MuddyWater Back with DarkBit

V labs.k7computing.com/index.php/muddywater-back-with-darkbit/

By Sudeep

April 21, 2023



Recently, we came across a <u>tweet</u> about DarkBit ransomware. An Iranian APT group, named MuddyWater, is reportedly behind the DarkBit ransomware. In this blog we will explore the ransomware's initial access method, the use of Cobalt Strike and the final ransomware payload.

Initial Access Method

The initial lure was delivered as an ISO file.

00008000 01 43 44 30 30 31 01 00 4C 49 4E 55 58 20 20 20 .CD001..LINUX 00008020 20 20 20 20 20 20 20 20 20 48 52 2D 50 6F 6C 69 63 HR-Polic V 00008050 66 03 00 00 00 03 66 00 00 00 00 00 00 00 00 f.....f..... - -~ ~ ---- -- -- -- -- -

```
Figure 1 – ISOFile
```

The payload included a shortcut file (with a .doc extension) and a zip file.

HR-Update.doc	3/21/2023 8:43 PM	Shortcut	2 KB
HR-Update.zip	3/21/2023 7:37 PM	Compressed (zipp	1,376 KB

Figure 2 – Contents Inside ISO File

The shortcut was using PrintBrm.exe to unpack the HR-Update.zip and run it as shown below. PrintBrm.exe is a windows inbuilt <u>command line tool</u>.

Compatibility	Se	curity	Details	Previous	/ersions	
General	Shortcut	Options	Font	Layout	Colors	
	HR-Update	e.doc				
Target type:	Applica	tion				
Target locatio	n: system	32				
Target:	ols\Prin	ntBrmlexe -	r f %TEMP%	HR-Update.	zip -d :	
Start in: Shortcut key: Run:	None Minimi	zed			•	Figure 3 – Shortcut File
Comment:				(2.1		
Open File	Location	Char	nge Icon	Advance		
	[ОК	Car	ncel	Apply	

cmd.exe /c xcopy .\HR-Update.zip %TEMP% /h /y && PrintBrm.exe -r -f %TEMP%\HR-Update.zip -d %TEMP%\unzip & %TEMP%\unzip\HR-Update.exe

🖃 🔜 cmd.exe	2,560 K	2,616 K	4840 Windows Command Processor	Microsoft Corporation
conhost.exe	10,664 K	13,704 K	4984 Console Window Host	Microsoft Corporation
HR-Update.exe	16,504 K	6,172 K	2128	
MpCmdRun.exe	2,084 K	6,948 K	3844	

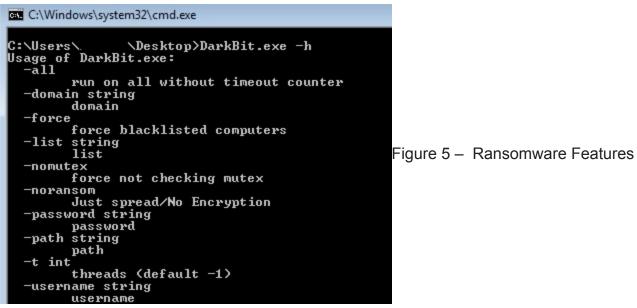


HR-Update.exe was a Cobalt Strike beacon. Cobalt Strike, a penetration testing tool, can also be used by attackers for gaining a foothold in the system. The final ransomware payload is downloaded with the help of Cobalt Strike.

At the time of writing the blog, we were unable to get the exact DarkBit ransomware payload. So we are using another available sample that belonged to the same campaign.

Analysis of Ransomware

This DarkBit ransomware sample is written in GoLang. It contains command line arguments.

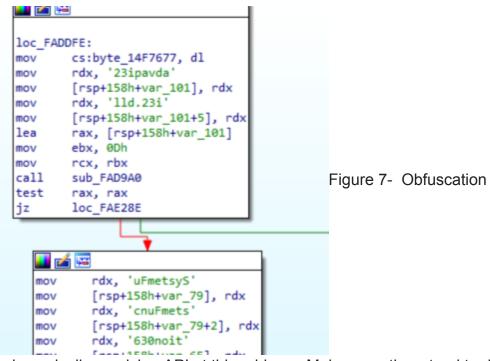


It also contains an inbuilt configuration file as shown in Figure 6.

<pre>}.],. "extensions": {</pre>	
. "msilog": 1,. "1	
. "msilog": 1,. "1 og": 1,. "ldf": 1,.	
"lock": 1,. "theme": 1, . "msi": 1,. "sys"	
. "msi": 1,. "sys"	
: 1,. "wpx": 1,. "	"limitMB": 25,.
cpl": 1,. "adv": 1,.	"limitMB": 25,. "parts": 1,. "eachP
"msc": 1,. "scr": 1,. "key": 1,. "ico":	art": -1. },. {.
"key": 1,. "ico":	"limitMB": 1000,.
1,. "dll": 1,. "ht	"limitMB": 1000,. "parts": 2,. "
a": 1,. "deskthemepack":	eachPart": 12000
<pre>1,. "dll": 1,. "ht a": 1,. "deskthemepack": 1,. "nomedia": 1,.</pre>	<pre>{. "limitMB": 400 0,. "parts": 3,. "eachPart": 10000.</pre>
"msu": 1,. "rtp": 1,.	0,. "parts": 3,.
"msp": 1,. "idx": 1,	"eachPart": 10000.
. "ani": 1,. "386"	1 / "limit
: 1,. "diagcfg": 1,.	MB": 7000,. "parts": 2,. "eachPart": 20000 . },. {.
"bin": 1,. "mod": 1,.	2,. "eachPart": 20000
"ics": 1,. "com":	. },. {.
1 Hhlands 1 Ham	"limitMB": 11000,.
1,. "nip": 1,. "sp 1": 1,. "nis": 1,.	"parts": 3,. "eachPar
"cab": 1,. "diagpkg": 1,	"limitMB": 11000,. "parts": 3,. "eachPar t": 30000. },. {.
. "icl": 1,. "ocx"	"limitMB": 51000,.
: 1,. "rom": 1,. "	"parts": 5,.
prf": 1,. "themepack": 1,	"eachPart": 30000. },.
<pre>. "msstyles": 1,.</pre>	{. "limitMB": 1
"icns": 1,. "mpa": 1,.	000000,. "parts": 3,. "eachPart": 1000000.
"drv": 1,. "cur": 1,	"eachPart": 1000000.
. "diagcab": 1,. "	},. {.
exe": 1,. "cmd": 1,.	"limitMB": 5000000,.

Figure 6 – InBuilt Config

Further analysis revealed that they had obfuscated some dll names like advapi32.dll and functions like SystemFunction036.



Its dynamically resolving API at this address. Malware authors tend to dynamically resolve API to avoid static detections.

T		
💶 🚄 🖼		1
<pre>call sub_454CA0 mov [rcx+0], rsi mov rsp, [rsi+38h] sub rsp, 40h and rsp, 0FFFFFFFFFFFFFF mov [rsp+arg_28], rdi mov rdi, [rdi+8] sub rdi, rdx mov [rsp+arg_20], rdi mov rdi, rbx mov rcx, rbx call rax mov rcx, gs:28h mov rdi, [rsp+arg_28] mov rsi, [rdi+8] sub rsi, [rsp+arg_20] mov [rcx+0], rdi mov rsp, rsi mov [rsp+arg_10], eax retn</pre>	loc_457AAF: sub rsp, 40h and rsp, 0FFFFFFFFFFFFFF6h	Figure 8 –
un averta alles Dia a lettera. A Di		

Dynamically Resolving API

CreateMutexW API is being used to check if an instance of the malware is already running. As can be seen in Figure 5 previously, they are also using multithreading.

As the customary prelude to file encryption, they are using vssadmin.exe to delete all the shadow copies.

2	🕽 Event Propertie	i		
	Event Process	Stack		1
	Date:	4/5/2023 8:48:06 AM		
	Thread:	2608		
	Class:	Process		
	Operation:	Process Create		Figure 9 – Delete
	Result:	SUCCESS		
	Path:	C:\Windows\system32\vssadmin.exe		
	Duration:	0.0000000		
	PID: Command line:	2436 vssadmin.exe del	ete shadows /all /Quiet	

Shadow all

Here they are using SystemFunction036 (documented in MSDN as <u>RtlGenRandom</u>) to generate a random key as shown in Figure 10.



Figure 10 – SystemFunction036

SystemFunction036 is accessed multiple times in the code with varying buffer sizes passed to it.

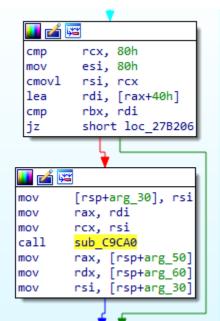


Figure 11 – Call To Dynamically Resolve API and

Then SystemFunction036

Here we can see that the buffer size of 80 is made available for SystemFunction036.

```
rsı, [rax+38n]
mov
        rax, [rsp+50h+arg_20]
mov
        rbx, [rsp+50h+arg_28]
mov
        rcx, [rsp+50h+arg_30]
mov
        rdi, [rsp+50h+arg_38]
mov
        rsi
call
        rdx, [rsp+50h+arg_18]
mov
        rsi, [rdx+38h]
mov
        rax, [rsp+50h+arg_20]
mov
mov
        rbx, [rsp+50h+var_10]
mov
        ecx, 4
        rdi, rcx
mov
                                  Figure 12 – Encrypting Key
call
        rsi
mov
        rdx, [rsp+50h+arg_18]
        rsi, [rdx+30h]
mov
        rax, [rsp+50h+arg_20]
mov
        rbx, [rsp+50h+var_18]
mov
xor
        ecx, ecx
mov
        rdi, [rsp+50h+var_20]
call
        rsi
        [rsp+50h+var_18], rax
mov
        [rsp+50h+var_28], rbx
mov
        [rsp+50h+var_20], rcx
mov
        rdx, [rsp+50h+arg_18]
mov
```

Later-on, the key used for encrypting the files is itself encrypted and attached to the encrypted files.

loc_BDA mov mov bswap mov	B2E: rbp, rsp rax, [rsi] rax [rbp+0], rax	
mov	rdx, 428A2F98D728AE22h	
add	r15, rax	
mov	rax, r12	
add	r15, rdx	
mov	rcx, r12	
ror	rax, 0Eh	Figure 12 Energy stion Algorithm
mov	rdx, r12	Figure 13 – EncryptionAlgorithm
ror	rcx, 12h	
xor	rax, rcx	
mov	rcx, r12	
ror	rdx, 29h	
and	rcx, r13	
xor	rdx, rax	
mov	rax, r12	
not	rax	
add	r15, rdx	
and	rax, r14	
xor	rax, rcx	

From Figure 13, we can see that It's encrypting. It is likely using AES to encrypt the files, as strings related to the same functions can be found elsewhere in this same sample.

FindFirstFileW, FindNextFileW are used to iterate through the file system, to find the appropriate file and then encrypt it.

At that point it was observed that its writing file in chunks and not as a whole. For doing the same it's using SetFilePointerEx API to move the file pointer to a specific address.

	000000007705AEFF	90	nop				
	000000007705AF00	48:83EC 28	sub rsp.28	SetFilePointerEx	RAX	00000007705AF00	"lfi(è\v" &"lfi(è\v"
•	000000007705AF04	E8 0600000	call <jmp.&setfilepointerex></jmp.&setfilepointerex>		RBX	000000000037600	å"Iji(e\v"
•	000000007705AF09	48:83C4 28	add rsp,28		RCX	0000000000000FC	.u. =
•	000000007705AF0D	C3	ret		RDX	00000000002920	
•	000000007705AF0E	90	nop		RBP	000000C000059758	&"0"\x05"
•	000000007705AF0F	90	nop		RSP	000000028E9F8F8	
•	000000007705AF10	90	nop		RSI	000000028E9F900	
٠	000000007705AF11	90	nop		RDI	000000028E9F930	
•	000000007705AF12	90	nop				
•	000000007705AF13	90	nop		R8	000000C0000598D8	
•	000000007705AF14	 FF25 1E270900 	jmp qword ptr ds:[<&SetFilePointerEx>]	JMP. &SetF1TeP01nterEx	R9	0000000000000001	
	000000007705AF1A	90 90	nop		R10	0000000000000000	
	000000007705AF1B 000000007705AF1C	90	nop		R11	000000000000000	
	000000007705AF1D	90	nop		R12	000000C000059980	&""ž\x05"
	000000007705AF1E	90	nop		R13	00000000000000000	
	000000007705AF1F	90	000		R14	000000C00002D860	
	000000007705AF20	¥ EB 06	jmp <jmp.&filetimetosystemtime></jmp.&filetimetosystemtime>	FileTimeToSystemTime	R15	FFFFFFFFFFFFFFFFF	
	000000007705AF22	90	nop				
	000000007705AF23	90	nop		0.70	000000077054500	diamalla FattilaDaintantus
	000000007705AF24	90	nop		4		•
	000000007705AF25	90	nop		-		
•	000000007705AF26	90	nop		. Defaul	t (x64 fastcall)	🔻 5 😒 🖾 Unlocked
•	000000077054537	00	Loop		1: r	cx 0000000000000FC	
-	4			ŀ		dx 000000000002920	
000028E9F8	3F8					8 000000C0000598D8	
						9 0000000000000001	
					5: [rsp+28] 000000000000000	0
000007705	AF00 kernel32.dll:	AF00 #A700 <setfilepoi< td=""><td>nterEx></td><td></td><td></td><td></td><td>*</td></setfilepoi<>	nterEx>				*

Figure 14 – SetFilePointerEx

It's then using the WriteFile API.

48:81C4 50010000	movq xmm2,r8 movq xmm3,r9 call rax add rsp,150		RBP RSP RSI	000000C00005D470 00000000008EF820 0000000008EF820	&"HÔ\x05"	
59 48:8941 18 66:0FD641 20 6548:882C25 30000000 8847 68 48:8941 28	<pre>pop rcx mov qword ptr ds:[rcx+16],rax movq qword ptr ds:[rcx+20],xmm0 mov rdi,qword ptr ds:[rd]+68] mov qword ptr ds:[rcx+28],rax</pre>	rdi+68:"DøŽ"	RDI <u>88</u> 89 810	00000000008EF850 0000000000007E0 000000000000000000000	L' '	
C3 48:83EC 30 48:C7C1 F4FFFFF 48:890C24 48:8805 C2872F00	ret sub rsp,30 mov rcx,FFFFFFFFFFFFF mov qword ptr ss:[rsp],rcx mov rax,qword ptr ds:[*&]MP.&GetStdHan	a 00000000301518+" ^{"+} +"	R11 R12 R13 R14 R15	00000000000000000000000000000000000000		
FFD0 48:89C1 48:890C24 48:8D15 9AD75100	<pre>mov qword ptr ds:[</pre>	00000000005E6F00:"runtime: signa	RTP	000000000000000000000000000000000000000	abr 00000000009710	r F

Figure 15 – WriteFile

All these functions are called one after another, till all the files are encrypted.

After encrypting the file, the key is stored at the end of the file.

Offset	0	1	2	3	4	- 5	6	- 7	8	- 9	A	В	C	D	E	F	Ascii
00002D90	09	2A	92	7D	B4	ΑE	BD	9D	D9	77	Β7	40	86	E8	0F	FD	.*´}´®½ Ù̈w @∎è©ý
00002DA0	63	27	Α7	OD	21	89	E6	6F	DB	F9	Α9	20	49	50	B6	C9	c'S.!∎æoÛù©_IP¶Ĕ
00002DB0	E9	ЗE	0C	2B	24	8F	0E	ΑB	C7	59	FE	CF	1Å	EΒ	7B	2A	é>∎+\$ ♬‹‹ÇYþÏ→ë{ *
00002DC0	B2	2F	93	73	6B	11	15	8A	CD	7D	5B	0E	68	20	Α5	4 A	²∕ [sk ∢ ∔∎] }[♬h.¥J
00002DD0	8E	CF	C4	37	97	В9	3B	D8	CE	Α4	39	11	24	1D	11	36	ĮÏÄ7 I ¹;ØÎ¤9⊲\$ ⊲6
00002DE0	D2	BF	98	76	DA	C5	6A	2C	04	51	90	53	F2	Α1	DA	15	Ò¿∎vŰÅj,⊐Q SòiŰ⊥
00002DF0	A6	66	19	60	36	25	08	25	28	<u>9A</u>	Α8	64	FD	B2	02	20	f -`6%⊡%(``dý²-
00002E00	82	2F	E9	33	2E	28	B5	E8	52	FD	<u>A4</u>	D1	BD	69	21	FC	Į∕ė́3.(µèRý¤N½i!ü
00002E10	C2	F5	Α7	B7	00	10	15	63	66	5F	<u>7C</u>	82	85	41	B9	8E	ÅõS·∎+∸cf_ ∎∎Ź∎
00002E20	FO	B8	69			B7		26	4F	8D	E5	85	41	98	44	91	ð,ifè∙ , &O å∎A∎D′
00002E30	44	41	52	4B	42	49	54	5F	45	4E	43	52		50	54	45	DARKBIT_ENCRYPTE
00002E40	44	5F	46	49	4C			7C	06		02		AC		C8	F8	D_FILES]}⊢∥h^¬∥Èø
00002E50	35	4B	FA	53	CF		11	CB	7B	6A	2D	76	44	E2	43	2F	5KúSI#∢É{j−vJåC∕
00002E60	EE	B6	29	DO	E4		D2		DE	2F	BB	D4	52	64	F8	24	î¶)ĐäļÒ}Þ∕≫ÔRdø\$
00002E70	DA	4E	A4	76			5E		F4	40_{D2}	67	21	53	6E	36	F2	ŰN¤vBŸ^∎ô@g!Sn6ò
00002E80	06	33 4A	2C	B2 73	D1 E6		$^{\rm D4}_{\rm 61}$	BE	D5 44	B3 41	F8	57 4B	6A	DA	BF 54	D9	—3,²Ñ;Ô¾Õ³ø₩jÚ¿Ù ⟨I∺=«D=∨D≯D/D/TT¤
00002E90	C1		F6 9A		<u>њ</u> в С1		ьт С1	Д7 С3		41 0C	52 91		42	49 EE	54 E3	$^{A4}_{D0}$	ÁJösæÐaxDARKBIT¤
00002EA0 00002EB0	8B B0	A4 DE	ЭА 31	D4 19	36	F2 0E	64	С3 36	6D 80	E1	91 1C	5D 24	85 3C	ee EC	£з 41	9B	∎¤∎ÔáòáÃm∎′]∎îãĐ °Þ1 ⊧6 яd6∎á \$<ìA∎
00002EC0	95	2A	$\frac{31}{16}$	с <u>5</u>	CE	7B	BD	36 21	00 95	33	63	Ξ4 B2	эс 94	E5	41 3E	F2	#_ÅÎ{½! 3c² å>ò
00002EC0	BE	2 н 9 D	99	B4	FB	16	09	$\frac{21}{11}$	1A	53 69	13	1C	C9	EO	BE		¾ [´û⊣. ∢ +i∥ Éà¾?
00002EE0	68	76	ЗF	52	38	Ē5	54	1Å	8Ē	4E	Ē1	FB	82	C5	59	E1	hv?R8åT→Náû ÅYá
00002EF0	42	82	4 A	FÅ	B8	66	2Å		08	DC		DO	0D	27	08	Ĉ3	B Jú,f*É⊐ÜNÐ. '⊐Ã
00002F00	5Č	50	A5		B6	79	ĂĒ		ЗĎ	4E	Č5	F6	33	F4	28	čč	\P¥U¶v©∔=NÅö3ô(Ì
00002F10	Ē8	B8	D7	čŏ	Ē9		14		DF	F8	$\tilde{B1}$	D7	$\tilde{B4}$	58	ĒĒ		è,×Àé′¶∄Bø±×′Xî∎
00002F20	ŌĂ	83	61		2Ć	97		9B	65	7Ē	01	74		ĂĎ	14	ĩĊ	. a, * e tò+
00002F30	78	Ŏ7	85	ÃŌ	Ā1		Ē6	46	94	$\frac{1}{48}$	ĒĒ	96	8F			82	x i of Hi > 1
00002F40	F8	35	5Ĕ		24	71		FČ	75	ЗĎ	B6	ćč	ЗD	80	Ē2	8D	ø5^5\$qeüu=¶Ì=∣â
00002F50	12	ĨĂ		56	56	D2		D5	57	Ă7	$\tilde{43}$	DF	ČĒ	ΒB		Ă6	1 +ºVVQ Õ₩SCBÏ≫ú¦
00002F60	Č9	70	D6		Å7			ЗČ	56	64	ĊĒ		FA		13	E9	ÉpÖMSùÕ <vdîáúz∥é< td=""></vdîáúz∥é<>
00002F70	68	DĒ	12		37		03	5Ā	FO	FD	8Ā	4D		ΟĈ			hH́tù7" -Zðý∎M _n ∎LÜ
00002F80	E2	ЗF	ĈŌ			Ā6		30		45				57			â?ÀÃ∗¦ĬOáEÆ ^W®.
00002F90	EC	42	Ē9	6B	ĒΒ	1B	18	07	5B	ŌŌ	86	1B	5A	F5	ED		ìBékë⊷∱⊷[.∎⊷Zõí
E: 10																	

Figure 16 – Ransomware key

The encrypted files are given '.darkbit' extension and also a ransom note is dropped in the respective folders.

Dear Colleagues,
We're sorry to inform you that we've had to hack we have network completely and transfer "all" data to our secure servers.
So, keep calm, take a breath and think about an apartheid regime that causes troubles here and there.
They should pay for their lies and crimes, their names and shames. They should pay for occupation, war crimes against humanity,
killing the people (not only Palestinians' bodies, but also Israelis' souls) and destroying the future and all dreams we had.
They should pay for firing high-skilled experts.
Anyway, there is nothing for you (as an individual) to be worried.
That's the task of the administration to follow up our instruction for recovering the network.
But, you can contact us via TOX messenger if you want to recover your files personally. (TOX ID: AB33BC51AFAC64D98226826E70B483593C81CB22E6A3B504F7A75348C38C86
Our instruction for the administration:
All your files are encrypted using AES-256 military grade algorithm. So,
1. Don't try to recover data, because the encrypted files are unrecoverable unless you have the key.
Any try for recovering data without the key (using third-party applications/companies) causes PERMANENT damage. Take it serious.
2. You have to trust us. This is our business (after firing from high-tech companies) and the reputation is all we have.
3. All you need to do is following up the payment procedure and then you will receive decrypting key using for returning all of your files and VMs.
 Payment method:
Enter the link below
http://iw6v2p3cruy7tqfup3y14dgt4pfibfa3ai4zgnu5df2q3hus31m7c7ad.onion/support
Enter the ID below and pay the bill (80 BTC)
dabda-bt2as4dfa-294jfajks-qt19-bm3xu2
You will receive decrypting key after the payment.
Veries shar you due have 10 hours 15 and the deadline a 205 merils will be added as she arise
Notice that you just have 48 hours. After the deadline, a 30% penalty will be added to the price. We put data for sale after 5 davs.
We put data for sole after 5 days. Take it serious and don't listen to probable advices of a stupid government.
Take to bestode and don o stablen of probable advices of a bodyed government.
Good Luck!

"DarkBit"

Figure 17 – Ransomware Note



Figure 18 – Tweet on

DarkBit

MuddyWaters uses different types of attacks for initial access like phishing email campaigns, using tools like MimiKatz to break into the system, etc. In this case, MuddyWaters made use of Cobalt Strike to get initial access into the system. From the ransomware note, we figured out that it was a politically motivated attack.

We at K7 Labs provide detection for DarkBit ransomware and all the latest threats. Users are advised to use a reliable security product such as "K7 Total Security" and keep it up-todate to safeguard their devices.

Indicators of Compromise (IOCs)

File Name	Hash	Detection Name
DarkBit.exe	9880FAE6551D1E9EE921F39751A6F3C0	Trojan (0058e3dd1)
hr-update.iso	1219A8880DEBDD10D081195E27A2A016	Trojan (0001140e1)

References

https://attack.mitre.org/groups/G0069/