

Grandoreiro goes north: From Brazil to Mexico with a new DLL sideloading campaign

A acronis.com/en/tru/posts/grandoreiro-goes-north-from-brazil-to-mexico-with-a-new-dll-sideloading-campaign

SantiagoPontioli



Summary

Introduction

Grandoreiro is a notorious banking trojan that has been operating across Latin America since at least 2016, targeting financial institutions and their customers through phishing campaigns and social engineering techniques. Written in Delphi and Brazilian in origin, it is one of the many of Latin American banking trojans (alongside Guildma, Javali and Melcoz). Over the years, the malware has gone through multiple iterations and infrastructure changes to evade detection and sustain its operations. In January 2024, a coordinated operation led by Brazil's Polícia Federal and coordinated through INTERPOL — with Spanish authorities and private-sector partners — disrupted significant portions of its infrastructure and reduced its overall reach.

Despite these actions, Grandoreiro has not been fully eradicated. Activity associated with the malware can still be observed, although at a considerably lower volume compared to its peak operations. In May 2026, we observed a renewed campaign in which Grandoreiro operators abused the legitimate application Duplicate Files Finder in combination with DLL sideloading techniques, continuing their established pattern of leveraging trusted software to stealthily execute malicious payloads.

Technical details

In the following sections, we analyze Grandoreiro's initial downloader, a lightweight component in terms of functionality; however, it is heavily protected and incorporates multiple

defensive mechanisms. These techniques are designed to detect sandboxed or virtualized environments and hinder analysis efforts, significantly increasing the complexity of reverse engineering.

Initial sample

The analyzed archive was delivered as a ZIP file named Fac-BH22DC0608_RevMQKSAC.zip (MD5: 82f771c3ec4fe979c3ae00372e8c3ac8). While the initial delivery vector could not be conclusively confirmed, both historical Grandoreiro distribution patterns and the invoice-like filename (with "Fac" likely referring to factura, meaning "invoice") have led us to assess with moderate confidence that the malware was delivered via a spam campaign.

Inside the archive is a hidden folder containing decoy PDF and XML files designed to evade heuristic detection.

! [~]			<DIR>	2026-04-30 23:44	--h-
dupfdll	dll	1,009,152	2008-06-01 07:52	-a--	
mingwm10	dll	7,091,712	2024-12-25 11:05	-a--	
_2IH4ENJIRRNCo	exe	2,718,208	2008-06-01 07:23	-a--	

Figure 1: Content of the ZIP archive

c:\Users\Steven\Downloads\~\zaqfa\rkku\cprh\edfqr*.*			
Name	Ext	Size	
[..]	<DIR>		2
cyziv1	pdf	33,586	2
gmxyycz6	pdf	25,221	2
naeygb5	pdf	21,113	2
onsr2	pdf	27,497	2
sche4	pdf	26,232	2
svfkp10	pdf	24,786	2
teavnz8	pdf	22,341	2
tymb7	pdf	25,712	2
xrie3	pdf	25,218	2
zmathi9	pdf	26,518	2

Figure 2: Dummy PDF files

DLL sideloading

DLL sideloading is a technique where an attacker places a malicious DLL beside a legitimate executable that automatically loads it through the normal Windows DLL search order. When the trusted application starts, the malicious DLL is loaded into the process, allowing attacker code to execute while appearing as part of a legitimate and often signed application.

In this campaign, the authors use the legitimate Duplicate Files Finder application but rename it to a randomly generated filename to obscure its purpose. When executed, it first loads another legitimate dependency, dupfdll.dll. This DLL then attempts to load its next dependency, mingwm10.dll, which is not legitimate. In this case, it is a malicious replacement version.

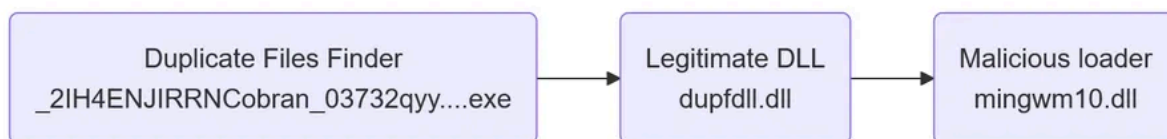


Figure 3: DLL sideloading chain

Filename / MD5 / Compilation time:

- _2IH4ENJIRNCobran_03732qyykae214BKXXN6180-Dt.exe —
cfbd8d062e9baa98737a0260996f48c6 — 2008/06/01
- dupfdll.dll — e882ee4a643b00871c98bcfe4b4d7cd1 — 2008/06/01
- mingwm10.dll — f1aed8cf2adafa86927fc58d5f24073e — 2026/05/01

Analysis of the PE timestamps reveals inconsistencies that are indicative of potential DLL sideloading activity. In particular, the legitimate parent application shows a compilation timestamp dating back to 2008, whereas the injected trojan implant carries a significantly more recent timestamp, appearing to originate from 2026. PE compilation timestamps are trivially forgeable and are frequently faked by malware authors, so the 2026 value should not be treated as authoritative on its own; nonetheless, combined with the sideload chain and the dependency mismatch, we assess with high confidence that a benign binary is being abused as a loader for the malicious payload.

Because Duplicate Files Finder is a GUI-based application that normally displays a user interface window, the malicious implant begins its execution by hiding the application window to avoid drawing user attention.

Sandbox detection and anti-analysis

After suppressing the GUI, it proceeds to execute a series of internal routines designed to determine whether the environment is sandboxed or being actively analyzed. These checks are intended to ensure execution only in genuine user environments while evading automated analysis systems.

```

push    ebp
mov     ebp, esp
push    ecx
push    ebx
push    esi
push    edi
xor     eax, eax
mov     [ebp+var_4], eax
call    hide_window
call    mutex
call    gettickcount_check
test    al, al
jz      short loc_9DEC83
call    software_check
test    al, al
jnz     short loc_9DEC83
call    anti_vm
test    al, al
jnz     short loc_9DEC83
xor     eax, eax

```

Figure 4: Main thread of the loader

The [malware](#) checks the system uptime of the current machine. Its threshold is set to 180,000 milliseconds, meaning the system must have been running for at least three minutes. This check can help bypass sandboxes that begin analysis in a fresh boot state.

```

009DEAEC 83 C4 F8          add     esp, 0FFFFFFF8h
009DEAEF E8 A0 C3 A3 FF    call   GetTickCount64
009DEAF4 89 04 24          mov     [esp+8+var_8], eax
009DEAF7 89 54 24 04       mov     [esp+8+var_4], edx
009DEAFB 83 7C 24 04 00    cmp     [esp+8+var_4], 0
009DEB00 75 07            jnz     short loc_9DEB09
009DEB02 81 3C 24 20 BF    cmp     [esp+8+var_8], 180000

```

Figure 5: System uptime verification

If the infected system passes the tick count verification check, the malware proceeds to inspect the software installed on the host. Rather than querying the operating system directly, it determines whether an application is present by checking for the existence of its corresponding desktop shortcut. The malware specifically checks for the following applications:

- Google Chrome
- CCleaner
- Firefox
- FileZilla Client
- Acrobat Reader DC
- Microsoft Edge
- Skype

Interestingly, the presence of all of these applications is treated as an indicator of an analysis environment rather than a legitimate user system. If shortcuts for all seven applications are present on the desktop, the malware classifies the system as a sandbox environment and terminates execution. Based on this behavior, we've assessed with moderate confidence that the operators are attempting to identify and blacklist a specific sandbox configuration that ships with this particular combination of software.

Following the installed software validation, the malware performs a series of additional system checks to further profile the infected host and determine whether it is running in a legitimate user environment. Specifically, it gathers information about:

- The amount of installed physical memory using `GlobalMemoryStatusEx`.
- The number of available processors.
- The amount of free disk space on the `C:\` drive.
- The screen resolution via `GetSystemMetrics`.
- Recent user activity by inspecting recently opened documents.

These checks are commonly used as anti-analysis measures, as sandbox and virtualized environments often exhibit characteristics that differ from those of real user systems, such as limited hardware resources, unusual display settings or a lack of evidence of normal user activity.

In addition to the previously described checks, the malware actively searches for artifacts associated with virtualized environments. It does so by looking for the presence of VMware and VirtualBox drivers and components, including:

- `VBoxMouse.sys`, `VBoxGuest.sys`, `VBoxSF.sys`, `VBoxVideo.sys`

- vmmouse.sys, vmhgfs.sys, vm3dgl.dll, vmci.sys

The malware also queries the `HARDWARE\DESCRIPTION\System\BIOS` registry path and examines the values of `SystemManufacturer`, `SystemProductName`, and `BIOSVendor`. These values are compared against a blacklist of virtualization-related keywords, including: `VBOX`, `VMWARE`, `VIRTUAL`, `QEMU`, `XEN` and `INNOTEK`.

Speaking of registry checks, the malware also queries the registry key `\SOFTWARE\VMware, Inc.\VMware Tools`, which indicates the presence of VMware Tools installed in a guest environment.

The next layer of Grandoreiro's anti-analysis mechanisms involves inspecting the list of currently running processes. The malware specifically checks for the presence of tools commonly used in debugging, reverse engineering, and network traffic analysis (49 process names in total — debuggers, disassemblers, network sniffers, and EDR / monitoring tools). It targets the following process names:

- regmon.exe, procmon.exe, filemon.exe, Wireshark.exe, ProcessHacker.exe
- ollydbg.exe, x64dbg.exe, x32dbg.exe, x96dbg.exe, cheatengine.exe, ollyice.exe
- fiddler.exe, devenv.exe, radare2.exe, ghidra.exe, frida.exe, binaryninja.exe, cutter.exe, hopper.exe, jd-gui.exe
- canvas.exe, pebrowsepro.exe, scylla.exe, cffexplorer.exe, angr.exe, hexworkshop.exe, sysexp.exe
- Capsa.exe, tcpdump.exe, nw.exe, snort.exe, pcap.exe, nmap.exe, nessusd.exe, prtg.exe, cain.exe
- NetworkAnalyzerPro.exe, netmon.exe, packetanalyzer.exe, packettotal.exe, tshark.exe, windump.exe, NetFlowAnalyzer.exe, SolarWinds.DataProcessor.exe
- apimonitor-x64.exe, apimonitor-x32.exe, fakenet.exe, procexp64.exe, procexp.exe

Next, the malware contacts `ip-api.com` to determine the victim's public IP address and geolocation. It then blacklists several countries, including `CZ`, `RU` and `NL`.

We have seen country blacklisting before, often for "patriotic" reasons in pay-per-install campaigns — for example, Russian-speaking malware avoiding infections in Russia or nearby regions. However, in this case, the selection appears less ideological and more operational. We assess with low-to-moderate confidence that traffic from these countries disproportionately includes security researchers, antivirus companies or automated sandbox environments.

```

lea     eax, [ebp+var_4]
call    ip_api
lea     edx, [ebp+var_C]
mov     eax, 5Eh ; '^'
call    get_string
mov     edx, [ebp+var_C]
lea     ecx, [ebp+var_8]
xor     eax, eax
call    decrypt_string ; CZ
mov     edx, [ebp+var_8]
mov     eax, [ebp+var_4]
call    sub_40C2A8
jz      short loc_9A5A36
lea     edx, [ebp+var_14]
mov     eax, 5Fh ; '_'
call    get_string
mov     edx, [ebp+var_14]
lea     ecx, [ebp+var_10]
xor     eax, eax
call    decrypt_string ; RU
mov     edx, [ebp+var_10]
mov     eax, [ebp+var_4]
call    sub_40C2A8
jz      short loc_9A5A36
lea     edx, [ebp+var_1C]
mov     eax, 62h ; 'b'
call    get_string
mov     edx, [ebp+var_1C]
lea     ecx, [ebp+var_18]
xor     eax, eax
call    decrypt_string ; NL
mov     edx, [ebp+var_18]
mov     eax, [ebp+var_4]
call    sub_40C2A8

```

Figure 6: Country check

After the geolocation check, the malware then validates both the current username and the machine's computer name. In some cases, the values appear to be designed as matching pairs, for example, a username like "Peter" paired with a hostname such as "PETER-PC". On the other hand, certain entries appear to trigger independently, for instance, a hostname such as "difusor" is sufficient to flag the system regardless of the logged-in user. The list below summarizes the blacklisted usernames and computer names:

- nagini / DESKTOP-XXXXXXX
- Bruno / -
- dx / LAPTOP-V865QAO5

- elias / WIN-LTLFLAII9W
- elx / COMPUTER-VMPWPE
- John / JOHN-PC
- Peter / PETER-PC
- - WDAGUtilityAccount / - (note: this is the default Windows Sandbox account)
- WORK / WORK
- - / DESKTOP2457
- - / difusor
- Win-StephyPC3 / -
- WIN-VUA6POUV5UP / -

The malware also verifies the existence of specific directories. In particular, it searches for the presence of the following paths:

- A:\TOOLS\ProcessInvestigator\ , C:\TOOLS\ProcessInvestigator\ , D:\TOOLS\ProcessInvestigator\
- F:\TOOLS\ProcessInvestigator\ , G:\TOOLS\ProcessInvestigator\ , H:\TOOLS\ProcessInvestigator\
- D:\programming , D:\script

If the infected computer is flagged as a sandbox, the loader shows an error message in Spanish that says "There was an error viewing your document, restart your computer and try again."

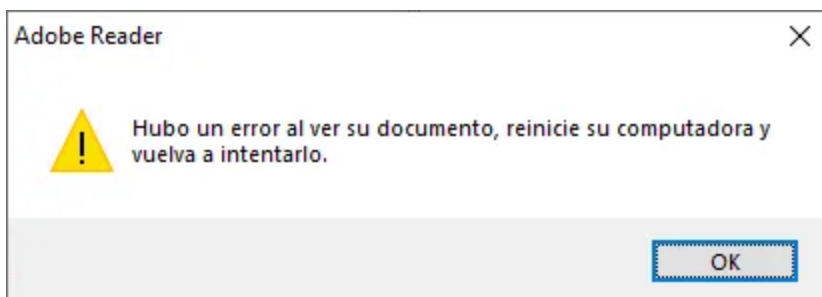


Figure 7: Simulated system error

C2 communication

Otherwise, if the system passes all validation checks, the malware proceeds to resolve the IP address of its hardcoded C2 server. For this purpose, it leverages Google's DNS-over-HTTPS service by sending a request to `hxxps://dns[.]google/resolve?name=` followed by the hardcoded domain. In this sample, the domain used is `b744156103040828396040[.]nhlfan[.]net`.

Unfortunately, the server was offline at the time of our analysis, preventing direct interaction with the C2 infrastructure. Nevertheless, static analysis revealed that the malware issues an HTTP GET request over TCP port 6432 to retrieve the second-stage payload. The request itself is encrypted and contains host-specific information, including identifiers associated with the infected user and system. Decoded, the request takes the following form (this is the decrypted representation, not the on-the-wire ciphertext):

```
CLIENT_SOLICITA_DDS_MDL|SOLICITADO*~+HU*~+Budapest*~+Budapest*~+DESKTOP-41NVP89*~+Steven*~+10*~+Windows Defender*~+0*~+0*~+0*~+1*~+42F69286*~+18-05-2026*~+00:55:57*~+*~+
```

The request begins with the command `CLIENT_SOLICITA_DDS_MDL`, which likely indicates a request for the second-stage payload. During static analysis, additional strings associated with C2 communication were identified. These are easily distinguishable as they are consistently written in uppercase:

- `CLIENT_SOLICITA_DDS_MDL`, `SOLICITADO`, `CLIENT_ENVIA_STATUS`, `PUBLIC`, `REPETIDO`, `RECUPERADO`
- `INFECTADO`, `AV_COMEU_MODULO`, `ERRO_EXECUCAO`, `ERRO_FALHA_DOWNLOAD`, `ERRO_EXTRACAO`

It is notable that, although the malware appears to have been developed by Portuguese-speaking authors, no telemetry observed targets in Brazil or Portugal. This is consistent with public reporting that the post-takedown Grandoreiro codebase forked, with a "light" variant now focused on Mexican targets — which aligns with the Mexico-heavy telemetry described below.

String obfuscation

A significant challenge during the analysis of the loader is the presence of encrypted strings. Grandoreiro implements a custom string obfuscation scheme that combines a proprietary decryption routine with Base64 encoding to encode runtime strings and complicate static analysis.

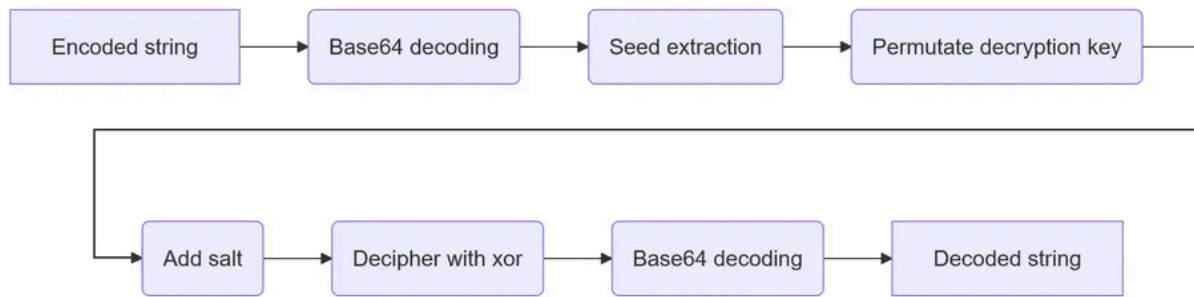


Figure 8: String decryption workflow

No publicly available ready-to-use scripts were identified for decoding strings in this specific variant. As a result, we implemented a custom Python-based decoder to reproduce the deobfuscation routine.

```

def unxor(s, key):
    s = bytes.fromhex(s.decode())
    res = b""

    for i in range(len(s) - 1):
        k = key[i % len(key)]
        key = b"\x00" * 16 + key[16:]
        ch2 = s[i + 1] ^ k
        ch1 = s[i]
        if ch2 <= ch1:
            ch2 += 255
        ch2 -= ch1
        res += bytes([ch2])

    return res

def addpadding(s):
    padding = len(s) % 4
    if padding:
        s += b"=" * (4 - padding)
    return s

def decrypt(s):
    salt = b"50600tvc0415"
    hardcoded_key = b"!@#$%^&*()_+=[ ]{|};:.,.<>?"
    seed, encoded_string = split(base64.b64decode(addpadding(s.encode())))
    key = permute(hardcoded_key, seed)[:0x14]
    decoded = unxor(encoded_string, key + salt)
    return base64.b64decode(addpadding(decoded)).decode()

```

Figure 9: Python implementation of the string deobfuscation routine

Several implementation details of the decryption algorithm are worth highlighting:

- The routine uses Base64 encoding without padding. As a result, when the input length is not divisible by four, the appropriate padding characters (=) must be appended before decoding.
- The algorithm permutes the decryption key and uses it to XOR-decrypt the encoded strings. During analysis, we observed that at a certain stage of the process, the first 16 bytes of the derived key are overwritten with null bytes. This behavior appears unintended and may indicate a bug in the implementation; however, it does not affect the decryption process, as the corresponding encryption routine applies the same transformation.

- The decryption key and salt are hardcoded within the executable.

```
push    dword ptr fs:[eax]
mov     fs:[eax], esp
mov     eax, [ebp+var_C]
call    sub_40B180
lea     eax, [ebp+var_4]
mov     edx, offset asc_99702C ; "!@#$$%^&*()_+-=[]{|};:,.<>?"
call    sub_40B5B8
mov     eax, [ebp+var_4]
test    eax, eax
jz      short loc_996F61
mov     eax, [eax-4]
```

Figure 10: Hardcoded decryption key

Telemetry

We have checked telemetry for the last 30 days in our database and observed a steady stream of activity associated with this family, indicating that despite its reduced scale, the campaign remains active and continues to evolve.

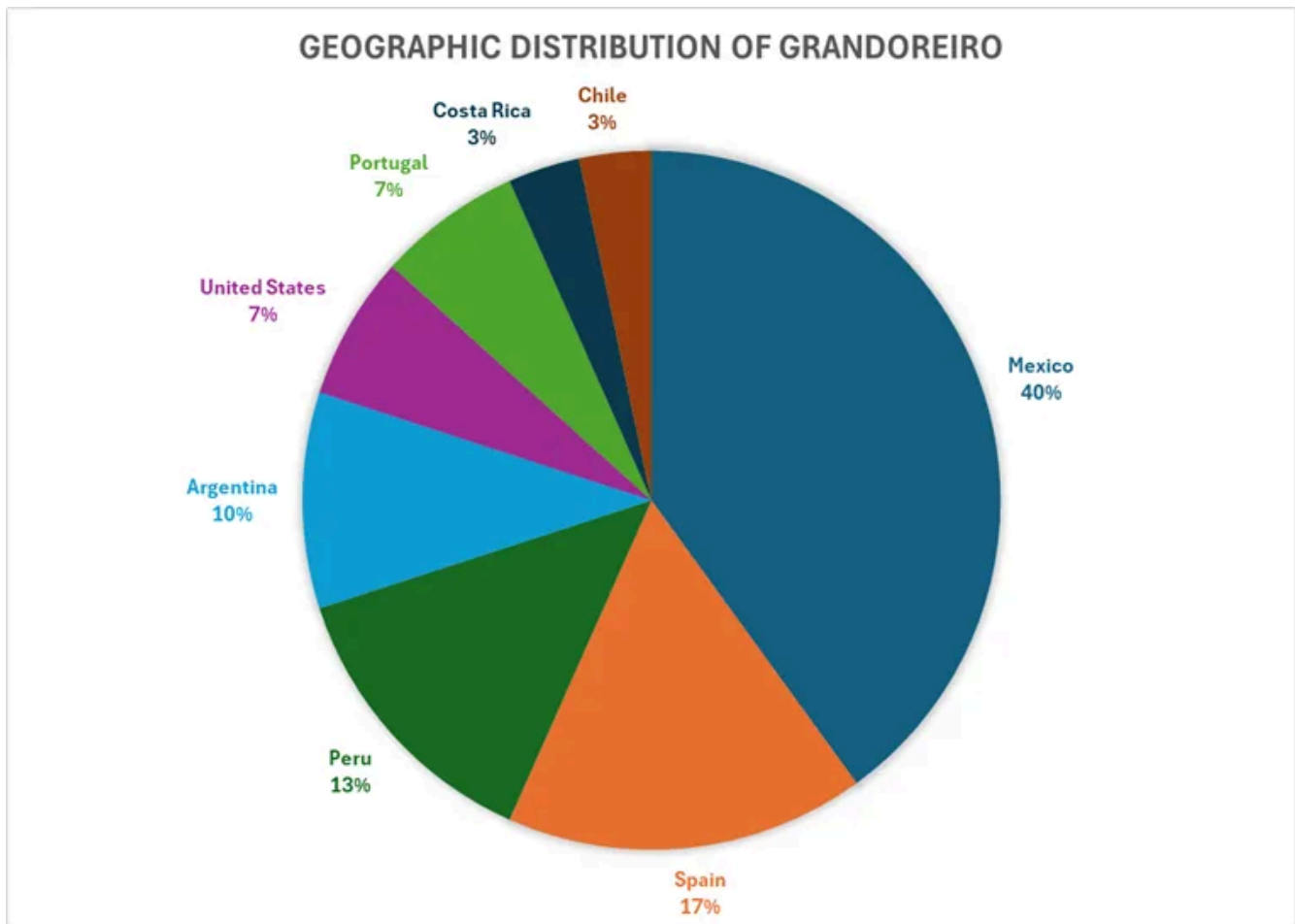


Figure 11: Geographic distribution of detections

This distribution highlights a clear concentration of activity in Mexico, which accounts for the largest share of observed samples. Spain and several Latin American countries follow, forming a secondary cluster of detections. Overall, the data suggests that Grandoreiro activity remains primarily focused on Latin American regions, with limited but notable presence in Europe (mostly Spain) and North America.

MITRE ATT&CK mapping

The observed behavior maps to the following ATT&CK techniques:

- T1574.002 — Hijack Execution Flow: DLL Side-Loading (DFF -> dupfdll.dll -> mingwm10.dll)
- T1497.001 — Virtualization/Sandbox Evasion: System Checks (uptime, installed-app shortcuts, VM drivers, BIOS keywords)
- T1082 — System Information Discovery; T1016 — System Network Configuration Discovery; T1614 — System Location Discovery (ip-api.com / country gating)
- T1057 — Process Discovery (49-tool analyst/EDR blacklist)

- T1140 — Deobfuscate/Decode Files or Information (custom Base64 + XOR string routine)
- T1036.005 — Masquerading: Match Legitimate Name or Location (renamed DFF binary)
- T1071.001 — Application Layer Protocol: Web Protocols (C2 over TCP/6432); T1071.004 — DNS, only if the DoH observation is substantiated
- T1564.003 — Hidden Window (GUI suppression on launch)

Detection and hunting

Defenders can hunt for the following high-signal behaviors observed in this campaign:

- A renamed copy of the Duplicate Files Finder executable loading mingwm10.dll from the same directory (sideload chain).
- Processes resolving a domain via `hxxps://dns[.]google/resolve?name=` immediately followed by an outbound TCP/6432 connection (if the DoH behavior is confirmed).
- The literal token `CLIENT_SOLICITA_DDS_MDL` in egress traffic or memory.
- A GUI process that suppresses its window and then queries `ip-api.com` and the `HARDWARE\DESCRIPTION\System\BIOS` registry path in quick succession.

Conclusion

Overall, this campaign highlights the continued evolution of Grandoreiro and its operators' focus on stealth and evasion. By abusing the legitimate Duplicate Files Finder application for DLL sideloading, the attackers effectively blend malicious execution into trusted software behavior, complicating detection efforts. The multi-stage architecture — starting from a heavily protected loader and progressing to a fully featured payload — demonstrates a deliberate effort to separate initial access from long-term operational capabilities. While overall activity associated with Grandoreiro has decreased compared to its peak, the campaign shows that it remains active and continues to adapt its tooling and infrastructure.

Detection by Acronis

This threat is detected and blocked by Acronis EDR / XDR.

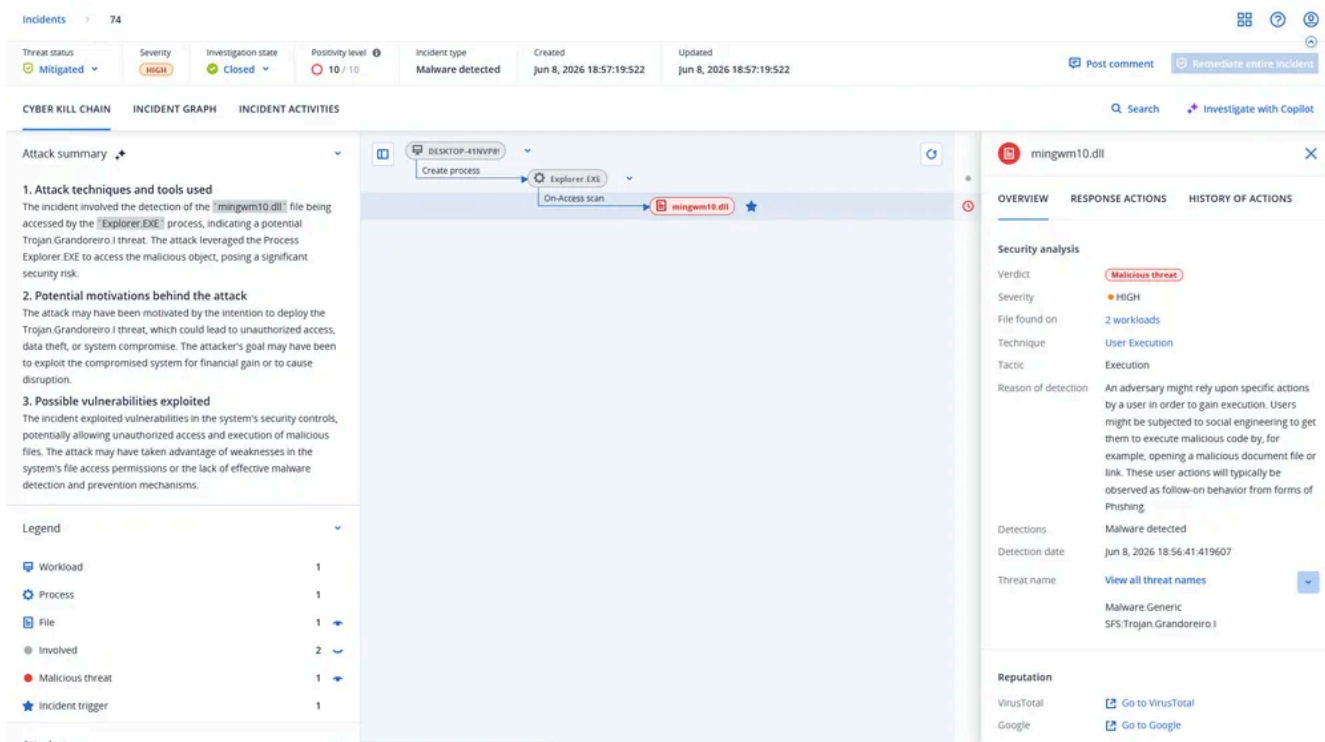


Figure 12: Acronis detects and blocks Grandoreiro

Indicators of Compromise (IoCs)

Samples (SHA256):

- 1b2fe30c5bf57f9623efb34688580fe5bbb2c55351c5a07a6c4313bb6faa29f1
- 1fe5a72aefc38afeeee72d8a939f9db50800a447b7313f4d8c504771bb7fa2de
- 2820a2e36f1cb537a7853fde2313a5158d1e3696ff81c3066ec8fe274358c22d
- 368246eb503585f26e0151431909792b8d9be0edc229bb207be882bfb374c005
- 37492ecd9deb8591ea7e179ebb8e13c6b486cdacae18a6a482d9dd18f08453a3
- 47d5a73b220813299f753ef0a96582a8d08b853391864128a1f15271dbb42e65
- 609755a8c73e53f332428786c366323b3979850aaa0e075d946e6c05aa62f867
- 65db035f79db85ad66fb3e0365e478f95f8f648545b18360d850a7ed179c9dab
- 684f5eb157ef2ba3ea17335d6c8c9c801f93e6eb3aa08ecbfaf5806a4b5e3b80
- a91c7cb932301454df4b0feed58082cae7f2d0e4078d7e6df3f53d806ed04f1d
- ac35843c81381107d4f816681477d706c43fa75f064f2c3d99237c7282f0b798
- ad5762fa98da2aff24d9d6b55be5dae21d07c54679ead67252a19c2521162a87
- c019cfa9b50a69ff07e98bd78b3a6fc489110e5db1c0d79ada716605a2320951
- ca33f5608b72ecca64a002074a4103c7cb83de902ecf5c69949eeeb7eef03b5a
- cc238813ab277cbb4324d37875c37985ca5baeaf8c412b4c85aa8ec5faec7096
- e0491eddb45425a674e479b2590517ffef2f108add431761bb89791fc208b6e9
- e99416ff71e4574de4fceeabdc34f3b9a6e0610f36b77b735b231bd39edb7a0a1

C2 domains (defanged):

- 445675885304004[.]pointto[.]us
- b744156103040828396040[.]nhlfan[.]net
- beeges[.]health-carereform[.]com
- smartpdfhub-7q2m[.]read-books[.]org
- streamlinepdf-8m2x[.]workisboring[.]com
- voyage[.]mydissent[.]net

© 2003–2026 Acronis International GmbH.