Let's Learn: Deeper Dive into "IcedID"/"BokBot" Banking Malware: Part 1

Goal: Reverse engineer and analyze one of the latest "IcedID" banking malware (also known to some researchers as "BokBot") focusing on its core functionality.

> 2018-09-05 - #Emotet #malspam infection with #IcedID #bankingTrojan and #AZORult - I've focused on Emotet malspam with PDF attachments, but there's still Emotet malspam with Word attachments and still Emotet malspam with just links to the Word docs - https://t.co/9HlgztSaJK pic.twitter.com/vBeXCMEHmQ
> — Brad (@malware_traffic) September 6, 2018

Malware:
Original Packed IcedID Loader (MD5: 78930770cb81ad779958da3523fcb829)
Unpacked Injector IcedID (M5: e42d8511c6237cd22ac6bc89a2c00861)
Outline:

```
I. Background
```

II. "Emotet" Malware Campaign Spreading "IcedId" Banker

```
III. Original Packed Loader "IcedID" 32-bit (x86) Executable
IV. Unpacked Process Injector "IcedID" 32-bit (x86) Executable
V. Minimalistic Process Injection: \

Hooking Engine ZwCreateUserProcess & RtlExitUserProcess

A. IcedID "HookMain"

B. Injector CreateProcessW API Execution (dwCreationFlags=0)
C. "myZwCreateUserProcess" Hook

D. "HookRtlExitUserProcess" Function
V. Yara Signature: IcedID Injector
```

I. Background

IcedID banker first publically identified in November 2017; IBM's X-Force research team published a report claiming to have spotted this new banking malware spreading via massive spam campaigns. Compromised computers were first infected with the Emotet downloader, which then grabbed IcedID from the attacker's domain. IcedID is able to maintain persistence on infected machines, and it has targeted companies mainly in the financial services, retail, and technology sectors. IcedID operators oftentimes collaborate with other groups such as TrickBot, for example.

Additionally, I highly recommend reading Fox-IT's paper titled "Bokbot: The (re)birth of a banker." They detail that the original discovery dates back to May 2017; additionally, it is notable that the IcedID banker appears to be a continuation of the Neverquest group activity, also known internally as "Catch."

II. "Emotet" Malware Campaign Spreading "IcedId" Banker

While reviewing one of the latest malware campaign spreading the Emotet loader as it was reported by Brad, I decided to dive deeper into this banker malware sample. It is notable that this specific malware campaign was spreading IcedID banker and "AZORult" stealer subsequently.

III. Original Packed Loader "IcedID" 32-bit (x86) Executable

The original IcedID loader was obfuscated and packed by pretty interesting crypter with the following executable information with the PDB path.

```
[IMAGE_DEBUG_DIRECTORY]
0x1E1E0    0x0   Characteristics:              0x0
0x1E1E4    0x4   TimeDateStamp:                0x4AA23E03 [Sat Sep  5 10:31:31 2009
UTC]
0x1E1E8    0x8   MajorVersion:                 0x0
0x1E1EA    0xA   MinorVersion:                 0x0
0x1E1EC    0xC   Type:                         0x2
0x1E1F0    0x10  SizeOfData:                   0x42
0x1E1F4    0x14  AddressOfRawData:             0x257B8
0x1E1F8    0x18  PointerToRawData:             0x245B8
Type: IMAGE_DEBUG_TYPE_CODEVIEW

[CV_INFO_PDB70]
0x245B8    0x0   CvSignature:                  0x53445352
0x245BC    0x4   Signature_Data1:              0x11439B10
0x245C0    0x8   Signature_Data2:              0x27C2
0x245C2    0xA   Signature_Data3:              0x49F4
0x245C4    0xC   Signature_Data4:              0x6EB6
0x245C6    0xE   Signature_Data5:              0x780D
0x245C8    0x10  Signature_Data6:              0x7D7BC8B5
0x245CC    0x14  Age:                          0x1
0x245D0    0x18  PdbFileName:
c:\Sea\Eat\Steam\First\Bone\boybehind.pdb
```

IV. Unpacked Process Injector "IcedID" 32-bit (x86) Executable

After unpacking the crypter/loader portion of IcedID, one of the first notable features of IcedID is its surreptitious process injection without using suspended process but relies on hooking ZwCreateUserProcess and RtlExitUserProcess. The injector appears to have been compiled on August 13, 2018. Its size is 25 KB with three sections and two imports.

```
[IMAGE_FILE_HEADER]
0xC4       0x0   Machine:                        0x14C
0xC6       0x2   NumberOfSections:               0x3
0xC8       0x4   TimeDateStamp:                  0x5B718995 [Mon Aug 13 13:37:25 2018
UTC]
0xCC       0x8   PointerToSymbolTable:           0x0
0xD0       0xC   NumberOfSymbols:                0x0
0xD4       0x10  SizeOfOptionalHeader:           0xE0
0xD6       0x12  Characteristics:                0x103
Flags: IMAGE_FILE_32BIT_MACHINE, IMAGE_FILE_EXECUTABLE_IMAGE,
IMAGE_FILE_RELOCS_STRIPPED
```

The injector contains three sections (.text, bss, .rdata) with two imported DLL:

- SHLWAPI.DLL
- KERNEL32.DLL

The rest of APIs, IcedID injector imports dynamically resolving NTDLL.DLL as follows.

```
HMODULE IcedId_resolve_ntdll()
{
  HMODULE nt; // eax@1
  HMODULE nt_1; // edi@1
  signed int v2; // esi@2
  int v3; // esi@2
  int v4; // esi@2
  int v5; // esi@2
  int v6; // esi@2
  int v7; // esi@2
  int v8; // esi@2
  int v9; // esi@2
  int v10; // esi@2

  nt = GetModuleHandleA("NTDLL.DLL");          // 9-5-2018: IcedID Injector "NTDLL.DLL" API Dynamic Loader
  nt_1 = nt;
  if ( nt )
  {
    dword_402000 = 1;
    v2 = Load_Dll((int)nt, (unsigned int)nt, 0x51, &ntdll_LdrGetProcedureAddress_0, (int)&addr_of_ntdll_func_0);
    v3 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0x82, &ntdll_LdrLoadDll, (int)&addr_of_ntdll_func_1) | v2;
    v4 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0xEB, &ntdll_ZwAllocateVirtualMemory, (int)&addr_of_ntdll_func_2) | v3;
    v5 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0x66, &ntdll_ZwCreateUserProcess, (int)&addr_of_ntdll_func_3) | v4;
    v6 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0xDF, &ntdll_ZwProtectVirtualMemory, (int)&addr_of_ntdll_func_4) | v5;
    v7 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0x94, &ntdll_ZwWriteVirtualMemory, (int)&addr_of_ntdll_func_5) | v6;
    v8 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0xE4, &ntdll_ZwWaitForSingleObject, (int)&addr_of_ntdll_func_6) | v7;
    v9 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0x7E, &ntdll_RtlDecompressBuffer, (int)&addr_of_ntdll_func_7) | v8;
    v10 = Load_Dll((int)nt_1, (unsigned int)nt_1, 0x26, &ntdll_RtlExitUserProcess, (int)&addr_of_ntdll_func_8) | v9;
    nt = (HMODULE)((v10 | Load_Dll(
                          (int)nt_1,
                          (unsigned int)nt_1,
                          0xEE,
                          &ntdll_ZwFlushInstructionCache,
                          (int)&addr_of_ntdll_func_9)) == 0);
  }
  return nt;
}
```

V. Minimalistic Process Injection: Hooking Engine *ZwCreateUserProcess & RtlExitUserProcess*

Essentially, IcedID injector starts checks if it is being with the "/u" parameter, and if it does, it sleeps for 5000 milliseconds. Otherwise, it resolves NTDLL.dll APIs dynamically and proceeds into the main hooking function hooking ZwCreateUserProcess and RtlExitProcess APIs. Eventually, it launches the main code via "svchost.exe."

The injector main function works as follows as pseudo-coded in C++:

```
//////////////////////////////////////////////////////
/////// IcedID Injector Start Function //////////////////
//////////////////////////////////////////////////////

void __noreturn IcedID_start()
{
  LPSTR v0;
  WCHAR path_svchost_exe;
  struct _STARTUPINFOW StartupInfo;
  WCHAR WINDIR_svchost_exe;
  struct _PROCESS_INFORMATION ProcessInformation;

  v0 = GetCommandLineA();
  if ( StrStrIA(v0, &unk_407DB8) )

  // "/u" - param check via CommandLineA
    Sleep(5000u);
  if ( Get_Param_Resolve_NTDLL((int)v0) )
  {
    get_decoder(&StartupInfo, 0x44);
    get_decoder(&ProcessInformation, 16);
    StartupInfo.cb = 0x44;

    // "IcedID" main hooking function
    if ( HookMain((int)ntdll_ZwCreateUserProcess,

        (int)my_ZwCreateUserProcess) )
    {
      GetSystemDirectoryW(&path_svchost_exe, 0x104u);

      // Set up %WINDIR%\System32 directory path
      SetCurrentDirectoryW(&path_svchost_exe);
      get_svchost((int)&WINDIR_svchost_exe);

      // "svchost.exe"
      lstrcatW(&path_svchost_exe, &WINDIR_svchost_exe);
      CreateProcessW(0, &path_svchost_exe, 0, 0, 0, 0, 0, 0, &StartupInfo,

      &ProcessInformation);
    }
  }
  ExitProcess(0);
}
```

Talos provides an excellent description of this technique as follows (copy/paste):

- *In the memory space of the IcedID process, the function ntdll!ZwCreateUserProcess is hooked.*
- *The function kernel32!CreateProcessA [CreateProcessW (Unicode) version- @VK_Intel) is called to launch svchost.exe and the CREATE_SUSPENDED flag is not set.*

- *The hook on ntdll!ZwCreateUserProcess is hit as a result of calling kernel32!CreateProcessA. The hook is then removed, and the actual function call to ntdll!ZwCreateUserProcess is made.*
- *At this point, the malicious process is still in the hook, the svchost.exe process has been loaded into memory by the operating system, but the main thread of svchost.exe has not yet started.*
- *The call to ntdll!ZwCreateUserProcess returns the process handle for svchost.exe. Using the process handle, the functions ntdll!NtAllocateVirtualMemory and ntdll!ZwWriteVirtualMemory can be used to write malicious code to the svchost.exe memory space.*
- *In the svchost.exe memory space, the call to ntdll!RtlExitUserProcess is hooked to jump to the malicious code already written*
- *The malicious function returns, which continues the code initiated by the call tokernel32!CreateProcessA, and the main thread of svchost.exe will be scheduled to run by the operating system.*
- *The malicious process ends.*

A. IcedID "HookMain"

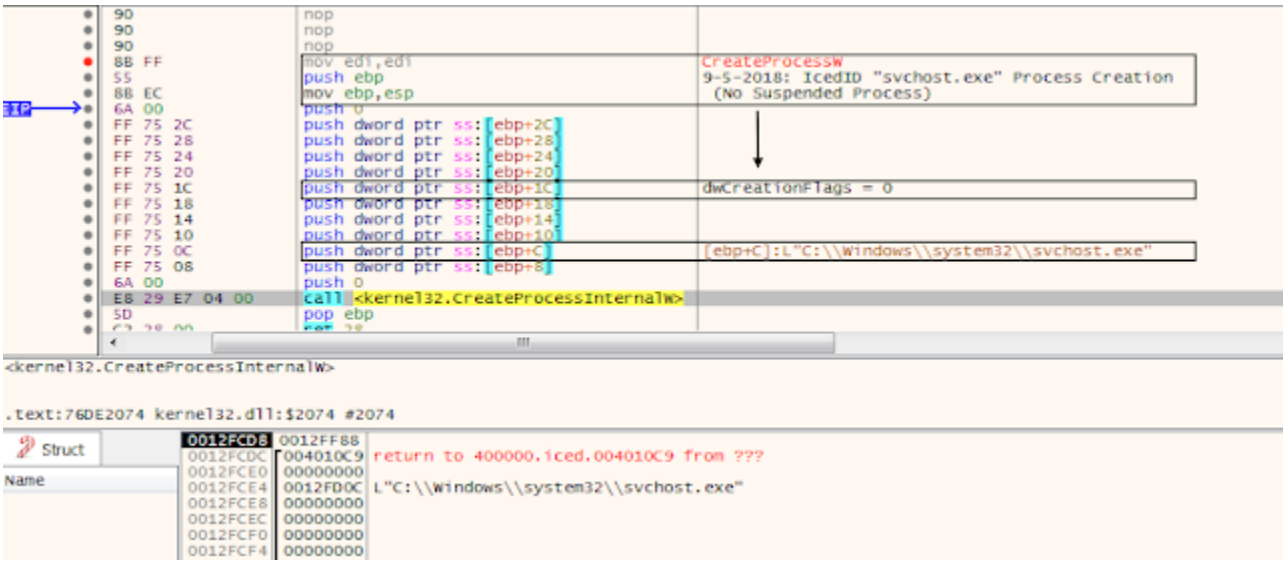
The IcedID malware BOOL-type "HookMain" function works as follows:

```
//////////////////////////////////////////////////////
/////// IcedID Injector HookMain Function ////////////////
//////////////////////////////////////////////////////
BOOL __cdecl HookMain(int relative_offset_opcode_jump, int a2)
{

  result = ntdll_ZwProtectVirtualMemory_0(0xFFFFFFFF, relative_offset_opcode_jump, 5,
64, (int)&v4);
  v3 = result;
  if ( result )
  {
//"0xE9" opcode for a jump with 32-bit relative
    *(_BYTE *)relative_offset_opcode_jump = 0xE9u;
    *(_DWORD *)(relative_offset_opcode_jump + 1) = a2 - relative_offset_opcode_jump -
5;
    ntdll_ZwProtectVirtualMemory_0(0xFFFFFFFF, relative_offset_opcode_jump, 5, v4,
(int)&v4);
    result = v3;
  }
  return result;
}
```

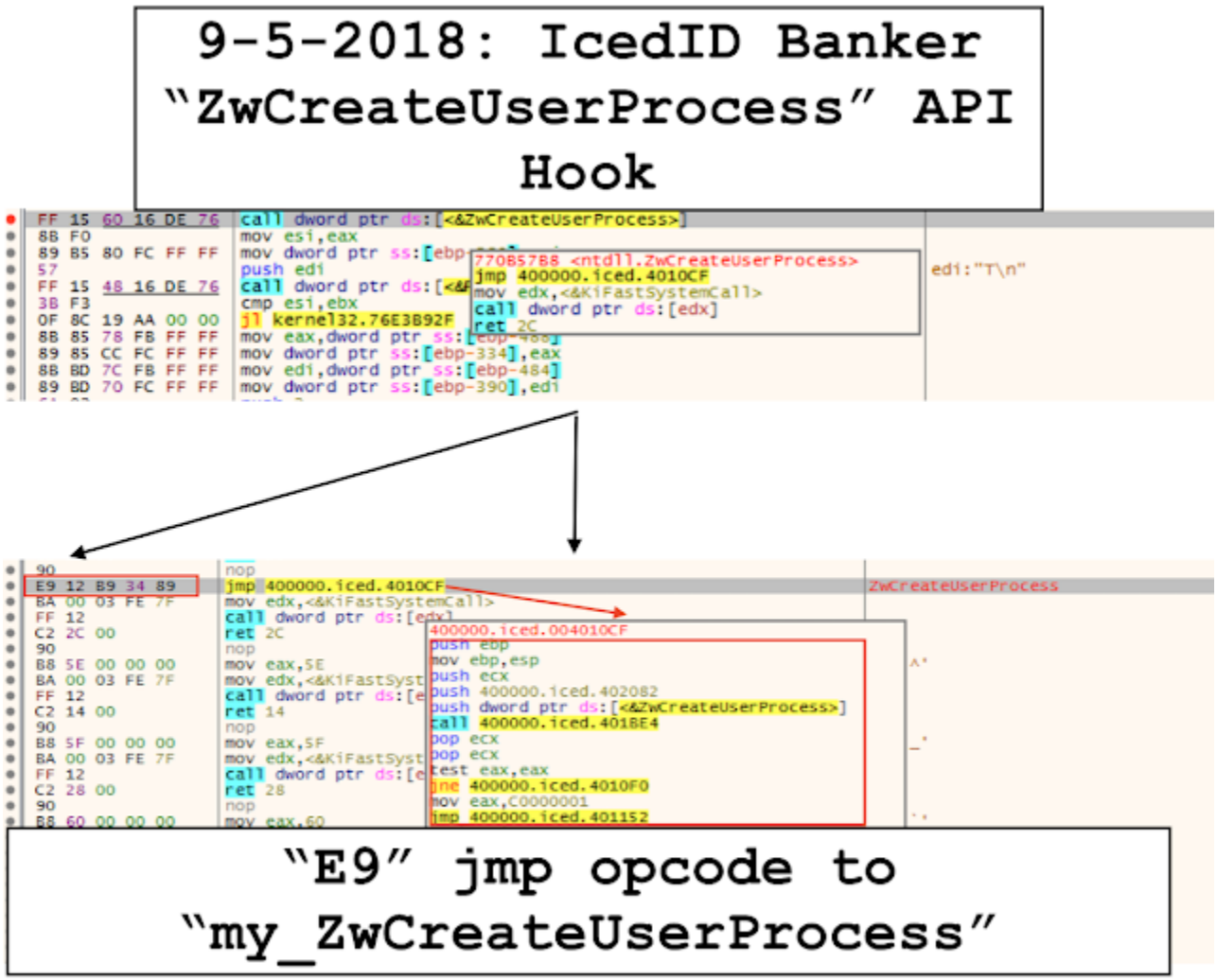
B. Injector CreateProcessW API Execution (dwCreationFlags=0)

IcedID sets up the process execution CreateProcessW with dwCreationFlags set to 0 with no suspended proceses.



Next, the malware sets up the hook for ZwCreateUserProcess (overwrites with relative opcode 0xe9 jump) and then decompressing the buffer via RtlDecompressBuffer API call. Subsequently, the malware sets another hook on RtlExitUserProcess.

C. IcedID "myZwCreateUserProcess" Hook



The IcedID signed int "my_ZwCreateUserProcess" function prototype is as follows:

```
//////////////////////////////////////////////////////
/////// IcedID Injector my_ZwCreateUserProcess Function ///

//////////////////////////////////////////////////////
signed int __thiscall my_ZwCreateUserProcess(void *this, _DWORD *a2, int a3, int a4,
int a5, int a6, int a7, int a8, int a9, int a10, int a11, int a12)
{

  v13 = this;
  if ( ZwProtectVirtualMemory((int)ntdll_ZwCreateUserProcess,
(int)&addr_of_ntdll_func_3) )
  {
    result = ntdll_ZwCreateUserProcess(a2, a3, a4, a5, a6, a7, a8, a9, a10, a11,
a12);
    if ( !result )
    {
      if ( ntdll_RtlDecompressBuffer_0(&a12, &v13) )
        result = HookRtlExitUserProcess(*a2, a12) != 0 ? 0 : 0xC0000001;
      else
        result = 0xC0000001;
    }
  }
  else
  {
    result = 0xC0000001;
  }
  return result;
}
```

Additionally, the malware enters the boolean-type function "HookRtlExitProcess," which deals with writing the malicious code via ntdll!ZwAllocateVirtualMemory and ntdll!ZwWriteVirtualMemory, which returns the call back to CreateProcessW to launch the execution of "svchost.exe" in memory.

D. "HookRtlExitUserProcess" Function

```
//////////////////////////////////////////////////////
//// IcedID Injector HookRtlExitUserProcess Function ////
//////////////////////////////////////////////////////
BOOL __cdecl HookRtlExitUserProcess(int a1, int a2)
{

  v2 = 0;
  v6 = a1;
  lpMem = 0;
  v8 = 0;
  v9 = a2;
  v3 = ntdll_ZwAllocateVirtualMemory_0(a1, 0x54, 4);
  if ( v3 )
  {
    v2 = zwAllocateVirtualMemory_DecoderMain((int)&v6);
    if ( v2 )
    {
      v4 = (char *)lpMem + *(_DWORD *)(v9 + 0x10);
      if ( v4 )
      {
        *(_DWORD *)v4 = v3;
        v2 = ntdll_ZwWriteVirtualMemory_Main((int)&v6);
        if ( v2 )
        {
          v2 = CreateHookRtlExitProcess(a1, ntdll_RtlExitUserProcess, v8 + *(_DWORD
*)(v9 + 0xC));
          if ( v2 )
            v2 = ntdll_ZwWriteVirtualMemory_0(a1, v3, (int)&dword_402000, 0x454);
        }
      }
    }
    if ( lpMem )
      GetProcessHeap_Free(lpMem);
  }
  return v2;
}
```

VI. Yara Signature: IcedID Injector

```
rule crime_win32_iceid_injector {
   meta:
      description = "Detects IcedID Banker Injector"
      author = "@VK_Intel"
      date = "2018-09-07"
      hash1 = "f48efe24259106d0d22bf764c90c96a03155f87710ec181830ea5e84c8d15a9f"
   strings:
      $s1 = "NTDLL.DLL" fullword ascii
      $s2 = "StrStrIA" fullword ascii
      $s3 = "StrToIntA" fullword ascii
      $s4 = "GetSystemDirectoryW" fullword ascii
      $s5 = "GetNativeSystemInfo" fullword ascii
      $s6 = "LoadLibraryA" fullword ascii

      $hook_rtlexitproces = { ff ?? ?? ff ?? e8 ?? ?? ?? ?? f7 d8 59 1b c0 25 ff ff
ff 3f 59 05 01 00 00 c0 5e 8b e5 5d c2 2c 00}
      $hook_main = { 55 8b ec 51 56 8b ?? ?? 8d ?? ?? 57 50 6a 40 6a 05 56 6a ff e8
?? ?? ?? ?? 8b f8 83 c4 14 85 ff 74 ?? 8b ?? ?? 2b c6 c6 ?? ?? 83 e8 05 89 ?? ?? 8d
?? ?? 50 ff ?? ?? 6a 05 56 6a ff e8 ?? ?? ?? ?? 83 c4 14 8b c7 5f 5e 8b e5 5d c3}
      $hook_zwcreate_user = { 55 8b ec 51 68 82 20 40 00 ff ?? ?? ?? ?? ?? e8 ?? ??
?? ?? 59 59 85 c0 75 ?? b8 01 00 00 c0 eb ?? 56 ff ?? ?? 8b ?? ?? ff ?? ?? ff ?? ??
ff ?? ?? ff ?? ?? ff ?? ?? ff ?? ?? ff ?? ?? ff ?? ?? 56 ff ?? ?? ?? ?? ??
85 c0 75 ?? 8d ?? ?? 50 8d ?? ?? 50 e8 ?? ?? ?? ?? 59 59 85 c0 75 ?? b8 01 00 00 c0
eb ?? ff ?? ?? ff ?? e8 ?? ?? ?? ?? f7 d8 59 1b c0 25 ff ff ff 3f 59 05 01 00 00 c0
5e 8b e5 5d c2 2c 00}
      $ntdll_resolver = { 57 68 c0 7e 40 00 ff ?? ?? ?? ?? ?? 8b f8 85 ff 75 ?? 5f c3
53 56 68 76 20 40 00 68 30 20 40 00 68 51 d4 0c e5 33 db c7 ?? ?? ?? ?? ?? ?? ?? ??
?? 53 57 57 e8 ?? ?? ?? ?? 68 70 20 40 00 68 28 20 40 00 68 b2 9f d8 b0 53 57 57 8b
f0 e8 ?? ?? ?? ?? 68 58 20 40 00 68 08 20 40 00 68 eb da 7b d3 53 57 57 0b f0 e8 ??
?? ?? ?? 83 c4 48 0b f0 68 82 20 40 00 68 40 20 40 00 68 66 5f b1 f4 53 57 57 e8 ??
?? ?? ?? 68 64 20 40 00 68 18 20 40 00 68 df 5d 79 8c 53 57 57 0b f0 e8 ?? ?? ?? ??
68 5e 20 40 00 68 10 20 40 00 68 94 9c 50 c5 53 57 57 0b f0 e8 ?? ?? ?? ?? 83 c4 48
0b f0 68 6a 20 40 00 68 20 20 40 00 68 e4 d1 46 ae 53 57 57 e8 ?? ?? ?? ?? 68 88 20
40 00 68 48 20 40 00 68 7e b7 06 fd 53 57 57 0b f0 e8 ?? ?? ?? ?? 68 7c 20 40 00 68
38 20 40 00 68 26 dd 7f 2d 53 57 57 0b f0 e8 ?? ?? ?? ?? 83 c4 48 0b f0 68 8e 20 40
00 68 50 20 40 00 68 ee 1a 0c 53 53 57 57 e8 ?? ?? ?? ?? 83 c4 18 0b c6 f7 d8 1b c0
5e 5b 40 5f c3 83 ?? ?? ?? ?? ?? ?? e8 ?? ?? ?? ?? 85 c0 0f ?? ?? ?? ?? ?? 53 57 e8
?? ?? ?? ?? 8b f8 8b da 8b cf 0b cb 0f ?? ?? ?? ?? ?? 55 53 57 e8 ?? ?? ?? ?? 8b e8
59 59 85 ed 0f ?? ?? ?? ?? ?? 56 68 0e 21 40 00 68 c8 20 40 00 68 51 d4 0c e5 53 57
55 e8 ?? ?? ?? ?? 68 08 21 40 00 68 c0 20 40 00 68 b2 9f d8 b0 53 57 55 8b f0 e8 ??
?? ?? ?? 68 f0 20 40 00 68 a0 20 40 00 68 eb da 7b d3 53 57 55 0b f0 e8 ?? ?? ?? ??
83 c4 48 0b f0 68 1a 21 40 00 68 d8 20 40 00 68 66 5f b1 f4 53 57 55 e8 ?? ?? ?? ??
68 fc 20 40 00 68 b0 20 40 00 68 df 5d 79 8c 53 57 55 0b f0 e8 ?? ?? ?? ?? 68 f6 20
40 00 68 a8 20 40 00 68 94 9c 50 c5 53 57 55 0b f0 e8 ?? ?? ?? ?? 83 c4 48 0b f0 68
02 21 40 00 68 b8 20 40 00 68 e4 d1 46 ae 53 57 55 e8 ?? ?? ?? ?? 68 20 21 40 00 68
e0 20 40 00 68 7e b7 06 fd 53 57 55 0b f0 e8 ?? ?? ?? ?? 68 14 21 40 00 68 d0 20 40
00 68 26 dd 7f 2d 53 57 55 0b f0 e8 ?? ?? ?? ?? 83 c4 48 0b f0 68 26 21 40 00 68 e8
20 40 00 68 ee 1a 0c 53 53 57 55 e8 ?? ?? ?? ?? 55 0b f0 e8 ?? ?? ?? ?? 83 c4 1c 85
f6 5e 75 ?? c7 ?? ?? ?? ?? ?? ?? ?? ?? ?? 5d 5f 5b c3}

   condition:
       uint16(0) == 0x5a4d and
         filesize < 80KB and
         ( all of ($s*) and $hook_main)
       or ( $hook_main and ($ntdll_resolver or $hook_rtlexitproces or
```

```
$hook_zwcreate_user))
}
```