Developing Yara Signatures for Malware - Practical Examples

embee-research.ghost.io/practical-signatures-for-identifying-malware-with-yara/

Matthew

October 4, 2023

Last updated on Nov 8, 2023

i i	00 00			
 		LAB_24e4d1410		24e4d143c(j)
	24e4d1413 4c 89 f2	MOV	param_2=>s_%2hhx_24e4dc000,R14	= "%2hhx" 1
	24e4d1416 48 89 f9	MOV		1
	24e4d1419 48 83 c7			1
				1
	24e4d141d e8 7e ff	CALL		undefined FUN_24e4d13a(1
	ff ff			1
	24e4d1422 48 89 £0	MOV		1
	24e4d1425 31 d2	XOR		1
	24e4d1427 48 83 c6		RSI,0x1	1
				1
	24e4d142b 48 f7 f5	DIV		2
	24e4d142e 41 0f b6	MOVZX	EAX, byte ptr [R12 + param_2*0x1]	2:
				2:
	24e4d1433 30 03	XOR	byte ptr [RBX],AL	- 2
	24e4d1435 48 83 c3			2.
				2
	24e4d1439 49 39 f5	CMP	R13,RSI	2
1	24e4d143c 75 d2	JNZ	LAB_24e4d1410	
		TAR 24e4d143e	YDEE[1].	24e4d1400(5)

The purpose of this article is to highlight some practical examples of indicators that can be used for detection using Yara.

The rules are not intended to be performance-optimized. Purely examples of indicators that can be used for detection. <u>Here is a great link if you're interested in performance optimization</u>.

If you wish to try building or testing Yara rules for yourself, I recommend signing up for a free or boosted (\$10USD) account on <u>unpacme</u> (which is what I personally use for testing). Unpacme has an excellent Yara hunting feature that allows you to test on a large collection of malware and legitimate samples.

Lu0Bot SFX Archives

Some recent <u>lu0bot samples</u> are using self-extracting archives (essentially an .exe that unpacks a .zip). I found this sample using an <u>any.run blog</u> and <u>Malware Bazaar</u>.

Inside the sfx/zip file are multiple .dat files that are used to create an exe. The final exe is executed using the randomly named .bat file.

Name
gpfnbokpdbv.dat
eybrqvtjku.dat
eybrqvtjku.dat.1
eybrqvtjku.dat.2
eybrqvtjku.dat.3
🖲 birsjxaakpif.bat
📄 lxeujjd.dat
File Edit View Help
rem 56337563 copy /b eybrqvtjku.dat + eybrqvtjku.dat.1 + eybrqvtjku.dat.2 + eybrqvtjku.dat.3 foixfky.exe foixfky.exe lxeujjd.dat %1%

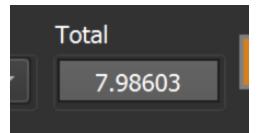
This introduces the following string artifacts inside the initial .exe file.

877	0002286f	0000005 A	PMSCF
878	000228ac	0000000f A	gpfnbokpdbv.dat
879	000228cc	000000e A	eybrqvtjku.dat
880	000228eb	00000010 A	eybrqvtjku.dat.1
881	0002290c	00000010 A	eybrqvtjku.dat.2
882	0002292d	00000010 A	eybrqvtjku.dat.3
883	0002294e	00000010 A	birsjxaakpif.bat
884	0002296f	000000b A	lxeujjd.dat

Which allows this signature to be created. Utilising the .dat.1.dat.2 etc as well as the presence of the .bat file.

Since the file is an archive, it is mostly zipped and compressed. This results in an overall entropy of 7.98, so I added an additional filter math.entropy(0,filesize) > 7. This provides more accuracy at the cost of additional compute resources.

math.entropy is dependent on the math module and is compute intensive. So you are free to remove this jf you run into timeout issues.



I also noticed another sfx artifact of "Win32 Cabinet Self-Extractor". I added this as a string in order to reduce false positives. This probably wasn't necessary, but something you can add to hone in on specific file types. (In this case, sfx files)

I ultimately used this rule to identify more samples. This is not necessarily the most efficient rule but it was able to find additional samples. The definition of "efficient" will depend on your exact situation and compute resources.

```
import "math"
rule win_lu0bot_sfx_packer_oct_2023
{
        meta:
                author = "Matthew @ Embee_Research"
                created = "2023/10/03"
                description = ""
                sha 256 =
"9c84cd037b061c177ee10c45f1f87b3ea05744f1638ab3f348d6b9a3b1cbcfbf"
        strings:
                $s1 = ".dat" ascii
                $s2 = ".dat.1" ascii
                $s3 = ".dat.2" ascii
                $s4 = ".dat.3" ascii
                $s5 = ".bat" ascii
                $t1 = "Win32 Cabinet Self-Extractor" wide
        condition:
                        math.entropy(0,filesize) > 7
                and
                        (all of ($s*))
                and
                        $t1
```

}

Testing on <u>unpacme</u> returned 21 results. With 0 hits for goodware. All of the returned samples appear to be Lu0Bot.

DarkGate XLL Loader

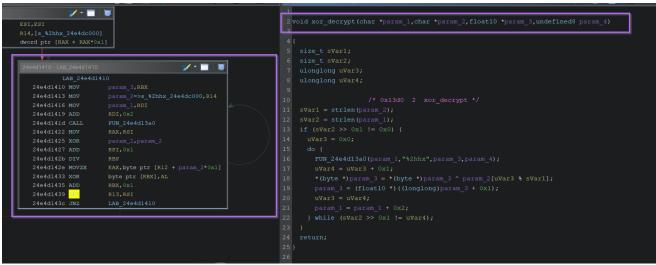
Darkgate has recently utilised XLL (Excel Add-in) files as part of the infection process. An XLL is essentially a DLL file that can be executed by Microsoft Excel.

When opened, the XLL will automatically execute the xlAutoOpen export. Once this export is called, a blob of hex bytes is xor decoded to produce a Wscript command containing C2 information and filenames.

The xlAutoOpen and bytecodes associated with the hex decoding can be used as indicators for a yara rule.

	1 2 ui	ndefined8 xlAutoOpen	(void)		
	3				
	4 {				
		<pre>float10 *_Source;</pre>			
		undefined8 in_R9;			
		undefined8 local_11			
		undefined8 local_11			
		undefined8 local_10	18;		
	10 11	/*	* 0x1450 1 x1#	AutoOpen */	
	12			<pre>\\mshta.exe",L"C:\\Users\\Public\\me.exe",0x1);</pre>	
		_Source = (float10	<pre>*)malloc(0xe9);</pre>		
	14	*(undefined *)((lone	glong)_Source +	0xe8) = 0x0;	
	15	<pre>xor_decrypt("12070c</pre>	:07114e7d57161a0	010c13065b023e19451b5358431c00037f2a060d1a13062a2a16350e060d!	
				51b5d17161c4d533c0601595c0643315f28033c0c17170a140139282c121(
				b06432f39160100062c37390906070f1b0628035a4b0f1116431a11002f5	
				e2a4a4841564554111d320e0a0c0745524243523c5139250616060016280:	
	16		//0a544943424c4i t_key",_Source,i	f7f1c0c17170a145c0618301800515a5e5f5d16172d02150d4d47"	
	17	local 118 = $0x73726$		¹¹ _ ³ ,	
	18	local 110 = $0x5c636$			
	19	$10cal_{108} = 0x20657$	8652e656d;		
		<pre>strcpy((char *)((log</pre>	nglong) &local_1	08 + 0x7),(char *)_Source);	
	21	WinExec((LPCSTR)&lo	cal_118,0x1);		
	22	<pre>free(_Source);</pre>			
	23		n internal erro	or has occurred.","Internal Error",0x10);	
	24 25 }	return 0x0;			
P	Recipe		2 🖿 î	Input + 🗅 🕀 📋	=
F	Fork		⊘ 11	12070c07114e7d57161a010c13065b023e19451b5358431c00037f2a060d1a13062a2a16350e060d5b4714010606361b1 0d061e09567650451b5d17161c4d533c0601595c06d3315f28033c0c17170a140139282c12160d1608504039283c1e171	
	Split de	limiter Merge delimiter		1b174559304b06432f39160100062c37390906070f1b0628035a4b0f1116431a11002f514a56464b57404b43694554404	44a26
	\n	\n	Ignore errors	0213462f1e08564257505443003606001606114343555279085f252f10101717070337150c11090a1139286e45131b004 555d644b12101d010c054b173304161c5b4c584e4a073c190c09075b41	24f52
,	From H	ev.	⊘ 11		
	Delimit Auto	er			
)	KOR		⊘ 11		
	Key secre	et_key UTF8 -	Scheme Standard		
[Nul	l preserving			
				пес 466 📻 1 Тү Raw Bytes	← LF
				Image: Weight of the second	
				Output about:" <script>var b = new ActiveXObject("wscript.shell"); b.run('cmd /c</td><td>::</td></tr><tr><td></td><td></td><td></td><th></th><td>Output 🖬 🗍 🖬</td><td>::</td></tr></tbody></table></script>	

By jumping into the xor_decrypt function. We can observe the primary logic used for performing the decoding. This is usually easy to tell (for simple decoding functions) because the logic will be inside a loop (green arrow on left).



Since decoding logic is often re-used across similar malware (albeit with different decoding keys), we can use the logic itself as an indicator that can be signatured.

To achieve this, we can click the decoding loop and then browse back to the listing window. From here, we can highlight the same instructions that we observed in the previous screenshot.

Observe on the left that we are looking at the same loop.

i 📃	00 00				
		LAB_24e4d1410			
	24e4d1410 49 89 d8	MOV			
	24e4d1413 4c 89 f2	MOV	param_2=>s_%2hhx_24e4dc000,R14		
	24e4d1416 48 89 f9	MOV			
	24e4d1419 48 83 c7	ADD			
	24e4d141d e8 7e ff	CALL			
	ff ff				
	24e4d1422 48 89 f0	MOV	RAX,RSI		
	24e4d1425 31 d2	XOR			
	24e4d1427 48 83 c6	ADD	RSI,0x1		
	24e4d142b 48 f7 f5	DIV			
	24e4d142e 41 0f b6	MOVZX	EAX,byte ptr [R12 + param_2*0x1]		
	24e4d1433 30 03	XOR	byte ptr [RBX],AL		1
	24e4d1435 48 83 c3	ADD			
	24e4d1439 49 39 f5	CMP	R13,RSI		
L -	24e4d143c 75 d2	JNZ	LAB_24e4d1410		
		LAB 240401430	YDE	PF[1] -	24e4d1400(i)

Copying the instructions into a text editor, gives the following. Noting that the highlighted bytecodes are what can be used as a signature.

4d1427 48 83 c6 ADD RSI, 0x1 01 01 01 01 104142b 48 67 f5 DIV RBP 4d142c 41 0f b6 MOVZX EAX, byte ptr [R12 + param_2 *0x1] 04 14 04 14 XOR byte ptr [RBX],AL 4d1433 30 03 XOR byte ptr [RBX],AL 4d1435 48 83 c3 ADD RBX, 0x1 01 01 01 01 01 01 4d1439 49 39 f5 CMP R13, RSI							
24e4d1413 4c 89 f2 MOV param_2 >> s %2hhx_24e4dc000, R14 = "%2hhx" 24e4d1414 88 89 f9 MOV param_1, RDI 24e4d1414 68 7 ADD RDI, Ox2 24e4d142 48 83 c7 ADD RDI, Ox2 24e4d1424 48 83 c6 ADD RSI, Ox1 24e4d1425 31 d2 XOR param_2, param_2 24e4d1424 48 83 c6 ADD RSI, Ox1 01 04 14 XOR byte ptr [R12 + param_2 * 0x1] 4441424 14 04 14 XOR byte ptr [R12 + param_2 * 0x1] 4441424 14 04 14 XOR byte ptr [R12, AL 4041425 14 04 14 XOR byte ptr [R12, AL 4041413 414 24e4d1433 03 3 ADD RBX, 0x1 = "%2hhx" 4444143 48 83 c7 ADD RDI, 0x2 0x1 24e4d1434 48 83 c	24e4d1410	49	89	d8	MOV	param 3 ,RBX	
24e4d1416 48 89 f9 MOV param_1,RD1 mov 24e4d1416 48 83 o7 ADD RD1,0x2 24e4d141 68 7e ff CALL FUN_24e4d13a0 undefined FUN_24e4d13a0(undefine 24e4d1425 31 d2 XOR param_2, param_2 param_2 data 24e4d1425 31 d2 XOR param_2, param_2 data data 24e4d1424 d4 ff ff fb MOV RAX, RSI data 24e4d1425 31 d2 XOR param_2, param_2 data data 24e4d1425 d4 ff ff fb MOV RAX, byte ptr [R12 + param_2<*0x1]	24e4d1413	4c	89	f2	MOV		= "%2hhx"
24e4d1419 46 83 c7 ADD RDI, Öx2 24e4d1414 e6 7e ff CALL FUN_24e4d13a0 undefined FUN_24e4d13a0(undefine 24e4d1424 84 69 f0 MOV RAX, RSI XOR param_2, param_2 24e4d1425 31 69 60 RSI, Öx1 Gata Gata Gata 24e4d1424 61 67 f5 DIV RBP RSI, Öx1 Gata							
02 24e4d141 e6 7e ff Gf CALL FUN_24e4d13a0 undefined FUN_24e4d13a0 (undefine 24e4d1422 48 69 f0 MOV RAX ,RSI XXX 24e4d1427 48 63 c6 ADD RSI ,Ox1 XXX Param_2 , param_2 24e4d1424 48 f7 f5 DIV RBP RXX ,byte ptr [R12 + param_2 * 0x1] XXX 24e4d1435 48 63 c3 ADD RBX ,0x1 RXX RXX 24e4d1435 48 83 c3 ADD RBX ,0x1 RXX RXX 24e4d1435 48 83 c3 ADD RBX ,0x1 RXX RXX 24e4d1435 48 83 c3 ADD RBX ,0x1 RXX RXX 24e4d1435 49 99 68 MOV param_3 ,RBX RXX RXX 24e4d1435 49 99 f5 CMP R13 ,RSI RXX RXX 4d1410 49 89 f6 MOV param_2 , param_2 , param_2 , param_2 , param_2 , p							
24e4d141 e8 7e ff							
ff ff ff ff ff ff 24e4d1425 8 99 f0 MOV RAX , RSI 24e4d1425 31 d2 XOR param_2 , param_2 24e4d1427 48 f7 f5 DIV RBP 24e4d1425 10 fb MOVZX EAX , byte ptr [R12 + param_2 * 0x1] 24e4d1435 48 63 c3 ADD RBX , 0x1 24e4d1435 48 93 c5 CMP R13 ,RS1 24e4d1435 49 39 f5 CMP R13 ,RS1 24e4d1436 49 69 f6 MOV param_3 ,REX 24e4d1436 49 39 f5 CMP R13 ,RS1 24e4d1436 69 f2 MOV param_1 ,NDT 24e4d1436 69 f6 MOV param_2 > s \$2hhx_24e4dc000 ,R14 = "\$2hhx" 4d1413 68 of ff GAD RDT ,No2 RDT ,NO2 4d1414 68 of ff MOV param_2 ,param_2 ,param_2 ,param_2 ,param_2 ,param_2 ,param	24e4d141d		7e	ff	CALL	FUN 24e4d13a0	undefined FUN 24e4d13a0(undefine
24441422 48 89 f0 MOV RXX,RSI 24441425 31 42 48 63 c6 ADD RSI,Ox1 01 04 14 04 14 24441424 48 f7 f5 DIV RBP 24441426 41 0f b6 MOVZX EAX,byte ptr [R12 + param_2 *0x1] 04 14 04 14 24441433 30 03 24441434 49 99 d8 MOV param_2 -> s*2hhr_24e4dcouo,R14 = "%2hhr" 411410 49 89 d8 MOV param_2 -> s*2hhr_24e4dcouo,R14 = "%2hhr" 411416 48 89 f2 MOV param_2 -> s*2hhr_24e4dcouo,R14 = "%2hhr" 411416 48 89 f2 MOV param_2 -> s*2hhr_24e4dcouo,R14 = "%2hhr" 411416 48 89 f0 MOV param_2 -> s*2hhr_24e4dcouo,R14 = "%2hhr" 411416 48 89 f0 MOV param_2, param_2 411416 48 89 f0 MOV RAX,RSI 411416 48 89 f0 MOV RAX,RSI 41142 48 83 c7 10 10 10 10 10 10 10 10 10 10							
24441425 31 d2 XOR param_2,param_2 24441425 48 67 7 f5 DIV RBP 4444142b 48 f7 f5 DIV RBP [12 + param_2 *0x1] 44441435 48 63 c3 ADD RBX,0x1 44441439 49 39 f5 CMP R13,RSI 44441439 49 39 f5 CMP R13,RSI 441414 48 67 f6 CALL FUN_24e4d13a0 undefined FUN_24e4d13a0(undefine ff ff 4142 48 67 f5 DIV RAP 41441 48 67 f5 DIV RAP 41441 48 68 c7 41441 48 68 c7 41441 48 68 c7 41442 48 68 c8 41441 48 68 c7 41442 48 68 c8 41443 68 c8 70 F ff 4142 48 68 c6 4142 48 68 c6 4142 48 68 c7 41442 48 68 c7 41442 48 c8 70 F ff 4142 48 68 c7 41442 48 c8 70 F ff 4142 48 c8 70 F ff 70 F F5 70 F RAP_2 F F F 70 F RAP_2 F F F 70 F RAP_2 F F F RAP_2 F F F RAP_2 F F RAP_2 F F F RAP_2 F F F RAP_2 F F F RAP_2 F F RAP_2 F F F RAP_2 F F F RAP_2 F F RAP_2 F F F RAP_2 F F F RAP_2 F F F RAP_2 F F RAP_2 F F F RAP_2 F F RAP_2 F F F RAP_2 F F RAP_2 F F RAP_2 F F RAP_2 F F F RAP_2 F F F RAP_2 F F F F F F RAP_2 F F F RAP_2 F F F RAP_2 F F RAP_2 F F RAP_2 F F RAP_2 F F F RAP_2 F F RAP_2 F F RAP_2 F F F RAP_2 F F RAP_2 F F F RAP_2 F F F F F F F F F F F F F F F F	24e4d1422			f0	MOV	RAX .RST	
24e4d1427 40 83 c6 ADD RSI, 0x1 - 24e4d1422 48 f7 f5 DIV RBP 24e4d1424 41 0f b6 MOVZX EAX, byte ptr [R12 + param_2 *0x1] 44d44143 30 c3 ADD RBX, 0x1 1 04 14 24e4d1433 49 89 d8 MOV param_3, RBX 24e4d1434 49 89 d8 MOV param_2 => s_%2hhx_24e4dc000, R14 = "%2hhx" 4d1410 49 89 d8 MOV param_1, RDI 24e4d1434 68 99 f2 MOV param_1, RDI 4d1414 68 99 f2 MOV param_2 => s_%2hhx_24e4dc000, R14 = "%2hhx" 4d1416 48 99 f2 MOV param_2, RBX 4d1416 48 99 f2 MOV param_2, RBX 4d1416 48 99 f2 MOV param_2, RBX 4d1416 88 of 7 ADD RDI, 0x2 4 02 4 02 4 02 4 02 4 02 4 04 4 04 4 05 76 ff 4 00 RAX, RSI 4 01 1 0 1 0 1 0 1 0 1 0 1 0 1 0				~ ~			
01 01 24e4d142b 48 67 75 DIV RBP 24e4d142b 48 67 76 MOVZX EAX ,byte ptr [R12 + param_2 * 0x1] 04 14 04 14 04 14 04 14 04 14 01 04 14 24e4d1435 48 83 c3 ADD RBX , 0x1 01 24e4d1439 49 39 f5 CMP R13 , RS1 4d1413 6 89 f2 MOV param_2 -> s %thhx_24eddc000 , R14 = "%2hhx" 4d1413 6 89 f2 MOV param_1 , RDI mov2 4d1414 68 7 ff GALL FUN_24ed13a0 undefined FUN_24ed13a0 (undefine 4d1416 68 7 ff MOV RAX , RSI XXR param_2 , param_2 4d1412 48 63 c6 MOV RAX , RSI XXR param_2 , param_2 4d1412 68 69 f0 MOV RAX , RSI XXR para				c6			
24e4d142b 48 f7 f5 DIV RBP 04 14 MOVZX EAX,byte ptr [R12 + param_2 *0x1] MOVZX EAX,byte ptr [RBX],AL 24e4d1435 48 83 c3 ADD RBX,0x1 01 04 44 39 f5 CMP R13,RS1 24e4d1435 48 99 f2 MOV param_3,REX 41410 49 99 f2 MOV param_3,REX 41414 68 99 f2 MOV param_1,RDT 41414 68 99 f6 MOV param_2 >> s_%2hhx_24e4dc000,R14 = "%2hhx" 41414 68 97 f1 CALL FUN_24e4d13a0 undefined FUN_24e4d13a0(undefine 14141 68 76 f1 CALL FUN_24e4d13a0 undefined FUN_24e4d13a0(undefine 14142 16 69 f5 DIV RBP MOVZX EAX,byte ptr [R12 + param_2 *0x1] Motfield f10 141425 16 f1 f5 DIV RBP MOVZX EAX,byte ptr [R12 + param_2 *0x1] <td< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td></td<>							
24e4d142e 41 0f b6 MOVZX EAX ,byte ptr [R12 + param_2 *0x1] 04 14 VXR byte ptr [RBX],AL 24e4d1433 48 83 c3 ADD RBX ,0x1 01 0 1 49 9 f5 CMP R13 ,RS1 24e4d143 49 89 d6 MOV param_3 ,RBX = "%2hhx" 24e4d143 49 9 f5 CMP R13 ,RS1 = "%2hhx" 4d141 46 89 f2 MOV param_1 ,RD1 = "%2hhx" 4d1414 48 83 c7 ADD RD1 ,0x2 = "%2hhx" 02 02 01 02 02 02 02 02 14142 68 9 f0 MOV RAX ,RS1 WXOR wateran_2 ,param_2 , Param_2 , Param_	24e4d142b		f7	f5	DIV	RBP	
04 14 XOR byte ptr [RBX],AL 24e4d1433 00 30 XOR byte ptr [RBX],AL 24e4d1435 48 83 c3 ADD RBX,0x1 24e4d1439 49 89 d8 MOV param_3,RBX 4d1410 49 89 d8 MOV param_3,RBX 4d1413 689 f9 MOV param_1,RDI 4d1416 48 89 f9 MOV param_1,RDI 4d1419 48 30 c7 ADD RDI,0x2 02 02 7 ADD RDI,0x2 undefined FUN_24e4d13a0 (undefine 16 ff Gf ADD RSI,0x1 undefined FUN_24e4d13a0 (undefine 16 ff MOV param_2, param_2 at1425 at142 at142 at142 at142 at14 at142 at1442 at14 at14							
24e4d1435 48 83 c3 ADD RBX , 0x1 24e4d1435 49 39 f5 CMP R13 , RS1						· ·····	
24e4d1435 48 83 c3 ADD RBX , 0x1 24e4d1435 49 39 f5 CMP R13 , RS1	24e4d1433	30	03		XOR	byte ptr [RBX],AL	
24e4d1439 01 49 99 65 CMP R13,RSI 4d1410 49 89 68 MOV param_3,RBX = "\$2hhx" 4d1413 4c 89 f2 MOV param_1,RDI = "\$2hhx" 4d1419 48 83 c7 ADD RDI,0x2 undefined FUN_24e4d13a0 (undefine 4d1424 48 69 f0 MOV RAX,RSI xOR param_2,param_2 4d1427 48 63 c6 ADD RSI,0x1 - - 4d1424 48 67 f5 DIV RBP MOVZ EAX,byte ptr [R12 + param_2 * 0x1] 4d1424 46 67 f5 DIV RBP 4d1425 48 63 c3 ADD RBX,0x1 4d1435 49 39 f5 CMP R13,RSI 4d1435 49 39 f5 CMP R13,RSI	24e4d1435	48	83	c3			
4d1410 49 89 d8 MOV param_3, RBX 4d1413 4c 89 f2 MOV param_2 >> s%2hhx_24e4dc000, R14 = "%2hhx" 4d1413 4c 89 f2 MOV param_1, RDI 4d1414 48 83 c7 ADD RDI, 0x2 02 02 02 undefined FUN_24e4dl3a0 undefine 4d1424 46 89 f0 MOV RAX, RSI 4d1425 31 d2 XOR param_2, param_2 4d1425 48 83 c3 ADD RSI, 0x1 01 04 14 40 16 MOVZX EAX, byte ptr [R12 + param_2 *0x1] 04 14 41435 48 83 c3 ADD RBX, 0x1 01 04 14 441435 49 39 f5 CMP R13, RSI		01					
<pre>4d1410 49 89 d8 MOV param_3 ,RBX MOV param_2 => s%2hhx_24e4dc000 ,R14 = "%2hhx" 4d1413 4c 89 f9 MOV param_1 ,RDI 4d1419 48 83 c7 ADD RDI, 0x2 0 4d141d e8 7e ff CALL FUN_24e4d13a0 undefined FUN_24e4d13a0(undefine ff ff 4d1422 48 89 f0 MOV RAX,RSI 4d1425 31 d2 XOR param_2, param_2 4d1424 48 83 c6 ADD RSI, 0x1 0 4d1426 48 63 c6 ADD RSI, 0x1 0 4d1426 48 63 c3 ADD RSI, 0x1 0 4d1435 48 83 c3 ADD RBX,0x1 0 4d1439 49 39 f5 CMP R13,RSI</pre>	24e4d1439	49	39	f5	CMP	R13 ,RSI	
02 4d141d e8 7e ff CALL FUN_24e4d13a0 undefined FUN_24e4d13a0(undefine ff ff 4d1422 48 89 f0 MoV RAX,RSI 4d1425 31 d2 XOR param_2,param_2 4d1427 48 83 c6 ADD RSI,0x1 01 04 4d1428 48 f7 f5 DIV RBP 4d142e 41 0f b6 MOVZX EAX,byte ptr [R12 + param_2 *0x1] 04 14 4d1433 30 03 XOR byte ptr [RBX],AL 4d1439 49 39 f5 CMP R13,RSI	24e4d1413 4c 24e4d1416 48	89					
ff ff			f9	1	MOV para	m_1 ,RDI	= "%2hhx"
4d1425 31 d2 XOR param_2, param_2 4d1427 48 83 c6 ADD RSI, 0x1 01 0 0 0 0 4d142b 48 f7 f5 DIV RBP 4d142b 41 0f b6 MOVZX EAX, byte ptr [R12 + param_2 *0x1] 04 4 4 4 4 4 4d1433 30 03 XOR byte ptr [REX], AL 4d1435 48 63 c3 ADD REX ,0x1 01 01 01 01 01 01 4d1439 49 39 f5 CMP R13 ,RSI	02	83	f9 c7	N Z	MOV para ADD RDI	m_1 ,RDI ,0x2	
4d1427 48 83 c6 ADD RSI, 0x1 01 01 01 01 01 04142b 48 f7 f5 DIV RBP 4d142c 41 0f b6 MOVZX EAX, byte ptr [R12 + param_2 *0x1] 04 44 41433 30 03 XOR byte ptr [RBX],AL 4d1435 48 83 c3 ADD RBX, 0x1 01 01 01 01 01 4d1439 49 39 f5 CMP R13, RSI	02 24e4d141d e8 ff	83 7e ff	f9 c7 ff	N Z	MOV para ADD RDI CALL FUN_	m_l ,RDI ,0x2 24e4d13a0	
01 4142b 40 f7 f5 4142b 41 0f b6 04 14 41433 00 3 XOR byte ptr [R12 + param_2 *0x1] 41433 00 3 XOR byte ptr [RBX],AL 41434 49 39 f5 CMP R13,RSI	02 24e4d141d e8 ff 24e4d1422 48	83 7e ff 89	f9 c7 ff	4 2 0 4	MOV para ADD RDI CALL FUN_ MOV RAX	m_l ,RDI ,0x2 24e4d13a0 ,RSI	
4d142e 41 0f b6 MOVZX EAX, byte ptr [R12 + param_2 *0x1] 04 14 03 03 XOR byte ptr [RBX], AL 4d1435 48 83 c3 ADD RBX, 0x1 01 01 CMP R13, RSI	02 24e4d141d e8 ff 24e4d1422 48 24e4d1425 31	83 7e ff 89 d2	f9 c7 ff f0	N 7 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	MOV para ADD RDI CALL FUN_ MOV RAX KOR para	m_l ,RDI ,0x2 24e4d13a0 ,RSI m_2 ,param_2	
04 14 4d1433 30 03 XOR 4d1435 48 83 c3 ADD RBX ,0x1 01 4d1439 49 39 45 CMP R13 ,RSI	02 24e4d141d e8 ff 24e4d1422 48 24e4d1425 31 24e4d1427 48 01	83 7e ff 89 d2 83	f9 c7 ff f0 c6	N 2 0 1 2 2 2	40V para ADD RDI CALL FUN_ 40V RAX KOR para ADD RSI	m_l ,RDI ,0x2 24e4d13a0 ,RSI m_2 ,param_2	
4d1433 30 03 XOR byte ptr [RBX],AL 4d1435 48 83 c3 ADD RBX ,0x1 01 01 01 CMP R13 ,RSI	02 24e4d141d e8 ff 24e4d1422 48 24e4d1425 31 24e4d1427 48 01 24e4d142b 48	83 7e ff 89 d2 83 f7	f9 c7 ff f0 c6 f5		40V para ADD RDI CALL FUN_ 40V RAX KOR para ADD RSI DIV RBP	m_1 ,RDI ,0x2 24e4d13a0 ,RSI m_2 ,param_2 ,0x1	
4d1435 48 83 c3 ADD RBX , 0x1 01 01 CMP R13 , RS1	02 24e4d141d e8 ff 24e4d1422 48 24e4d1425 31 24e4d1427 48 01 24e4d1420 48 24e4d1426 41	83 7e ff 89 d2 83 f7 0f	f9 c7 ff f0 c6 f5		40V para ADD RDI CALL FUN_ 40V RAX KOR para ADD RSI DIV RBP	m_1 ,RDI ,0x2 24e4d13a0 ,RSI m_2 ,param_2 ,0x1	
101 139 f5 CMP R13,RSI	24e4d141d e8 ff 24e4d1422 48 24e4d1425 31 24e4d1427 48 01 24e4d142b 48 24e4d142e 41 24e4d142e 41 24e4d142e 41	83 7e ff 89 d2 83 f7 0f 14	f9 c7 ff f0 c6 f5	N Z Z Z Z I N	40V Para ADD RDI CALL FUN_ 40V RAX KOR Para ADD RSI DIV RBP 40VZX EAX	m_l,RDI ,Ox2 24e4d13a0 ,RSI m_2,param_2 ,Ox1 ,byte ptr [R12 + param_2 *0x1]	
4d1439 49 39 f5 CMP R13,RSI	02 24e4d141d 8 ff 24e4d1422 48 24e4d1425 31 24e4d1425 48 24e4d1424 48 24e4d1424 41 24e4d1433 30	83 7e ff 89 d2 83 f7 0f 14 03	f9 c7 ff f0 c6 f5 b6	N 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	40V para ADD RDI CALL FUN_ 40V RAX KOR para ADD RSI 01V RBP 40VZX EAX	m ⁻ 1,RDI ,0x2 24e4d13a0 ,RSI m ⁻ 2,param_2 ,0x1 ,byte ptr [R12 + param_2 *0x1] .ptr [RBX],AL	
	02 24e4d141d 8 24e4d1422 24e4d1427 24e4d1427 48 24e4d1427 48 24e4d1428 41 04 24e4d1428 30 24e4d1433 30	83 7e ff 89 d2 83 f7 0f 14 03	f9 c7 ff f0 c6 f5 b6	N 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	40V para ADD RDI CALL FUN_ 40V RAX KOR para ADD RSI 01V RBP 40VZX EAX	m_1 ,RDI ,0x2 24e4d13a0 ,RSI m_2 ,param_2 ,0x1 ,byte ptr [R12 + param_2 *0x1] .ptr [RBX],AL	
Remove These	02 24e4d141d e8 ff 24e4d1422 48 24e4d1425 31 24e4d1427 48 01 24e4d1420 48 24e4d1426 41	83 7e ff 89 d2 83 f7 0f	f9 c7 ff f0 c6 f5		40V para ADD RDI CALL FUN_ 40V RAX KOR para ADD RSI DIV RBP	m_1 ,RDI ,0x2 24e4d13a0 ,RSI m_2 ,param_2 ,0x1	

If you're using Notepad++, you can hold alt and select the right column and hit delete. The same can be done with the left column.

24e4d1410 49	8	9 0	d8	MOV	param_3 ,RBX		
24e4d1413 4c				MOV	param 2 => s %2hhx 24e4dc000 ,R14	= "%2hhx"	
24e4d1416 48				MOV	param 1 ,RDI		
24e4d1419 48	8	3 (c7	ADD	RDI , 0x2		
02							
24e4d141d e8	7	e :	ff	CALL	FUN_24e4d13a0	undefined FUN 24e4d13a0(undefine	
ff	f	f					
24e4d1422 48	8	9 :	fO	MOV	RAX ,RSI		
24e4d1425 31				XOR	param_2 ,param_2		
24e4d1427 48		3 (сб	ADD	RSI ,0x1		
01							
24e4d142b 48	f	7 :	f5	DIV	RBP		
24e4d142e 41			b6	MOVZX	EAX ,byte ptr [R12 + param_2 *0x1]		
04							
24e4d1433 30				XOR	byte ptr [RBX],AL		
24e4d1435 48		3 (c3	ADD	RBX ,0x1		
01							
24e4d1439 49	3	9 :	f5	CMP	R13 ,RSI		

This leaves only the bytecodes.

_	-			-	-	
1						
2	49	89	d8			
3	4c	89	f2			
4	48	89	f9			
5	48	83	c7			
6	02					
7	e8	7e	ff			
8	ff	ff				
9	48	89	f0			
10	31	d2				
11	48	83	сб			
12	01					
13	48	f7	f5			
14	41	0f	b6			
15	04	14				
16	30	03				
17	48	83	c3			
18	01					
19	49	39	f5			
20						

Keeping only the opcodes and constant values, leaves this

20											
27	49	??	??								
28	4c	??	??								
29	48	??	??								
30	48	??	??								
31	02										
32	e8	??	??								
33	ff	??									
34	48	??	??								
35	31	??									
36	48	??	??								
37	01										
38	48	??	??								
39	41	??	??								
40	04	??									
41	30	??									
42	48	??	??								
43	01										
44	49	??	??								
45											
				1							

Which results in the following yara rule. I also added in xlAutoOpen to narrow the results down only (ideally) to XLL files.

}

Using unpacme, this resulted in 24 additional samples of the Darkgate XLL loader.

Redline stealer samples have a relatively consistent pattern associated with the method that stores configuration settings. The bytecodes associated with this method can be used to create a yara rule that matches on similar samples.

