WHITEPAPER

Bitdefender

Security

Loading DLLs for illicit profit. A story about a Metamorfo distribution campaign

www.bitdefender.com

Contents

Authors:

Janos Gergo SZELES – Senior Software Engineer, Bitdefender

Ruben Andrei CONDOR – Security Researcher, Bitdefender

Introduction

Late last year, we noticed a massive ongoing campaign of banker malware concentrated primarily in Brazil. The threat actors behind this campaign are not your run-of–the-mill banker crowd: they have a predilection for **defense evasion**, with their signature modus operandi revolving around a technique named **dynamic-link library (DLL) hijacking** [1].

What initially captured our attention was that one or more potential actors opted to execute malicious actions inside processes launched from digitally signed executable files instead of merely running malicious executable and scripts. These applications correspond to various trusted and popular software packages. The processes run from files stored in uncommon locations (a subfolder with a random name located in the *Public* user's library - *Documents*, *Music*, *Pictures*, *Videos* -, *ProgramData* or *Downloads*) with seemingly random names and unusual extensions such as *.SCR* or *.PIF*. During the malicious rampage, these actors were using living-off-the-land binaries[2] (usually script interpreters such as *wscript.exe* and *powershell.exe*) to further evade defenses.

In some cases, attackers also resorted to code injection techniques to move the execution flow inside legitimate applications running on the system. By identifying similar tools, techniques and procedures, we were able to map a multitude of different client infections to activities of the same malicious threat actor that the security community refers to as ***Metamorfo***[2], a notorious banker group with a history of targeting Brazilian users.

Since our initial discovery, several valuable articles presenting parts of this Metamorfo malware distribution campaign were published [4][5][6]. We do not want to diminish the work of other researchers or to cast blame on software vendors. Instead, in the following lines, we merely want to share some thoughts on Dll Hijacking and to present how Bitdefender technologies and telemetry saw the Metamorfo outbreak and the malware distribution campaign.

DLL Hijacking as a method to defeat defenses

Special conditions such as the POSIX subsystem or Windows Subsystem for Linux aside, a constant for all Windows applications is that they need to interact with DLLs for most of their actions.

Even the simplest program needs to link *ntdll.dll*, the native system library, to merely start. If the program is not a native application (written by Microsoft developers or some AV company), chances are that the process needs to use *kernel32.dll*, *advapi32.dll*, *user32.dll* and a plethora of other libraries built by Microsoft or other independent software vendors. It's common practice for companies to structure their applications using custom implemented DLLs. For instance, Bitdefender has a few dozen different DLLs developed by teams spread across multiple development locations. This reliance on DLLs exposes applications to different classes of vulnerabilities. DLL Search-Order Hijacking, DLL Hijacking, DLL pre-loading (aka. binary planting), DLL side-loading are all "Thorns in the Side of the Anti-Virus Industry"[7].

We won't dive deeper into any of these techniques, as they have been thoroughly studied and documented elsewhere.

A primer to hijacking DLLs

A digital signature, as implemented by Authenticode is considered a token of trust. Among other factors, an Authenticode-signed executable file looks less alarming to users when requesting elevated privileges. Subsequently, if the User Account Control (UAC) prompts users that their trusted anti-virus vendor wants to make changes to the system, they likely won't question it.

Organizations sometimes (mis)configure their intrusion detection system to allow digitally signed applications to run undisturbed, ignoring their malicious behavior. Some antimalware solutions likely won't detect the EXE since it's presumed to originate from a trustworthy source.

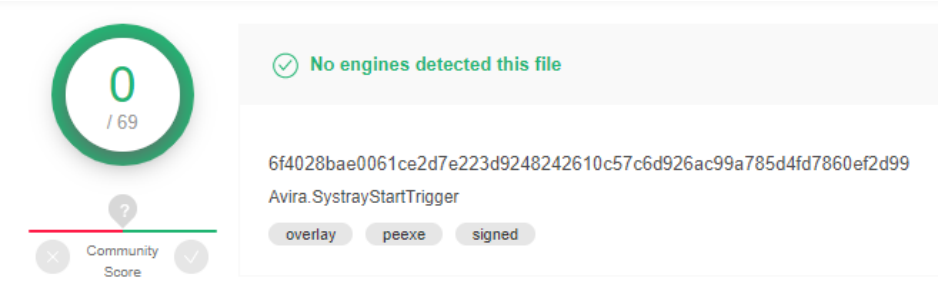
Last, what even is a DLL to the average PC user? Will they ever look up DLLs on VirusTotal?

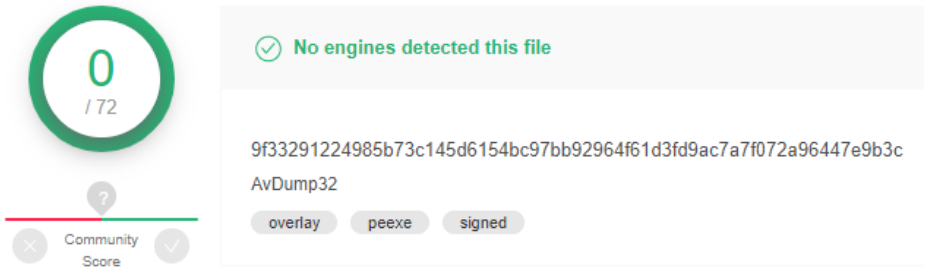
During the time we monitored the Metamorfo campaign, we've seen 5 different software components, manufactured by respected software vendors, abused in the attack. They come from *Avira*, *AVG* and *Avast*, *Damon Tools*, *Steam* and *NVIDIA*. Are any of these software companies at fault? Well, they only have the misfortune of developing popular software, as they become more prominent targets than less-known applications.

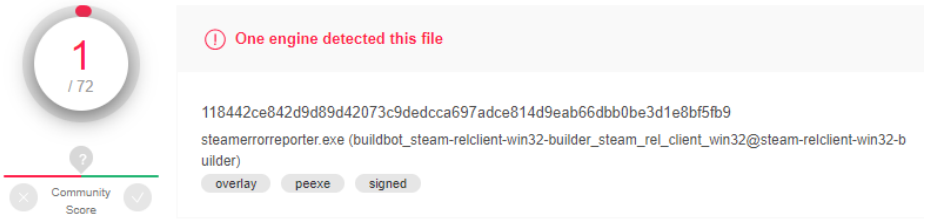
We mention the components and their distribution in the campaign in the following table.

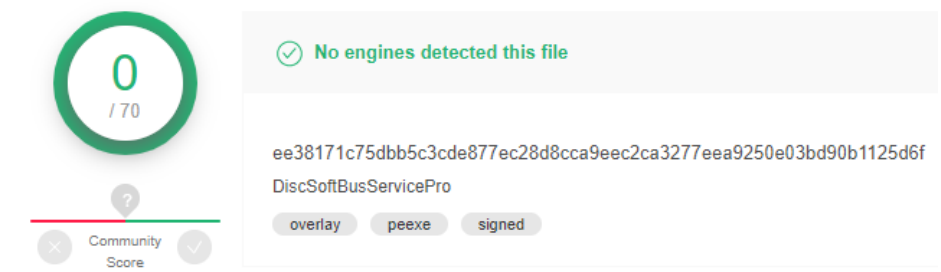
Component InternalName	Count of affected users
Avira.SystrayStartTrigger	3457
AvDump32	512
DiscSoftBusServicePro	734
SteamErrorReporter	299
NvSmartMaxApp	193

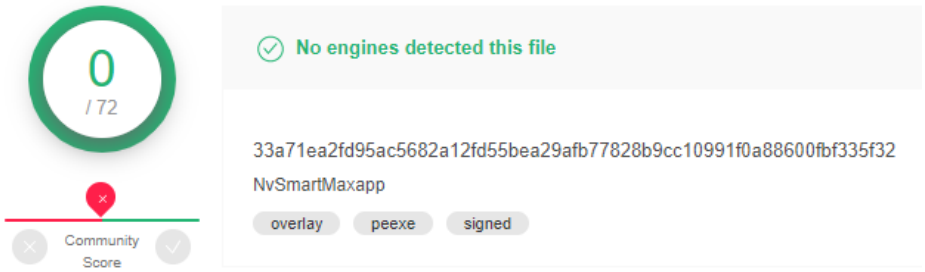
Let's quickly look these files up on VirusTotal to double check whether cyber-security companies consider them malicious.











As you can see, all these components are clean (and some security solutions don't like Steam reporting errors). Users are not to blame for trusting that processes created by these files are not malicious.

Brief technical analysis of a Metamorfo infection

Initial access

The malware arrives on the system in a *MSI* installer that seems legitimate to the user. However, the attackers rigged the file to include some legitimate files along with the target executable and the malicious DLL. Upon execution, the program installs the legitimate component and the malware in a subfolder, either under a Public user's library folder (Documents, Music, Pictures, Videos or Downloads) or under ProgramData. After the installation is complete, it executes the legitimate binary that automatically loads the malicious DLL.

Execution flow

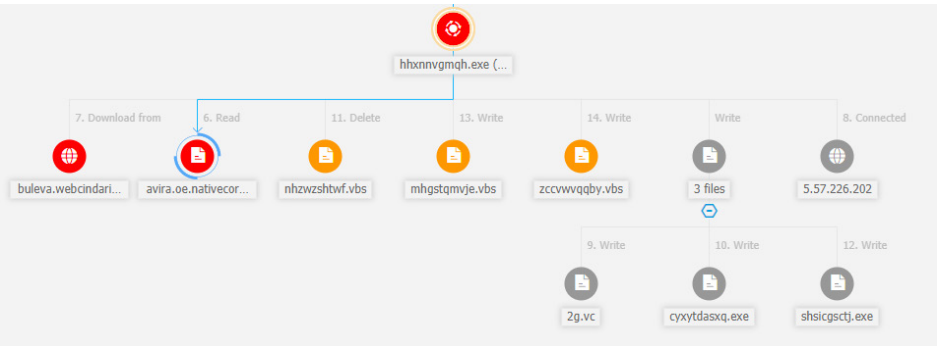


Fig.1 Execution of Avira.SystrayStartTrigger with a random name.exe, as captured by Bitdefender GravityZone

When execution starts, the target process loads its imported DLLs, including the malicious one. Besides executing the code from the DLL's startup routine, the legitimate application runs as it would in a clean scenario. The malware DLL has all of its exported functions pointing to the same address in the code. Thus, a legitimate call from the clean application triggers the execution of the malicious code.

Avira.SystrayStartTrigger

The most widespread legitimate component used in this campaign is Avira.SystrayStartTrigger.

Version Info:

Company Name:	**Avira Operations GmbH & Co. KG**
Product Name:	Avira
Internal Name:	Avira.SystrayStartTrigger
File Description:	Avira

Digital Signature:

Issuer Name:	**Symantec Class 3 Extended Validation Code Signing CA - G2**
Subject:	Avira Operations GmbH & Co. KG
Serial Number:	2482b97529c45783c6c2e0b95654eb1f

Our analysis of this component reveals that it is normally used to check for the existence of another Avira process named *Avira.Systray.exe*. If that process is not already running, it writes to a Named Pipe *\Avira.*

ExternalCommunicationTaskPipe to signal the *Avira.ServiceHost.exe* process to launch *Avira.Systray.exe*.

```
Client file opened
File name: \Avira.ExternalCommunicationTaskPipe
File ID:   0xFFFFB50B60EB5350
Process:   \Device\HarddiskVolume3\Users\Public\Documents\hhiwsugtuh\hhiwsugtuh.exe
PID:       5224
Server file opened
File name: \Avira.ExternalCommunicationTaskPipe
File ID:   0xFFFFB50B6052E140
Process:   \Device\HarddiskVolume3\Program Files (x86)\Avira\Launcher\Avira.ServiceHost.exe
PID:       2192
File ID 0xFFFFB50B60EB5350:
0000  7B 22 70 22 3A 7B 22 50 61 74 68 22 3A 22 43 3A   {"p":{"Path":"C:
0010  5C 5C 55 73 65 72 73 5C 5C 50 75 62 6C 69 63 5C   \\Users\\Public\
0020  5C 44 6F 63 75 6D 65 6E 74 73 5C 5C 68 68 69 77   \Documents\\hhiw
0030  73 75 67 74 75 68 5C 5C 41 76 69 72 61 2E 53 79   sugtuh\\Avira.Sy
0040  73 74 72 61 79 2E 65 78 65 22 2C 22 43 6F 6D 6D   stray.exe","Comm
0050  61 6E 64 4C 69 6E 65 50 61 72 61 6D 65 74 65 72   andLineParameter
0060  73 22 3A 22 2F 63 6F 6E 6E 65 63 74 54 6F 48 6F   s":"/connectToHo
0070  73 74 20 22 2C 22 53 65 73 73 69 6F 6E 49 64 22   st ","SessionId"
0080  3A 31 7D 2C 22 63 22 3A 22 53 59 53 54 52 41 59   :1},"c":"SYSTRAY
0090  2E 53 54 41 52 54 22 2C 22 69 64 22 3A 22 32 33   .START","id":"23
00A0  37 31 22 7D                                       71"}
```

Fig. 2 Avira.ExternalCommunicationTaskPipe Signal

If *Avira.ServiceHost.exe* fails to respond, it is responsible for launching *Avira.Systray.exe*. Based on the previous observation, we can state that this component can launch any executable named *Avira.Systray.exe* in the current working directory. As of yet, we have found no indication that attackers abuse the application in this manner.

Fig. 3 Launching a powershell executable renamed to Avira.Systray.exe

Avira.OE.NativeCore.dll

The theory that we're dealing with a DLL hijacking is confirmed by looking at the modules used by this process. *Avira.OE.NativeCore.dll* stands out. The DLL dropped next to the legitimate Avira component immediately raises suspicion, as the former is more than 20 MB. The version info of the file corresponds to the legitimate DLL developed by Avira. However, the first significant difference is that the larger one was compiled in Delphi instead of Visual C/C++. Malware authors frequently use Delphi, as this allows for rapid prototyping of a GUI for their executable, so it has lots of standard libraries, easily embeddable in a single executable file. The advantage of having all malicious code in a DLL is that the malicious actions appear to be performed by the legitimate process. Thus, the attacker might evade behavioral detection if binaries with a trusted digital signature are whitelisted.

The Version Info of the malicious DLL mimics that of the clean one.

Company Name:	**Avira Operations GmbH & Co. KG**
Product Name:	Avira
Internal Name:	Avira.OE.NativeCore
File Description:	Avira.OE.NativeCore

The DLL has a verification mechanism so that it does not execute unless the computer is configured with Brazilian locales. After checking the requirements of the Brazilian environment, the malware creates a mutex with a seemingly random name (*holexrel*), to ensure that it does not run the same malicious code every time the clean process calls one of the exported functions. It then creates a copy of the original, legitimate executable file, in the same folder where it is running from but under another random name with an .EXE, .SCR or .PIF extension. Next, it writes a VBS script into *AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup* with a random name to secure persistence. If there is already a script file in that location, the malware deletes it.

The script is responsible for launching the legitimate executable, executing the malicious DLL next to it every time the computer starts. The script instantiates the COM object **WScript.Shell**, with CLSID **{72c24dd5-d70a-438b-8a42-98424b88afb8}**, as well as **FileSystemObject** with CLSID **{0d43fe01-f093-11cf-8940-00a0c9054228}**. The method of referring COM objects instead of interacting directly with the file system and processes through built-in functions helps evade classic detection mechanisms.

The contents of the dropped VBS are similar to the one below, and only the variable names are randomized. The script checks if the side-loaded DLL is present in the same folder as the executable, then it starts the EXE.

Startup Script

```
on error resume next
Set ZIMJAS = CreateObject("WScript.Shell")
set XWRWSY = createobject("scripting.filesystemobject")
COJNZE = "cmd /k c: & cd\ & cd C:\Users\Ion Testalescu\Documents\x8rehgrg5\ & msdgghwi-
ux.exe"
if XWRWSY.fileexists ("C:\Users\Ion Testalescu\Documents\x8rehgrg5\Avira.OE.NativeCore.
dll") then
ZIMJAS.run(COJNZE),0
End If
```

The malware searches for some banking-related software on the system and it waits for the user to access banking pages. It monitors them by looking at window titles and comparing them with the following set of strings:

Decrypted Strings

Applications:
Aplicativo sicoob; sicoob; AplicativoBradesco.exe; Aplicativo bradesco; Banco Bradesco; NavegadorExclusivoBradesco.exe; core.exe; C0R3; RapportService.exe; R4pp0rt

Banking Pages
[bb.com.br]; Banco do Brasil; Banco Bradesco | Pessoa Física, Exclusive, Prime e Private; Bradesco; Pessoa jurídica | Bradesco; Bradesco JuJu; Banco Itaú; Banco Itáu; Santander; Banco Santander; sicredi; Banco Sicredi; Mercantil; internetbanking; Caixa Economica; Banco Sicoob; Unicred Portal; Unicred; Internet Banking BNB; Banco BNB; Banco Inter; Banco Intermedium; Banco MUFG Brasil S.A.; Banestes - Internet Banking; Banestes; Internet Banking; Bancos Geral; Banco do Estado do Pará S/A; Cetelem | Login; Cetelem; Cooperativa de Crédito; Nova Home | Internet; Banco Safra; SafraNet; BANCO PAULISTA; UniprimeCentral; Uniprime; Bem vindo ao seu BMG; BMG; Portal - Banco Votorantim; Votorantim; Pine Online; NBC BANK; Tribanco Online; Tribanco; Banco Alfa; Banco Indusval & Partners; Banco Indusval; Portal Internet Banrisul; Banrisul; Banco Original; Acesse sua conta Celcoin; Celcoin; Login - Nubank; Nubank; BRB Banknet; Banco de Brasília; Banco da Amazônia; Banese; BancoTopazioInternetBanking; Banco Topazio; BancoIndustrial; Banco Industrial; Daycoval; bradesco; CIDETRAN; Viacredi; PagueVeloz Serviços de Pagamentos; Pague Veloz; Banco Safra - Internet Banking Pessoa Jurídica; Pessoa Jurídica | Internet; Safra Jurídica; Pessoa Física | Internet; Safra Física; Bradesco Exclusive; Bradesco Prime; Prime; Banco Bradesco S/A; Bradesco Private Bank; Bradesco Private

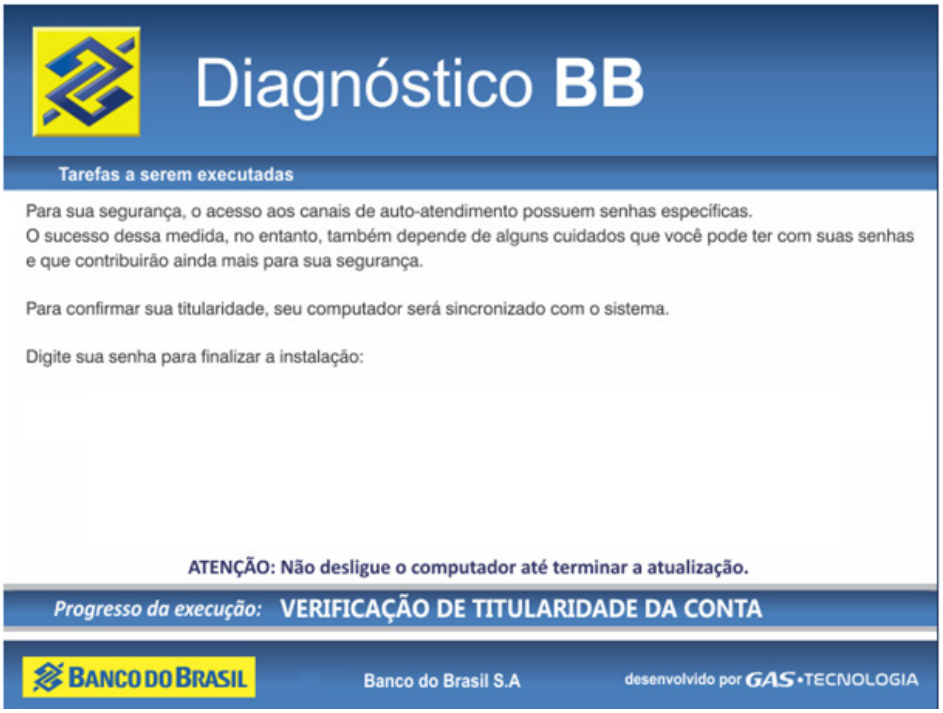
It then attempts to connect to *buleva[.]webcindario[.]com*, which translates to the IP 5.57.226.202. This IP has been used for various name resolutions before, as shown in **Appendix 2**. These domain names mislead users as they seem legitimate, except for the *webcindario[.]com* part (e.g. bankofamerica[.]webcindario[.]com, disney[.]webcindario[.]com).

The malware is also capable of downloading files from the C&C server in the same folder under random names. Finally, the executable keeps running in the background, waiting for commands from the server. It is capable of logging keypresses on the system when the user fills in a password on one of the banking pages. It can also disable the autocomplete feature of browsers so the user has to retype the password manually.

The malware also has the capability of capturing screenshots with the help of the Windows Magnifying API, imported from *Magnification.dll*. We have captured some of the commands along with application names searched, file names and web resources by decrypting the malware's resources, as shown in **Appendix 3**.

The resource directory of the DLL contains some images that mimic security notifications from various Brazilian banks, tricking the user into believing that something happened with their bank account. This panic can then be used to the attacker's advantage to steal sensitive information.

Example images:



The authors of the DLL also repurposed several public libraries available on the internet. One of them is DCPCrypt [9], a library responsible for encrypting buffers with various algorithms. Each of these algorithm classes have string identifiers starting with DCP (e.g. 'DCP_blockcipher128'). The libraries facilitate the encrypted communication with the C2 server via https.

Other Legitimate Components Abused

There are some more executables susceptible to DLL hijacking, and we observed that attackers abuse them the same way they did with those from Avira. Naturally, they are delivered with a different DLL name, with Version Info specific to each one, but with the same functionality.

AvDump32 from AVG

Version Info:

Company Name:	**AVG Technologies CZ, s.r.o.**
Product Name:	AVG Internet Security System
Internal Name:	AvDump32
File Description:	AVG Dump Process

Digital Signature:

Issuer Name:	**Symantec Class 3 SHA256 Code Signing CA**
Subject:	AVG Technologies CZ, s.r.o.
Serial Number:	34f34f4b5727a1636c768808abf89e96

The variant abusing this component has slightly different behavior at the start of its execution. It first checks whether it runs in the context of *AvDump32* and, if so, it injects itself into *wmplayer.exe* or *dllhost.exe*.
If loaded into *wmplayer.exe/dllhost.exe*, it executes the rest of its code [3]. With this, the malware introduces another layer of defense evasion as another well-known process performs malicious actions, and an AV might not detect them.
Another difference we observed is the persistence mechanism of this variant. Instead of a script written to the startup folder, the malware schedules a task to execute *AvDump32* at every restart.

DiscSoftBusServicePro from Daemon Tools

Version Info:

Company Name:	**Disc Soft Ltd**
Product Name:	DAEMON Tools Pro
Internal Name:	DiscSoftBusServicePro
File Description:	Disc Soft Bus Service Pro

Digital Signature:

Issuer Name:	**COMODO RSA Code Signing CA**
Subject:	Disc Soft Ltd
Serial Number:	00f6e3d0098bf4e24d22bbb9550c55343e

SteamErrorReporter from Steam

Version Info:

Company Name:	**Valve Corporation**
Product Name:	Steam
Internal Name:	`steamerrorreporter.exe (buildbot_steam-relclient-win32-builder_steam_rel_client_win32@steam-relclient-win32-builder)`
File Description:	`steamerrorreporter.exe`

Digital Signature:

Issuer Name:	**DigiCert SHA2 Assured ID Code Signing CA**
Subject:	Valve
Serial Number:	054f466ceccbe9d6bee81f5435e64d47

NvSmartMaxapp from NVIDIA

Version Info:

Company Name:	
Product Name:	NVIDIA Smart Maximise Helper Host version 100.03
Internal Name:	NvSmartMaxapp
File Description:	NVIDIA Smart Maximise Helper Host

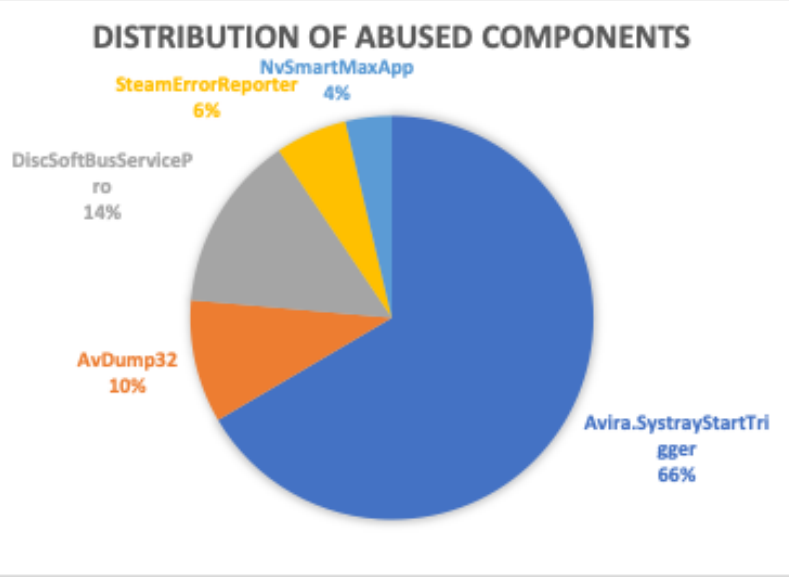
Digital Signature:

Issuer Name:	**Symantec Class 3 SHA256 Code Signing CA**
Subject:	NVIDIA Corporation
Serial Number:	7bc15af21367d0758beddcca118642de

Impact

Metamorfo is a potent piece of malware, whose primary capability is theft of banking information and other personal data from the user and exfiltration of it to the C2 server. We also noticed that the malware tries to download other files from the C2 server, suggesting that it could download an updated version of itself with an extended command set as well.

Campaign distribution and evolution



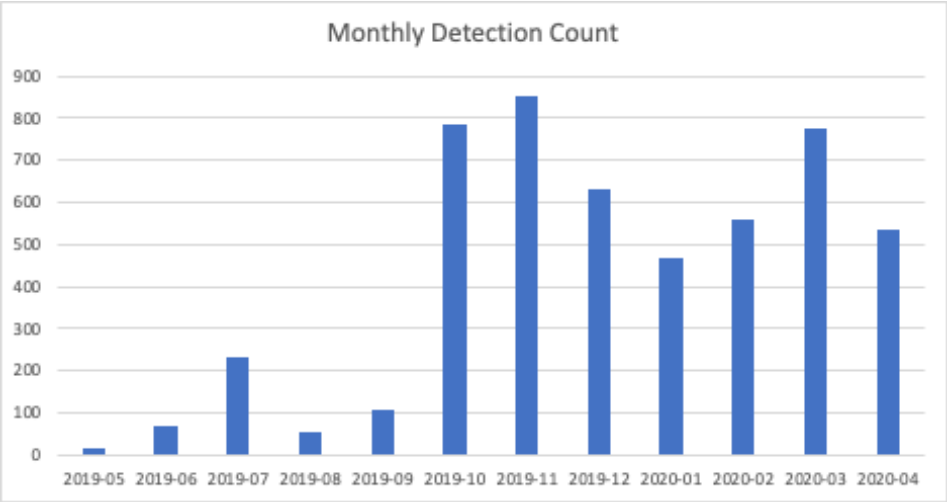
Region Distribution

The campaign mostly targets Brazilian users, but we also traced infections back to Switzerland and Argentina as well.

Region	Count
Brazil	5188
Switzerland	7
Argentina	1

Evolution Over Time

Month	Detection Count
2019-05	14
2019-06	71
2019-07	230
2019-08	56
2019-09	105
2019-10	784
2019-11	853
2019-12	629
2020-01	469
2020-02	557
2020-03	775
2020-04	536



Hardening software to increase the cost of an attack

DLL hijacking attacks have been around for as long as the Windows operating system has been. At this point, many developers are thinking along the lines of "*But this is a Windows issue! Can I, as a software developer, fix it?*"

Yes, you can. You surely can, but it's not easy.

The essence of the problem is the way Windows searches for DLLs that are required to load a program[8]. When loading DLLs by name (as opposed to the full path) the default DLL search path is:

- The system directory (returned by the GetSystemDirectory function).
- The 16-bit system directory.
- The Windows directory (returned by the GetWindowsDirectory function).
- The current directory for the process.
- The directories listed in the PATH environment variable.

There are two exceptions to the rule, KnownDLLs (DLLs listed in the registry under *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\KnownDLLs* and their dependencies) and DLLs already loaded in the process.

The solution should be simple, namely to never load DLLs by name. However, this is not enough. Windows loads DLLs when a developer explicitly asks for a DLL through *LoadLibrary** function calls and implicitly for compile-time linked DLLs. Compile-time DLLs are resolved by DLL name alone before execution even begins, so they are beyond the programmer's control. Fortunately, all operating systems starting with Windows 7 Service Pack 1 and above offer mitigations for the DLL hijacking problem.

On modern Windows systems, the correct way to explicitly load a DLL is by using *LoadLibraryEx* and using a full path or passing the *LOAD_LIBRARY_SEARCH_SYSTEM32* flag. This is, sadly, not enough because DLLs may implicitly or explicitly load other DLLs. To solve the problem completely, a programmer must call the *SetDefaultDllDirectories* function at the start of the executable and pass in the *LOAD_LIBRARY_SEARCH_SYSTEM32* flag.

Fixing the problems caused by compile-time DLLs is a bit more challenging. As stated in the introduction, a Windows application needs DLLs to function, and you can't ask programmers to load all DLLs by hand using *LoadLibraryEx* with full paths all the time.

For DLLs that are part of the software packages bundled with the application, developers can create manifests within Visual Studio as a basic check to validate the loaded libraries. An entry in the manifest would look like:

```
<file name="dllname.dll" hash="DA39A3EE5E6B4B0D3255BFEF95601890AFD80709" hashalg="SHA1">
```

The elegant solution is to make sure your application has regular compile-time dependencies only to DLLs listed in *KnownDLLs* (since they are not vulnerable to hijack). Also change all other library dependencies for delay loading at link-time (using */DELAYLOAD:<dllname>.dll* if you compile and link using the Microsoft tool-set). For delayed loads, the loader defers loading a DLL until the first function inside it is called. This gives programmers a chance to call *SetDefaultDllDirectories* before dangerous implicit loading.

Ok, I did all this, is the problem solved now?

As our friends at AVG said in an e-mail when discussing the problem, *sadly you don't have a time machine*. Once you delivered digitally signed and vulnerable versions of a component, you can never take it back. The internet holds all things, and you can never truly get rid of something after it reaches the aether.

In a perfect world, all software vendors would immediately revoke all certificates of vulnerable applications, Microsoft would force Windows to not start applications with revoked certificates and cyber-security solutions would be able to detect and block the tainted binaries.

The entire process would probably cost millions of dollars in development time, re-signing of all binaries and deployment of all updates. We don't live in that world. Instead, all defenders should probably stop trusting digital signatures. The main executable, as well as any of the loaded DLLs, needs to be scanned by security solutions. Users and security specialists alike should configure exceptions only in the most extreme cases and, even then, they should be as explicit as possible. Exceptions should specify the full path of the executable, the certificate and, if possible, even the DLLs loaded and the actions an application is allowed to perform.

Conclusion

Metamorfo is a known piece of Banker malware that resurfaced at the end of last year and almost exclusively focuses on Brazilian victims. Novelty aside, the exciting aspect of this malware is the way it employs various techniques to evade malware defenses and the tremendous work put into finding executables from various well-known software vendors that are susceptible to DLL side-loading.

But why would an actor put so much effort into finding these legitimate vulnerable executables when attackers could launch *rundll32.exe* or *regsvr32.exe* to load any DLL? The answer is that, this way, attackers increase their chances of evading user-suspicious detection. Anti-malware solutions generally monitor actions performed by Windows binaries like *rundll32.exe* or *regsvr32.exe* as they are the go-to tools for mainstream attackers. Also, executing their malware DLL with the help of these processes would raise a red flag for incident responders and threat hunters. Therefore, by choosing prominent software vendors to mask their actions, attackers can remain undetected. A digital signature and familiar name in Version Info might seem benign to casual users as well. The usage of COM objects for starting up the executables and loading the DLLs also helps actions go under the radar if the AV does not monitor them.
Some variants have a second layer of defense evasion consisting of DLL injection into legitimate processes like *wmplayer.exe* or *dllhost.exe*, making the injection from a trusted AV component seem plausible, then masking malicious actions behind a seemingly benign Windows process.

Metamorfo bypasses detection of AV vendors that whitelist applications based on trusted digital signatures. Bitdefender products employ advanced algorithms to monitor process behavior, and therefore are unaffected by whitelists.

We hope software developers reading our article understand the problems caused by digitally signed components vulnerable to DLL Hijacks and attempt to mitigate them by loading DLLs in a safe manner.

Bibliography

[1] https://attack.mitre.org/techniques/T1073/

[2] https://lolbas-project.github.io/

[3] https://blog.trendmicro.com/trendlabs-security-intelligence/analysis-abuse-of-custom-actions-in-windows-installer-msi-to-run-malicious-javascript-vbscript-and-powershell-scripts/

[4] https://securityintelligence.com/posts/taking-over-the-overlay-reverse-engineering-a-brazilian-remote-access-trojan-rat/

[5] https://medium.com/@chenerlich/the-avast-abuser-metamorfo-banking-malware-hides-by-abusing-avast-executable-ac9b8b392767

[6] https://www.cybereason.com/blog/brazilian-financial-malware-banking-europe-south-america

[7] https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-dll-sideloading.pdf

[8] https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order

[9] https://bitbucket.org/wpostma/dcpcrypt2010/src/default/

MITRE Techniques

Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Collection	Command and Control	Exfiltration
Execution through Module Load	DLL Search Order Hijacking	DLL Search Order Hijacking	DLL Search Order Hijacking	Input Capture	Application Window Discovery	Input Capture	Commonly Used Port	Exfiltration Over Command and Control Channel
Component Object Model and Distributed COM	Registry Run Keys / Startup Folder		DLL Side-Loading		File and Directory Discovery	Screen Capture		
			Masquerading		Software Discovery			
			Process Injection					

Appendix 1. Indicators of Compromise

Hashes

A list of hashes is available in this Github project maintained by Bitdefender.

URL

hxxp://buleva[.]webcindario[.]com/my/

IP Address

5.57.226.202

Files dropped

The names of the dropped .vbs files are random, for example:

\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\shzvmmraec.vbs

Mutex

Creates a named mutex, “holexrel”

Appendix 2. The malicious IP's previous DNS resolutions

A list of domains is available in this Github project maintained by Bitdefender.

Appendix 3. Decrypted Resources

A list of decrypted strings is available in this Github project maintained by Bitdefender.

An up-to-date list of indicators of compromise is available to Bitdefender Advanced Threat Intelligence users. More information about the program is available at https://www.bitdefender.com/oem/advanced-threat-intelligence.html.

Why Bitdefender

Proudly Serving Our Customers

Bitdefender provides solutions and services for small business and medium enterprises, service providers and technology integrators. We take pride in the trust that enterprises such as **Mentor, Honeywell, Yamaha, Speedway, Esurance or Safe Systems** place in us.

Leader in Forrester's inaugural Wave™ for Cloud Workload Security

NSS Labs "Recommended" Rating in the NSS Labs AEP Group Test

SC Media Industry Innovator Award for Hypervisor Introspection, 2nd Year in a Row

Gartner® Representative Vendor of Cloud-Workload Protection Platforms

Dedicated To Our +20.000 Worldwide Partners

A channel-exclusive vendor, Bitdefender is proud to share success with tens of thousands of resellers and distributors worldwide.

CRN 5-Star Partner, 4th Year in a Row. Recognized on CRN's Security 100 List. CRN Cloud Partner, 2nd year in a Row

More MSP-integrated solutions than any other security vendor

3 Bitdefender Partner Programs - to enable all our partners – resellers, service providers and hybrid partners – to focus on selling Bitdefender solutions that match their own specializations

Trusted Security Authority

Bitdefender is a proud technology alliance partner to major virtualization vendors, directly contributing to the development of secure ecosystems with **VMware, Nutanix, Citrix, Linux Foundation, Microsoft, AWS, and Pivotal.**

Through its leading forensics team, Bitdefender is also actively engaged in countering international cybercrime together with major law enforcement agencies such as FBI and Europol, in initiatives such as NoMoreRansom and TechAccord, as well as the takedown of black markets such as Hansa. Starting in 2019, Bitdefender is also a proudly appointed CVE Numbering Authority in MITRE Partnership.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

CRN AV-TEST AV Gartner 451 Research FORRESTER IDC GLOBAL

TECHNOLOGY ALLIANCES

Microsoft NUTANIX aws Pivotal Cloud Foundry CITRIX

Bitdefender

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-PR-WhitepaperMetamorfo-creat4500en-EN
05.07.20May 22 2020 11 11 am05.22.20