Emotet Evolves With new Wi-Fi Spreader

binarydefense.com/emotet-evolves-with-new-wi-fi-spreader/

February 7, 2020

Emotet is a highly sophisticated trojan that typically also serves as a loader for other malware. A key functionality of Emotet is its ability to deliver custom modules or plugins that are suited for specific tasks, including stealing Outlook contacts, or spreading over a LAN. Recently, Binary Defense has identified a new loader type that takes advantage of the wlanAPI interface to enumerate all Wi-Fi networks in the area, and then attempts to spread to these networks, infecting all devices that it can access in the process (as seen in Figure 1).

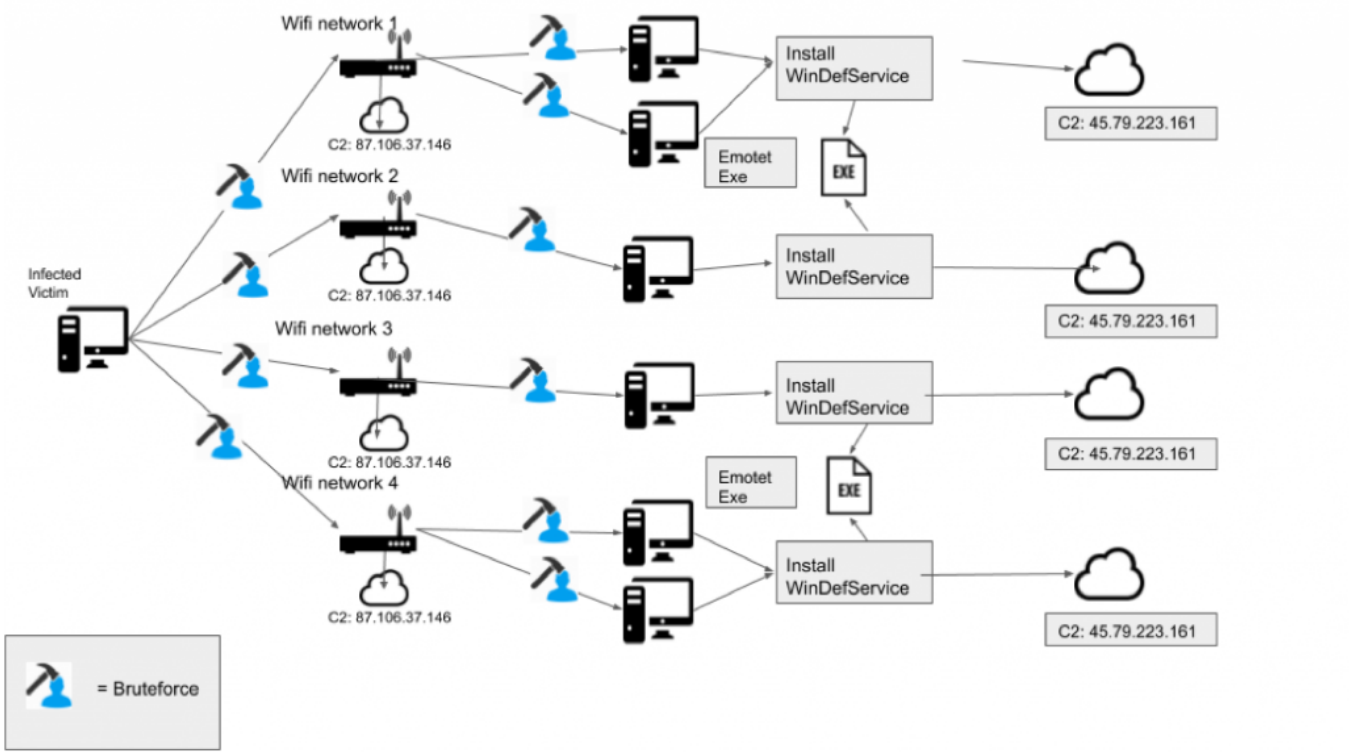

Figure 1 Wi-Fi spreader overview

Emotet's Protocol

Before getting into the analysis, Emotet's Protocol should be explained. The protocol is based on Google's Protobufs to serialize data sent to and from the server. The documentation for Protobufs shows that data is defined through the use of protocol buffer message types in a .proto file. While Emotet uses several of these "messages", the one this analysis will be focusing on is the Deliverable message, which is used when the server sends a response containing data to be loaded/executed. The message is as follows:

```
message Deliverable {
   required int32 ID = 1;
    required int32 executeFlag = 2;
    required bytes blob = 3;
}
```

In the above protobuf message, ID is the module ID, blob is the binary data, and executeFlag determines how the binary loaded. The executeFlag field can be one of the following:

- 1: Reserved for payloads and standalone executables, like Trickbot. Drops in C:\ProgramData and executes.
- 2: Like Type 1, but duplicates user's token.
- 3: Loads the binary into memory. Typically used by modules, as they are mainly DLLs which can be easily loaded into memory.

Initial Binary

The initial binary for the Wi-Fi spreader arrives on a system with module ID: 5079 and executeFlag: 1, which means it will be dropped into C:\ProgramData before executing, like loader updates or other malware, like TrickBot. This initial binary is a self-extracting RAR file which contains the 2 binaries used for Wi-Fi spreading.

```
Scanning the drive for archives:
1 file, 556318 bytes (544 KiB)

Extracting archive: 9.file
--
Path = 9.file
Type = zip
Physical Size = 556318
Embedded Stub Size = 156672
Comment = ;Ðàñïîëîæåíîé íèæå êîììåíòàðèé ñîäåðæèò êîìàíäû SFX-ñöåíàðèÿ

Setup=worm.exe
Silent=1
Overwrite=1


Everything is Ok

Files: 2
Size:       936448
Compressed: 556318
```

Figure 2 RAR Extraction

The self-extracting RAR contains 2 files: service.exe and worm.exe. Worm.exe is configured as the setup file, as seen in Figure 2, for the self-extracting RAR, which means worm.exe will execute automatically once the RAR file has unpacked itself.

Worm.exe

Worm.exe is the main executable used for spreading. This executable has a timestamp of 04/16/2018 and was first submitted to VirusTotal on 05/04/2018. The executable with this timestamp contained a hard-coded IP address of a Command and Control (C2) server that was used by Emotet. This hints that this Wi-Fi spreading behavior has been running unnoticed for close to two years. This may be in part due to how infrequently the binary is dropped. Based on our records, 01/23/2020 was the first time that Binary Defense observed this file being delivered by Emotet, despite having data going back to when Emotet first came back in late August of 2019.

Operation of Worm.exe

Upon startup of Worm.exe, the first action it takes is to copy the service.exe string to a variable that will be used during file spreading. Next, it steps into the main loop and immediately begins profiling the wireless network using wlanAPI.dll calls in order to spread to any networks it can access, as seen in Figure 3.

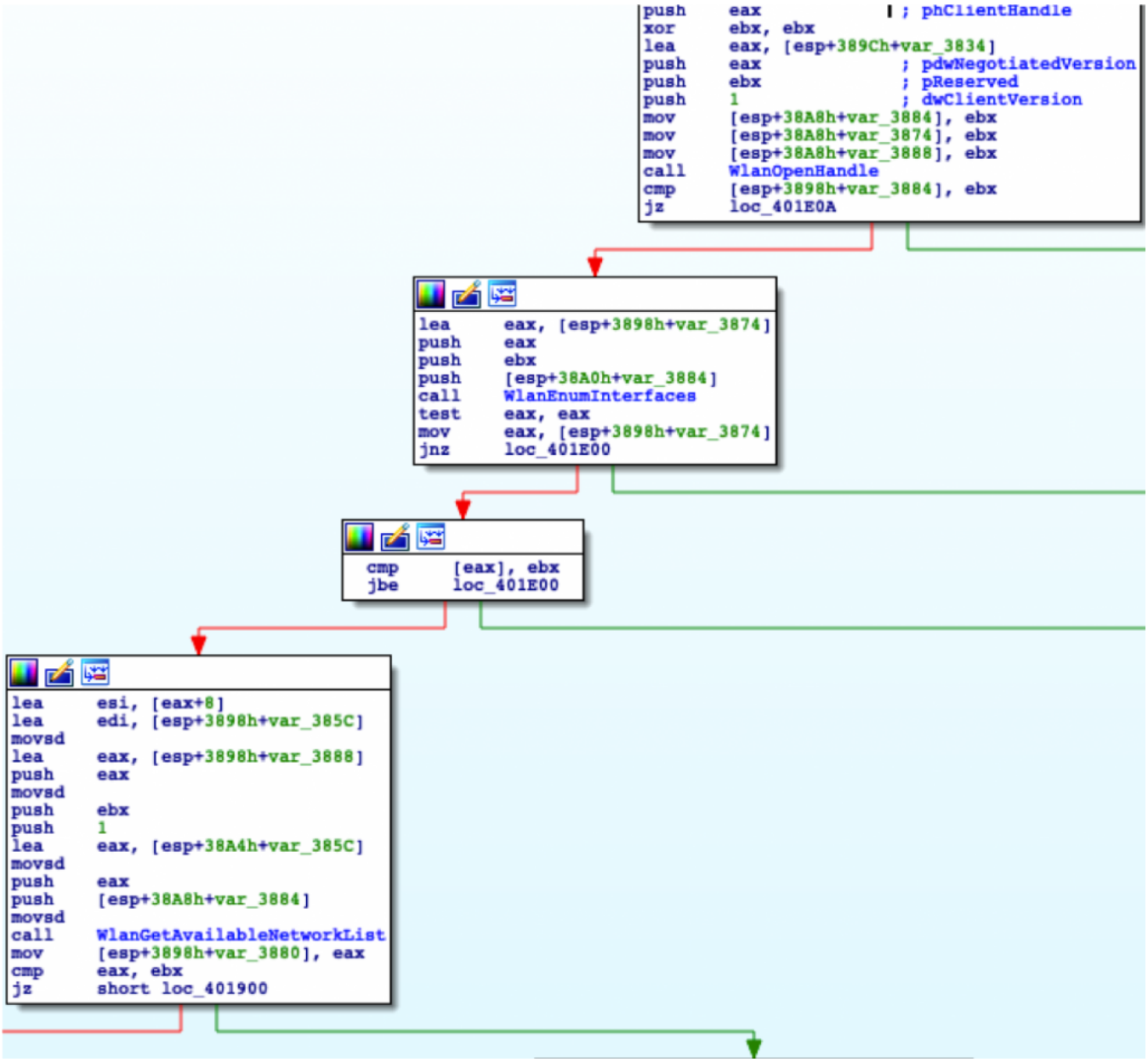

Figure 3 Worm.exe Start

The use of purely wlanAPI.dll calls for network profiling makes sense; it is one of the libraries used by Native Wi-Fi to manage wireless network profiles and wireless network connections. A possibly unintended consequence of this is that researchers running this malware in VMs/automated sandboxes will not see any of worm.exe's spreading behavior if a Wi-Fi card is not present. This can be bypassed partially by starting the WLAN auto config service in Windows 10; however, this only allows opening a handle to interface with. The subsequent calls will crash the program at that point, so the bypass is not recommended.

Worm Information Gathering

As seen in Figure 3, once a handle has been obtained, WlanEnumInterfaces is called. This function enumerates all Wi-Fi devices currently enabled on the local computer, which it returns in a series of structures. These structures contain all the information relating to the Wi-Fi device, including the device's GUID and description.

Using the first available Wi-Fi device, WlanGetAvailableNetworkList is called to obtain a list of all available networks. Interestingly, this has the dwFlag parameter set to WLAN_AVAILABLE_NETWORK_INCLUDE_ALL_ADHOC_PROFILES, which is an invalid parameter on Windows XP with SP3 and Wireless LAN API for Windows XP with SP2, meaning this malware cannot properly run on Windows XP.

Next, the malware begins profiling the network, saving the following information to a buffer:

SSID: %s
SIGNAL: %d
SECURITY: [WPA|WPA2|UNKNOWN|WEP|OPEN]
encryption: [UNKNOWN|WEP104|CCMP|TKIP|WEP40|NONE]
Note: [Current Connecting| OR
WLAN_AVAILABLE_NETWORK_HAS_PROFILE| OR
WLAN_AVAILABLE_NETWORK_CONSOLE_USER_PROFILE]

This information is gathered for every available network in the list of networks. Then, a massive switch case used to handle connection notifications is passed to the Wi-Fi device using WlanRegisterNotification. Finally, the network authentication method is obtained and encryption is parsed again. This can be one of the following:

- WPA2PSK
- WPAPSK
- UNKNOWN
- WEP
- OPEN

The encryption method can be one of the following:

- TKIP
- NOEN (None misspelled)
- WEP
- AES

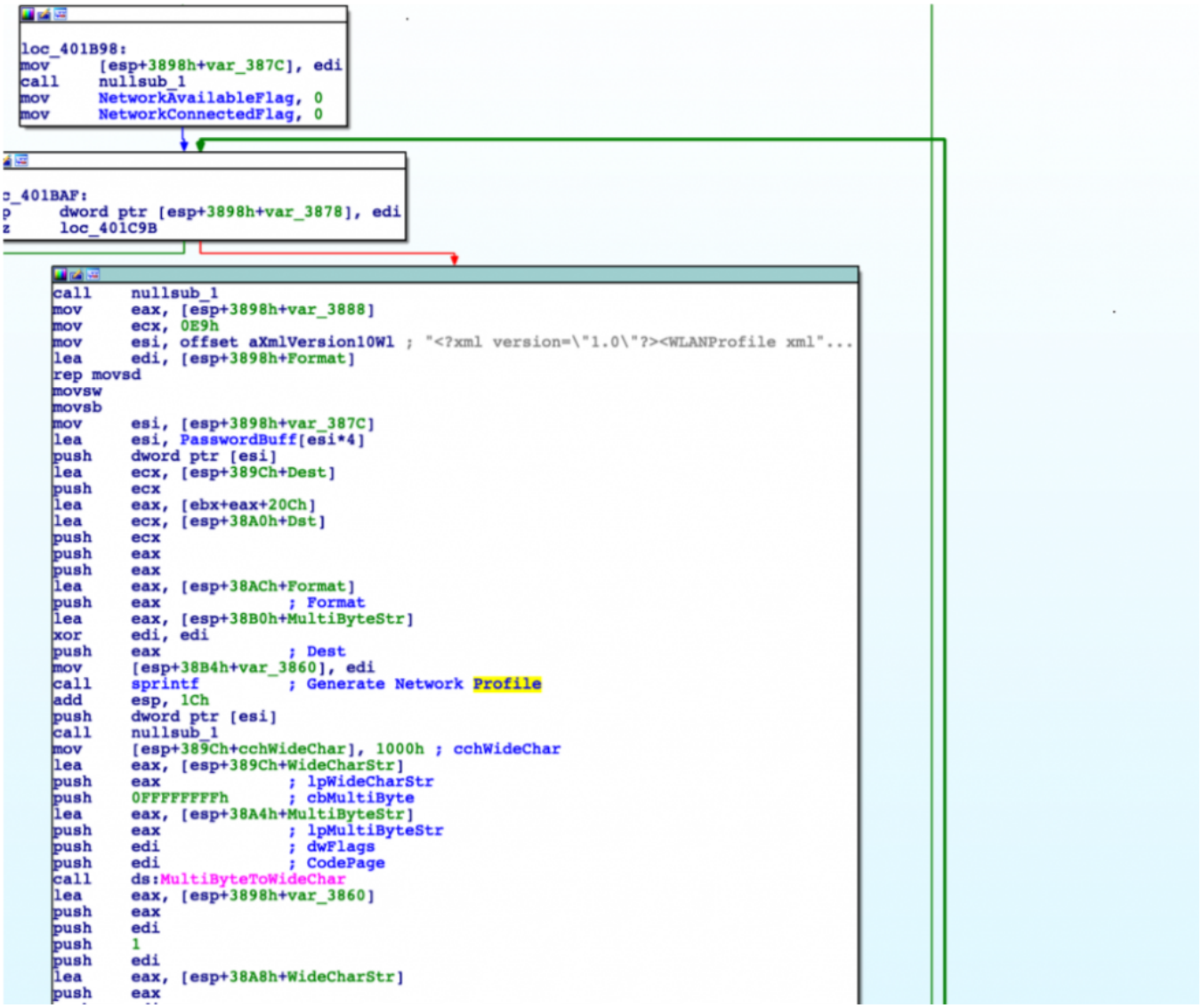

Figure 4 Worm Generating Network Profile

Worm Connection Brute-Forcing

As seen in Figure 4, once each the information for each network has been obtained, the malware moves into the connection, brute-forcing loops. The first step is to zero out two important flags and then to use the data obtained to fill in the below template that will be used to create the Network Profiles:

```
<?xml version=”1.0" encoding=”UTF-8"?>
    <WLANProfile xmlns=”http://www.microsoft.com/networking/WLAN/profile/v1">      <name>%s</name>
    <SSIDConfig>
        <SSID>
            <name>%s</name>
        </SSID>
    </SSIDConfig>
    <connectionType>ESS</connectionType
<connectionMode>auto</connectionMode>
    <MSM>
        <security>
            <authEncryption>
                <authentication>%s</authentication>
                <encryption>%s</encryption>
                <useOneX>false</useOneX>
            </authEncryption>
            <sharedKey>
```

```
            <keyType>passPhrase</keyType>
            <protected>false</protected>
            <keyMaterial>%s</keyMaterial>
        </sharedKey>
    </security>
</MSM>
</WLANProfile>
```

In the keyMaterial field, a password obtained from one of two internal password lists is used. Next, the profile is set and a connection is attempted.

- If the connection results in “wlan_notification_acm_network_available”, NetworkAvailableFlag is set to 1.
- If the connection results in “wlan_notification_msm_connected”, NetworkAvailableFlag and NetworkConnectedFlag are both set to 1.
- If the connection is not successful, NetworkAvailableFlag and NetworkConnectedFlag are both set to 0, and the function loops after moving to the next password in the password list.

If the connection is successful, the malware sleeps for 14 seconds before sending an HTTP POST to its Command and Control (C2) server at 87.106.37[.]146 on port 8080. The resource requested is a static, hard-coded value:
87.106.37[.]146:8080/230238982BSBYKDDH938473938HDUI33/index.php

The data contained in the POST is:
c=<WirelessNetworkConnected>:<Password>

Once a connection is established with the Wi-Fi network, worm.exe begins enumerating users and attempting to brute-force passwords for all users on the network.

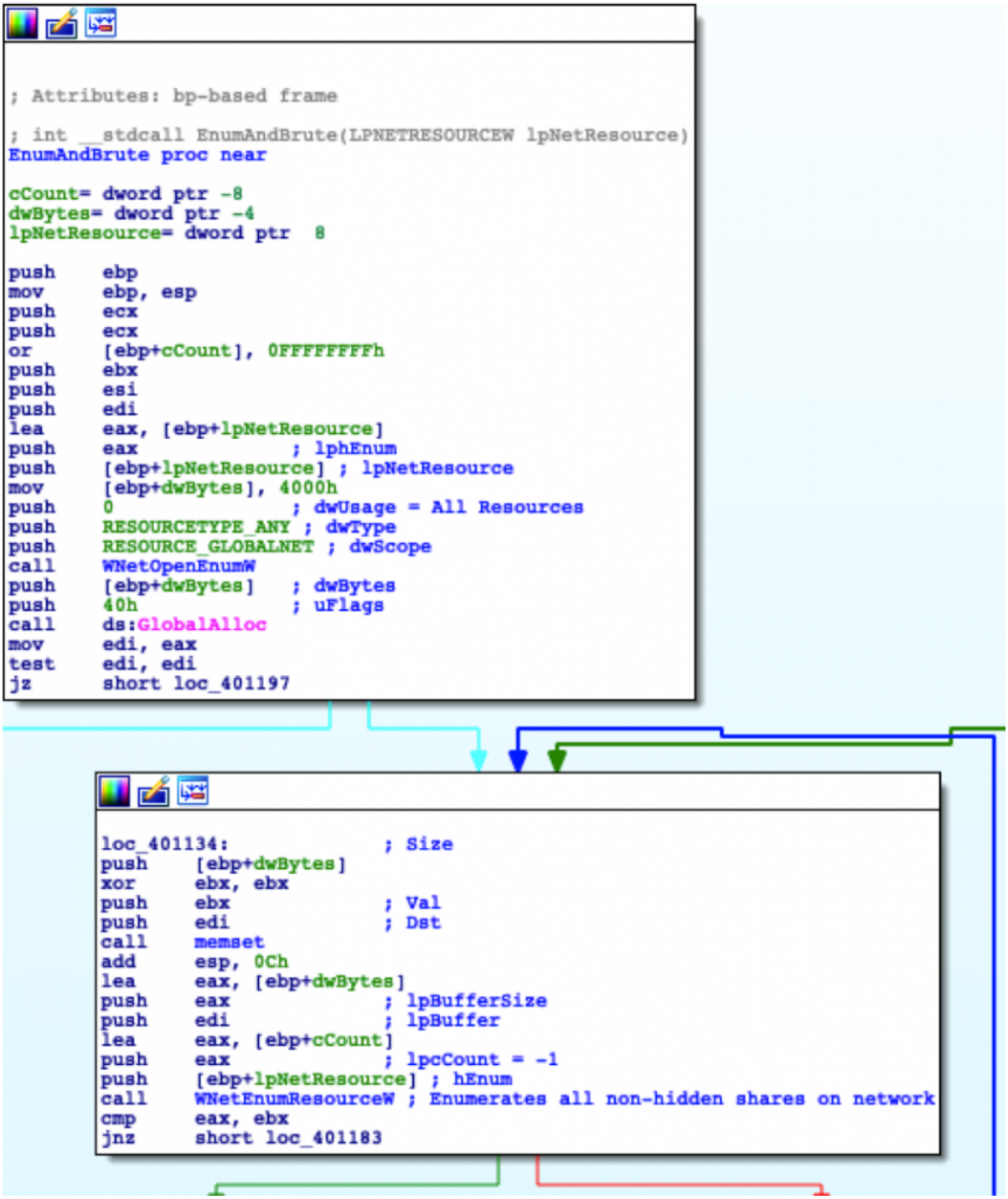

Figure 5 Enumerating all non-hidden shares

Worm User Brute-Forcing

With the infected victim now connected to a new network, the malware begins enumerating all non-hidden shares on the network, as seen in Figure 5. Once shares have been discovered, the malware attempts to connect to the IPC$ share for the network resources. Using IPC$, it attempts to enumerate all users connected to the network resource. Using the second password list contained in the malware, the malware attempts to then brute-force its way into all users enumerated, saving each successful attempt to 2 buffers: one for the username and one for the password.

If it is unable to guess passwords for any users, it pivots to attempting to brute-force the “Administrator” account for the network resource.

If either of these brute-force attempts are successful, it then moves onto the spreader function.

Spreader

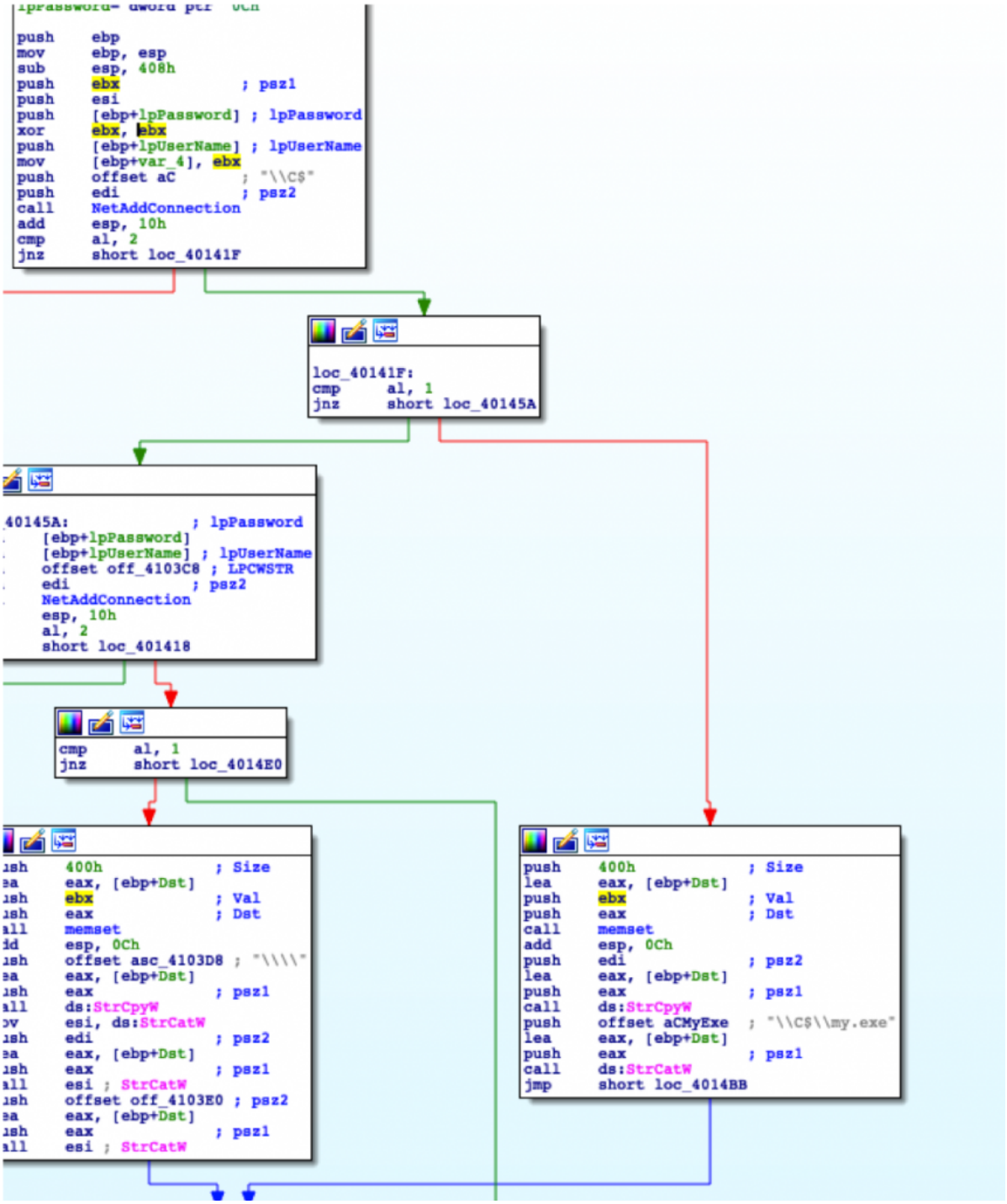

Figure 6 Connecting and generating "my.exe"

With buffers containing either a list of all usernames successfully brute-forced and their passwords, or the Administrator account and its password, worm.exe can now begin spreading service.exe to other systems.

The malware first attempts to gain access to the C$\\ share for the connected network resource. This gives it access to the C drive of the specified username. From there, it drops service.exe as my.exe in C:\\, as seen in Figure 6. Additionally, it adds a new service with the following information using the recently dropped my.exe.

Binary Path Name: C:\\my.exe
Desired Access: SERVICE_ALL_ACCESS
Display Name: WinDefService
Service Name: Windows Defender System Service

It then starts this service, executing service.exe as my.exe on a remote system.

Service.exe

Service.exe is the infected payload installed on remote systems by worm.exe. This binary has a PE timestamp of 01/23/2020, which was the date it was first found by Binary Defense. This binary is installed as a service called WinDefService by worm.exe.

Main Thread

Service.exe's startup is simple. It has a ServiceMain setup, which calls StartServiceCtrlDispatcher to connect the main thread of the service process to the service control manager, meaning the main thread is run when the service is executed.

In the main thread, service.exe starts a new thread, which serves two purposes:

1. Communicate back to a new C2 when the service is installed.
2. Drop and execute the Emotet binary embedded in service.exe.

Service Communication

The first action service.exe takes is to open a connection to 45.79.223[.]161:443 and send a request to /09FGR20HEU738LDF007E848F715BVE.php. Although the connection to the server uses port 443, which is normally used for Transport Layer Security (TLS) encrypted communications, the connection is unencrypted HTTP. When opening this connection, service.exe sets the following flags:

- 0x80000000 – INTERNET_FLAG_RELOAD
- 0x04000000 – INTERNET_FLAG_NO_CACHE_WRITE
- 0x00040000 – INTERNET_FLAG_NO_AUTH
- 0x00008000 – INTERNET_FLAG_IGNORE_REDIRECT_TO_HTTP
- 0x00004000 – INTERNET_FLAG_IGNORE_REDIRECT_TO_HTTPS
- 0x00002000 – INTERNET_FLAG_IGNORE_CERT_DATE_INVALID
- 0x00001000 – INTERNET_FLAG_IGNORE_CERT_CN_INVALID
- 0x00000400 – INTERNET_FLAG_HYPERLINK
- 0x00000200 – INTERNET_FLAG_NO_UI
- 0x00000100 – CACHE_ENTRY_ACCTIME_FC

The resource requested is hardcoded and static. This IP is currently an active Emotet Command and Control server (C2). The data contained in the request is:

"c=installed"

This notifies the server that service.exe has been properly installed. As both this request and the request sent by worm.exe are not sent over TLS/SSL, a Suricata rule is included in this analysis.

```
lea     eax, [ebp-168h]
movaps  xmmword ptr [ebp-120h], xmm0
push    eax             ; lpStartupInfo
push    0               ; lpCurrentDirectory
push    0               ; lpEnvironment
push    0               ; dwCreationFlags
push    0               ; bInheritHandles
push    0               ; lpThreadAttributes
push    0               ; lpProcessAttributes
lea     eax, [ebp-110h]
push    eax             ; lpCommandLine
push    0               ; lpApplicationName
call    ds:CreateProcessA
```

Figure 7. Flags used to create the new Emotet process

Emotet Process

After service.exe has installed itself and communicated back to the C2, it begins dropping the embedded Emotet executable. First, it obtains the path to %TEMP%, and concatenates "setup.exe" to the end of it. Next, it enters a function which locates the embedded executable by specifying a hard-coded buffer offset. Once the embedded executable is located, it creates a new file at %Temp%\\setup.exe and writes the embedded executable to this file. It then calls CreateProcess in order to create a new process running the Emotet Executable as seen in Figure 7.

Closing

With this newly discovered loader-type used by Emotet, a new threat vector is introduced to Emotet's capabilities. Previously thought to only spread through malspam and infected networks, Emotet can use this loader-type to spread through nearby wireless networks if the networks use insecure passwords. Binary Defense's analysts recommend using strong passwords to secure wireless networks so that malware like Emotet cannot gain unauthorized access to the network. Detection strategies for this threat include active monitoring of endpoints for new services being installed and investigating suspicious services or any processes running from temporary folders and user profile application data folders. Network monitoring is also an effective detection, since the communications are unencrypted and there are recognizable patterns that identify the malware message content.

IOCs

Name	Hash	C2	Additional IOCs
9.file	865cf5724137fa74bd34dd1928459110385af65ffa63b3734e18d09065c0fb36	87.106.37.146:8080, 45.79.223.161:443	
Worm.exe	077eadce8fa6fc925b3f9bdab5940c14c20d9ce50d8a2f0be08f3071ea493de8	87.106.37.146:8080	
Service.exe	64909f9f44b02b6a4620cdb177373abb229624f34f402335ecdb4d7c8b58520b	45.79.223.161:443	WinDefService %Temp%//setup.exe

Suricata:

- *alert http $HOME_NET any <> 87.106.37.146 8080 (msg: "BDS BACKDOOR Emotet Wi-Fi spreader likely";content:"POST";http_method;content:"/230238982BSBYKDDH938473938HDUI33/index.php";http_uri;classtype:backdoor-activity;sid:1;rev:1;)*
- *alert http $HOME_NET any <> 45.79.223.161 443 (msg: "BDS BACKDOOR Emotet Wi-Fi spreader likely";content:"POST";http_method;content:"/09FGR20HEU738LDF007E848F715BVE.php";http_uri;classtype:backdoor-activity;sid:2;rev:1;)*

Yara:

```
rule Emotet_Wifi_Worm {

 meta:

  title = "Emotet Wi-Fi spreader, Worm.exe"

    author = "james.quinn@binarydefense.com"

  strings:

    $string1 = " NOTE : WLAN_AVAILABLE_NETWORK_HAS_PROFILE"

    $string2 = "IPC$" wide

    $ParseEnc =   { 8d 04 0e ff b0 64 02 00 00 05 0c 02 00 00 50 68 ?? ?? ?? ?? e8 ?? ?? ?? ?? 83 c4 0c 68 ?? ?? ?? ?? e8 ?? ?? ?? ?? 8b 44 ?4 14 8b 84 06 6c 02 00 00 59 83 f8 01 74 ?? 83 f8 02 74 ?? 7e ?? 83 f8 05 7e ?? 83 f8 07 7f ?? 68 ?? ?? ?? ?? eb ?? 68 ?? ?? ?? ?? eb ?? 68 ?? ?? ?? ?? eb ?? 68 ?? ?? ?? ?? eb ?? 68 ?? ?? ?? ?? e8 ?? ?? ?? ?? 59 68 ?? ?? ?? ?? e8 ?? ?? ?? ?? 8b 44 ?4 14 8b 84 06 70 02 00 00 2b c3 59 74 ??   }

  condition:

    all of them and uint16(0) == 0x5A4D

}

rule Emotet_Wifi_Service {

  meta:

    title = "Emotet Wi-Fi Spreader, Service.exe"

    author = "james.quinn@binarydefense.com"

  strings:

    $string1 = "WinDefService" wide
```

```
    $string2 = “c=installed”

    $CommsFunc = { 6a 00 68 00 f7 04 84 6a 00 6a 00 68 ?? ?? ?? ?? 68 ?? ?? ?? ?? 8d 45 f4 50 56 ff 15 ?? ?? ?? ?? 8b d8 85 db 74 ??  }

  condition:

    all of them and uint16(0) == 0x5A4D

}
```

About the Author

James Quinn is a Threat Researcher and Malware Analyst for Binary Defense. When he is not working at Binary Defense, he works as a freelance malware analyst and participates in security intelligence sharing groups. James is a major contributor to research of the Emotet botnet with the Cryptolaemus security researcher group.

About Binary Defense

Binary Defense is a managed security services provider with leading cybersecurity solutions that include SOC-as-a-Service, Managed Detection & Response, Security Information & Event Management and Counterintelligence. With our human-driven, technology-assisted approach, Binary Defense is able to provide their clients with immediate protection and visibility, combating and stopping the next generation of attacks that your business faces. The company isheadquartered in Stow, Ohio at 600 Alpha Parkway.