ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 15 / 30 · COMMODITY MALWARE

HookBot Panels, Multi-Port RATs and Telegram Kits

Commodity Threat Infrastructure

How Android banking panels, multi-port Windows RAT servers and an open directory of flood-themed files run side by side on low-cost hosting

Two-week observation window ending 18 November 2026 · Prepared November 2026
Full Report · Public edition · Version 1.1

238
Mobile and desktop RAT listener observations (236 unique family, IP and port combinations) on 160 hosts

12
HookBot feed entries: 1 corroborated panel and 11 older entries for watchlists

121
AsyncRAT listeners on 55 IPs (2.20 per host), up to 22 ports on one Contabo VPS

18
PureRAT candidate listeners on 14 IPs; 9 of them on ports 56001 to 56003

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

How Android banking panels, multi-port Windows RAT servers and an open directory of flood-themed files run side by side on low-cost hosting

01	Executive Summary	04
02	Three Separate Ecosystems	07
03	HookBot Android Banking Panels	10
04	Multi-Port Desktop RAT Servers	14
05	The Port-755 Open Directory	18
06	What to Block and What to Watch	21
07	What Defenders Should Do	24
08	Detection Rules	26
09	Indicators	30
10	Scope and Limits	33
11	Sources	35
A	Indicators of compromise (6)	37
B	Detection rules (4 files)	38

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense mapped 238 mobile and desktop RAT listeners on 160 hosts, plus one open directory holding 40 botnet-themed files. Three separate ecosystems appear: HookBot Android banking panels, multi-port Windows RAT servers and Telegram-themed flood kits. Only six hosts carry enough signals to block. Block those six, hunt default RAT certificates and ports 56001 to 56003, and limit Telegram Bot API traffic to approved hosts.

01

CHAPTER 01

Executive Summary

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense mapped 238 mobile and desktop RAT listeners on 160 hosts, plus one open directory holding 40 botnet-themed files. Three separate ecosystems appear: HookBot Android banking panels, multi-port Windows RAT servers and Telegram-themed flood kits. Only six hosts carry enough signals to block. Block those six, hunt default RAT certificates and ports 56001 to 56003, and limit Telegram Bot API traffic to approved hosts.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 05

01 Executive Summary

Commodity cybercrime runs separate markets for mobile banking panels, desktop remote access trojans (RATs) and distributed denial-of-service (DDoS) kits. All three rent cheap virtual private servers (VPS) and use public messaging APIs. This report maps where these tools listen, which fingerprints give them away, and which hosts are safe to block.

1. **Twelve HookBot feed entries hold one corroborated panel and eleven older entries.**
   A server in Germany, 152.70.24[.]149, shows a dedicated HookBot panel certificate and JARM fingerprint in November 2026. The other 11 entries carry first-seen dates from 2025 and no fresh panel markers. They belong on watchlists. Public research documents HookBot's Accessibility abuse, overlays and VNC screen control; we saw no victim devices.

2. **Desktop RAT operators pack up to 22 listener ports onto one VPS.**
   Across 226 desktop listeners in seven families, AsyncRAT averages 2.20 listeners per host. One Contabo server binds 22 ports. PureRAT places 9 of its 18 candidate listeners on the sequential ports 56001 to 56003.

3. **An open directory on port 755 lists 40 botnet-themed files.**
   The files total 14.91 MB, including 7 ZIP archives and 31 JavaScript files. Their 39 stored files yield 33 unique SHA-256 hashes in five duplicate groups. File contents and server ownership are unverified, so the host stays on a watchlist.

Three Decisions for Leaders

1. **Treat Android Accessibility grants on banking devices as a fraud signal.** Require Google Play Integrity attestation, and detect active Accessibility services and screen projection during transfers. Block the corroborated panel 152.70.24[.]149 and keep the 2025 HookBot entries on watchlists.
2. **Detect RATs by certificate, JARM and port density, not single ports.** Block only the six multi-signal hosts. Hunt the default AsyncRAT and Quasar certificates and TLS handshakes on ports 56001 to 56003. Keep candidate hosts on watchlists.
3. **Allow Telegram Bot API traffic only from approved automation hosts.** Alert when script interpreters, abused .NET framework binaries or programs in user-writable folders connect to Telegram API domains or `/bot` paths.

The Study in Numbers

Number	What it is
238	Mobile and desktop RAT listener observations (236 unique family, IP and port combinations) on 160 hosts
12	HookBot feed entries: 1 corroborated panel and 11 older entries for watchlists
121	AsyncRAT listeners on 55 IPs (2.20 per host), up to 22 ports on one Contabo VPS
18	PureRAT candidate listeners on 14 IPs; 9 of them on ports 56001 to 56003
40	Files (14.91 MB) indexed on the open directory on port 755
33	Unique SHA-256 hashes across the 39 stored files, in 5 duplicate groups
6	Hosts with enough independent signals to block

How to use this report.

Security and fraud leaders can read Chapter 1, Chapter 6 and Chapter 7. SOC and detection engineers should start with Chapters 7 to 9. CTI analysts will find the HookBot, RAT and open-directory detail in Chapters 2 to 5.

02

CHAPTER 02

Three Separate Ecosystems

IN SHORT

The data shows three populations that share no IP address, certificate or verified operator link. We compare them with an analytical model of four roles, not as one coordinated group.

INSIDE THIS CHAPTER

What was observed 07

Families at a glance 08

The four-role model 08

02 Three Separate Ecosystems

What was observed

In the window, Odin by Brandefense recorded 238 listener observations. They cover one Android banking framework, HookBot, and seven Windows RAT or stealer families: AsyncRAT, Quasar, Remcos, PureRAT, DCRat, RedLine Stealer and VenomRAT. Separately, one open HTTP directory on port 755 listed 40 botnet-themed files.

The 12 HookBot hosts, the 148 desktop RAT hosts and the port-755 server share no IPv4 address. Together they make 161 studied hosts. Timestamps in this report are observation times, not infection times.

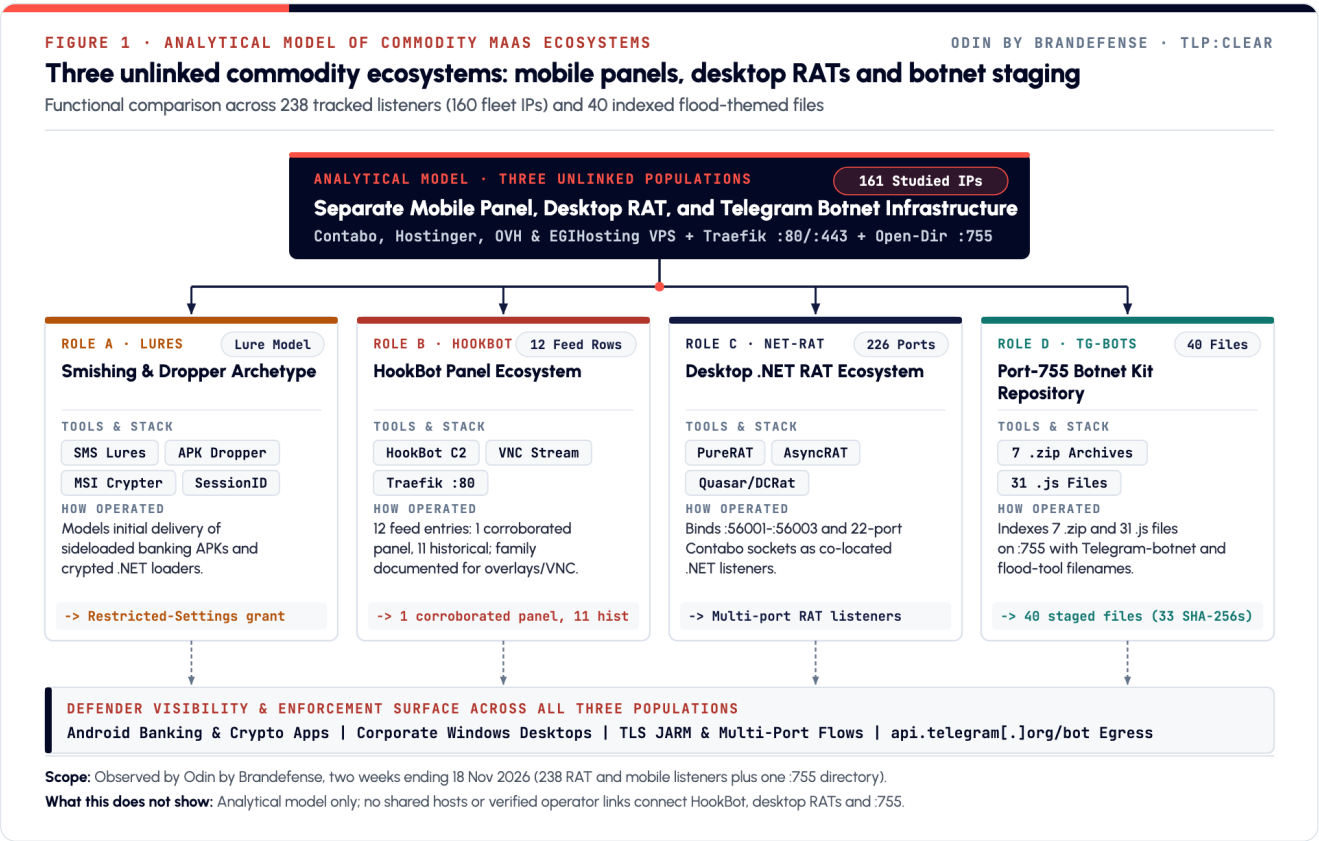


Figure 1: Three unlinked commodity ecosystems: mobile panels, desktop RATs and botnet staging

Families at a glance

Density is listeners per host. A value above 1.00 means operators stack several ports on one server.

Family	Platform and protocol	Listeners	Distinct IPs	Density	Status
AsyncRAT	Windows .NET, MessagePack over TLS	121	55	2.20x	1 confirmed, 11 suspected, 109 candidate
Quasar	Windows .NET, Protobuf over TLS	33	30	1.10x	2 confirmed (1 to block, 1 on a dynamic ISP), 4 suspected, 27 candidate
Remcos	Windows C++, RC4 or TLS	32	30	1.07x	25 suspected, 7 candidate
PureRAT	Windows .NET, Protobuf over TLS	18	14	1.29x	18 candidate (9 on ports 56001 to 56003)
DCRat	Windows .NET, modular C2	13	12	1.08x	1 confirmed, 3 suspected, 9 candidate
HookBot	Android, HTTP and VNC behind Traefik	12	12	1.00x	12 feed-confirmed (1 corroborated, 11 older)
RedLine Stealer	Windows .NET, HTTP or TCP stealer	5	5	1.00x	3 feed-confirmed (all watchlist), 1 suspected, 1 candidate
VenomRAT	Windows .NET, HVNC fork	4	4	1.00x	2 confirmed, 2 candidate

Adding the per-family hosts gives 162 addresses. The true count is 160, because two Colombian telecom addresses host more than one family. One binds AsyncRAT on ports 5010 and 5011 plus DCRat on 5020. The other, a dynamic ISP address, binds Remcos on 3000 and 3001 plus DCRat on 30001.

The 238 observations reduce to 236 unique sockets. Two sockets were recorded twice: 102.220.163[.]109:443 (AsyncRAT) and 141.98.157[.]101:2288 (Quasar).

The four-role model

This is an analytical model. It compares how commodity malware-as-a-service (MaaS) markets specialize. It does not claim handoffs between groups.

Role	What it covers	Evidence base
A. Delivery	Sideloaded Android `.apk` droppers that use session-based installers to get past Android 13 to 15 Restricted Settings, plus crypted Windows `.msi` and `.zip` loaders	Public reporting only
B. HookBot panels	Web panels built to control Android implants with overlays, PIN and 2FA capture, and VNC screen control	12 feed entries; no victim handsets seen
C. Multi-port desktop RATs	VPS on Contabo, OVH, HostPapa and EGIHosting binding several ports per host	226 desktop listeners; any split by affiliate or build is unverified
D. Botnet kit directory	Open directory with ZIP and JavaScript files whose names suggest Telegram botnets and flood tools	40 indexed files; contents not inspected

CHAPTER 03

HookBot Android Banking Panels

IN SHORT

Of 12 HookBot feed entries, only 152.70.24[.]49 in Germany shows November 2026 panel markers. The other 11 carry first-seen dates from 2025 and go to watchlists.

INSIDE THIS CHAPTER

The corroborated panel 10
The eleven older entries 10
What HookBot is built to do 11

03 HookBot Android Banking Panels

HookBot descends from the leaked Ermac and Hook Android banking trojans. CSIRT KNF, the Polish financial supervisor's CSIRT, documents it as an on-device fraud (ODF) trojan. On-device fraud means the attacker starts transfers from the victim's own phone and logged-in banking session. Our view is the panel side only. We saw no victim handsets.

The corroborated panel

Field	Value
Address	152.70.24[.]49, port 80 (also port 8443)
Hosting	Oracle Corporation, Germany
TLS certificate	`CN=panel.pecufos[.]com`
JARM	`1dd40d40d00040d00042d43d000000831b6af40378e2dd35eeac4e9311926e`
Page title	`HOOKBOT PANEL` (reported on port 8443)
Observed	13 November 2026

The certificate and JARM on port 80 are directly observed. Feed records add the `HOOKBOT PANEL` title on port 8443 and eight separate HookBot fingerprint matches. These cover the panel title, HTTP body hashes, certificate subject and SAN, a fork's panel title, and two host-rotation hashes. Together, these signals support blocking.

The eleven older entries

The other 11 entries carry earliest first-seen dates between July and November 2025. None shows a panel body hash or fresh November 2026 panel marker. Their addresses are withheld.

Country	Hosting	Port	TLS and JARM	Earliest first-seen
ID	Amazon.com, Inc.	443	`CN=TRAEFIK DEFAULT CERT`, JARM `29d29d...5a77db`	11 Nov 2025
CL	The Constant Company	443	`CN=TRAEFIK DEFAULT CERT`, JARM `26d26d...9cdc13`	26 Nov 2025
DE	Contabo GmbH	80	`CN=TRAEFIK DEFAULT CERT`, JARM `3fd3fd...51d0e6`	3 Oct 2025
DE	Contabo GmbH	80	`CN=TRAEFIK DEFAULT CERT`, JARM `3fd3fd...51d0e6`	4 Oct 2025
US	RouterHosting LLC	80	`CN=TRAEFIK DEFAULT CERT`, JARM `3fd3fd...7899fb`	23 Sep 2025
GB	EE Limited	443	Feed tag only	11 Jul 2025
ID	Hostinger International	443	Feed tag only	23 Oct 2025
LT	Hostinger International	80	JARM `0003fd...1f6145`	18 Aug 2025
LT	Hostinger or unattributed	80	Feed tag only	16 Oct 2025
NL	Ultahost, Inc.	80	Feed tag only	13 Oct 2025
Unattributed	VPS	80	Feed tag only	14 Sep 2025

Five of these hosts present the default certificate of the Traefik reverse proxy. The two German Contabo servers share the exact JARM `3fd3fd00000000000043d3fd3fd43d79451d8c63b099acafdbabb24551d0e6`. Their first-seen dates are 18 hours apart. That fits a standard Docker reverse-proxy template.

A first-seen date marks when a record entered the feed, not how long the panel stayed up. Addresses may have been reassigned since. One entry sits in EE Limited telecom space. Keep all 11 on watchlists until a live `HOOKBOT PANEL` banner is seen again.

What HookBot is built to do

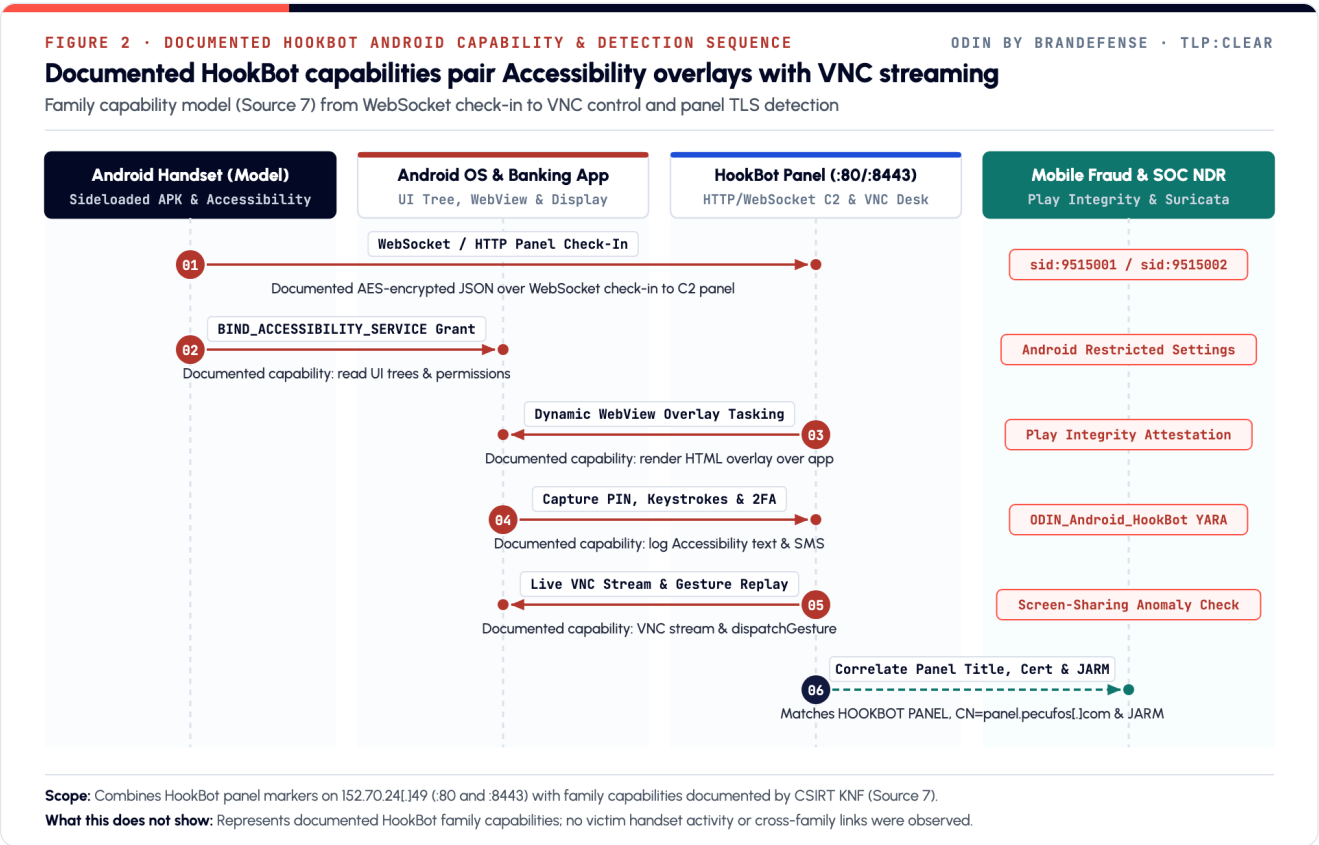


Figure 2: Documented HookBot capabilities pair Accessibility overlays with VNC streaming

CSIRT KNF's reverse engineering, with MITRE ATT&CK Mobile and the Android reference, documents four capabilities. None was seen on a victim device in this study.

Capability	How it works	ATT&CK
Accessibility abuse	Requests `BIND_ACCESSIBILITY_SERVICE` to read UI trees (`AccessibilityNodeInfo`), auto-approve `MediaProjection` and SMS permissions, and block Settings or uninstall	T1516
WebView overlays	Watches the foreground app and draws a C2-supplied HTML login form over banking apps; talks to the panel with AES-encrypted JSON over WebSocket	T1417.002
Keylogging and 2FA capture	Logs text-change events such as lock-screen PINs, seed phrases and one-time codes; intercepts SMS and bank push notifications	T1417.002
VNC control	Streams the screen to the panel behind a black overlay, so an operator can replay taps with `dispatchGesture` inside a logged-in session	T1516

Because the implant encrypts its WebSocket traffic with AES, network rules match the panel's HTTP title and TLS certificate, not implant payloads.

CHAPTER 04

Multi-Port Desktop RAT Servers

IN SHORT

Desktop RAT operators stack many listeners on one rented server. AsyncRAT reaches 22 ports on one Contabo host, and half of PureRAT's candidates sit on ports 56001 to 56003.

INSIDE THIS CHAPTER

PureRAT on sequential high ports 14

AsyncRAT: vertical stacking and horizontal sprawl 15

Sister families: default certificates, spoofing and cloned templates 16

04 Multi-Port Desktop RAT Servers

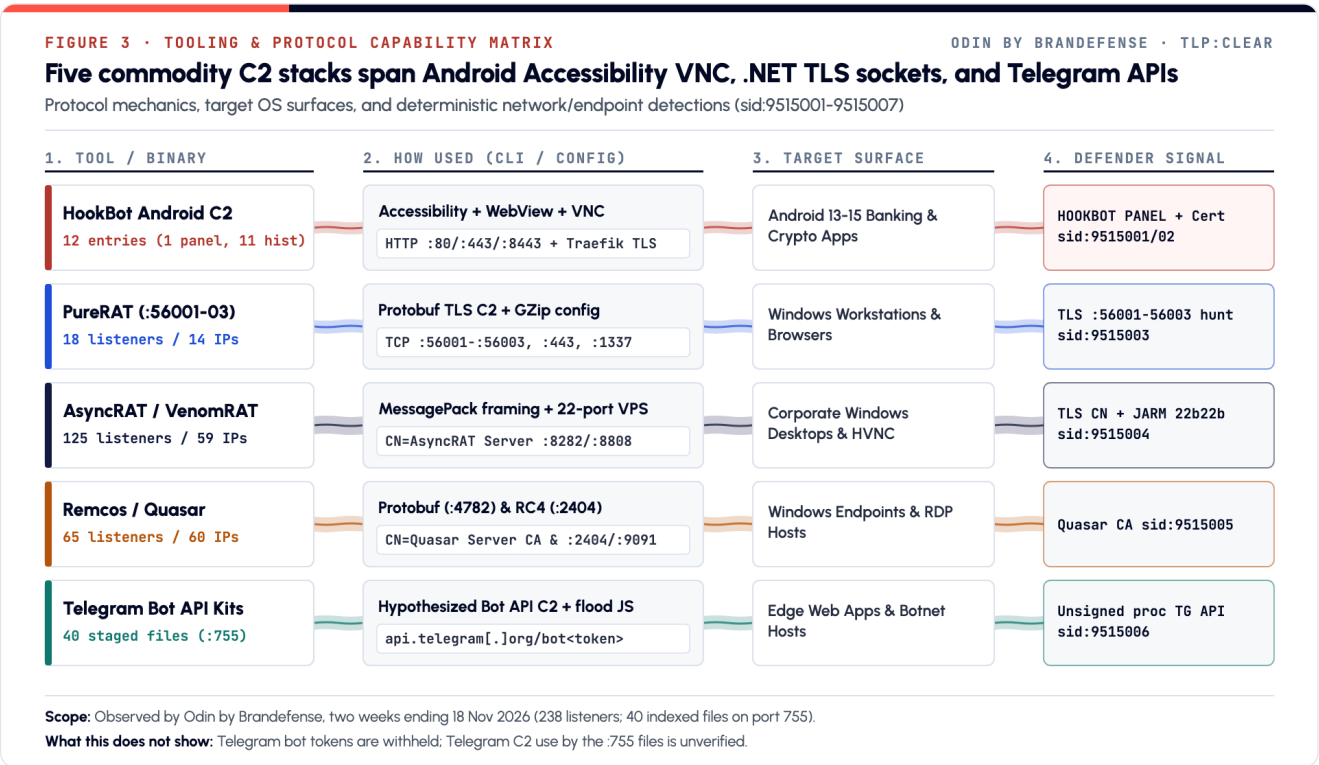


Figure 3: Five commodity C2 stacks span Android Accessibility VNC, .NET TLS sockets and Telegram APIs

PureRAT on sequential high ports

PureRAT, also tracked as PureHVNC, is a commercial modular .NET RAT. Public analyses document Hidden VNC, a reverse SOCKS5 proxy and credential-stealing plugins. On 17 November 2026 we saw 18 PureRAT candidate listeners on 14 hosts.

Half of them (9 of 18, 50.0%) use a three-port window. Port 56001 has 2 sockets, 56002 has 3 and 56003 has 4. The other nine use 443 (3), 442, 1337, 4443, 6767, 7500 and 7777 (one each).

Four candidate hosts bind two ports each. Their addresses are withheld, as are the custom certificate names.

Hosting	Country	Ports	What it shows
OVH SAS	SG	56002, 56003	Custom domain certificate on both sockets (name withheld), JARM `15d3fd...ccbc0b6b`
Tier.Net Technologies LLC	US	56002, 56003	Sequential pair, no certificate name recorded
ReliableSite.Net LLC	US	56002, 56003	Sequential pair, no certificate name recorded
Antbox Networks Limited	SG	442, 443	Adjacent HTTPS-like pair, no certificate name recorded

Huntress and Cyderes describe three layers in PureRAT builds:

- The .NET loader unpacks its payload with AES decryption and GZip decompression.

- The final configuration is stored as Base64-encoded, GZip-compressed protobuf.
- C2 traffic uses length-prefixed `protobuf-net` messages inside a .NET `SslStream` TLS session.

TLS hides the protobuf messages on the wire. So the Suricata rule for ports 56001 to 56003 is a hunt for TLS server handshakes, not a verdict. Pair it with endpoint and YARA hunting for `SslStream`, `protobuf-net` and `GZipStream` loader traits.

AsyncRAT: vertical stacking and horizontal sprawl

AsyncRAT shows two opposite patterns.

> **Vertical stacking on Contabo.**
>
> Contabo hosts 53 of the 121 AsyncRAT listeners (43.8%): 30 on Contabo Inc. and 23 on Contabo GmbH. Three Contabo candidate hosts bind 52 ports between them. Their addresses are withheld.

Host	Country	Ports	Pattern
Contabo candidate 1	FR	22	6006 plus 21 ports between 10100 and 30900
Contabo candidate 2	US	16	Ports 80 to 50100
Contabo candidate 3	US	14	Ports 70 to 20700
102.220.163[.]109 (confirmed)	SI, VPS Dedicated LLC	11	443 with `CN=AsyncRAT Server`, plus 10 high ports from 2502 to 8808

The confirmed host's JARM on port 443 is `22b22b07b22b22b22b22b22b22b22bfd9c9d14e4f4f67f94f0359f8b28f532`.

> **Horizontal sprawl on EGIHosting.**
>
> Sixteen candidate hosts in one EGIHosting /24 subnet each run a single AsyncRAT listener. All 16 use TCP 8282 and the generic certificate `CN=Server`. The subnet is withheld.

Sister families: default certificates, spoofing and cloned templates

The other desktop families reuse default RAT certificates, spoof software brands or clone Windows RDP templates. All seven custom domain certificates seen on candidate or suspected hosts are withheld.

Family	Host and port	Hosting	What it shows	Action
Quasar	141.98.157[.]101:2288	GB, EUGAMEHOST LTD	Stock `CN=Quasar Server CA`, JARM `2ad2ad...3585be`, seen twice	Block
Quasar	Withheld, port 5000	CA, Shaw Communications	Stock `CN=Quasar Server CA` on a cable subscriber ISP	Short watchlist
DCRat	148.113.214[.]176:443 (and 80)	CA, OVH SAS	Wildcard certificate spoofing a download-manager brand (name withheld)	Block
DCRat	Withheld, port 3232	ES, IONOS SE	Default `CN=DcRat`	Watchlist
VenomRAT	5.161.43[.]175:443	US, Hetzner	`CN=akashicsymbiosis[.]ai`, JARM `27d40d...`	Block
VenomRAT	154.12.90[.]49:443	HK, cognetcloud INC	`CN=dgtwxj[.]com`, JARM `3fd3fd...`	Block
RedLine Stealer	Withheld, port 5800	FI, Hetzner Online GmbH	Feed tag and port only; no certificate, JARM or banner	Watchlist
RedLine Stealer	Two hosts withheld, port 3389	NL, RootLayer Web Services	Feed tag plus a shared Windows RDP hostname certificate	Watchlist
Remcos	9 hosts withheld; ports 3389, 2404 (4) and 9091 (5)	US, Interserver, Inc	Port 3389 rows share one Windows RDP hostname certificate	Watchlist

A shared RDP hostname certificate points to a cloned Windows VPS image, not to RAT software. That is why those RedLine and Remcos rows stay on watchlists.

05

CHAPTER 05

The Port-755 Open Directory

IN SHORT

One open HTTP directory on port 755 lists 40 files with Telegram-botnet and flood-tool names. We hashed the files but did not open them, so the host stays on a watchlist.

INSIDE THIS CHAPTER

What the directory held 18

Hypothesis: Telegram Bot API as C2 19

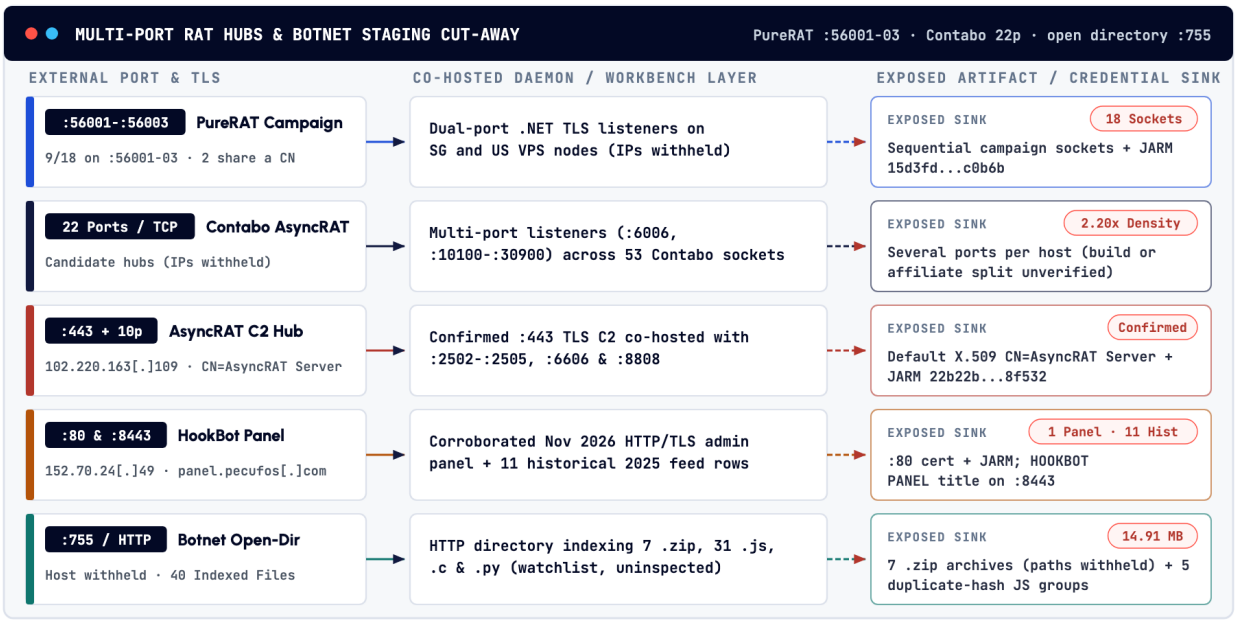
Fact: five groups of identical files 19

05 The Port-755 Open Directory

FIGURE 4 · MULTI-PORT RAT & OPEN-DIRECTORY SERVER BLUEPRINT ODIN BY BRANDEFENSE · TLP:CLEAR

Operators consolidate up to 22 RAT ports per VPS and list 40 indexed flood-themed files on port 755

Architectural cut-away of PureRAT :56001-:56003, Contabo AsyncRAT hubs, HookBot :80/:8443, and the :755 open directory



Scope: Observed by Odin by Brandefense, two weeks ending 18 Nov 2026 (33 unique SHA-256s on :755).
What this does not show: Slices represent distinct observed server archetypes rather than a single shared physical machine.

Figure 4: Operators consolidate up to 22 RAT ports per VPS and list 40 indexed flood-themed files on port 755

What the directory held

The directory listed 40 files totaling 14.91 MB: 7 `.zip` archives, 31 `.js` files, 1 `.c` file and 1 `.py` file. Thirty-nine were stored and hashed. One 158,720-byte entry was listed but not stored.

We recorded names, sizes and whole-file SHA-256 hashes. We did not unpack the archives or read the scripts. Hashes alone cannot show whether the server belongs to an attacker, a third-party mirror, a lab or a honeypot. So the host address and all 40 paths are withheld.

File (described by its name)	Size (bytes)	Note
Telegram DDoS-named archive (`.zip`)	616,510	Not unpacked
Telegram botnet-named archive (`.zip`)	2,198	Not unpacked
Bot-counter-named archive (`.zip`)	2,533	Not unpacked
File-host stresser-named archive (`.zip`)	1,091,073	Not unpacked
Encoded layer-7 DDoS-named archive (`.zip`)	965,898	Not unpacked; two related archives also present
Two `dstat` -named scripts (`.js`)	83,968 and 34,816	Names suggest a flood meter
Two HTTP flood-named scripts (`.js`)	46,080 and 34,816	Names suggest HTTP flooders
Flood-named `.c` and `.py` sources	6,011 and 5,256	Names suggest layer-4 or game-server flooders

None of the 40 paths contained a literal `api.telegram[.]org` string.

Hypothesis: Telegram Bot API as C2

Four of the seven archives have bot or Telegram names. The other three have DDoS stresser names. The 31 `.js` files carry flood and bypass themes.

Public reporting on Telegram-controlled kits describes a two-way pattern (ATT&CK T1102.002, bidirectional communication). A bot polls `GET /bot<token>/getUpdates` on `api.telegram[.]org` and returns output with `POST /bot<token>/sendMessage`. This differs from a dead-drop resolver (T1102.001).

For this directory, Telegram polling, a Node.js runtime and HTTP/2 flood behavior are hypotheses drawn from file names. Defenders should still watch which processes talk to `api.telegram[.]org`.

Fact: five groups of identical files

Comparing hashes across the 39 stored files shows five groups of byte-identical `.js` files saved under different names. The groups contain 11 files in total.

Group	Files	Size (bytes) each	Name pattern
1	2	1,363,148	Base name and a copy with a `(100$)` price suffix; identical content, hash withheld
2	2	873,472	AI-themed labels
3	2	34,816	HTTP and HTTP/2 fast-flood labels
4	2	34,816	Same script in two directories
5	3	204,800	HTTP and HTTPS raw-flood labels

The identical hashes show duplication under price-tagged and AI-themed names. Whether anyone sold the script, or paid USD 100 for it, is an inference from file names only.

CHAPTER 06

What to Block and What to Watch

IN SHORT

Internet-wide RAT data mixes real panels with proxy defaults, cloned Windows templates, old feed tags and home broadband addresses. Only six hosts carry enough independent signals to block.

INSIDE THIS CHAPTER

The six hosts to block 21

What not to block 22

Other explanations we could not rule out 22

Telegram detections need scope 22

06 What to Block and What to Watch

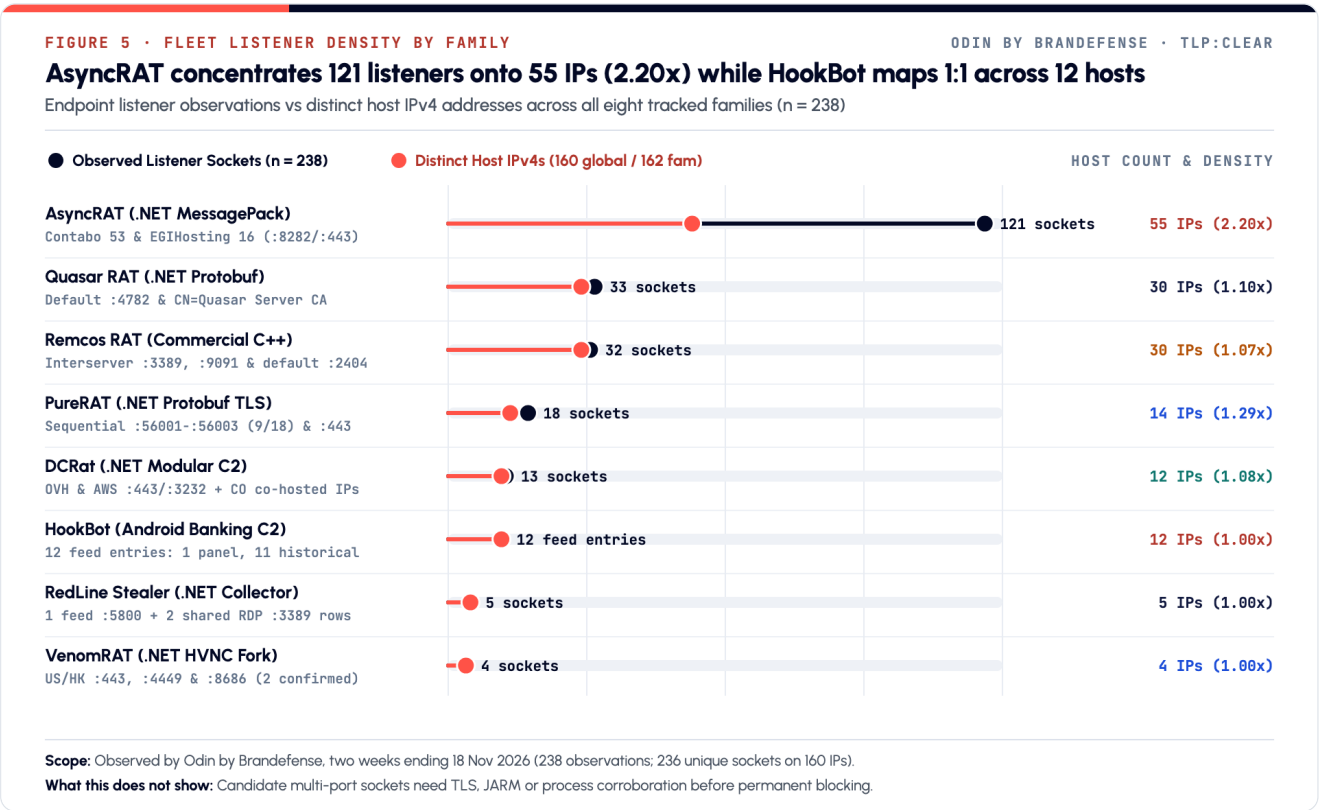


Figure 5: AsyncRAT concentrates 121 listeners onto 55 IPs (2.20x) while HookBot maps 1:1 across 12 hosts

The six hosts to block

Each of these shows a RAT or panel certificate plus a JARM fingerprint, or a panel certificate plus feed-reported panel markers.

Host	Family	Main signals
152.70.24[.]49	HookBot	`CN=panel.pecufos[.]com`, JARM `1dd40d...11926e`, `HOOKBOT PANEL` title
102.220.163[.]109	AsyncRAT	`CN=AsyncRAT Server`, JARM `22b22b...8f532`, 10 more high ports
141.98.157[.]101	Quasar	`CN=Quasar Server CA`, JARM `2ad2ad...3585be`, seen twice
148.113.214[.]176	DCRat	Spoofed download-manager wildcard certificate, ports 443 and 80
5.161.43[.]175	VenomRAT	`CN=akashicsymbiosis[.]ai`, JARM `27d40d...`
154.12.90[.]49	VenomRAT	`CN=dgtwxj[.]com`, JARM `3fd3fd...`

What not to block

Group	Why it is not a block	Action
11 older HookBot entries	First-seen dates from 2025 and no fresh panel marker; addresses may have changed hands	Watchlist until the panel is seen again
Port 22 on five HookBot hosts	Ordinary OpenSSH administration on rented Linux VPS	Ignore
Uncorroborated feed rows, such as RedLine on port 5800	Feed tag only	Watchlist
RDP-template rows (RedLine and Remcos on 3389)	Shared Windows hostname certificate from a cloned VPS image	Watchlist
Several suspected AsyncRAT rows on Alibaba Cloud	A commercial cloud AI gateway certificate on port 443 collides with the pattern	Watchlist
Dynamic or cable ISP addresses (Colombia, Canada)	Addresses rotate between customers	Short watchlist
Contabo, EGIHosting and PureRAT candidates	Port and JARM only, with no application handshake	Watchlist
The port-755 directory	Contents and ownership unverified	Watchlist

Other explanations we could not rule out

The three populations share no address, certificate or operator link. So evidence from one cannot clear or condemn another. We weighed each one on its own.

- **Multi-port RAT candidates.** For the five desktop hosts to block, RAT certificates and JARM values support a C2 listener. The 14 to 22 port Contabo hosts, the 16 EGIHosting hosts and the 18 PureRAT sockets lack an application-layer handshake. A researcher honeypot, vendor sinkhole or lab could look the same.
- **The port-755 directory.** Without content or operator data, it could be a third-party repository, an academic malware mirror or a deception honeypot. Hunt by its file-name themes instead of blocking the host; its file hashes are withheld.
- **Older HookBot entries.** Their first-seen dates show when they entered the feed, not that panels ran in November 2026. VPS reassignment or harmless reuse of a Traefik proxy stays possible. Five of the 11 show the Traefik default certificate.

Telegram detections need scope

Many DevOps and SOC teams send real webhook alerts to `api.telegram[.]org/bot...`. Staff may also run the Telegram desktop client from `C:\Program Files`. The KQL and Sigma rules therefore exclude named approved automation hosts and the official client in `C:\Program Files`.

They fire when a script interpreter, or a binary in `%APPDATA%`, `%TEMP%` or `%PROGRAMDATA%`, connects to `api.telegram[.]org`. The interpreters are `powershell.exe`, `pwsh.exe`, `wscript.exe`, `cscript.exe`, `mshta.exe`, `curl.exe`, `node.exe` and `python.exe`. Only the KQL query also covers `RegAsm.exe`, `InstallUtil.exe`, `MSBuild.exe` and `/bot` command lines.

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

- [ ] Block the six hosts in Chapter 6 on perimeter firewalls, mobile VPN gateways, DNS and proxy logs.
- [ ] Put the 10 withheld watchlist entries and other candidate hosts on SIEM watchlists that need a certificate, JARM or endpoint signal before action.
- [ ] Re-check certificate or banner state before keeping any VPS block past 90 days, since addresses get reassigned.

Within 72 hours

- [ ] Enforce Google Play Integrity attestation. Block banking transfers while an untrusted `AccessibilityService` or `MediaProjection` stream is active. Disable APK sideloading on managed Android devices.
- [ ] Restrict outbound `api.telegram[.]org` traffic, through proxy or EDR policy, to designated webhook servers and the official Telegram client.
- [ ] Review any host where a script or a user-writable binary calls `/bot`. Isolate it if its owner and purpose cannot be confirmed, or if other malicious behavior appears.

Within 7 to 30 days

- [ ] Deploy Suricata SIDs 9515001 to 9515007, keeping 9515003 and 9515007 as hunts.
- [ ] Enable Attack Surface Reduction rules and the Sigma rules for `%APPDATA%` or `%TEMP%` loaders that start `RegAsm.exe`, `InstallUtil.exe`, `MSBuild.exe` or `vbc.exe`, or create `schtasks.exe` tasks.
- [ ] Turn on HTTP/2 Rapid Reset rate limits, TLS or JA4-based bot challenges and upstream DDoS scrubbing against commodity layer-7 flood tools.
- [ ] Hunt for Telegram DDoS, botnet and stresser archive names in EDR file-write data and mail and web gateway archive scanners.

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 26

How the KQL query works 27

How the Sigma and YARA rules work 27

How to sort a host 28

08 Detection Rules

The full rules are in Appendix B. Each file targets a signal that survives a change of IP address.

Rule file	What it catches
`odin_hookbot_purerat_asyncrat_telegram.rules`	Suricata 7: seven signatures for the HookBot panel, PureRAT high-port handshakes, AsyncRAT and Quasar default certificates, and non-browser Telegram Bot API calls
`sentinel_rat_beacons_and_telegram_c2_egress.kql`	Microsoft Sentinel and Defender XDR: egress to the six blocked hosts, RAT-port sockets from suspicious processes, unapproved Telegram API calls, and scheduled-task persistence
`sigma_windows_dotnet_rat_and_telegram_bot_c2.yml`	Sigma (3 rules): .NET framework utilities started from user-writable folders, unapproved Telegram API connections, and a Linux flood-tool hunt
`odin_hookbot_purerat_and_telegram_botnet_kits.yar`	YARA (4 rules): HookBot DEX and panel artifacts, AsyncRAT and VenomRAT stubs, PureRAT loaders, and Telegram botnet kits

How the Suricata signatures work

SID	Looks for	Why
9515001	`<title>HOOKBOT PANEL</title>` in an HTTP response	Catches the HookBot panel page (T1417.002)
9515002	TLS certificate subject `CN=panel.pecufos[.]com` from the server	Catches the corroborated panel's certificate
9515003	Any TLS server certificate from ports 56001 to 56003, once per source every 300 seconds	Hunt for PureRAT sequential ports; pair with endpoint process data (T1219)
9515004	Certificate subject `CN=AsyncRAT Server` on any port	Catches the AsyncRAT default certificate
9515005	Certificate subject `CN=Quasar Server CA` on any port	Catches the Quasar default certificate
9515006	Outbound request to `api.telegram[.]org` with a `/bot<token>/sendMessage`, `sendDocument` or `getUpdates` path and a script user agent	Catches non-browser Telegram C2 calls (T1102.002)
9515007	Outbound request to `api.telegram[.]org/file/bot<token>/` with a script user agent	Hunt for payloads pulled through the Bot API file endpoint

The script user agents are `python-requests`, `Go-http-client`, `curl`, `node-fetch`, `axios`, `PowerShell` and `Mozilla/4.0`.

Two limits apply. First, certificate rules 9515002 to 9515005 read the certificate in clear only in TLS 1.2 and earlier. TLS 1.3 encrypts the server certificate (RFC 8446), so TLS 1.3 needs TLS inspection, active JARM checks or endpoint data. Second, 9515006 and 9515007 read HTTPS URIs and need TLS inspection too. Exempt only named webhook gateway addresses from them, never whole workstation subnets.

How the KQL query works

The query looks back 14 days over `DeviceNetworkEvents` and `DeviceProcessEvents` and has four branches.

Branch	Fires when
1. Blocked-host egress	A device connects to one of the six hosts in Chapter 6
2. RAT-port sockets	A .NET framework binary (`RegAsm.exe`, `InstallUtil.exe`, `MSBuild.exe`, `vbc.exe`, `AppLaunch.exe`, `cvtres.exe`) or a user-writable binary connects to a public IP on a commodity RAT port; ports 56001 to 56003 get their own label
3. Telegram API	An unapproved process calls `api.telegram[.]org`, judged by interpreter name, .NET binary, user-writable path or a `/bot` command line
4. Scheduled tasks	`schtasks.exe /create` points at `AppData`, `ProgramData`, `wscript` or `powershell`

The port list holds 17 values, including 2404, 4782, 8282, 8808, 10100 and 56001 to 56003. Watchlist hosts are withheld, so branch 2 finds them by behavior rather than by IP. Fill in your own approved Telegram hosts before use.

These events show a suspicious launch or socket. They do not show in-memory process hollowing; that needs EDR injection data. If developers run portable tools from `%LOCALAPPDATA%\Programs\`, add host-plus-path exclusions or check file signatures, rather than excluding the whole folder.

How the Sigma and YARA rules work

Rule	Fires when
Sigma 1	A binary in `%APPDATA%`, `%TEMP%` or `%PROGRAMDATA%` starts a .NET framework utility, or creates `schtasks.exe /create` persistence
Sigma 2	A script or user-writable process connects to `api.telegram[.]org`, outside approved hosts and the official client
Sigma 3 (hunt)	A Linux `node`, `python` or `curl` process calls `api.telegram[.]org/bot`, or runs from `/tmp/`, `/var/tmp/` or `/dev/shm/` with HTTP/2 flood and proxy-list arguments
`ODIN_Android_HookBot_Overlay_And_VNC_Dropper`	Strings in an extracted Android `classes.dex` file or captured panel HTML; unpack the `.apk` first
`ODIN_DotNet_AsyncRAT_VenomRAT_MessagePack_Stub`	.NET MessagePack RAT stub strings
`ODIN_DotNet_PureRAT_Sequential_Port_Loader` (hunt)	`SslStream` with `ProtoContract` or `ProtoMember` markers, plus `GZipStream` config code or the 56001 to 56003 port strings
`ODIN_Telegram_BotAPI_DDoS_And_Botnet_Builder_Kit` (hunt)	`api.telegram[.]org/bot` methods with Node.js `http2.connect` or `tls.connect` calls

The YARA kit rule does not cover the uninspected port-755 files.

How to sort a host

The triage logic never blocks on uncorroborated evidence.

What the host shows	Action
Confirmed AsyncRAT, or HookBot with two or more independent panel signals	Block
HookBot feed entry with only one panel marker	Analyst review
Older HookBot feed entry	Watchlist
Multi-port RAT candidate cluster	Watchlist
Feed-tagged desktop RAT row with only a port, or only a certificate	Watchlist
Suspected RAT on a dynamic ISP	Short watchlist
Uninspected open directory	Watch only
Telegram Bot API call from an unlisted `%APPDATA%` script, nothing else	Review before isolating
The same call plus another malicious behavior	Treat as a Telegram C2 beacon
Port-only SSH match	Ignore
Approved Telegram webhook server	Allow

09

CHAPTER 09

Indicators

IN SHORT

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

INSIDE THIS CHAPTER

Hosts to block 30
Watchlist hosts 30
File names for hunting 30
MITRE ATT&CK mapping 31

09 Indicators

Appendix A holds the 6 published hosts. The 10 watchlist hosts and 12 file hashes are withheld.

Type	Count
Hosts to block (IPv4)	6
Watchlist hosts (withheld)	10
Withheld file hashes from the port-755 directory (12 of 33 unique)	12

Supporting fingerprints: the JARM values and certificate subjects named in Chapters 3, 4 and 6.

Hosts to block

Host	Port	Family	Country, hosting	Key fingerprint
152.70.24[.]49	80	HookBot	DE, Oracle	`CN=panel.pecufos[.]com`, JARM `1dd40d...11926e`
102.220.163[.]109	443	AsyncRAT	SI, VPS Dedicated LLC	`CN=AsyncRAT Server`, 10 more high ports
141.98.157[.]101	2288	Quasar	GB, EUGAMEHOST	`CN=Quasar Server CA`, seen twice
154.12.90[.]49	443	VenomRAT	HK, cognetcloud INC	`CN=dgtwxj[.]com`, JARM `3fd3fd...`
148.113.214[.]176	443	DCRat	CA, OVH	Spoofed download-manager certificate, also port 80
5.161.43[.]175	443	VenomRAT	US, Hetzner	`CN=akashicsymbiosis[.]ai`, JARM `27d40d...`

Watchlist hosts

The 10 watchlist entries are withheld. They cover five of the 11 older HookBot entries, the RedLine port-5800 feed row, the port-755 directory, a PureRAT dual-port candidate, the 22-port Contabo AsyncRAT hub and a Colombian dynamic ISP host. The two RedLine RDP-template hosts on port 3389 are also watched but are not among the 28 indicators in this chapter (6 hosts, 10 watchlist entries and 12 hashes).

File names for hunting

Twelve of the directory's 33 unique file hashes were selected as hunting indicators, and all 12 are withheld. The files were not opened and the host's owner is not established. Hunt by the file-name themes in Chapter 5 instead: Telegram DDoS and botnet archives, stresser and layer-4 or layer-7 flood scripts, and `dstat` meters.

Withheld: the 10 watchlist addresses, the two RDP-template hosts, all candidate and suspected addresses, custom certificate names on non-confirmed hosts, the EGIHosting subnet, the port-755 host and its paths, and all Telegram bot tokens.

MITRE ATT&CK mapping

Tactic	Technique	How it appears	Detection
Credential Access (Mobile)	T1417.002 Input Capture: GUI Input Capture	HookBot WebView overlays and PIN or 2FA capture	SID 9515001, 9515002; HookBot YARA rule
Impact / Defense Evasion (Mobile)	T1516 Input Injection	HookBot Accessibility abuse and `dispatchGesture` replay over VNC	Play Integrity and Accessibility monitoring
Command and Control	T1219 Remote Access Tools	AsyncRAT, Quasar, Remcos, PureRAT, DCRat and VenomRAT listeners	SID 9515003 to 9515005; KQL branches 1 and 2
Command and Control	T1095 Non-Application Layer Protocol	MessagePack and protobuf framing over raw TLS sockets	SID 9515003; PureRAT and AsyncRAT YARA rules
Command and Control	T1102.002 Web Service: Bidirectional Communication	Telegram Bot API polling and replies, as hypothesized for the kits	SID 9515006, 9515007; KQL branch 3; Sigma 2 and 3

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 findings cover a two-week observation window ending 18 November 2026. Some records carry earlier first-seen dates, as far back as 2023 for a few candidates and 2025 for the older HookBot entries. These are earliest first-seen dates, not a tracking start.
- **Panels, not victims.** We saw servers, certificates and listings. We saw no victim device, no infection and no financial loss.
- **Feed records.** The HookBot feed entries, the port 8443 title and the eight fingerprint matches on 152.70.24[.]49 come from feed records. The certificate and JARM on port 80 were observed directly.
- **Candidates are leads.** Multi-port candidate hosts rest on port and JARM data alone. Do not block them without a second signal.
- **The open directory is unread.** File names suggest Telegram botnets and flood tools. We did not open the files, and we do not know who runs the server.
- **The role model is a model.** Nothing links the three populations to each other.
- **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 Mobile. T1417.002 GUI Input Capture and T1516 Input Injection. https://attack.mitre.org/techniques/T1417/002/ (verified 19 November 2026)
2. Android Developers. `AccessibilityService` reference. https://developer.android.com/reference/android/accessibilityservice/AccessibilityService (verified 19 November 2026)
3. MITRE ATT&CK Enterprise. T1219 Remote Access Tools and T1095 Non-Application Layer Protocol. https://attack.mitre.org/techniques/T1219/ (verified 19 November 2026)
4. MITRE ATT&CK Enterprise. T1102.002 Web Service: Bidirectional Communication. https://attack.mitre.org/techniques/T1102/002/ (verified 19 November 2026)
5. Telegram. Bot API documentation. https://core.telegram.org/bots/api (verified 19 November 2026)
6. Salesforce Engineering. Easily identify malicious servers on the internet with JARM. https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a/ (verified 19 November 2026)
7. CSIRT KNF. HookBot, a new Android banking trojan. https://cebrf.knf.gov.pl/images/HOOKBOT_CSIRT_KNF_ENG.pdf (verified 19 November 2026)
8. Huntress. PureRAT threat actor evolution. https://www.huntress.com/blog/purerat-threat-actor-evolution (verified 19 November 2026)
9. Cyderes Howler Cell. Job offer attack chains analysis. https://www.cyderes.com/howler-cell/job-offer-attack-chains-analysis (verified 19 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 (6) 37

B. Detection rules (4 files) 38

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
R15-IOC-01	IPv4	152.70.24[.]49	80	HookBot	BLOCK
R15-IOC-02	IPv4	102.220.163[.]109	443	AsyncRAT	BLOCK
R15-IOC-03	IPv4	141.98.157[.]101	2288	Quasar	BLOCK
R15-IOC-04	IPv4	154.12.90[.]49	443	VenomRAT	BLOCK
R15-IOC-05	IPv4	148.113.214[.]176	443	DCRat	BLOCK
R15-IOC-06	IPv4	5.161.43[.]175	443	VenomRAT	BLOCK

B Detection Rules

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

odin_hookbot_purerat_and_telegram_botnet_kits.yar

YARA

```
/*
 * Odin by Brandefense - Report #15 YARA Detection & Exploratory Hunting Suite
 * Classification: TLP:CLEAR
 * Scope: 4 YARA rules covering extracted Android HookBot DEX payloads and panel artifacts,
 *        AsyncRAT/VenomRAT MessagePack .NET stubs, and exploratory hunting rules for
 *        PureRAT/PureHVNC protobuf-over-SslStream loaders (Huntress; Cyderes Howler Cell) and generic
 *        Telegram Bot API / Node.js HTTP/2 flood scripts (MITRE ATT&CK T1102.002; Telegram Bot API
documentation).
 */

rule ODIN_Android_HookBot_Overlay_And_VNC_Dropper
{
    meta:
        author = "Odin by Brandefense"
        date = "2026-11-18"
        classification = "TLP:CLEAR"
        description = "Detects extracted Android HookBot raw DEX payloads (classes.dex) or captured panel HTML/
config artifacts combining AccessibilityService abuse with WebView overlay and VNC screen streaming (unpack APK
ZIP containers before scanning)."
        mitre_attack = "T1417.002, T1516"
    strings:
        $dex_hdr = { 64 65 78 0A 30 33 }
        $acc_svc = "android.permission.BIND_ACCESSIBILITY_SERVICE" ascii wide
        $proj = "MediaProjectionManager" ascii wide
        $ovl1 = "overlay_inject" ascii wide nocase
        $ovl2 = "performGlobalAction" ascii wide
        $ovl3 = "dispatchGesture" ascii wide
        $c2_1 = "HOOKBOT PANEL" ascii wide nocase
        $c2_2 = "panel.pecufos.com" ascii wide nocase
        $c2_3 = "sms_intercept" ascii wide nocase
    condition:
        filesize < 35MB and (
            ($dex_hdr at 0 and $acc_svc and ($proj or $ovl3) and 1 of ($ovl1, $ovl2, $c2_1, $c2_2, $c2_3))
            or (2 of ($c2_1, $c2_2, $ovl1, $c2_3))
        )
}

rule ODIN_DotNet_AsyncRAT_VenomRAT_MessagePack_Stub
{
    meta:
        author = "Odin by Brandefense"
        date = "2026-11-18"
        classification = "TLP:CLEAR"
        description = "Detects .NET AsyncRAT, VenomRAT, and DCRat client stubs using MessagePack serialization
and encrypted certificate/port configuration blocks."
        mitre_attack = "T1219"
    strings:
        $mz = { 4D 5A }
        $dotnet = "mscoree.dll" ascii
        $mp1 = "MessagePackLib.MessagePack" ascii wide
        $mp2 = "MsgPack" ascii wide
        $cn1 = "AsyncRAT Server" ascii wide
        $cn2 = "DcRat" ascii wide
        $cfg1 = "ServerCertificate" ascii wide
        $cfg2 = "Pastebin" ascii wide
        $cfg3 = "Anti_Process" ascii wide
        $hvnc = "HVNC" ascii wide
    condition:
        filesize < 15MB and (
            ($mz at 0 and $dotnet and 1 of ($mp1, $mp2) and 2 of ($cn1, $cn2, $cfg1, $cfg2, $cfg3, $hvnc))
```

```
            or ($cn1 and $mp1 and $cfg1)
        )
}

rule ODIN_DotNet_PureRAT_Sequential_Port_Loader
{
    meta:
        author = "Odin by Brandefense"
        date = "2026-11-18"
        classification = "TLP:CLEAR"
        deployment_tier = "exploratory_hunt"
        description = "Exploratory hunting rule for .NET PureRAT / PureHVNC client loaders based on published
family analysis (length-prefixed protobuf-net C2 serialization over SslStream TLS and GZip-compressed protobuf
configuration blocks; Huntress, Cyderes Howler Cell) and sequential 56001-56003 port clusters."
        mitre_attack = "T1219"
    strings:
        $mz = { 4D 5A }
        $dotnet = "mscoree.dll" ascii
        $s_ssl = "SslStream" ascii wide
        $s_proto1 = "protobuf-net" ascii wide
        $s_proto2 = "ProtoContract" ascii wide
        $s_proto3 = "ProtoMember" ascii wide
        $s_gzip = "GZipStream" ascii wide
        $p1 = "PureRAT" ascii wide nocase
        $p2 = "PureClient" ascii wide nocase
        $p3 = "PureHVNC" ascii wide nocase
        $port_seq = /56001[,;|:\s]+56002|56002[,;|:\s]+56003/ ascii wide
    condition:
        filesize < 20MB and (
            ($mz at 0 and $dotnet and $s_ssl and 1 of ($s_proto1, $s_proto2, $s_proto3) and (1 of ($p1, $p2, $p3,
$s_gzip) or $port_seq))
            or (1 of ($p1, $p2, $p3) and $port_seq)
        )
}

rule ODIN_Telegram_BotAPI_DDoS_And_Botnet_Builder_Kit
{
    meta:
        author = "Odin by Brandefense"
        date = "2026-11-18"
        classification = "TLP:CLEAR"
        deployment_tier = "exploratory_hunt"
        description = "Exploratory hunting rule for generic Telegram Bot API (T1102.002) and Node.js HTTP/2 flood
scripts based on public tradecraft (MITRE ATT&CK T1102.002; Telegram Bot API documentation); does not claim
content coverage of the uninspected port-755 files."
        mitre_attack = "T1102.002, T1498.001"
    strings:
        $tg_api = "api.telegram.org/bot" ascii wide nocase
        $tg_m1 = "/sendMessage" ascii wide
        $tg_m2 = "/getUpdates" ascii wide
        $tg_m3 = "/sendDocument" ascii wide
        $h2_1 = "require('http2')" ascii wide
        $h2_2 = "require(\"http2\")" ascii wide
        $h2_3 = "http2.connect(" ascii wide
        $h2_4 = ":path" ascii wide
        $h2_5 = "tls.connect(" ascii wide
    condition:
        filesize < 10MB and (
            ($tg_api and 2 of ($tg_m1, $tg_m2, $tg_m3))
            or ($tg_api and 1 of ($tg_m1, $tg_m2, $tg_m3) and 1 of ($h2_1, $h2_2, $h2_3, $h2_5))
            or ($h2_3 and 1 of ($h2_1, $h2_2) and $h2_4 and $h2_5)
        )
}
```

odin_hookbot_purerat_asyncrat_telegram.rules

SURICATA

```
# Odin by Brandefense - Report #15 Suricata 7.x Mobile HookBot, Multi-Port .NET RAT & Telegram C2 Suite
# Classification: TLP:CLEAR
# Scope Note: Covers HookBot Android panel HTTP title and TLS certificate markers, PureRAT sequential
#             high-port TLS certificate handshakes (exploratory hunt), AsyncRAT & Quasar default TLS
#             certificates (passive TLS 1.2 or decrypted/probed TLS 1.3), and
#             non-browser Telegram Bot API C2 / file-endpoint retrieval.

# 1. HookBot Android Banking C2 Panel HTTP Title Marker
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-MOBILE HookBot Android C2 Panel HTTP Title Detected";
flow:established,from_server; http.response_body; content:"HOOKBOT PANEL"; nocase; fast_pattern; pcre:"/
<title>\s*HOOKBOT PANEL\s*<\/title>/i"; metadata:created_at 2026_11_18, mitre_technique_id T1417_002;
classtype:trojan-activity; sid:9515001; rev:1;)

# 2. HookBot Corroborated C2 Panel TLS Server Certificate (CN=panel.pecufos.com; passive TLS 1.2 or decrypted TLS
1.3)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-MOBILE HookBot Corroborated C2 TLS Certificate
CN=panel.pecufos.com"; flow:established,from_server; tls.cert_subject; content:"CN=panel.pecufos.com"; nocase;
fast_pattern; metadata:created_at 2026_11_18, mitre_technique_id T1417_002; classtype:trojan-activity; sid:
9515002; rev:1;)

# 3. Exploratory Hunt: PureRAT Sequential High-Port (:56001-:56003) TLS Server Certificate Handshake (passive TLS
1.2 or decrypted TLS 1.3)
alert tls $EXTERNAL_NET 56001:56003 -> $HOME_NET any (msg:"ODIN-RAT HUNT PureRAT Sequential High-Port 56001-56003
TLS Server Certificate Handshake"; flow:established,from_server; tls.cert_subject; content:"CN="; fast_pattern;
threshold:type limit, track by_src, count 1, seconds 300; metadata:created_at 2026_11_18, deployment_tier
exploratory_hunt, mitre_technique_id T1219; classtype:trojan-activity; sid:9515003; rev:2;)

# 4. AsyncRAT Default Self-Signed TLS Server Certificate (CN=AsyncRAT Server; passive TLS 1.2 or decrypted TLS
1.3)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-RAT AsyncRAT Default TLS Certificate CN=AsyncRAT Server";
flow:established,from_server; tls.cert_subject; content:"CN=AsyncRAT Server"; fast_pattern; metadata:created_at
2026_11_18, mitre_technique_id T1219; classtype:trojan-activity; sid:9515004; rev:1;)

# 5. Quasar RAT Default TLS Certificate Authority (CN=Quasar Server CA; passive TLS 1.2 or decrypted TLS 1.3)
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"ODIN-RAT Quasar RAT Default TLS Certificate CN=Quasar Server
CA"; flow:established,from_server; tls.cert_subject; content:"CN=Quasar Server CA"; fast_pattern;
metadata:created_at 2026_11_18, mitre_technique_id T1219; classtype:trojan-activity; sid:9515005; rev:1;)

# 6. Suspicious Non-Browser Script or Loader Calling Telegram Bot API C2 ( behind TLS inspection )
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BOTNET Non-Browser Loader Calling Telegram Bot API C2";
flow:established,to_server; http.host; content:"api.telegram.org"; fast_pattern; http.uri; content:"/bot";
startswith; pcre:"/^\/bot[0-9]{8,11}:[A-Za-z0-9_-]{30,}\/(sendMessage|sendDocument|getUpdates)/";
http.user_agent; pcre:"/^(python-requests|Go-http-client|curl|node-fetch|axios|PowerShell|Mozilla\/4\.0)/i";
metadata:created_at 2026_11_18, mitre_technique_id T1102_002; classtype:command-and-control; sid:9515006; rev:1;)

# 7. Exploratory Hunt: Non-Browser Script Retrieving Staged Payload via Telegram Bot API File Endpoint ( behind
TLS inspection )
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ODIN-BOTNET HUNT Non-Browser Loader Retrieving Telegram Bot
API Staged File"; flow:established,to_server; http.host; content:"api.telegram.org"; fast_pattern; http.uri;
content:"/file/bot"; startswith; pcre:"/^\/file\/bot[0-9]{8,11}:[A-Za-z0-9_-]{30,}\//"; http.user_agent; pcre:"/
^(python-requests|Go-http-client|curl|node-fetch|axios|PowerShell|Mozilla\/4\.0)/i"; metadata:created_at
2026_11_18, deployment_tier exploratory_hunt, mitre_technique_id T1102_002; classtype:trojan-activity; sid:
9515007; rev:2;)
```

sentinel_rat_beacons_and_telegram_c2_egress.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Report #15 Microsoft Sentinel / Defender XDR Hunting Query
// Classification: TLP:CLEAR
// Hunts confirmed HookBot/RAT C2 egress (6 multi-signal confirmed IPs), sequential PureRAT (:56001-:56003) &
// multi-port AsyncRAT sockets, suspicious .NET framework binary outbound sockets, and non-browser
api.telegram.org calls.
let Lookback = 14d;
let ConfirmedC2IPs = dynamic([
    "152.70.24.49", "102.220.163.109", "141.98.157.101",
```

```
    "154.12.90.49", "148.113.214.176", "5.161.43.175"
]);
let CommodityRatPorts = dynamic([
    2404, 3232, 4449, 4782, 5800, 6606, 7707, 8282, 8686, 8808, 9091,
    10100, 20100, 30100, 56001, 56002, 56003
]);
let DotNetFrameworkBins = dynamic(["regasm.exe", "installutil.exe", "msbuild.exe", "vbc.exe", "applaunch.exe",
"cvtres.exe"]);
let TelegramScriptInterpreters = dynamic([
    "powershell.exe", "pwsh.exe", "wscript.exe", "cscript.exe", "mshta.exe",
    "curl.exe", "node.exe", "python.exe", "pythonw.exe"
]);
let UserWritablePathRegex = @"(?i)\\(AppData\\(Local\\(Temp|Programs)|Roaming)|ProgramData)\\";
let SchtasksPersistenceRegex = @"(?i)(\\AppData\\Roaming\\|\\AppData\\Local\\Temp\\|\\ProgramData\\|wscript|
powershell)";
let ApprovedTelegramHosts = dynamic(["SRV-SOC-WEBHOOK01", "SRV-DEVOPS-NOTIFY01"]);
// Exact executable paths of approved Telegram and browser clients (lower case); adjust to your fleet.
let ApprovedTelegramClientPaths = dynamic([
    @"c:\program files\telegram desktop\telegram.exe",
    @"c:\program files\google\chrome\application\chrome.exe",
    @"c:\program files (x86)\microsoft\edge\application\msedge.exe",
    @"c:\program files\mozilla firefox\firefox.exe",
    @"c:\program files\bravesoftware\brave-browser\application\brave.exe"
]);
union isfuzzy=true
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback) and ActionType == "ConnectionSuccess"
    | where RemoteIP in (ConfirmedC2IPs)
    | extend HuntSignal = "Confirmed_HookBot_Or_Desktop_RAT_C2_Egress",
             ArtifactDetail = strcat(RemoteIP, ":", tostring(RemotePort), " (", InitiatingProcessFolderPath, ")")
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName = InitiatingProcessFileName,
ArtifactDetail
),
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback) and ActionType == "ConnectionSuccess"
    | where RemoteIPType == "Public" and RemotePort in (CommodityRatPorts)
    | where InitiatingProcessFileName in~ (DotNetFrameworkBins)
        or InitiatingProcessFolderPath matches regex UserWritablePathRegex
    | extend HuntSignal = case(
             RemotePort between (56001 .. 56003), "PureRAT_Sequential_HighPort_56001_56003_Beacon",
             InitiatingProcessFileName in~ (DotNetFrameworkBins),
"DotNet_Framework_Binary_Suspicious_RAT_Socket",
             "UserWritable_Binary_Commodity_RAT_Port_Egress"
        ),
        ArtifactDetail = strcat(RemoteIP, ":", tostring(RemotePort), " <- ", InitiatingProcessCommandLine)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName = InitiatingProcessFileName,
ArtifactDetail
),
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback) and ActionType == "ConnectionSuccess"
    | where RemoteUrl contains "api.telegram.org" or InitiatingProcessCommandLine contains "api.telegram.org/bot"
    | where not(DeviceName in~ (ApprovedTelegramHosts))
    | extend ClientPath = tolower(iff(InitiatingProcessFolderPath endswith InitiatingProcessFileName,
InitiatingProcessFolderPath, strcat(InitiatingProcessFolderPath, @"\", InitiatingProcessFileName)))
    | where ClientPath !in (ApprovedTelegramClientPaths)  // exact paths, no folder prefix
    | where InitiatingProcessFileName in~ (TelegramScriptInterpreters)
        or InitiatingProcessFileName in~ (DotNetFrameworkBins)
        or InitiatingProcessFolderPath matches regex UserWritablePathRegex
        or InitiatingProcessCommandLine contains "/bot"
    | extend HuntSignal = "Unapproved_Process_Telegram_Bot_API_C2_Egress",
             ArtifactDetail = strcat(InitiatingProcessFolderPath, " | ", InitiatingProcessCommandLine, " -> ",
RemoteUrl)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName = InitiatingProcessFileName,
ArtifactDetail
),
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where FileName =~ "schtasks.exe" and ProcessCommandLine contains "/create"
```

```
    | where ProcessCommandLine matches regex SchtasksPersistenceRegex
    | extend HuntSignal = "Commodity_DotNet_RAT_ScheduledTask_Persistence",
             ArtifactDetail = ProcessCommandLine
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, FileName, ArtifactDetail
)
| summarize
    FirstSeen = min(Timestamp),
    LastSeen = max(Timestamp),
    EventCount = count(),
    SampleDetails = make_set(ArtifactDetail, 5)
    by DeviceName, HuntSignal, InitiatingProcessFileName, FileName
| order by EventCount desc
```

sigma_windows_dotnet_rat_and_telegram_bot_c2.yml

SIGMA

```
title: Suspicious .NET Framework Binary Launch from User-Writable Path or AppData Scheduled Task Persistence
id: 7f15a101-8c2d-4a91-b501-001500000001
status: experimental
description: Detects user-writable binaries spawning .NET framework utilities (RegAsm.exe, InstallUtil.exe,
MSBuild.exe, vbc.exe, AppLaunch.exe, frequently abused as RAT host processes) or registering scheduled tasks from
user-writable directories. Process-creation telemetry identifies suspicious launches rather than confirming in-
memory hollowing.
author: Odin by Brandefense
date: 2026/11/18
references:
  - https://attack.mitre.org/techniques/T1219/
tags:
  - attack.command_and_control
  - attack.t1219
  - attack.defense_evasion
  - attack.t1218.009
logsource:
  category: process_creation
  product: windows
detection:
  selection_dotnet_framework_spawn:
    ParentImage|contains:
      - '\AppData\Local\Temp\'
      - '\AppData\Roaming\'
      - '\AppData\Local\Programs\'
      - '\ProgramData\'
    Image|endswith:
      - '\RegAsm.exe'
      - '\InstallUtil.exe'
      - '\MSBuild.exe'
      - '\vbc.exe'
      - '\AppLaunch.exe'
  selection_rat_schtasks_flags:
    Image|endswith: '\schtasks.exe'
    CommandLine|contains|all:
      - '/create'
      - '/sc '
  selection_rat_schtasks_paths:
    CommandLine|contains:
      - '\AppData\Roaming\'
      - '\AppData\Local\Temp\'
      - '\ProgramData\'
  filter_legit_visualstudio:
    # Exact IDE executables only; adjust edition and year to your fleet.
    ParentImage:
      - 'C:\Program Files\Microsoft Visual Studio\2022\Enterprise\Common7\IDE\devenv.exe'
      - 'C:\Program Files\Microsoft Visual Studio\2022\Professional\Common7\IDE\devenv.exe'
      - 'C:\Program Files\Microsoft Visual Studio\2022\Community\Common7\IDE\devenv.exe'
  condition: (selection_dotnet_framework_spawn or (selection_rat_schtasks_flags and
selection_rat_schtasks_paths)) and not filter_legit_visualstudio
falsepositives:
  - Custom developer build scripts running out of ProgramData CI workspaces (exclude explicit runner service
paths)
```

```
  level: high
  ---
  title: User-Writable or Script Process Outbound Connection to Telegram Bot API
  id: 7f15a102-8c2d-4a91-b501-001500000002
  status: experimental
  description: Detects Windows script interpreters, LOLBins, or user-writable binaries outside approved automation
  hosts establishing TLS connections to api.telegram.org for bidirectional web-service C2 (T1102.002) or stealer
  exfiltration.
  author: Odin by Brandefense
  date: 2026/11/18
  references:
    - https://attack.mitre.org/techniques/T1102/002/
  tags:
    - attack.command_and_control
    - attack.t1102.002
  logsource:
    category: network_connection
    product: windows
  detection:
    selection_telegram_api:
      Initiated: 'true'
      DestinationHostname|endswith: 'api.telegram.org'
    selection_suspicious_origin:
      Image|contains:
        - '\AppData\Local\Temp\'
        - '\AppData\Roaming\'
        - '\AppData\Local\Programs\'
        - '\ProgramData\'
        - '\powershell.exe'
        - '\pwsh.exe'
        - '\wscript.exe'
        - '\cscript.exe'
        - '\mshta.exe'
        - '\curl.exe'
        - '\node.exe'
        - '\python.exe'
    filter_approved_hosts:
      Computer:
        - 'SRV-SOC-WEBHOOK01'
        - 'SRV-DEVOPS-NOTIFY01'
    filter_official_telegram_desktop:
      # Exact executable path only; add per-user installs you approve.
      Image: 'C:\Program Files\Telegram Desktop\Telegram.exe'
    condition: selection_telegram_api and selection_suspicious_origin and not filter_approved_hosts and not
  filter_official_telegram_desktop
  falsepositives:
    - Authorized DevOps notification scripts on additional automation hosts not yet listed in filter_approved_hosts
    - Portable tools in AppData\Local\Programs initiating Telegram notifications (correlate with process-creation
  Authenticode signature events)
  level: high
  ---
  title: Exploratory Hunt - Linux Script Runtime Invoking Telegram Bot API or HTTP/2 Flood Arguments
  id: 7f15a103-8c2d-4a91-b501-001500000003
  status: experimental
  description: Exploratory hunting rule for Linux script interpreters (node, python, curl) invoking Telegram Bot
  API endpoints or executing from temporary directories with proxy-list and HTTP/2 flood CLI arguments.
  author: Odin by Brandefense
  date: 2026/11/18
  references:
    - https://attack.mitre.org/techniques/T1498/001/
    - https://attack.mitre.org/techniques/T1102/002/
  tags:
    - attack.impact
    - attack.t1498.001
    - attack.command_and_control
    - attack.t1102.002
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_runtime:
      Image|endswith:
```

```
      - '/node'
      - '/nodejs'
      - '/python3'
      - '/python'
      - '/curl'
  selection_telegram_or_temp_origin:
    CommandLine|contains:
      - 'api.telegram.org/bot'
      - '/getUpdates'
      - '/sendMessage'
      - '/tmp/'
      - '/var/tmp/'
      - '/dev/shm/'
  selection_botnet_or_flood_flags:
    CommandLine|contains:
      - 'api.telegram.org/bot'
      - 'proxies.txt'
      - 'proxy.txt'
      - '--threads'
      - '--rps'
      - 'http2'
  condition: selection_runtime and selection_telegram_or_temp_origin and selection_botnet_or_flood_flags
falsepositives:
  - Authorized monitoring or notification scripts running from /tmp on non-hardened Linux hosts
level: medium
```

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/hookbot-panels-multi-port-rats-and-telegram-kits/
Short Read · brandefense.io/odin-research/blog/mobile-banking-panels-multi-port-rats-and-telegram-kits-share/
Defense Kit · brandefense.io/odin-research/defense-kit/