

A Deep Dive Into IcedID Malware: Part III - Analysis of Child Processes

fortinet.com/blog/threat-research/deep-dive-icedid-malware-analysis-of-child-processes.html

July 22, 2019

Address	Hex	ASCII
00391FE8	04 FD 2C 92 DE E7 66 C3 7A 65 75 78 DD 2A 0F 00	.y,.pçfÅzeusY*..
00391FF8	EC FD 5D 6F 1B CB 86 28 8A 79 6F 2C 1C 9C BD 6F	iÿ]o.Éñ(C.yo,..%o
00392008	76 80 83 7B 73 91 00 F7 9C 4B 6B AD 69 8A 93 B2	v..{s...+Kk.i..²
00392018	C8 FE 6E 4A A6 BD 29 92 FA A2 24 4A A2 BE 6C 4D	Épnj!%}.úç\$)çK%IM
00392028	AD 89 FE 24 29 51 6C AA BB A9 96 6C EB BE 1C 24	..p\$)qIª»@.lè%.\$
00392038	4F 09 02 24 AF 01 2E 82 3C E4 35 EF 79 C8 8F 08	o..\$~...<a5ÿË..
00392048	10 20 40 7E 40 90 97 93 3F 10 20 F5 D5 D5 D5 EC	. @~@...?. ðððði
00392058	6E B2 25 7B AE 35 D7 DA B6 6C AB BB BA 6A D4 A8	n²%{*5xú!} «»°jô
00392068	51 63 8C 1A 55 35 46 D5 1F FE 7F 2B FF 3B ED 1F	Qc..u5F0..p.+ÿ;i.
00392078	5E BD BA 7B F5 EA D5 7F F8 43 F9 D5 AB 7F 18 5B	Λ½°{ðè0.øcù0«..[
00392088	BE 67 19 53 D7 5A D5 CC C9 AA E1 DC 56 5C EB 5E	%g.Sx20IÉªáuv\èA
00392098	33 6F 2B BA E6 0D 8D 8A 3F B0 6E AD 55 5B 33 2C	3o+ªe...?ªn.U[3,
003920A8	EF D5 3F 82 42 CE 3F FC BB 57 AF FE F1 9D EE 98	iô?.BÏ?ü»w`bñ.î.
003920B8	8F AF 96 5F BD FA 03 7A 2A 78 FE E3 C8 AA 2F 99	..%ú.zªxpãÈª/.
003920C8	43 6F 32 D2 1E D7 0A 63 67 6C AD 2F FD CB 3F BF	Co20.x.cg]./ÿÉ?¿
003920D8	AB 78 86 3B 9C F8 EF 5F FD B4 F4 E2 0A FF 09 56	«x.;.oi_ÿ'ôã.ÿ.V
003920E8	58 F1 87 FE C8 7A FF EA FF F9 1A D4 49 5E 00 7C	xñ.pèzyèÿü.0IÀ.Í
003920F8	02 FE 5F FE F9 5E 73 0B BF 56 1F AA 96 25 D5 2F	.p_bùAs.¿V.ª.%/
00392108	8B BF 3C C8 06 F8 27 81 7F 16 F8 A7 FC F2 A0 88	.¿<È.ª...ø\$ùò .
00392118	E0 B7 5A 5C 81 9F 6A 38 59 A9 82 7F 12 FE 24 E9	à.Z\..j8ÿ0...p\$é
00392128	24 1D 3E 83 6F 82 F9 CB 03 0F 8B 09 04 4C 08 4E	\$.>.o.ùÈ....L.N
00392138	20 F9 39 FC 5D 32 11 48 0A 4A A0 B5 28 32 78 E4	ù9ù]2.H.J µ(2xã
00392148	70 49 F8 49 96 70 E5 3C 49 86 78 81 4A 45 8B BC	pIøI.pã<I.x.JE.%
00392158	9B 34 8B 64 13 60 30 AB 81 4B A9 E4 15 02 92 A3	.4.d.0ªKèã...f
00392168	24 91 FC AB E2 AA 05 0D 60 65 E3 7F 30 BB 22 45	\$..ú«ªª...eã.0»"E
00392178	2D 85 95 28 35 02 C6 26 2D 02 FF F8 10 01 02 07	-..(5.¿&-ÿø....
00392188	BD 0B 24 0F A4 82 8D AB D3 48 12 22 1E 4E B2 09	½.\$..ª...«0H."N².
00392198	F4 10 B1 10 4A 0D 63 83 48 CE 91 DF 84 76 10 68	ò.±.J.c.HI.B.v.k
003921A8	DA 7C 85 36 39 0D 84 94 00 11 CB 8A 6A A8 32 58	Ù].69.....É.j²X
003921B8	90 9E 92 48 6D 0A A2 0F 2E 12 F2 41 88 88 1C 31	...Hm.C...òA...1
003921C8	01 7A 17 69 7B 10 44 89 36 31 04 8A BA 3C AC 98	.z.i{.D.61...°<.
003921D8	0B 3B 34 6A A6 A8 C6 F1 47 DF 78 0C 95 23 FD 53	.;4j!ªñGBx.ª#ÿS
003921E8	25 34 AF 11 4A 84 CD AA 45 6D 40 FD 06 40 F1 80	%4-.J.ÿªEm@ÿ.0ñ.
003921F8	EE 8A 12 FD 83 EF 08 69 D2 AF A8 5F 4C 96 4B 18	î..ÿ.î.í0`_L.K.

Magic number

Decrypted yxuvgoshcb.dat

Threat Research

By Kai Lu | July 22, 2019

FortiGuard Labs Threat Analysis Report Series

In [Part II](#) of this blog series, we identified three child processes that were created by the IcedID malware. In Part III below, we'll provide a deep analysis of those child processes.

Let's get started!

0x01 Child process A (entry offset: 0x168E)

This first child process is primarily responsible for performing web injection in browsers and acting as a proxy to inspect and manipulate traffic. It can also hook key functions in browsers.

The following is the pseudo code of the entry point.

Figure 1. The pseudo code of the entry point in the trampoline code

In this function, the process first unhooks the `RtlExitUserProcess` API and then loads a number of dynamic libraries. The function `sub_0x1A9F()` is the core function.

Figure 2. The core function `sub_0x1A9F()`

Here's a list of the key functionalities of this function.

1. Build a C2 server list
2. Create a thread to set IPC with file mapping technique
3. Create a thread and then call the `QueueUserAPC` function to add a user-mode asynchronous procedure call (APC) object to the APC queue of the specified thread. In APC, it can read the DAT config file, decrypt it with an RC4 key, and then decompress the data as follows.

Figure 3. The decrypted web injection DAT config file

This DAT config file is used for performing web injections. It uses a Magic number, "zeus". IcedID then uses a customized algorithm to decode the content. The following is the decompressed data.

Figure 4. The decompressed data of web injection

4. Add self-signed certificate into the certificate store and then create a proxy server which is bound to 127.0.0.1 on TCP port 61420. Next, it calls the `RegisterWaitForSingleObject` function to register a WSA (Windows Socket API) event handler, then uses the socket of the initialized proxy server to handle all connect, send, and receive network requests.

Figure 5. Proxy server handles network requests

Additionally, in order to perform a MiTM attack on SSL connections, the proxy server has to generate a certificate and add it into the cert store. The following is that implementation.

Figure 6. Adding a self-signed cert into the cert store

We can also see that this `svchost.exe` child process is listening on TCP port 61420.

5. Create a thread to perform code injection into the browser. The following is the thread function of the browser code injection.

Figure 7. The browser injection function

It uses the `ZwQuerySystemInformation` function to gather a list of all current running processes. If a browser process is found, it performs code injection into the browser process and sets up a hook on the `ZwWaitForSingleObject` function. The following is the function that

checks to see if a running process is a browser process. It first generates a hash with the process name using a specified algorithm. Then, it compares the hash with the given hash of four browsers: Firefox, Edge, IE, and Chrome.

Figure 8. Checking the hash of the process name

Before performing its code injection, it first checks to see if this process is running on 64 bits by calling the `IsWow64Process` function. It then performs a code injection into the browser process, and depending on the process bits version, it calls the corresponding hook function to set up a hook on the `ZwWaitForSingleObject` function.

Figure 9. Process injection and setting up a hook in a browser

Here we will use Firefox to demonstrate how it performs its process injection and sets up a hook.

Figure 10. Process injection into Firefox

It sets up a hook on the `ZwWaitForSingleObject` API in the Firefox process as follows.

Figure 11. Hooked `ZwWaitForSingleObject` function

When Firefox calls the `ZwWaitForSingleObject` function, it jumps to the trampoline code. The entry point of trampoline code is at offset `0x1856` from the injected memory region.

Let's take a closer look at the trampoline code (offset:`0x1856`).

In this trampoline code, it first unhooks the `ZwWaitForSingleObject` API. Then it sets up a hook on the `SSL_AuthCertificateHook` API (in `nss3.dll` for Firefox.) The `nss3.SSL_AuthCertificateHook` function specifies a certificate authentication callback function that is called to authenticate an incoming certificate.

The following is the hooked `nss3.SSL_AuthCertificateHook` function.

Figure 12. The hooked `nss3.SSL_AuthCertificateHook` function

It configures the `nss3.SSL_AuthCertificateHook` function to always return `SECSuccess`.

Note that it can set up a hook for browser-specific functions depending on the type of browser. However, we won't be providing details for any other browsers in this blog.

Next, it continues to set up a hook on the `connect` API in `ws2_32.dll`. The following is the hooked `connect` API.

Figure 13. The hooked `connect` API in `ws2_32.dll`

The following is the pseudo code of the trampoline code for the hooked **`connect`** API.

Figure 14. The pseudo code of the trampoline code for the hooked connect API

Once the **connect** function returns 0 (the connection has succeeded), it sends 12 bytes of data to proxy server **127.0.0.1:61420**, which was created in this svchost.exe child process. The captured traffic is shown in Figure 15.

Figure 15. Brower sends 12 bytes of data to proxy server

The structure of these 12 bytes consists of four parts, as follows:

0x00: Unknown

0x04: Target website's IP address

0x08: Port

0x0A: Browser type

0x02 Child Process B (entry offset: 0x1E0A)

This second child process is used to communicate with the C2 server. It will attempt to send an HTTP request to the C2 server via WebSocket, as follows.

Figure 16. Requesting data from the C2 via WebSocket

It also communicates with the parent svchost.exe process using a mapping file technique. And, depending on the shared info, it may attempt to make network requests to a C2 server over SSL, and then create a new process, perform code injections, and set up a hook on the RtlExitUserProcess function.

0x03 Child Process C (entry offset: 0x10DF)

This process communicates with the parent svchost.exe process by using a mapping file technique. It is also able to perform some registry operations.

0x04 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.

0x05 Conclusion

In this series of posts, I have provided a detailed analysis of a new IcedID malware sample. The entire detailed analysis is divided into three parts. The first two part are available here: [Part I: Unpacking, Hooking, and Process Injection](#) and [Part II: Analysis of the Core IcedID Payload \(Parent Process\)](#).

IcedID is a sophisticated and complicated banking trojan that performs web injection in browsers and acts as proxy to inspect and manipulate traffic. It is designed to steal information – such as credentials – from victims and then send that stolen information to attacker-controlled servers. To accomplish this, IcedID uses a large number of hooking and process injection techniques, and it also disguises itself as several svchost.exe processes, which we examined in this deep dive analysis series.

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

Copyright © 2022 Fortinet, Inc. All Rights Reserved

[Terms of Services](#)[Privacy Policy](#)
| [Cookie Settings](#)