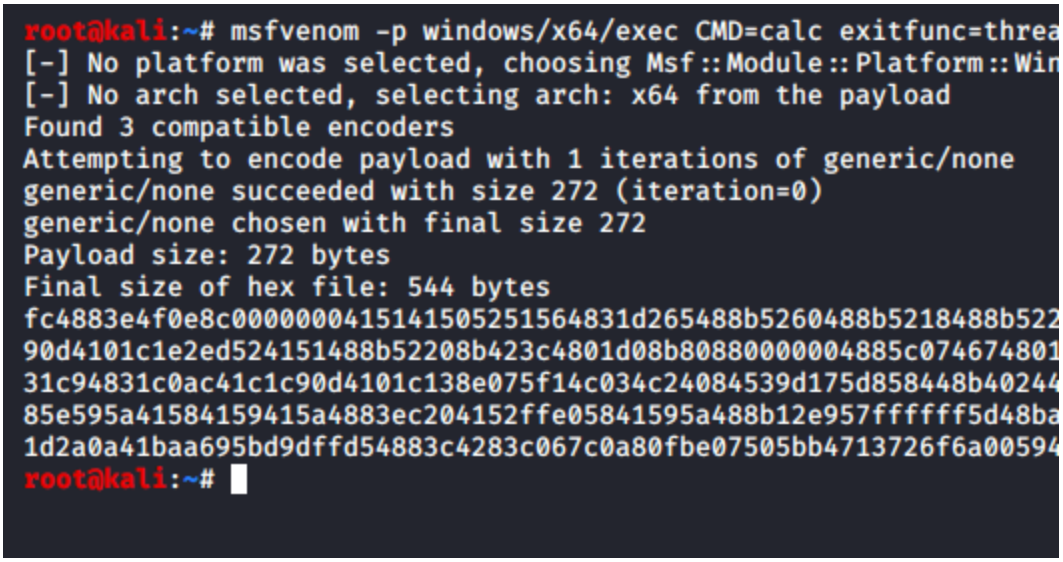
Process Injection - Part V

3xpl01tc0d3r.blogspot.com/2019/12/process-injection-part-v.html



Hello All,

Wondering where is the part IV of the process injection blog post series ?
The part IV of this series was written by Renos on Parent PID Spoofing technique. Parent PID Spoofing technique is useful for evading detection.

In this post I will cover about Asynchronous Procedure Calls (APC) Queue Process Injection technique. Parent PID Spoofing can also be used with APC Queue Process Injection. The tool can be found on my github repo.

What is Asynchronous Procedure Calls (APC) ?

This is best described by Microsoft in their documentation. Below is the short brief about APC from the Microsoft Docs.

An asynchronous procedure call (APC) is a function that executes asynchronously in the context of a particular thread. When an APC is queued to a thread, the system issues a software interrupt. The next time the thread is scheduled, it will run the APC function. An APC generated by the system is called a kernel-mode APC. An APC generated by an application is called a user-mode APC. A thread must be in an alertable state to run a user-mode APC.

Each thread has its own APC queue. An application queues an APC to a thread by calling the QueueUserAPC function. The calling thread specifies the address of an APC function in the call to QueueUserAPC. The queuing of an APC is a request for the thread to call the APC function.

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. The thread enters into alertable state when it calls any of the below mentioned functions.

- SleepEx - Suspends the current thread until the specified condition is met. Execution resumes when one of the following occurs:
    - An I/O completion callback function is called.
    - An asynchronous procedure call (APC) is queued to the thread.
    - The time-out interval elapses.

  SingnalObjectAndWait - Signals one object and waits on another object as a single operation.

  MsgWaitForMultipleObjectsEx - Waits until one or all of the specified objects are in the signaled state, an I/O completion routine or asynchronous procedure call (APC) is queued to the thread, or the time-out interval elapses. The array of objects can include input event objects, which you specify using the dwWakeMask parameter.

  WaitForMultipleObjectsEx - Waits until one or all of the specified objects are in the signaled state, an I/O completion routine or asynchronous procedure call (APC) is queued to the thread, or the time-out interval elapses.

  WaitForSingleObjectEx - Waits until the specified object is in the signaled state, an I/O completion routine or asynchronous procedure call (APC) is queued to the thread, or the time-out interval elapses.

In this APC Queue Process Injection technique 7 Windows API are used:

CreateProcess - The CreateProcess function creates a new process and its primary thread. The new process runs in the security context of the calling process.

OpenProcess - The OpenProcess function returns a handle of an existing process object.

VirtualAllocEX - The VirtualAllocEx function is used to allocate the memory and grant the access permissions to the memory address.

WriteProcessMemory - The WriteProcessMemory function writes data to an area of memory in a specified process.

OpenThread - The OpenThread function returns a handle of an existing thread object.

QueueUserAPC - The QueueUserAPC function adds a user-mode asynchronous procedure call (APC) object to the APC queue of the specified thread.

ResumeThread - The ResumeThread function decrements the thread's suspend count. When the suspend count is decremented to zero, the execution of the thread is resumed.

Overview of APC Queue Process Injection

APC Queue Process Injection is a technique used by malware authors for being stealthy as it allows execution of malicious code before the entry point of the main thread of a process.

Below are the steps followed while adding the APC Queue Process Injection technique in the tool.

Step 1:- Create a new target process in suspended state. This can be achieve by passing Create_Suspended value in dwCreationFlags parameter of CreateProcess Windows API.

Step 2:- Once the process is created obtain the handle of the target process using OpenProcess Windows API.

Step 3:- Allocate the memory space for our shellcode in the target process using VirtualAllocEX Windows API.

Step 4:- Write the shellcode in the allocated memory space using WriteProcessMemory Windows API.

Step 5:- Obtain the handle of the primary thread from the target process using OpenThread Windows API.

Step 6:- After obtaining the handle of the thread from the target process we will add a user-mode asynchronous procedure call (APC) object to the APC queue of the specified thread using QueueUserAPC Windows API which will point to the memory address of our shellcode.

Step 7:- To trigger our shellcode we will resume the suspended thread using ResumeThread Windows API.

Demo

In this demo I have used MSFVenom to generate the shellcode. You can use Donut for generating the shellcode or use any tool of your choice. For the demo purpose I will generate the shellcode in hex format.

APC Queue Process Injection

Required Parameters:

- **/ppath:-** Specify the path of the process which will be created in suspended state.
- **/path:-** Specify the path of the shellcode file.
- **/f:-** Specify the format of the shellcode that was generated.
- **/t:-** Specify the technique id

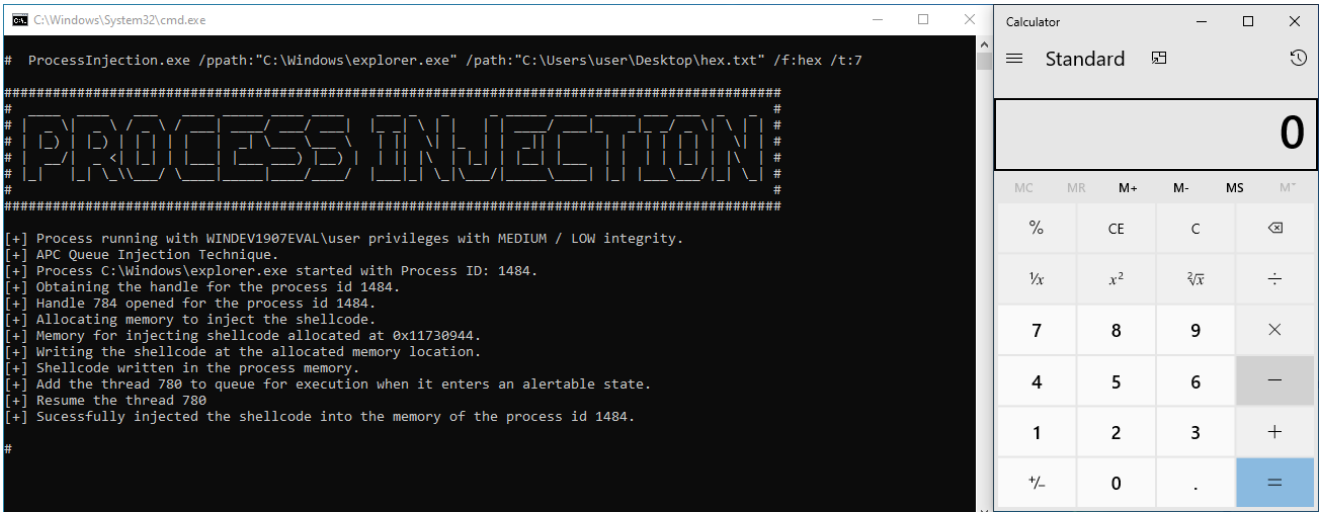
Generate the shellcode using MSFVenom.

msfvenom -p windows/x64/exec CMD=calc exitfunc=thread -b ""\x00"" -f hex

```
root@kali:~# msfvenom -p windows/x64/exec CMD=calc exitfunc=thread -b ""\x00"" -f hex
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 3 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none succeeded with size 272 (iteration=0)
generic/none chosen with final size 272
Payload size: 272 bytes
Final size of hex file: 544 bytes
fc4883e4f0e8c0000000415141505251564831d265488b5260488b5218488b5220488b7250480fb74a4a4d31c94831c0ac3c617c022c2041c1c
90d4101c1e2ed524151488b52208b423c4801d08b80880000004885c074674801d0508b4818448b40204901d0e35648ffc9418b34884801d64d
31c94831c0ac41c1c90d4101c138e075f14c034c24084539d175d858448b40244901d066418b0c48448b401c4901d0418b04884801d04158415
85e595a41584159415a4883ec204152ffe05841595a488b12e957ffffff5d48ba0100000000000000488d8d0101000041ba318b6f87ffd5bbe0
1d2a0a41baa695bd9dffd54883c4283c067c0a80fbe07505bb4713726f6a00594189daffd563616c6300
root@kali:~#
```

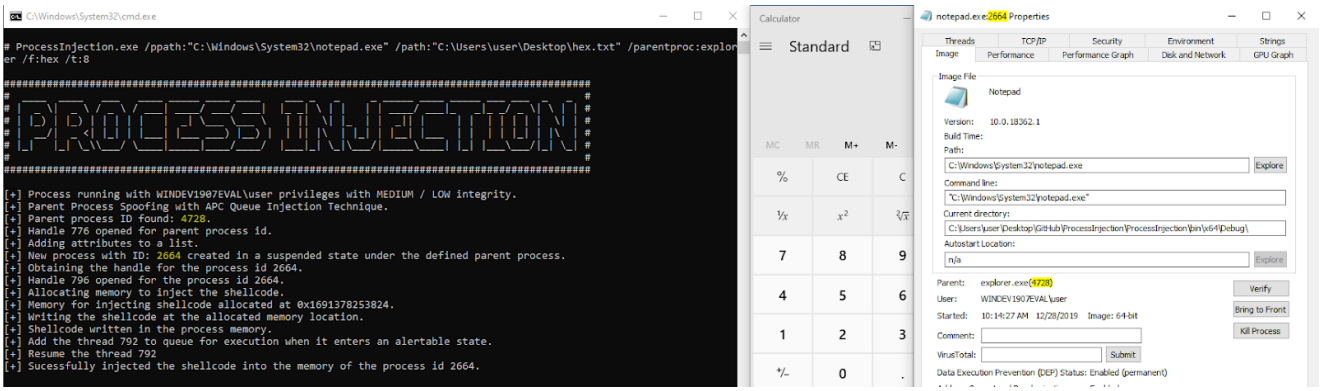
Execute the APC Queue Process Injection technique

ProcessInjection.exe /ppath:"C:\Windows\explorer.exe" /path:"C:\Users\user\Desktop\hex.txt" /f:hex /t:7



Parent PID Spoofing with APC Queue Process Injection

ProcessInjection.exe /ppath:"C:\Windows\System32\notepad.exe" /path:"C:\Users\user\Desktop\hex.txt" /parentproc:explorer /f:hex /t:8



Detection

Monitor process which are created in suspended state. Monitor Windows API calls like OpenThread, QueueUserAPC and those that can be used to modify memory within another process such as WriteProcessMemory. Also monitor process especially under (c:\Windows\System32* or c:\Windows\SysWOW64*) for abnormal behavior such as opening network connections.

References

https://ired.team/offensive-security/code-injection-process-injection/apc-queue-code-injection
https://modexp.wordpress.com/2019/08/27/process-injection-apc/
https://i.blackhat.com/USA-19/Thursday/us-19-Kotler-Process-Injection-Techniques-Gotta-Catch-Them-All.pdf
https://attack.mitre.org/techniques/T1055/

Thanks for reading the post.

Special thanks to all my friends who help / supported / motivated me for writing blogs.