

VectraRAT: An Undocumented Full-Stack MaaS Built From Scratch

 socradar.io/blog/vectrarat-undocumented-stack-maas

ameer

September 14, 2026

Author: SOCRadar Threat Research Unit (STRU)

Table Of Content

Free Dark Web Report

Is your Domain/Email Exposed?

Related Articles

Sep 14, 2026

29 Mins Read

Summarize with:

SOCRadar's Threat Research Unit (**STRU**) has documented VectraRAT, a Malware-as-a-Service platform built entirely from scratch rather than forked from leaked RAT code. Renting from \$250 a month, it gives operators hidden-desktop control, keylogging, clipboard hijacking, browser credential theft, and a UAC bypass that elevates with no prompt. The operator "Vectra" is a rebrand of "Nyxel", **active** since at least **August 2022** with **no prior public reporting**. STRU traced live infrastructure from an exposed open directory across more than ten servers and campaigns using Amadey and ClickFix, where **48%** of recovered **victim** entries were **corporate Windows editions**, including exfiltration from Windows Server 2025.

Most remote access tools sold on [crimeware forums](#) are borrowed goods. A leaked AsyncRAT build, a cracked XWorm license, a QuasarRAT fork with a new icon and a new name. The code changes hands, the panel stays recognizable, and defenders keep detecting the same family under a dozen different brands.

VectraRAT is a different case. Every layer of it, the Linux control server, the Windows implant, the protocol between them, the licensing that keeps operators paying, was written by the same developer. None of it had been publicly documented.

STRU found the first live infrastructure on June 23, 2026, through an open directory that should never have been reachable. What followed was a pivot across more than ten servers, dozens of samples, panel logs belonging to real operators, and a Telegram conversation with the developer themselves.

[Download the full report \(PDF\)](#) for the complete technical analysis, including the decompiled protocol internals, panel artifacts, and the full indicator set.

Key Takeaways

- VectraRAT is a previously undocumented, full-stack **Malware-as-a-Service** platform built from scratch. It is not a reskin of AsyncRAT, XWorm, or QuasarRAT.
- It pairs a **Go control server called VectraHub**, with a Vue3 operator panel compiled into the binary, with a native C++ Windows implant. The two speak a proprietary binary TCP protocol using MessagePack over port 3308.
- The operator “**Vectra**” is a rebrand of an older identity, “**Nyxel**”, with a **YouTube channel** dating back to **August 2022**. Nearly four years of activity with no public reporting.
- Capabilities span both the **RAT and the stealer** space: hidden desktop (HVNC), keylogging, SOCKS5 relay, clipboard hijacking with regex replacement, remote shell, and mass browser credential theft, plus a UAC bypass that elevates without a prompt.
- Rented from **\$250 per month** and delivered through the Amadey loader and ClickFix pages, with **48%** of observed **victim** entries on corporate Windows editions, including active exfiltration from Windows Server 2025 hosts.

What Is VectraRAT?

VectraRAT is a rental-only Malware-as-a-Service platform that gives a paying operator full [remote control](#) of a Windows host, along with automated credential and file collection on first connection. The developer keeps the source code, sells access, and handles support over Telegram.

Attribute	Detail
Name	VectraRAT
Previous name	Nyxel
Actor aliases	Vectra on HackForums and Exploit.in, VectraNode on YouTube
First infrastructure observed	June 23, 2026
Earliest actor footprint	August 2022 (YouTube channel creation)
Classification	Hybrid RAT and stealer
Business model	Rental only, from \$250 per month

Attribute	Detail
Control server	VectraHub, written in Go, Linux ELF with an embedded Vue3 panel
Client	Native Windows C++ PE using Win32 and GDI+
Protocol	Proprietary binary TCP framing, 5-byte header, MessagePack payloads
Ports observed	TCP 3308 (C2), 8080 and 8888 (operator panel), 4457, 6667
Delivery observed	Amadey loader, ClickFix pages, tax-themed filenames
Victims observed	38 genuine sessions in under one week, 48% corporate Windows editions

How We Found It

The entry point was an HTTP-accessible open directory on 86.109.75.168, a GorillaServers node in AS53850. It held the whole operation in one folder: multiple variants of **Vectra.exe**, the Linux server binary **VectraHub**, license files, and operational panel logs.

Directory listing for /

- [.data/](#)
- [license.json](#)
- [license_public.pem](#)
- [logs/](#)
- [nohup.out](#)
- [TurboTax.exe](#)
- [TurboTax2025.exe](#)
- [TurboTax2026.exe](#)
- [Vectra.exe](#)
- [Vectra Customer Installation and Usage Guide.txt](#)
- [VectraHub](#)

Open directory on the entry node exposing the VectraRAT ecosystem: the VectraHub ELF, stub binaries, and license files. From there the investigation branched in three directions at once. The binaries pointed at other C2 addresses. The license files revealed the RSA scheme protecting the product. The panel logs contained victims.

Pivoting on the pattern (the same directory layout, the same port set, the same stub metadata) surfaced more than ten servers following the original template, along with dozens of samples that appeared to belong to separate operations. The same node had also served Etsy phishing redirects, and one address in the neighboring range was running a ClickFix panel.

The Business: What \$250 a Month Buys

The developer does not sell source code and does not distribute cracked builds. What an operator rents is access to VectraHub, the Windows client, the payload builder inside the panel, and continuous support through Telegram.

PRICING

Monthly \$250
3 months \$650
6 months \$1,100
Annual \$2,000

Tiered rental pricing advertised for VectraRAT, starting at \$250 per month.

The sales thread sits on HackForums, with a parallel listing on [Exploit.in](#) and negotiation handled over a premium Telegram account. Trial licenses are offered, but only to HackForums members with more than a year of account history. It is a neat filter: researchers spinning up fresh accounts are excluded, and the operators who do get trials arrive with reputations that make their reviews worth something on the forum.

SOCradar®

Threat Hunting

I'm looking for...

Dashboards

Attack Surface Management

Digital Risk Protection

Cyber Threat Intelligence

Threat Hunting

Vulnerability Intelligence

Identity & Access Intelligence

Tactical Intelligence

[https://hackforums.net/showthread.php?tid=6327192](#)
● Deep Diver Finding - 2026-08-13 08:55

ransomwareglobalexplorerremotedesktophvnc---

Vectra \n[n00b@HF:]\\n\\n\\nPosts:\\n1\\nThreads:\\n1\\nCredibility:\\n000\\nPopularity:\\n0\\nBytes:\\nβ 2,015\\nGame XP:\\n0\\n*, \\\"content\\\": '\\'-—————\\nVECTRA(old NYXEL)\\nStable RAT · In-house C2 · Active...

[https://hackforums.net/showthread.php?tid=6327192](#)
● Deep Diver Finding - 2026-06-17 11:55

ransomwareglobaloneremotedesktopcmd---

Vectra \n[n00b@HF:]\\n\\n\\nPosts:\\n1\\nThreads:\\n1\\nCredibility:\\n000\\nPopularity:\\n0\\nBytes:\\nβ 2,015\\nGame XP:\\n0\\n*, \\\"content\\\": '\\'-—————\\nVECTRA(old NYXEL)\\nStable RAT · In-house C2 · Active...

SOCRadar Platform Threat Hunting results linking the HackForums sales thread to the Vectra handle and its NYXEL reference.

The catalog extends past the RAT itself. In conversation with our team, the developer priced FUD (Fully Undetectable) crypting services between \$100 and \$350, with a bundled package including the stub and crypting quoted above \$2,000. They demonstrated scan results against named antivirus products, while noting that detection varies by product, version, and configuration.

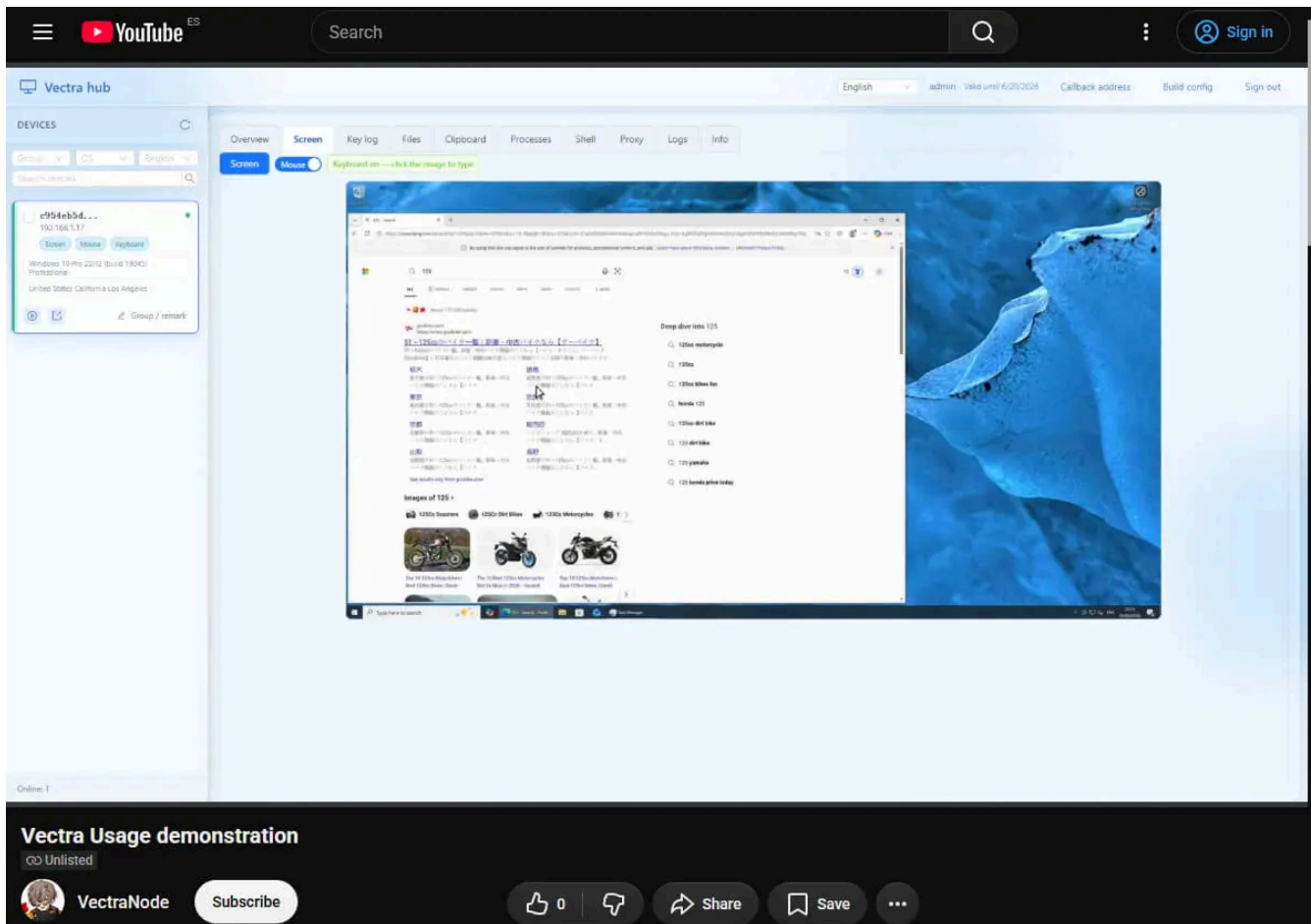
The same conversation produced a complaint that says something about the market the developer operates in: other actors have been impersonating them and selling fraudulent builds under the Vectra name. Our own search for other sellers of Nyxel or Vectra turned up no recent competitor, which suggests the impersonation is opportunistic rather than organized.

YouTube completes the sales funnel. The VectraNode channel hosts a product demonstration published on June 17, 2026, days before the first live panel appeared. Video demos have become standard practice for MaaS sellers, giving buyers something to watch before they pay.

SOCRadar customers are already covered. The sales thread behind this research, and the Nyxel reference inside it, were surfaced with [Threat Hunting](#), the same module customers use to pivot on actor handles and artifacts. VectraRAT and the Vectra/Nyxel actor are now tracked in [Threat Actor and Malware Intelligence](#), every indicator in this report is searchable in [IOC Radar](#), and [Attack Surface Management](#) alerts you if this infrastructure appears against your own assets.

Inside VectraHub: The Operator Side

VectraHub is a single Go ELF binary that carries its own web application. The Vue3 frontend, built on Ant Design components, is packaged into the executable through Go's ***embed.FS***, so the hub needs no external web server and no assets directory. Drop the binary on a VPS, run it, and the panel is live on port 8080.



VectraHub operator panel on port 8080, showing the client overview and module tabs.

Authentication uses JWT with HS256. The signing key is generated at runtime rather than hardcoded, so every hub instance has a unique secret, and issued tokens last 30 days. Real-time updates reach the panel over WebSocket: `/api/clients/ws` streams client status as JSON, while HVNC and screen sessions use separate binary channels so the keylog feed, the screen feed, and the command channel never block each other.

The panel ships with three languages: **English**, **Russian**, and **Mandarin Chinese**. That detail matters later.

Traffic between the hub and the implants runs over TCP 3308 using framing the developer wrote from scratch. Each message carries a fixed 5-byte header (type, length, padding) followed by a MessagePack payload. There is no HTTP anywhere in the exchange, which means passive inspection tools have nothing familiar to parse and signature sets built around HTTP beaconing see nothing at all.

```

/* nyxel/internal/protocol/message.go:56 */
while (&stack0x000000000 <= *(undefined1 **)(unaff_R14 + 0x10)) {
    /* nyxel/internal/protocol/message.go:56 */
    runtime.morestack_noctxt();
}

/* nyxel/internal/protocol/message.go:58 */
/* nyxel/internal/protocol/message.go:57 */
runtime.makeslice(&datatype.Uint8.uint8,5,5);
/* https://github.com/golang/go/blob/go1.26.0/src/io/io.go#L354 */
io.ReadAtLeast(uStack000000000000000008,uStack000000000000000010,extraout_RAX,5,5,5);
/* nyxel/internal/protocol/message.go:58 */
if (extraout_RBX == 0) {
    /* nyxel/internal/protocol/message.go:61 */
    runtime.newobject(&datatype.Struct.protocol.MessageHeader);
    nyxel/internal/protocol.ParseHeader(extraout_RAX,5,5);
    *extraout_RAX_00 = extraout_AL;
    *(undefined4 *)(&extraout_RAX_00 + 4) = extraout_EBX;
    /* nyxel/internal/protocol/message.go:62 */
    if (extraout_RCX != 0) {
        /* nyxel/internal/protocol/message.go:63 */
        return;
    }
    /* nyxel/internal/protocol/message.go:65 */
    return;
}

```

Five-byte header read and the ParseHeader call confirming the framing layout.

One opcode is worth naming. **0x22** carries a callback address update, with a JSON payload containing **host** and **port** fields. That single message is how an operator moves every implant to new infrastructure without rebuilding a binary.

The Stub Builder

The builder lives in the panel. An operator supplies a base stub, a C2 address, and whatever PE metadata they want the file to claim, and the hub patches all of it from Linux.

The C2 patching mechanism is blunt and effective. The stub contains one or more 256-byte null blocks acting as placeholders. The **patchExeHostReplaceRuns** function locates each block with **bytes.Index** and overwrites it with **memmove**. Icon and VERSIONINFO edits go through the public [winres](#) library, which lets a Linux host rewrite Windows PE resources without touching the Windows SDK.

Build configuration

Server address

Source file

ip port

Up to 256 UTF-8 bytes (same as client placeholder size)

filename

filename sets the output name of the built exe (matches server output).

☐ Bypass UAC

When checked, uacEnabled is written into the build config embedded in the PE; on first launch as a non-admin, the client attempts Bypass UAC elevation (not the standard run-as-admin prompt).

Note: %TEMP%\callback.json stores the client GUID. The file address is used only when callback is true; otherwise the embedded exe IP is used. Saving a callback address here sets callback:true and pushes it to clients. Default port is 3308.

Icon & assembly

Icon and version fields apply to this executable's resources and properties.

icon

Product trademarks

Description original name

company product version

copyright file version

Stub build configuration in the panel: C2 address, output name, UAC bypass, and PE metadata.

Operators who skip the metadata step leave a usable fingerprint behind. The panel's default values are Product **Vectra**, Description **Vectra client**, Company **Vectra**, Copyright **Copyright © 2026 Vectra**, and version **0.2**. Any PE carrying that set in its VERSIONINFO resources is an unmodified VectraRAT stub, and we found plenty of them during pivoting.

A checkbox labeled Bypass UAC writes **uacEnabled: true** into the JSON configuration block embedded in the PE's **RT_RCDATA** resource. Privilege escalation, in other words, is a build option.

File transfers use a staging system that issues a single-use token per operation, through **/api/file/staging/download?token=** and **/api/file/staging/complete?token=**.

The License Check

Before the hub does anything else at startup, it validates its license. Three chained functions handle it, and the signature scheme is RSA-PSS-SHA256 with a 32-byte salt: the payload JSON is compacted, hashed with SHA256, and verified against a public key embedded in the binary. Without the developer's private key, a valid license cannot be forged.

```
"payload": {  
  "product": "VectraHub",  
  "licenseId": "LIC-2026-,  
  "issuedAt": "2026-06-,  
  "expiresAt": "2026-07-,  
}
```

License payload recovered from the open directory, naming the product and its validity window.

The licenses recovered from the exposed directories expired in July and August 2026, which points to several recent purchases and an active testing phase during our analysis window.

Inside the Implant: What Runs on the Host

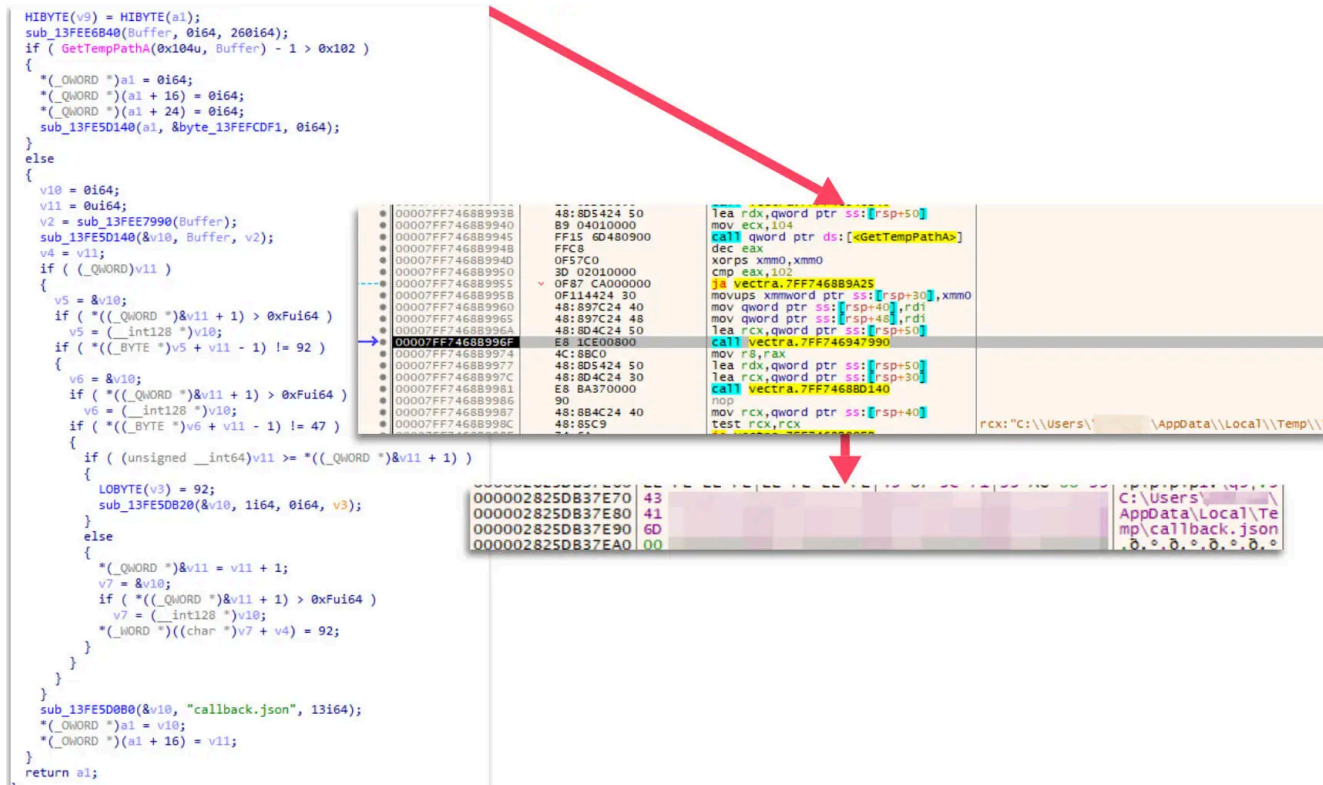
The client is a native C++ PE, multithreaded, with each module running in its own thread after initialization. GDI+ handles screen capture and Winsock handles the network.

Its startup sequence is short and worth knowing in order, because most of it is observable.

First it checks for a named mutex, ***LocalVectra.Client.SingleInstance***, and exits if it already exists. The name has stayed static across every sample we analyzed, which makes it one of the strongest host-based indicators available.

Then it calculates a per-device UUID before any connection attempt. The hub uses that UUID to recognize a returning implant as the same session rather than a new victim, so reconnections do not inflate the client list.

Next it looks for ***%TEMP%callback.json***. If the file exists and contains ***"callback": true***, the implant reads its C2 address from there instead of the address baked into the PE. The hub writes that file when an operator changes the address from the panel, which is the delivery mechanism behind opcode ***0x22***.



Lookup of %TEMP%callback.json, used to override the C2 address embedded in the PE.

Finally it connects, and checks the server's answer. If the response does not match the expected protocol format, the process terminates. That check is what blocks analysts from pointing a sample at their own hub: reproducing the session needs the right bytes, a valid license, and the developer's RSA keys.

Once the session is up, the capability set is broad:

- **Hidden desktop:** the *[ImageSender]* module captures frames with GDI+ (*CreateCompatibleBitmap*, *GdiplusBitmapGetPixel*), serializes them as RGB, compresses with LZNT1, and ships them to the hub, which converts them to JPEG for the panel. HVNC runs on a desktop isolated from the one the victim is looking at.
- **Remote shell:** CMD or PowerShell, selected by the operator from the panel.
- **Keylogging:** a global keyboard hook feeding a searchable, exportable log in the panel.
- **Clipboard control:** live clipboard sync plus configurable regex replacement rules, which is how cryptocurrency address hijacking gets implemented.
- **SOCKS5 relay:** the compromised host becomes an egress node, referred to internally as a "Slave."
- **File operations:** a remote browser that enumerates victim drives, with bidirectional transfer.
- **Process enumeration:** through a WMI query against *Win32_Process*, rendered live in the panel.

The UAC Bypass

This is the part of VectraRAT that separates it from the \$50 tier. The technique corresponds to [UACME](#) #41, with an extra evasion layer built on debug object handle hijacking. It runs in three phases.

Phase one: The implant launches *winver.exe* with the **DEBUG_PROCESS** flag, creating a kernel debug object bound to that process. **NtQueryInformationProcess** with the **ProcessDebugObjectHandle** class retrieves the handle, **NtRemoveProcessDebug** detaches the object, and the process is killed. What remains is a live debug object attached to nothing, held by the malware.

Phase two: **DbgUiSetThreadDebugObject** injects that orphaned handle into the current thread, and *computerdefaults.exe* is launched in debug mode. That binary is a Windows auto-elevating executable, so the system raises it to High Integrity with no prompt. Because the debug session reuses the pre-existing object, it never registers through the standard mechanism, and tooling watching for **DebugActiveProcess** sees nothing.

```

memset(&hProcess, 0, sizeof(hProcess));
memset(hObject, 0, sizeof(hObject));
sub_13FEE6B40(&DebugEvent, 0i64, 176i64);
sub_13FEE6B40(v34, 0i64, 1040i64);
LoadImport(v2);
if ( !lpCommandLine )
{
    v3 = (void *)*((_QWORD *)lpBuffer + 277);
    if ( v3 )
        SetEvent(v3);
    return 3221225485i64;
}
ExceptionRecord = 0i64;
sub_13FED0740(v33, 64i64, L"WinSta0\\Default");
sub_13FED0740(v34, 520i64, lpBuffer + 261);
sub_13FED07D0(v34, 520i64, &dw_13FEFFB0C);
sub_13FED07D0(v34, 520i64, L"winver.exe");
if ( (unsigned __int8)sub_13FEB0BE0((unsigned int)v34, (unsigned int)v34, 0, v6) )
{
    LoadImport(v7);
    v9 = hProcess.hProcess;
    if ( qword_13FF18400 )
    {
        v8 = qword_13FF18400(hProcess.hProcess, 30i64, &v19, 8i64);
        if ( v8 >= 0 )
        {
            if ( qword_13FF183F0 )
                qword_13FF183F0(v9, v19);
            TerminateProcess(v9, 0);
            CloseHandle(hProcess.hThread);
            CloseHandle(v9);
            memset(&hProcess, 0, sizeof(hProcess));
            sub_13FED0740(v34, 520i64, lpBuffer + 261);
            sub_13FED07D0(v34, 520i64, &dw_13FEFFB0C);
            sub_13FED07D0(v34, 520i64, L"computerdefaults.exe");
            memset(hObject, 0, sizeof(hObject));
            memset(&DebugEvent, 0, sizeof(DebugEvent));
            if ( (unsigned __int8)sub_13FEB0BE0((unsigned int)v34, (unsigned int)v34, 1, v10) )
            {
                ...
            }
        }
    }
}

```

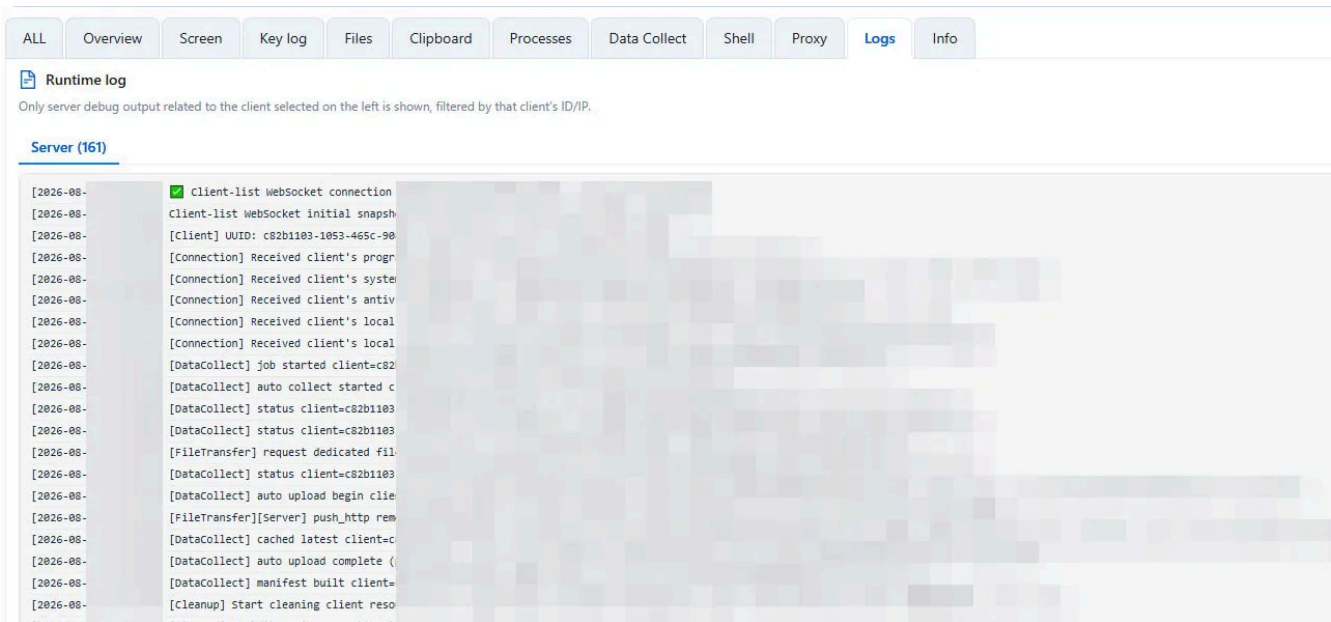
Debug object hijacking, detaching the object from winver.exe before launching computerdefaults.exe.

Phase three: The implant waits for the **CREATE_PROCESS_DEBUG_EVENT**, takes the elevated process handle from the event's **hProcess** field, duplicates it with **PROCESS_ALL_ACCESS** through **NtDuplicateObject**, and calls **CreateProcessW** with **computerdefaults.exe** declared as the parent. Windows hands the child its parent's token. The payload starts at High Integrity, and the victim never sees a dialog.

Automated Collection on First Connection

A **DataCollect** module fires as soon as the implant reaches the hub, with no operator involvement. It runs collectors for Chromium, Firefox, and Internet Explorer credentials, captures active network connections through netstat, and sweeps for **.env**, **.conf**, and **.config** files, the usual homes for API keys, database strings, and infrastructure secrets.

Everything lands in the hub as structured per-client files, viewable in the panel or downloadable in bulk. From there the operator decides whether a host is worth working by hand.



Runtime log tab in the panel, showing the DataCollect sequence for a connected client.

Campaigns: Amadey, ClickFix, and Vidar

VectraRAT is in the hands of multiple operators running very different playbooks, and none of the campaigns we documented belong to the developer. They belong to buyers.

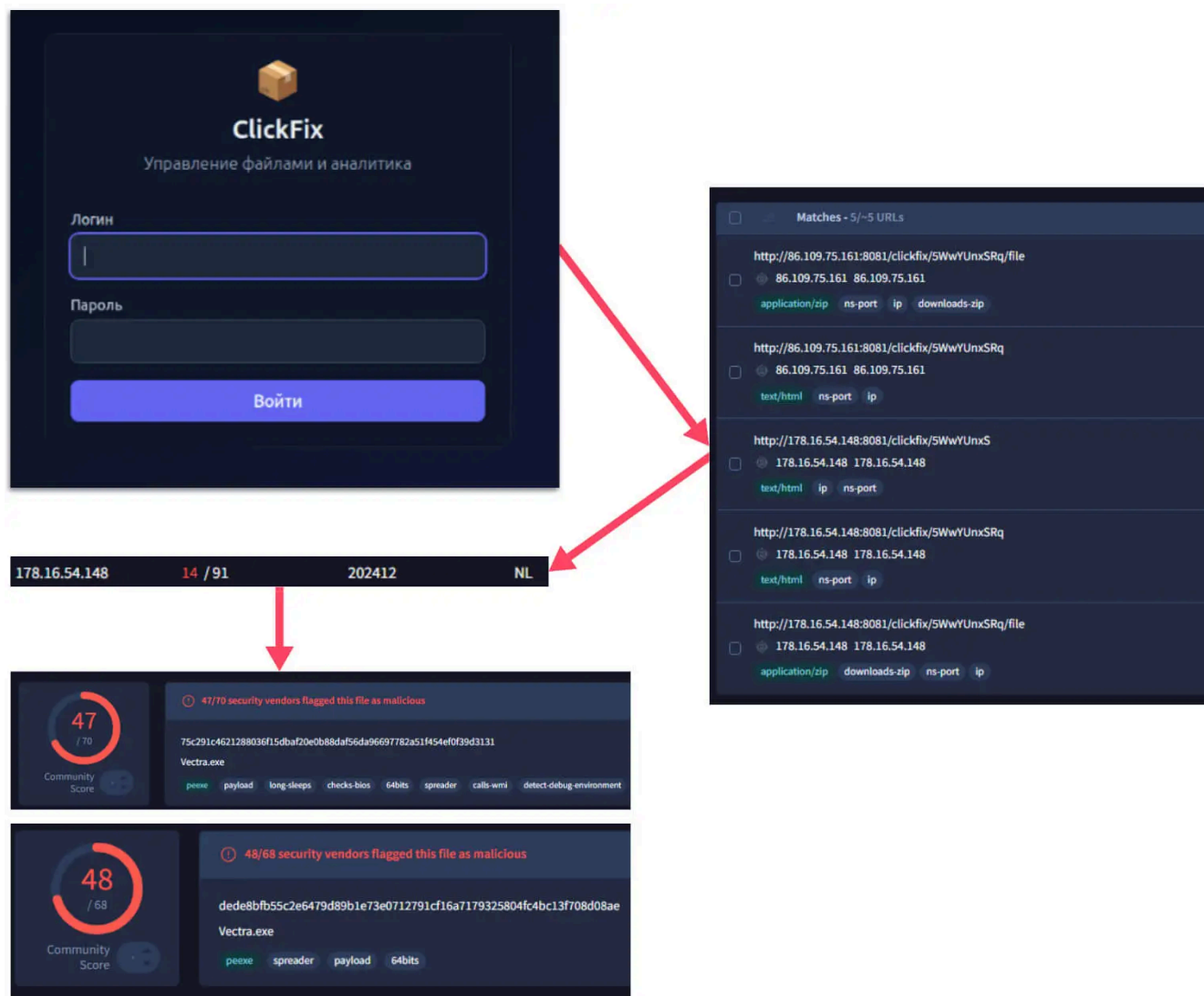
Amadey: One operator uses the Amadey loader, a MaaS staple documented since 2018, to pull VectraRAT onto hosts it has already compromised. The Amadey panel sits at 91.92.242.236 behind a Russian-language login, with a staging URL serving **Vectra.exe**. Swapping or supplementing Amadey’s usual stealer payload with VectraRAT trades a single credential grab for a persistent channel with HVNC, shell, and proxy access.

MALWARE FAMILY	URL
Amadey	91.92.242.236/oPvj94jfe/Login.php

Amadey panel identified at 91.92.242.236, the staging point for the VectraRAT download.

ClickFix: A second operator delivers VectraRAT through [ClickFix](#), the fake CAPTCHA technique that convinces a user to paste and run a command themselves. The panel, again Russian-language, sits at 178.16.54.148 on Omegatech LTD (AS202412), and the lures impersonate TurboTax. That choice lines up with the victim geography: the US tax market is the target.

The same infrastructure also distributes NetSupport RAT, so the panel functions as a general-purpose delivery platform rather than a VectraRAT-only channel. The domain **verify-cloud.digital**, resolving to 86.109.75.161, is flagged across multiple sources as both a phishing domain and a NetSupport vector, and the payload arrives through a PowerShell one-liner served from **/clickfix/5WwYUnxSRq/file** on port 8081.



Russian-language ClickFix panel at 178.16.54.148 and the delivery URLs serving VectraRAT.

Vidar: At 195.63.145.106 (AS216071, Servers Tech Fzco), Vidar and VectraRAT samples share TCP 3308. The pattern matches Amadey: an infostealer for the initial credential sweep, the RAT for durable access to the hosts worth keeping.

Infrastructure: One Provider, Two Roles

The infrastructure story has one detail that is hard to read as coincidence. The VectraRAT C2 and panel sit at 86.109.75.168. The ClickFix distribution panel sits at 86.109.75.161. Same /24, **same provider**, a provider with a long record of tolerating abuse complaints.

Whoever runs that pair is centralizing an entire operation with one host: the tool that controls victims and the page that creates them.

The open ports on the main node were TCP 3308 (C2 protocol), 8080 (panel), 4457, and 6667. Port 4457 appears in the internal protocol as an auxiliary channel. Port 6667, normally IRC territory, lines up with the SOCKS relay references found in the hub binary.

A second cluster turned up at 195.20.115.77 on ServerAstra (AS56322), with both the VectraHub login page and the directory publicly reachable, holding Vectra samples renamed around tax themes. Pivoting on that naming convention pulled in unrelated actors using the same seasonal hooks, a reminder that tax-themed filenames are shared territory rather than an attribution signal. Further pivoting reached 91.219.236.179, also on ServerAstra.

195.20.115.77:8000 10h3c... 999+

Directory listing for /
195.20.115.77
United States of America / - / -

56322
ServerAstra Kft.
2026-07-08
SimpleHTTP/0.6 Python/3.11.2

Header Products

HTTP/1.0 200 OK
Connection: close
Content-Length: 744
Content-Type: text/html; charset=utf-8
Date: Wed, 08 Jul 2026 09:00:21 GMT
Server: SimpleHTTP/0.6 Python/3.11.2

86.109.75.168 +p8/... +

Directory listing for /
86.109.75.168
United States of America / California / Los Angeles
53850
GorillaServers, Inc.
2026-06-22
SimpleHTTP/0.6 Python/3.12.3

Header Products

HTTP/1.0 200 OK
Connection: close
Content-Length: 672
Content-Type: text/html; charset=utf-8
Date: Sun, 21 Jun 2026 16:22:13 GMT
Server: SimpleHTTP/0.6 Python/3.12.3

Directory listing for /

- [.data/](#)
- [license.json](#)
- [license_public.pem](#)
- [logs/](#)
- [nohup.out](#)
- [TurboTax.exe](#)
- [TurboTax2025.exe](#)
- [TurboTax2026.exe](#)
- [Vectra.exe](#)
- [Vectra Customer Installation and Usage Guide.txt](#)
- [VectraHub](#)

Vectra hub English

Username

Password

Sign in

Open directory and VectraHub login page exposed at 195.20.115.77 on ServerAstra.

Who Is Vectra?

The public identity is only a few months old. The sales threads went up on HackForums and Exploit.in in early June 2026, titled **VECTRA(old NYXEL)**.

That parenthetical is the actor doing our attribution work for us, and the binaries confirm it three times over:

- **The Go module:** the hub's ELF declares its module as *nyxel*, visible in the *.gopclntab* symbol and in package paths such as *nyxel/screen* and *nyxel/internal/protocol*. Module names are fixed at compile time, so changing this would mean rebuilding the entire hub.
- **Panel storage keys:** the Vue3 frontend still writes *nyxel_workspace_tab_v2* and *nyxel_client_list_snapshot_v1*, hardcoded in JavaScript that is readable from any live server.
- **Server environment variables:** the same original name persists in the hub's runtime configuration.

The rebrand, in short, is marketing deep and no deeper. The developer has also been observed presenting the old Nyxel Hub and the new VectraHub side by side, with no functional difference between them.

The HackForums profile contains a contradiction that is itself an indicator. The account carries **n00b@HF** status with zero credibility and zero experience points, the profile of a complete newcomer. The product attached to it has a proprietary protocol, a full web panel, an RSA-signed licensing system, and a privilege escalation chain built on undocumented ntdll calls. Newcomers do not arrive with that. The gap is consistent with identity migration: an established actor opening a clean account to launch a rebranded product.

On Exploit.in the same handle is considerably more active, posting module updates and arguing with buyers who report bugs, mostly in Russian.

[Rent] Vectra RAT WebPanel/HVNC/PROXY/SHELL/

Jun 16, 2026 11:17 PM (This post was last modified: Jun 17, 2026 11:10 AM by Vectra.)

Vectra

[n00b@HF:]

L33T

Posts: 1

Threads: 1

Credibility: 0

Popularity: 0

Bytes: 2,015

Game XP: 0

VECTRA(old NYXEL)

Stable RAT · In-house C2 · Actively maintained

Vectra(old Nyxel) is a C2 Panel + Native Client stack — not a leaked-source repack, not an open-source RAT reskin.

▸ Hub (C2 Panel)

Server-side deploy, built-in WebUI Operator Console. Handles client check-ins, session management, and command dispatch.

▸ Node (Client)

Native Win32 implant on target Windows hosts (Linux client on the roadmap). Persistent process, TCP callback to C2, executes operator commands and exfiltrates data back.

▸ Full in-house stack

Protocol, panel, client, WebUI — all built from scratch. Callback chain and session keepalive tuned over time: stable check-ins, clean reconnect on drop, long sessions that actually hold. Command modules are pluggable — add new capability without touching the transport layer. Active dev, updates pushed regularly.

Delivery: .exe · no source

WHY VECTRA(old NYXEL)

▸ Stable check-ins

TCP long-lived connection + heartbeat keepalive · mutex single-instance · auto-reconnect on disconnect

▸ In-house architecture

Custom binary protocol · in-house C2 · in-house client · in-house panel

▸ Modular expansion

New commands/channels without rewriting core

▸ Active maintenance

Bugfixes and new modules land fast

STACK

Hub — C2 Panel

Go · embedded Vue3 / Vite / Ant Design WebUI

HTTP/WebSocket (Operator Console) · TCP (client callback)

Custom binary protocol · JWT auth · RSA license

Node — Client

C++ / Win32 · GDI+ capture · Winsock TCP

Custom binary protocol

HUB — C2 PANEL FEATURES

Spin up Hub, hit the WebUI in your browser. Left pane: live client list + real-time session status. Right pane: operator workspace for the selected host.

- ## Where the Developer Sits

- **Chosen geolocation APIs:** two of the four services queried by the hub, *ip.cn* and *my.ip.cn*, are oriented almost exclusively toward the Chinese market. A developer outside that ecosystem would be unlikely to reach for them.

Pointing the other way: the forum presence, the working hours, the Russian strings observed during operations, and the colloquial Russian syntax in conversation all match the modus operandi of Russian-speaking operators, though the actor also shows clear signs of relying on translators or AI.

One further oddity sits alone. The YouTube demo's **defaultLanguage** metadata field is set to *ja*, reflecting the uploader's account interface language at the time. It is the weakest of the indicators and stands only as a possible sign of East Asian language use.

The honest conclusion is that the technical evidence points toward a developer with an origin or connection in the East Asian linguistic sphere, operating inside a Russian-speaking crimeware market. Nationality cannot be established with confidence from what we have.

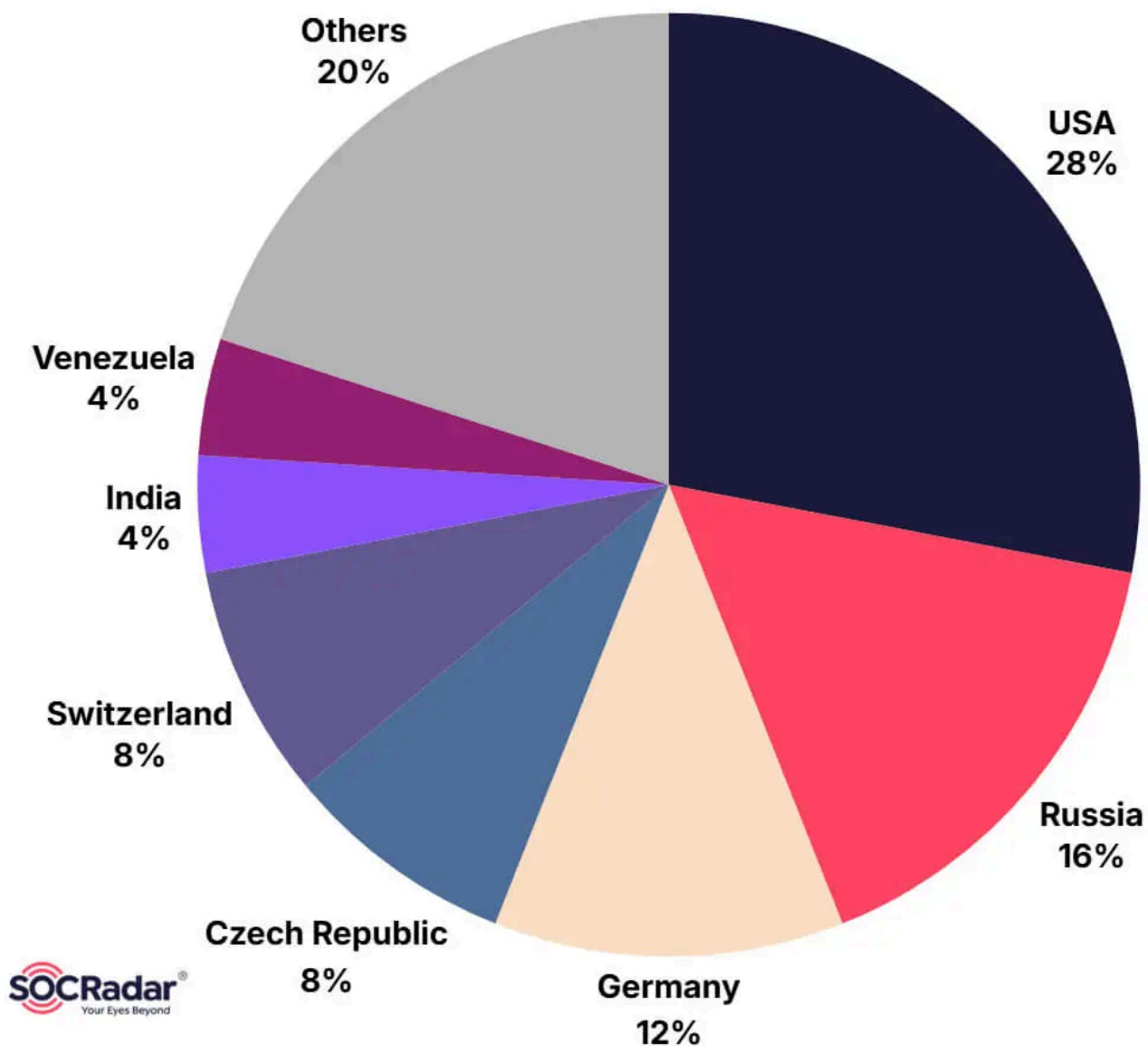
Victimology: 48% Corporate Windows

The victim data comes from client records pulled from live C2 infrastructure, covering a window of no more than one week. Scanner traffic (Censys, GreyNoise), Tor nodes, and the actor's own address ranges were removed before counting.

Metric	Count
Total recorded sessions	50+
Genuine victim sessions	38
Victims with confirmed exfiltration	8
Exfiltrated file transfers	21

Extrapolating the primary C2's observed rate of 7.6 sessions per day across 30 days puts unique victims in the 150 to 300 range. Adding the confirmed secondary C2 and the separate ClickFix and Amadey delivery channels pushes the real figure higher.

Geographically, the United States leads with seven unique victims, followed by Russia with four and Germany with three, with further hits in Switzerland, the Czech Republic, India, and Venezuela. The spread suggests broad distribution rather than targeting, with operators selecting which established implants are worth working.



Geographic distribution of identified victims.

The operating system breakdown is the finding that matters most. **48% of victim entries with recorded system information correspond to corporate Windows editions**, including Enterprise, Enterprise LTSC, IoT Enterprise LTSC, and Windows Server 2025. That is not the distribution a consumer software lure produces.

Two cases illustrate what that means in practice. In the Czech Republic, a Windows Server 2025 host had six file transfers executed from it in under 25 minutes, alongside panel-issued commands and the self-deletion of a secondary payload afterward, which reads as coordinated exfiltration across servers belonging to one organization. Separately, two endpoints in Arlington, Virginia fall inside the same network range. Two hosts are not enough to claim government targeting, and we are not claiming it. What they do show is how far outside the consumer bracket this implant is landing.

One panel log carries a small human detail worth keeping. A victim file path contained the Russian word *путь*, meaning a Cyrillic user profile directory on a compromised machine, which places a Russian-speaking victim inside an operation whose lure was a US tax product.

Detection and Hunting Guidance

VectraRAT leaves a usable trail, provided you know what to look for. The following are the highest-value checks from the analysis:

- **Named mutex:** hunt for ***LocalVectra.Client.SingleInstance***. It has not changed across samples.
- **C2 override file:** the presence of ***callback.json*** in ***%TEMP%***, particularly containing a ***callback*** field set to true.
- **Outbound TCP 3308:** unexpected long-lived sessions to a non-standard port with no recognizable application-layer protocol. Multiple simultaneous connections from one process to one endpoint is part of the design.
- **Auto-elevation abuse:** ***computerdefaults.exe*** running with a child process it has no reason to spawn, or ***winver.exe*** launched and terminated in quick succession under a debug flag.
- **Debug API sequence:** ***NtQueryInformationProcess*** with ***ProcessDebugObjectHandle***, followed by ***NtRemoveProcessDebug*** and ***DbgUiSetThreadDebugObject*** in the same process.
- **Default stub metadata:** PE files whose VERSIONINFO reports Product ***Vectra***, Company ***Vectra***, or version ***0.2***.
- **Hidden PowerShell:** execution with ***-NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden***, especially followed by the deletion of the file just run.
- **WMI process sweeps:** queries selecting ***ProcessId, Name, WorkingSetSize, KernelModeTime, UserModeTime, ExecutablePath, CommandLine*** from ***Win32_Process***.
- **Secrets file access:** bursts of reads across ***.env***, ***.conf***, and ***.config*** files on developer and server hosts.
- **ClickFix behavior:** PowerShell spawned from the Run dialog shortly after a clipboard write, the signature of a paste-and-run lure.

On the human side, ClickFix remains a user-awareness problem before it is a tooling problem. A verification page that asks someone to open the Run dialog and paste a command is never legitimate, and that single rule closes the delivery path used in the most active campaign we documented.

Indicators of Compromise

Remaining indicators, including sample-level detail and infrastructure enrichment, are tracked on the [SOCRadar platform](#).

Hash Values (SHA-256)

e2db5db12564d2a9da7ef3a57aa23d95782f5eaddc8bd35eb7c35ae6b844a0f0
dede8bfb55c2e6479d89b1e73e0712791cf16a7179325804fc4bc13f708d08ae
ddb6d636f6dfd475dc0c75bf9f6f873fa35b9062dee9d37b3377ae7b9acdcd0c9
b926cfc3d3f4b07fe6001c39f46e40225ff8000411198d13782538d770c54ae5e
bff3583d04f0d5603ced9831eb7c45c1923bd90e2f7d5e5d2b32942d38cf6dc5
c708d413720848f8788f43a4f47ddce016fca9af10c9ba4113f47bf2c9244dc5
8745e872ff8aa41b0e03737f76bf35b6c934106c987dff98afe34120e47caf91
bba58f99e14e3512c04a5a74a079d7851abf935dd258cff4c80874ce7cfc82e3
d8f15ba122cd6da01f83fe05294df80a6eadbce0f66dac7c2bcc0904f066e0de
3ab56c9fb6b7c404c1e5b36788959c877ea819fb124c3847fa0498e9915ef9a7
7b82f08120e0d9b16cd5b9ec59d24fb68e35735c82311a233d370e7f264af650
b738c03fef5e3d26419e4aab1818a0a7ad206c67fb3eedcdfb3ef1ee07eb620

IP Addresses

Address	Role
86.109.75.168	Primary C2 and panel, exposed open directory (GorillaServers, AS53850)
86.109.75.161	ClickFix distribution panel, resolves <i>verify-cloud.digital</i>
178.16.54.148	ClickFix panel serving VectraRAT and NetSupport RAT (Omegatech LTD, AS202412)
195.20.115.77	Secondary cluster, exposed panel and directory (ServerAstra, AS56322)
91.219.236.179	Related infrastructure (ServerAstra)
91.92.242.236	Amadey panel and VectraRAT staging (Omegatech LTD, AS202412)
195.63.145.106	Shared TCP 3308 infrastructure with Vidar (Servers Tech Fzco, AS216071)

Domain

verify-cloud.digital

Host and Network Artifacts

Type	Value
Mutex	<i>LocalVectra.Client.SingleInstance</i>
File	<i>%TEMP%callback.json</i>
PE resource	<i>RT_RCDATA</i> 1001, JSON configuration block with <i>uacEnabled</i>
C2 port	TCP 3308
Panel ports	TCP 8080, 8888
Auxiliary ports	TCP 4457, 6667
Default PE metadata	Product Vectra , Company Vectra , version 0.2
ClickFix path	<i>/clickfix/5WwYUnxSRq/file</i> on port 8081

MITRE ATT&CK TTPs

Tactic	Technique	Procedure
TA0001: Initial Access	T1566.002: Phishing: Spearphishing Link	An operator distributes VectraRAT through ClickFix lures hosted on domains such as verify-cloud.digital.
TA0002: Execution	T1059.001: Command and Scripting Interpreter: PowerShell	The remote shell module executes PowerShell with - NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden under operator instruction, confirmed in panel logs through secondary payload execution.
TA0002: Execution	T1059.003: Command and Scripting Interpreter: Windows Command Shell	The remote shell module supports CMD as an alternative interpreter, selectable from the panel through the shell field in the JSON request.

Tactic	Technique	Procedure
TA0002: Execution	T1106: Native API	The UAC bypass uses undocumented ntdll functions including NtQueryInformationProcess, NtRemoveProcessDebug, NtDuplicateObject, and DbgUiSetThreadDebugObject, avoiding standard Win32 wrappers.
TA0004: Privilege Escalation	T1548.002: Abuse Elevation Control Mechanism: Bypass User Account Control	VectraRAT implements a bypass equivalent to UACME #41, combining debug object handle hijacking on winver.exe with parent process spoofing on computerdefaults.exe to obtain a High Integrity token without user interaction.
TA0004: Privilege Escalation	T1134.004: Access Token Manipulation: Parent PID Spoofing	The final payload is launched through CreateProcessW with PROC_THREAD_ATTRIBUTE_PARENT_PROCESS pointing to the duplicated handle of computerdefaults.exe, inheriting its elevated token.
TA0005: Defense Evasion	T1036.005: Masquerading: Match Legitimate Resource Name or Location	Stubs have been distributed with tax-related naming such as TurboTax2026.exe to reduce user suspicion in North American campaigns during tax season.
TA0005: Defense Evasion	T1036.001: Masquerading: Invalid Code Signature	The stub builder customizes PE VERSIONINFO metadata and recalculates the PE checksum through ApplyStubPE.ForceChecksum and ApplyStubPE.WithAuthenticode from Linux.
TA0005: Defense Evasion	T1140: Deobfuscate/Decode Files or Information	The UAC bypass configuration is stored encrypted in the PE's RT_RCDATA 1001 resource as JSON, parsed in memory through cJSON at runtime without writing to disk.

Tactic	Technique	Procedure
TA0006: Credential Access	T1555.003: Credentials from Password Stores: Credentials from Web Browsers	The DataCollect module includes collectors for Chromium, Firefox, and Internet Explorer that extract stored credentials and exfiltrate them on first connection.
TA0006: Credential Access	T1056.001: Input Capture: Keylogging	The [Keyboard] module installs a global keyboard hook recording all keystrokes in real time, with paginated search and export in the panel.
TA0006: Credential Access	T1552.001: Unsecured Credentials: Credentials In Files	The editor_config module collects .env , .conf , and .config files, common locations for API credentials, database connection strings, and infrastructure secrets.
TA0007: Discovery	T1057: Process Discovery	The implant enumerates processes through a WMI query against Win32_Process, with results displayed live in the Processes tab of the panel.
TA0007: Discovery	T1082: System Information Discovery	On first connection the implant transmits operating system, build number, edition, computer name, public IP address, and installed antivirus.
TA0007: Discovery	T1049: System Network Connections Discovery	The netstat module of DataCollect captures active network connections as part of initial automated collection.
TA0009: Collection	T1113: Screen Capture	The [ImageSender] module captures desktop frames through GDI+, compresses them with LZNT1, and transmits raw RGB data to the hub, which converts it to JPEG for the panel.
TA0009: Collection	T1115: Clipboard Data	The clipboard module synchronizes victim clipboard content with the panel in real time, with configurable regex replacement rules supporting cryptocurrency address hijacking.

Tactic	Technique	Procedure
TA0009: Collection	T1119: Automated Collection	The DataCollect module executes automatically on first connection to the hub, without operator intervention.
TA0010: Exfiltration	T1041: Exfiltration Over C2 Channel	File transfers are exfiltrated through the same TCP 3308 channel using a staging system with single-use tokens.
TA0011: Command and Control	T1090.001: Proxy: Internal Proxy	A SOCKS5 relay turns the compromised host into an egress node, identified internally as “Slave” in the hub code.
TA0011: Command and Control	T1095: Non-Application Layer Protocol	Hub and implant communications use a proprietary binary TCP protocol with a 5-byte framing header and MessagePack payloads, without HTTP.
TA0011: Command and Control	T1571: Non-Standard Port	The C2 protocol operates over TCP 3308, avoiding HTTP proxy inspection and reducing the likelihood of blocking under standard port policies.
TA0011: Command and Control	T1568: Dynamic Resolution	The C2 address is updated dynamically through <i>callback.json</i> in <i>%TEMP%</i> , letting the hub overwrite the embedded address without redistributing a binary.
TA0011: Command and Control	T1219: Remote Access Tools	The HVNC module establishes a hidden desktop session isolated from the visible desktop, allowing keyboard, mouse, and application control without alerting the victim.

Conclusion

VectraRAT is what a professionalized MaaS market looks like in 2026. The tools that used to require in-house development, a custom protocol, a purpose-built panel, a working UAC

bypass, and a licensing system that survives resale attempts are now rentable for \$250 a month by operators who could not write any of it themselves.

The four-year gap is the part worth sitting with. The developer's YouTube channel has existed since August 2022, and the product only surfaced publicly in June 2026 because someone left a directory open. Everything documented here was already in use before anyone outside the operation had a name for it.

And it is landing on serious hosts. Nearly half the victim entries we recovered are corporate Windows builds, with confirmed exfiltration from Windows Server 2025 in enterprise networks, over a sampling window of less than a week from short-lived servers. Chained behind Amadey or a ClickFix page, the outcome of a \$250 subscription runs from credential theft to persistent access inside corporate infrastructure.

SOCRadar platform users can track VectraRAT and Nyxel indicators against their own environment and receive alerts if this infrastructure appears in traffic to or from their assets, with the actor and malware profiles updated as the operation develops.

For the full investigation, including the decompiled protocol analysis, panel artifacts, license internals, and the complete indicator set, [download the full VectraRAT report \(PDF\)](#).

Frequently Asked Questions

Is VectraRAT based on AsyncRAT, XWorm, or QuasarRAT?

No. Both components were written from scratch by the same developer, in different languages, with a custom protocol between them. It shares no code lineage with leaked or open-source RAT families.

Who operates VectraRAT?

An actor using the handle "Vectra" on HackForums and Exploit.in, previously known as "Nyxel." The rebrand is confirmed by leftover references inside the binaries, including the Go module name, panel storage keys, and server environment variables. Technical indicators point toward a developer working in Mandarin Chinese, while their forum behavior matches Russian-speaking crimeware markets. Nationality is not established.

How much does VectraRAT cost?

Rental starts at \$250 per month, in tiers. The same actor also sells crypting services between \$100 and \$350, with a bundled stub and crypting package quoted above \$2,000.

How is VectraRAT delivered?

Through buyers' own campaigns rather than the developer's. Observed vectors include the Amadey loader as a second-stage download, ClickFix fake verification pages using TurboTax lures, and tax-themed executable filenames. Vidar has been seen on shared C2 infrastructure.

What does VectraRAT do on an infected machine?

It runs hidden desktop (HVNC) sessions, logs keystrokes, opens CMD or PowerShell shells, transfers files in both directions, relays traffic through a SOCKS5 proxy, hijacks and rewrites clipboard content, enumerates processes, and automatically collects browser credentials, network connections, and configuration files on first connection.

How does the VectraRAT UAC bypass work?

It hijacks a kernel debug object created for **winver.exe**, injects that handle into its own thread, and uses it to debug **computerdefaults.exe**, a Windows binary that auto-elevates without a prompt. It then duplicates the elevated process handle and spawns its payload with that process declared as parent, inheriting a High Integrity token. No user interaction is required. The technique is equivalent to UACME #41 with an added evasion layer.

How can VectraRAT be detected?

The strongest host-based indicators are the static mutex **LocalVectra.Client.SingleInstance**, the presence of **%TEMP%callback.json**, and **computerdefaults.exe** spawning unexpected children. On the network side, look for long-lived outbound sessions on TCP 3308 carrying no recognizable application-layer protocol. Stubs built with default metadata also report Product and Company values of **Vectra**.

Who does VectraRAT target?

There is no narrow geographic or sector targeting. Distribution is broad, with the United States, Russia, and Germany most represented in the victim data, and operators selecting high-value hosts for hands-on exfiltration afterward. Notably, 48% of victim entries correspond to corporate Windows editions, including Windows Server 2025.

Hunt Threats Like a Pro

Easily Search
Hackers' Footprints

→ Start Free Trial