## TokyoX: DLL side-loading an unknown artifact

☐ Iab52.io/blog/tokyox-dll-side-loading-an-unknown-artifact/

During Christmas holidays, Lab52 has been analyzing a sample which loads an artifact that we have decided to refer to as "TokyoX" since no similarities have been found as to any known malware, which we usually detect in open sources. However, we cannot confirm so far that it is indeed a new family of malware.

The first thing we identified was a DLL

(382b3d3bb1be4f14dbc1e82a34946a52795288867ed86c6c43e4f981729be4fc) which had the following timestamps in VirusTotal at the time of the current analysis, and was uploaded from Russia via web site:

Creation Time 2021-12-09 02:46:43 First Submission 2021-12-09 08:48:20 Last Submission 2021-12-09 08:48:20 Last Analysis 2021-12-23 23:38:08

Some antivirus engines tagged the sample as PlugX, but it seems that the attribution might be due to the final payload's loading mechanism: DLL sideloading with an encrypted payload in the same directory. After analyzing the final payload we could not find any similarities with other known samples from PlugX other than the loading TTPs.

This DLL had a related .zip file with the name планирование.zip (translated to as planning.zip). When unzipping, the following files are observed:

💿 Creative.exe
📄 Data
line version.dll

The legitimate file Creative.exe, an encrypted Data file and the version.dll DLL, which implements the loader function for the Data file, and therefore responsible of mapping the "TokyoX".

If we execute it from a path which is not final or the expected by the malware, it replicates to another path and executes from there, which is something it does have in common with some PlugX dll loaders:

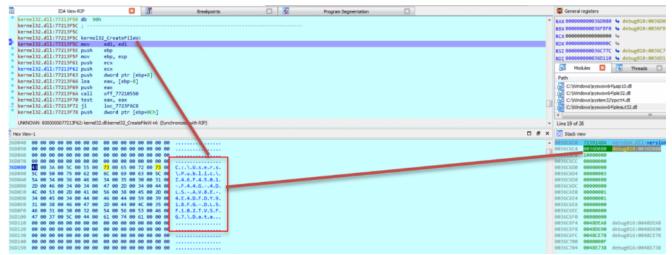
« Disco loc	al (C:)  Users  Public  Z46F4501-F44G-4DLS	S-AV8E-4E4DFDY918FG-	DL5F102TVSFG7	<b>▼</b> <sup>4</sup> 7
🖬 Abrir	Compartir con 🔻 Nueva carpeta			
	Nombre	Fecha de modifica	Тіро	Tamaño
as	💿 123.exe	08/12/2021 1:15	Aplicación	415 KB
0	Data	07/12/2021 17:40	Archivo	97 KB
cientes	🚳 Version.dll	08/12/2021 16:46	Extensión de la apl	79 KB

Once executed, we observe how the netsh.exe process tries to establish connections with port 443 of the IP address 31.192.107[.]187.

In this analysis we will focus on different aspects about the process; from double-clicking the binary 123.exe process (which is a copy of Creative.exe but in another path) to the execution of "TokyoX" already decrypted in memory.

The first thing we observe within the process is how the version.dll library prepares the decryption and the final payload's loading in the remote process:

In fact, we can see how the content of the Data file is read in the code section of version.dll:



If we edit the Data file with a hexadecimal editor we will see their values, which will help us to identify it in memory later (beginning with E3 84):

💌 HxD - [C:\Use	ers\Lu	cas\[	Desk	top\	Data	]											
🔝 Archivo Ec	dición	Bu	iscar	Ve	r Ai	nálisi	s Ex	dras	Ver	ntana	as A	yuda	а				
🗋 👌 🕶 🔲		U.	<u>.</u>	•	••• 1	16		• \	Vind	ows	(ANS	SI)		•	hex		•
🔝 Data																	
Offset(h)	00	01	02	03	04	05	06	07	08	09	OA	0B	oc	OD	0E	OF	Texto decodificado
00000000	E3	84	CE	D2	89	F1	AF	FO	86	E8	13	A5	F8	60	В3	A7	ã"ÎÒ‰ñ¯ðtè.¥ø`³§
00000010	4F	DO	6D	СВ	F7	E2	46	4B	F1	00	ED	BD	9E	CF	E9	40	OÐmË÷âFKñ.í⅔žÏé@
00000020	38	2F	<b>A</b> 8	A5	D3	13	C4	28	29	9E	17	58	BF	CF	50	4D	8/"¥Ó.Ä()ž.X¿ÏPM
00000030	AO	BC	90	ЗF	81	67	F8	04	33	6E	в0	34	14	F9	6A	79	<b>4</b> .?.gø.3n°4.ùjγ
00000040	79	D5	DF	4E	D4	FE	Α4	FF	B6	4B	F4	AA	4D	20	6D	E4	yÕßNÔþ¤ÿ¶Kô≞M mä
00000050	10	15	BF	38	70	C2	B1	F8	BA	8A	4C	A9	46	21	4C	2B	¿8p±ø°ŠL©F!L+
00000060	AF	D9	2B	5A	1E	67	2C	88	C1	85	C7	6E	06	E2	FO	D5	¯Ù+Z.g,^Á…Çn.âðÕ
00000070	B6	D9	C1	49	22	F4	6D	68	5B	3C	91	34	84	22	8E	00	¶ÙÁI"ômh[<`4,,"Ž.
00000080	46	8C	19	9C	52	65	F9	44	EF	E1	Α9	12	FA	00	CE	ЗF	FŒ.œReùDïá©.ú.Î?
00000090	31	ЗD	E7	72	5D	СС	51	F5	E6	DE	13	C2	BE	02	0C	73	1=çr]ÌQõæÞ.¾s
000000A0	50	5A	F7	98	4C	25	AA	61	95	8A	29	64	8D	BE	77	ΟA	PZ÷″L%°a•Š)d.≯w.
ftor roading t	ho fi	ilo f	ron	n di	ek	a d	hild	nrc		e n	otel		vo id	e or	oota	- he	This just_created child

After reading the file from disk, a child process netsh.exe is created. This just-created child process is where several new memory segments will be located (a total of 5, including the final decrypted payload) to decrypt the final "TokyoX" payload. The APIs which were observed for the creation and writing of the remote process are the native APIs NtAllocateVirtualmemory and NtwriteVirtualmemory.

First, it creates two segments: 100Kb where the encrypted payload is located and which comes from the disc, and another one of 4Kb. In the 4Kb segment we observe how the following string is set (which will be the string used for the decrypting process):

	ī	l netsh.exe	= (36	40) (		=000	0 - 0	)x8e1	1000	)	_	_	_		_	_	_	_	R8 FE18
)+3		000000000																	!Up?fOR.D*wN
:h-		00000020			00												00		=
• 1		00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	·····
38		00000040																	
		00000050																	
CR		00000060	~ ~	~ ~		~ ~	~ ~	~ ~			~ ~		~ ~	~ ~	~ ~			~ ~	
In	e	other me	mo	ry s	egi	mei	nt c	DT 10	JUr	D C	con	tain	IS tr	ne i	OIIC	owir	ng (	enc	crypted content, as we see

how it matches the content from Data file on Disk):

#### netsh.exe (3640) (0x8c0000 - 0x8d9000)

00000000	e3	84	ce	d2	89	f1	af	fO	86	e8	13	a5	f8	60	b3	a7		
00000010	4f	<b>d</b> 0	6d	cb	f7	e2	46	4b	f1	00	ed	bd	9e	cf	e9	40	0.mFK@	
00000020	38	2f	<b>a</b> 8	a5	d3	13	c4	28	29	9e	17	58	bf	cf	50	4d	8/()XPM	
00000030	a0	bc	90	3f	81	67	f8	04	33	6e	b0	34	14	f9	6a	79	?.g3n.4jy	
00000040	79	d5	df	4e	<b>d</b> 4	fe	a4	ff	b6	4b	f4	aa	4d	20	6d	e4	yNKM m.	
00000050	10	15	bf	38	70	c2	b1	f8	ba	8a	4c	a9	46	21	4c	2b	8pL.F!L+	
00000060	af	d9	2b	5a	1e	67	2c	88	c1	85	c7	6e	06	e2	fO	d5	+Z.g,n	
00000070	b6	d9	c1	49	22	f4	6d	68	5b	3c	91	34	84	22	8e	00	I".mh[<.4."	
00000080	46	8c	19	9c	52	65	f9	44	ef	e1	a9	12	fa	00	ce	3f	F?	
00000090	31	3d	e7	72	5d	cc	51	f5	e6	de	13	c2	be	02	0c	73	1=.r].Qs	
000000a0	50	5a	f7	98	4c	25	aa	61	95	8a	29	64	8d	be	77	0a	PZL%.a)dw.	
000000ъ0	8a	4d	be	97	a6	e3	02	e1	0e	25	2a	40	e2	32	6b	71	.M%*@.2kq	
00000c0	<b>c</b> 0	<b>c</b> 8	f2	12	b0	3e	29	0b	05	86	45	39	f4	4d	24	5b	>)E9.M\$[	
000000d0	76	e3	c2	28	2b	91	10	8b	42	9b	a4	cf	<b>d</b> 0	1f	80	0d	v(+B	
000000e0	53	75	7f	ab	b9	8e	2e	70	d2	e7	7c	e3	13	ea	9f	83	Sup	
000000f0	06	01	<b>d</b> 8	49	2f	63	95	21	84	f1	65	92	36	ae	3f	97	I/c.!e.6.?.	
00000100	8e	99	1e	38	d2	81	e2	80	53	89	c0	b5	a8	04	a1	94	8S	
00000110	96	92	16	ac	51	e1	d9	2e	bc	45	41	79	70	dd	69	8b	QEAyp.i.	
00000120	ca	f9	57	71	8e	6a	d2	14	49	83	30	b3	b2	6a	bc	a7	Wq.jI.0j	
00000130	ae	0a	15	4d	34	27	86	13	<b>d</b> 6	9e	89	25	50	0b	c6	f4	M4'%P	
00000140	b7	c1	59	bb	89	<b>d</b> 4	9a	56	4f	5b	e5	fb	98	68	2e	ce	YV0[h	
00000150	88	7f	3f	e4	9e	0b	70	78	3d	f4	ed	96	75	0a	84	1e	?px=u	
00000160	61	ea	09	dd	07	ad	1b	ed	d5	57	8f	b5	38	2d	87	c9	aW8	
00000170	3e	44	b5	13	1a	ca	c0	ea	ab	87	bd	ba	9b	e1	3b	6c	>D;1	
00000180	40	e6	26	81	18	fb	7f	80	6d	cd	24	6a	c9	b2	b5	4c	0.sm.\$jL	
00000190	49	c7	de	f2	55	60	49	3b	8f	f8	2f	ee	77	ba	d0	c2	IU`I;/.w	
000001a0																	F.w.hN5y	
	-		_				-			_		-	-					

After the creation of these two segments, a third segment is allocated, where it is loaded the absolute memory addresses from several win32 APIs (VirtualAlloc, LoadLibrary,

GetProcAddress, the home address of the coded payload, etc.) for its later use by the loader:

🗾 🗹	4 🖼		
0000	000072651	000	
0000	000072651	C00 loc_72651C00:	
0000	000072651	COO mov al, [ecx]	
		CO2 inc ecx	
		CO3 test al, al	
0000	000072651	C05 jnz short loc_72651C00	
_			
		<b>T</b>	
🗾 🚄 🖼			
000000072651007	mov	eax, off_7265D008 ; GetProcaddr	ress
0000000072651C00	sub	ecx, edx	
0000000072651C0E	mov	[esp+31A0h+var_300C], eax	
0000000072651C15	mov	eax, off_7265D044 ; LoadLibrary	/
0000000072651C1A		40h	
0000000072651C10		[esp+31A4h+var_3010], eax	
0000000072651C23		<pre>eax, off_7265D048 ; VirtualAllo</pre>	oc 🛛
0000000072651C28		3000h	
0000000072651C20		[esp+31A8h+var_3008], eax	
0000000072651C34		eax, [esp+31A8h+var_3194+8]	
0000000072651C38	push	eax	
0000000072051050	push		

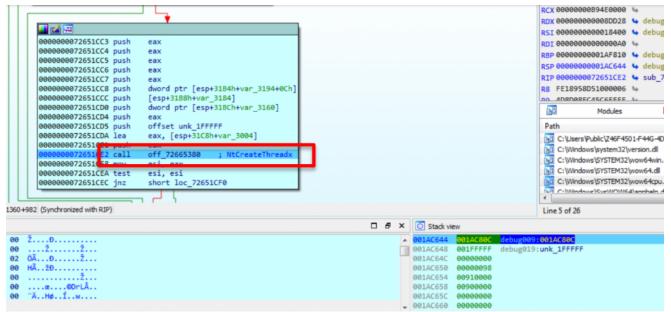
We can notice how the segment will have the memory addresses (starting from 123.exe they are located in netsh.exe segment through the version.dll code):

💷 netsh.e	(36	i40) (	(0x9(	0000	0 - 0	)x901	1000	)									
0000001	17	49	21	77	22	12	21	77	56	18	21	77	00	00	00	00	.I!w".!wV.!w

Then, another segment of 4Kb is created where it loads the code that will decrypt and load the final payload.

💷 netsh.exe (3	640)	(0x91	1000	0 - 0	x911	1000	)									- • •
00000000 5	5 8b	ec	81	ec	14	02	00	00	8b	55	08	33	c0	53	56	VV.3.5V
																W.Z].f
																@=  =
																r.3:3
00000040 84	4 0d	ec	fd	ff	ff	41	81	f9	00	01	00	00	7c	e7	33	A .3
																.3@.f
00000060 84	a 94	35	ec	fe	ff	ff	0f	b6	84	35	ec	fd	ff	ff	03	
00000070 f	8 Of	b6	ca	03	f9	81	e7	ff	00	00	80	79	08	4f	81	y.O.
00000080 c:	£ 00	ff	ff	ff	47	8a	84	3d	ec	fe	ff	ff	88	84	35	G=5
00000090 e	c fe	ff	ff	46	88	94	3d	ec	fe	ff	ff	81	fe	00	01	E=
000000a0 0	0 00	7c	bc	8b	7d	80	33	f6	89	75	fc	39	77	08	76	}.3u.9w.v
																q3.Cy.K
000000c0 f:	f ff	ff	43	8a	8c	1d	ec	fe	ff	ff	Of	b6	d1	03	f2	C
000000d0 8:	1 e6	ff	00	00	80	79	08	4e	81	ce	00	ff	ff	ff	46	F
								_	_							5
																.5M
		_			_	_	_	_	_	_					_	0
																.0E.@.E.;G.r
																]K<.Gj@h
																MqqPj.
																·····}···u^.
																[]zT.
																v#+.3E
																IA.C;ZTr
																}].3.3M.f;B.
00000190 73 000001a0 3																s>.~t).~t#3. 9V.vFNB
000001a0 3		_													42	• · · · · · · · · · · · · · · · · · · ·
Re-read		Writ				to.					er ro			•		Save Close

Finally, the "TokyoX" loader runs from the DLL (version.dll) in netsh.exe through the API NtcreateThreadEx and we see the start of the last page created in the stack:



After the execution of NtCreateThreadEx, as indicated, the loader is initiated in netsh.exe in the segment:

debug039:00910000	55									push	ebp	
debug039:00910001	8B	EC								mov	ebp,	esp
debug039:00910003	81	EC	14	02	00	00				sub	esp,	214h
debug039:00910009	8B	55	<b>0</b> 8							mov	edx,	[ebp+arg_0]
debug039:0091000C	33	<b>C0</b>								xor	eax,	eax
debug039:0091000E										push	ebx	
debug039:0091000F	56									push	esi	
debug039:00910010										push	edi	
debug039:00910011										mov	ebx,	[edx+0Ch]
debug039:00910014										mov		var_8], ebx
debug039:00910017	66	ØF	1F	84	00	00	00	00	00	nop	word	ptr [eax+eax+00000000h]
debug039:00910020												
debug039:00910020										loc_9100	020:	
debug039:00910020		84	05	EC	FE	FF	FF			mov	[ebp+	eax+var_114], al
debug039:00910027										inc	eax	
debug039:00910028			01	00	00					cmp	eax,	100h
debug039:0091002D										jl		loc_910020
debug039:0091002F			04							mov	esi,	[edx+4]
debug039:00910032										xor	ecx,	ecx
debug039:00910034	8B	ЗA								mov	edi,	[edx]
debug039:00910036												

Once the execution is moved to the netsh.exe process, it takes the string located in the initial 4Kb segment, copies it into the stack and replicates it (0x100, 256 bytes) to match the specific block size of 256bytes. In the following screenshots we can observe how the block ends with the string "!Up?" when it reaches the value 0x100 in hexadecimal.

					-	5										
📕 🗹 🛛	<u>7</u>				-											
000000		003	6	_	_	_	_		_	_	_	_	_	1		
000000				ov s	tri	ng	to	stad	ck:					1		
000000							ς, ε	-								
000000							ς ε									
000000						est	-									
000000	000091	003	C mo	v		al	ſe	edx+e	edi	1						
000000	000091	003	Fmo	v				ecx+			¥],	al				
000000						ec				-						
000000	000091	004	7 сп	пр		ec	c, 2	256								
000000	000091	004	D j]	L		sho	ort	mov	str	ing	g_to	_st	ack	c I		
							-		-	_	_	_	_			
0313FD50	00 00	00	00	00	00	60	00	00	00	00	00	00	00	00	00	
0313FD60	00 00	90	00	21	55	70	3F	66	30	52	2E	44	2A	77	4E	!Up?f0R.D*wN
0313FD70	2D 20	) <mark>21</mark>	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	!Up?f0R.D*wN
0313FD80	<mark>21</mark> 55	5 70	ЗF	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	!Up?f0R.D*wN!U
0313FD90	70 3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	p?f0R.D*wN!Up?
0313FDA0	66 30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	f0R.D*wN!Up?f0
0313FDB0	52 28	44	2A	77	4E	2D	2D	<mark>21</mark>	55	70	3F	66	30	52	2E	R.D*wN!Up?f0R.
0313FDC0	44 24	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	D*wN!Up?f0R.D*
0313FDD0	77 48								30							wN!Up?f0R.D*wN
0313FDE0									2E							!Up?f0R.D*wN
0313FDF0									2A							!Up?f0R.D*wN!U
0313FE00	70 3F								4E							p?f0R.D*wN!Up?
0313FE10	66 30								2D							f0R.D*wN!Up?f0
0313FE20	52 28								55							R.D*wN!Up?f0R.
0313FE30									3F							D*wN!Up?f0R.D*
0313FE40									30							wN!Up?f0R.D*wN
0313FE50	2D 20 21 55								2E							!Up?f0R.D*wN
0313FE60	21 55						12	04	05	100	17	10	10	1A	100	!Up?
							41 <sub>-</sub> -		Line a		1 - 4					ion from 00 to FE are fo

After the block is created with the replicated string, the values from 00 to FF are found and used for the decrypting process.

~~																
66	00	90	00	21	55	70	3F	66	30	52	2E	44	2A	77	4E	<pre>!Up?f0R.D*wN</pre>
2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	!Up?f0R.D*wN
21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	!Up?f0R.D*wN!U
70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	p?f0R.D*wN!Up?
66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	f0R.D*wN!Up?f0
52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	R.D*wN!Up?f0R.
44	2A	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	D*wN!Up?f0R.D*
77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	wN!Up?f0R.D*wN
2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	!Up?f0R.D*wN
21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	!Up?f0R.D*wN!U
70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	p?f0R.D*wN!Up?
66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	f0R.D*wN!Up?f0
52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	R.D*wN!Up?f0R.
44	2A	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	D*wN!Up?f0R.D*
77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	wN!Up?f0R.D*wN
2D	2D	21	55	70	25	~	20	50	25		24	77	45	20	20	!Up?f0R.D*wN
21	55	70	3F	00	01	02	03	04	05	06	07	08	09	ØA	ØE	!Up?
DOI:																
9C	ØD	ØE	ØF	10	11	12	13	14	15	16	17	18			18	
	0D 1D									16 26			19	1A	18	·!"#\$%&'()*+
10		1E	1F	20	21	22	23	24	25		27	28	19 29	1A 2A	18 28	
1C 2C	1D	1E 2E	1F 2F	20 30	21 31	22 32	23 33	24 34	25 35	26	27 37	28 38	19 29 39	1A 2A 3A	18 28 38	!"#\$%&'()*+
1C 2C 3C	1D 2D	1E 2E 3E	1F 2F 3F	20 30 40	21 31 41	22 32 42	23 33 43	24 34 44	25 35 45	26 36	27 37 47	28 38 48	19 29 39 49	1A 2A 3A 4A	18 28 38 48	·!"#\$%&'()*+ ,/0123456789:;
1C 2C 3C 4C	1D 2D 3D	1E 2E 3E 4E	1F 2F 3F 4F	20 30 40 50	21 31 41 51	22 32 42 52	23 33 43 53	24 34 44 54	25 35 45 55	26 36 46	27 37 47 57	28 38 48 58	19 29 39 49 59	1A 2A 3A 4A	18 28 38 48 58	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK
1C 2C 3C 4C 5C	1D 2D 3D 4D	1E 2E 3E 4E 5E	1F 2F 3F 4F 5F	20 30 40 50 60	21 31 41 51 61	22 32 42 52 62	23 33 43 53 63	24 34 44 54 64	25 35 45 55 65	26 36 46 56	27 37 47 57 67	28 38 48 58 68	19 29 39 49 59 69	1A 2A 3A 4A 5A	18 28 38 48 58 68	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[
1C 2C 3C 4C 5C 6C 7C	1D 2D 3D 4D 5D 6D 7D	1E 2E 3E 4E 5E 6E 7E	1F 2F 3F 4F 5F 6F 7F	20 30 40 50 60 70 80	21 31 41 51 61 71 81	22 32 42 52 62 72 82	23 33 43 53 63 73 83	24 34 44 54 64 74	25 35 45 55 65 75	26 36 46 56 66	27 37 47 57 67 77	28 38 48 58 68 78	19 29 39 49 59 69 79	1A 2A 3A 4A 5A 6A 7A	18 28 38 48 58 68 78	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk lmnopqrstuvwxyz{  }~.€.,f,,†‡^‰Š<
1C 2C 3C 4C 5C 6C 7C	1D 2D 3D 4D 5D 6D	1E 2E 3E 4E 5E 6E 7E	1F 2F 3F 4F 5F 6F 7F	20 30 40 50 60 70 80	21 31 41 51 61 71 81	22 32 42 52 62 72 82	23 33 43 53 63 73 83	24 34 44 54 64 74 84	25 35 45 55 65 75 85	26 36 46 56 66 76	27 37 47 57 67 77 87	28 38 48 58 68 78 88	19 29 39 49 59 69 79 89	1A 2A 3A 4A 5A 6A 7A 8A	18 28 38 48 58 68 78 88	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk Imnopqrstuvwxyz{  }~.€.,f,,,,,+‡^‰Š< Œ.Ž,.,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1C 2C 3C 4C 5C 6C 7C 8C 9C	1D 2D 3D 4D 5D 6D 7D 8D 9D	1E 2E 3E 4E 5E 6E 7E 8E 9E	1F 2F 3F 4F 5F 6F 7F 8F 9F	20 30 40 50 60 70 80 90 A0	21 31 41 51 61 71 81 91 A1	22 32 42 52 62 72 82 92 A2	23 33 43 53 63 73 83 93 A3	24 34 54 64 74 84 94 A4	25 35 45 55 65 75 85 95 A5	26 36 46 56 66 76 86 96 A6	27 37 47 57 67 77 87 97 A7	28 38 48 58 68 78 88 98 A8	19 29 39 49 59 69 79 89 99 A9	1A 2A 3A 5A 5A 7A 8A 9A AA	18 28 38 48 58 68 78 88 98 AB	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk Imnopqrstuvwxyz{  }~.€.,f,,+‡^%š< E.Ž <sup>c</sup> ?""~™š> æ.žŸ.;¢£¤¥¦§~@≧«
1C 2C 3C 5C 6C 7C 8C 9C AC	1D 2D 3D 4D 5D 6D 7D 8D 9D AD	1E 2E 3E 4E 5E 6E 7E 8E 9E AE	1F 2F 3F 4F 5F 6F 7F 8F 9F AF	20 30 40 50 60 70 80 90 A0 B0	21 31 41 51 61 71 81 91 A1 B1	22 32 42 52 62 72 82 92 A2 B2	23 33 43 53 63 73 83 93 A3 B3	24 34 54 64 74 84 94 A4 B4	25 35 45 55 65 75 85 95 A5 85	26 36 56 66 76 86 96 A6 B6	27 37 47 57 67 77 87 97 A7 B7	28 38 48 58 68 78 88 98 A8 B8	19 29 39 49 59 69 79 89 99 A9 89	1A 2A 3A 4A 5A 6A 7A 8A 9A AA BA	18 28 38 48 58 68 78 88 98 A8 88 B8	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk lmnopqrstuvwxyz{  }~.€.,f,,,,,+‡^&Š< E.Ž''``,,,,,*** æ.ŽŸ`;¢£¤¥¦§`©≧« ,,,,,12)
1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC	1D 2D 3D 4D 5D 6D 7D 8D 9D AD BD	1E 2E 3E 4E 5E 6E 7E 8E 9E AE BE	1F 2F 3F 4F 5F 6F 7F 8F 9F AF BF	20 30 40 50 60 70 80 90 A0 B0 C0	21 31 51 61 71 81 91 A1 B1 C1	22 32 42 52 62 72 82 92 A2 B2 C2	23 33 43 53 63 73 83 93 A3 B3 C3	24 34 54 64 74 84 94 A4 B4 C4	25 35 45 55 65 75 85 95 A5 85 C5	26 36 56 66 76 86 96 A6 B6 C6	27 37 47 57 67 77 87 97 A7 87 67 C7	28 38 58 68 78 88 98 A8 88 68 C8	19 29 39 59 69 79 89 99 A9 89 69 C9	1A 2A 3A 5A 6A 7A 8A 9A AA BA CA	18 28 38 48 58 68 78 88 98 A8 98 AB B8 CB	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk Imnopqrstuvwxyz{  }~.€.,f,,†‡^&Š< E.Ž <sup>(]</sup> "=~""Š> æ.ŽŸ·;¢£¤¥¦§~©≧« ~ <sup>©-°</sup> ± <sup>23</sup> ´µJ·, <sup>1</sup> 2» X%%¿ÀÁÂÃÄÅÆÇÈÉÊË
1C 2C 3C 5C 6C 7C 8C 9C AC BC CC	1D 2D 3D 4D 5D 6D 7D 8D 9D AD 8D CD	1E 2E 3E 4E 5E 6E 7E 8E 9E AE BE CE	1F 2F 3F 4F 5F 6F 7F 8F 9F AF BF CF	20 30 40 50 60 70 80 90 A0 B0 C0 D0	21 31 51 61 71 81 91 A1 B1 C1 D1	22 32 42 52 62 72 82 92 A2 B2 C2 D2	23 33 43 53 63 73 83 93 A3 B3 C3 D3	24 34 54 64 74 84 94 A4 B4 C4 D4	25 35 45 55 65 75 85 95 A5 85 C5 D5	26 36 56 66 76 86 96 A6 B6 C6 D6	27 37 47 57 67 77 87 97 A7 87 67 C7 D7	28 38 58 68 78 88 98 A8 88 C8 D8	19 29 39 49 59 69 79 89 99 89 89 69 09	1A 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA DA	18 28 38 48 58 68 78 88 98 AB 88 BB CB DB	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk Imnopqrstuvwxyz{  }~.€.,f,,+‡^&Š< E.Ž <sup>2</sup> «??•~?™Š> æ.ŽŸ`;¢£¤¥¦§~©²« ~ <sup>©-°</sup> ± <sup>23</sup> µJ. <sup>1</sup> 2» X½X;ÀÁÂÃÄÅÆÇÈÉËË ÌÍÎÏĐÑÒÓÕÕÖרÙÚÛ
1C 2C 3C 5C 6C 7C 8C 9C AC BC CC DC	1D 2D 3D 4D 5D 6D 7D 8D 9D AD 8D CD DD	1E 2E 3E 5E 6E 7E 8E 9E AE BE CE DE	1F 2F 3F 4F 5F 6F 7F 8F 9F AF 8F 0F	20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0	21 31 41 51 61 71 81 81 61 81 C1 D1 E1	22 32 42 52 62 72 82 92 A2 82 C2 D2 E2	23 33 43 53 63 73 83 93 A3 B3 C3 D3 E3	24 34 54 64 74 84 94 84 84 64 C4 D4 E4	25 35 45 55 65 75 85 95 85 85 65 05 E5	26 36 56 66 76 86 96 A6 B6 C6 D6 E6	27 37 47 57 67 77 87 97 87 87 67 C7 D7 E7	28 38 48 58 68 78 88 88 88 88 88 88 88 88 88 88 88 88	19 29 39 49 59 69 79 89 99 89 89 89 09 89 09 59 09 59	1A 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA	18 28 38 48 58 68 78 88 98 A8 98 A8 B8 C8 D8 E	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk Imnopqrstuvwxyz{  }~.€.,f,,,+±^%Š< E.Ž <sup>(3</sup> <sup>(2)</sup> • <sup>-</sup> **š) æ.žŸ·;¢£¤¥¦§*©≧« - <sup>©-°</sup> ± <sup>23</sup> ´µJ·, <sup>1</sup> 2» X%X;ÀÁÂÃÄÅÆÇÈÉËË ÌÍÎÏĐÑÒÓÕÕרÙÚÛ ÜÝÞßàáâãäåæçèéë
1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC CC DC EC	1D 2D 3D 4D 5D 6D 7D 8D 9D AD 8D CD DD ED	1E 2E 3E 4E 5E 6E 7E 8E 9E 8E 8E 0E DE EE	1F 2F 3F 4F 5F 6F 7F 8F 9F 8F 9F 6F DF EF	20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0	21 31 41 51 61 71 81 91 A1 B1 C1 D1 E1 F1	22 32 42 52 62 72 82 92 A2 B2 C2 D2 E2 F2	23 33 43 53 63 73 83 93 A3 83 03 C3 D3 E3 F3	24 34 54 64 74 84 94 A4 84 C4 D4 E4 F4	25 35 45 55 65 75 85 95 85 05 85 C5 55 F5	26 36 56 66 76 86 96 A6 B6 C6 D6 E6 F6	27 37 47 57 67 77 87 97 87 87 67 07 67 F7	28 38 48 58 68 78 88 98 88 88 88 08 88 58 F8	19 29 39 49 59 69 79 89 99 89 99 89 09 09 09 59 59 F9	1A 2A 3A 5A 6A 7A 8A 9A AA BA CA EA FA	18 28 38 48 58 68 78 88 98 A8 98 AB BB CB DB EB FB	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk Imnopqrstuvwxyz{  }~.€.,f,,,+‡^&Š< E.Ž <sup>2</sup> «20~~mš> æ.žŸ.;¢£¤¥¦§~©≧« ~©~°± <sup>23</sup> ´µJ. 12» X%%;ÀÁÂÃĂÅÆÇÈÉËË ÌÍÎÏĐÑÒÓÕÕÖרÙÚÛ ÜÝÞßàáâãäåæçèéë ìíîīðñòóôõö÷øùúû
1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC CC DC EC	1D 2D 3D 4D 5D 6D 7D 8D 9D AD 8D CD DD	1E 2E 3E 4E 5E 6E 7E 8E 9E 8E 8E 0E DE EE	1F 2F 3F 4F 5F 6F 7F 8F 9F 8F 9F 6F DF EF	20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0	21 31 41 51 61 71 81 91 A1 B1 C1 D1 E1 F1	22 32 42 52 62 72 82 92 A2 B2 C2 D2 E2 F2	23 33 43 53 63 73 83 93 A3 83 03 C3 D3 E3 F3	24 34 54 64 74 84 94 A4 84 C4 D4 E4 F4	25 35 45 55 65 75 85 95 85 05 85 C5 55 F5	26 36 56 66 76 86 96 A6 B6 C6 D6 E6 F6	27 37 47 57 67 77 87 97 87 87 67 07 67 F7	28 38 58 68 78 88 98 88 88 88 08 88 08 58 F8	19 29 39 49 59 69 79 89 99 89 99 89 09 09 09 59 59 F9	1A 2A 3A 5A 6A 7A 8A 9A AA BA CA EA FA	18 28 38 48 58 68 78 88 98 A8 98 A8 B8 C8 D8 E	!"#\$%&'()*+ ,/0123456789:; <=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ[ \]^_`abcdefghijk Imnopqrstuvwxyz{  }~.€.,f,,,+±^%Š< E.Ž <sup>(3</sup> <sup>(2)</sup> • <sup>-</sup> **š) æ.žŸ·;¢£¤¥¦§*©≧« - <sup>©-°</sup> ± <sup>23</sup> ´µJ·, <sup>1</sup> 2» X%X;ÀÁÂÃÄÅÆÇÈÉËË ÌÍÎÏĐÑÒÓÕÕרÙÚÛ ÜÝÞßàáâãäåæçèéë

At this point, the loader transforms the 00-FF block with a series of additions combining the replicated string's block with the 00-FF block, as we can see:

	<b>Ý Ý</b>
📕 🛃 🖂	
0000000000910060	
0000000000910060 0	ffset calculate: ; Value from chunk 00 - FF ; inital value edi = 0
0000000000910060 m	ov dl, [ebp+esi+chunk 00 FF]
0000000000910067 m	ovzx eax, [ebp+esi+chunk_pass_repeated] ; Value from chunk string repeated
000000000091006F a	
0000000000910071 💼	ovzx ecx, dl
0000000000910074 a	dd edi, ecx
0000000000910076 a	nd edi, 800000FFh
000000000091007C j	ns short save_new_value_stack
	<b></b>
	00000000091007E dec edi 000000000091007F or edi, 0FFFFFF00h
	00000000000000000000000000000000000000
	00000000010005 Inc 601
	÷ •
I 🖌 🖂	• •
000000000910086	
000000000910086 sa	we_new_value_stack:
00000000910086 mc	<pre>w al, [ebp+edi+chunk_00_FF]</pre>
00000000091008D mc	<pre>w [ebp+esi+chunk_00_FF], al ; overwrite chunk_00_FF</pre>
000000000910094 in	ac esi
00000000000000000000000000000000000000	
	<pre>vv [ebp+edi+chunk_00_FF], dl ; overwrite chunk_00_FF</pre>
000000000910095 mc 000000000910095 mc 00000000091009C cm 0000000009100A2 il	p esi, 100h

The combination of the blue block (in following image) and the 00-FF block (pointed in red in previous image) results in the following block in memory, marked in red in the image:

0313FD60	00	00	90	00	21	55	70	3F	66	30	52	2E	44	2A	77	4E	!Up?f0R.D*wN
0313FD70	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	!Up?f0R.D*wN
0313FD80	21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	!Up?f0R.D*wN!U
0313FD90	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	p?f0R.D*wN!Up?
0313FDA0	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	ЗF	66	30	f0R.D*wN!Up?f0
0313FDB0	52	2E	44	2A	77	4E	2D	2D	21	55	70	ЗF	66	30	52	2E	R.D*wN!Up?f0R.
0313FDC0	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	D*wN!Up?f0R.D*
0313FDD0	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	wN!Up?f0R.D*wN
0313FDE0	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	!Up?f0R.D*wN
0313FDF0	21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	!Up?f0R.D*wN!U
0313FE00	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	p?f0R.D*wN!Up?
0313FE10	66	30	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	f0R.D*wN!Up?f0
0313FE20	52	2E	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	R.D*wN!Up?f0R.
0313FE30	44	2A	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	D*wN!Up?f0R.D*
0313FE40	77	4E	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	wN!Up?f0R.D*wN
0313FE50	2D	2D	21	55	70	3F	66	30	52	2E	44	2A	77	4E	2D	2D	!Up?f0R.D*wN
0313FE6	21	55	70	3F	97	89	F9	2B	37	CΔ	22	57	Δ3	Δ6	78	B4	!Up?—‱é+7Ê"W£¦x´
00201201		22		21	21	22				20	~~	27	~	1.0		04	Topi werre main
0313FE70					85										C2		(#R¡µHp¨D.§Òž
	28	23	52	A1		48	70	A8	18	08	44	9D	A7	D2		9E	
0313FE70	28 47	23 ØF	52 DD	A1 F5	85	48 10	70 F0	A8 EE	18 3F	08 1F	44 8C	9D 11	A7 02	D2 DE	C2	9E AA	(#R¦µHp∵D.§Òž
0313FE70 0313FE80	28 47 24	23 ØF AD	52 DD EC	A1 F5 71	85 C9	48 10 3A	70 F0 74	A8 EE 60	18 3F 7C	08 1F DF	44 8C E0	9D 11 FD	A7 02 F9	D2 DE 87	C2 E4	9E AA 9C	(#R¦µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª
0313FE70 0313FE80 0313FE90	28 47 24 5B	23 ØF AD 8E	52 DD EC 1C	A1 F5 71 9A	85 C9 9F	48 10 3A 4C	70 F0 74 0C	A8 EE 60 7A	18 3F 7C BF	08 1F DF B9	44 8C EØ 96	9D 11 FD D6	A7 02 F9 C3	D2 DE 87 49	C2 E4 12	9E AA 9C ØB	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh'
0313FE70 0313FE80 0313FE90 0313FEA0	28 47 24 58 35	23 ØF AD 8E 54	52 DD EC 1C 01	A1 F5 71 9A B2	85 C9 9F B6	48 10 3A 4C C5	70 F0 74 0C F1	A8 EE 60 7A D4	18 3F 7C BF 0D	08 1F DF B9 98	44 8C EØ 96 BE	9D 11 FD D6 ØA	A7 02 F9 C3 1A	D2 DE 87 49 F6	C2 E4 12 14	9E AA 9C 0B 91	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z¿¹-ÖÃI
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0	28 47 24 58 35 3C	23 ØF AD 8E 54 16	52 DD EC 1C 01 92	A1 F5 71 9A B2 45	85 C9 9F B6 42	48 10 3A 4C C5 38	70 F0 74 0C F1 94	A8 EE 60 7A D4 D9	18 3F 7C BF 0D 03	08 1F DF 89 98 C4	44 8C E0 96 BE 04	9D 11 FD D6 0A D5	A7 02 F9 C3 1A 4E	D2 DE 87 49 F6 6A	C2 E4 12 14 68	9E AA 9C 0B 91 AC	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh' <.'E=8"Ù.Ä.ÕNj…¬ áĐå.ÏØº.QukÍ~*÷À
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0 0313FEC0	28 47 24 5B 35 3C E1	23 ØF AD 8E 54 16 D0	52 DD EC 1C 01 92 E5	A1 F5 71 9A B2 45 81	85 C9 9F 86 42 3D	48 10 3A 4C C5 38 D8	70 F0 74 0C F1 94 BA	A8 EE 60 7A D4 D9 17	18 3F 7C BF 0D 03 51	08 1F DF 89 98 C4 75	44 8C 96 8E 04 6B	9D 11 FD D6 0A D5 CD	A7 02 F9 C3 1A 4E 7E	D2 DE 87 49 F6 6A 2A	C2 E4 12 14 68 85	9E AA 9C 0B 91 AC C0	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh' <.'E=8"Ù.Ä.ÕNj…¬ áĐå.ÏØº.QukÍ~*÷À OçMcÌJ%Ñ™.o³-w,V
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0 0313FEC0 0313FED0	28 47 24 58 35 3C E1 4F	23 ØF AD 8E 54 16 D0 E7	52 DD EC 1C 01 92 E5 4D	A1 F5 71 9A B2 45 81 63	85 C9 9F 86 42 3D CF	48 10 3A 4C C5 38 D8 4A	70 F0 74 0C F1 94 BA 25	A8 EE 60 7A D4 D9 17 D1	18 3F 7C BF 0D 03 51 99	08 1F DF 89 98 C4 75 8F	44 8C 96 8E 04 6B 6F	9D 11 FD D6 0A D5 CD B3	A7 02 F9 C3 1A 4E 7E 2D	D2 DE 87 49 F6 6A 2A 77	C2 E4 12 14 68 85 F7	9E AA 9C 0B 91 AC C0 56	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z;¹-ÖÃI 5T.²BÅñÔ.~¾öh' <.'E=8"Ù.Ä.ÕNj…¬ áĐå.ÏØº.QukÍ~*÷À OçMcÌJ%Ñ™.o³-w,V 3.Y°×UÁŠ¥¤'‹}
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0 0313FEC0 0313FED0 0313FEE0	28 47 24 35 35 30 E1 4F 33	23 ØF AD 8E 54 16 D0 E7 06	52 DD EC 1C 01 92 E5 4D 59	A1 F5 71 9A B2 45 81 63 B0	85 C9 9F 86 42 3D CF CC	48 10 3A 4C C5 38 D8 4A 55	70 F0 74 0C F1 94 BA 25 C1	A8 EE 60 7A D4 D9 17 D1 8A	18 3F 7C BF 0D 03 51 99 A5	08 1F DF 98 C4 75 8F A4	44 8C 96 8E 04 6B 6F 07	9D 11 FD 06 0A D5 CD 83 0E	A7 Ø2 F9 C3 1A 4E 7E 2D 2E	D2 DE 87 49 F6 6A 2A 77 27	C2 E4 12 14 68 85 F7 B8	9E AA 9C 0B 91 AC C0 56 7D	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh' <.'E=8"Ù.Ä.ÕNj…¬ áĐå.ÏØº.QukÍ~*÷À OçMcÌJ%Ñ™.o³-w,V
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0 0313FEC0 0313FED0 0313FEE0 0313FEF0	28 47 24 58 35 3C E1 4F 33 AB 29	23 ØF AD 8E 54 16 D0 E7 06 69 76	52 DD EC 1C 01 92 E5 4D 59 00 79	A1 F5 71 9A B2 45 81 63 B0 CE FC	85 C9 9F 86 42 3D CF CC D7 D8 C7	48 10 3A 4C C5 38 D8 4A 55 5A 50	70 F0 74 0C F1 94 BA 25 C1 1D 5F	A8 EE 60 7A D4 D9 17 D1 8A EB 88	18 3F 7C BF 0D 03 51 99 A5 90 6C	08 1F DF 98 C4 75 8F A4 3B ED	44 8C 96 8E 04 6B 6F 07 93 DC	9D 11 FD 06 0A D5 CD 83 0E 6D 2C	A7 Ø2 F9 C3 1A 4E 7E 2D 2E FA 1E	D2 DE 87 49 F6 6A 2A 77 27 E8 2F	C2 E4 12 14 68 85 F7 B8 88 F2 7F	9E AA 9C 0B 91 AC C0 56 7D 5C A0	<pre>(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh' &lt;.'E=8"Ù.Ä.ÕNj¬ áĐå.ÏØº.Qukí~*÷À OcMcÌJ%Ñ™.o³-w,V 3.Y°×UÁŠ¥¤'&lt;} «i.ÎÛZ.ë.;"múèò\ )vyüCP_^1IÜ,./</pre>
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0 0313FEC0 0313FED0 0313FEF0 0313FEF0	28 47 24 35 3C E1 4F 33 AB 29 26	23 ØF AD 8E 54 16 D0 E7 06 69 76 C8	52 DD EC 01 92 E5 4D 59 00 79 E2	A1 F5 71 9A B2 45 81 63 B0 CE FC 15	85 C9 9F 86 42 3D CF CC D7 D8 C7 30	48 10 3A 4C 55 38 4A 55 5A 50 A9	70 F0 74 0C F1 94 BA 25 C1 1D 5F EF	A8 EE 60 7A D4 D9 17 D1 8A EB 88 5E	18 3F 7C BF 0D 03 51 99 A5 90 6C 19	08 1F DF 98 C4 75 8F A4 3B ED A2	44 8C 96 8E 04 6F 07 93 DC 39	9D 11 FD 06 0A D5 CD 83 0E 6D 2C 32	A7 Ø2 F9 C3 1A 4E 7E 2D 2E FA 1E 6E	D2 DE 87 49 F6 6A 2A 77 27 E8 2F 64	C2 E4 12 68 85 F7 B8 85 F7 B8 85 F7 B8 75 E6	9E AA 9C 0B 91 AC C0 56 7D 5C A0 FE	<pre>(#R;µHp"D.§Òž G.ÝôÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.š¶L.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh' &lt;.'E=8"Ù.Ä.ÕNj¬ áĐå.ÏØº.QukÍ~*÷À OcMcÌJ%Ñ™.o³-w.V 3.Y°×UÁŠ¥¤'&lt;} «i.ÎÛZ.ë.;"múèò\ )vyüCP_1IÜ,./. &amp;Èâ.0©ï^.¢92ndæþ</pre>
0313FE70 0313FE80 0313FE90 0313FE80 0313FE80 0313FE00 0313FE00 0313FE70 0313FF00 0313FF10 0313FF10	28 47 24 35 3C E1 4F 33 AB 29 26 40	23 ØF AD 8E 54 16 D0 E7 06 69 76 C8 05	52 DD EC 1C 01 92 E5 4D 59 00 79 E2 80	A1 F5 71 9A B2 45 81 63 B0 CE FC 15 CB	85 C9 9F 86 42 3D CF CC D7 D8 C7 30 FB	48 10 3A 4C C5 38 4A 55 5A 50 A9 B1	70 F0 74 0C F1 94 BA 25 C1 1D 5F EF 78	A8 EE 60 7A D4 D9 17 D1 8A EB 88 5E 53	18 3F 7C BF 0D 03 51 99 A5 90 6C 19 65	08 1F DF 89 98 C4 75 8F A4 3B ED A2 43	44 8C 96 8E 04 6F 07 93 DC 39 09	9D 11 FD 0A 05 CD 83 0E 6D 2C 32 73	A7 Ø2 F9 C3 1A 4E 7E 2D 2E FA 1E 6E 31	D2 DE 87 49 F6 6A 2A 77 27 E8 2F 64 DA	C2 E4 12 14 68 85 F7 88 85 F7 88 F2 7F E6 34	9E AA 9C 0B 91 AC C0 56 7D 5C A0 FE 95	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.šJL.z;¹-ÖÃI 5T.²BÅñÔ.~¾öh' <.'E=8"Ù.Ä.ÕNj…¬ áĐå.ÏØº.QukÍ~*÷À OçMcÌJ%Ñ™.o³-w,V 3.Y°×UÁŠ¥¤'<} «i.ÎÛZ.ë.;"múèò\ )vyüÇP_^1íÜ,./. &Èâ.0©ī^.¢92ndæþ @.€Ëû±{SeC.s1Ú4•
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0 0313FEC0 0313FED0 0313FEF0 0313FF00 0313FF10 0313FF10	28 47 24 35 3C E1 4F 33 AB 29 26 40	23 ØF AD 8E 54 16 D0 E7 06 69 76 C8 05	52 DD EC 1C 01 92 E5 4D 59 00 79 E2 80	A1 F5 71 9A B2 45 81 63 B0 CE FC 15 CB	85 C9 9F 86 42 3D CF CC D7 D8 C7 30	48 10 3A 4C C5 38 4A 55 5A 50 A9 B1	70 F0 74 0C F1 94 BA 25 C1 1D 5F EF 78	A8 EE 60 7A D4 D9 17 D1 8A EB 88 5E 53	18 3F 7C BF 0D 03 51 99 A5 90 6C 19 65	08 1F DF 89 98 C4 75 8F A4 3B ED A2 43	44 8C 96 8E 04 6F 07 93 DC 39 09	9D 11 FD 0A 05 CD 83 0E 6D 2C 32 73	A7 Ø2 F9 C3 1A 4E 7E 2D 2E FA 1E 6E 31	D2 DE 87 49 F6 6A 2A 77 27 E8 2F 64 DA	C2 E4 12 68 85 F7 B8 85 F7 B8 85 F7 B8 75 E6	9E AA 9C 0B 91 AC C0 56 7D 5C A0 FE 95	<pre>(#R;µHp"D.§Òž G.ÝôÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.šJL.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh' &lt;.'E=8"Ù.Ä.ÕNj¬ áĐå.ÏØº.QukÍ~*÷À OçMcÌJ%Ñ™.o³-w,V 3.Y°×UÁŠ¥¤'&lt;} «i.ÎÛZ.ë.;"múèò\ )vyüÇP_11Ű,./.· &amp;Èâ.0©ï^.¢92ndæþ @.€Ëû±{SeC.s1Ú4• ó&gt;bÿãfF,.ôÓ.†ø%ê</pre>
0313FE70 0313FE80 0313FE90 0313FE80 0313FE80 0313FE00 0313FE00 0313FE70 0313FF00 0313FF10 0313FF10	28 47 24 58 35 20 E1 4F 33 AB 29 26 40 F3 84	23 ØF AD 8E 54 16 D0 E7 06 69 76 C8 05 3E 36	52 DD EC 1C 92 E5 4D 59 00 79 E2 80 62 1B	A1 F5 71 9A 82 45 81 63 80 CE FC 15 CB FF 83	85 C9 9F 86 42 3D CF CC D7 D8 C7 30 FB E3 20	48 10 3A 4C 53 8 8 4A 55 5A 81 66 21	70 F0 74 0C F1 94 BA 25 C1 1D 5F EF 7B 46 C6	A8 EE 60 7A D9 17 D1 8A EB 88 5E 53 82 4B	18 3F 7C BF 0D 03 51 99 A5 90 6C 19 65 8D AF	08 1F DF 89 64 75 8F A4 3B A2 43 F4 AE	44 8C 96 8E 04 6B 6F 07 93 DC 39 02 03 8C	9D 11 FD 0A D5 CD 83 0E 6D 2C 32 73 13 B7	A7 02 F9 C3 1A 4E 7E 2D 2E FA 1E 6E 31 86 58	D2 DE 87 49 6A 2A 77 27 E8 2F 64 DA F8 5D	C2 E4 12 68 85 F7 88 85 F7 E6 34 BD 41	9E AA 9C 0B 91 AC C0 56 7D 5C A0 FE 95 EA 61	(#R;µHp"D.§Òž G.ÝõÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.šJL.z;¹-ÖÃI 5T.²BÅñÔ.~¾öh' <.'E=8"Ù.Ä.ÕNj…¬ áĐå.ÏØº.QukÍ~*÷À OçMcÌJ%Ñ™.o³-w,V 3.Y°×UÁŠ¥¤'<} «i.ÎÛZ.ë.;"múèò\ )vyüÇP_^1íÜ,./. &Èâ.0©ī^.¢92ndæþ @.€Ëû±{SeC.s1Ú4•
0313FE70 0313FE80 0313FE90 0313FEA0 0313FEB0 0313FED0 0313FED0 0313FEF0 0313FF700 0313FF10 0313FF10 0313FF30 0313FF30	28 47 24 58 35 20 E1 4F 33 AB 29 26 40 F3 84	23 ØF AD 8E 54 16 D0 E7 06 69 76 C8 05 3E 36	52 DD EC 1C 92 E5 4D 59 00 79 E2 80 62 1B	A1 F5 71 9A 82 45 81 63 80 CE FC 15 CB FF 83	85 C9 9F 86 42 3D CF CC D7 D8 C7 30 FB E3	48 10 3A 4C 53 8 8 4A 55 5A 81 66 21	70 F0 74 0C F1 94 BA 25 C1 1D 5F EF 7B 46 C6	A8 EE 60 7A D9 17 D1 8A EB 88 5E 53 82 4B	18 3F 7C BF 0D 03 51 99 A5 90 6C 19 65 8D AF	08 1F DF 89 64 75 8F A4 3B A2 43 F4 AE	44 8C 96 8E 04 6B 6F 07 93 DC 39 02 03 8C	9D 11 FD 0A D5 CD 83 0E 6D 2C 32 73 13 B7	A7 02 F9 C3 1A 4E 7E 2D 2E FA 1E 6E 31 86 58	D2 DE 87 49 6A 2A 77 27 E8 2F 64 DA F8 5D	C2 E4 12 68 85 F7 B8 88 F2 7F E6 34 BD	9E AA 9C 0B 91 AC C0 56 7D 5C A0 FE 95 EA 61	<pre>(#R;µHp"D.§Òž G.ÝôÉ.ðî?.ŒÞäª \$ìqŸ:t` ßàýù‡.œ [Ž.šJL.z¿¹-ÖÃI 5T.²BÅñÔ.~¾öh' &lt;.'E=8"Ù.Ä.ÕNj¬ áĐå.ÏØº.QukÍ~*÷À OçMcÌJ%Ñ™.o³-w,V 3.Y°×UÁŠ¥¤'&lt;} «i.ÎÛZ.ë.;"múèò\ )vyüÇP_11Ű,./.· &amp;Èâ.0©ï^.¢92ndæþ @.€Ëû±{SeC.s1Ú4• ó&gt;bÿãfF,.ôÓ.†ø%ê</pre>

On the next step, the loader reads the initial argument, arg0, whose value is 0x900000 and points at the 4Kb block, which stores the absolute addresses to different API from Win32:

💶 🎿 📼	▼
00000000009100A4 mov	edi, [ebp+arg 0] ; Arg0 value = 0x900000
00000000009100A7 xor	esi, esi
00000000009100A9 mov 00000000009100AC cmp	[ebp+var_4], esi [edi+8], esi
000000000009100AC cmp	short loc_910122

After this, the decrypting process for the final payload begins. The decrypting process gets two values from the second block, exchanges and adds them, and the result serves as a final index to recover the element from the second block with which the xor will be achieved through the coded block.

This description of the decryption algorythm has been identified as the **RC4 algorythm**.



After the decryption process, we find a PE binary, as seen in the following image. In this case, the payload does not start with the traditional MZ header but the string "tokyo":

### netsh.exe (3640) (0x8c0000 - 0x8d9000)

		-																
	000000000																	tokyo
-																		
	00000030											00						
	00000040																	!L.!
	00000050											00						
	00000060	00	00	00	00	00	00											
	00000070	00	00	00	00	2e	0d	0d	0a	24	00	00	00	00	00	00	00	\$
	00000080	30	19	ba	86	74	78	d4	d5	74	78	d4	d5	74	78	d4	d5	0txtxtx
	00000090	60	13	d7	d4	7e	78	d4	d5	60	13	d1	<b>d</b> 4	f8	78	d4	d5	`~x`x
	000000a0	60	13	d0	d4	66	78	d4	d5	26	0d	d0	d4	65	78	d4	d5	`fx&ex
	000000b0	26	0d	<b>d</b> 7	d4	65	78	<b>d</b> 4	d5	26	0d	d1	<b>d</b> 4	5e	78	d4	d5	£ex£^x
	00000c0	60	13	d5	d4	7f	78	d4	d5	74	78	d5	d5	0b	78	d4	d5	`xtxx
	000000d0	c1	0d	dd	d4	72	78	d4	d5	c1	0d	2b	d5	75	78	d4	d5	rx+.ux
	000000e0	c1	0d	d6	<b>d</b> 4	75	78	<b>d</b> 4	d5	52	69	63	68	74	78	d4	d5	uxRichtx
	000000f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	00000100	50	45	00	00	4c	01	05	00	17	29	b0	61	00	00	00	00	PEL).a
	00000110	00	00	00	00	e0	00	02	01	0b	01	0e	1d	00	fc	00	00	
	00000120	00	90	00	00	00	00	00	00	b1	59	00	00	00	10	00	00	
	00000130	00	10	01	00	00	00	40	00	00	10	00	00	00	02	00	00	
	00000140	06	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00	
	00000150	00	<b>d</b> 0	01	00	00	04	00	00	00	00	00	00	03	00	40	81	@.
	00000160	00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00	
	00000170	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	
	00000180	50	6b	01	00	78	00	00	00	00	a0	01	00	e0	01	00	00	Pkx
	00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	000001a0	00	b0	01	00	fO	10	00	00	e0	5e	01	00	38	00	00	00	^8
	000001b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	000001c0	00	00	00	00	00	00	00	00	18	5f	01	00	40	00	00	00	
	000001d0	00	00	00	00	00	00	00	00	00	10	01	00	сс	01	00		
	000001e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0	
	000001f0	00	00	00	00	00	00	00	00	2e	74	65	78	74	00	00	0	text
	00000200	08	fa	00	00	00	10	00	00	00	fc	00	00	00	04	00	0	
	00000210																0	
	00000220	2e	72	64	61	74	61	00	00	86	65	00	00	00	10	01	0	.rdatae
	00000230	00	66	00	00	00	00	01	00	00	00	00	00	00	00	00	0	.f
	00000240	00	00	00	00	40	00	00	40	2e	64	61	74	61	00	00	0	@@.data
	00000250	10	14	00	00	00	80	01	00	00	0a	00	00	00	66	01	0	f
	00000260																	
	00000270																	.rsrc
	00000280																	p
	00000290																	@@.reloc
	000002a0																0	r
	000002b0																2	ßВ
	000002-0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Then, we see how it loads the VirtualAlloc absolute address (0x77211856) from the segment previously created:

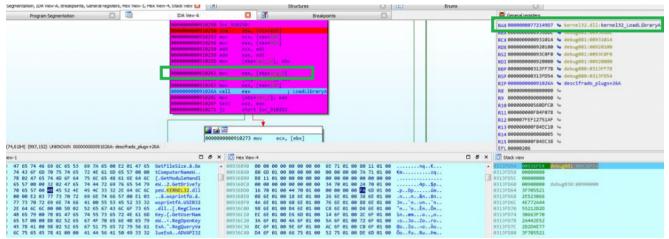
Program Segmentation		IDA View-A		1	Breakpoints	(3)		10	ieneral registers	
								RAX	000000077211856	& kernel32.dll:kernel32 VirtualAl
		0000000000910122 0000000000910122 loc						ROX		depugats reascados
			9101221							debug035:008C0100
		000000000918125 mov	eax, e	die SERVI					00000000000000057	
									030000000000000000	
		000000000091812A pust 00000000091812C pust								debug038:00900000
		0000000000910131 mov		- 41, ecx						debug@80:0313FF78
		000000000918134 mov	a esi, ser	nd ptr [ecx+(=)]						debug060:0313FD4C
		000000000918138 push		tr [ecx+501]						<pre>% descifrado_plugx+138</pre>
		0000000000910138 add	esi, 18						00000000000000000	4
		0000000000010140 add	esi, ec						000000000000000000000000000000000000000	4
		0000000000910142 call							000000000560DFC0	
		000000000918144 mov	edi, es						000000000F84F078	
		00000000000018146 mov	edi, ed	c_C], edi					000007FEF12751AF	
		0000000000918148 jnz		oc 910159					000000000000000000000000000000000000000	
			_						00000000F84EC38	
2,2680) (968,166) UNKINOWN 00000000091011	5: descifrado_plugx+	125							0000206	
1			K 🖸 Hex Vie	a-4			08	C Sta	ck view	
		ža.	· 0088AFC0	00 00 00 00 00 00 0		00 00		· 0313F0		bug001:00010000
7 49 21 77 22 12 21 77 56 18 21 7	7 00 00 00 00	xIlw".lwV.lw	eessafde	00 00 00 00 00 00 0	0 00 00 00 00 00 00 00	00 00			058 00001000	
0 00 00 00 00 00 00 00 00 00 00 00 0	00 00 00 00 00		0058AFE0	00 00 00 00 00 00 00	0 00 00 00 00 00 00 00	00 00		0313F0		
	0 00 00 00 00		00804FF0	6F 68 79 6F 00 0	0 00 00 00 00 00 00 00 00	00 00 tokyo			758 00000000 75C 00000000	
0 00 00 00 00 00 00 00 00 00 00 00 0	0 00 00 00 00		00800010		0 00 40 00 00 00 00 00			0313F0		bug038:00900000

This creates another memory segment in the process netsh.exe with RWX licenses (that of 116Kb) which will be used to load the PE:

0x8c0000	Private: Commit	100 kB		100 kB	100 kB	
0x8e0000	Private: Commit	4 kB	RWX	4 kB	4 kB	
0x900000	Private: Commit	4 kB	RWX	4 kB	4 kB	
0x910000	Private: Commit	4 kB	RWX	4 kB	4 kB	
0x920000	Private: Commit	116 kB	RWX	4 kB	4 kB	

In this new segment, it maps the binary using the virtual addresses as the regular Windows PE loader would do.

Then, it calls the API LoadLibraryA (it has the address since the DLL saved it in the memory segment) of the strings located in the mapped block:



Then it calls GetProcAddress() to get the addresses of certain functions:

Edit Jump Search View Debugger Options Windows Help		N 10		
Library function 📕 Regular function 📕 Instruction 📗 Data 👹 Unexplored 📗	External symbol			
gram Segmentation, IDA View-A, Breakpoints, General registers, Hex View-1, Hex Vie			Enums	
Program Segmentation	IDA View-A	Breakpoints		General registers      RAX 09000000036DFC
	**			RBX 0000000000000 %
				ECX 0000000007211222 & debue001-000160FA EDX 0000000077211222 & kernel32.dll:kernel32_GetProcAddress
000000000318294 loc_91829 00000000318294 pub	82 4X			EDI 000000000000000 S debug031:00920000
00000000018295 push 000000000918295 call e	ebp+wsr_0] dxj GetProcecAddress			80P 00000000313FF78
	scx, [ebp+-ut_10]			RIP 00000000910298 & descifrado_plugx+298
neeeooeoogs182A1 lea e	ebxvecx], eax (bx, ds:0[es1*4]			13 00000000000000 %
000000009102A8 mov e	sax, [ebp+sso_14] scx, [ebx+eax]			R10 00000000560FC0 % R11 00000000F84F078 %
	ncx, ecx phort loc_910200			R12 000007FEF12751AF %
• • •				R14 000000000000005 %
0% (321,6869) (930,166) UNIVIOUN 000000000910298: descfrado_plugx+298				R15 00000000F84EC38 %+ EFL 00000205
tex Wex-1 COP0 08 36 14 36 18 36 1C 36 38 36 3C 36 48 36 4C 36 .6	🗆 🕴 X 🚫 Hex View-4			X Stack view     81157050 77200000 kernel32.dll177200000
C000 08 36 14 36 18 36 17 36 38 36 57 36 68 36 64 36 68 36 67 36 96 C000 58 36 57 36 58 36 57 36 68 36 64 36 68 36 67 36 96 C000 78 36 7C 36 88 36 84 36 88 36 82 36 98 36 94 36 x6	T6X6\6'6d6h616009368A8 80 60 4	00 00 00 00 00 00 06 55 71 01 00 50 11 01 0 01 00 00 00 00 00 00 00 00 00 7A 71 01 0 01 00 00 00 00 00 00 00 00 00 00 00 00 0	0 €mzq	0315/054 003260C debug001:009360C 0313/D58 00300000
COE0 04 39 24 39 44 39 64 39 80 39 9C 39 C4 39 00 00 .9	\$909d9€9æ9Ä9 009368C0 00 00 0	00 00 00 00 00 00 34 70 01 00 24 70 01 0 01 00 44 70 01 00 00 00 00 00 00 00 00 00 00 00 00	e4p\$p	0313FD5C 00000000 0313FD5C 00000000 debug038:00900000
C100 00 00 00 00 00 00 00 00 00 00 00 00		01 00 18 6E 01 00 26 6E 01 00 38 6E 01 0 01 00 60 6E 01 00 76 6E 01 00 38 6E 01 0	0 .n8n8n	0313FD54 3F705521 0313FD68 2E523066
Cline         Construction         Construction <thconstruction< th="">         Construction</thconstruction<>		01 00 84 6E 01 00 C8 6E 01 00 D4 6E 01 0 01 00 E6 6D 01 00 14 6F 01 00 2C 6F 01 0	0 "n"nEnÔn	0313FD6C 45772A44 0313FD70 55212020
C140 00 00 00 00 00 00 00 00 00 00 00 00 0	00936C20 3A 6F	01 00 4A 6F 01 00 5A 6F 01 00 72 6F 01 0 01 00 9E 6F 01 00 AC 6F 01 00 C0 6F 01 0	0 :030Zoro	0313FD74 30663F70 0313FD78 2A442E52
C160 00 00 00 00 00 00 00 00 00 00 00 00 0	00936C48 D4 6F	01 00 66 75 01 00 52 75 01 00 DE 6D 01 0	0 00fuRuÞm	0313FD7C 20204E77
OVAV 00000000193C0F0: debug081:0093C0F0 (Synchronized with RSI)	+ UNINOWN 00000000	09368DC: debug081:009368DC (Synchronized with RBN)		<ul> <li>UNKNOWN 00000000313F050: debug080:0313F050 (Synchronized with RSP)</li> </ul>
🗾 🦽 🔤				
0000000000910294				
0000000000910294 loc_91	.0294:	; CreatePipe		
0000000000910294 push	eax			
0000000000910295 push	[ebp+var_C]			
0000000000910298 call	edx	; GetProcecAd	Idness	
000000000091029A mov	ecx, [ebp+var	• 10]		
000000000091029D inc	esi			
000000000091029E mov	[ebx+ecx], ea	x		
00000000009102A1 lea	ebx, ds:0[esi			
00000000009102A8 mov	eax, [ebp+var	-		
00000000009102AB mov	ecx, [ebx+eax			
00000000009102AE test	ecx, ecx			
00000000009102B0 jnz	short loc 910	280		

Next, the libraries and functions block may be appreciated:

Hex View-1																	
00936E30	75	70	49	6F	66	6F	41	99	E3	99	43	72	65	61	74	65	upInfoA.ã.Create
00936E40						73									60		ProcessAó.Mult
00936E50						54									72		iByteToWideChar.
00936E60						65									60		WideCharToMult
00936E70	69	42	79	74	65	00	83	01	46	69	6E	64	46	69	72	73	<pre>iByte.f.FindFirs</pre>
00936E80	74	46	69	6C	65	57	00	00	8F	01					4E		tFileWFindNe
00936E90	78	74	46	69	6C	65	57	00	65	01	45	78	70	61	6E	64	<pre>xtFileW.e.Expand</pre>
00936EA0	45	6E	76	69	72	6F	6E	6D	65	6E	74	53	74	72	69	6E	EnvironmentStrin
00936EB0	67	73	57	00	BB	04	52	65	6D	6F	76	65	44	69	72	65	gsW.».RemoveDire
00936EC0	63	74	6F	72	79	57	00	00	78	01	46	69	6E	64	43	6C	ctoryWx.FindCl
00936ED0	6F	73	65	00	1F	03	47	65	74	56	6F	6C	75	6D	65	49	oseGetVolumeI
00936EE0						61									43		nformationA.Ï.Cr
00936EF0						69									74		eateFileW.j.GetL
00936F00						6C									69		ogicalDriveStrin
00936F10						01									54		gsW.m.FileTimeTo
00936F20						6D									44		SystemTimeDe
00936F30						69									74		leteFileW.æ.GetS
00936F40						49			6F						65		ystemInfo.ö.Crea
00936F50 00936F60						65 73			00						74 79		teThread(.GetW
00936F70						69									40		<pre>indowsDirectoryAl.FileTimeToLo</pre>
00936F80						6C			69						47		calFileTime.t.Ge
00936F90						41									4E		tProcAddressN.
00936FA0						60									47		GetFileSize.â.Ge
00936FB0						75									00		tComputerNameW
00936FC0						4D									64		{.GetModuleHandl
00936FD0	65	57	00	00	32	02	47	65							54		eW2.GetDriveTy
00936FE0	70	65	57	00	4B	45	52	4E	45	4C	33	32	2E	64	6C	6C	peW.KERNEL32.dll
00936FF0	00	00	E2	03	77	73	70	72	69	6E	74	66	57	00	E1	03	â.wsprintfW.á.
00937000	77	73	70	72	69	6E	74	66	41	00	55	53	45	52	33	32	wsprintfA.USER32
00937010	2E	64	6C	6C	00	00	5B	02	52	65	67	43	6C	6F	73	65	.dll[.RegClose
00937020	4B	65	79	00	7B	01	47	65	74	55	73	65	72	4E	61	6D	Key.{.GetUserNam
00937030						02			67						65		eW RegOpenKey
00937040						02			67						56		ExA. <sup>~</sup> .RegQueryVa
00937050						41									33		lueExAADVAPI32
00937060						00									65		.dlly.HttpOpen
00937070						73									74		RequestWI.Inte
00937080						75									6E		<pre>rnetQueryOptionA</pre>
00937090 009370A0						6E 00									69 65		eFile.É.Internet
009370B0						00									65		OpenW.Ü.Internet
009370C0						74									48		SetOptionA~.Ht
009370D0						72									72		tpQueryInfoWr.
009370E0						6E									57		HttpEndRequestW.
009370F0						70									65		€.HttpSendReques
00937100						00									6E		tExA,.HttpSend
00937110	52	65	71	75	65	73	74	57							74		RequestWInte
00937120	72	6E	65	74	43	6C	6F	73	65	48	61	6E	64	6C	65	00	rnetCloseHandle.
00937130	9C	00	49	6E	74	65	72	6E	65	74	43	6F	6E	6E	65	63	<pre>œ.InternetConnec</pre>
00937140	74	57	00	00	CE	00	49	6E	74	65	72	6E	65	74	52	65	tWÎ.InternetRe
00937150	61	64	46	69	6C	65	00	00	DF	00	49	6E	74	65	72	6E	adFileß.Intern
00937160	65	74	53	65	74	4F	70	74	69	6F	6E	57	00	00	57	49	etSetOptionWWI
00937170						2E									33		NINET.dll.WS2_32
00937180						00									65		.dll0.QueryPer
00937190						6E									72		formanceCounter.
009371A0	18	02	47	65	74	43	75	72	72	65	6E	74	50	72	6F	63	GetCurrentProc

After the correct mapping and having loaded the necessary libraries for its proper functioning, it calls EAX to run the decrypted and mapped payload:

Library function 📕 Regular function 👹 Instruction 📃 Data 🌉 Unexplor	ed 📕 External symbol				
rogram Segmentation, 3DA View-A, Breakpoints, General registers, Hex View-4, 1		Structures			Eruns
2 Program Segmentation 🔲 🕼	DA Ven-A	000000000910285 mov 000000000910285 mov 000000000910280 tost 000000000910280 tost	Dealpoints [ebp+var_8], eax ecx, [eax] ecx, ecx short loc_910250	8	Concert registers           KAX 000000000000000000000000000000000000
		009102C1 mov edx, [ebp+var_	9] 		Ex 000000000011/773 tw (science)011/773 Ex 0000000011/773 tw (science)011/773 Ex 0000000011/2753 tw (science)011/753 Ex 00000000000000 tw (science)1/253 Ex 000000000000000 tw (science)1/253 Ex 000000000000000 tw (science)1/253 Ex 00000000000000000 tw (science)1/253 Ex 000000000000000000000 tw (science)1/253 Ex 000000000000000000000000000000000000
00% (-49,7241) (120,192) UKRIJON 00000009152CC: deudinadu JR	00000	00000012122 (m. 1110) 00000121222 (m. 1110) 0000012122 (m. 1110) 00000122 (m. 1110) 00000122 (m. 1110) 00000122 (m. 1110) 00000122 (m. 1110) 00000122 (m. 1110) 00000122 (m. 1110) 000000000000000000000000000000000		00005850 83 00005850 85 00005850 85 00005850 85 00005880 85 00005880 85 00005880 85 0000580 86 0000580 86 0000580 85 00005910 85 00005910 85	(304)         (3040) </th
Hex Ven-4           35554 85 65 01 10 10 00 00 00 00 00 00 00 00 00 7A 71 01 00 35658 85 11 01 10 00 00 00 00 00 00 00 00 00 00				00003940 6a 00003950 f1 00005970 00 00005970 00 00005980 04 00005940 80 00005940 66 00005940 86 00005940 86 00005940 85 00005940 85 00005940 44	00       00 <td< td=""></td<>
<pre>debug081:009259A2   debug081:009259A3   debug081:009259A8   debug081:009259A8</pre>	call s push d	si ub_927798 word ptr [el ub_92775C	bp-20h]		
debug081:00925980 debug081:00925981	int 3				; Trap to Debugger
debug081:009259B1 debug081:009259B6		ub_92 <mark>5C7B</mark> .oc 92582F			
400000000000000000000000000000000000000		_			
debug081:00925988 debug081:00925988 debug081:00925988 debug081:00925988 debug081:00925988			JBRO	UTI	I N E
debug081:009259BB	sub_92598	B proc near			; CODE XREF: sub_9256FB†j

To summarize, this article goes through the process followed in memory after executing the Creative Cloud application until deploying TokyoX in memory. This DLL sideloading style is often linked to APT groups whose attribution is also linked to China, however being a known technique as it is, we are not able to consider any feasible attribution at the moment.

As reviewed at the beginning of the article, what we have named as "TokyoX" has not been identified as a known malware so far (at least, with the sources that we have).

Additionally, at some point of the analysis we identified a tool used by this group for the creation of version.dll, which pretends to be a Windows DLL located in SysWOW/System32. The string "AheadLib" found among the code of the malicious version.dll drew our attention, and we quickly found two chinese (casually or not) GitHub repositories with the source code of some tool called AheadLib.

https://github.com > strivexjun > AheadLib-x86-x64

#### GitHub - strivexjun/AheadLib-x86-x64: hijack dll Source ...

AheadLib-x86-x64 hijack dll Source Code Generator. support x86/x64 snapshot screen. 不支持 导出符号带有??的方法! NOTE. Pay attention to the generated file header prompt information

Actions · Releases 1 · Notifications · Issues

https://github.com > Yonsm > AheadLib

#### Yonsm/AheadLib: Fake DLL Source Code Generator - GitHub

AheadLlb. Fake DLL Source Code Generator. AheadLlb 2.2.150 - 自动生成一个特洛伊 DLL 分析 代码的工具 ...

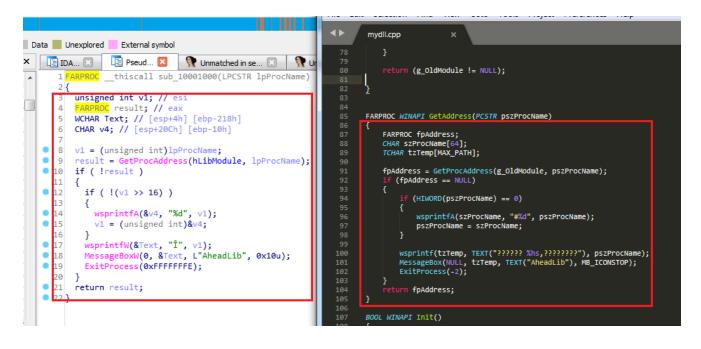
README.md

# AheadLib

Fake DLL Source Code Generator

AheadLib 2.2.150 - 自动生成一个特洛伊 DLL 分析代码的工具

Basically, this tool will allow you to create a C++ source code file, implementing a DLL with the same exported functions as a given DLL. For the purpose of the current analysis we generated a source code file using this tool and giving the legitimate version.dll as input.



In the shown screenshot we can see on the left side the pseudocode generated by IDA Pro while analyzing the malicious version.dll sample. On the right side, we can observe the source code automatically generated by AheadLib using the legitimate version.dll as input. Even though the exported functions are not shown in the previous image, we can appreciate how there is a perfect match between both snippets.

Read the second part of the analysis of the final "TokyoX" RAT and its capacities here.

## IOCs

- 382b3d3bb1be4f14dbc1e82a34946a52795288867ed86c6c43e4f981729be4fc
- 31.192.107[.]187:443

Customers with Lab52's APT intelligence private feed service already have more tools and means of detection for this campaign.

In case of having threat hunting service or being client of S2Grupo CERT, this intelligence has already been applied.

If you need more information about Lab52's private APT intelligence feed service, you can contact us through the <u>following link</u>