

MedusaHVNC: A Hidden Desktop That Steals Live Windows Sessions

 blackfog.com/medusahvnc-a-hidden-desktop

Darren Williams

July 27, 2026



|Last Updated: July 28th, 2026|6 min read|Categories: [Data Exfiltration](#), [Breach](#), [Exploits](#)

Contents

We recently came across a sample of MedusaHVNC, a new remote access trojan (RAT) being sold as malware-as-a-service (MaaS). When we took it apart, we found a hidden virtual network computing (HVNC) module that opens a browser on a separate Windows desktop, out of sight of the victim.

The browser still runs on the victim's device, so it can load an existing profile, including cookies and session state. This gives the operator access to live, logged-in sessions while the activity continues to come from the victim's usual machine.

The seller's website and operator panel show what MedusaHVNC claims to offer. Our sample reveals the other side of it, a five-stage infection chain ending in a native 64-bit payload with a hard-coded command-and-control (C2) server.

The Marketplace Listing and the Operator Panel

MedusaHVNC is promoted through a dedicated website and Telegram channel. Its advertised features include several functions associated with credential and session access.

“Mem Exec” is described as a way to run .NET and native payloads in memory, with an AMSI patch and ETW bypass. “Browser Recovery” is advertised as being able to extract passwords, cookies and browsing history.

The seller also lists Chrome, Edge, Brave, Firefox and Telegram as supported applications.

MEDUSA RAT

x64 NativeSyscalls

Professional-grade remote administration. Full remote control, silent operation, and a comprehensive toolkit — all from a single panel.

CORE MODULES

File Manager
Upload, download, browse, execute

Registry Manager
Browse, create, edit, delete keys

Process Manager
View, inspect, terminate targets

Clipper
Clipboard intercept + crypto swap

Reverse Proxy
SOCKS tunnel via target

Keylogger
Silent keystroke capture

VNC
Live desktop, mouse/key toggle

HVNC
Hidden desktop, invisible session

Drop & Exec
Upload and execute to disk

Mem Exec
.NET / native, direct syscalls

Browser Recovery
Passwords, cookies, history

HVNC — Hidden Virtual Desktop

Spawns a cloned, invisible Windows desktop session. The target sees nothing. Full browser and app sessions run in parallel with profile cloning. Clones Chrome, Edge, Brave, Firefox, and Telegram while they are actively logged in.

ChromeEdgeBraveFirefoxTelegram

OutlookOutlook (New)ThunderbirdFoxmail (auto-close before HVNC)

MEMORY EXECUTION

Mem Exec — .NET & native PE via syscallsAMSI PatchETW BypassNo disk write

ACCESS PLANS

All modules included · No feature gating

1 Month\$175 / month

3 Months\$450 \$150/mo · Save \$75

6 MonthsMOST POPULAR\$840 \$140/mo · Save \$210

12 Months\$1,500 \$125/mo · Save \$600

Lifetime\$2,400 One-time · All future updates

USD · Crypto payment · Delivery within minutes

View Plans & Get Access →

medusahvnc.com

Recovery: Chrome · Edge · Brave · Firefox · Opera · Avast HVNC: Chrome · Edge · Brave · Firefox · Telegram

© 2026 Medusa

Figure 1: The MedusaHVNC sales page.

The client panel brings those options into one interface. An operator can select an application and configure the hidden desktop session.

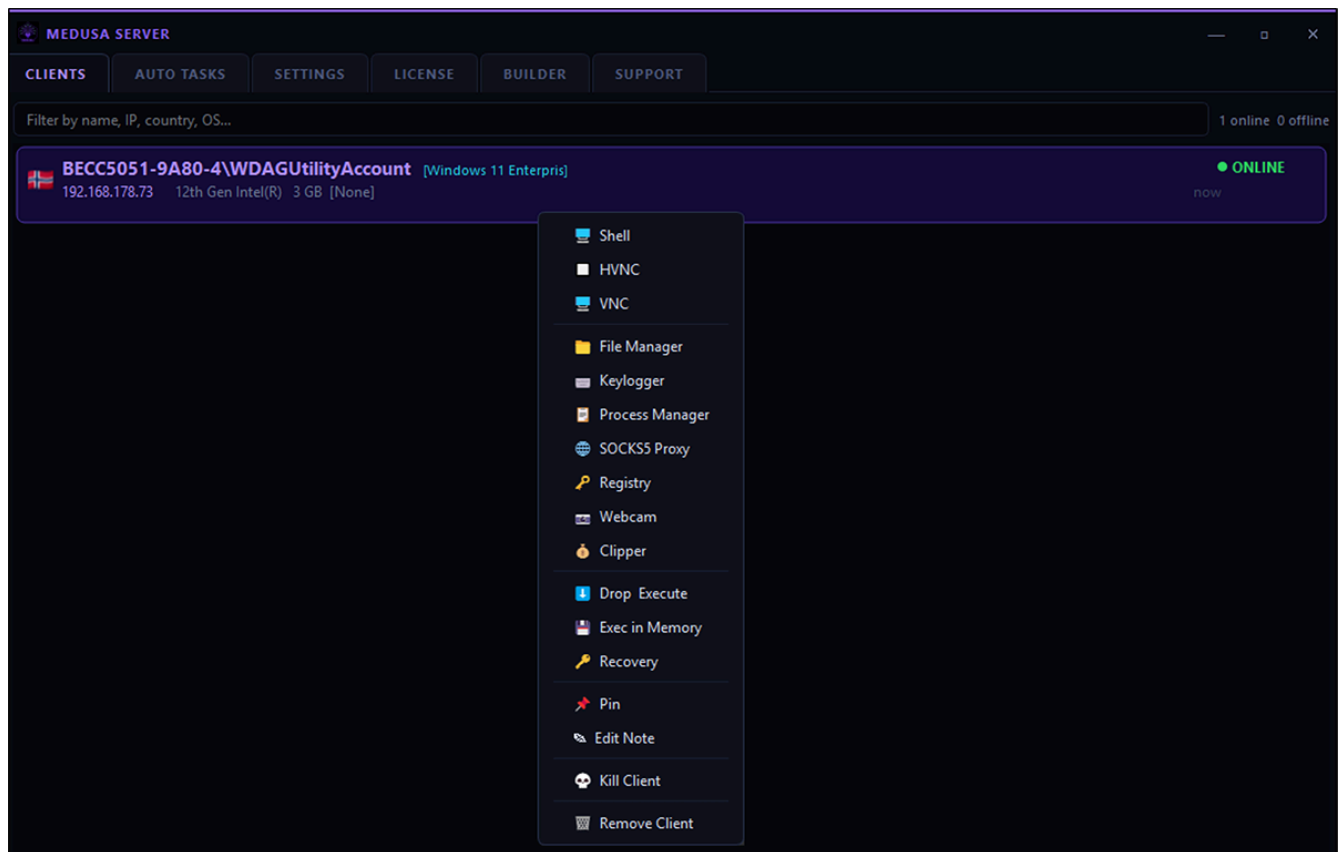


Figure 2: MedusaHVNC client options.

In a video we obtained, Google Chrome is open on Google's homepage inside a hidden desktop. The browser appears to be signed in to an account, and the panel includes controls for frame rate and image quality.

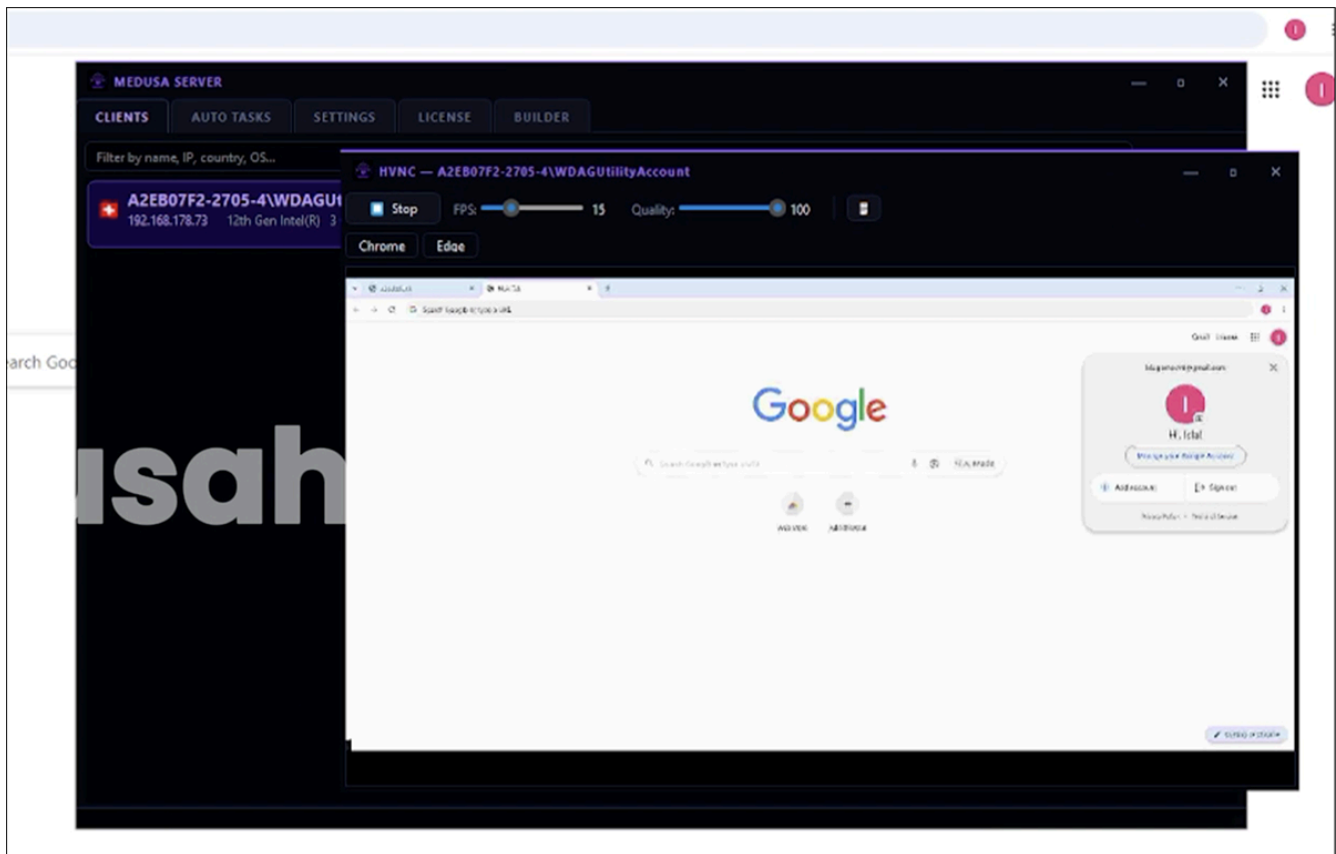


Figure 3: An active MedusaHVNC session.

The video captures the operator's view once the hidden session is running. The sample itself reveals how MedusaHVNC builds it.

From JScript to a Hidden Implant

The chain starts with obfuscated JScript. We worked through each stage offline and checked our findings against available sandbox data.

1. JScript Launcher

The chain begins when Windows Script Host, **wscript.exe**, executes the JScript launcher. The script pauses for 7,584 milliseconds before rebuilding its embedded files under **%TEMP%\Nx2981Okkr2**.

2. Staging and Persistence

The script then writes several components to disk. These include an AutoIt interpreter named **eepcxlhgdz.exe**, a configuration file, and an encrypted payload with no file extension named **zorsxklxfehdoals**. It also places **AFLivOscPj.bat** in the Startup folder. That gives the infection a way to run again after the system restarts.

3. Autolt Decryption

The next stage uses Autolt, a legitimate Windows automation tool. Running without a visible window, the Autolt component decrypts the dropped payload with a single-byte XOR key, **0xAE**. That produces the first native 64-bit executable in the chain.

4. Character Map Launch

The Autolt stage then starts **C:\Windows\System32\charmap.exe**, the standard Windows Character Map utility. The public sandbox process tree shows **charmap.exe** as a child of the Autolt process, and the loader is injected into it, using the trusted system binary as a host for the payload. The same sandbox run records screenshot capture during this stage.

5. Layered Unpacking

The loader now running inside **charmap.exe** contains two further encryption layers before the final payload. The first applies a 16-byte repeating XOR operation to 1,009,152 bytes from the **.data** section. The second uses ChaCha20 to decrypt 998,912 bytes of ciphertext with a 32-byte key, a 12-byte nonce, and an initial counter value of 1.

We replayed that unpacking process offline and recovered the final payload byte for byte. The recovered keys and offsets are redacted in the example below:

```
#!/usr/bin/env python3
"""Offline replay of MedusaHVNC's unpacking chain.

Keys and offsets were recovered statically from the injected loader
(redacted here); nothing below executes live malware code.
"""

import hashlib
from Cryptodome.Cipher import ChaCha20

XOR16_KEY = bytes.fromhex("<16-byte repeating-XOR key>") # redacted
CHACHA_KEY = bytes.fromhex("<32-byte ChaCha20 key>") # redacted
CHACHA_NONCE = bytes.fromhex("<12-byte nonce>") # redacted
DATA_OFF, DATA_LEN = "<.data offset>", 1_009_152 # encrypted blob in .data
CT_OFF, CT_LEN = "<blob offset>", 998_912 # ciphertext within the
blob
FINAL_SHA256 =
"e261888f0807b9b372b54aa45487cb5786a939fa3e87094ff270b6b8db5fdfb6"

def rep_xor(buf: bytes, key: bytes) -> bytes:
    return bytes(b ^ key[i % len(key)] for i, b in enumerate(buf))

# 1. AutoIt stage: single-byte XOR over the dropped blob.
blob = open("zorsxklxfehdoals", "rb").read()
stage1 = bytes(b ^ 0xAE for b in blob)

# 2. Injected loader: 16-byte repeating XOR over 1,009,152 bytes from .data.
data = rep_xor(stage1[DATA_OFF:DATA_OFF + DATA_LEN], XOR16_KEY)
ct = data[CT_OFF:CT_OFF + CT_LEN]

# 3. ChaCha20: 32-byte key, 12-byte nonce, initial block counter 1
# ____ (seek past the first 64-byte keystream block).
cipher = ChaCha20.new(key=CHACHA_KEY, nonce=CHACHA_NONCE)
cipher.seek(64)
payload = cipher.decrypt(ct)

assert hashlib.sha256(payload).hexdigest() == FINAL_SHA256
open("medusahvnc-payload.bin", "wb").write(payload)
print(f"recovered {len(payload)} bytes, sha256={FINAL_SHA256}")
```

Figure 4: Redacted Python excerpt showing the offline unpacking process.

Recovering the final payload allowed us to move from the delivery chain to the HVNC functionality itself. The file is an unsigned PE32+ x86-64 console executable containing a **.pay** section and the family string MedusaHVNC.

The payload communicates with its operator over a custom TCP protocol using native Winsock functions, including **WSAStartup**, **socket**, **connect**, **send**, and **recv**. The C2 address **51.89.204.28:4444** is hard-coded into the binary.

The rest of the binary contains the components needed to interact with applications running on a separate Windows desktop.

Browser-specific strings reference Chrome, Edge, and Firefox. Functions such as **BitBlt**, **EnumWindows**, and **PrintWindow** support screen and window capture, while **SendInput** and **SetWindowsHookExW** are associated with synthetic input and interaction.

Clipboard functions, including **OpenClipboard**, **GetClipboardData**, and **SetClipboardData**, provide another way to move information into or out of the session.

Taken together, these findings are consistent with the core HVNC workflow. The browser references identify the applications involved, and the capture, input, window-management, and clipboard functions provide the mechanisms needed to observe and control activity inside the hidden desktop.

Defending Against MedusaHVNC

For the operator to use the hidden session, MedusaHVNC has to send it off the infected endpoint. That makes outbound traffic a practical place to detect and interrupt the attack, regardless of the loader or evasion technique used earlier in the chain.

An unapproved connection to an attacker's server is the same event whether it comes from a browser or a native client. Blocking traffic to known C2 infrastructure and alerting on unexpected outbound connections works regardless of how the session was hidden. Blocking the C2 address and the hashes in the IOC table is a concrete starting point.

This is also the layer BlackFog's [anti data exfiltration \(ADX\)](#) technology is built for. It watches outbound traffic on the device and blocks transfers to command-and-control servers in real time, regardless of which process opens the connection. If the hijacked session doesn't make it out, the operator has nothing to receive.

Indicators of Compromise

Type	Value
C2 IPv4:port (verified static configuration)	51.89.204.28:4444
JScript launcher SHA-256	9eb4effa6b02b43d66a23bc84f3d19f14011b2e5815cd5973934c9517971693
JScript launcher MD5	a3dcfdcb0bb3ad52ae6f2ef2a3e10118
Autolt interpreter SHA-256	5d69a932a077fee044b193c28e84564143f5c7e51079ab48e88fef74ab0b77b7
Final payload SHA-256	e261888f0807b9b372b54aa45487cb5786a939fa3e87094ff270b6b8db5fdb6
Temporary staging directory	%TEMP%\Nx2981Okkr2\
Startup artefact	AFLlvOscPj.bat
Encrypted payload file	zorsxklxfehdoals
Autolt configuration file	wwwxrhlfxquucjpewcitbkip.cfg
Injection target	C:\Windows\System32\charmap.exe



[MedusaHVNC: A Hidden Desktop That Steals Live Windows Sessions](#)

MedusaHVNC is a newly uncovered hidden desktop malware family that gives operators access to live, logged-in browser sessions on a victim's Windows device.



[American Alarm Moves From Reactive Security To Proactive Data Protection With](#)

[BlackFog](#)

Discover how American Alarm strengthened cybersecurity with BlackFog, preventing data exfiltration, improving AI governance, and reducing cyber risk.



[What Are Recommended Best Practices For Implementing AI Assurance In Organizations?](#)

Discover best practices for implementing AI assurance to build secure, transparent and trustworthy AI usage across your organization.



[What Are The Latest Trends In AI Assurance Standards And Regulations?](#)

Explore the latest AI assurance standards, regulations and governance trends shaping transparent, compliant and trustworthy AI.



[How Can AI Be Used to Improve Cybersecurity Measures?](#)

Learn how AI improves cybersecurity through faster threat detection, incident response, AI monitoring, behavioral analysis and security automation.



[What Should I Look For In An AI Security App To Ensure My Device Is Protected?](#)

Learn what to look for in an AI security app, from real-time visibility and data protection to defense against AI-powered threats, to keep your device secure.