Orion Threat Alert: Flight of the BumbleBee

⋮ 4/14/2022

By: Max Malyutin – Orion Threat Research Team Leader

Orion, Cynet's Threat Research and Intelligence team, spotted a new malware campaign in the wild: BumbleBee.

Wondering what's going on? Let us fill you in.

We noticed a new trend in Initial Access Brokers' (IAB) tactics to gain access to their victims' machines. Initial Access Brokers refers to a cybercrime group that specializes in gaining initial access to organizations for the sole purpose of offering it to other threat actor groups. The trend started earlier this year and our team recently spotted their new BumbleBee campaign.

Usually, we observe malicious spam (MalSpam) campaigns that deliver malicious documents (MalDoc) to lure the victims to interact with the MalDoc and execute the malicious macro code by clicking "Enable Content." That in turn downloads and executes the malicious payload, for example, the notorious Emotet campaigns.

We expected these groups to change the initial access methods. We believe there is a direct relation to the changes Microsoft applied recently to the default policy in their Office products: "Macros from the internet will be blocked by default in Office" and "Excel 4.0 (XLM) macros are disabled by default." These changes impact IABs because they have been abusing Office documents with malicious macros for years.

It appears that they've come up with a plan B.

In this post, we will cover what this campaign is, and how the IAB distributes the BumbleBee malware and its TTPs. We will also explain each TTP according to the MITRE ATT&CK model, and its purpose.

A new campaign in the wild: BumbleBee

From our initial analysis, BumbleBee is a custom new loader that is used by different IAB groups. This malware was observed injecting Cobalt Strike shellcodes in memory and using several tactics, techniques, and procedures (TTPs) in order to compromise the victim's environment.

As part of the campaign, the threat actors abuse spoofed companies' identities (like fake employee email addresses, fake websites, etc.) and use legitimate public storage services to deliver the malicious ISO image file. The ISO image file is responsible for luring the victim to execute the BumbleBee malware.

We've seen Living Off the Land Binaries (LOLBins) execution with rundll32, which allows threat actors to avoid defenses. BumbleBee also creates a scheduled task on the compromised host for persistence and

executes a Visual Basic script via the scheduled task. The IAB relies on the user (victim) execution to execute the BumbleBee payload by luring the victim to mount an ISO image file and click on a Windows shortcut (LNK) file.

The malware name, BumbleBee, was chosen because of its unique user agent, “bumblebee,” that was used as part of the communication with the command and control server (C2).

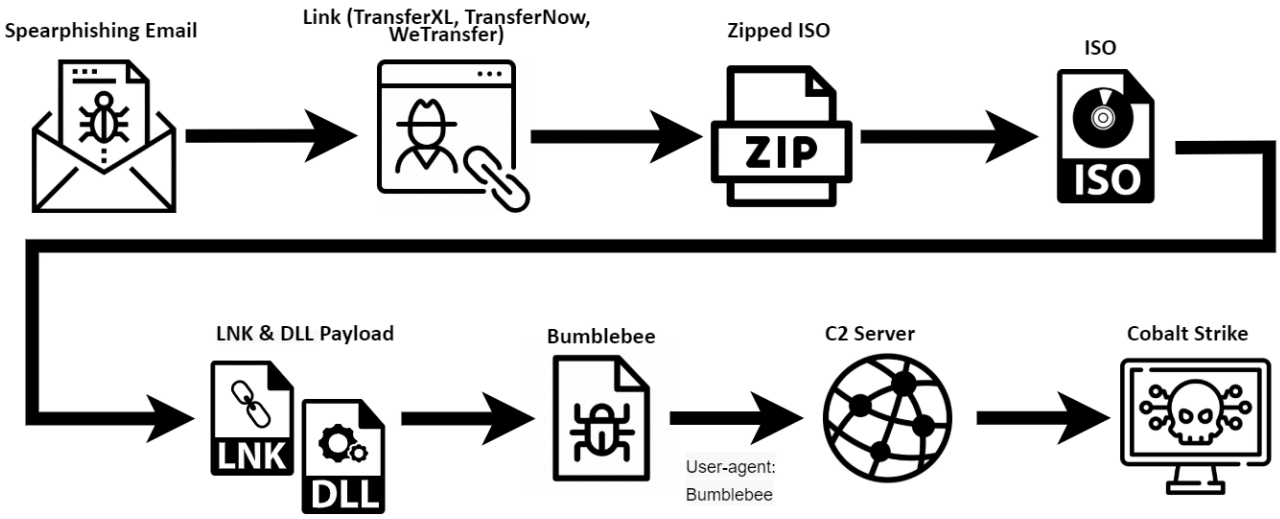
Threat Analysis Group (TAG) shared observations on the financially motivated threat actor, EXOTIC LILY, that use the BumbleBee malware. In addition, TAG mentioned an interesting point of collaboration between EXOTIC LILY and the WIZARD SPIDER threat group.

Orion’s observations

This type of attack is new, and the cybersecurity community is still gathering data to glean more insights on the nature of this attack and its targets.

Orion found a high number of targeted companies based in the US with the following distribution method that delivers the BumbleBee malware: Spear phishing email > URL Link (TransferXL, TransferNow, WeTransfer) > Zipped ISO > ISO (contains the LNK file and the BumbleBee payload).

You can see the execution flow in the image below.



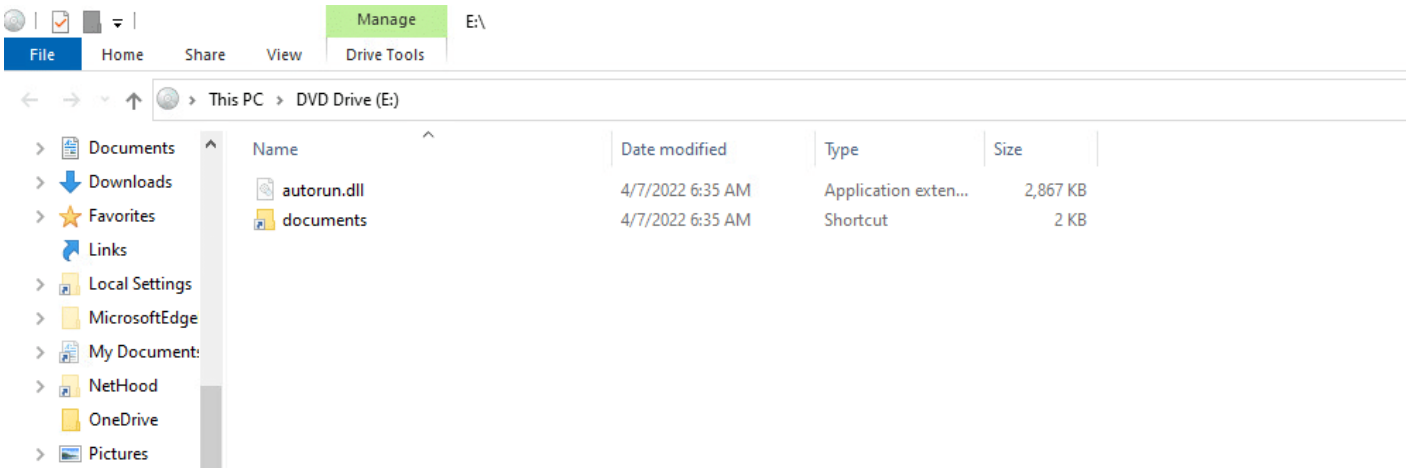
The infection flow

We’ve handled several incident response (IR) cases where threat actors distributed BumbleBee malware. After the initial infection, the threat actors inject Cobalt Strike shellcode in memory and execute discovery commands to collect info about the victim’s network. We believe that threat actors performed this data collection in order to execute the next stage of the infection.

The next stage is probably related to ransomware operations. We’re still investigating IR cases in order to find conclusive evidence that the next stage delivers ransomware.

On April 12, 2022, the BumbleBee IAB group was spotted using IMG file format in addition to ISO file format.

You can see an example in the image below.

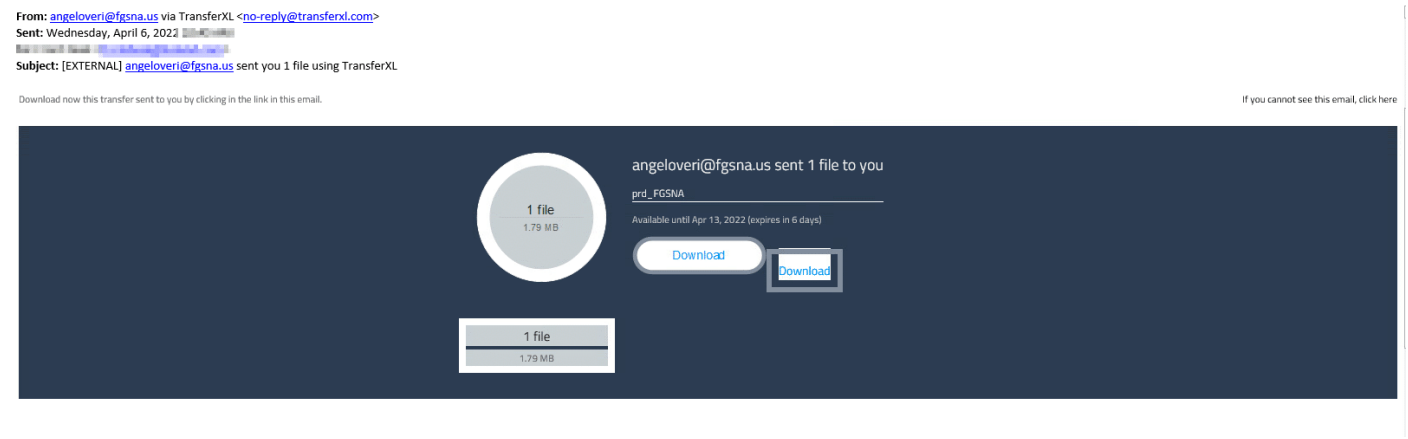


The IMG file, which contains LNK and DLL

Orion's technical analysis

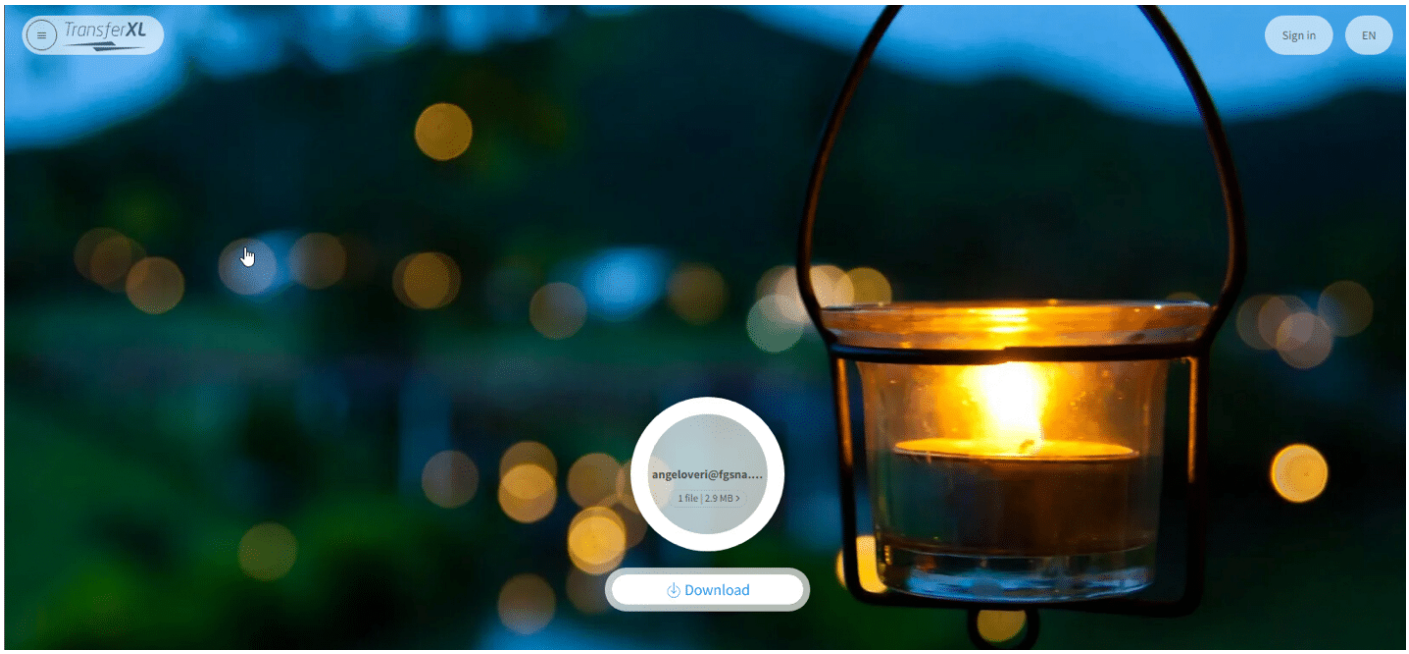
Initial Access

The BumbleBee payload was delivered via a spear phishing email that was sent from a spoofed email address. The email contains a URL link to the legitimate public storage service, TransferXL.



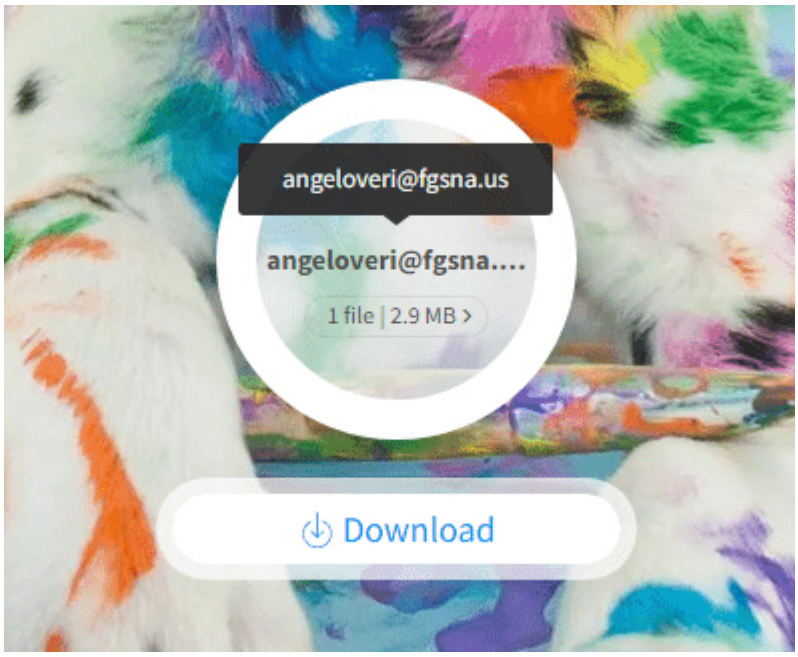
Spear phishing email with a link to TransferXL

Below you'll see the legitimate public storage site, which leads the victim to the link to the malicious file.



TransferXL legitimate public storage services

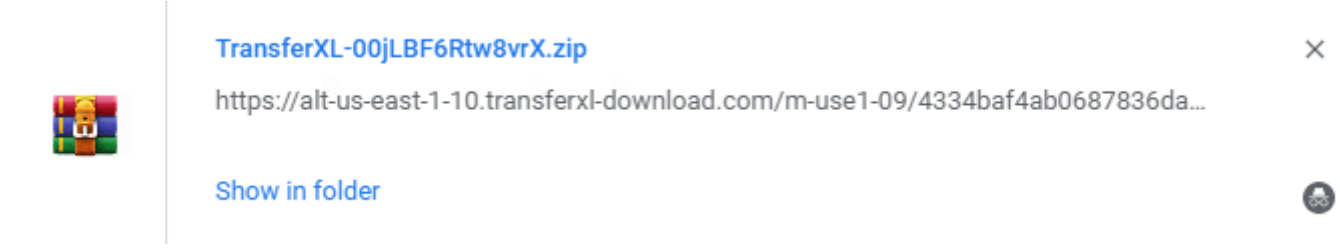
Once they click download, the victim receives a ZIP folder that contains the malicious ISO image files.



Spoofed company email address

Execution

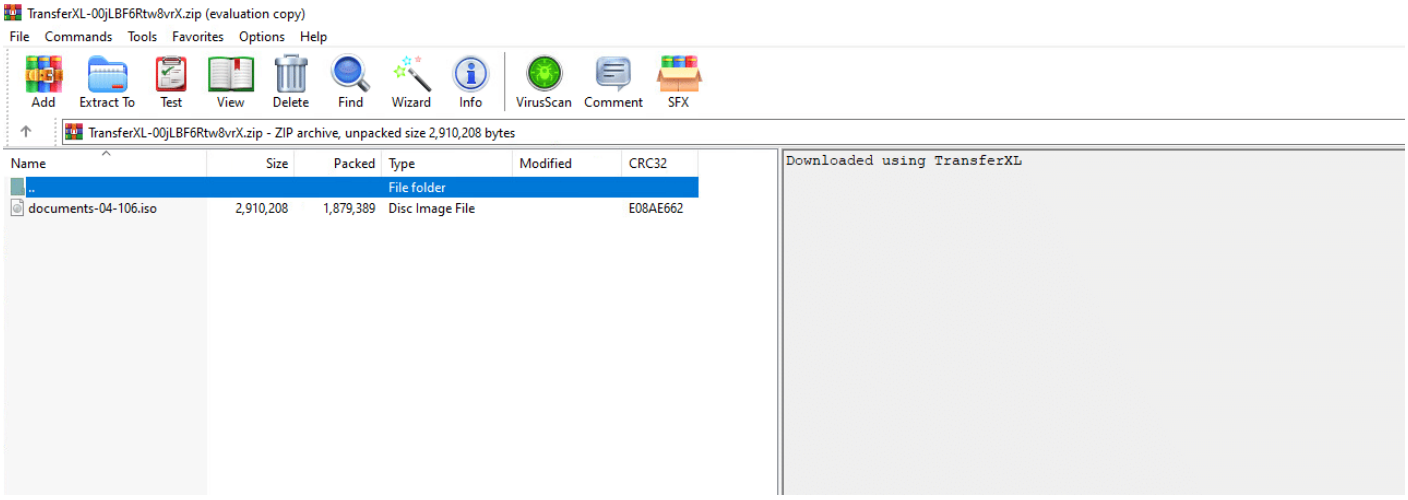
Below is an example of what the ZIP file from the TransferXL link looks like.



ZIP file download from TransferXL

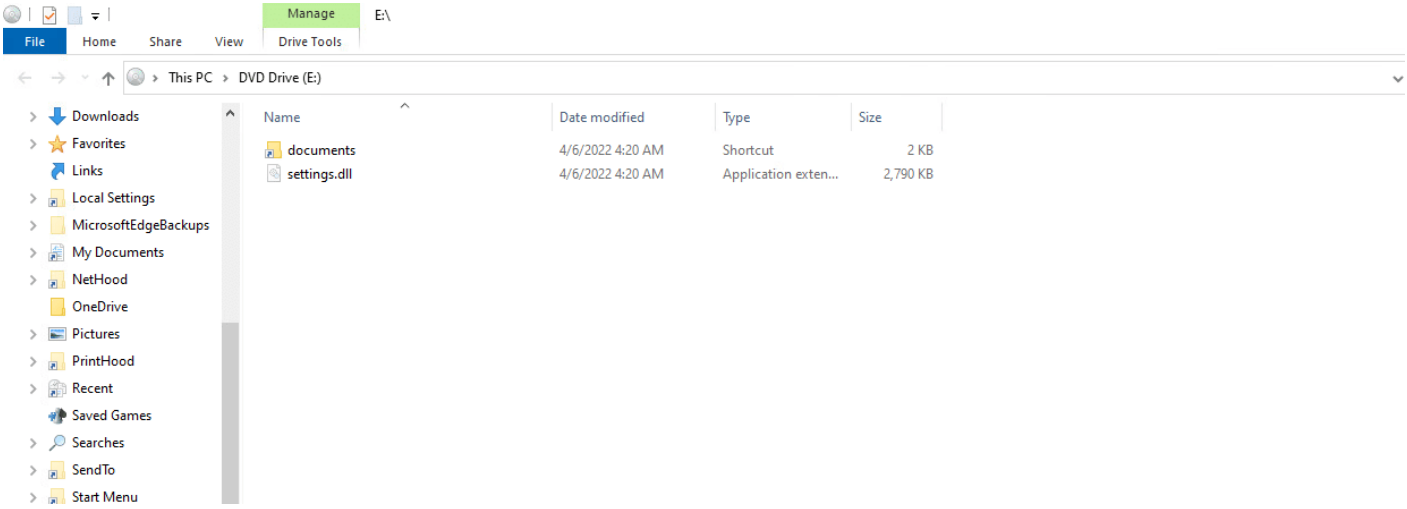
The ZIP file contains an ISO image file with the following name "*documents-04-106.iso."* Note that the following ISO image file name pattern was used for all the files that we have analyzed:

- *documents-[0-9]{1,4}-[0-9]{1,4}\.iso*



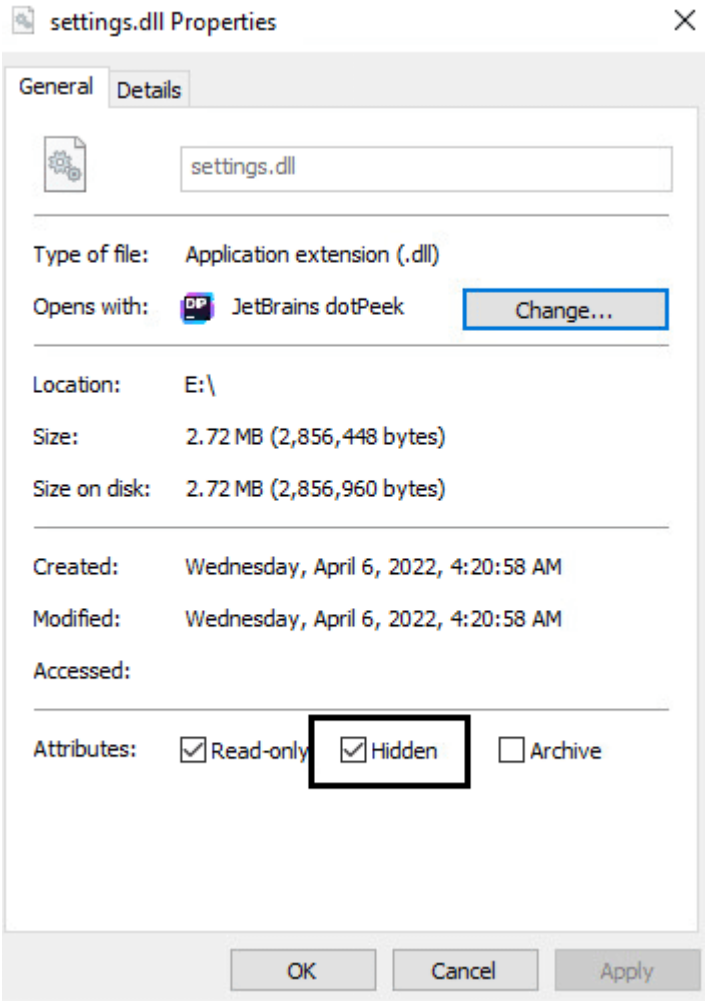
ISO image file

From this step, threat actors rely on the victim (user) interaction with the ISO image file. The threat actors use a masquerading technique by setting the LNK file icon to be a folder icon in order to lure the victim to click on the LNK file:



ISO image file contains LNK and DLL

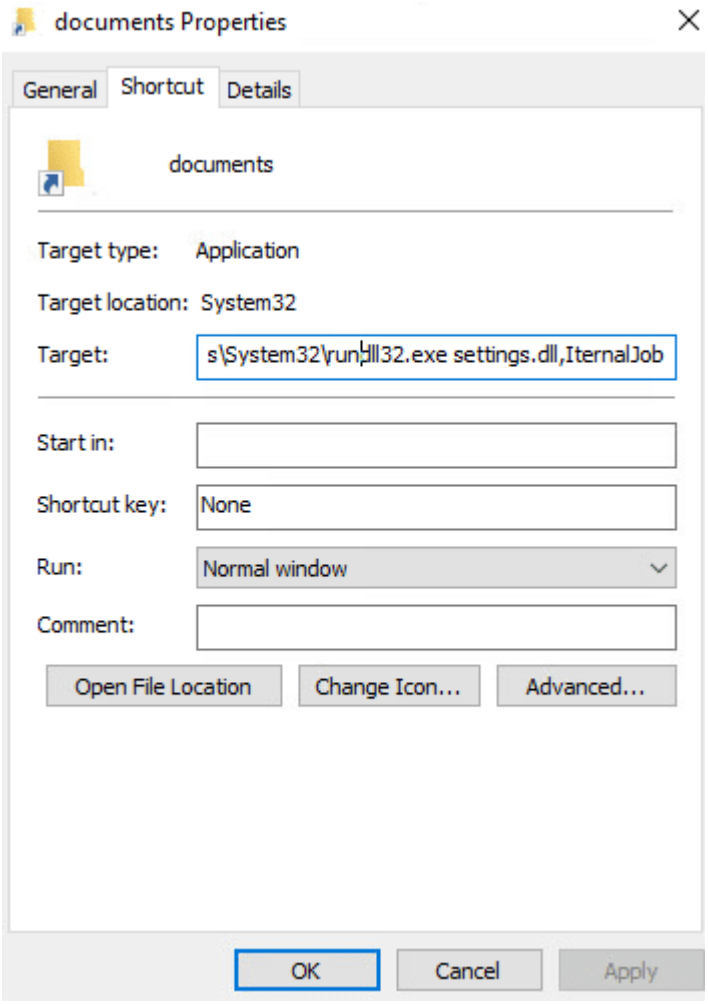
In addition, the DLL payload attribute is set as "Hidden" in order to hide the DLL payload from the user when interacting with the ISO image file:



Hidden attribute for the DLL

The masqueraded LNK file properties show that the execution target is as follows:

- *C:\Windows\System32\rundll32.exe settings.dll,IternalJob*



LNK executes the DLL via rundll32 command

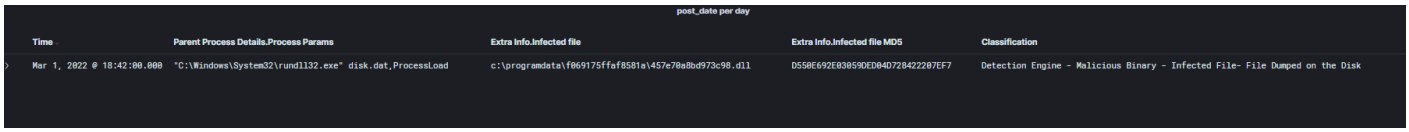
After the initial execution, the BumbleBee DLL is copied to the %programdata%/{RandomDir} directory. In addition to the DLL, a VBS script is also dropped to the same directory:

- [a-z]:\\programdata\\[a-z0-9]{16}\\[a-z0-9]{16}\.[vbs|dll]

action	process_path	file_path	new_process_command_line
Write	c:\windows\system32\rundll32.exe	c:\programdata\045241ad770d73bd\8cbc96f0e4e2ae45.vbs	-
Create New	c:\windows\system32\rundll32.exe	c:\programdata\045241ad770d73bd\8cbc96f0e4e2ae45.vbs	-
Execute New Process	c:\windows\system32\wbem\wmiprvse.exe	c:\windows\system32\wscript.exe	wscript.exe C:\\ProgramData\\045241ad770d73bd\\8cbc96f0e4e2ae45.vbs
Execute New Process	c:\windows\explorer.exe	c:\windows\system32\rundll32.exe	'C:\\Windows\\System32\\rundll32.exe' settings.dll,IternalJob
Execute New Process	c:\windows\explorer.exe	c:\windows\system32\rundll32.exe	'C:\\Windows\\System32\\rundll32.exe' settings.dll,IternalJob
Execute New Process	c:\windows\explorer.exe	c:\windows\system32\rundll32.exe	'C:\\Windows\\System32\\rundll32.exe' settings.dll,IternalJob

TTPs indicators during the execution

We have other artifacts from different IR cases, where we have observed the following activity. The screenshot below shows an event that detected a creation of a payload in the %ProgramData%\{Random} directory the DLL payload is a copy of the initial BumbleBee loader that executed by Rundll32 from the ISO image file:



Copy of the BumbleBee DLL to %Programdata% directory

In other IR cases, we observed an execution flow that's bit different. For example, a LNK that points to the following execution targets:

- cmd.exe /c start rundll32 neqw.dll,IternalJob
- rundll32.exe advpack.dll,RegisterOCX sysctl.exe

Persistence

We detected a scheduled task execution during the BumbleBee infection:

Grandparent process:

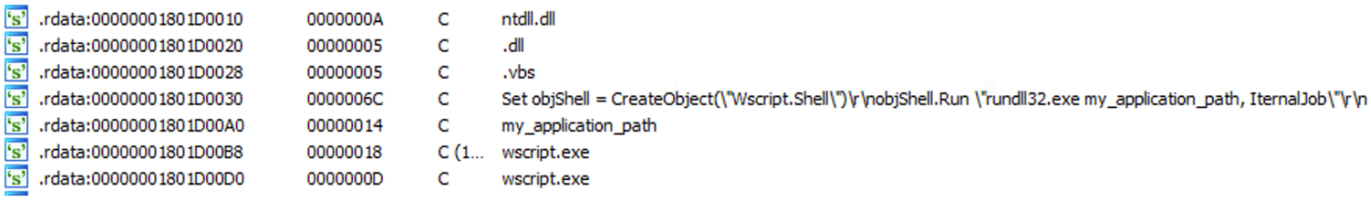
svchost.exe -k netsvcs -p -s Schedule

Parent process:

wscript.exe [a-z]:\\programdata\\[a-z0-9]{16}\\[a-z0-9]{16}\.vbs

Child process:

rundll32.exe [a-z]:\\programdata\\[a-z0-9]{16}\\[a-z0-9]{16}\.dll,{Export}



Strings from the BumbleBee loader show the VBS script and the execution method

We also observed WMI execution. The VBS file that was executed via a scheduled task, was also executed through WMI:

Grandparent process:

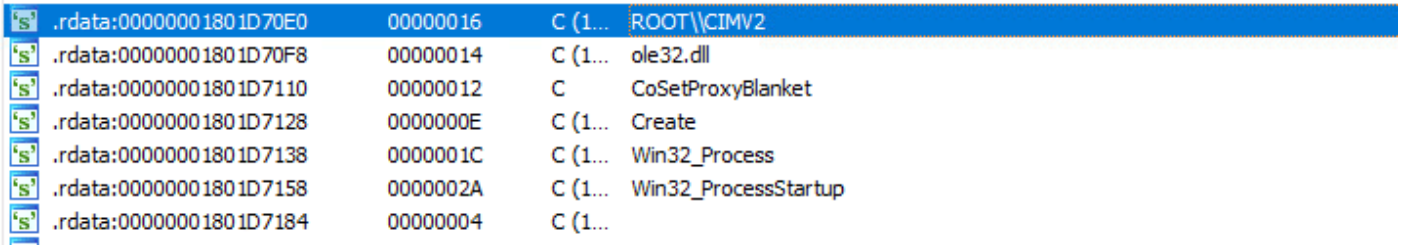
svchost.exe -k DcomLaunch

Parent process:

wmiprvse.exe -Embedding

Child process:

wscript.exe [a-z]:\\programdata\\[a-z0-9]{16}\\[a-z0-9]{16}\.vbs



Strings from the Bumblebee loader show the WMI Win32_Process execution

Defense Evasion

In our labs, we observed that BumbleBee uses several anti-VM methods to avoid detection.

One of the anti-VM checks is related to the VirtualBox product:

```
lea     rcx, asc_1801D8CA8 ; "\b"
call    sub_180041A90
test    eax, eax
mov     esi, edi
setz    sil
and     esi, ebp
call    sub_18003E3A0
mov     ecx, edi
test    eax, eax
setz    cl
xor     edx, edx        ; lpWindowName
and     esi, ecx
lea     rcx, ClassName  ; "VBoxTrayToolWndClass"
call    cs:FindWindowW
lea     rdx, WindowName ; "VBoxTrayToolWnd"
xor     ecx, ecx        ; lpClassName
mov     rbx, rax
call    cs:FindWindowW
test    rbx, rbx
jnz     short loc_18003D9AA
```

Check for lpWindowName if matches VirtualBox

Other anti-VM artifacts were found after unpacking, as can be seen in the following strings:

Offset	Type	Strings found
001D8573	UNICODE	VBOX__
001D85AB	UNICODE	VBOX__
001D85E3	UNICODE	VBOX__
001D8BE9	UNICODE	VBoxControl.exe
001D868E	UNICODE	VBoxGuest
001D8CDC	UNICODE	VBoxGuest
001DA478	UNICODE	VBoxGuest
001D8891	UNICODE	VBoxGuest.sys
001D8D01	UNICODE	VBoxMiniRdDN
001D8CB4	UNICODE	VBoxMiniRdrDN
001D86DE	UNICODE	VBoxMouse
001DA490	UNICODE	VBoxMouse
001D8851	UNICODE	VBoxMouse.sys
001D878E	UNICODE	VBoxSF
001DA468	UNICODE	VBoxSF
001D88D1	UNICODE	VBoxSF.sys
001D872E	UNICODE	VBoxService
001D8D2C	UNICODE	VBoxTrayIPC
001D8D51	UNICODE	VBoxTrayIPC
001D8DD0	UNICODE	VBoxTrayToolWnd
001D8DA0	UNICODE	VBoxTrayToolWndClass
001D87DE	UNICODE	VBoxVideo
001D8909	UNICODE	VBoxVideo.sys
001D8F58	UNICODE	VBoxVideoW8
001D8F70	UNICODE	VBoxWddm
001D9A68	UNICODE	VMSrvc.exe
001D9A80	UNICODE	VMUSrvc.exe
001D9A60	ASCII	VMWARE
001D92B8	UNICODE	VMWARE
001D9938	UNICODE	VMWare
001D9868	UNICODE	VMWare\
001D9A58	ASCII	VMware
001D94F9	UNICODE	VMware, Inc.\VMware Tools
001D9948	UNICODE	\\.\HGFS
001D8CD8	UNICODE	\\.\VBoxGuest

List of strings that are related to VMware and VirtualBox

BumbleBee also detects if it is running within a VM by checking for known services that are related to different VM products:

Offset	Type	Strings recognized as registry key
001D9AF0	UNICODE	SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters
001D9410	UNICODE	SYSTEM\ControlSet001\Control\SystemInformation
001D9E20	UNICODE	SYSTEM\ControlSet001\Services\BALLOON
001D9E70	UNICODE	SYSTEM\ControlSet001\Services\BalloonService
001D8670	UNICODE	SYSTEM\ControlSet001\Services\VBoxGuest
001D86C0	UNICODE	SYSTEM\ControlSet001\Services\VBoxMouse
001D8770	UNICODE	SYSTEM\ControlSet001\Services\VBoxSF
001D8710	UNICODE	SYSTEM\ControlSet001\Services\VBoxService
001D87C0	UNICODE	SYSTEM\ControlSet001\Services\VBoxVideo
001D9D60	UNICODE	SYSTEM\ControlSet001\Services\VirtIO-FS Service
001D9DC0	UNICODE	SYSTEM\ControlSet001\Services\VirtioSerial
001D9ED0	UNICODE	SYSTEM\ControlSet001\Services\netkvm
001D9CC0	UNICODE	SYSTEM\ControlSet001\Services\vioscsi
001D9D10	UNICODE	SYSTEM\ControlSet001\Services\viostor

List of services that are related to VM products

BumbleBee checks whether certain user names reside in the victim's machine by comparing against a hardcoded list of user names. This allows BumbleBee to detect sandboxes and labs that are used for malware analysis:

's'	.rdata:0000000180 1D88C8	00000018	C (1...	CurrentUser
's'	.rdata:0000000180 1D88E0	00000010	C (1...	Sandbox
's'	.rdata:0000000180 1D88F0	0000000C	C (1...	Emily
's'	.rdata:0000000180 1D8900	00000010	C (1...	HAPUBWS
's'	.rdata:0000000180 1D8910	00000012	C (1...	Hong Lee
's'	.rdata:0000000180 1D8928	00000012	C (1...	IT-ADMIN
's'	.rdata:0000000180 1D8940	00000010	C (1...	Johnson
's'	.rdata:0000000180 1D8950	0000000E	C (1...	Miller
's'	.rdata:0000000180 1D8960	0000000E	C (1...	milozs
's'	.rdata:0000000180 1D8970	0000001A	C (1...	Peter Wilson
's'	.rdata:0000000180 1D8990	0000000C	C (1...	timmy
's'	.rdata:0000000180 1D89A0	00000012	C (1...	sand box
's'	.rdata:0000000180 1D89B8	00000010	C (1...	malware
's'	.rdata:0000000180 1D89C8	00000010	C (1...	maltest
's'	.rdata:0000000180 1D89D8	00000014	C (1...	test user
's'	.rdata:0000000180 1D89F0	0000000C	C (1...	virus
's'	.rdata:0000000180 1D8A00	00000012	C (1...	John Doe
's'	.rdata:0000000180 1D8A20	00000046	C (1...	Checking if username matches : %s
's'	.rdata:0000000180 1D8A68	0000000E	C (1...	VMWare

List of hardcoded usernames which are related to sandboxes and labs

In addition, it uses WMI queries to collect system details and information:

- SELECT * FROM Win32_BaseBoard
- SELECT * FROM Win32_Bus
- SELECT * FROM Win32_ComputerSystem
- SELECT * FROM Win32_Fan
- SELECT * FROM Win32_NTEventlogFile
- SELECT * FROM Win32_OperatingSystem

- SELECT * FROM Win32_PnPDevice
- SELECT * FROM Win32_PnPEntity

Discovery

We found that the threat actors used the AdFind tool to enumerate and map the victim's network. The ADFind tool was found in the %ProgramData% directory.

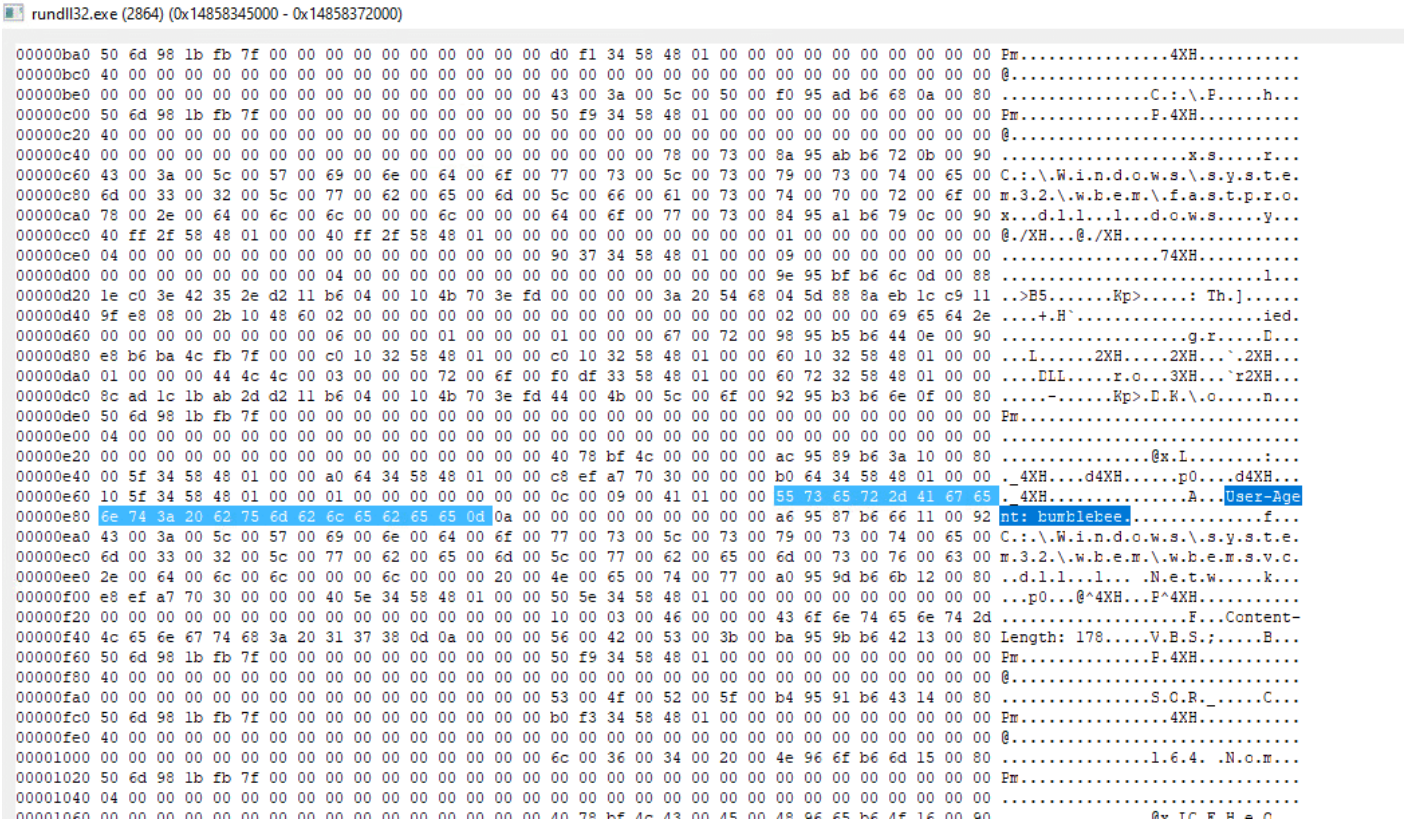
In the instance we observed, the following commands were used:

- adfind.exe -gcb -sc trustdmp
- adfind.exe -f "(objectcategory=group)"
- adfind.exe -f "(objectcategory=organizationalUnit)"
- adfind.exe -f "objectcategory=computer"
- adfind.exe -f "(objectcategory=person)"

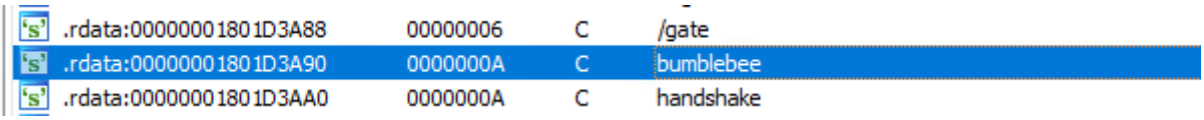
Command and Control

After the initial execution, the BumbleBee process (Rundll32) communicated with the Command-and-Control server (C2). We've seen several C2 servers from different IR cases:

- *IP: 23.82.19[.]208:443*
- *IP: 192.236.198[.]63:433*
- *IP: 45.147.229[.]177:433*



Example of the unique User-Agent: BumbleBee in the payload's memory



Additional reference to the BumbleBee malware name

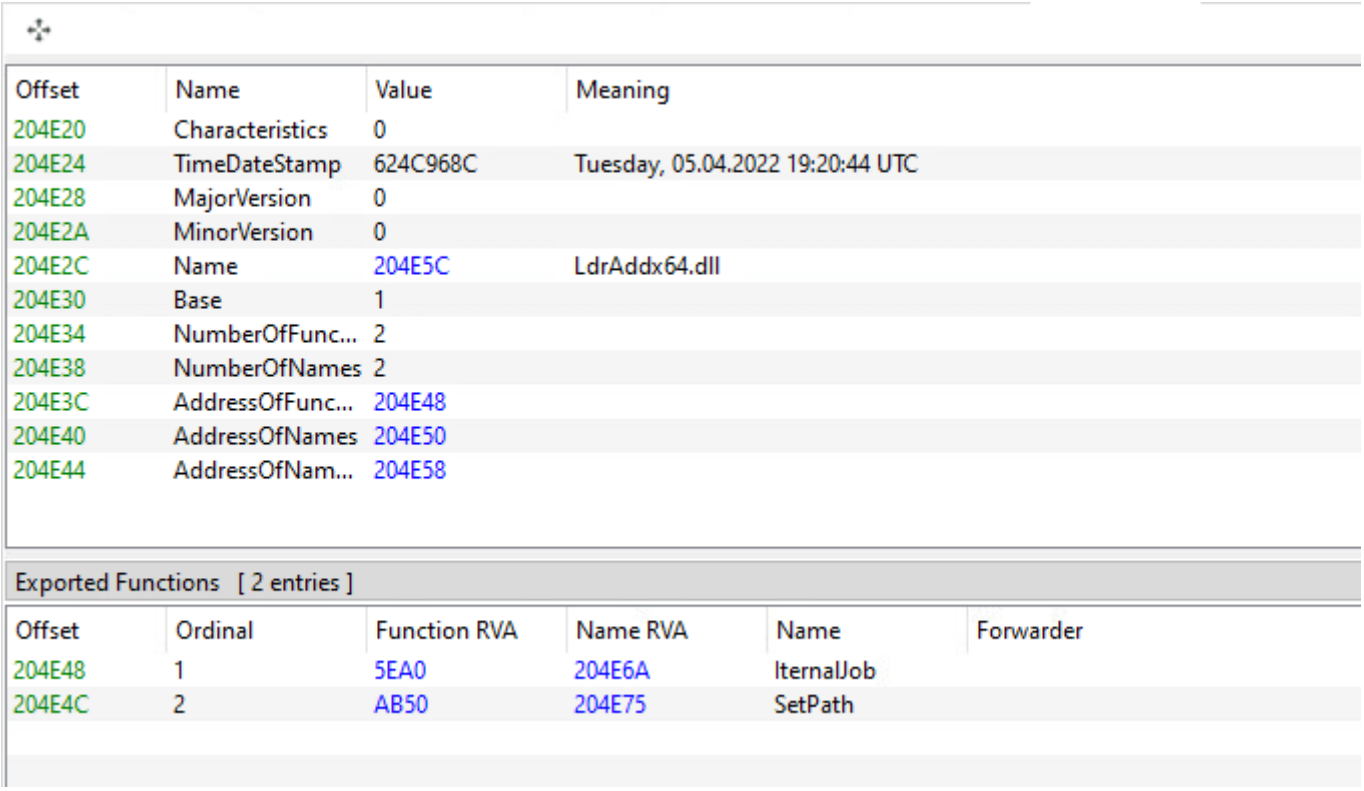
All the collected system and network information is sent to the C2 server, which sends back a response containing the next step/command to execute based on that info.

BumbleBee binary analysis

In this section, we will cover some interesting indicators and artifacts that highlighted the BumbleBee actions and heuristics. These artifacts also help us to identify the BumbleBee malware.

We analyzed several payloads and all of them had the same artifacts.

After unpacking the BumbleBee loader and by searching in the metadata of the unpacked payload, we identified BumbleBee's internal name, "LdrAddx64.dll," and two export functions – "IternalJob" and "SetPath."



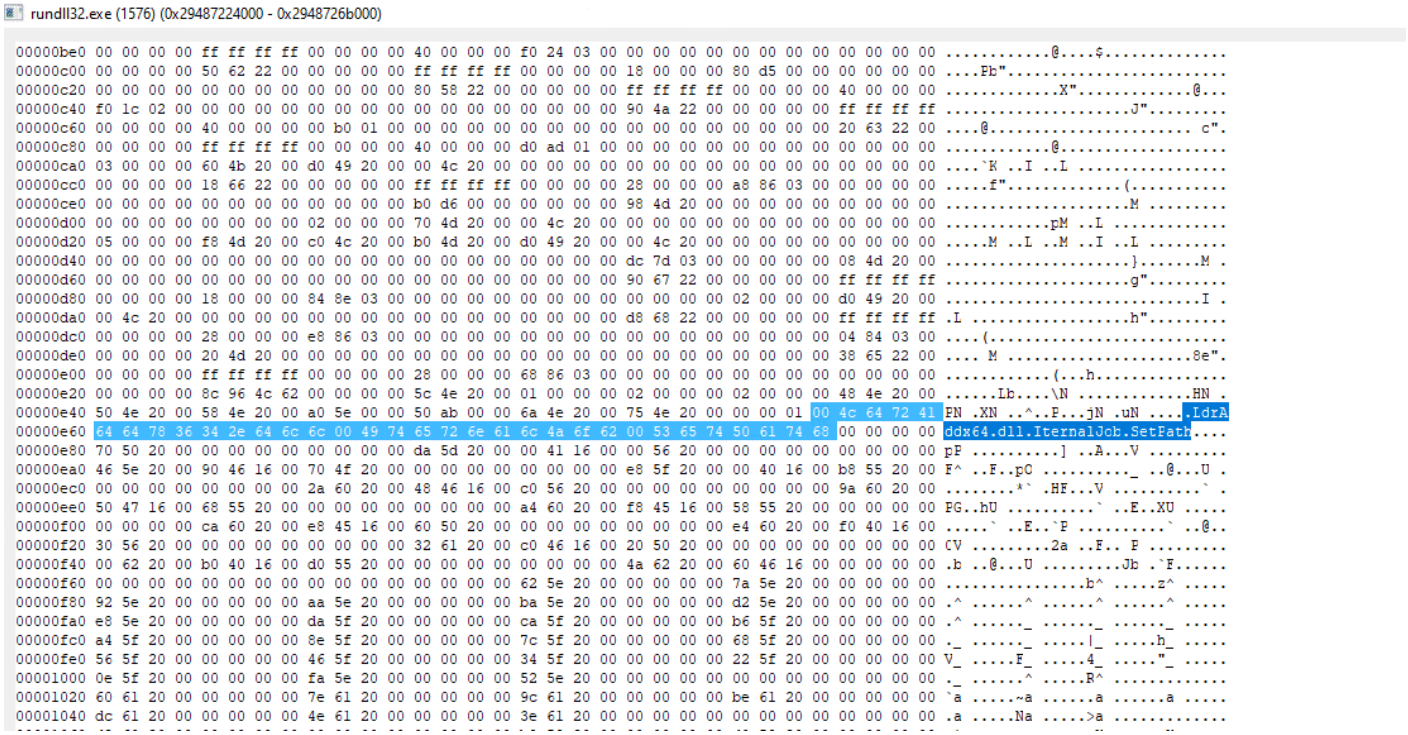
Offset	Name	Value	Meaning
204E20	Characteristics	0	
204E24	TimeDateStamp	624C968C	Tuesday, 05.04.2022 19:20:44 UTC
204E28	MajorVersion	0	
204E2A	MinorVersion	0	
204E2C	Name	204E5C	LdrAddx64.dll
204E30	Base	1	
204E34	NumberOfFunc...	2	
204E38	NumberOfNames	2	
204E3C	AddressOfFunc...	204E48	
204E40	AddressOfNames	204E50	
204E44	AddressOfNam...	204E58	

Exported Functions [2 entries]

Offset	Ordinal	Function RVA	Name RVA	Name	Forwarder
204E48	1	5EA0	204E6A	IternalJob	
204E4C	2	AB50	204E75	SetPath	

BumbleBee internal name, export functions, and TimeDateStamp

In the image below, we found the BumbleBee internal name and export function inside the process Rundll32.exe that executed the BumbleBee DLL loader:



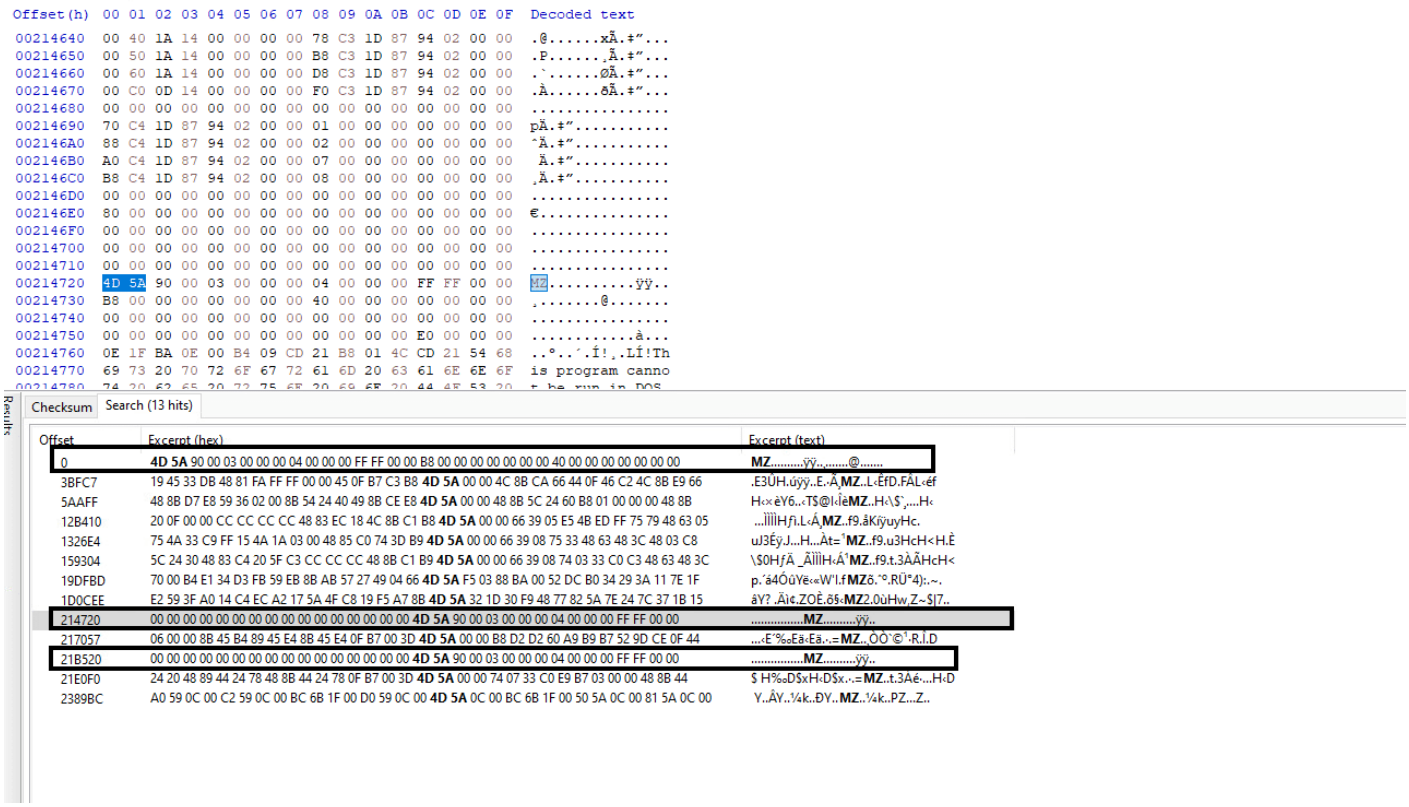
Bumblebee's internal name and the export functions names in the memory

By inspecting the unpacked BumbleBee sections, we discovered that the .data section contains two executables:

property	value	value	value	value	value	value	value
name	.text	.rdata	.data	.pdata	.gfids	.tls	.reloc
md5	7855722A7B96091D08D348E...	66C85B9E435112AAEDD1C1...	07A36690C4A70C0EE985D806FF1F25AD	490087E1686172414BBE5439...	1429057928C52BB91F168CF...	BE966AE9956FBCDBEB8EDF...	FD7C575D75D1F69FC4C79D...
entropy	6.409	5.667	5.264	6.194	3.579	0.003	5.440
file-ratio (98.23%)	60.35 %	27.68 %	5.39 %	3.56 %	0.02 %	0.21 %	1.02 %
raw-address	0x00001000	0x00164000	0x00207000	0x0022D000	0x00242000	0x00243000	0x00245000
raw-size (2361856 bytes)	0x00162400 (1451008 bytes)	0x000A2800 (665600 bytes)	0x0001FA00 (129536 bytes)	0x00014E00 (85504 bytes)	0x00000200 (512 bytes)	0x00001400 (5120 bytes)	0x00006000 (24576 bytes)
virtual-address	0x0000000080001000	0x0000000080164000	0x0000000080207000	0x000000008022D000	0x0000000080242000	0x0000000080243000	0x0000000080245000
virtual-size (2385799 bytes)	0x0016234E (1450830 bytes)	0x000A270C (665356 bytes)	0x00025B1C (154396 bytes)	0x00014DF0 (85488 bytes)	0x000001D4 (468 bytes)	0x000013D1 (5073 bytes)	0x00005E7C (24188 bytes)
entry-point	**0x0012BC18**	-	-	-	-	-	-
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040	0x40000040	0xC0000040	0x42000040
writable	-	-	**x**	-	-	**x**	-
executable	**x**	-	-	-	-	-	-
shareable	-	-	-	-	-	-	-
discardable	-	-	-	-	-	-	x
initialized-data	-	x	x	x	x	x	x
uninitialized-data	-	-	-	-	-	-	-
unreadable	-	-	-	-	-	-	-
self-modifying	-	-	-	-	-	-	-
virtualized	-	-	-	-	-	-	-
file	-	-	executable, offset: 0x00214720, size: 27492	-	-	-	-
file	-	-	executable, offset: 0x0021B520, size: 25460	-	-	-	-

PEStudio shows the unpacked Bumblebee section and highlighted the .data section

We extracted the two hidden payloads from the .data section by using Hex-Editor tool:



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00214640  00 40 1A 14 00 00 00 00 78 C3 1D 87 94 02 00 00  .@......xÃ.‡”...
00214650  00 50 1A 14 00 00 00 00 B8 C3 1D 87 94 02 00 00  .P......¸Ã.‡”...
00214660  00 60 1A 14 00 00 00 00 D8 C3 1D 87 94 02 00 00  .`......ØÃ.‡”...
00214670  00 C0 0D 14 00 00 00 00 F0 C3 1D 87 94 02 00 00  .À......ðÃ.‡”...
00214680  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00214690  70 C4 1D 87 94 02 00 00 01 00 00 00 00 00 00 00  pÄ.‡”...........
002146A0  88 C4 1D 87 94 02 00 00 02 00 00 00 00 00 00 00  ˆÄ.‡”...........
002146B0  A0 C4 1D 87 94 02 00 00 07 00 00 00 00 00 00 00   Ä.‡”...........
002146C0  B8 C4 1D 87 94 02 00 00 08 00 00 00 00 00 00 00  ¸Ä.‡”...........
002146D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
002146E0  80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  €...............
002146F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00214700  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00214710  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00214720  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00214730  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00214740  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00214750  00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00  ............à...
00214760  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00214770  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00214780  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
```

Results

Checksum | Search (13 hits)

Offset	Excerpt (hex)	Excerpt (text)
0	**4D 5A** 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	**MZ**.....ÿÿ..¸.....@.....
3BFC7	19 45 33 DB 48 81 FA FF FF 00 00 45 0F B7 C3 B8 **4D 5A** 00 00 4C 8B CA 66 44 0F 46 C2 4C 8B E9 66	.E3ÛH.úÿÿ..E.·Ã¸**MZ**..L‹ÊfD.FÂL‹éf
5AAFF	48 8B D7 E8 59 36 02 00 8B 54 24 40 49 8B CE E8 **4D 5A** 00 00 48 8B 5C 24 60 B8 01 00 00 00 48 8B	H‹×èY6..‹T$@I‹Îè**MZ**..H‹\$`¸....H‹
12B410	20 0F 00 00 CC CC CC CC 48 83 EC 18 4C 8B C1 B8 **4D 5A** 00 00 66 39 05 E5 4B ED FF 75 79 48 63 05	...ÌÌÌÌHƒì.L‹Á¸**MZ**..f9.åKíÿuyHc.
1326E4	75 4A 33 C9 FF 15 4A 1A 03 00 48 85 C0 74 3D B9 **4D 5A** 00 00 66 39 08 75 33 48 63 48 3C 48 03 C8	uJ3Éÿ.J...H…Àt=¹**MZ**..f9.u3HcH<H.È
159304	5C 24 30 48 83 C4 20 5F C3 CC CC CC 48 8B C1 B9 **4D 5A** 00 00 66 39 08 74 03 33 C0 C3 48 63 48 3C	\$0Hƒä _ÃÌÌÌH‹Á¹**MZ**..f9.t.3ÀÃHcH<
19DFBD	70 00 B4 E1 34 D3 FB 59 EB 8B AB 57 27 49 04 66 **4D 5A** F5 03 88 BA 00 52 DC B0 34 29 3A 11 7E 1F	p.´á4ÓûYë‹«W'I.f**MZ**õ.ˆº.RÜ°4):.~.
1D0CEE	E2 59 3F A0 14 C4 EC A2 17 5A 4F C8 19 F5 A7 8B **4D 5A** 32 1D 30 F9 48 77 82 5A 7E 24 7C 37 1B 15	âY? .Äì¢.ZOÈ.õ§‹**MZ**2.0ùHw‚Z~$|7..
214720	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 **4D 5A** 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	..............**MZ**.....ÿÿ..
217057	06 00 00 8B 45 B4 89 45 E4 8B 45 E4 0F B7 00 3D **4D 5A** 00 00 B8 D2 D2 60 A9 B9 B7 52 9D CE 0F 44	...‹E´‰Eä‹Eä·.=**MZ**..ÒÒ`©¹·R.Î.D
21B520	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 **4D 5A** 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	..............**MZ**.....ÿÿ..
21E0F0	24 20 48 89 44 24 78 48 8B 44 24 78 0F B7 00 3D **4D 5A** 00 00 74 07 33 C0 E9 B7 03 00 00 48 8B 44	$ H‰D$xH‹D$x.·.=**MZ**..t.3Aé·...H‹D
2389BC	A0 59 0C 00 C2 59 0C 00 BC 6B 1F 00 D0 59 0C 00 **4D 5A** 0C 00 BC 6B 1F 00 50 5A 0C 00 81 5A 0C 00	Y..ÂY..¼k..ÐY..**MZ**..¼k..PZ...Z..

Hex-Editor shows 3 MZ headers: the first one is the Bumblebee, and the other two are additional payloads

The first payload from the .data section is a 32-bit DLL payload:

property	value
md5	36D49170F3115D378F8B6A3A45B23525
sha1	AE1A95DA9B7488B51C8549C52DE8E2F73C022608
sha256	EED2D5DD3B0FCCD71FA30B79708004E7393E83AAC8566E80808F1162936BC1F2
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	28160 (bytes)
size-without-overlay	n/a
entropy	6.323
imphash	1369F81AACB871DA7C04248B77211BB2
signature	n/a
entry-point	55 8B EC 51 8B 45 0C 89 45 FC 83 7D FC 01 74 02 EB 0D 8B 4D 08 51 8B 55 10 52 E8 C1 FF FF FF B8 01
file-version	n/a
description	n/a
file-type	**dynamic-link-library**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x624C9623 (Tue Apr 05 12:18:59 2022)
debugger-stamp	0x624C9623 (Tue Apr 05 12:18:59 2022)
resources-stamp	n/a
import-stamp	0x00000000 (empty)
exports-stamp	0x624C9623 (Tue Apr 05 12:18:59 2022)
version-stamp	n/a
certificate-stamp	n/a

PEStudio showing the payload's metadata

We found a few interesting functions in the payload strings indicating that this payload has process injection capabilities. For example, "CreateProcess," "NtWriteVirtualMemory," "CreateRemoteThread,"and "WinExec."

encoding (2)	size (bytes)	location	blacklist (82)	hint (54)	value (382)
ascii	19	0x00005ED8	x	-	NtReadVirtualMemory
ascii	19	0x00005EEC	x	-	NtFreeVirtualMemory
ascii	23	0x00005F00	x	-	NtAllocateVirtualMemory
ascii	14	0x00005F18	x	-	NtResumeThread
ascii	18	0x00005F28	x	-	NtSetContextThread
ascii	23	0x00005F3C	x	-	NtSetInformationProcess
ascii	22	0x00005F54	x	-	NtSetInformationThread
ascii	15	0x00005F6C	x	-	NtSuspendThread
ascii	20	0x00005F7C	x	-	NtUnmapViewOfSection
ascii	11	0x00005FC8	x	-	NtOpenEvent
ascii	20	0x00005FF4	x	-	NtWriteVirtualMemory
ascii	25	0x0000600C	x	-	NtQueryInformationProcess
ascii	23	0x00006028	x	-	NtAdjustPrivilegesToken
ascii	18	0x0000605C	x	-	NtTerminateProcess
ascii	13	0x00006070	x	-	NtOpenProcess
ascii	13	0x00006080	x	-	NtOpenSection
ascii	17	0x000060B4	x	-	RtlExitUserThread
ascii	19	0x000060C8	x	-	KiUserApcDispatcher
ascii	25	0x000060DC	x	-	KiUserExceptionDispatcher
ascii	12	0x000060F8	x	-	NtOpenThread
ascii	19	0x00006108	x	-	RtlDecompressBuffer
ascii	13	0x000061A0	x	-	CreateProcess
ascii	21	0x000061B0	x	-	CreateProcessInternal
ascii	21	0x000061C8	x	-	CreateProcessInternal
ascii	13	0x000061E0	x	-	CreateProcess
ascii	18	0x000061F0	x	-	CreateRemoteThread
ascii	15	0x00006204	x	-	FindFirstFileEx
ascii	15	0x00006218	x	-	FindFirstFileEx
ascii	31	0x00006288	x	-	RtlInstallFunctionTableCallback
ascii	7	0x000062A8	x	-	WinExec
ascii	18	0x00006310	x	-	CreateRemoteThread
ascii	13	0x0000634C	x	-	FindFirstFile
ascii	13	0x0000635C	x	-	FindFirstFile
ascii	16	0x000065AA	x	-	PathFindFileName

PEStudio showing the payload's strings that could be related to process injection

The second payload that we extracted from the .data section is a 64-bit DLL payload:

property	value
md5	FC3535258586EAF20511A45BA099D14E
sha1	425E0123D2BE8B84F7B58B89A5B06711EA564344
sha256	FAC5701CCAC0C1AC224FD601EFD6DBBCF867E0BFA02AFE336ADAFA80E0399A45
md5-without-overlay	7476D0AA4BA606A51450093ECF0086ED
sha1-without-overlay	F181B15536CE3D28B8603972000B8192F5B3D04C
sha256-without-overlay	E1B7382F6D5588DED0BB9BC305CC9CF2D11272DFB0CCC0584F915D7B9B48746B
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	195296 (bytes)
size-without-overlay	26112 (bytes)
entropy	5.480
imphash	20A787DCB5EC1605108FA6BA85DA6A52
signature	n/a
entry-point	4C 89 44 24 18 89 54 24 10 48 89 4C 24 08 48 83 EC 38 8B 44 24 48 89 44 24 20 83 7C 24 20 01 74 02
file-version	n/a
description	n/a
file-type	**dynamic-link-library**
cpu	**64-bit**
subsystem	GUI
compiler-stamp	0x624C962D (Tue Apr 05 12:19:09 2022)
debugger-stamp	0x624C962D (Tue Apr 05 12:19:09 2022)
resources-stamp	0x00000000 (empty)
import-stamp	0x00000000 (empty)
exports-stamp	0x624C962D (Tue Apr 05 12:19:09 2022)
version-stamp	n/a
certificate-stamp	n/a

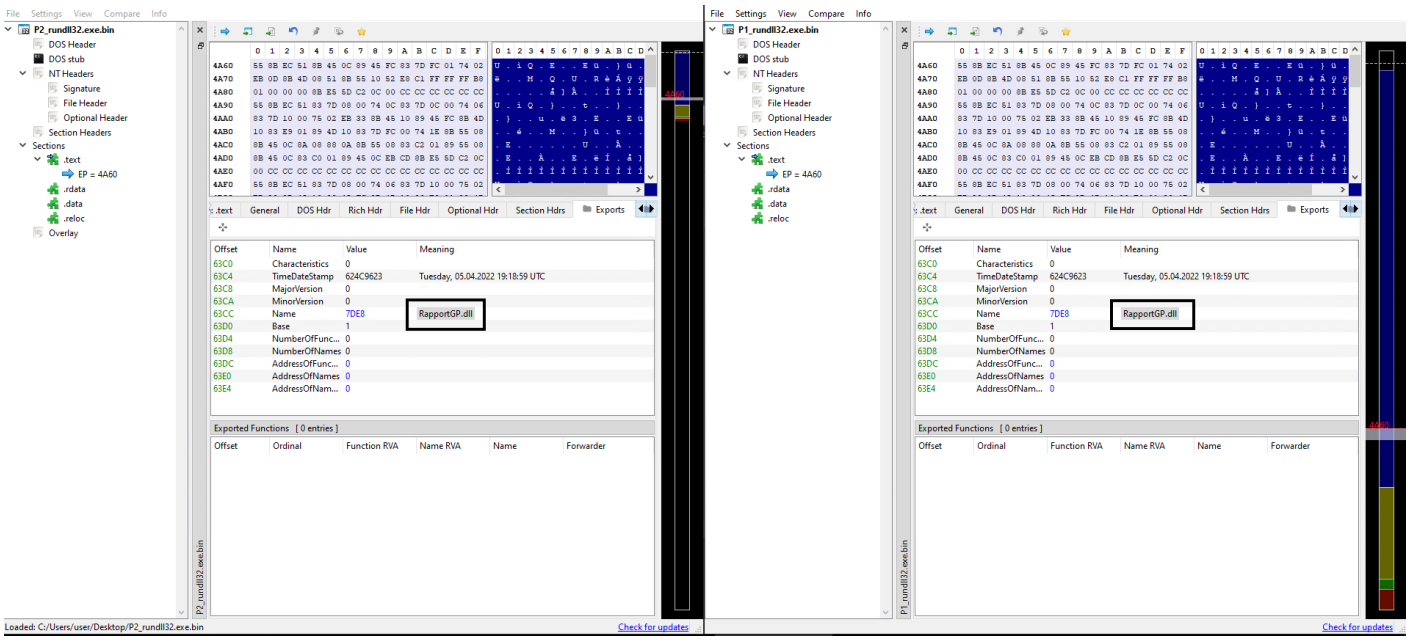
PEStudio showing the payload's metadata

We analyzed the payload binary and noticed that this payload is responsible for communicating with BumbleBee's C2 server:

In the strings we can see the C2 server's IP address and port

Both DLL payloads have the same internal name "RapportGP.dll." An interesting point regarding the payloads internal name is that there is a legitimate DLL named "RapportGP.dll" that is part of a "Trusteer

Ltd” product from a computer security division of IBM.

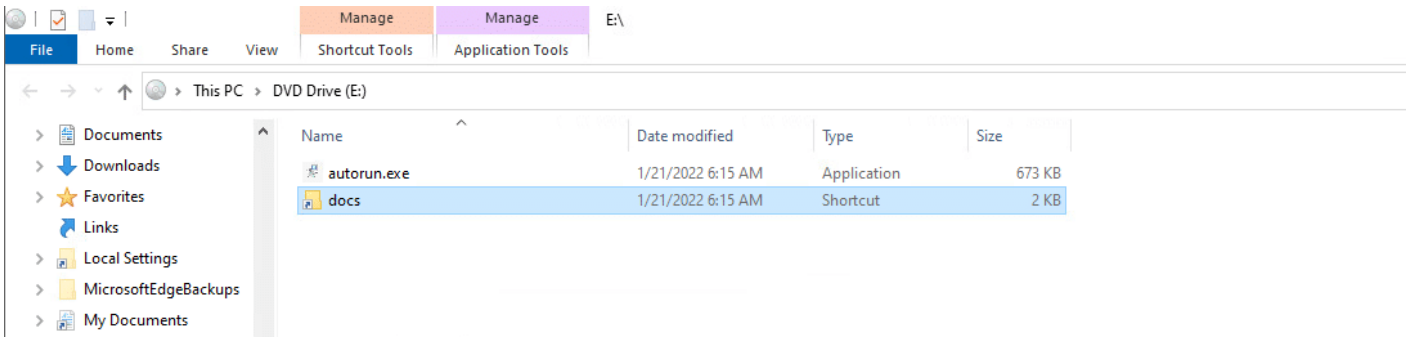


Payloads internal name and TimeDateStamp

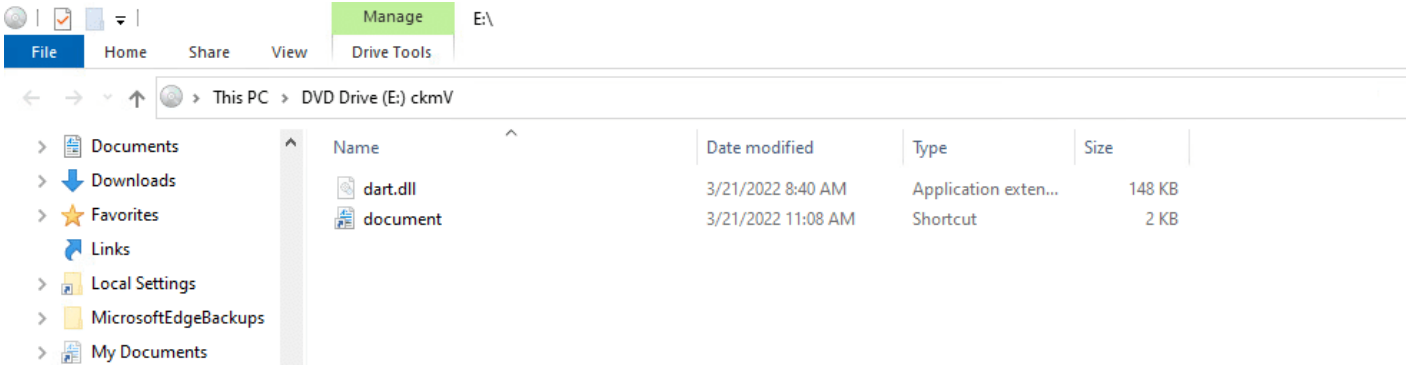
Final notes

BumbleBee threat actors are not the first to change the initial access method from malicious office documents to malicious ISO image files. The ISO image file abuse was also seen a few years ago, but in recent months, we have observed an increase in “ISO campaigns.”

Different threat actors abuse ISO image files to deliver their payloads. For example, BazarISO deploys Bazarloader, and IcedID started to use ISO image files instead of MalDocs like in the two examples below.



Documents-17.iso (Bazarloader)



Invoice_pdf_1.iso (IcedID)

In most of the cases, we've seen that during different IR cases, the campaigns escalated to full-blown ransomware attacks. We believe that IAB groups work and collaborate with ransomware affiliates like CONTI, LockBit, AvosLocker, and more. For example, we observed an IcedID infection that leads to CONTI ransomware attack (Shelob Moonlight)..

The Orion team is constantly monitoring BumbleBee and the IAB group's activities closely and analyzing them to better understand their motivation. As we learn more, we will publish our findings and artifacts to share additional insights for BumbleBee infection to ransomware post-attack chain.

We're expecting to see more malware campaigns that will use the ISO delivery method in the near future. So, stay vigilant.

As a final note, we'd like to share these indicators of compromise with you.

Indicators of compromise:

BumbleBee payload

```
88F5AE9691E6BCDD4065A420EAFAF3E3AA32C69605BF564A42FFD8ECD25C9920

4a49e2f06ba48d3a88fdeb83fb8021f3d165535e8ea5319b16a7ebe4da9c0751

08cd6983f183ef65eabd073c01f137a913282504e2502ac34a1be3e599ac386b

186145f84ed6a473ec6bc4afa66bff156057888938793b12afd17659041ddbba

4063fab9176db3960fa6014173b6c7ba52f19424887f5a6205ff73aa447ada61

53b3ebaa3c485772f8e6abaa0f366ef192137496a7064e015ced4e6fc204b3c8

d74a3f9b35d657516eb53d4e70582f93d22077d3e0936758cc4ef76d5171075d

8f47c3962a7c418bae71fec42bbca9524b72f8f0fd2dd81d1175138f7d20b2f7

c97b8bffcbe424cbc2a6e1135068d071c6f4e8f020fccd2db3dbee3aa80102ac
```

BumbleBee C2 server

IP: 23.82.19[.]208 Port 443

IP: 192.236.198[.]63 Port 433

IP: 45.147.229[.]177 Port 433

Cobalt Strike C2 server

hojimizeg[.]com - 45.147.228[.]197

notixow[.]com - 23.19.58[.]154

rewujisaf[.]com - 142.234.157[.]176