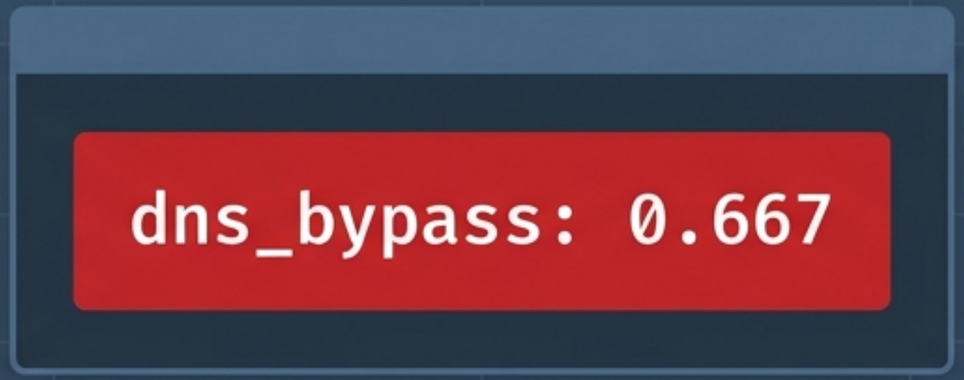
Deconstructing a DNS Bypass Alert



A case study in network observability, asymmetric trade-offs, and UniFi API engineering.

Trust the Signal, Verify the Math

A red security card is a hypothesis. Recomputing the metric manually proved the dashboard was right.

Math: `17 / 26 = 0.654`

Note: Lines up cleanly with the 0.667 dashboard alert within polling jitter.

Conclusion Box

Two-thirds coverage is not a noisy alert. It is a real gap.

The Intersection Engine

One Red Alert. Two Distinct Failure Modes.

Default Subnet

Symptom

Resolving DNS, but skipping Pi-hole.

Root Cause

DHCP option 6 round-robins across Pi-hole (`172.16.27.227`), Quad9 (`9.9.9.9`), Cloudflare (`1.1.1.1`), and Google (`8.8.8.8`).

Impact

3 out of 4 queries bypass the internal resolver.

VLAN 2 & VLAN 3

Symptom

Cannot reach Pi-hole at all.

Root Cause

`dhcpd_dns_enabled: false` meant no advertised resolver.

`network_isolation_enabled: true` triggered a ZBF BLOCK rule at index 30000.

Impact

Kills internal cross-VLAN traffic. Completely broken DNS.

Designing an Asymmetric Trade-off

The dashboard wants 100% coverage. Real networks require resilience.

Default Subnet

Keep public DNS fallbacks.

Accept a non-green dashboard to guarantee uptime on the house's most critical network.

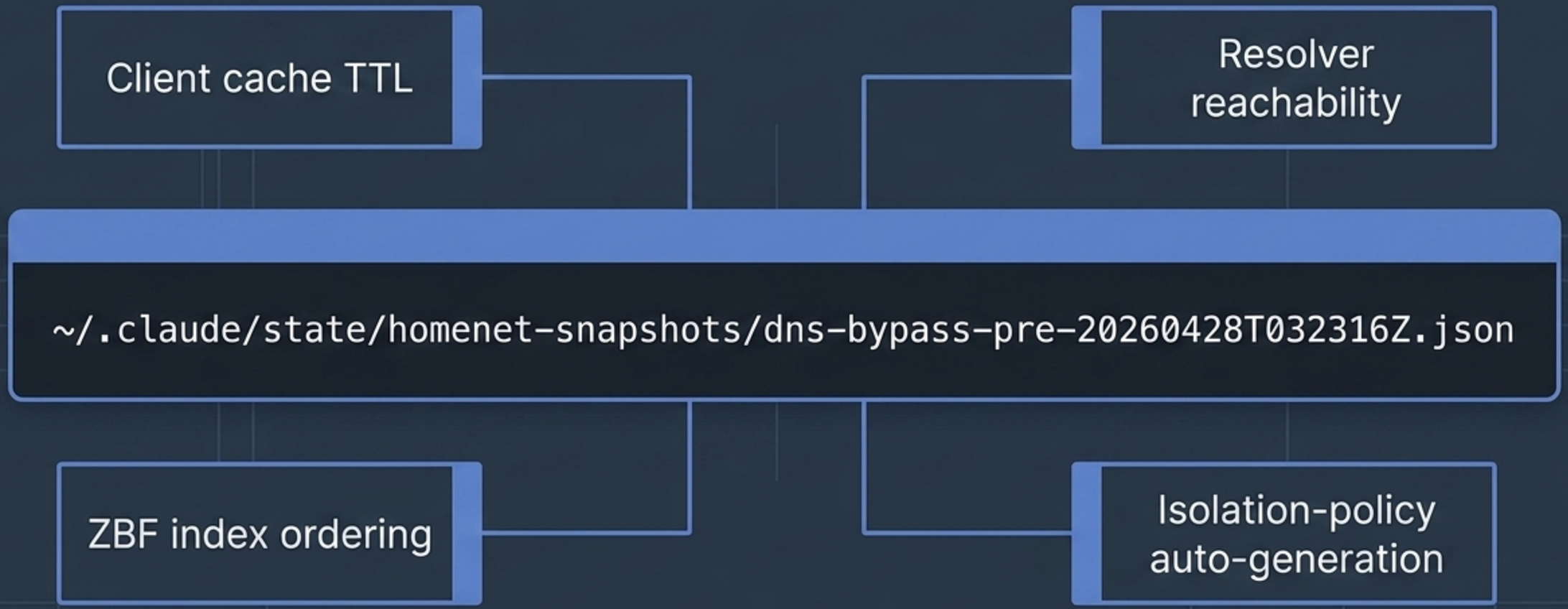
VLAN 2 & 3

Force full closure.

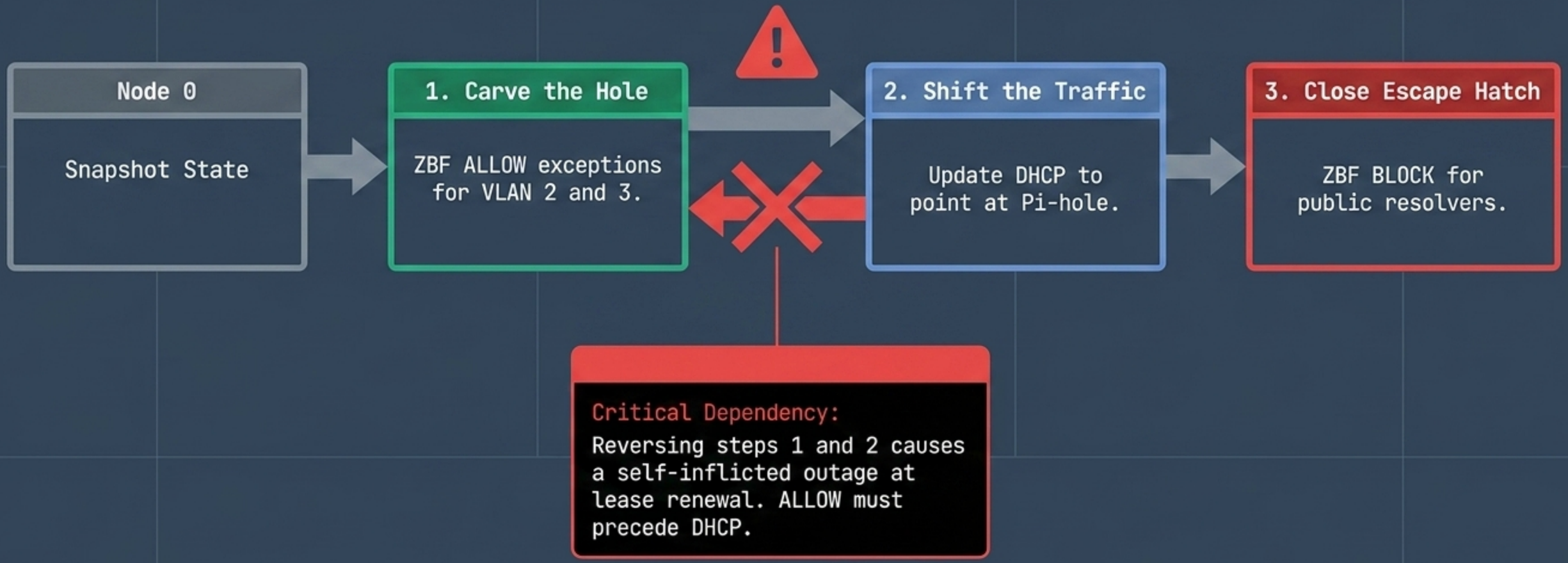
These are isolated trust tiers where observability matters more than uptime.

The Pre-flight Rollback Contract

A network change involving DHCP and firewalls touches four volatile components. Manual recovery takes hours. A JSON snapshot takes six seconds.



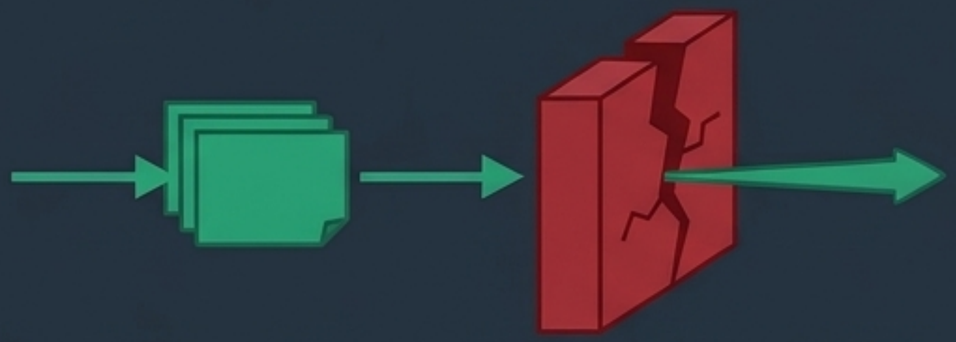
The Remediation Pipeline



Execution: Carving the Hole and Flipping DHCP

Panel 1: ZBF Allow (Index 10000)

Bypasses the index-30000 isolation policy using chris2ao/unifi-mcp wrappers.

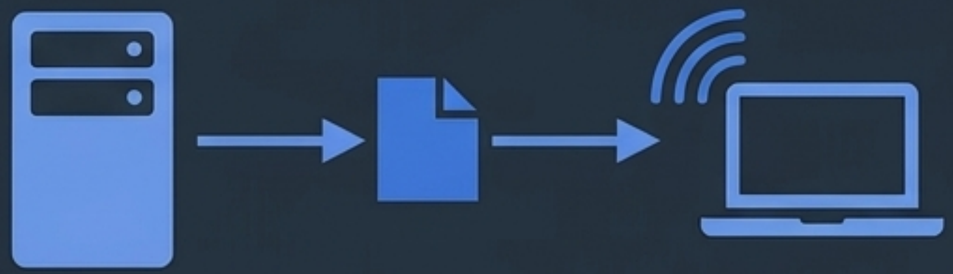


Payload

```
src: VLAN 2 & 3
dst: 172.16.27.227
port: 53 (UDP+TCP)
```

Panel 2: DHCP Update

Forces new leases and reconnects to use the internal resolver.

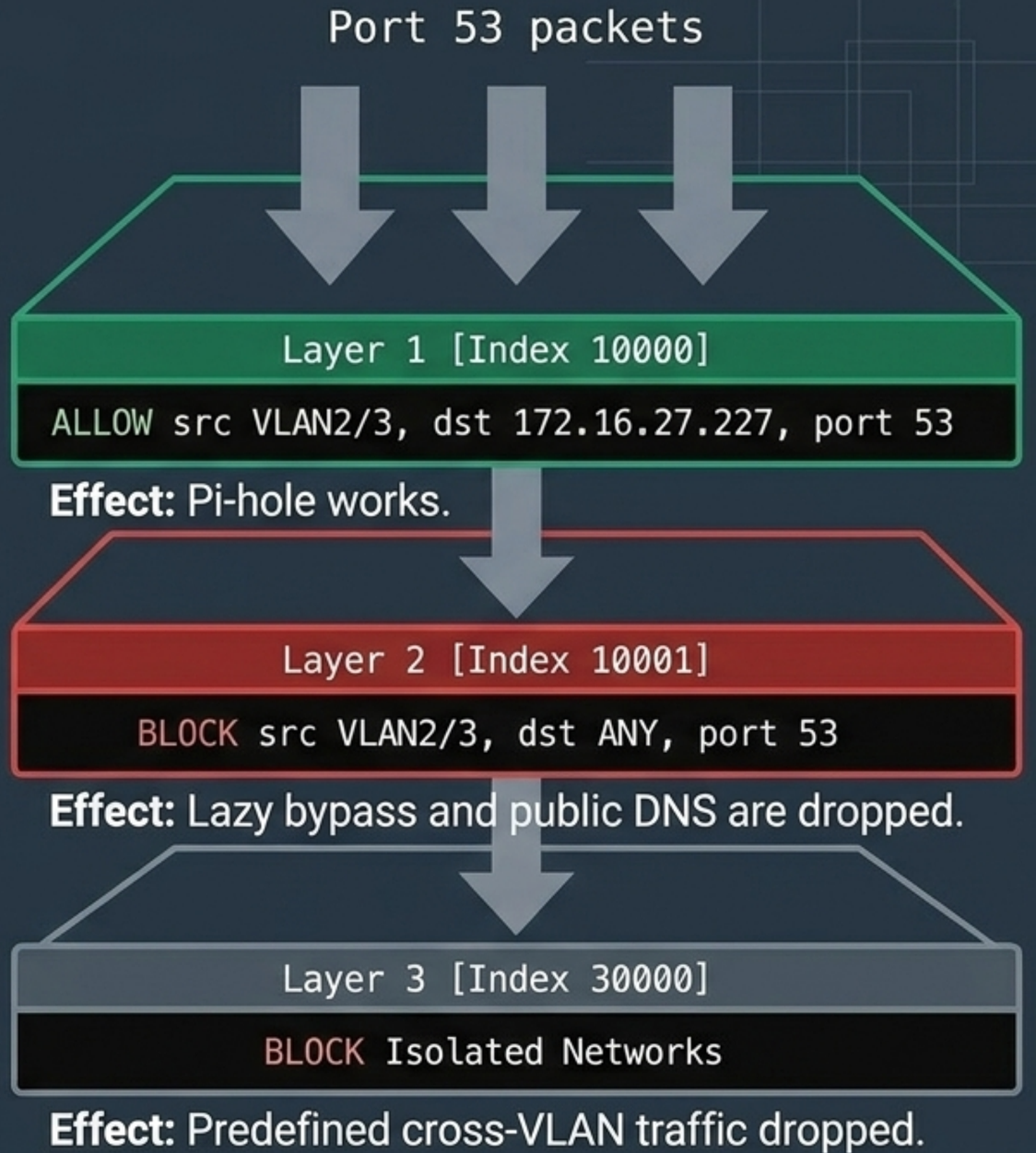


Payload

```
dhcpd_dns_enabled: true
dhcpd_dns_1: 172.16.27.227
```

Closing the Escape Hatch

Caveat Note: DNS-over-HTTPS (TCP 443) passes through. Blocking DoH requires SNI inspection, stepping up the threat model beyond the port-53 critical path.



The Tooling Pivot: Traffic Rules vs. ZBF

Initial instinct was to build API wrappers for UniFi Traffic Rules. The wrapper tests passed, but the logic failed. Traffic Rules cannot filter by destination port.

Capability	Traffic Rules	ZBF Policies
Target Devices / Schedules	Yes ✅	No ❌
App-Category Blocks	Yes ✅	No ❌
L4 Port-Level Filtering	No ❌	Yes ✅
Zone Indexing / Tuple Match	No ❌	Yes ✅

UniFi v2 API Schema Gotchas

What you expect	What the controller accepts
`name` field	`description` (Silently drops name on persist)
`ports` array	None (Feature simply doesn't exist)
Payload: `{ exclude: true }`	Normalizes away (Cannot anti-target devices)
`GET /trafficrules/{id}`	**405** Method Not Allowed (Must fetch collection instead)
Standard creation on BLOCK	`create_allow_respond: false` (Rejects with api.err.FirewallPolicyCreateRespondTrafficPolicyNotAllowed otherwise)

The Final State

VLAN 2 & VLAN 3

Fully closed. 100% Pi-hole coverage.

Default Subnet

Deliberately partial. Option-6 fallbacks remain. Dashboard sits at ~0.85.

DoH Bypass

Untouched. Left for future SNI-level inspection.

Open-Source Tooling

Shipped `chris2ao/unifi-mcp` **v0.4.0**. Six new tools, 13 new tests, and a fix for empty 2xx DELETE responses.

Principles of Pragmatic Engineering

1. Verify Before Fixing

A red card is a hypothesis. Recompute the underlying data before changing the network.

2. Resilience Over Coverage

A network that satisfies a dashboard but fails the operator is a worse network. Accept asymmetric trade-offs.

3. Match Surface to Rule

Tools aren't interchangeable. Traffic rules are for app categories; ZBF is for port-level routing.

4. State is Code, Not Memory

Snapshot volatile states in JSON. A six-second script prevents hours of manual rollback.