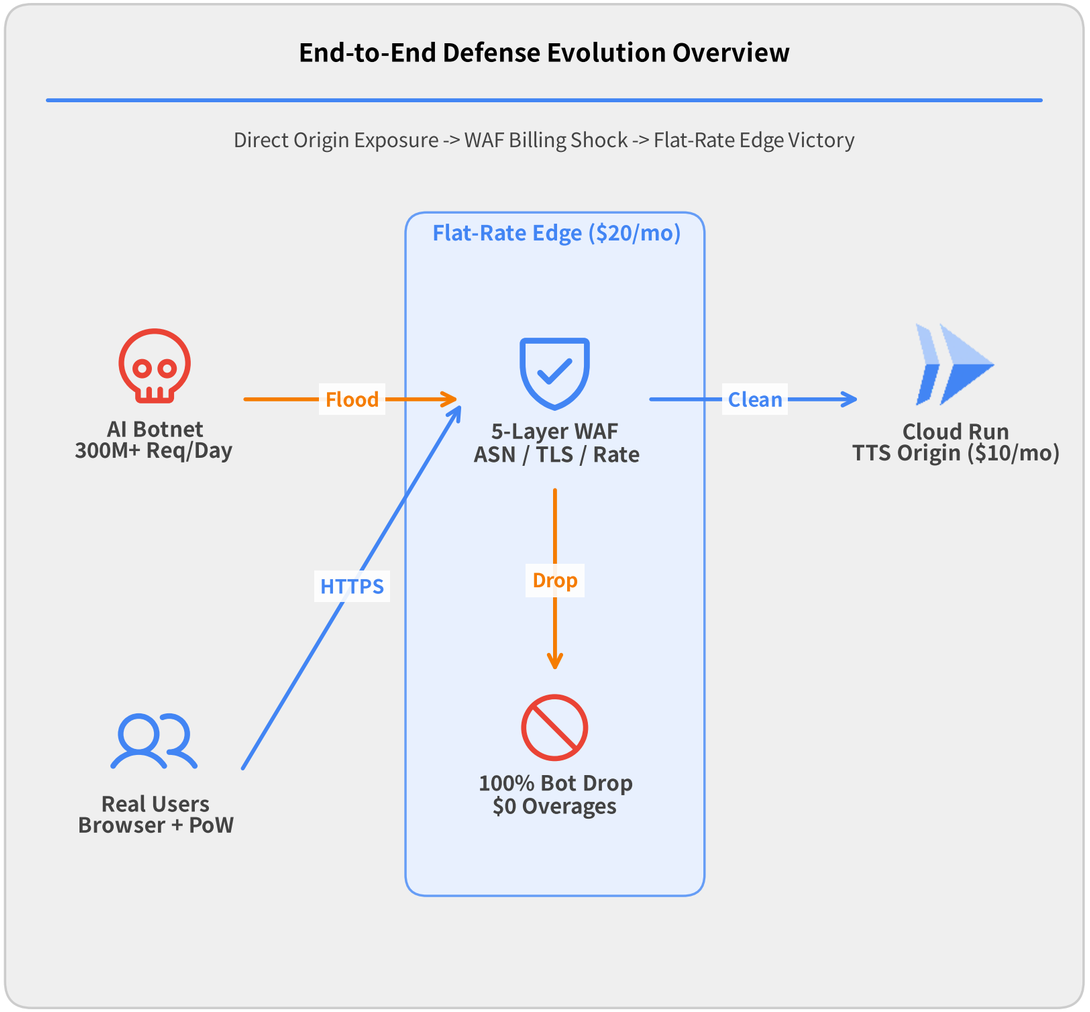
Surviving a 300M+ Req/Day DDoS & EDoS Attack

A Real-World Serverless Postmortem on Google Cloud Run & Edge WAF Architecture

How an AI-orchestrated botnet pushed a 10-year hobby speech synthesis service from $10/month to a $6,750/month WAF billing crisis — and how 5-layer edge defense defeated it for $20/month flat.

Incident Postmortem & Cloud Security Case Study · 2026



Agenda & Incident Executive Summary

1. **Act 1: The Incident Onset**
   10 Years of VPS Peace to Serverless Bill Shock
2. **Act 2: Perimeter WAF & Evasion**
   5-Step Cycles, GCLB WAF, 4 Geo-Waves & JS Crack
3. **Act 3: PoW & EDoS Pricing Trap**
   SHA-256 PoW, 300M/Day Flood & $225/Day WAF Crisis
4. **Act 4: Multi-Layered Victory**
   Flat-Rate Cloudflare Pro, 5 Layers & AI Forensics

300M+
Peak Attack / Day
Volumetric flood (5,000+ nodes)

$6,750
Projected WAF Bill
$225/day Cloud Armor fee trap

<50ms
Browser SHA-256 PoW
Asymmetric Web Worker puzzle

$20/mo
Flat-Rate Defense
Cloudflare Pro + 5 edge layers

ACT / CHAPTER

01

Act 1 of 4

The Incident Onset

From 10 Years of VPS Peace to Serverless Cloud Run Bill Shock

1.1 10-Year Legacy VPS vs. Public Serverless Cloud Run Exposure

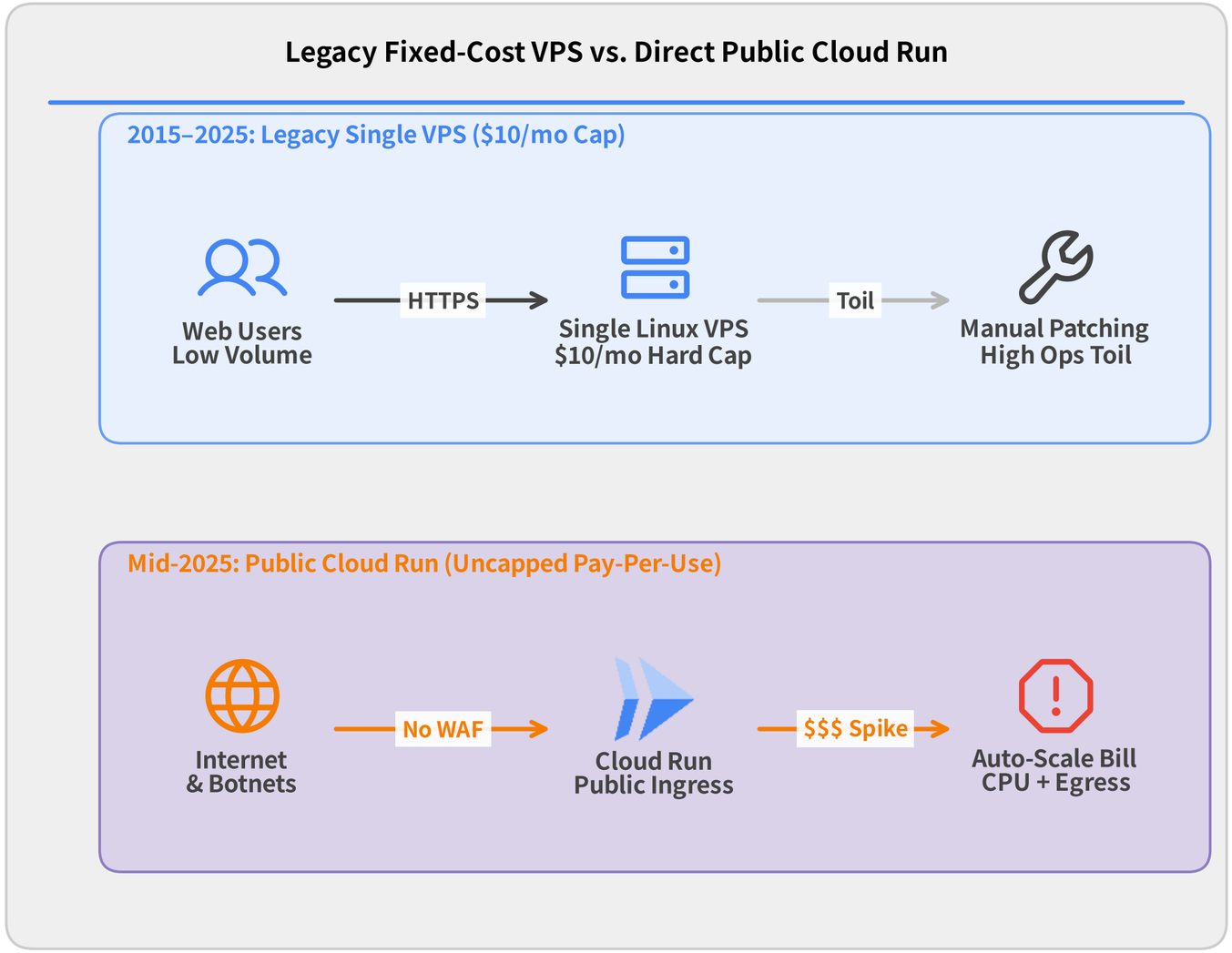
1.2 Cache-Busting TTS Synthesis Flood & 4-5x Weekly Bill Spike

1.3 Initial Triage: Why Origin Clamping Fails vs. Edge Drop

Background: A 10-Year Hobby Service Enters the Cloud

Architecture Evolution

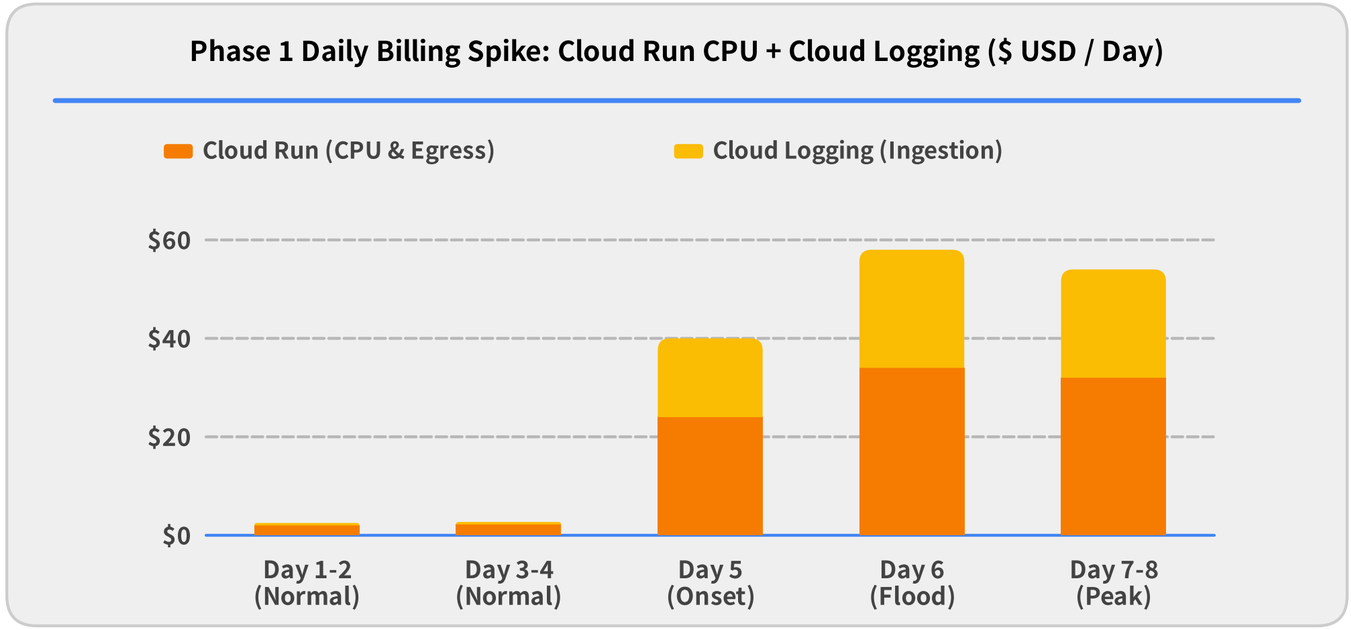
- **10 Years of Peace (2015–2025)** Japanese TTS web service running at **$10/mo** on a single VPS.
- **Serverless Migration (Mid-2025)** Containerized onto **Google Cloud Run** for zero-ops maintenance.
- **Hidden Vulnerability** Direct public URL (`run.app`) with **no CDN or Edge WAF** and uncapped pay-per-use scaling.

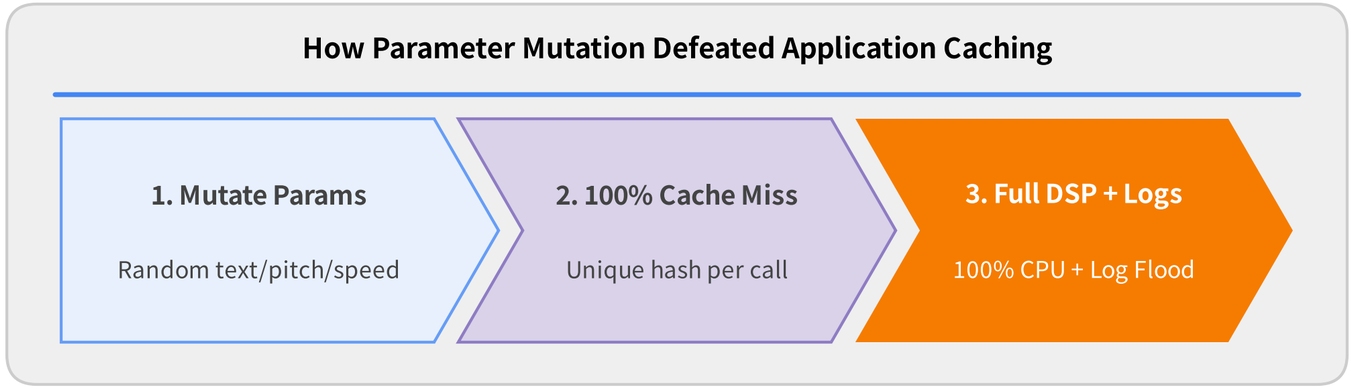


The Attack Onset: Cache-Busting TTS Flood

Telemetry & Billing Shock

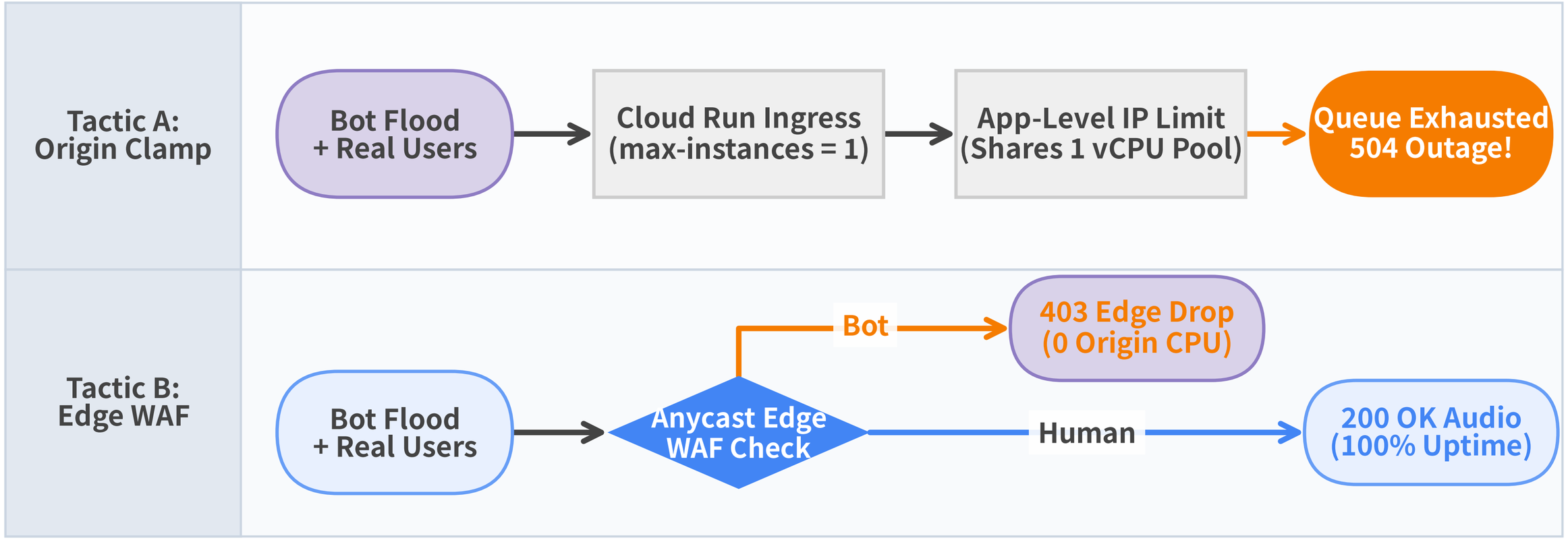
- **Cloud Run + Cloud Logging Surge** Compute/egress and high-volume request logging jumped **15–20x above daily baseline**.
- **100% Cache-Miss Exploitation** Bots randomized `text`, `pitch`, and `speed` query parameters on every call.
- **CPU & Log Saturation** Every request forced cold DSP audio synthesis and wrote gigabytes of access logs.





Initial Triage: Origin Clamping vs. Edge Filtering

- **Tactic A (`max-instances=1` + App Rate Limit)**: Caps runaway Cloud bill immediately, but **starves legitimate users (504 Gateway Timeout)** as bots fill the queue.
- **Tactic B (Dedicated Edge WAF Layer)**: Drops malicious floods at the **Anycast POP (403 Forbidden)** so only clean traffic reaches the Cloud Run container.



ACT / CHAPTER

02

Act 2 of 4

Perimeter WAF & Evasion Loops

5-Step Adaptation Cycles, 4 Geo-Hopping Waves & JS Token Cracking

2.1 Defender vs. Attacker: Two 5-Step Adaptation Cycles

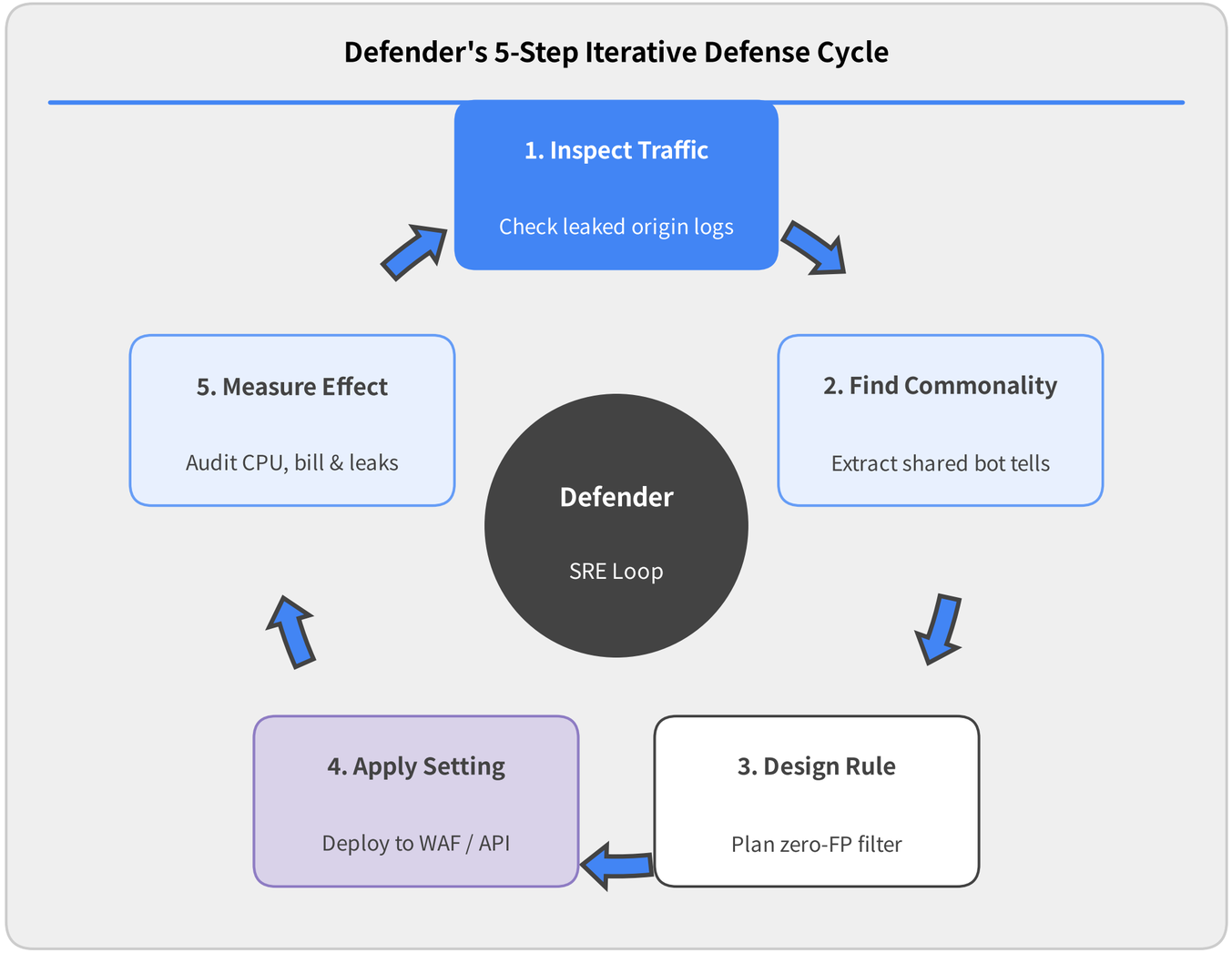
2.2 GCLB + Cloud Armor WAF & 4 Proxy Geo-Hopping Waves

2.3 Static Client JS Token Spoofing (Cracked in 3 Hours)

The Defender's Cycle: 5-Step Iterative Mitigation

Telemetry-Driven Defense Loop

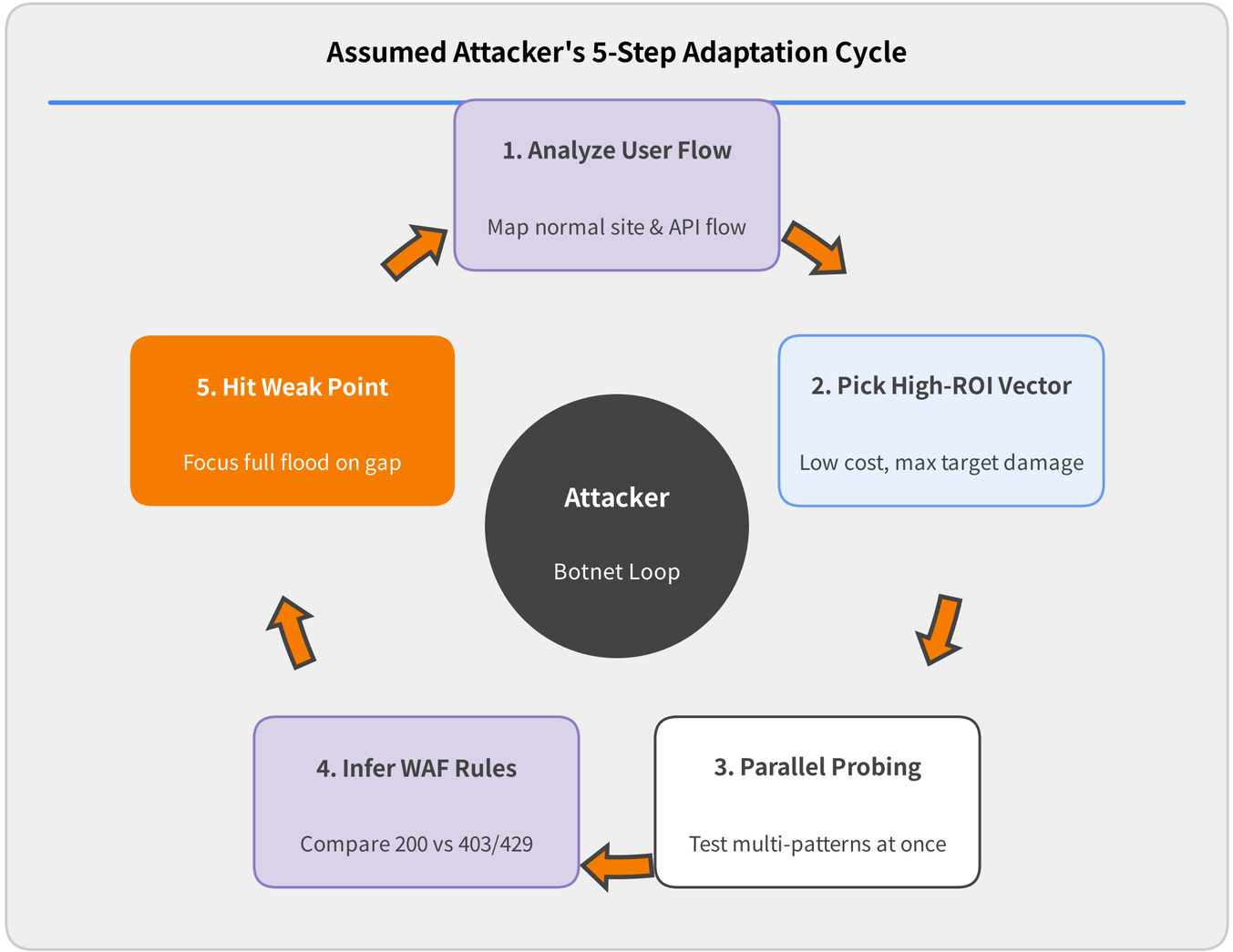
- **Observe & Extract (Steps 1–2)** Inspect attack traffic reaching the origin and isolate shared bot fingerprints (Geo, UA, ASN, HTTP protocol).
- **Design & Apply (Steps 3–4)** Engineer targeted blocking rules that avoid false positives on organic users, then deploy to Edge WAF or API.
- **Measure & Iterate (Step 5)** Evaluate block rate, origin CPU, and cloud billing—looping back to Step 1 whenever residual traffic leaks through.



The Attacker's Cycle: 5-Step Probing & WAF Inference

Assumed Adversary Loop

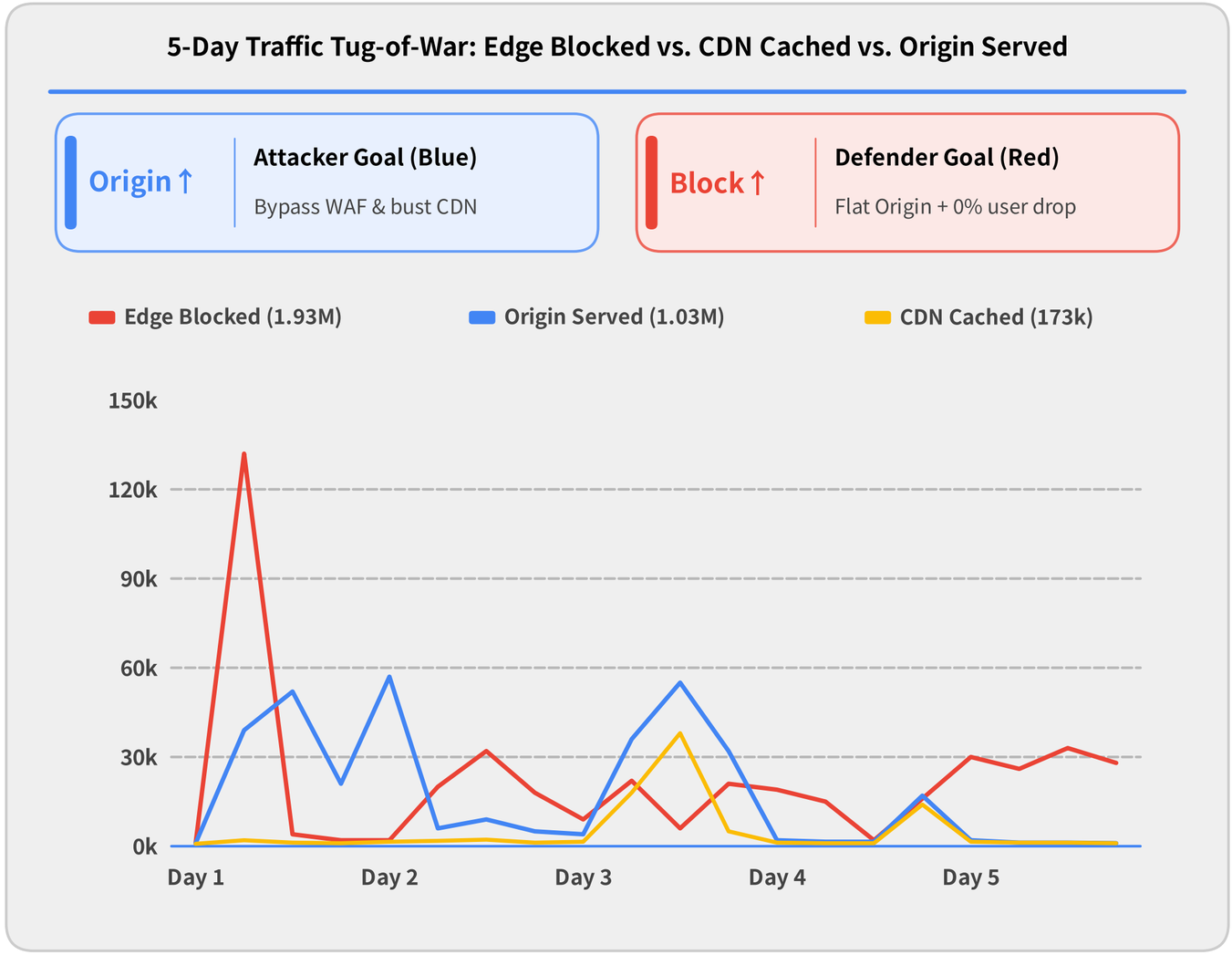
- **Flow Analysis & Vector Selection (Steps 1–2)** Analyze legitimate user access flows and select an attack method with minimal bot cost and maximum target CPU or billing damage.
- **Parallel Probing & Rule Inference (Steps 3–4)** Test multiple attack patterns in parallel and compare passed (`200`) vs. blocked (`403` / `429`) responses to infer WAF rules.
- **Blind-Spot Concentration (Step 5)** Once the unblocked weak spot is found, concentrate all botnet traffic onto that pattern—and repeat when blocked.



Cycle Tug-of-War: Block, CDN & Origin Telemetry

Competing Objectives in Logs

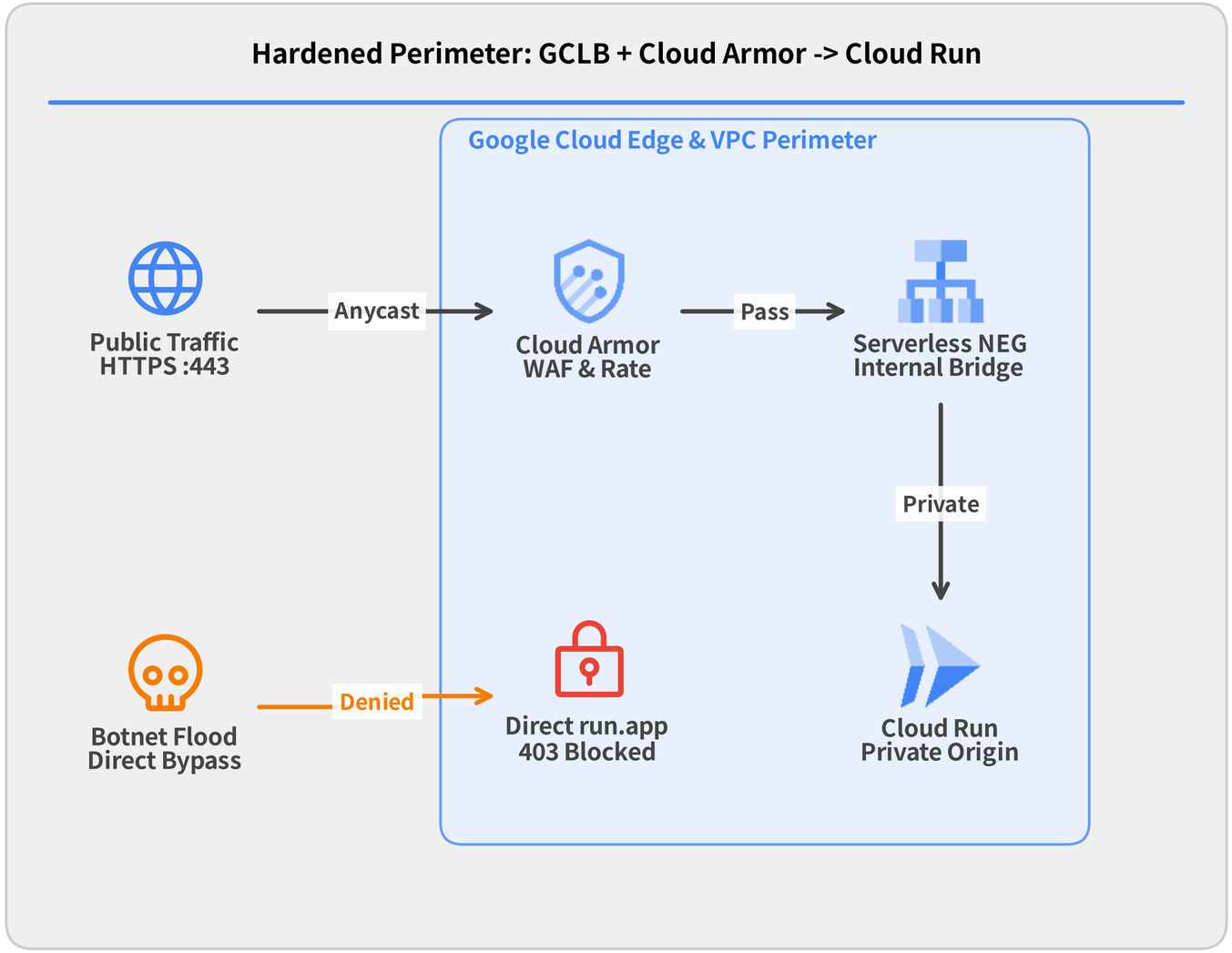
- **Attacker's Goal: Drive Origin Up** Evade WAF filters and randomize parameters so requests penetrate to **Origin (Blue)** and burn server resources.
- **Defender's Goal: Drive Block & CDN Up** Absorb attacks via **Edge Block (Red)** and **CDN Cache (Amber)** while keeping **Origin flat** with **zero false positives**.
- **5-Day Cat-and-Mouse Waveform** Every **Blue spike** marks an attacker bypass; every **Red takeover** marks a new defense rule sealing the leak.



Perimeter Defense: GCLB & Cloud Armor WAF

Edge Shield Architecture

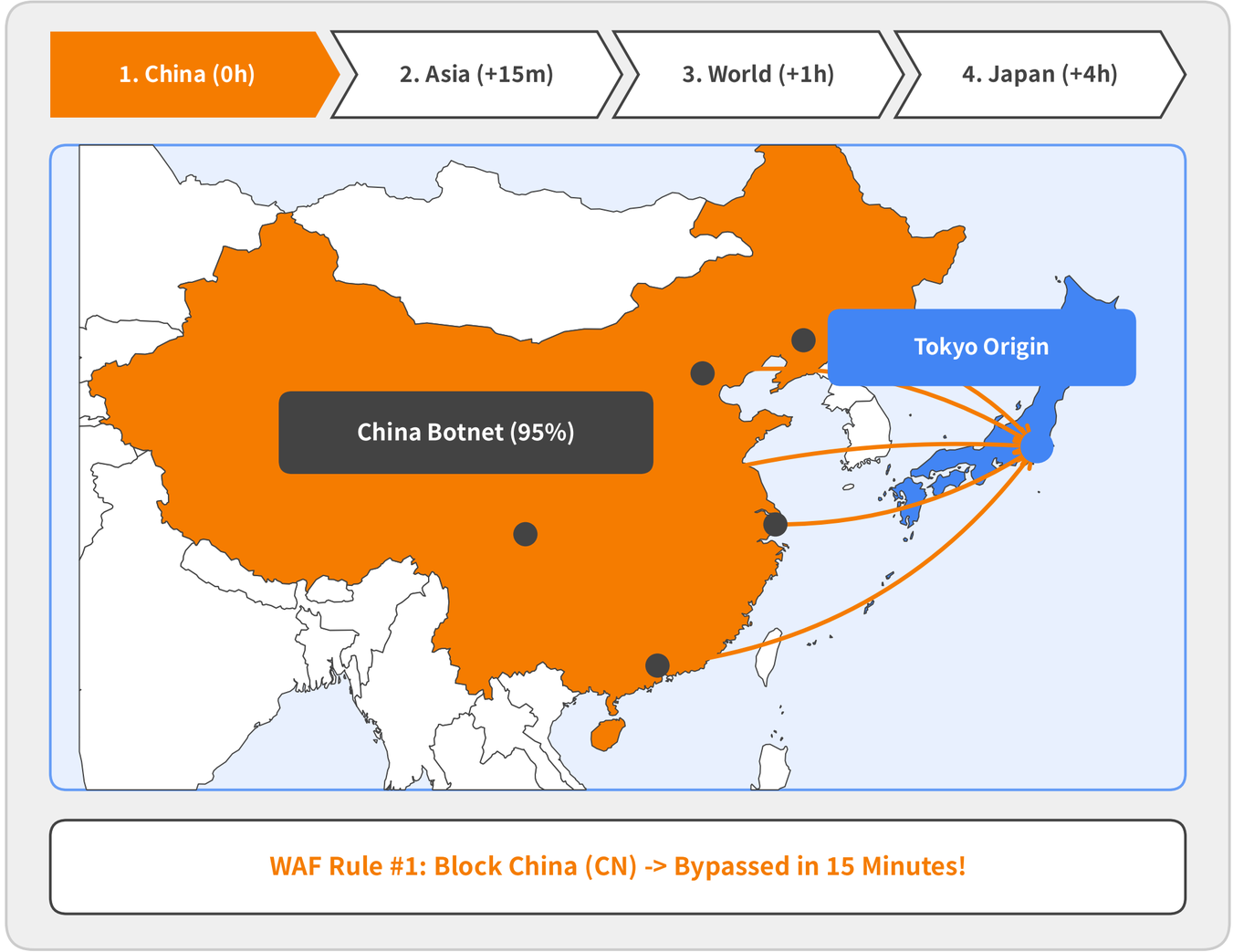
- **Private Origin Isolation** Locked down `run.app` direct URL (`ingress = internal-and-cloud-load-balancing`).
- **Global Anycast Ingress** Placed **Google Cloud Load Balancing (GCLB)** with **Serverless NEG** in front of Cloud Run.
- **Edge L7 WAF Policies** Attached **Cloud Armor** for edge IP rate limiting and geo-filtering before container invocation.



Why Geo-Blocking Failed (1/4): Initial Flood from China

Wave 1: Single-Country Origin

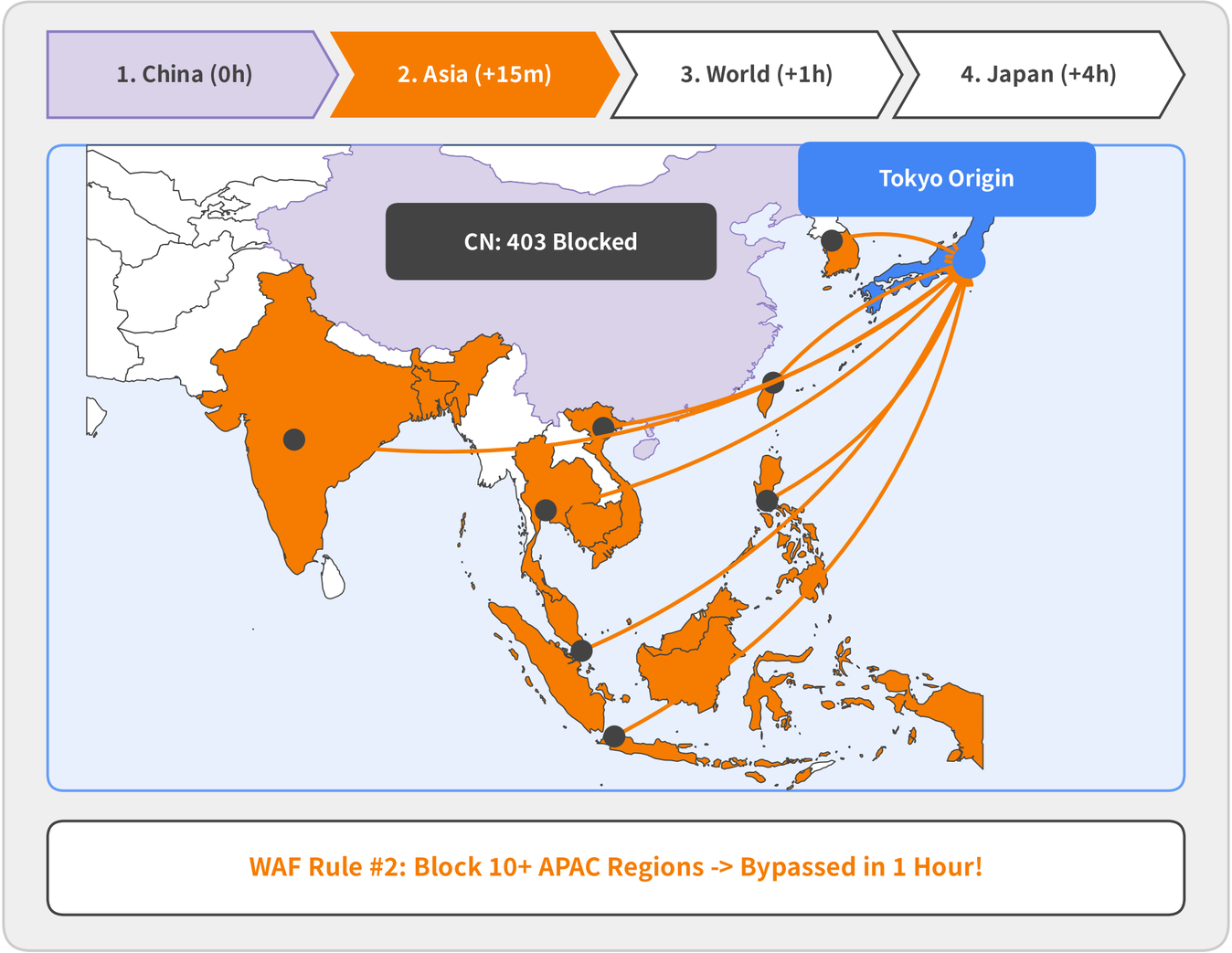
- **95%+ Traffic from China (`CN`)** Initial Cloud Armor telemetry showed almost the entire cache-busting TTS flood originating from Chinese IP ranges.
- **WAF Rule #1: Block `CN` Region** Deployed an edge geo-filter in Cloud Armor to drop all requests from China (`403 Forbidden`).
- **Bypassed in 15 Minutes** Attack volume dropped to zero briefly—until the botnet's automated failover shifted proxy regions.



Why Geo-Blocking Failed (2/4): Regional Shift Across Asia

Wave 2: Neighboring Asian Proxies

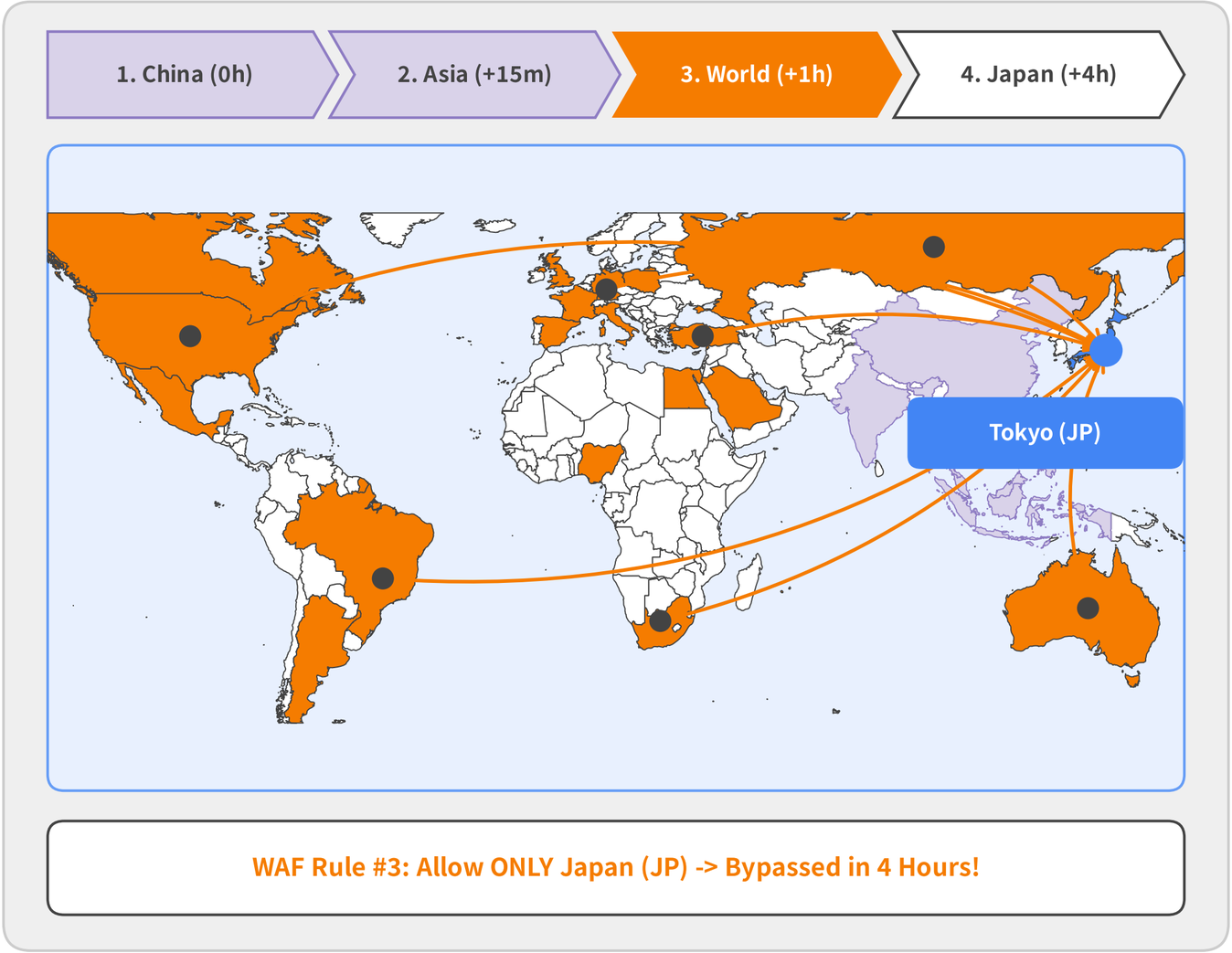
- **Instant Regional Failover (+15m)** Just 15 minutes after blocking `CN`, the flood resumed at full volume from surrounding Asian countries.
- **APAC Stepping-Stone Swarm** Traffic poured in simultaneously from **Korea (`KR`), Taiwan (`TW`), Vietnam (`VN`), Singapore (`SG`), Indonesia (`ID`), Philippines (`PH`), and India (`IN`)**.
- **WAF Rule #2: Block 10+ Asian Regions** Expanded the geo-blacklist across APAC—only to trigger a global rotation within 1 hour.



Why Geo-Blocking Failed (3/4): Global Proxy Mesh (120+ Countries)

Wave 3: Worldwide Escalation

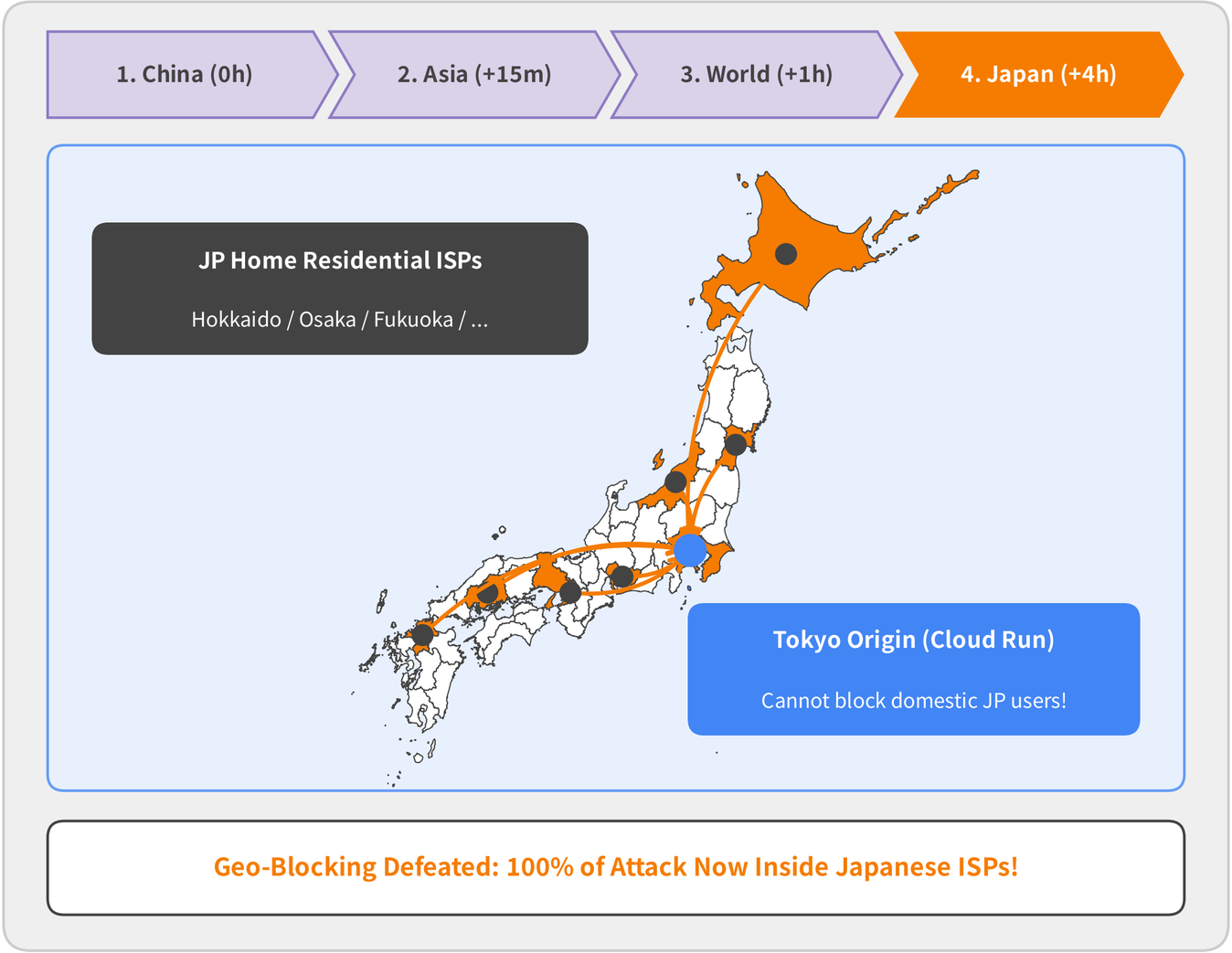
- **120+ Countries Activated (+1h)** Once Asian regions were blocked, the botnet unleashed a global residential proxy pool across the Americas, Europe, Africa, and Oceania.
- **Country Blacklisting Exhausted** Adding individual countries became futile as traffic arrived from `US`, `BR`, `DE`, `GB`, `TR`, `ZA`, and `AU`.
- **WAF Rule #3: Allow ONLY Japan (`JP`)** Inverted our WAF policy to block the entire world outside Japan—buying just 4 hours.



Why Geo-Blocking Failed (4/4): Domestic Japanese Proxies

Wave 4: Inside the Perimeter

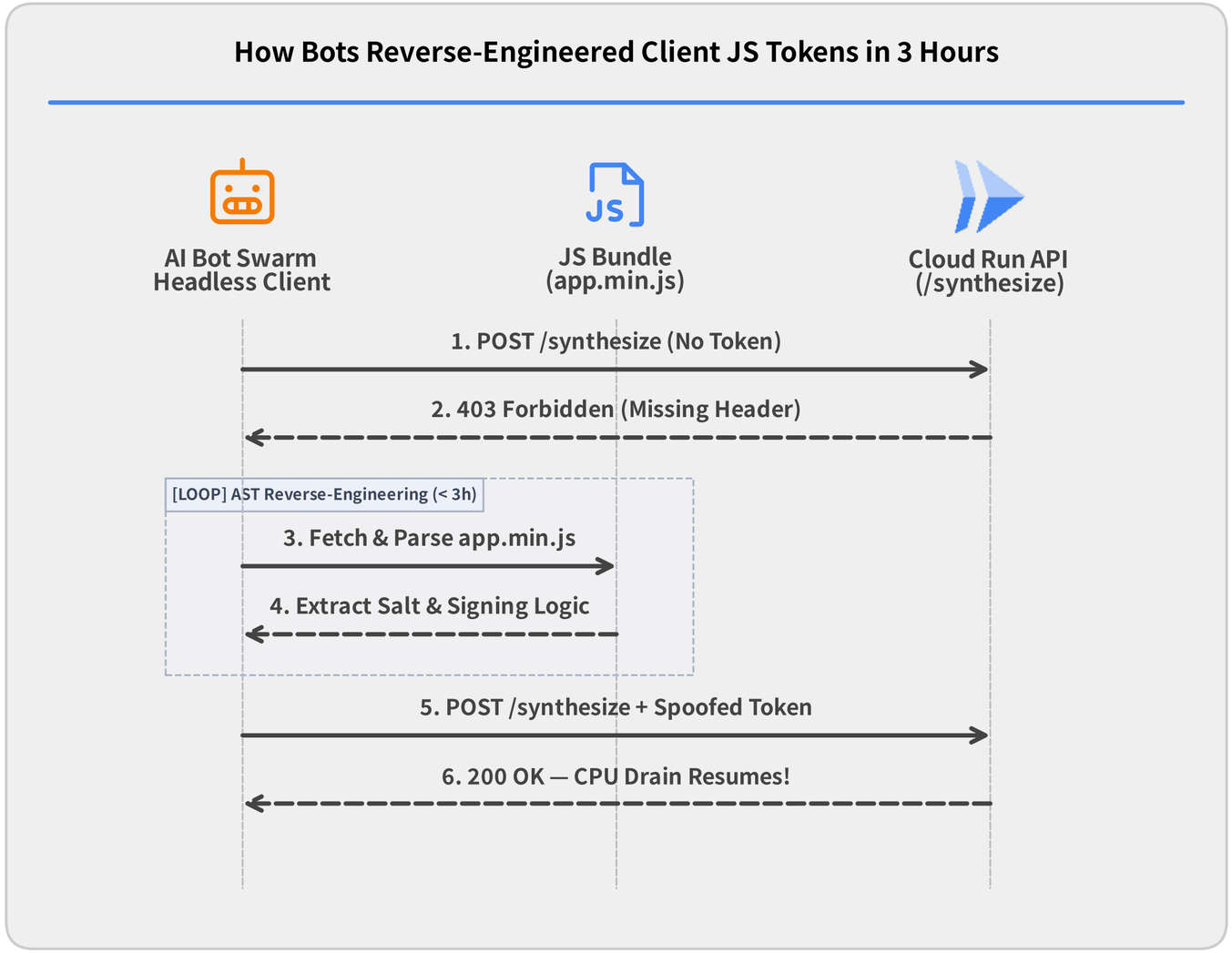
- **100% Domestic Japanese IPs (+4h)** Within 4 hours of enabling our Japan-only whitelist, the flood returned at full scale—originating entirely inside Japan.
- **Compromised Home Routers & Proxies** Attack traffic routed through domestic consumer fiber ISPs across **Hokkaido, Miyagi, Aichi, Osaka, Hiroshima, and Fukuoka**.
- **Geo-Blocking Defeated** Blocking Japanese IPs would lock out our own real users—forcing us to move up to application-layer defenses.



Application-Layer Tactics: Static JS Token Spoofing

Why "Secret" JS Logic Failed

- **Custom Header Verification** Added obfuscated frontend JS to compute `X-Client-Signature` on `/synthesize` calls.
- **3 Hours of Relief** Simple HTTP flood bots were dropped immediately (`403 Forbidden`).
- **Automated AST Reverse-Engineering** Within **3 hours**, the attacker parsed our JS bundle and replicated the exact HMAC algorithm in their headless bot scripts.



ACT / CHAPTER

03

Act 3 of 4

SHA-256 PoW & The EDoS Pricing Trap

How Stopping TTS Abuse Triggered a 300M/Day Flood & $6,750/mo WAF Crisis

3.1 Asymmetric SHA-256 Proof-of-Work & Attacker Goal Shift

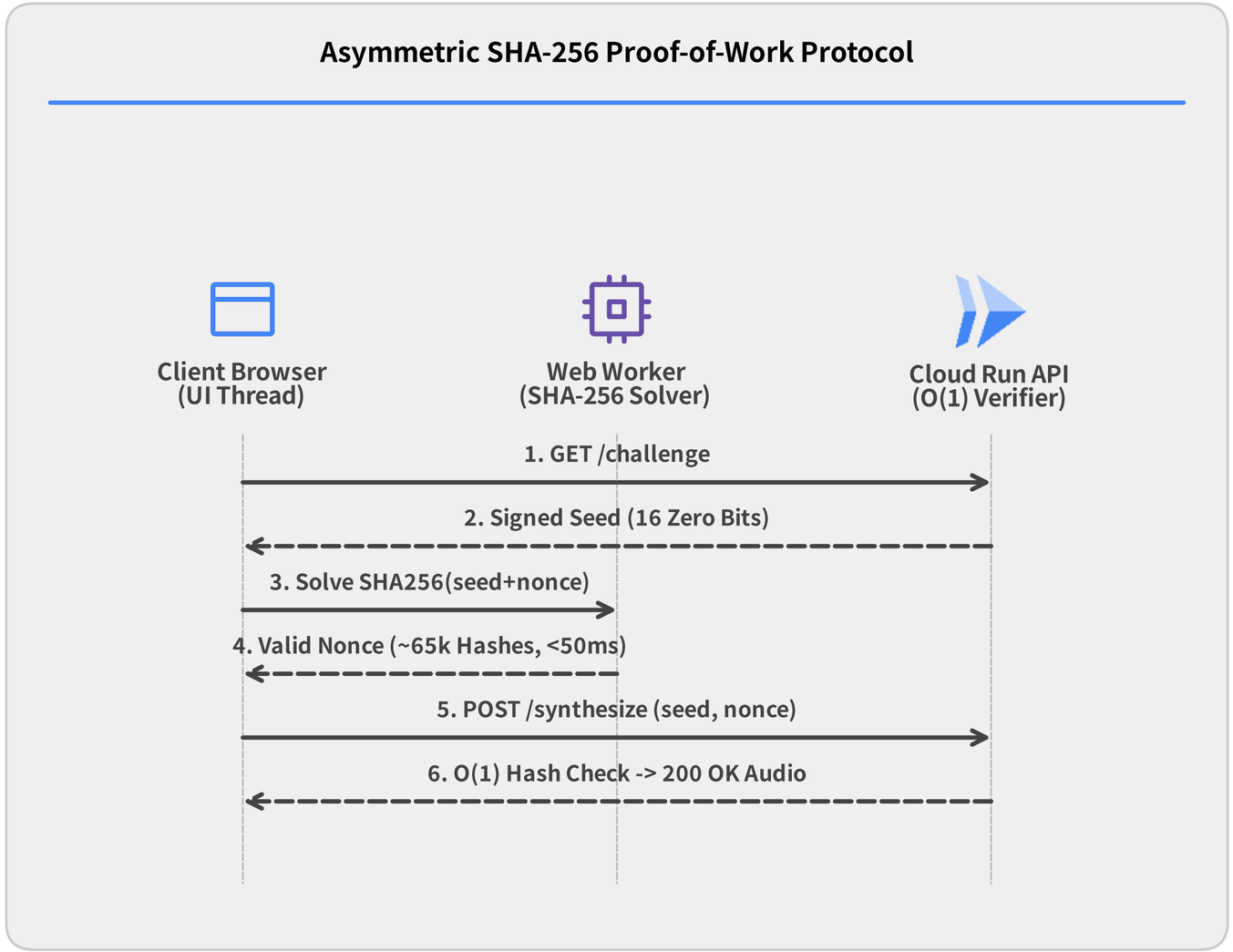
3.2 Cloud Armor Trap: USD 1/Day Origin vs. USD 225/Day WAF

3.3 Midnight Emergency GCLB Teardown & Flat-Rate WAF Choice

Asymmetric Defense: SHA-256 Proof-of-Work

Cryptographic Cost Inversion

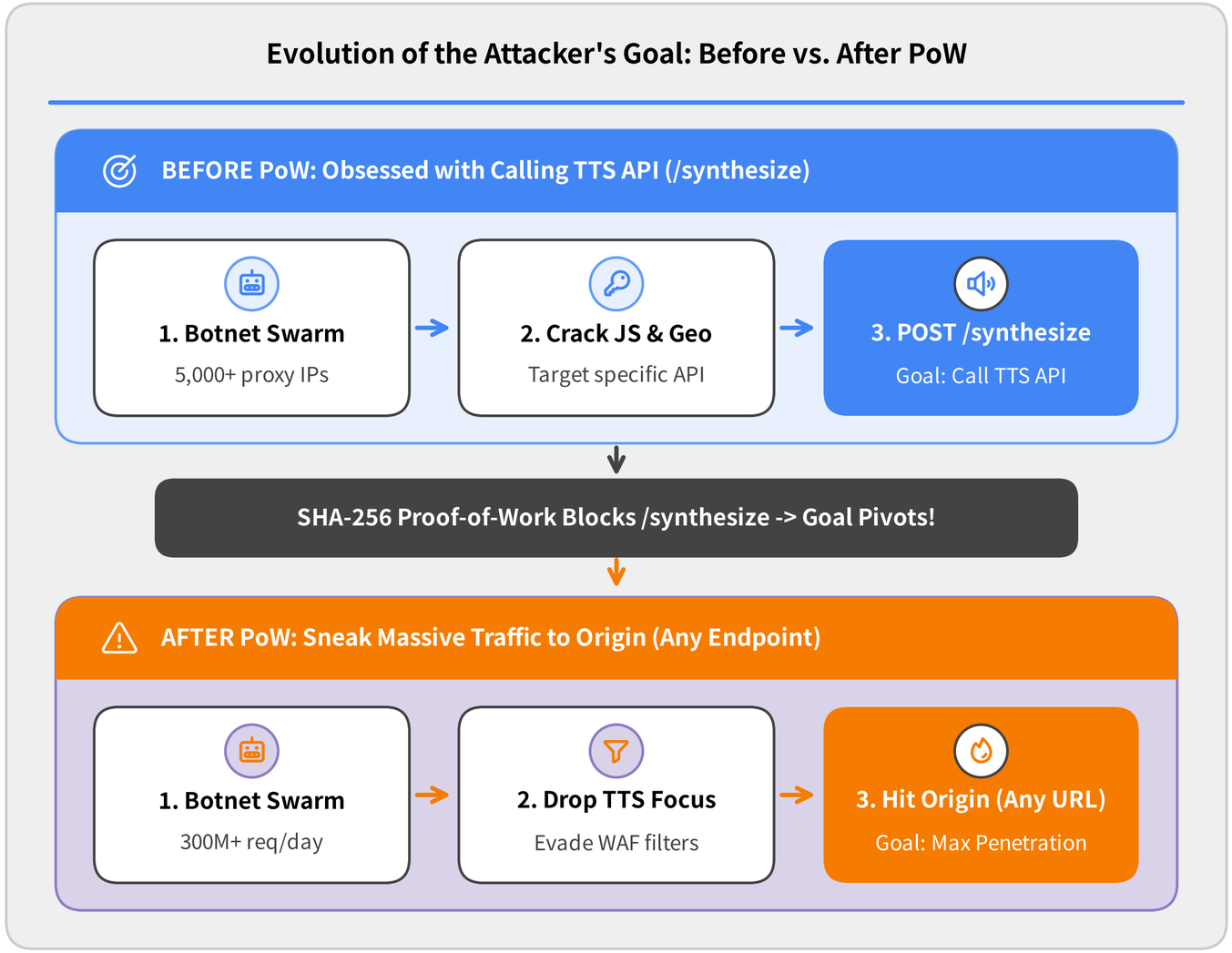
- **One-Time Challenge Seed** Server issues an HMAC-signed `seed` with a short TTL via `GET /challenge`.
- **Browser Web Worker (<50ms)** Client finds a `nonce` where `SHA256(seed + nonce)` starts with `0000...` (~65k hashes).
- **100% Synthesis Protection** Server verifies in **O(1)** (`1 hash`), while 1,000 RPS costs the botnet **65M+ hashes/sec**.



The Turning Point: How PoW Shifted the Attacker's Goal

A Shift in Adversary Objective

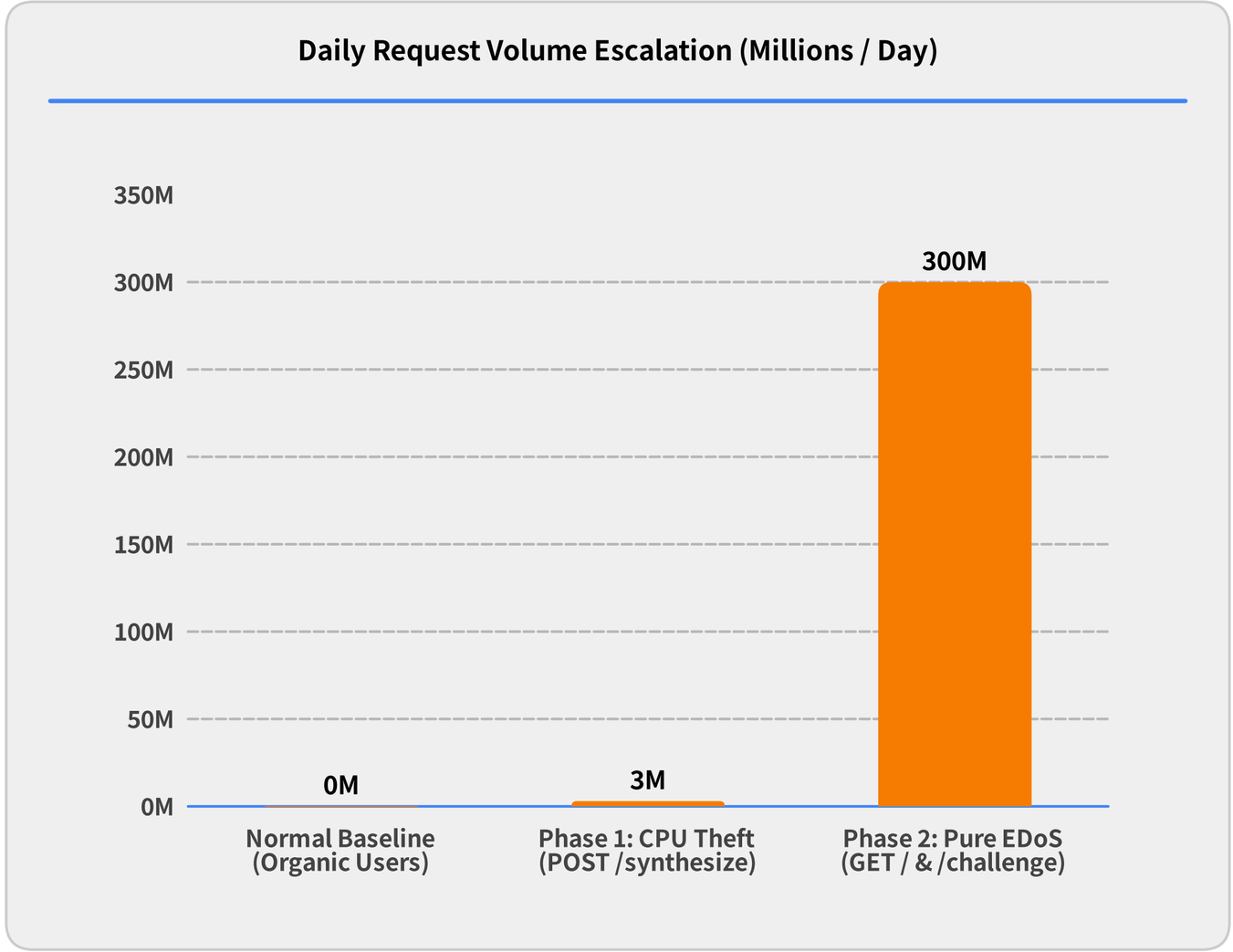
- **Before PoW: Focused on Calling TTS API** Up to this point, every tactic (4 geo-waves & JS cracking) aimed specifically at calling `POST /synthesize`.
- **SHA-256 PoW Closed the Synthesis Door** Requiring 65,000 hashes per call made mass TTS generation computationally infeasible.
- **After PoW: "Just Reach the Origin"** No longer fixated on the TTS API, the attacker's new goal became sneaking massive request volume past our defenses to hit the Origin on *any* endpoint.



The Attacker's Pivot: From Resource Theft to Pure EDoS

Strategic Shift in Attack Vector

- **Phase 1 Blocked (`POST /synthesize`)** SHA-256 PoW made unauthorized TTS generation computationally infeasible.
- **Phase 2 Retaliation (`GET /challenge` & `/`)** Botnet pivoted to unauthenticated endpoints that require **zero PoW**.
- **300 Million Requests / Day** A **100x volumetric surge** (~3,500–10,000 RPS) designed purely to inflate cloud request & WAF bills.



Cloud Billing Log: How Cloud Armor Became the #1 Cost

39.6%

#1 Bill: Cloud Armor

WAF fees beat all compute!

99.9%

Edge Block Rate

Origin CPU stayed ~$1/day

$225/d

Peak WAF Fee Spike

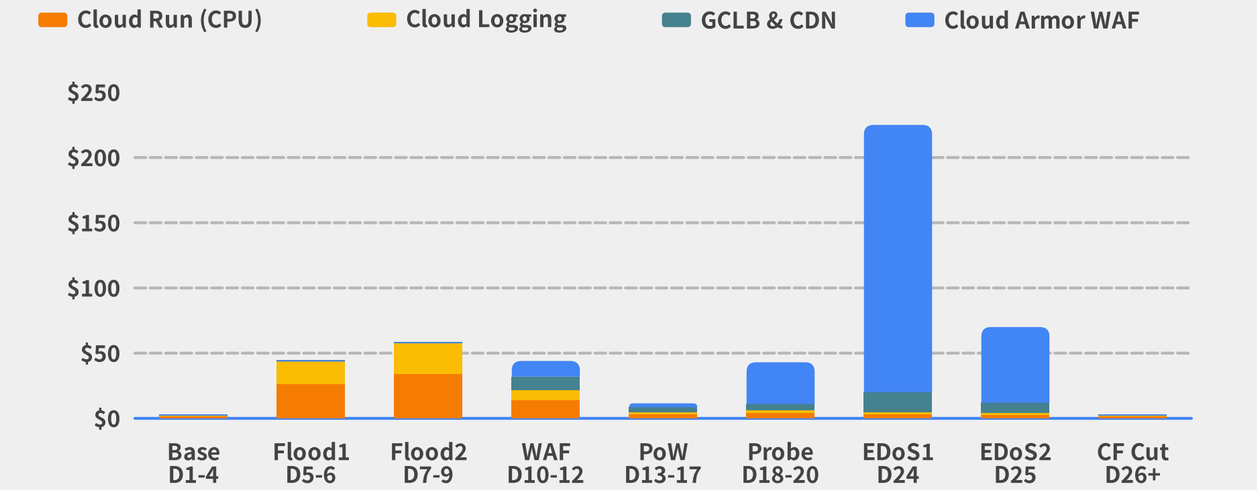
$0.75/1M reqs during EDoS

98% ↓

Post-CF Cutover

GCP bill dropped to flat base

Daily GCP Billing by Service Across Incident Phases ($ USD / Day)



Monthly GCP Billing Breakdown by Service (Ranked by Total Cost)

GCP Service	Primary Cost Driver	Peak Phase	Bill Share
1. Google Cloud Armor	$0.75 / 1M evaluated reqs	D24-25 EDoS Spike	39.6% (#1)
2. Cloud Run (Compute)	Uncached TTS vCPU & egress	D5-9 Direct Flood	34.4% (#2)
3. Logging + GCLB/CDN	Flood access logs & L7 LB	D5-9 & D10-25	26.0% (18%+8%)

Midnight Emergency: Tearing Down GCLB & Choosing a Flat-Rate WAF

Midnight Emergency Cutover Sequence (Stopping the $225/Day Bleeding)



Architectural Comparison: Pay-Per-Request Cloud WAFs vs. Flat-Rate Edge Proxy

Defense Option	Base Fee	Per-Request WAF Fee	Cost @ 300M/Day	Verdict
Google Cloud Armor	$5/mo + GCLB	$0.75 / 1M evaluated	~$6,750 / mo	REJECT: EDoS Bankruptcy
AWS / Azure WAF	$5–10/mo + ALB	$0.60 / 1M evaluated	~$5,400 / mo	REJECT: Per-Req Trap
Self-Hosted Nginx VPS	$10/mo fixed	$0 (Fixed Bandwidth)	$10 / mo	REJECT: 1 NIC Saturates
Cloudflare Pro (Flat)	**$20/mo FLAT**	**$0.00 (Unmetered)**	**$20 / mo FLAT**	**SELECTED: Zero EDoS Risk**

ACT / CHAPTER

04

Act 4 of 4

Multi-Layered Edge Victory

Traffic Forensics, 5 Defense Pillars, and AI Adversary Profiling

4.1 Cloudflare Pro Migration & Sub-Threshold Bot Leak (25%)

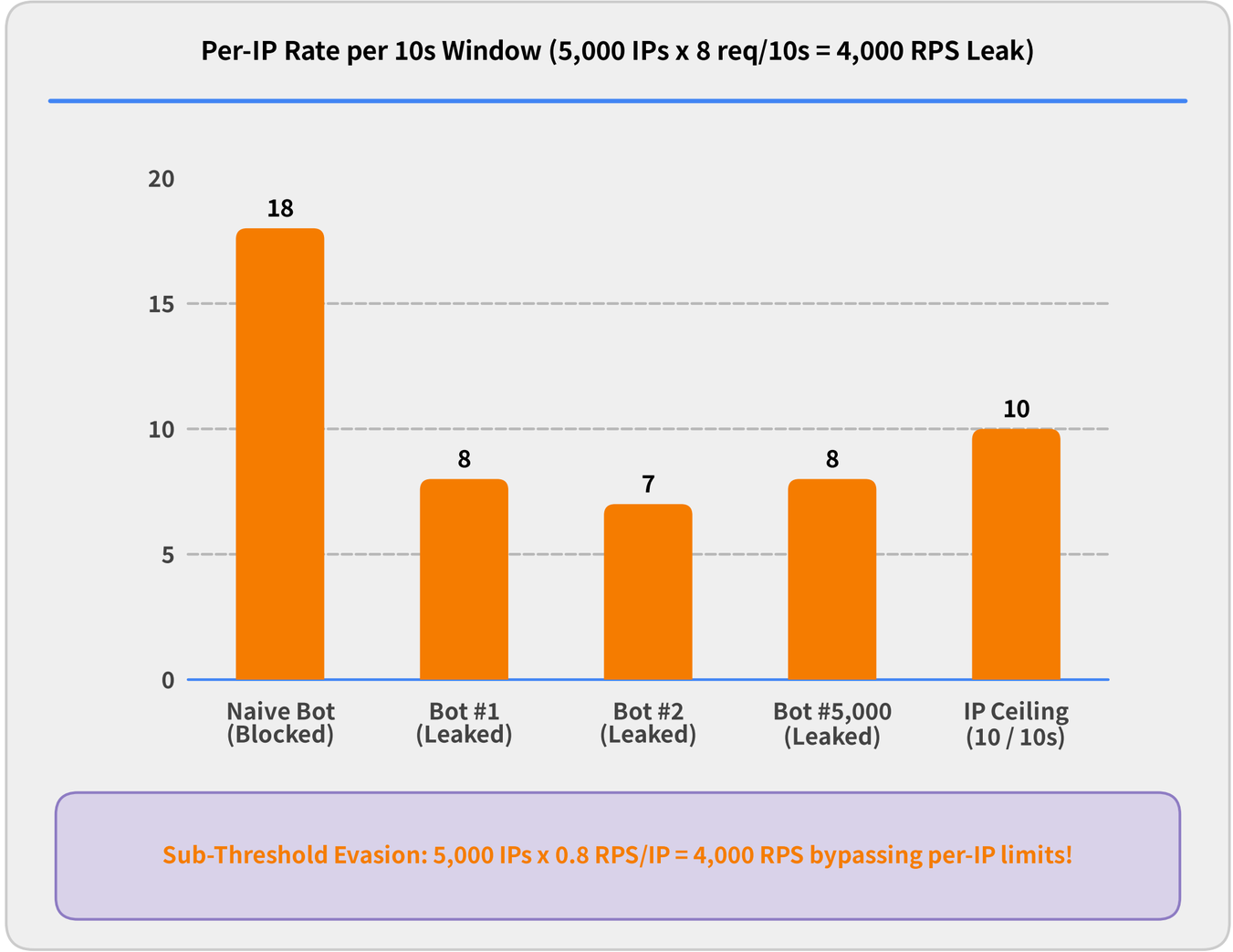
4.2 Traffic Forensics: Cloud VMs vs. Home Residential Proxies

4.3 5-Layer Defense Architecture, WAF Rules & AI Bot Profile

Cloudflare Pro Migration & The 25% Sub-Threshold Leak

Why Per-IP Rate Limits Leaked

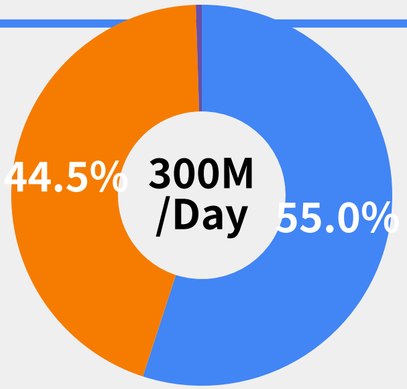
- **Flat $20/mo Cost Secured** Cut over DNS to Cloudflare Pro; 75% of the volumetric flood was blocked immediately at $0 overage.
- **10 Req / 10s Per-IP Threshold** Setting stricter per-IP limits risked blocking legitimate Japanese offices & carrier NATs.
- **Distributed Sub-Threshold Swarm 5,000+ bot IPs** throttled down to **6–8 req / 10s each**, leaking **4,000 RPS (25%)** to origin!



Traffic Forensics: Decomposing the Two Botnet Fleets

- **Log Forensics Breakthrough**: Inspecting Cloudflare request logs by **ASN**, **HTTP Protocol Version**, and **User-Agent** revealed the attack was actually **two distinct botnet fleets**.
- **Protocol Mismatch Tell**: Fleet B spoofed modern Chrome/Safari User-Agents (`Chrome/134`), but its proxy software spoke **HTTP/1.1** (real browsers always negotiate **HTTP/2 or HTTP/3**!).

Attack Fleet Breakdown



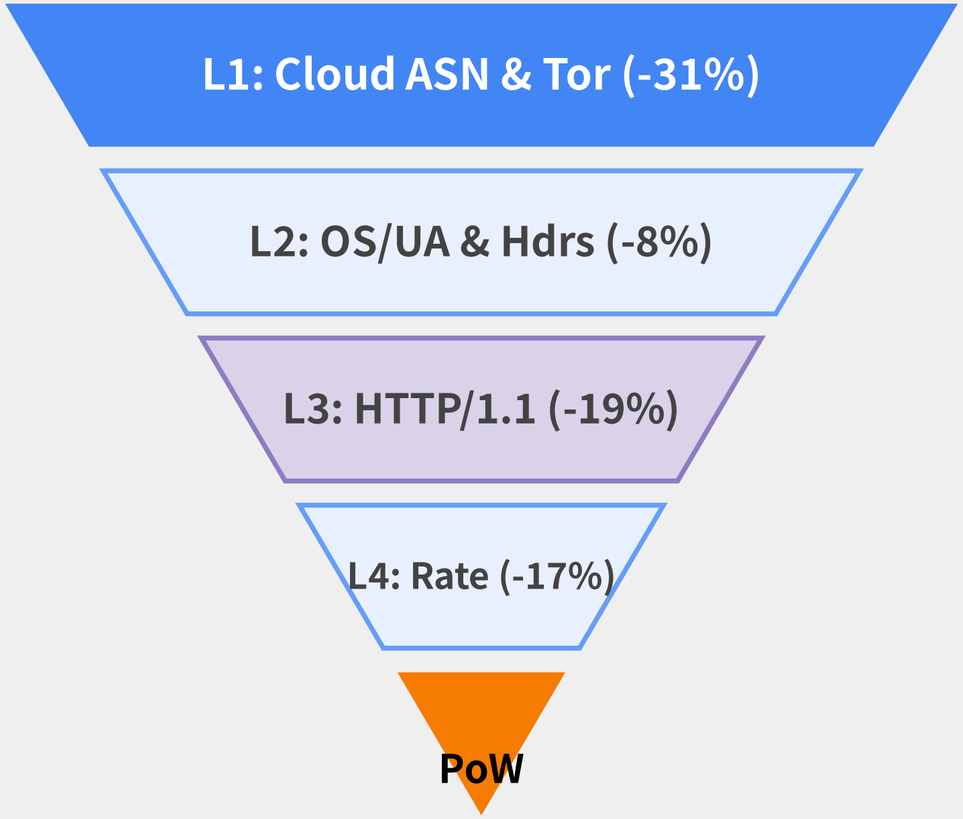
- **Fleet A: Cloud ASNs (55%)**
- **Fleet B: Home HTTP/1.1 (44.5%)**
- **Real Users: HTTP/2+3 (0.5%)**

Forensic Fingerprint Comparison: Bot Fleets vs. Real Users

Signal	Fleet A: Cloud VMs	Fleet B: Home Proxies	Real Users (0.5%)
Network ASN	AWS / GCP / OVH	JP Home Fiber ISPs	JP Consumer ISPs
Rate per IP	Burst (20–100 / 10s)	Stealth (6–8 / 10s)	1–3 req / min
User-Agent	python / Go / Old UA	**Spoofed Chrome 130**	Chrome / Safari
HTTP Version	HTTP/1.1 or HTTP/2	**HTTP/1.1 ONLY!**	HTTP/2 or HTTP/3
WAF Action	Block Hosting ASNs	Block HTTP/1.1	Zero-Friction Pass

The 5-Layer Defense-in-Depth Architecture

5 Orthogonal Defense Layers: Filtering Multi-Vector Bot Swarms Down to Clean Origin Traffic



L1: Network ASN & Tor Filter (~31% of Blocks)
Drop Cloud/VPS ASNs + Tor & Anon proxies (T1, A1, XX)

L2: Fingerprint & Header Hygiene (~8% of Blocks)
Drop unknown OS/UA, bad Sec-Fetch hdrs & /cdn-cgi/trace

L3: Wire Protocol Enforcement (~19% of Blocks)
Block legacy HTTP/1.1 on /api/*; require HTTP/2 or HTTP/3

L4: Dual-Speed Edge Rate Limit (~17% of Blocks)
Strict 5 req/30s on /api/challenge + 15 req/30s global cap

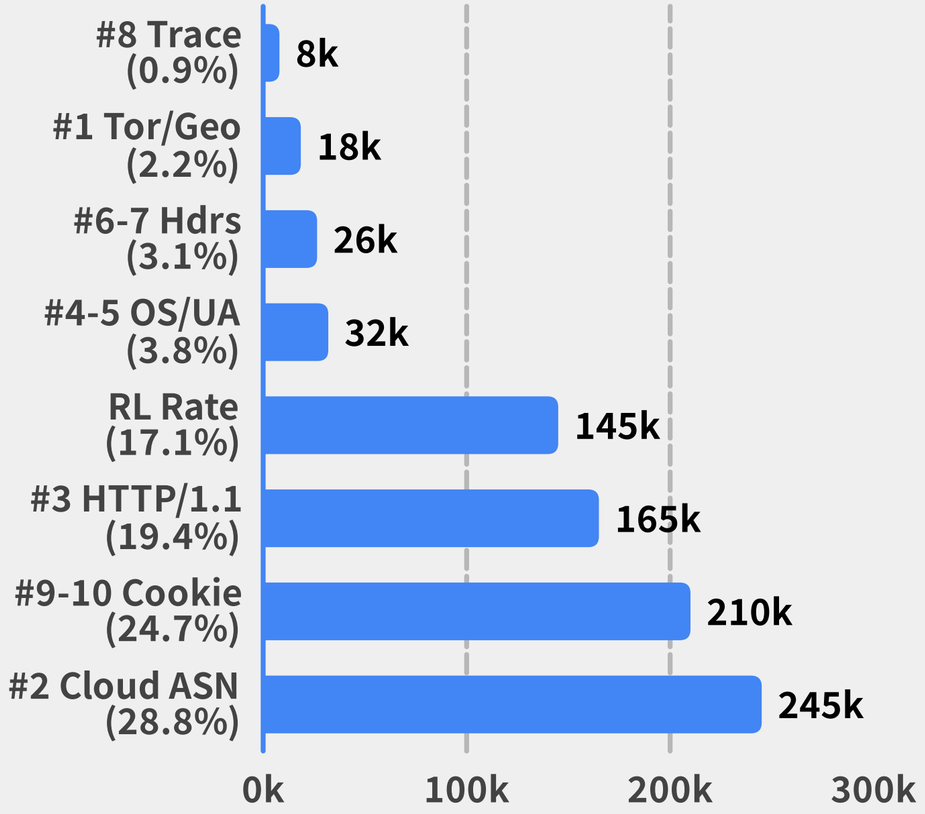
L5: Page Challenge + Cookie Gate + PoW (~25%)
Page Managed Challenge -> cf_clearance on /api/* + PoW

Composite WAF Rules: 10 Custom Rules & 2 Rate Limits

Production WAF Rule Chain (10 Custom + 2 Rate Limits)

Rule Layer	Match Condition	Action	Events
#1 Tor / Anon Geo	Country in {T1, A1, XX}	Block	18.5k
#2 Cloud ASNs	ASN in {Cloud / VPS}	Block	245.0k
#3 HTTP/1.1 API	URI /api/* & HTTP/1.1	Block	165.0k
#4-5 OS / Browser	URI /api/* & UA Anomaly	Block	32.0k
#6-7 Client Hdrs	URI /api/* & Sec-Fetch	Block	26.5k
#8 Trace Recon	URI == /cdn-cgi/trace	Block	8.0k
#9-10 Chal+Cookie	Page Chal -> API Cookie	Chal+Blk	210.0k
RL #1-2 Rate Cap	5/30s (/chal) & 15/30s	Block	145.0k

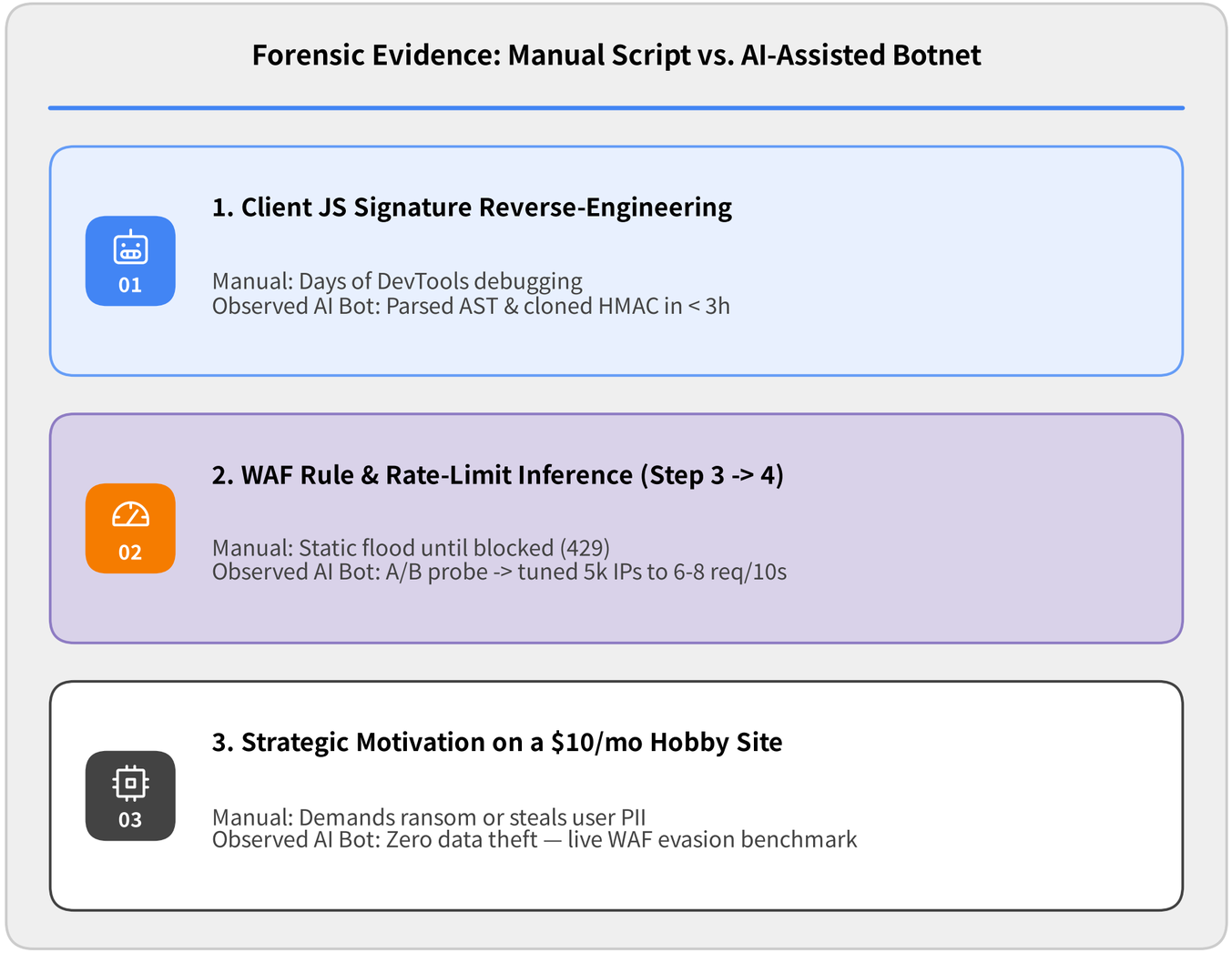
Mitigated Events by Rule (850k Total)



Threat Actor Profile: Anatomy of an AI-Assisted Adversary

Why Their Cycle Ran So Fast

- **LLM-Speed JS Cracking (<3 Hours)** De-obfuscated our custom frontend JS bundle and replicated HMAC signing in under 180 minutes.
- **Automated A/B Threshold Probing** Binary-searched our WAF rate rule and tuned 5,000+ nodes to `6-8 req/10s` (just below `10 req/10s`).
- **Adversarial Training Sandbox** Used our low-stakes hobby service as a live proving ground to train automated WAF-evasion agents.



CLOSING

05

Summary & Wrap-Up

Conclusion & Key Takeaways

4 SRE Golden Rules & Final Defense Metrics

Summary: 4 Golden Rules for Serverless DDoS Defense

4 SRE Golden Rules for Serverless & Edge Security

Rule 1

No Direct Serverless

Lock run.app to LB only;
place flat-rate Edge in front

Rule 2

Audit WAF EDoS Fees

$0.75/M WAF fees bankrupt
you at 300M req/day flood

Rule 3

Orthogonal Signals

Combine Cloud ASN, UA/OS,
HTTP/2+ ALPN & Rate Caps

Rule 4

Asymmetric PoW

JS secrets crack in hours;
use SHA-256 Web Worker PoW

100%	**Service Availability** Zero downtime for JP users
0%	**Unauthorized TTS CPU** SHA-256 PoW stopped abuse
99.9%	**Edge Drop Efficiency** 5-layer rules block 300M/d
$20/mo	**Predictable Flat Cost** Down from $6,750/mo WAF bill

THANK YOU • Q&A

Thank You!

Questions & Discussion

Surviving a 300M+ Req/Day DDoS & EDoS Attack on Google Cloud Run & Edge WAF