Architectural VPN Vulnerabilities, Disclosure Fatigue, and Structural Failures

William J. Tolley[†‡§], Everett Morse[†], Gabriel Hogan[¶], Jeffrey Knockel[⋆], and Jedidiah R. Crandall[‡§]

[†]Hampden–Sydney College [‡]Arizona State University [§]Breakpointing Bad
[¶]Washington & Lee University [⋆]Bowdoin College

Why This Matters

- VPNs are widely recommended for:
  - Circumventing censorship
  - Avoiding surveillance
  - Protecting journalists and dissidents
- Used in authoritarian contexts (e.g., Russia's TSPU deployment)
- Assumed to provide confidentiality and unlinkability
- If VPNs leak metadata, that assumption collapses

Two Facts

- VPNs today remain vulnerable to blind in/on-path inference attacks
- We have known this since 2019
- Multiple CVEs assigned
- Multiple vendor patch cycles
- Exploit primitives still reproducible in 2025

Motivation

- Architectural vulnerabilities differ from implementation bugs
- They span vendors, protocols, and operating systems
- They resist patch-and-close workflows
- Current disclosure systems assume discrete, ownable defects

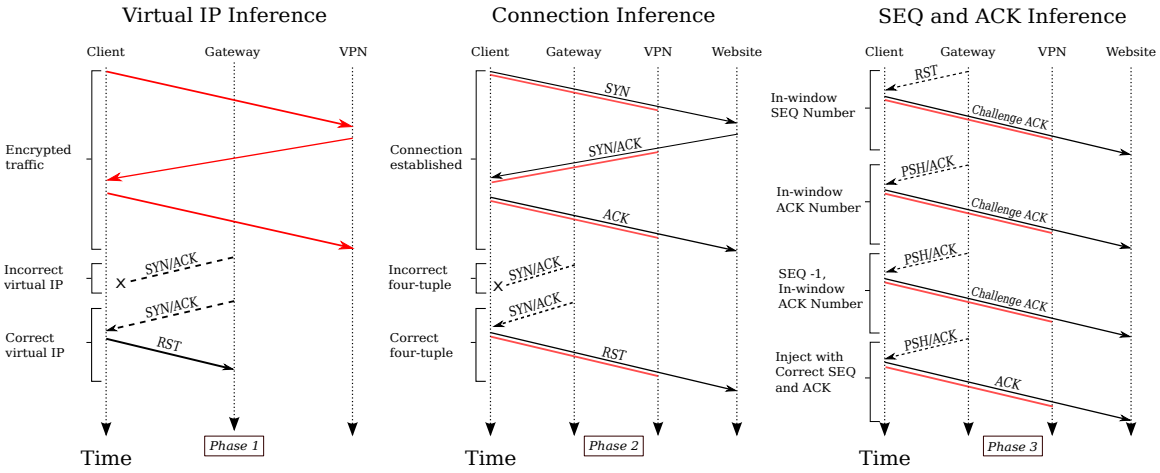
Central Question

- What happens when an architectural vulnerability persists?
- How does the CVE ecosystem represent cross-vendor risk?
- Who owns systemic conditions?
- What does “fixed” mean for architecture?

Scope of This Talk

- Technical attack published at USENIX Security 2021
  - Disclosed in 2019
- Packet-level mechanics well documented
- Blog posts and artifacts available
- This paper focuses on:
  - Longitudinal disclosure
  - Vendor dynamics
  - Structural gaps

Client-Side Blind In/On-Path Attack



Why the Server-Side Variant Matters

- Does not require malicious WiFi
- Works from in-path ISP or state vantage
- No vendor mitigation published
- No CVE tracks this architectural condition
- Remains unacknowledged

Architectural Characteristics

- No cryptographic break
- No memory corruption
- No misconfiguration
- Exploits:
  - Metadata exposure
  - Predictable NAT behavior
  - Deterministic tunnel responses
- Nothing is broken; the system behaves as designed

Systemic Properties

- Cross-vendor
- Cross-protocol
- Cross-operating-system
- Cross-layer (OS + routing + VPN semantics)
- Persists across patch cycles
- Cannot be resolved in a single codebase

From Architecture to Governance

- Architectural flaws do not map cleanly to CVEs
- CVE assumes:
  - Discrete defect
  - Single product ownership
  - Patch $\rightarrow$ closure
- Architectural risk violates those assumptions

Related Architectural VPN Vulnerabilities

- TunnelCrack — Nian Xue et al.
  - Multiple CVEs
  - Routing exceptions leak VPN traffic
- Port Shadows — Ben Mixon-Baca et al.
  - Connection tracking inference
  - Persistent cross-vendor behavior
- Architectural patterns recur across protocols

2019: Initial Disclosure

- Vulnerability disclosed to major VPN providers and OS vendors
- Public disclosure on oss-security
- CVE-2019-9461 (Android)
- CVE-2019-14899 (Linux and Unix-like systems)
- Server-side variant deemed ineligible for CVE assignment

Early Vendor Responses

- Some vendors issued partial mitigations
- Others characterized issue as “out of scope”
- Responsibility often attributed to OS behavior
- No party claimed ownership of the architectural condition

2021: Peer-Reviewed Publication

- USENIX Security 2021 paper
- Formalized client- and server-side variants
- Emphasized architectural, cross-layer nature
- Established that vulnerability was not implementation-specific

Post-Publication Re-Engagement

- Vendors re-contacted with refined proofs-of-concept
- Some reopened triage discussions
- Institutional memory often absent
- Disclosure process effectively restarted

2021–2024: Re-Testing and New CVEs

- Re-tested on fully updated devices
- Vulnerability remained exploitable
- CVE-2024-49734 assigned
- Multi-year delay between report and bulletin publication

2025: Fully Patched Systems

- Pixel 10 Pro XL
- Android 16 with October 2025 security updates
- Same inference primitives successful:
  - Internal IP discovery
  - Connection-state inference
  - TCP reset injection
- New CVE pending

Illustrative Results Across Tools

- OpenVPN — vulnerable
- WireGuard — vulnerable
- NordWhisper — vulnerable
- Lantern — vulnerable
- Psiphon — vulnerable
- Protocol choice did not eliminate attack primitives

Administrative Evolution vs Technical Persistence

- Multiple CVEs assigned
- Multiple patch cycles
- Security bulletins published
- Media coverage shifted narratives
- Core architectural behavior unchanged

Vendor Response Patterns

- Patch-and-close workflows applied to architectural risk
- Partial mitigations scoped to specific platforms
- Server-side variant left untracked
- Divergent interpretations of responsibility

Key Observation from Seven Years

- Vulnerability evolved administratively
- Exploitability persisted technically
- No durable cross-vendor ownership emerged
- Architectural risk remained visible only through re-testing

Five Lessons from Seven Years

- Architectural flaws defy CVE closure
- Institutional memory decays over multi-year timelines
- Media narratives shape perceived severity
- Partial mitigations regress or fragment
- At-risk users lack persistent visibility tools

Lesson 1: Architectural Flaws Defy Closure

- CVE assumes discrete defect
- Patch $\rightarrow$ resolution
- Architectural risk spans implementations
- “Closed” CVEs do not imply eliminated exploitability

Lesson 2: Institutional Memory Decays

- Multi-year vulnerabilities outlive triage teams
- Re-reporting resets disclosure workflows
- Context and prior understanding are lost
- Knowledge accumulation is not durable

Lesson 3: Narrative Shapes Outcome

- Early vendor framing influences media coverage
- Media narratives influence reviewer perception
- Architectural risk can be minimized rhetorically
- Absence of durable reference point enables distortion

Lesson 4: Mitigations Decay Over Time

- Fixes scoped to specific builds or configurations
- Regressions across releases and forks
- Partial filtering or routing adjustments insufficient
- Persistence masked by patch cycles

Lesson 5: Users Lack Persistent Visibility

- CVE records are version-scoped and vendor-scoped
- Users cannot easily determine current exploitability
- “Fully patched” does not mean architecturally safe
- High-risk users need longitudinal transparency

What Is Missing in Current Infrastructure

- No cross-vendor architectural identifier
- No mechanism for longitudinal exploitability tracking
- No human-centered impact model
- No durable public ledger of persistence

A Sketch of a Framework

- Not a replacement for CVE
- Not a complete solution
- A minimal set of orthogonal dimensions
- A starting point for community discussion

Dimension 1: Persistent Cross-Vendor Identity

- Single identifier spanning implementations
- Survives patch cycles and version changes
- Tracks architectural behavior, not code defects

Dimension 2: Human-Centered Impact

- Focus on harm potential, not CVSS
- Can it reveal tool usage?
- Can it infer access to censored services?
- Can it enable disruption or punishment?

Dimension 3: Longitudinal Verification

- Timestamped re-testing history
- Independent confirmation hooks
- Explicit record of persistence or regression
- Counteracts closure bias

Dimension 4: Structured Public Interface

- Clear layperson impact summaries
- Vendor status matrices
- Shareable language for advocates and trainers
- Reduces narrative distortion

Dimension 5: Public Transparency

- Open and searchable entries
- Configuration-level relevance
- Integration with tool documentation
- Enables user self-assessment

Relationship to CVE

- CVE remains valuable for implementation flaws
- Architectural risk exceeds patch-based models
- Complementary infrastructure is necessary
- Augment, do not replace

Call to Action

- Researchers: track persistence beyond publication
- Internet Freedom community: treat systemic behavior as threat surface
- Vendors: acknowledge architectural limitations
- Build shared institutional memory

Conclusion

- Seven years of disclosure
- Multiple CVEs and patch cycles
- Persistent exploit primitives
- Architectural vulnerabilities require architectural memory