ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 08 / 30 · MALWARE BUILD CHAINS

Exposed Build Factories
A Rust Implant Compiler and an XWorm Workbench

How a 313-crate Rust workspace, XWorm 5.6 source and per-user COM hijack scripts expose modern Windows tradecraft

Prepared November 2026
Full Report · Public edition · Version 1.1

1,305
Exposed build-factory files on two servers: 1,096 on the Rust host and 209 on the XWorm host

313
Unique Rust crates in the implant's Cargo workspace, across five capability layers

25
Direct build inputs to the Rust implant.exe, listed in its exposed manifest

121
Files in the /rdp/ folder: cookie grafters, HVNC builds and the COM hijack script

What Is in This Report. 11 chapters and two appendices.

How a 313-crate Rust workspace, XWorm 5.6 source and per-user COM hijack scripts expose modern Windows tradecraft

01	Executive Summary	04
02	Two Servers, Two Kinds of Tooling	07
03	The Rust Build Factory	10
04	The XWorm 5.6 Workbench	14
05	Browser Session Theft and Per-User COM Hijacking	18
06	Who Runs These Servers	22
07	What Defenders Should Do	24
08	Detection Rules	26
09	Indicators	30
10	Scope and Limits	34
11	Sources	36
A	Indicators of compromise (21)	38
B	Detection rules (4 files)	40

IN SHORT

In files indexed 13 to 16 November 2026, Odin by Brandefense found two exposed attacker build servers. A Canadian host leaked a 1,096-file Rust implant build tree. A Russian host, 193.233.220[.]165, exposed 209 files: XWorm 5.6 source, keyboard hooks, browser cookie grafters and a per-user COM hijack script. Only 6 of 920 recent commodity RAT candidates were confirmed live. Alert on per-user CLSID writes into user-writable folders, and block RAT addresses only after live confirmation.

01

CHAPTER 01

Executive Summary

IN SHORT

In files indexed 13 to 16 November 2026, Odin by Brandefense found two exposed attacker build servers. A Canadian host leaked a 1,096-file Rust implant build tree. A Russian host, 193.233.220[.]65, exposed 209 files: XWorm 5.6 source, keyboard hooks, browser cookie grafters and a per-user COM hijack script. Only 6 of 920 recent commodity RAT candidates were confirmed live. Alert on per-user CLSID writes into user-writable folders, and block RAT addresses only after live confirmation.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 05

01 Executive Summary

Two exposed servers show two different ways to build Windows intrusion tooling. One compiles a custom Rust implant on Linux for Windows. The other stages leaked .NET RAT source next to PowerShell scripts that live off the land. This report explains what each server held, how the tooling is designed to work, and which signals defenders can hunt. It also compares these high-signal servers with large, noisy lists of commodity RAT addresses.

1. **A leaked Rust build tree maps a custom implant and 313 crates.**
   A Canadian host exposed 1,096 files from a Rust cross-compile tree under a custom project folder (path withheld). Its build manifests list 25 direct inputs to `implant.exe`. Their names point to a Beacon Object File loader, kernel-state routines, UAC bypass and persistence modules.

2. **A Russian workbench pairs XWorm 5.6 source with browser and COM hijack scripts.**
   The host 193.233.220[.]65 exposed 323,901 bytes of unobfuscated XWorm 5.6 Visual Basic .NET source. A 121-file `/rdp/` folder holds scripts written to graft stolen cookies into Chrome and to register a per-user COM server. None of this needs administrator rights.

3. **Only 6 of 920 recent RAT candidates were confirmed live C2 servers.**
   Historical records for six commodity RAT families total 19,391. Of 920 recently active candidates, 6 were confirmed live between 13 and 16 November 2026. Passive RAT lists age fast on recycled VPS addresses.

Three Decisions for Leaders

1. **Stop per-user COM registrations that point into user-writable folders.** Block or alert when a non-installer process writes a per-user `CLSID\...\InProcServer32` value pointing into `C:\Users\Public\` or `Temp`. This persistence needs no UAC prompt.
2. **Watch browser remote debugging and PowerShell hook compilation.** Alert when a shell or scheduled task starts Chrome, Edge or Brave with a remote debugging port and a custom profile folder. Also alert when PowerShell compiles a `user32` keyboard hook. Exclude known test automation.
3. **Block commodity RAT addresses only after live confirmation.** Most RAT-tagged addresses are stale. Confirm a live controller before blocking, and invest in file and compiler-path signatures that survive address changes.

The Study in Numbers

Number	What it is
1,305	Exposed build-factory files on two servers: 1,096 on the Rust host and 209 on the XWorm host
313	Unique Rust crates in the implant's Cargo workspace, across five capability layers
25	Direct build inputs to the Rust `implant.exe`, listed in its exposed manifest
323,901 bytes	Unobfuscated XWorm 5.6 Visual Basic .NET source in four files
121	Files in the `/rdp/` folder: cookie grafters, HVNC builds and the COM hijack script
6 of 920	Recently active RAT candidates confirmed as live C2 servers (0.652%)
19,391	Historical commodity RAT records across six families (6 confirmed live, 0.031%)

How to use this report.

Security leaders can read Chapter 1, Chapter 6 and Chapter 7. Endpoint and browser engineers should read Chapters 4, 5 and 7. Detection engineers will find the Rust workspace detail in Chapter 3 and the rule logic in Chapter 8.

02

CHAPTER 02

Two Servers, Two Kinds of Tooling

IN SHORT

Exposed build servers leak compiler paths, source code and registry keys that outlast any IP address. Lists of suspected commodity RAT addresses, by contrast, go stale quickly.

INSIDE THIS CHAPTER

What each server held 07

Commodity RAT records 08

Read the ratios with care 08

02 Two Servers, Two Kinds of Tooling

The two servers in this study are unrelated. No shared address, certificate or path links them. They are useful together because they show two ends of the Windows tooling spectrum.

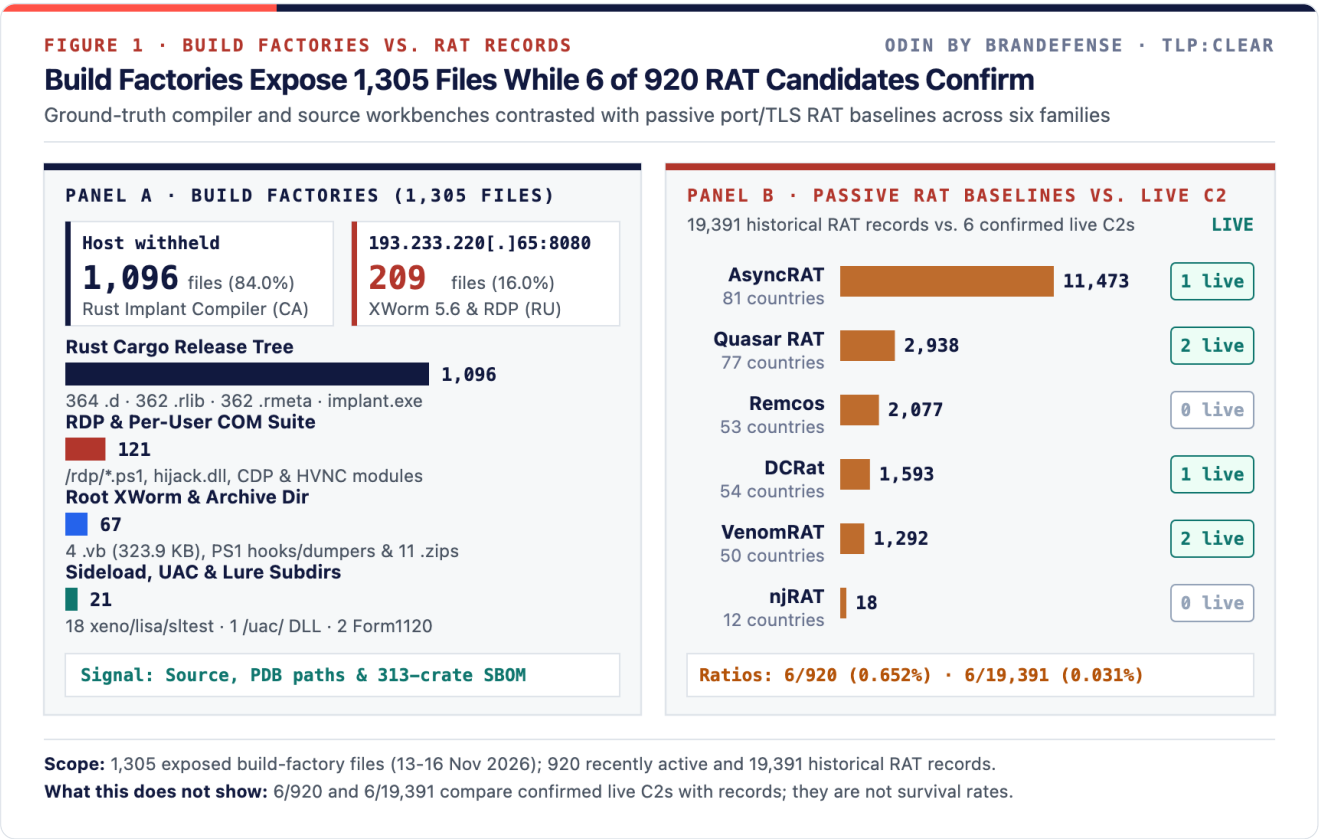


Figure 1: Build factories expose 1,305 files while 6 of 920 RAT candidates confirm

What each server held

Server	Files	Main contents	What defenders get from it
Rust implant factory (Canada, address withheld, indexed 13 Nov 2026)	1,096 (84.0%)	364 `.d`, 362 `.rlib`, 362 `.rmeta`, 2 `.exe`, 2 `.pdb`, 1 `.lib`	Linux build paths under a custom project folder (path withheld), 25 implant inputs, and a 313-crate workspace list
XWorm 5.6 and RDP workbench (Russia, 193.233.220[.]65:8080, indexed 16 Nov 2026)	209 (16.0%)	123 `.ps1`, 22 `.dll`, 15 `.bin`, 15 `.bat`, 13 `.zip`, 4 `.vb`	VB.NET source, plain-HTTP staging on port 443, Chrome cookie grafting and per-user COM hijacks

The Rust host's address is withheld because an authorized red-team owner remains plausible (see Chapter 6).

Commodity RAT records

The commodity RAT side tells a different story. Many feeds flag RAT controllers from a port, a banner or a self-signed certificate subject such as `CN=default`, `CN=AsyncRAT Server` or `CN=Quasar Server CA`. Those signals persist when a VPS address passes to a new tenant.

Family	Historical records	Share	Countries	Confirmed live (13 to 16 Nov)	Confirmed C2
AsyncRAT	11,473	59.2%	81	1	102.220.163[.]109:443, `CN=default`
Quasar	2,938	15.2%	77	2	141.98.157[.]101:2288; one more withheld (consumer broadband)
Remcos	2,077	10.7%	53	0	None
DCRat	1,593	8.2%	54	1	148.113.214[.]176:443, `CN=default`
VenomRAT	1,292	6.7%	50	2	5.161.43[.]175:443 and 154.12.90[.]49:443, `CN=default`
njRAT	18	0.1%	12	0	None
Total	**19,391**			**6**	

VenomRAT and njRAT together hold 1,310 records. All six confirmed controllers presented `CN=default`.

Read the ratios with care

Six of 920 recently active candidates is 0.652%. Six of all 19,391 records is 0.031%. These ratios compare confirmed live servers with a list. They are not survival rates, because not every record was checked in the window.

The other 19,385 records are unverified, not proven benign. One candidate host on Contabo alone exposes 14 candidate ports. Six unconfirmed Quasar candidates show the same `CN=default` subject as the confirmed servers. A certificate subject alone cannot separate a live controller from a recycled address.

An exposed build directory works differently. It yields compiler paths, assembly strings, registry keys and staging URIs. Those signals stay useful after the operator moves to a new address.

03

CHAPTER 03

The Rust Build Factory

IN SHORT

A Canadian host exposed the release folder of a Rust toolchain that cross-compiles a Windows implant on Linux. Its build manifests reveal 25 implant modules and a 313-crate workspace that includes a C2 server.

INSIDE THIS CHAPTER

What the release folder held 10

Why dependency files are so revealing 11

The 25 implant inputs 11

The 313-crate workspace 11

Key manifests 12

03 The Rust Build Factory

On 13 November 2026, a Canadian host served an open HTTP directory listing. It exposed the output folder of a Rust cross-compile toolchain for 64-bit Windows: a `target/x86_64-pc-windows-msvc/release/` folder (project path withheld). All 40 stored manifests point to this single build root.



Figure 2: Exposed Cargo manifests reconstruct 313 Rust crates and 25 implant inputs

What the release folder held

File type	Files	Manifests stored	Example paths and role
`.d` (Cargo dependency info)	364	40	`/implant.d`, `/deps/implant.d`, and 362 hashed library manifests in `/deps/`
`.rlib` (Rust static library)	362	0	the custom library (name withheld), `libclipboard_win-dc113935604f0ab2.rlib` (163,452 bytes)
`.rmeta` (metadata stub)	362	0	the custom library (name withheld), `libclipboard_win-dc113935604f0ab2.rmeta` (122,805 bytes)
No extension (Cargo lock files)	3	0	`/.cargo-artifact-lock`, `/.cargo-build-lock`, `/.cargo-lock`
`.exe` (Windows PE64)	2	0	`/implant.exe`, `/deps/implant.exe`
`.pdb` (debug symbols)	2	0	`/implant.pdb`, `/deps/implant.pdb`
`.lib` (import library)	1	0	`/deps/implant.lib`

Why dependency files are so revealing

When Cargo builds a Rust project, the compiler writes `.d` files that track which source files each output depends on. Cargo uses them for incremental rebuilds. Each `.d` file lists exact Linux paths of the operator's modules and of the third-party crates it pulled in.

The project's `src/` folder was not exposed, so no `.rs` source bodies are available. The manifests still name every module. One counting detail matters: library files carry a `lib` prefix (`liblibc-...rlib`) while `.d` files do not (`libc-...d`). Removing the prefix only from `.rlib` and `.rmeta` names avoids double-counting crates such as `libc`, `libsqlite3_sys` and `libz_rs_sys`.

The 25 implant inputs

The manifest `/implant.d` is 928 bytes. It lists all 25 direct inputs to `implant.exe`: 22 module paths in the project's `src/` folder, plus `build.rs`, `config.toml` and the generated `out/config.rs`. The module names suggest four subsystems. These roles are assessed from file names only.

Assessed subsystem	Module paths	What the names suggest
COFF and Beacon Object File loader	`bof.rs` plus five custom loader modules and a C shim (names withheld)	A Rust and C loader modeled on Cobalt Strike Beacon Object File and `BeaconPrintf` conventions
Kernel-state modules	`src/kernel_state.rs`, `src/routes/kernel_state.rs` and two custom modules (names withheld)	Kernel-state inspection or driver-assisted tasking
Host awareness, UAC bypass, persistence	`src/awareness.rs`, `src/uac.rs`, `src/persistence.rs`, `src/metacmd.rs`, `src/extensions.rs`, `src/symbols.rs`, `src/types.rs`, `src/lib.rs`, `src/implant/bin/implant.rs`	Host discovery, UAC bypass, persistence and symbol resolution (with crates `pdb2`, `pdb_addr2line`, `symsrv`)
Build config and C2 routes	`build.rs`, `config.toml`, `out/config.rs`, `src/routes/checkin.rs`, `src/routes/poll.rs`, `src/routes/download.rs`	Build-time configuration plus shared check-in, polling and download routes

The build script `build.rs` reads `config.toml` and generates `out/config.rs`.

The 313-crate workspace

A Cargo workspace caches compiled dependencies for every member built for the same target. So the `/deps/` folder holds more than the implant. It also holds the C2 server and listener crates (`actix-web`, `actix-server`, `actix-files`, `sqlx`, `libsqlite3_sys`), operator command-line crates (`clap`, `rustyline`, `cli_table`) and a Windows clipboard crate (`clipboard-win`).

Deduplicating the 362 hashed library manifests gives 313 unique crates in five layers.

Layer	Crates	Examples and versions	Assessed role
Core runtime and build support	228 (72.8%)	`sqlx`, `libsqlite3_sys`, `libc`, `pdb2`, `goblin`, `regex`, `clap`, `zstd`, `zeroize`	A custom library (name withheld), PE/COFF/PDB parsing, C2 SQLite state and memory wiping
Async C2 HTTP and network	39 (12.5%)	`actix-web` 4.12.0, `actix-server` 2.6.0, `actix-files` 0.6.6, `reqwest`, `hyper`, `tokio`, `tokio-tungstenite`, `cookie`	C2 listener and staging stack plus HTTP/1.1, HTTP/2, cookie jar and WebSocket clients
TLS and cryptography	20 (6.4%)	`rustls`, `ring`, `native-tls`, `schannel`, `chacha20`, `rand_chacha`, `sha2`, `sha1`, `digest`, `subtle`, `base64`	Both `rustls` and Windows `schannel` TLS stacks, ChaCha20 and RustCrypto digests
Windows OS and clipboard	19 (6.1%)	`clipboard-win` 4.5.0, `ntapi`, `winapi`, `windows`, `winreg`, `sysinfo`	Native NT bindings, registry access, host profiling and clipboard access
Serialization and task parsing	7 (2.2%)	`serde`, `serde_json`, `prost`, `nom`, `bytes`, `serde_plain`, `serde_spanned`	JSON and Protocol Buffers messages, binary packet and COFF parsing

Of the 313 crates, 312 are public `crates[.]io` packages. Only one, a custom library whose name is withheld, is not public. That is why the crate list alone is a poor detection signal, and why the custom module names matter.

Key manifests

Path on the Rust host	Size (lines)	What it shows
`/implant.d`	928 B (1)	The 25 direct implant inputs
`/deps/clipboard_win-dc113935604f0ab2.d`	1,757 B (10)	`clipboard-win-4.5.0` compiled into the workspace
`/deps/actix_web-17e7c7e05985a20a.d`	34,788 B (93)	`actix-web-4.12.0` HTTP C2 routing
`/deps/actix_server-c549e8f6e7ceaa5c.d`	5,155 B (19)	`actix-server-2.6.0` multi-worker listener
`/deps/actix_files-75b175c3000ece1a.d`	3,941 B (16)	`actix-files-0.6.6` chunked payload delivery

Their hashes are withheld, because a red-team or tool-developer owner remains plausible for this host.

04

CHAPTER 04

The XWorm 5.6 Workbench

IN SHORT

A Russian staging server exposed 209 files, including full XWorm 5.6 source that poses as a Windows Update component. Small PowerShell scripts add keystroke capture, screenshots, command polling and file theft without a custom executable.

INSIDE THIS CHAPTER

XWorm 5.6 source, ready to compile 15

Collection scripts 15

The operator did not trust their own panel 16

04 The XWorm 5.6 Workbench

On 16 November 2026, the Russian host 193.233.220[.]165 served 209 files on port 8080. Scripts on the host also reference ports 443, 8090 and 8091 on the same address. The files sit in three groups: 121 in `/rdp/`, 67 in the root folder (11 of the 13 `.zip` archives), and 21 in subfolders such as `xeno_*`, `lisa`, `sltest`, `/uac/` and `/docs/Form1120.*`.

Of the 209 files, 45 source and script files were examined in detail. This report describes them by structure only. No working script body is reproduced.

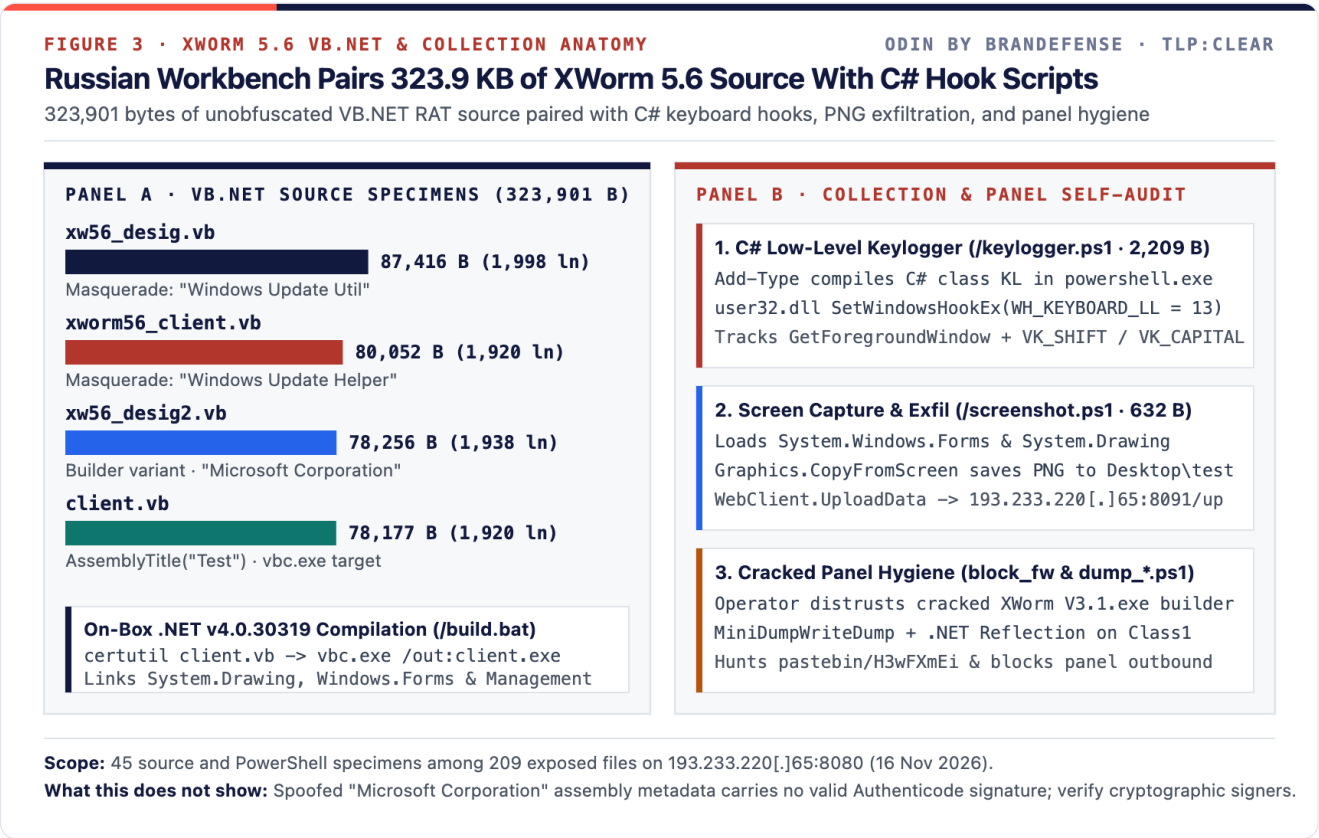


Figure 3: Russian workbench pairs 323.9 KB of XWorm 5.6 source with C# hook scripts

XWorm 5.6 source, ready to compile

XWorm is a commodity .NET remote access trojan whose source has leaked. Instead of only using prebuilt stubs that antivirus already knows, this operator staged four complete Visual Basic .NET source files: 323,901 bytes and 7,776 lines in total.

File	Size (lines)	Role	Masquerade strings
`/xworm56_client.vb`	80,052 B (1,920)	Full XWorm 5.6 client source	`AssemblyTitle("Windows Update Helper")`, `"Windows Update Helper Service"`, `"Microsoft Corporation"`
`/xw56_desig.vb`	87,416 B (1,998)	Builder or designer variant	`"Windows Update Util"`, `"Windows Update Util Service"`, `"Microsoft Corporation"`
`/xw56_desig2.vb`	78,256 B (1,938)	Second builder variant	`"Microsoft Corporation"`
`/client.vb`	78,177 B (1,920)	Client staged for on-box compilation	`AssemblyTitle("Test")`

The client imports .NET namespaces for cryptography, WMI management, sockets, image capture and the registry. Its file version is written as a string concatenation (`"1" + "." + "0" ...`), which the compiler folds to `1.0.0.0`.

A 569-byte `/build.bat` is written to fetch `client.vb` from the workbench with `certutil` and compile it with the built-in .NET Framework `vbc.exe` as a windowed executable (`/target:winexe /out:client.exe`). Compiling on the target avoids shipping a known binary hash.

The "Microsoft Corporation" strings are only metadata. Output from `vbc.exe` carries no valid Microsoft Authenticode signature. Allowlists must check the signature chain, not the company name.

Collection scripts

Three small script families collect data without a custom implant. They use the workbench's secondary ports.

Script	Size	Structure	ATT&CK
`/keylogger.ps1`	2,209 B, 47 lines	Compiles an inline C# class at run time with `Add-Type`. The class calls `SetWindowsHookEx` with a low-level keyboard hook (`WH_KEYBOARD_LL`) and records the foreground window title	T1056.001
`/screenshot.ps1`	632 B, 13 lines	Captures the primary screen with `Graphics.CopyFromScreen`, saves a PNG and uploads it with `WebClient.UploadData` to port 8091, path `/up`	T1113
`/oldzip.bat`, `/runner.ps1`	377 B, 1,846 B	Package SMB share folders with `tar.exe` and post them to port 8091 (share names withheld)	
`/np.bat`	398 B, 13 lines	Polls port 8090, path `/cmd?box=`, with `curl.exe`, runs any returned command and posts the output back to `/r2?box=`	
`/capture8002.ps1`	1,153 B	Opens a local listener on port 8002 to capture the handshake of an `xw56.exe` build	

For hunting, the stable strings are the API names (`SetWindowsHookEx`, `WH_KEYBOARD_LL`, `CopyFromScreen`), the paths `/up`, `/cmd?box=` and `/r2?box=`, and a `curl` user agent on a polling loop.

The operator did not trust their own panel

Four scripts show the operator auditing a cracked XWorm V3.1 builder for a hidden backdoor. The backdoor pointed at a Pastebin paste, `H3wFXmEi`.

Script	What it is written to do
`block_fw.ps1`	Resolves `pastebin[.]com` and adds a Windows Firewall rule named `Block XWorm Panel Outbound` against `XWorm V3.1.exe`
`dump_fresh.ps1`	Writes a memory dump of the panel with `MiniDumpWriteDump` and searches it for `pastebin` and `H3wFXmEi`
`reflect_decrypt.ps1`	Loads the panel through .NET reflection to call its obfuscated `Class1` string decryptor
`deploy_clean.ps1`	Backs up the panel binary and swaps in `XWormV3.1_clean.exe`

Cracked RAT builders often carry their own backdoors. This operator checked for one before use, which fits a criminal workbench rather than a research lab.

CHAPTER 05

Browser Session Theft and Per-User COM Hijacking

IN SHORT

More than half of the workbench sits in `/rdp/`. These scripts are written to stage a DLL over plain HTTP on port 443, register it as a per-user COM server, graft stolen cookies into Chrome and run real browser profiles on a hidden desktop.

INSIDE THIS CHAPTER

The per-user COM hijack script 18

Cookie grafting over the Chrome DevTools Protocol 19

Real-profile HVNC against App-Bound Encryption 19

Key workbench files and their hashes 20

05 Browser Session Theft and Per-User COM Hijacking

Chrome added two defenses that matter here. Since version 127, App-Bound Encryption (ABE) ties cookie decryption keys to an elevated system service. Since version 136, Chrome ignores `--remote-debugging-port` on the default profile folder. The `/rdp/` scripts are built around both. We did not see these scripts run on any target system.

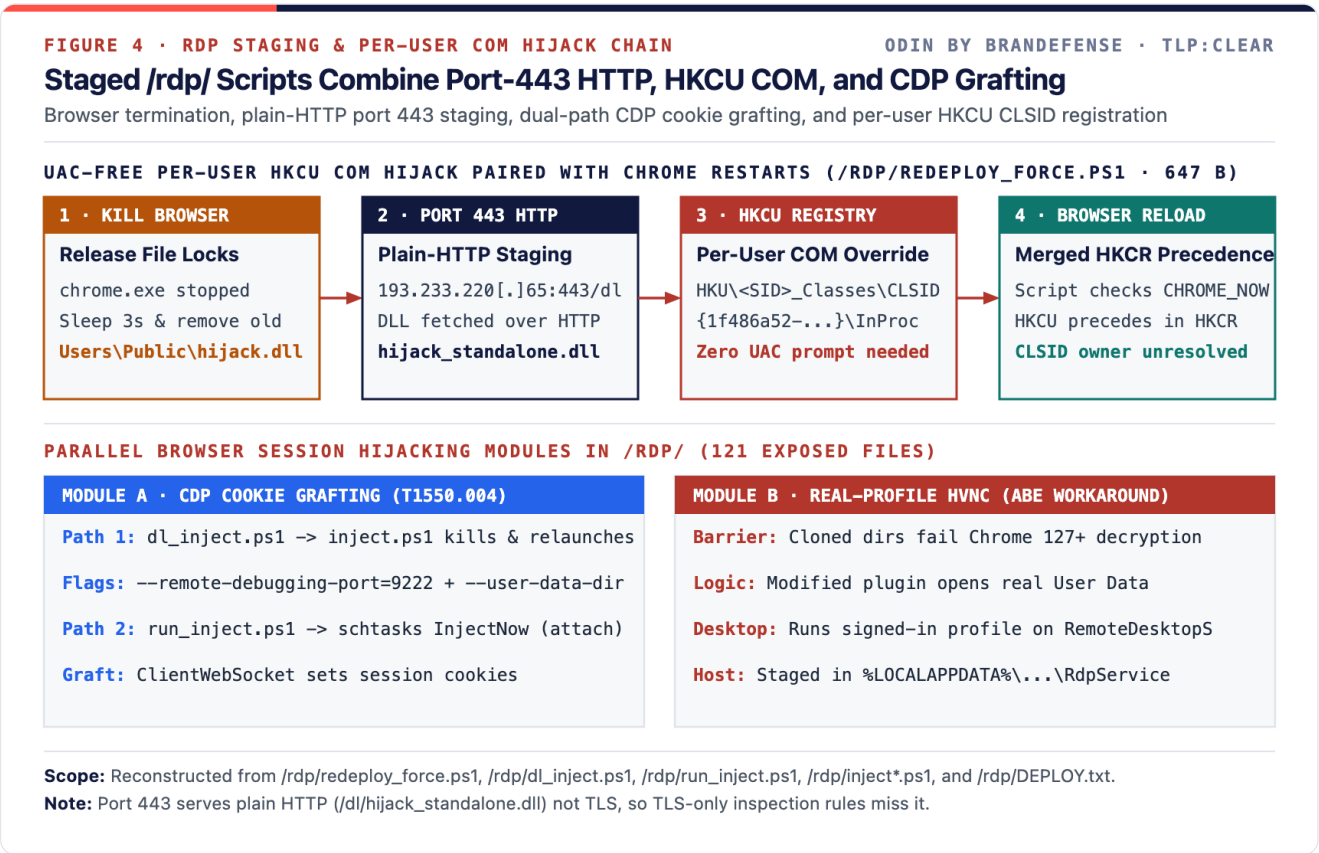


Figure 4: Staged /rdp/ scripts combine port-443 HTTP, HKCU COM and CDP grafting

The per-user COM hijack script

`/rdp/redeploy_force.ps1` is 647 bytes and 10 lines. It is structured as a four-step sequence that needs no administrator rights. The command text is withheld.

Step	What the script is written to do	Why it matters
1. Stop the browser	Stop `chrome.exe`, pause, and delete an older `hijack.dll` from the public user folder	Releases file locks on a loaded DLL
2. Stage over plain HTTP on port 443	Download `hijack_standalone.dll` from `193.233.220[.]65:443/dl` over unencrypted HTTP and save it as `hijack.dll` in the public folder	Egress rules that exempt port 443 from inspection miss the protocol mismatch
3. Write a per-user COM entry	Set `HKCU\Software\Classes\CLSID\{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}\InProcServer32` to the staged DLL	Per-user entries need no UAC prompt (T1546.015)
4. Check for a browser restart	Count Chrome processes (`CHROME_NOW`)	Suggests the operator expects Chrome to load the class

Windows merges per-user and machine COM registrations into one view, `HKEY_CLASSES_ROOT`. The per-user entry wins. That is why this persistence needs no elevation.

The owner of CLSID `{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}` is not resolved in public Windows symbol tables. That Chrome loads it is an inference from the script's browser stop and restart check. Endpoint image-load records were outside the scope of this study.

> **Where to look in logs.**
>
> `HKCU\Software\Classes` is stored in the user's `UsrClass.dat` hive. Sysmon logs the write as `HKU\<SID>_Classes\CLSID\...`, and Defender XDR as `HKEY_USERS\<SID>_Classes\CLSID\...`. Neither path contains `\Software\`. Rules must match the `_Classes\CLSID\` substring, or they will miss it.

Cookie grafting over the Chrome DevTools Protocol

The scripts named `inject*.ps1` do not inject into process memory (T1055). They graft session cookies that were stolen earlier (T1539) into a browser through the Chrome DevTools Protocol (CDP) WebSocket (T1550.004). CDP is the debugging interface Chrome exposes on a local port.

Path	Scripts	Structure
Path 1: relaunch Chrome	`/rdp/dl_inject.ps1` (244 B) then `/rdp/inject.ps1` (20,209 B)	The first script fetches `inject.ps1` from `:443/dl/` into `C:\Users\Public\`. The second stops Chrome and restarts it with `--remote-debugging-port=9222` and a sibling profile folder named `Chrome Data` beside `User Data`
Path 2: attach without relaunch	`/rdp/run_inject.ps1` (828 B), `/rdp/setup_inject.ps1` (654 B), `dl_inject_only.ps1` (240 B), then `/rdp/inject_only.ps1` (20,000 B) or `/rdp/inject_standalone.ps1` (3,489 B)	Fetches `inject_only.ps1` as `C:\Users\Public\inj.ps1`, writes `C:\Users\Public\run_inj.bat`, and starts a one-shot scheduled task named `InjectNow`. The grafter polls `127.0.0.1:9222/json` and pushes cookie arrays over `ClientWebSocket`

The sibling profile folder is deliberate. Chrome 136 and later ignores the debugging port on the default `User Data` folder, so the script points Chrome at a non-default folder under the vendor directory.

Real-profile HVNC against App-Bound Encryption

The operator's notes in `/rdp/DEPLOY.txt` (1,682 bytes, titled `HVNC REAL-PROFILE BUILD - ALL BROWSERS`) say that cloned profiles cannot decrypt Chrome, Edge or Brave 127+ cookies. HVNC (hidden VNC) gives an attacker a second, invisible desktop on the victim's machine.

The notes describe a modified HVNC plugin that skips profile cloning. It opens the user's real `User Data` profile in place on a hidden desktop named `RemoteDesktopS`. Patch details are withheld.

File	Size	Role
`/rdp/hijack.dll`	42,496 B	Same size as the modified HVNC plugin described in `DEPLOY.txt`
`/hijack.dll`	233,836 B	A different DLL in the root folder
`/hvnc_test.dll` (`HVNC.dll` in the notes)	59,904 B	Size matches the HVNC build listed in `DEPLOY.txt`
`/rdp/hvnc_new.dll` to `HVNC_final.dll`	60,416 to 83,456 B	Four HVNC builds: `hvnc_new.dll`, `HVNC_patched.dll` (75,264 B), `HVNC_mark3.dll` (76,288 B), `HVNC_final.dll` (83,456 B)
`/rdp/shell_rp.bin` (`shell.bin` in the notes)	10,348,140 B	Bundled payload, written to `%LOCALAPPDATA%\Microsoft\RdpService\shell.bin`
`/rdp/shell.bin`, `/lisa/shell.bin`	96,255 B and 85,100 B	Smaller stage payloads; five `/xeno_*/shell.bin` copies also measure 85,100 B
`%LOCALAPPDATA%\Microsoft\RdpService\mstsc.exe`	Not listed	A masqueraded `mstsc.exe` paired with `version.dll` sideloaders (`/rdp/version.dll` 16,384 B, `/sltest/version.dll` 93,375 B)

A real `mstsc.exe` lives in `System32`. A copy under `AppData\Local` is a strong masquerade signal (T1036.005).

Key workbench files and their hashes

Path	Size (lines)	SHA-256	Function
`/xworm56_client.vb`	80,052 B (1,920)	`d7907bd930b212675b2686f17a702339c7be3703f6b9e1c4c0c99a3b8df81ce2`	XWorm 5.6 client posing as "Windows Update Helper"
`/xw56_desig.vb`	87,416 B (1,998)	`ca163c0a6e102d9bf1058fbe99db94964115bd414514accf7cb2b31111a54bb6`	XWorm 5.6 builder posing as "Windows Update Util"
`/client.vb`	78,177 B (1,920)	`1fd0695487372a62306721a3f4f70ef042bf224bf96adc31ea889c41febc2b34`	RAT client staged for `vbc.exe` compilation
`/keylogger.ps1`	2,209 B (47)	`c7caa7cadec4f43a0e72347a50da4cbf0473fa79f8ed09f7c1d5584e29415ae7`	Inline C# low-level keyboard hook
`/screenshot.ps1`	632 B (13)	`d747bd06488cf50c826327a7b0deb794d4cfa6c7051af9c2bd4809a75f99ba0a`	Screen capture uploaded to `:8091/up`
`/rdp/dl_inject.ps1`	244 B (3)	`1a0d0943bb4382a0e1d55bfd02f4970f572d4f7627691464f36755d354d7034e`	Path 1 downloader for `inject.ps1`
`/rdp/inject.ps1`	20,209 B (44)	`ceb17c70555ff457c2402f1d970626f59c3481377c1277ae6e8831c51cb12ecb`	Chrome relaunch with debugging port, cookie grafting
`/rdp/inject_standalone.ps1`	3,489 B (73)	`5ef9a57c68ce939589e0689b0ee86c0caadd48a2eb0eb587974403ddc7ed14ee`	Cookie grafting over `ClientWebSocket`
`/rdp/redeploy_force.ps1`	647 B (10)	`7f231f37f35f2283b492b28d3987be19cc43a0a22956a37e23ea46c0e1832565`	Browser stop, port-443 DLL staging, per-user CLSID write

CHAPTER 06

Who Runs These Servers

IN SHORT

The Russian workbench fits an intrusion and crimeware operator. The Rust factory is offensive, but an authorized red-team or tool developer cannot be ruled out.

INSIDE THIS CHAPTER

Avoiding false positives 22

06 Who Runs These Servers

We saw only exposed files, not victim systems. So we weighed four other explanations for each server.

Hypothesis	For	Against	Assessment
1. Authorized red team or adversary simulation	The Rust host builds a modular COFF loader with structured check-in and polling routes, and holds no victim data	A public, unauthenticated `target/` folder and `implant.pdb` break red-team operational security. The Russian host stages cracked XWorm builders and tax-form (Form 1120) `.lnk` lures	Plausible for the Rust host; unlikely for the Russian host
2. Research lab or analysis sandbox	The Russian host holds diagnostic scripts that inspect XWorm memory dumps	The same host stages captured session cookies, SMB share theft scripts, tax lures and port-443 payload delivery	Unlikely; the panel checks fit an operator who distrusts a cracked tool
3. Honeypot or canary	Both directory listings were open to anyone	The Rust host holds a complete 1,096-file Cargo cache with consistent hashes. The Russian host held real share names and session cookies (withheld)	Refuted for the Russian host; low likelihood for the Rust host
4. Open-source mirror or training set	312 of 313 crates are public, and XWorm source has leaked	`/implant.d` compiles non-public custom modules (names withheld). The Russian host adds custom CDP grafters and target-specific share lists	Refuted; both hosts show active building and customization

Our assessment. The evidence fits 193.233.220[.]65 as an intrusion and crimeware staging workbench prepared for Windows post-exploitation. Target-side outcomes are unverified. The Canadian host fits an offensive Rust implant build factory. A red-team or tool-developer owner remains plausible, because no victim data sits on it. Nothing links the two servers, so we treat them as separate toolchains.

Avoiding false positives

Pitfall	Why it misleads	What to do instead
Treating historical RAT records as live	`CN=default` certificates and port patterns persist on recycled VPS addresses (Contabo, M247, OVH). 19,385 records have no live confirmation	Hold them from automatic blocking. Confirmed live controllers take priority
Alerting on common Rust crates	`tokio`, `actix-web`, `reqwest`, `rustls` and `serde` power many legitimate services	Require a custom marker from your own case data plus a second signal
Alerting on generic file names	`client.vb`, `screenshot.ps1`, `keylogger.ps1` and `implant.exe` appear in developer repositories	Require a high-specificity marker (`xworm56_client.vb`, `xw56_desig*.vb`, `redeploy_force.ps1`, `inject_only.ps1`, `inject_standalone.ps1`, the CLSID, or a known hash) plus a second signal
Alerting on every remote debugging launch	Selenium `chromedriver.exe`, `msedgedriver.exe`, Puppeteer, Playwright and Cypress use it, often with `--headless`, `--enable-automation`, `--remote-debugging-port=0` or `%TEMP%\scoped_dir*` profiles	Suppress approved test automation by exact host, not by parent name or flags. Alert on shell or task parents, `--remote-allow-origins=*`, or vendor-folder profiles
Trusting "Microsoft Corporation" in version info	The XWorm builds copy the string but carry no valid signature	Validate Authenticode chains, not `CompanyName`

A single hash match is also only a file match. It does not show that a host runs a workbench.

07

CHAPTER 07

What Defenders Should Do

IN SHORT

This chapter lists the actions defenders should take, in priority order.

07 What Defenders Should Do

Within 24 hours

- [ ] Search firewall, proxy, DNS and NetFlow logs for 193.233.220[.]65 on ports 8080, 443, 8090 and 8091, and for the five published RAT controllers in Chapter 9.
- [ ] Hunt registry events for writes under `_Classes\CLSID\` with `\InProcServer32` that point into `C:\Users\Public\`, `%TEMP%` or `%APPDATA%`. Check CLSID `{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}` first.
- [ ] Inspect plain HTTP on TCP 443 instead of exempting port 443 from non-TLS inspection.

Within 72 hours

- [ ] Use WDAC or AppLocker to block unsigned DLL loads from `C:\Users\Public\` and `mstsc.exe` launches from `%LOCALAPPDATA%` (T1036.005).
- [ ] Keep Chrome on version 136 or later, and set the `RemoteDebuggingAllowed` policy to false on non-developer endpoints.
- [ ] Confirm App-Bound Encryption on Chrome 127+, verify the vendor rollout for Edge and Brave, and watch for hidden desktop creation such as `RemoteDesktopS`.
- [ ] Enable PowerShell Script Block Logging (Event ID 4104) and AMSI content monitoring for `Add-Type` with `SetWindowsHookEx` or `CopyFromScreen`.

Within 7 to 30 days

- [ ] Alert on `vbc.exe /target:winexe` started by `cmd.exe`, PowerShell or script hosts outside `MSBuild.exe` and `devenv.exe`.
- [ ] Add the YARA rules for XWorm 5.6 source and the hijack and keylogger scripts to file and artifact scanning.
- [ ] Change commodity RAT feed handling: block only after live C2 confirmation, endpoint process evidence and an ASN reassignment check.

Control	Owner
Per-user CLSID `InProcServer32` writes into user-writable paths	Endpoint security / EDR operations
WDAC or AppLocker rules for `C:\Users\Public\` DLLs and AppData `mstsc.exe`	Workstation engineering
App-Bound Encryption and hidden desktop monitoring	Enterprise browser / EDR operations
Chrome 136+ and `RemoteDebuggingAllowed = false`	Enterprise browser / IAM
Script Block Logging and AMSI for hook compilation	Detection engineering / SOC
Plain HTTP on port 443 and `:8090/cmd?box=` polling	Network security
Live confirmation before RAT blocklisting	CTI / threat hunting

CHAPTER 08

Detection Rules

IN SHORT

This chapter describes each published rule and where to deploy it.

INSIDE THIS CHAPTER

How the Suricata signatures work 27

How the KQL query works 27

How the Sigma and YARA rules work 28

How to sort an observation 28

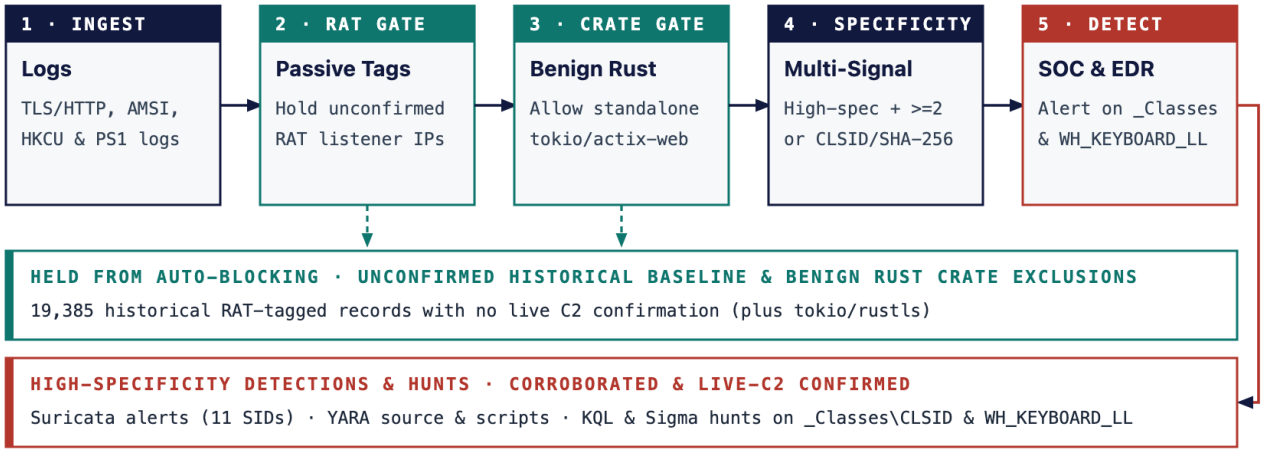
08 Detection Rules

The full rules are in Appendix B. Each one targets a signal that outlasts an address change.

FIGURE 5 · COMPILER ARTIFACT & ENDPOINT DEFENSE ODIN BY BRANDEFENSE · TLP:CLEAR

Layered Rules Hold 19,385 Unconfirmed RAT Tags and Alert on COM Hooks

Layered checks across Suricata 7.x, Sentinel KQL, Sigma and YARA



Scope: Logic of the detection rules in this report. Test them in your own environment first.
What this does not show: Records without live C2 confirmation are unverified, not proven benign; corroborate with endpoint logs.

Figure 5: Layered rules hold 19,385 unconfirmed RAT tags and alert on COM hooks

Rule file	What it catches
`suricata_rust_xworm_and_rdp_staging.rules`	Suricata 7: eleven signatures for DLL and script staging, screenshot upload, command polling and RAT certificate subjects
`sentinel_clsid_hijack_and_ps_hook_hunt.kql`	Microsoft Sentinel and Defender XDR: per-user COM writes, public-folder DLL loads, hook and CDP command lines, and AMSI script content
`sigma_per_user_com_hijack_and_rdp_inject.yml`	Sigma (4 rules): per-user COM hijack, debugging-port browser launches, inline keyboard hook compilation, masqueraded `mstsc.exe`
`yara_xworm56_and_rdp_workbench.yar`	YARA (3 rules): XWorm 5.6 source, the CLSID hijack script, and the inline C# keylogger

How the Suricata signatures work

These rules need HTTP parsing on TCP 443 as well as port 80, so cleartext requests on 443 are seen as HTTP. The certificate rules need visibility of the TLS certificate message.

SID	Looks for	Why
9802601	Outbound request for `hijack.dll` or `hijack_standalone.dll`, optionally under `/dl/` or `/rdp/`	COM hijack DLL staging, including plain HTTP on 443
9802602	Outbound request for `inject.ps1`, `inject_only.ps1` or `inject_standalone.ps1` under `/dl/` or `/rdp/`	Cookie grafter retrieval
9802603	`POST` to exactly `/up` with a body starting with the PNG header	Screenshot upload; the exact path excludes `/upload`
9802605	Request for `xworm56_client.vb`	XWorm 5.6 client source; ignores generic `client.vb`
9802606	Certificate subject `CN=AsyncRAT Server`	Stock AsyncRAT
9802607	Certificate subject exactly `CN=default`	Commodity RAT default; corroborate with endpoint process data, do not block alone
9802608	Certificate subject `CN=Quasar Server CA`	Stock Quasar
9802609	Certificate subject exactly `CN=DcRat`	Stock DCRat
9802610	Request for `xw56_desig.vb` or `xw56_desig2.vb`	XWorm 5.6 builder source
9802611	`GET` to a path starting `/dl/` over cleartext HTTP to port 443	Plain HTTP on the TLS port (policy rule)
9802612	URI with `/cmd?box=` and a `curl/` user agent	The batch command poller

SID 9802604 is retired and not used in this release, so the eleven SIDs run from 9802601 to 9802612 with one gap.

All six confirmed RAT controllers in the window presented `CN=default`. Appliances and development servers also use it. Before acting on SID 9802607, match the destination against EDR network events to find an unsigned .NET or script-launched process. For SID 9802603, exclude internal services that use an exact `/up` route by destination subnet.

How the KQL query works

The query runs over a 14-day lookback and joins four branches.

Branch	Looks for
Registry	`RegistryValueSet` or `RegistryKeyCreated` on `_Classes\CLSID\...\InProcServer32` that names the staged CLSID, points into `C:\Users\Public\` or `\AppData\Local\Temp\`, or names `hijack.dll`
Image loads	Chrome, Edge, Brave, Explorer or `mstsc.exe` loading a DLL from `C:\Users\Public\`, or a DLL named like the hijack and HVNC builds
Process	PowerShell or script hosts with hook, capture or grafter names; `vbc.exe /target:winexe` from shells (not `MSBuild.exe` or `devenv.exe`); `curl`, `certutil` or `bitsadmin` fetching `:443/dl/` or `/cmd?box=`; `mstsc.exe` under AppData; non-QA browser launches with a debugging port
AMSI	PowerShell or AMSI script content containing the CLSID, `SetWindowsHookEx`, `WH_KEYBOARD_LL`, `CopyFromScreen` or `ClientWebSocket`

The image-load and AMSI branches need Defender for Endpoint EDR data and AMSI auditing. Workspaces with only Windows Event IDs 4688 and 4104 should use the Sigma rules instead. Tune by excluding signed installers that register Temp COM servers, developer device groups, and named CI build accounts. Keep `C:\Users\Public\` and the staged CLSID unexempted.

How the Sigma and YARA rules work

Rule	Fires when	ATT&CK
Sigma 1: Per-user COM hijack	Sysmon Event ID 13 writes to `_Classes\CLSID\` or `\Classes\CLSID\` `InProcServer32` naming the staged CLSID or pointing into Public or Temp	T1546.015
Sigma 2: Debugging-port browser launch	Chromium starts with `--remote-debugging-port=` and a shell or task parent, `--remote-allow-origins=*` or a vendor-folder profile; no exclusions are built in, so suppress approved test automation by exact host	T1550.004
Sigma 3: Inline keyboard hook	PowerShell Event ID 4104 shows `Add-Type` with `SetWindowsHookEx` and `WH_KEYBOARD_LL`	T1056.001
Sigma 4: Masqueraded `mstsc.exe`	`mstsc.exe` runs from a user-writable AppData or Public path	T1036.005
YARA `ODIN_XWorm56_VBNet_Windows_Update_Helper_Masquerade`	XWorm 5.6 source or builds with the Windows Update masquerade strings, or `build.bat` with `vbc.exe /target:winexe /out:client.exe`	
YARA `ODIN_RDP_PerUser_CLSID_InProcServer32_Hijacker`	The CLSID with an `InProcServer32` or registry marker and a hijack DLL name	T1546.015
YARA `ODIN_PowerShell_CSharp_WH_KEYBOARD_LL_Keylogger`	`Add-Type` with the low-level keyboard hook and at least two supporting API names	T1056.001

Each YARA rule caps the scanned file size (2 MB to 25 MB depending on the rule) and matches both ASCII and wide strings.

How to sort an observation

The triage logic never escalates on uncorroborated evidence.

What the observation shows	Action
A confirmed live RAT controller, even next to benign crates such as `tokio`	Treat as confirmed C2
A corroborated build factory (custom build markers or XWorm markers plus a second signal)	Critical
Generic file names in a pair, such as `client.vb` with `screenshot.ps1`	Workbench review
A single generic file name	Review
A lone hash match	Artifact review
An unconfirmed RAT candidate on a VPS	Hold
A benign `actix-web` or `rustls` repository	Ignore

09

CHAPTER 09

Indicators

IN SHORT

This chapter lists the published indicators and explains what stays withheld.

INSIDE THIS CHAPTER

Network indicators 30

Hashes that could reach endpoints 31

Staging-server hashes 31

MITRE ATT&CK mapping 32

09 Indicators

Appendix A holds 21 published indicators: 6 addresses and 15 file hashes. The two withheld addresses are described below.

Type	Count	Where to hunt
Network indicators (IPv4)	8 (6 published, 2 withheld)	Firewall, proxy, DNS and NetFlow logs
Script and source hashes that could reach endpoints (SHA-256)	10	EDR file-creation logs, proxy downloads, Script Block logs
Staging-server hashes (SHA-256); build-manifest hashes withheld	5	Infrastructure hunting and file provenance, not endpoint blocking

Withheld: the Rust build host's address and port, and one confirmed Quasar controller on consumer broadband that may be a subscriber host.

Network indicators

Indicator	Port	Role	Hosting	Confirmed
193.233.220[.]165	8080 (also 443, 8090, 8091)	XWorm 5.6, CDP and per-user COM workbench	RU	16 Nov 2026
102.220.163[.]109	443	AsyncRAT C2, `CN=default`	VPS Dedicated LLC (SI)	13 Nov 2026
148.113.214[.]176	443	DCRat C2, `CN=default`	OVH SAS (CA)	16 Nov 2026
141.98.157[.]101	2288	Quasar C2, `CN=default`	EUGAMEHOST LTD (GB)	16 Nov 2026
5.161.43[.]175	443	VenomRAT C2, `CN=default`	Hetzner Online GmbH (US)	16 Nov 2026
154.12.90[.]49	443	VenomRAT C2, `CN=default`	cognetcloud INC (HK)	16 Nov 2026

All six addresses sit on commercial cloud, VPS or ISP networks where addresses are reassigned. Use them for log hunting and time-bound watchlists. Recheck the listener, ASN, certificate subject or staging path before a lasting block.

Hashes that could reach endpoints

These scripts are written to land on targets, for example in `C:\Users\Public\` or `Desktop\test\`. Seven of them also appear in the Chapter 5 table.

File	Size	SHA-256
`/client.vb`	78,177 B	`1fd0695487372a62306721a3f4f70ef042bf224bf96adc31ea889c41febc2b34`
`/build.bat`	569 B	`01393dbf9a23130e2d3899b1f1c80df9d5ed28cff01d60c5472e56a737b28923`
`/keylogger.ps1`	2,209 B	`c7caa7cadec4f43a0e72347a50da4cbf0473fa79f8ed09f7c1d5584e29415ae7`
`/screenshot.ps1`	632 B	`d747bd06488cf50c826327a7b0deb794d4cfa6c7051af9c2bd4809a75f99ba0a`
`/rdp/dl_inject.ps1`	244 B	`1a0d0943bb4382a0e1d55bfd02f4970f572d4f7627691464f36755d354d7034e`
`/rdp/run_inject.ps1`	828 B	`d657df005418f9cbee07391a0f2e05de7e908e87ec35db228062e6bbf2678278`
`/rdp/inject.ps1`	20,209 B	`ceb17c70555ff457c2402f1d970626f59c3481377c1277ae6e8831c51cb12ecb`
`/rdp/inject_only.ps1`	20,000 B	`e29bbc5f3650cf3851ca8d8e25b22865809872216372b4672fff0322081109f4`
`/rdp/inject_standalone.ps1`	3,489 B	`5ef9a57c68ce939589e0689b0ee86c0caadd48a2eb0eb587974403ddc7ed14ee`
`/rdp/redeploy_force.ps1`	647 B	`7f231f37f35f2283b492b28d3987be19cc43a0a22956a37e23ea46c0e1832565`

Staging-server hashes

These five files come from the Russian workbench. The Rust build-manifest hashes are withheld. The two XWorm files also appear in the Chapter 5 table.

File	Size	SHA-256
`/xworm56_client.vb`	80,052 B	`d7907bd930b212675b2686f17a702339c7be3703f6b9e1c4c0c99a3b8df81ce2`
`/xw56_desig.vb`	87,416 B	`ca163c0a6e102d9bf1058fbe99db94964115bd414514accf7cb2b31111a54bb6`
`/xw56_desig2.vb`	78,256 B	`c4c51d14e97504864ff493b202f57cb6bdc43c6a9b66f935ae878f706ea7837a`
`/rdp/DEPLOY.txt`	1,682 B	`bacba54b7b2cc61001c213089bcf104850d576dca37060aa03d0a141c48ef664`
`/reflect_decrypt.ps1`	1,350 B	`0a0b9cf262dd57c04f8a7061f7841e1544f1080ece9ace2f1385743191c902c4`

Supporting strings: CLSID `{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}`, scheduled task `InjectNow`, hidden desktop `RemoteDesktopS`, folder `%LOCALAPPDATA%\Microsoft\RdpService\`, and URI paths `/dl/`, `/up`, `/cmd?box=` and `/r2?box=`.

MITRE ATT&CK mapping

Tactic	Technique	How the tooling uses it	Detection
Persistence, Privilege Escalation	T1546.015 Component Object Model Hijacking	Per-user `InProcServer32` override pointing to a DLL in the public folder	Sigma 1; KQL registry branch; YARA CLSID rule
Collection	T1056.001 Input Capture: Keylogging	Inline C# `WH_KEYBOARD_LL` hook compiled by PowerShell	Sigma 3; KQL AMSI branch; YARA keylogger rule
Collection	T1113 Screen Capture	`CopyFromScreen` PNG capture uploaded to `:8091/up`	SID 9802603; KQL process and AMSI branches
Credential Access	T1539 Steal Web Session Cookie	Assessed upstream theft of the cookies the grafters reuse	Not directly covered
Defense Evasion, Lateral Movement	T1550.004 Use Alternate Authentication Material: Web Session Cookie	CDP WebSocket cookie grafting on port 9222	Sigma 2; SID 9802602; KQL process branch
Defense Evasion	T1036.005 Masquerading: Match Legitimate Name or Location	`mstsc.exe` under `%LOCALAPPDATA%\Microsoft\RdpService\`; "Windows Update Helper" assembly strings	Sigma 4; KQL process branch; YARA XWorm rule

10

CHAPTER 10

Scope and Limits

IN SHORT

This chapter sets out what the evidence can show and what it cannot.

10 Scope and Limits

- **Window.** The build-factory files were indexed on 13 November 2026 (Rust host) and 16 November 2026 (Russian host). The six RAT controllers were confirmed live between 13 and 16 November 2026.
- **Files, not outcomes.** We saw files on exposed servers, not activity on victim systems. Scripts are described by what they are written to do. Whether any of them ran against a target is unknown.
- **Partial content.** Of the 1,305 files, 85 text, source and manifest files have recorded SHA-256 values. The other 1,220 are known by name and size only. The Rust `src/` folder was not exposed, so module roles come from file names.
- **RAT ratios are not survival rates.** The 920 recently active candidates and 19,391 historical records were not each checked in the window. The 19,385 unconfirmed records are unverified, not benign.
- **Attribution.** The Russian workbench fits a criminal operator. A red-team owner of the Rust factory remains plausible. Nothing links the two servers.
- **CLSID owner.** The COM class behind `{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}` is unresolved. Chrome loading it is an inference.
- **Rule testing.** Every published rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma and Microsoft's Kusto.Language analyzer, checked on 20 November 2026). The rules were not replayed against live traffic, so test and tune them in your own environment before production use.

11

CHAPTER 11

Sources

IN SHORT

This chapter lists the public references used in the report.

11 Sources

1. MITRE ATT&CK. T1546.015 Component Object Model Hijacking. https://attack.mitre.org/techniques/T1546/015/ (verified 18 November 2026)
2. Rust Cargo Book. Build cache and dep-info files. https://doc.rust-lang.org/cargo/guide/build-cache.html (verified 18 November 2026)
3. MITRE ATT&CK. T1056.001 Input Capture: Keylogging. https://attack.mitre.org/techniques/T1056/001/ (verified 18 November 2026)
4. MITRE ATT&CK. T1113 Screen Capture. https://attack.mitre.org/techniques/T1113/ (verified 18 November 2026)
5. MITRE ATT&CK. T1550.004 Use Alternate Authentication Material: Web Session Cookie. https://attack.mitre.org/techniques/T1550/004/ (verified 18 November 2026)
6. MITRE ATT&CK. Software S1087 AsyncRAT. https://attack.mitre.org/software/S1087/ (verified 18 November 2026)
7. Google Security Blog. Improving the security of Chrome cookies on Windows. https://security.googleblog.com/2024/07/improving-security-of-chrome-cookies-on.html (verified 18 November 2026)
8. Chrome for Developers. Remote debugging switch changes in Chrome 136. https://developer.chrome.com/blog/remote-debugging-port (verified 18 November 2026)
9. Microsoft Learn. Merged view of HKEY_CLASSES_ROOT. https://learn.microsoft.com/en-us/windows/win32/sysinfo/merged-view-of-hkey-classes-root (verified 18 November 2026)

APPENDICES

Indicators and Detection Rules

FOR DEFENDERS

Appendix A lists the indicators published for this study. Appendix B carries every published detection rule in full.

INSIDE

A. Indicators of compromise (21) 38

B. Detection rules (4 files) 40

A Indicators of Compromise

Values are defanged for display. The CSV, JSON and STIX files on the online edition carry the raw values. Values the report marks as withheld are not listed.

ID	Type	Indicator	Port	Context	Action
R08-IOC-01	IPv4	193.233.220[.]65	8080	Exposed XWorm 5.6 VB.NET, CDP Cookie Grafting & Per-User COM Hijack Workbench	BLOCK
R08-IOC-02	SHA-256	d7907bd930b212675b2686f17a702339c7be3703f6b9e1c4c0c99a3b8df81ce2	8080	XWorm 5.6 Client Visual Basic Source (Windows Update Helper Masquerade) SHA-256	HUNT
R08-IOC-03	SHA-256	ca163c0a6e102d9bf1058fbe99db94964115bd414514accf7cb2b31111a54bb6	8080	XWorm 5.6 Builder/Designer Visual Basic Source (Windows Update Util) SHA-256	HUNT
R08-IOC-04	SHA-256	c4c51d14e97504864ff493b202f57cb6bdc43c6a9b66f935ae878f706ea7837a	8080	XWorm 5.6 Secondary Builder Visual Basic Source (Windows Update Util) SHA-256	HUNT
R08-IOC-05	SHA-256	1fd0695487372a62306721a3f4f70ef042bf224bf96adc31ea889c41febc2b34	8080	XWorm On-Box vbc.exe Client Visual Basic Source (AssemblyTitle Test) SHA-256	HUNT
R08-IOC-06	SHA-256	01393dbf9a23130e2d3899b1f1c80df9d5ed28cff01d60c5472e56a737b28923	8080	On-Box certutil & .NET v4.0.30319 vbc.exe Compiler Batch Script SHA-256	HUNT
R08-IOC-07	SHA-256	c7caa7cadec4f43a0e72347a50da4cbf0473fa79f8ed09f7c1d5584e29415ae7	8080	PowerShell C# SetWindowsHookEx WH_KEYBOARD_LL Keylogger SHA-256	HUNT
R08-IOC-08	SHA-256	d747bd06488cf50c826327a7b0deb794d4cfa6c7051af9c2bd4809a75f99ba0a	8080	screenshot.ps1 · PowerShell screen grab and uploader to port 8091	HUNT
R08-IOC-09	SHA-256	1a0d0943bb4382a0e1d55bfd02f4970f572d4f7627691464f36755d354d7034e	8080	RDP Stage-1 Plain-HTTP :443/dl Cookie-Injector Downloader SHA-256	HUNT
R08-IOC-10	SHA-256	d657df005418f9cbee07391a0f2e05de7e908e87ec35db228062e6bbf2678278	8080	Detached schtasks (InjectNow) CDP Cookie-Grafting Launcher SHA-256	HUNT
R08-IOC-11	SHA-256	ceb17c70555ff457c2402f1d970626f59c3481377c1277ae6e8831c51cb12ecb	8080	Chrome CDP Port-9222 Relaunch & WebSocket Cookie Grafting Script SHA-256	HUNT
R08-IOC-12	SHA-256	e29bbc5f3650cf3851ca8d8e25b22865809872216372b4672fff0322081109f4	8080	Chrome CDP Port-9222 WebSocket Cookie Grafting Script (No-Relaunch Variant) SHA-256	HUNT
R08-IOC-13	SHA-256	5ef9a57c68ce939589e0689b0ee86c0caadd48a2eb0eb587974403ddc7ed14ee	8080	Chrome CDP Port-9222 Standalone JSON Cookie Grafting Script SHA-256	HUNT
R08-IOC-14	SHA-256	7f231f37f35f2283b492b28d3987be19cc43a0a22956a37e23ea46c0e1832565	8080	Chrome Termination & Per-User HKCU CLSID {1f486a52-…} InProcServer32 Hijacker SHA-256	HUNT
R08-IOC-15	SHA-256	bacba54b7b2cc61001c213089bcf104850d576dca37060aa03d0a141c48ef664	8080	Real-Profile HVNC Chrome 127+ App-Bound Encryption Workaround Runbook SHA-256	HUNT
R08-IOC-16	SHA-256	0a0b9cf262dd57c04f8a7061f7841e1544f1080ece9ace2f1385743191c902c4	8080	Reflective .NET Assembly::LoadFile XWorm V3.1 Backdoor Decryptor SHA-256	HUNT
R08-IOC-17	IPv4	102.220.163[.]109	443	Confirmed Live ASYNCRAT Commodity RAT C2 Listener	BLOCK
R08-IOC-18	IPv4	148.113.214[.]176	443	Confirmed Live DCRAT Commodity RAT C2 Listener	BLOCK

ID	Type	Indicator	Port	Context	Action
R08-IOC-19	IPv4	141.98.157[.]101	2288	Confirmed Live QUASAR Commodity RAT C2 Listener	BLOCK
R08-IOC-20	IPv4	5.161.43[.]175	443	Confirmed Live VENOMRAT Commodity RAT C2 Listener	BLOCK
R08-IOC-21	IPv4	154.12.90[.]49	443	Confirmed Live VENOMRAT Commodity RAT C2 Listener	BLOCK

B Detection Rules

Each rule file loads in its own engine (Kusto, pySigma, Suricata 7, YARA-X). Tune thresholds and allow-lists to your environment before alerting or blocking. The same files can be downloaded from the online edition.

sentinel_clsid_hijack_and_ps_hook_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Report #08 Microsoft Sentinel & Defender XDR Hunting Suite
// Classification: TLP:CLEAR
// Purpose: Hunts UAC-free per-user HKCU CLSID {1f486a52-3cb1-48fd-8f50-b8dc300d9f9d} InProcServer32 hijacks
//          (matching HKEY_USERS\<SID>_Classes\CLSID\... and \Classes\CLSID\... via substring predicates),
//          C:\Users\Public\*.dll browser loads, curl/certutil staging, PowerShell/AMSI hooks, mstsc.exe
masquerading,
//          and Chromium --remote-debugging-port cookie grafting (excluding QA/CI chromedriver/Playwright/
Puppeteer).

let Lookback = 14d;
let StagedPerUserClsid = "{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}";
let BrowserBins = dynamic(["chrome.exe", "msedge.exe", "brave.exe", "explorer.exe", "mstsc.exe"]);
let regHijacks =
    DeviceRegistryEvents
    | where Timestamp >= ago(Lookback) and ActionType in ("RegistryValueSet", "RegistryKeyCreated")
    | where (RegistryKey contains "_Classes\\CLSID\\" or RegistryKey contains "\\Classes\\CLSID\\")
        and RegistryKey contains "\\InProcServer32"
    | where RegistryKey contains StagedPerUserClsid
         or RegistryValueData contains "C:\\Users\\Public\\"
         or RegistryValueData contains "\\AppData\\Local\\Temp\\"
         or RegistryValueData has_any ("hijack.dll", "hijack_standalone.dll")
    | project Timestamp, DeviceName, DetectionSurface = "HKCU_CLSID_InProcServer32_Hijack",
              ActorProcess = InitiatingProcessFileName, CommandLine = InitiatingProcessCommandLine,
              Artifact = strcat(RegistryKey, " -> ", RegistryValueData);
let publicDllLoads =
    DeviceImageLoadEvents
    | where Timestamp >= ago(Lookback) and InitiatingProcessFileName in~ (BrowserBins)
    | where FolderPath startswith "C:\\Users\\Public\\"
         or FileName in~ ("hijack.dll", "hijack_standalone.dll", "HVNC.dll", "hvnc_test.dll", "HVNC_patched.dll",
"HVNC_final.dll")
    | project Timestamp, DeviceName, DetectionSurface = "UserWritable_DLL_Loaded_Into_Browser_Or_Shell",
              ActorProcess = InitiatingProcessFileName, CommandLine = InitiatingProcessCommandLine,
              Artifact = strcat(iff(FolderPath endswith FileName, FolderPath, strcat(trim_end(@"\\+",
FolderPath), "\\", FileName)), " (SHA256=", SHA256, ")");
let psHooksAndCdp =
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where (FileName in~ ("powershell.exe", "pwsh.exe", "csc.exe", "schtasks.exe", "wscript.exe", "cscript.exe")
             and (ProcessCommandLine has_any ("SetWindowsHookEx", "WH_KEYBOARD_LL", "CopyFromScreen",
                                              "keylogger.ps1", "screenshot.ps1", "redeploy_force.ps1",
"inject.ps1",
                                              "inject_only.ps1", "inject_standalone.ps1", "inj.ps1",
"run_inj.bat", "InjectNow")
                  or ProcessCommandLine contains "193.233.220.65"))
         or (FileName =~ "vbc.exe"
             and ProcessCommandLine contains "/target:winexe"
             and (ProcessCommandLine contains "/out:client.exe" or InitiatingProcessFileName in~ ("cmd.exe",
"powershell.exe", "pwsh.exe", "wscript.exe")))
         or (FileName in~ ("curl.exe", "certutil.exe", "bitsadmin.exe")
             and (ProcessCommandLine contains "193.233.220.65" or ProcessCommandLine contains ":443/dl/" or
ProcessCommandLine contains "/cmd?box=" or ProcessCommandLine contains "client.vb"))
         or (FileName =~ "mstsc.exe" and FolderPath contains "\\AppData\\Local\\")
         or (FileName in~ ("chrome.exe", "msedge.exe", "brave.exe")
             and ProcessCommandLine contains "--remote-debugging-port="
             and (InitiatingProcessFileName in~ ("powershell.exe", "pwsh.exe", "cmd.exe", "svchost.exe",
"taskeng.exe", "taskhostw.exe", "wscript.exe", "cscript.exe", "mshta.exe", "rundll32.exe")
                  or ProcessCommandLine contains "--remote-allow-origins=*"
                  or (ProcessCommandLine contains "--user-data-dir="
                      and (ProcessCommandLine contains "\\AppData\\Local\\Google\\Chrome\\"
                           or ProcessCommandLine contains "\\AppData\\Local\\Microsoft\\Edge\\"
```

```
                          or ProcessCommandLine contains "\\AppData\\Local\\BraveSoftware\\"))))
    | project Timestamp, DeviceName, DetectionSurface = "PowerShell_Hook_VbcCompile_Fetcher_Mstsc_Or_CDP",
              ActorProcess = FileName, CommandLine = ProcessCommandLine,
              Artifact = InitiatingProcessCommandLine;
let amsiScriptHooks =
    DeviceEvents
    | where Timestamp >= ago(Lookback) and ActionType in~ ("PowerShellCommand", "AmsiScriptContent",
"AmsiScriptDetection")
    | where AdditionalFields contains StagedPerUserClsid
        or AdditionalFields has_any ("SetWindowsHookEx", "WH_KEYBOARD_LL", "CopyFromScreen", "ClientWebSocket")
    | project Timestamp, DeviceName, DetectionSurface = "AMSI_PowerShell_ScriptBody_Hook_Or_CDP",
              ActorProcess = InitiatingProcessFileName, CommandLine = InitiatingProcessCommandLine,
              Artifact = tostring(AdditionalFields);
union regHijacks, publicDllLoads, psHooksAndCdp, amsiScriptHooks
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp), EventCount = count(),
            SampleArtifacts = make_set(Artifact, 5), SampleCommands = make_set(CommandLine, 5)
        by DeviceName, DetectionSurface, ActorProcess
| order by LastSeen desc
```

sigma_per_user_com_hijack_and_rdp_inject.yml

SIGMA

```
title: Per-User HKCU COM CLSID InProcServer32 Hijack to Public Directory
id: d4b8e312-7c94-4a5f-9e31-6c18f0294a7b
status: experimental
description: >
  Detects per-user HKCU Component Object Model (COM) InProcServer32 hijacking targeting
  CLSID {1f486a52-3cb1-48fd-8f50-b8dc300d9f9d} (paired with Chrome restarts in redeploy_force.ps1,
  CLSID owner unresolved) or pointing COM server DLL paths into user-writable directories
  (C:\Users\Public\ or AppData\Local\Temp\) for UAC-free persistence. Matches UsrClass.dat
  mounts in Sysmon (HKU\<SID>_Classes\CLSID\...) and Defender XDR (HKEY_USERS\<SID>_Classes\CLSID\...)
  as well as HKLM/HKCU \Software\Classes\CLSID\ paths.
author: Odin by Brandefense
date: 2026-11-16
logsource:
  category: registry_set
  product: windows
detection:
  selection_clsid_hive:
    TargetObject|contains:
      - '_Classes\CLSID\'
      - '\Classes\CLSID\'
  selection_inproc:
    TargetObject|contains: '\InProcServer32'
  selection_specific_clsid:
    TargetObject|contains: '{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}'
  selection_suspicious_payload:
    Details|contains:
      - 'C:\Users\Public\'
      - '\AppData\Local\Temp\'
      - 'hijack.dll'
      - 'hijack_standalone.dll'
  condition: selection_clsid_hive and selection_inproc and (selection_specific_clsid or
selection_suspicious_payload)
falsepositives:
  - Rare per-user developer COM shims explicitly registered from C:\Users\Public or Temp
level: critical
tags:
  - attack.persistence
  - attack.privilege-escalation
  - attack.t1546.015
---
title: Chromium Relaunched with Remote Debugging Port for CDP Cookie Grafting
id: a7f2c904-3b61-4d8e-b152-8e47d9301c6a
status: experimental
description: >
  Detects shell interpreters, script hosts, or scheduled task launchers relaunching Chromium
  browsers with --remote-debugging-port= (or pairing remote debugging with --remote-allow-origins=*
  or a custom --user-data-dir= inside vendor browser AppData\Local directories to bypass Chrome 136+
```

```
    default-profile debugging restrictions) while excluding Selenium, chromedriver, Puppeteer,
    Playwright, and Cypress QA automation harnesses.
  author: Odin by Brandefense
  date: 2026-11-16
  logsource:
    category: process_creation
    product: windows
  detection:
    selection_browser:
      Image|endswith:
        - '\chrome.exe'
        - '\msedge.exe'
        - '\brave.exe'
      CommandLine|contains: '--remote-debugging-port='
    selection_parent:
      ParentImage|endswith:
        - '\powershell.exe'
        - '\pwsh.exe'
        - '\cmd.exe'
        - '\svchost.exe'
        - '\taskeng.exe'
        - '\taskhostw.exe'
        - '\wscript.exe'
        - '\cscript.exe'
        - '\mshta.exe'
        - '\rundll32.exe'
    selection_vendor_profile_sibling:
      CommandLine|contains: '--user-data-dir='
    selection_vendor_appdata_root:
      CommandLine|contains:
        - '\AppData\Local\Google\Chrome\'
        - '\AppData\Local\Microsoft\Edge\'
        - '\AppData\Local\BraveSoftware\'
    selection_allow_origins:
      CommandLine|contains: '--remote-allow-origins=\*'
    condition: selection_browser and (selection_parent or selection_allow_origins or
  (selection_vendor_profile_sibling and selection_vendor_appdata_root))
  falsepositives:
    - Custom developer scripts invoking PowerShell to attach non-headless Chrome debugging sessions against local
  profiles
  level: high
  tags:
    - attack.lateral-movement
    - attack.t1550.004
  ---
  title: PowerShell Inline CSharp Low-Level Keyboard Hook Compilation
  id: 5e91b648-2d73-4f1c-8a94-3b05e7281d9f
  status: experimental
  description: >
    Detects PowerShell scripts compiling inline CSharp P/Invoke definitions via Add-Type to
    install low-level Windows keyboard hooks (SetWindowsHookEx with WH_KEYBOARD_LL = 13).
  author: Odin by Brandefense
  date: 2026-11-16
  logsource:
    category: ps_script
    product: windows
  detection:
    selection_hook:
      ScriptBlockText|contains|all:
        - 'Add-Type'
        - 'SetWindowsHookEx'
        - 'WH_KEYBOARD_LL'
    selection_win32:
      ScriptBlockText|contains:
        - 'GetForegroundWindow'
        - 'CallNextHookEx'
    condition: selection_hook and selection_win32
  falsepositives:
    - Specialized accessibility or hotkey remapping scripts authored in inline PowerShell CSharp
  level: high
  tags:
    - attack.collection
```

```
    - attack.credential-access
    - attack.t1056.001
---
title: Masqueraded mstsc.exe Execution from User-Writable AppData Path
id: 3c64a819-9f20-4e7b-a538-1d72b4906e5c
status: experimental
description: >
    Detects execution of an executable named mstsc.exe from user-writable AppData directories
    (such as %LOCALAPPDATA%\Microsoft\RdpService\mstsc.exe documented in /rdp/DEPLOY.txt for
    HVNC DLL sideloading) rather than legitimate Windows System32 or SysWOW64 paths.
author: Odin by Brandefense
date: 2026-11-16
logsource:
    category: process_creation
    product: windows
detection:
    selection_img:
        Image|endswith: '\mstsc.exe'
    selection_user_appdata:
        Image|contains:
            - '\AppData\Local\'
            - '\AppData\Roaming\'
            - 'C:\Users\Public\'
    filter_system_dirs:
        Image:
            - 'C:\Windows\System32\mstsc.exe'
            - 'C:\Windows\SysWOW64\mstsc.exe'
    condition: selection_img and selection_user_appdata and not filter_system_dirs
falsepositives:
    - None expected in standard Windows enterprise environments
level: high
tags:
    - attack.defense-evasion
    - attack.t1036.005
```

suricata_rust_xworm_and_rdp_staging.rules

SURICATA

```
# Odin by Brandefense - Report #08 Suricata 7.x Build-Factory, XWorm 5.6 & RDP CLSID Staging Suite
# Classification: TLP:CLEAR
# Purpose: Calibrated NDR rules for plain-HTTP port-443 per-user COM DLL staging, CDP cookie-injector downloads,
#          raw PNG /up exfiltration, :8090 batch command polling,
#          XWorm 5.6 VB.NET source files (>=6-byte fast_pattern), and RAT TLS subjects.
# Note: sid:9802604 is retired; the remaining eleven rules keep their original SIDs.

# 1. Plain-HTTP staging of per-user COM InProcServer32 hijack DLL (including unencrypted HTTP over TCP 443)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BLD Per-User COM Hijack DLL Plain-HTTP Staging
(hijack.dll / hijack_standalone.dll)"; flow:established,to_server; http.uri; content:"hijack"; nocase;
fast_pattern; pcre:"/\/(?:dl\/|rdp\/)?hijack(?:_standalone)?\.dll(?:\?|$)/i"; classtype:trojan-activity; sid:
9802601; rev:3;)

# 2. RDP session & Chrome DevTools Protocol (CDP) PowerShell cookie-grafting script retrieval
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BLD RDP Session and Chrome CDP PowerShell Cookie-
Injector Staging (inject*.ps1)"; flow:established,to_server; http.uri; content:"inject"; nocase; fast_pattern;
pcre:"/\/(?:dl|rdp)\/inject(?:_only|_standalone)?\.ps1(?:\?|$)/i"; classtype:trojan-activity; sid:9802602; rev:
3;)

# 3. PowerShell CopyFromScreen raw PNG screenshot exfiltration via HTTP POST to /up (anchored to exclude /
upload*)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BLD PowerShell CopyFromScreen Raw PNG Exfiltration POST
(/up)"; flow:established,to_server; http.method; content:"POST"; http.uri; content:"/up"; startswith; pcre:"/^\/
up(?:\?|$)/"; http.request_body; content:"|89 50 4e 47 0d 0a 1a 0a|"; startswith; fast_pattern; classtype:trojan-
activity; sid:9802603; rev:3;)


# 5. Unobfuscated XWorm 5.6 VB.NET client source retrieval (17-byte fast_pattern; excludes generic /client.vb)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BLD XWorm 5.6 Visual Basic .NET Client Source Retrieval
(xworm56_client.vb)"; flow:established,to_server; http.uri; content:"xworm56_client.vb"; nocase; fast_pattern;
pcre:"/\/xworm56_client\.vb(?:\?|$)/i"; classtype:trojan-activity; sid:9802605; rev:4;)
```

```
# 6. Stock AsyncRAT self-signed TLS certificate subject (corroborate with active endpoint egress process)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-BLD Stock AsyncRAT Self-Signed TLS Certificate Subject
(CN=AsyncRAT Server)"; flow:established,to_client; tls.cert_subject; content:"CN=AsyncRAT Server"; nocase;
fast_pattern; classtype:command-and-control; sid:9802606; rev:3;)

# 7. Self-signed CN=default TLS certificate subject observed on confirmed RAT C2s and candidates (pair with
process/IP telemetry)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-BLD Commodity RAT Self-Signed CN=default TLS Certificate
Subject (Corroborate Egress Process)"; flow:established,to_client; tls.cert_subject; content:"CN=default";
nocase; fast_pattern; pcre:"/(?:^|,\s*)CN=default(?:,|$)/i"; classtype:command-and-control; sid:9802607; rev:3;)

# 8. Stock Quasar RAT self-signed TLS certificate subject (corroborate with active endpoint egress process)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-BLD Stock Quasar RAT Self-Signed TLS Certificate Subject
(CN=Quasar Server CA)"; flow:established,to_client; tls.cert_subject; content:"CN=Quasar Server CA"; nocase;
fast_pattern; classtype:command-and-control; sid:9802608; rev:1;)

# 9. Stock DcRat self-signed TLS certificate subject (multi-RDN compatible; corroborate with endpoint process)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-BLD Stock DcRat Self-Signed TLS Certificate Subject
(CN=DcRat)"; flow:established,to_client; tls.cert_subject; content:"CN=DcRat"; nocase; fast_pattern; pcre:"/
(?:^|,\s*)CN=DcRat(?:,|$)/i"; classtype:command-and-control; sid:9802609; rev:1;)

# 10. Unobfuscated XWorm 5.6 VB.NET builder/designer source retrieval (10-byte fast_pattern)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BLD XWorm 5.6 Visual Basic .NET Builder Source Retrieval
(xw56_desig*.vb)"; flow:established,to_server; http.uri; content:"xw56_desig"; nocase; fast_pattern; pcre:"/\/
xw56_desig2?\.vb(?:\?|$)/i"; classtype:trojan-activity; sid:9802610; rev:1;)

# 11. Unencrypted Cleartext HTTP GET /dl/ Payload Staging Over TCP Port 443
alert http $HOME_NET any -> $EXTERNAL_NET 443 (msg:"ODIN-BLD Cleartext HTTP /dl/ Payload Staging Over TCP Port
443"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/dl/"; startswith;
http.protocol; content:"HTTP/1."; fast_pattern; classtype:policy-violation; sid:9802611; rev:1;)

# 12. Batch HTTP Command Poller Loop (/cmd?box= via curl User-Agent, matching np.bat)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BLD Batch HTTP Command Poller Heartbeat (/cmd?box=)";
flow:established,to_server; http.uri; content:"/cmd?box="; nocase; fast_pattern; pcre:"/^\/cmd\?box=[A-Za-
z0-9_.-]+/i"; http.user_agent; content:"curl/"; startswith; classtype:trojan-activity; sid:9802612; rev:1;)
```

yara_xworm56_and_rdp_workbench.yar

YARA

```
/*
   Odin by Brandefense - Report #08 YARA Malware Build-Factory & Workbench Rules
   Classification: TLP:CLEAR
*/


rule ODIN_XWorm56_VBNet_Windows_Update_Helper_Masquerade {
    meta:
        author = "Odin by Brandefense"
        description = "Detects XWorm 5.6 VB.NET source files (incl. /client.vb), /build.bat vbc.exe stagers, and
masquerade-string .NET assemblies"
        classification = "TLP:CLEAR"
    strings:
        $m1 = "Windows Update Helper Service" ascii wide
        $m2 = "Windows Update Util Service" ascii wide
        $v1 = "Imports Microsoft.VisualBasic.CompilerServices" ascii wide
        $v2 = "AssemblyFileVersion(\"1\" + \".\" + \"0\" + \".\" + \"0\" + \".\" + \"0\")" ascii wide
        $n1 = "System.Net.Sockets" ascii wide
        $n2 = "System.Security.Cryptography" ascii wide
        $n3 = "System.Management" ascii wide
        $pe_vb = "Microsoft.VisualBasic.CompilerServices" ascii wide
        $b1 = "vbc.exe /target:winexe /out:client.exe" ascii wide nocase
        $b2 = "certutil -urlcache -split -f" ascii wide nocase
    condition:
        filesize < 10MB and (
            (($m1 or $m2) and ($v1 or $v2 or ($pe_vb and $n1 and $n2)))
            or ($v1 and $v2 and $n1 and $n2 and $n3)
            or ($b1 and $b2)
        )
```

```
}

rule ODIN_RDP_PerUser_CLSID_InProcServer32_Hijacker {
    meta:
        author = "Odin by Brandefense"
        description = "Detects PowerShell scripts terminating Chrome and registering per-user HKCU CLSID
{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d} InProcServer32"
        classification = "TLP:CLEAR"
    strings:
        $clsid = "{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}" ascii wide nocase
        $reg = "Classes\\CLSID\\" ascii wide nocase
        $inproc = "\\InProcServer32" ascii wide nocase
        $dll1 = "C:\\Users\\Public\\hijack.dll" ascii wide nocase
        $dll2 = "hijack_standalone.dll" ascii wide nocase
    condition:
        filesize < 2MB and $clsid and ($inproc or $reg) and ($dll1 or $dll2)
}

rule ODIN_PowerShell_CSharp_WH_KEYBOARD_LL_Keylogger {
    meta:
        author = "Odin by Brandefense"
        description = "Detects PowerShell Add-Type inline CSharp low-level keyboard hook keyloggers
(WH_KEYBOARD_LL)"
        classification = "TLP:CLEAR"
    strings:
        $a1 = "Add-Type" ascii wide nocase
        $h1 = "WH_KEYBOARD_LL" ascii wide
        $h2 = "SetWindowsHookEx" ascii wide
        $h3 = "GetForegroundWindow" ascii wide
        $h4 = "0x0104" ascii wide
        $h5 = "CallNextHookEx" ascii wide
    condition:
        filesize < 2MB and $a1 and $h1 and $h2 and 2 of ($h3, $h4, $h5)
}
```

ODIN

About Odin Research

Odin Research is the public research series of Odin by Brandefense. It documents adversary infrastructure seen on the internet and turns each finding into indicators and detection rules that defenders can use. Victims and third parties stay anonymous, addresses appear only for confirmed adversary infrastructure, and no working attack code is published.

Odin is the agentic analyst inside the Brandefense threat intelligence platform, which unifies Cyber Threat Intelligence, Digital Risk Protection and External Attack Surface Management.

Online edition, indicators and rules · brandefense.io/odin-research/reports/exposed-build-factories/
Short Read · brandefense.io/odin-research/blog/two-exposed-build-servers-show-how-windows-malware-is-made/
Defense Kit · brandefense.io/odin-research/defense-kit/