New 'Early Bird' Code Injection Technique Discovered

By sharon

Published: 2018-04-11 · Archived: 2026-04-05 16:32:15 UTC

This injection technique allows the injected code to run before the entry point of the main thread of the process, thereby allowing to avoid detection by anti-malware products' hooks.

Code injection is commonly used by malware to evade detection by injecting a malicious code into a legitimate process. This way the legitimate process serves as camouflage so all anti-malware tools can see running is the legitimate process and thus obfuscates the malicious code execution.

We researched a code injection technique that appeared in malware samples at the Cyberbit malware research lab. It is a simple yet powerful code injection technique. Its stealth allows execution of malicious code before the entry point of the main thread of a process, hence – it can bypass security product hooks if they are not placed before the main thread has its execution resumed. But before the execution of the code of that thread, the APC executes.

Click to watch code injection video

We saw this technique used by various malware. Among them – the "TurnedUp" backdoor written by APT33 – An Iranian hackers group, A variant of the notorious "Carberp" banking malware and by the DorkBot malware.

The malware code injection flow works as follows:

1. Create a suspended process (most likely to be a legitimate windows process)
2. Allocate and write malicious code into that process
3. Queue an asynchronous procedure call (APC) to that process
4. Resume the main thread of the process to execute the APC

Hooks are code sections that are inserted by legitimate anti-malware products when a process starts running. They are placed on specific Windows API calls. The goal of the hooks is to monitor API calls with their parameters to find malicious calls or call patterns.
In this post, we explain how APC execution flow works within a resume of a suspended process.

This code injection technique can be drawn like this:


code injection diagram

Synopsys of Technical Analysis of Early Bird Code Injection Technique

While analyzing samples at our lab, we came across a very interesting malware sample (SHA256: 9173b5a1c2ca928dfa821fb1502470c7f13b66ac2a1638361fda141b4547b792)

It starts with the .net sample deobfuscating itself, then performing process hollowing and filling the hollowed process with a native Windows image. The native Windows image injects into the explorer.exe process. The payload inside explorer.exe creates a suspended process – svchost.exe and injects into it. The sample consists of three different injection methods (We consider process hollowing to be an injection technique as well). The SHA256 of the payload inside svchost.exe is c54b92a86c9051172954fd64573dd1b9a5e950d3ebc581d02c8213c01bd6bf14. As of 20 March 2018, this payload was signed by only 29 out of 62 anti-malware vendors. The original sample, which dates back to 2014, was signed by 47 out of 62 vendors.

While the process hollowing and the second injection into explorer.exe are trivial, the 3rd technique caught our attention. Let's have a look at the debugger before the injection to svchost.exe happens.

Figure 1 – A suspended svchost.exe process is created


Early Bird Code Injection - suspended svchost.exe process is created

At this point the malware creates a suspended svchost.exe process. Common legitimate Windows processes are among malwares' favorite choices. svchost.exe is a Windows process designated to host services.

After creating the process, the malware allocates memory in it and writes a code in the allocated memory region. To execute this code, it calls NtQueueApcThread to queue an asynchronous procedure call (APC) on the main thread of svchost.exe. Next, it calls NtResumeThread to decrease the suspend count of that thread to zero, consequently the main thread of svchost.exe will resume execution – if this thread in alertable state, the APC will execute first.

Figure 2 – Queuing an APC to svchost.exe **main thread and resuming the thread**


Early Bird Code Injection - Queuing an APC to svchost.exe main thread and resuming the thread

When queuing an APC to a thread, the thread must be in an alertable state in order for that APC to execute. According to the Microsoft documentation:

"When a user-mode APC is queued, the thread to which it is queued is not directed to call the APC function unless it is in an alertable *state. A thread enters an* alertable *state when it calls the SleepEx, SignalObjectAndWait, MsgWaitForMultipleObjectsEx, WaitForMultipleObjectsEx, or WaitForSingleObjectEx function"*

But the thread has not even started its execution since the process was created in a suspended state. How does the malware "know" that this thread will be alertable at some point? Does this method work exclusively on svchost.exe or will it always work when a process is created in a suspended state?

To check this out, we patched the malware so it will inject to other processes of our choice and witnessed it also working with various other processes. We went further to research what is going on when a main thread is resumed after its process is created in a suspended state.

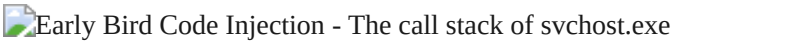
By putting a breakpoint on the call to NtQueueApcThread, we can see the APC address on svchost.exe is at 0x00062f5b. We attached a debugger to this process and put a breakpoint on that address. Here is what the svchost.exe process looks like at 0x000625fb (start address of the APC).

Figure 3 – The APC starts execution at svchost.exe


Early Bird Code Injection - The APC starts execution at svchost.exe

Let's look at the call stack (figure 4) after resuming the thread. Our breakpoint on 0x00062f5b was hit as expected:

Figure 4 – The call stack of svchost.exe


Early Bird Code Injection - The call stack of svchost.exe

We first have to note that every user-mode thread begins its execution at the LdrInitializeThunk function. When we look at the bottom of the call stack we see that LdrpInitialize, which is called from LdrInitializeThunk (figure 5), was called. We trace into LdrpInitialize and see that it jumps to the function _LdrpInitialize (figure 6). Inside _LdrpInitialize, we see a call to NtTestAlert (figure 7) which is a function responsible for checking if there is an APC queued to the current thread – if there is one – it notifies the kernel. Before returning to user-mode, the kernel prepares the user-mode thread to jump to KiUserApcDispatcher which will execute the malicious code in our case.

Figure 5 – 0x76e539c1 led to an address after the call from LdrpInitialize


Early Bird Code Injection - 0x76e539c1 led to an address after the call from LdrpInitialize

Figure 6 – Inside LdrpInitialize there is a jump to _LdrpInitialize


Early Bird Code Injection - Inside LdrpInitialize there is a jump to _LdrpInitialize

Figure 7 – Inside _LdrpInitialize there is a call to NtTestAlert


Early Bird Code Injection - Inside _LdrpInitialize there is a call to NtTestAlert

We can see evidence that this APC was executed by KiUserApcDispatcher (figure 8), by looking at the call stack again, and see that the return address of 0x00062f5b is 0x76e36f9d – right after the call from KiUserApcDispatcher.

Figure 8 – KiUserApcDispatcher executed the APC


Early Bird Code Injection - KiUserApcDispatcher executed the APC

To sum it up, the execution flow that led to the execution of the APC is:
LdrInitializeThunk → LdrpInitialize → _LdrpInitialize → NtTestAlert → KiUserApcDispatcher

An important note about this injection method is that it loads the malicious code in a very early stage of thread initialization, before many security products place their hooks – which allows the malware to perform its malicious actions without being detected.

In the wild

This technique was seen in other samples in our lab (SHA 256):

165c6f0b229ef3c752bb727b4ea99d2b1f8074bbb45125fbd7d887cba44e5fa8

368b09f790860e6bb475b684258ef215193e6f4e91326d73fd3ab3f240aedddb

a82c9123c12957ef853f22cbdf66561949566620d486a4b37f5d2767f8d33dc4d

d17dce48fbe81eddf296466c7c5bb9e22c39183ee9828c1777015c1652919c30

5e4a563df904b1981d610e772effcb005a2fd9f40e569b65314cef37ba0cf0c7

The last two samples in this list are the most recent, dated from 31 October 2017. These samples are the "TurnedUp" backdoor written by the Iranian hackers group APT33. Figure 9 is a screenshot from the last sample which shows the use of this technique in an injection to rundll32.exe – a legitimate Windows process used to run exported functions from dll files.

Figure 9 – A suspended rundll32.exe process is created, followed by injection using QueueUserApc


Early Bird Code Injection - A suspended rundll32.exe process is created, followed by injection using QueueUserApc

In this sample, the APC is used to maintain persistence on the system. In figure 10 you can see where the APC starts (0x90000) inside rundll32.exe. Going just a bit further reveals that the malware will write a key to the Windows registry to maintain persistence by executing ShellExecuteA with cmd and a specific command (figure 11). The cmd command is:

"/c REG ADD HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run /v RESTART_STICKY_NOTESS /f /t REG_SZ /d \"C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\StikyNote.exe\""\

In this case, if a hook on ShellExecuteA was placed after the APC was called, this 'Early Bird' call and its parameters will sneak by before the hook, hence, failing to detect an important malware behavior.

Figure 10 – APC starts execution at rundll32.exe


Early Bird Code Injection - APC starts execution at rundll32.exe

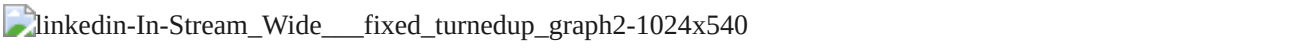
Figure 11 – The call to ShellExecuteA


Early Bird Code Injection - The call to ShellExecuteA

The third sample in the list (a82c9123c12957ef853f22cbdf66561949566620d486a4b37f5d2767f8d33dc4d) dates back to 2011 and is variant D of the notorious Carberp malware.

Cyberbit provides an Endpoint Detection and Response solution (EDR) which successfully detects the 'Early Bird' injection technique. To learn more visit the Cyberbit EDR page.

Analysis of 'Early Bird' injection automatically created by Cyberbit EDR


linkedin-In-Stream_Wide___fixed_turnedup_graph2-1024x540

Hod Gavriel is a malware analyst at Cyberbit.

Boris Erbesfeld is a principal software engineer at Cyberbit.
