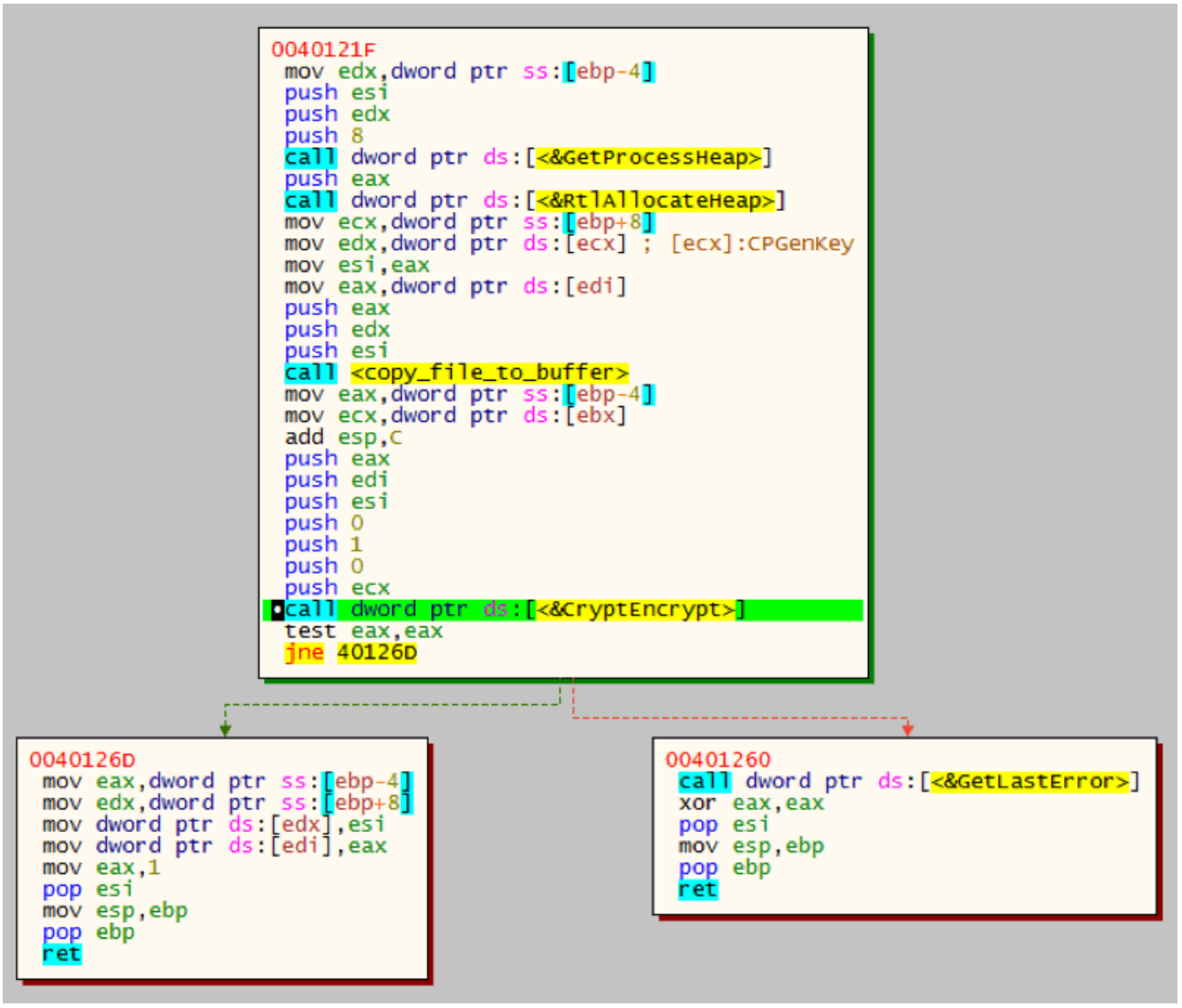

Figure 14

Wrap Up

One of the factors that affects the populariy of a ransomware is the timing of when it is released. Jaff was released at almost the same time as WannaCry, thus killing its dream of stardom in an instant. Or maybe, it was released intentionally at that moment to add stealth to its infection.

Either way, we should always be ready for any malware or ransomware by keeping our defenses regularly updated.

Sha256: 387812ee2820cbf49812b1b229b7d8721ee37296f7b6018332a56e30a99e1092

Detection: W32/Jaff.ED11!tr.ransom

Learn more about FortiGuard Labs and the FortiGuard Security Services portfolio. Sign up for our weekly FortiGuard Threat Brief.

Know your vulnerabilities – get the facts about your network security. A Fortinet Cyber Threat Assessment can help you better understand: Security and Threat Prevention, User Productivity, and Network Utilization and Performance.

Read about the FortiGuard Security Rating Service, which provides security audits and best practices.
