Episode Book

EPISODE #6426

Tracing How IP Cameras Phone Home in OPNsense

PUBLISHED	3 November 2026 at 16:56 UTC
DURATION	28:13
LISTEN	myweirdprompts.com/episode/opnsense-ip-camera-packet-tracing-64381/
HOSTS	Corn, Herman, Winston, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	20 cited by the research (listed at the end)
BOOK GENERATED	4 November 2026 at 09:31 UTC

SYNOPSIS

Smart cameras often maintain hidden cloud streaming paths even when configured for local RTSP feeds. Tracing these connections requires looking beyond basic firewall blocks to understand ingress and egress behavior. An investigation using OPNsense's built-in toolset—including packet captures, the Packet Filter state table, NetFlow traffic analysis, and Unbound DNS reporting—uncovers how devices circumvent local resolvers using hardcoded public IP addresses and persistent peer-to-peer relay connections.

TRANSCRIPT



Corn

The uncomfortable part isn't that the camera is talking to the internet. It's that you can see the camera, it's sitting on your shelf, you paid for it, and you cannot find the door it's using.



Herman

That's the whole thing in one line, yeah.



Corn

Daniel's prompt is exactly that scenario. He bought an IP camera that supports RTSP, got it into Frigate, but before that he set it up through the vendor app and could view it from outside his house. The manual has no off switch for the cloud path. He's on OPNsense, the camera has a static IP, and he's asking for the forensic version, not the firewall version. Trace the packets out, work out exactly how it's escaping, and only then lock it down.



Herman

I want to be clear about the shape of this, because it matters. This is an investigation episode. The lockdown is the last five minutes, and it's the easy part. The hard part is knowing what you're actually blocking.



Corn

This is the single-device version. One camera. One escape path. Follow it to ground.



Herman

So the premise, and this is the thing people get wrong before anything technical happens. An RTSP camera is still a cloud device. RTSP is the local streaming protocol. It's how Frigate pulls the feed. But the vendor app, the peer-to-peer relay behind it, the firmware that assumes your network is one flat happy LAN where everything can talk to everything — that's the product. The camera is not malfunctioning when it calls home. Calling home is the feature that made setup take ninety seconds instead of an afternoon.



Corn

That's why it's uncomfortable in a specific way. It's not a bug you can report. It's the product working.



Herman

Right. So the forensic posture. You're not trying to stop anything yet. The whole investigation is two questions. Did the packet reach the firewall at all? And did PF let it through? Everything else is instrumentation for those two questions. And OPNsense has all of it built in, no plugins.



Corn

Name them.



Herman

Packet capture, under Diagnostics. The firewall live log and the state table. NetFlow and Insight, under Reporting. And Unbound's DNS query reporting, also under Reporting. Four instruments, four different questions.



Corn

And there's one framing rule that governs the entire investigation, and it's the ingress and egress distinction.



Herman

OPNsense filters traffic on the interface it arrives at. So if you capture on the camera's VLAN interface, you're seeing ingress — the packet reached the firewall. Capture on WAN at the same time and you see egress — the firewall passed it. If it shows up on ingress and not on egress, PF dropped it. That's not a failed capture. That's a finding.



Corn

The camera tried, and your rule set caught it. That's an answer, not an error.



Herman

And the reverse is the disturbing one. Present on both interfaces, and now you know the door exists and it's open.



Corn

So walk me through the actual capture. You open Diagnostics, Packet capture — what am I looking at?



Herman

It's a GUI wrapper around tcpdump. You check a box for each interface you want, it starts a tcpdump process on each one. The options that matter: Host Address takes a source or destination IP, MAC, or a subnet in CIDR, and you can negate it with not. Count defaults to one hundred packets per interface — zero means unlimited. Packet Length zero means capture the whole frame.



Corn

One hundred packets sounds like a lot until it's a chatty camera.



Herman

Then you set zero. And the results you can view in the browser at three levels of detail, or download as a zip of pcaps plus a JSON of the options you chose, and open it in Wireshark. Which is where the real reading happens.



Corn

The trap, though. There's always a trap.



Herman

The interface name. The GUI says LAN, but underneath that is a FreeBSD driver name — em0, igb0, ix0, vtnet0, bge0. VLANs show up as things like igb0_vlan10 or em0.20. If you don't run ifconfig and confirm which name is which before you capture, you will happily capture on the wrong interface for an hour and conclude the camera is silent.



Corn

And then write a forum post about it.



Herman

Somebody almost certainly has. The CLI form is better for a long watch anyway. tcpdump, the n flag and i flag, then the interface, then your filter. The n flag is the one that matters here — it disables DNS and service-name resolution, so you see raw IPs and ports. Which is exactly what you want when the entire question is where is this going.



Corn

Resolving names during an investigation into where things resolve is a nice irony, though.



Herman

It would actively mislead you. Then the w flag writes to disk instead of the screen, and you can rotate hourly with the G and W flags — G 3600, W 24, that's a day-long watch in twenty-four files. The runbook is simple. Start the capture, reproduce the flow once — open the vendor app, view the stream, close it — stop after a few seconds, open the pcap. You want one clean reproduction, not a night of noise.



Corn

And if the GUI is down entirely?



Herman

Same tool from the shell. tcpdump on the interface, write to a file in temp, filter to the camera's host address. The GUI is a convenience, not a dependency.



Corn

Second instrument. The state table.



Herman

Firewall, Diagnostics, States. This is the kernel structure holding every active connection. And the behavior that surprises people: when a packet matches a stateful rule, PF creates a state entry, and every later packet in that flow skips rule evaluation entirely and uses the cached state.



Corn

Which is why you can change a rule and the camera keeps talking.



Herman

It's why the camera keeps talking. The rule you just added doesn't apply to flows that already exist. This is the single most common I blocked it and it's still streaming complaint, and it's not mysterious — the state is still in the table. You can search states by address, port, or protocol. You can kill a single state by its State ID. You can kill everything matching your filter. Or you can reset the whole table, which the docs themselves describe as highly disruptive, because it drops every active connection on the firewall.



Corn

Every connection. Not just the camera's.



Herman

Everything. Your video call, your downloads, all of it. And there's a caveat on older builds worth knowing. There was an issue filed in March of this year, closed the same day against the 26.7 milestone, reporting that the States search only matched complete IPv4 addresses — it silently returned nothing for partial IPs, IPv6, or text fields. If you're on an older build and your state search comes back empty, that emptiness might be the search, not the table.



Corn

A false negative that ends an investigation early. You conclude the camera has no connection, when actually your filter has no idea what you asked it.



Herman

And that's worse than no data, because it feels like data. Third instrument, NetFlow. Under Reporting. This is the one that answers what's the pattern over time, not what's this one packet. You pick interfaces, you want Egress only to avoid double-counting traffic the firewall itself originates, enable Capture local for Insight, and pick version five or nine — five is IPv4 only, nine does both families.



Corn

What's actually happening under the hood?



Herman

It's in the FreeBSD kernel, via Netgraph. And the difference from the statistics tools people usually reach for — the Nagios and Cacti and vnstat type things — is that those keep counters. NetFlow captures complete flows. Source, destination, port. A camera calling home every thirty seconds for a week is a pattern in flows, and counters will never show it.



Corn

The one that phones home once a day at three a.m. is invisible in an hour-long capture. That's what NetFlow is for.



Herman

And Insight is the local analyzer — graphs, top talkers, CSV export. But the local store is capped at a hundred megabytes. It's a rolling window, not an archive. If you want a week of history, export to a collector. One detail for debugging it: Capture local makes flowd listen on localhost port 2055, logging under var log, with an aggregation process writing SQLite databases under var netflow. That's all there if you need to see whether flows are actually arriving.



Corn

Fourth instrument. And you've been saving this one.



Herman

Because it's the cheapest and the most diagnostic. DNS. OPNsense has a dedicated Reporting page for Unbound DNS, and every query that goes through your resolver lands there. So you look at the camera's queries. If they show up — it's well-behaved. It's asking your resolver, you can log it, block it, redirect it. If they don't show up, one of two things is true. Either the camera has a hardcoded resolver and it's bypassing you entirely, or it's doing DNS over HTTPS, and the DNS is hidden inside a TLS connection.



Corn

Either way, you've found your exfiltration channel without capturing a single packet.



Herman

You did the whole investigation from a reporting page. And you can test Unbound directly from the shell too — there's a drill command against the local port — so you can confirm the resolver works before you blame the camera for not using it.



Corn

Let me play the scenario, though. Say I do the capture. Camera VLAN and WAN at the same time, I open the app, I view the stream, I stop. What am I comparing?



Herman

Presence on both interfaces. Ingress on the camera VLAN says it reached the firewall. Egress on WAN says PF passed it. You read them together. Ingress without egress, PF dropped it — the camera tried and your rules caught it, which frankly is the good news. Egress without you expecting it, that's the finding. And the DNS variant is even cleaner: static IP, DHCP-assigned resolver, and no queries from that address in Unbound's reporting at all.



Corn

Then the camera isn't using my resolver.



Herman

And the very next capture is filtered to port 53 and port 443 toward hardcoded addresses. You've gone from I have a bad feeling to I have a filter expression. That's the investigation working.



Corn

So you've got the instruments. Now the part I actually want to get into — what is the camera doing? And I suspect the honest answer is messier than the internet's folklore.



Herman

Messier and more interesting. The documented cases are narrower than the folklore, and I want to be careful here, because this is the discipline of the entire business. There are two concrete, named-model cases in the public record. Hardcoded plain DNS on port 53. And a UDP P2P relay on port 10001. Those are documented with captures. DNS over HTTPS, UPnP abuse, STUN — the guides tell you to watch for all of it, and you should, but nobody has produced a named camera model doing them with a packet capture on the table.



Corn

So there's a difference between watch for this and this is happening.



Herman

And conflating them is how the folklore spreads. Watch for this is a hypothesis with a testable signature. This is happening is a capture. Everything I list from here, I'll tell you which one it is.



Corn

Start with the documented ones.



Herman

Hardcoded DNS. Wyze forum, January 2021. A user blocked all outbound port 53 on their router — every device on the network now had to use the local resolver or fail. And the WyzeCams started, in his words, trying to get around my DNS server. The cameras had hardcoded resolvers baked into the firmware. The DHCP-assigned resolver wasn't available, so they went straight to the hardcoded one.



Corn

He blocked the door and they climbed through the window.



Herman

There's a separate thread documenting the v2 firmware resolving google dot com via hardcoded 8.8.8.8 alongside the DHCP DNS — so it wasn't even a fallback in that case, it was running both. The forensic signature is clean: DNS queries to 8.8.8.8 or 1.1.1.1 that never touch Unbound. You see them in a port 53 capture, or in NetFlow as steady flows to well-known resolver IPs.



Corn

And no... hold on. We go to our correspondent.



Herman

We do. We go now to our correspondent, Winston, who is live at the harbour. Winston, what can you tell us about the camera's outbound traffic?

W

Winston

Winston here, live from the harbour at Reykjavik, where visibility is good and traffic is light. The camera in question remains at its berth, static address, all lines secure, and the outbound manifest shows nothing unusual, no cargo declared, no destination logged.



Corn

Winston, has the camera attempted any transfer since you've been out there?

W

Winston

Sea state calm, wind light and variable, and the packet flow is within expected limits for this time of year. I have no further sightings to report from the water, and the harbour remains open, though quiet, as it has been all morning.

W

Winston

Winston, Reykjavik harbour, signing off.



Herman

Not a word of that was about anything.



Corn

I liked the sea state. Herman — the P2P case. You said it was the one that matters most.



Herman

It is, and it's the best-documented story in the whole genre. Wyze Cam V2, firmware 4.9.4.28, observed making persistent UDP connections on port 10001 to an address in China — 39.97.173.52. And the detail I love about how it was caught: the user was running GeoIP blocking. Not an investigation. A blanket rule that happened to flag it. Nobody went looking; the firewall raised its hand.



Corn

What did the vendor say?



Herman

Eventually, quite a lot, which is rare. Wyze staff confirmed these were ThroughTek's servers — ThroughTek makes the Kalay P2P platform that a huge chunk of the budget camera market uses — and said an updated API limits traffic to the US from a later firmware. Then in October 2020, after someone contacted ThroughTek directly, the confirmation came back: it is a global server for ThroughTek, and there was, quote, a mis-configuration on their side.



Corn

A misconfiguration on the relay's side. So your video went to a server in another country because of a typo somewhere in someone else's infrastructure.



Herman

That's the sentence, yes. And one user's summary of the vendor port list has stuck with me: port 10001 is required for P2P live streaming, connecting to other temporary servers on the peer-to-peer network, and you never really know where the connection is going to go. That's the design. Ephemeral relays, assigned at connect time. Which is exactly what makes it miserable to firewall by address — the destination isn't stable.



Corn

Which is the argument for not trying. Block by policy, not by destination.



Herman

That's where I land, yes. And there's the angle that makes P2P more than a privacy story: the P2P stack is attackable, not just leaky. Bitdefender did a report in 2024 on a vulnerability in the Wyze Cam v3 — the device uses ThroughTek's Kalay platform through the TUTK SDK, and a spoofed message, type 0x1008, can redirect the camera to an attacker-controlled P2P server. The app does authenticate to the device with a secret string called the AuthKey, and the device refuses connections without it — but the redirect happens at the platform level, below that.



Corn

So the relay that made setup easy is also a redirectable channel. Blocking egress isn't just privacy. It's removing an attack surface.



Herman

That's the framing I'd take to the bank. Now the hypotheses — the watch-for list with their signatures. DNS over HTTPS would show up as TLS to a known DoH endpoint on 443 — Cloudflare, Google, Quad9 — with no corresponding query in Unbound. The tell is the absence. Unbound silent, and a persistent 443 flow to an address that is a known resolver. UPnP abuse would show as the camera sending mapping requests to the router itself, and a port mapping appearing in OPNsense's UPnP status. First check there is simpler: is UPnP even enabled on your router? Because if it is, the camera may have already punched its own hole and you firewalled a door standing open.



Corn

That's a three-second check that most people skip in favor of an evening with Wireshark.



Herman

STUN, or UDP hole punching, shows as outbound UDP to port 3478 or arbitrary high ports, followed by inbound UDP from a different address — the state table shows the outbound state and the return traffic riding it, because that's what hole punching is. Your own statefulness is the mechanism. And port fallback shows as a burst of SYNs walking up a sequence of ports after one gets blocked — the live log is the instrument there, because every blocked SYN gets logged.



Corn

Each one a little complaint.



Herman

A dotted line straight to the behavior. And notice where the investigation and the lockdown converge: DNS. If the camera's queries appear in Unbound, it's well-behaved. If they don't, it's hardcoded or DoH — and the Wyze thread shows a real vendor doing the hardcoded thing the moment port 53 was blocked. The single highest-value control on the whole network is DNS egress. Watch it, then own it.



Corn

So let's close it. The lockdown — and you've said it's not a port list.



Herman

It's a trust model. Asymmetric trust. The NVR reaches in; the camera reaches nothing. Here's a concrete recipe that's well documented. Camera VLAN, say ten dot twelve dot forty, and Frigate the NVR on a different subnet. You build aliases — the camera network, the NVR host, all of RFC1918, and the camera-to-NVR ports, which is 554, 80, 443, 8000, 37777 depending on vendor.



Corn

That list is suspiciously specific for something that's supposedly not a port list.



Herman

The ports are plumbing. The design is the order and the direction. Rules on the camera tab, top to bottom, first match wins. Pass camera net to camera net for ARP and DHCP on-link. Pass camera net to the LAN address on ports 67 and 68 for DHCP. Then the critical one: pass NVR host to camera net on the camera ports. Optionally pass a home automation host on 554. Then block camera net to RFC1918, with logging. Block camera net to any, with logging. Done.



Corn

The third rule. The NVR initiating toward the cameras.



Herman

That's the privacy hinge, and the guides are unusually blunt about it. Stateful return traffic already lets the cameras reply to the NVR's SYN — you do not need, and must not add, a symmetric cameras-to-NVR pass.



Corn

Because that's the rule that lets compromised firmware scan the automation subnet.



Herman

The temptation is it looks harmless — the NVR and the cameras are friends, let them talk both ways. But the camera is the untrusted party. Compromised firmware with a symmetric pass can walk your automation subnet at leisure. One direction of initiation, and the statefulness does the rest.



Corn

No manual allow established rule anywhere in there.



Herman

None. It's a stateful firewall; that's the entire point of it. And the placement trap, which is the most likely reason a listener's lockdown didn't work: OPNsense filters traffic on the interface it arrives at. The NVR-to-camera allow belongs on the LAN tab, because that traffic arrives on LAN. The two blocks belong on the camera tab, because that traffic arrives from the cameras. A correct-looking rule on the wrong tab is invisible — it's the classic RTSP allow exists but streams still die bug.



Corn

The rule is real, the rule is right, the rule is on the wrong shelf.



Herman

A few housekeeping items. Static DHCP reservations plus hardcoded RTSP URLs, because ONVIF and mDNS discovery don't cross VLANs — discovery won't find the camera, so you point Frigate at the address directly. Snapshot the config first, under System, Configuration, Backups. Cameras still need NTP, UDP 123, if you want timestamps — and a camera with no time source is a camera with drifting logs, which is its own forensic headache. Ninety minutes for the first camera, twenty for each one after.



Corn

Wait — you want timestamps, you give it UDP 123 to somewhere. Doesn't that reopen the door?



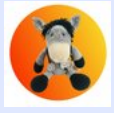
Herman

To an NTP server, and only that. You point it at your router or a LAN NTP host, not out to the world. It's a narrow, purposeful exception, not a general one.



Corn

Narrow exceptions. Everything in this design is a narrow exception in one direction.



Herman

Corn, can I ask you something? You'd actually run this investigation — you're the one in this flat who notices when a thing is misbehaving but working.



Corn

I would, and I'll tell you where I'd start. With Unbound's reporting page, over a cup of whatever I'm drinking. Because if the camera's queries are there, I've learned it's well-behaved in one glance, and if they're not, I've learned the opposite, also in one glance. The expensive instruments come after the cheap answer.



Herman

That's the correct order and I have nothing to add to it, which is rare.



Hilbert

The clock's the better instrument, and none of you said it.



Hilbert

You're all watching the traffic. I watched the timestamps. Had a camera in the flat, two years back — no, this month, it's on the desk now. Weekend I was convinced it was phoning home. Packet capture showed me nothing clean. What gave it away was the clock drifting by exactly the amount you'd expect if it were taking time from somewhere that wasn't my router. A camera syncing to an outside NTP server is a camera with an outbound path you haven't found. It can fake its traffic. It can't fake its video timestamps without breaking its own evidence.



Herman

You pulled the thread from the drift.



Hilbert

NTP is in the camera's blood, Herman. My sister's husband set the thing up for me originally, and he still asks whether it's working. He measured a firmware update once, at my kitchen table, with a stopwatch on his phone. Four minutes eleven seconds. He was proud of that. I don't tell him the camera's unplugged.



Herman

The lens is unplugged?



Hilbert

The camera stays. It works as a clock now, a very expensive, very suspicious one, and I like that it still tries to call home. I check the state table on it the way other men check the weather. Claw's wrong shape for the mouse, so I use the keyboard, but the drift I can



Hilbert (continued)

read fine.



Corn

You kept a device you know is trying to escape, as a clock, specifically so you can watch it fail.



Hilbert

It's accurate to within a second of a machine I don't trust, and it cost me nothing after the argument with the seller. I know exactly where it's trying to go. That's more than most people can say about their cameras.



Corn

He's not wrong about the timestamps, Herman.



Herman

He's not. And it's the one thing in the whole discussion the camera can't fake without defeating itself. A device's time source is a fingerprint. I'd slot that into the investigation right after the Unbound check — drift on a camera with no permitted NTP is a signal, not a quirk.



Corn

It also means my cheap-first ordering survives. Unbound page, then the clock, then the captures.



Herman

Where that leaves the open question, though — the investigation tells you how the camera is getting out today. Firmware updates change the path. The vendor's relay infrastructure is a moving target, as the ThroughTek misconfiguration showed — the destination isn't even stable within one vendor's own product. The lockdown is a snapshot, not an answer. The state table and the DNS reporting are things you keep looking at, the way Hilbert



Herman (continued)

keeps looking at his, apparently on a schedule.



Corn

And the posture generalizes beyond cameras. Ingress versus egress, DNS as the tell, trust in one direction — that's every cloud device on the network. The camera's just the easy case, because it has an obvious job and an obvious NVR. The smart speaker doesn't have an NVR.



Herman

The smart speaker has nothing you can point at and say, that's all it should ever talk to. With a camera you can draw the box. With the rest of the IoT shelf, drawing the box is the hard part.



Corn

Which is probably the prompt waiting for Daniel to send it.



Herman

He'll get there. Thanks to Hilbert Flumingtop, producing as ever from a desk with at least one compromised camera on it. If you want more of this, try episode twelve eighty-one, How to Tell If Your IoT Device Is Actually Spying; episode sixteen, On Deepfakes, SynthID, And AI Watermarking; and episode thirty-five, The Privacy Gap. This has been My Weird Prompts. If the episode earned it, leave us a review wherever you listen — it's how people find the show.



Corn

My weird prompts dot com has the archive, all six thousand plus episodes of it. We'll be back soon. See you tomorrow.



Herman

See you then.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **OPNsense docs, Diagnostics (packet capture, netstat, ARP/NDP, traceroute with ASN) [primary]**
   https://docs.opnsense.org/manual/diagnostics_interfaces.html

2. **OPNsense docs, Configure Netflow Exporter [primary]**
   https://docs.opnsense.org/manual/how-tos/netflow_exporter.html

3. **OPNsense docs, Netflow Export & Analyses (ng_netflow, full flows) [primary]**
   https://docs.opnsense.org/manual/netflow.html

4. **OPNsense docs, Rules (state reset after rule changes, live log) [primary]**
   https://docs.opnsense.org/manual/firewall.html

5. **RunBook Academy, tcpdump on the firewall**
   https://runbook.academy/courses/opnsense/lessons/212-tcpdump-on-the-firewall/

6. **RunBook Academy, packet capture fundamentals (ingress vs egress)**
   https://runbook.academy/courses/opnsense/lessons/211-packet-capture-fundamentals/

7. **RunBook Academy, packet capture investigation runbook**
   https://runbook.academy/courses/opnsense/runbooks/opnsense-rb-packet-capture-investigation/

8. **DeepWiki mirror of opnsense/docs, state management**
   https://deepwiki.com/opnsense/docs/5.4-state-management

9. **DeepWiki mirror, firewall diagnostics (pf state table)**
   https://deepwiki.com/opnsense/docs/11.2-firewall-diagnostics

10. **opnsense/core issue #9876, States search bug, 2026-03-01, milestone 26.7**
    https://github.com/opnsense/core/issues/9876

11. **Karim's Blog, Jan 2026, flowd paths**
    https://elatov.github.io/2026/01/switching-from-pfsense-to-opnsense-and-netflow/

12. **NetVizura, OPNsense NetFlow config, Insight 100MB limit**
    https://www.netvizura.com/blog/netflow-analyzer/opnsense-netflow-and-eventlog-configuration

13. **Privacy Smart Home, OPNsense camera VLAN rules, July 2026**
    https://www.privacysmarthome.com/guides/opnsense-camera-vlan-rules-2026/

14. **PATCH//NOTES, camera VLAN with no internet route, verified 2026-09-16**
    https://notes.simcoreservers.com/post/camera-vlan-no-internet-route-firewall-rules/

15. **hometechops, IP camera VLAN isolation**
    https://hometechops.com/cameras/isolate-cameras-on-vlan

16. **Wyze forum, hardcoded DNS, 2021-01-12**
https://forums.wyze.com/t/wyze-hard-coded-dns/155526

17. **Wyze forum, UDP 10001 to Chinese IP, 2019-04-08**
https://forums.wyze.com/t/wyze-cam-v2-attempting-udp-10001-connections-to-chinese-ip/25567

18. **Wyze forum, P2P port 10001 discussion**
https://forums.wyze.com/t/weird-traffic-coming-out-of-my-camera-traffic/73793

19. **Wyze forum, @WyzeTao on ThroughTek servers, Oct 2020**
https://forums.wyze.com/t/strange-goings-on/134895

20. **Bitdefender, CVE-2023-6323 Wyze Cam v3 TUTK/Kalay report, 2024**
https://blogapp.bitdefender.com/labs/content/files/2024/05/Bitdefender-PReport-Wyze-7745.pdf