Iran's Multi-Layer Internet Censorship Architecture

A Technical Whitepaper for Network Engineers, ISPs, and Measurement Researchers

---

Executive summary

This whitepaper describes the architecture and observable behavior of a modern, multi-layer Internet censorship system as deployed in Iran. It focuses on mechanisms that affect network engineers, ISPs, and measurement practitioners:

- Centralized, state-controlled filtering at national chokepoints.
- DNS interference (poisoning, forging, and hijacking).
- HTTP-level Host and keyword filtering with packet injection.
- TLS SNI-based blocking and connection resets.
- Protocol whitelisting and silent dropping of disallowed protocols.
- Selective throttling of encrypted traffic.
- Stealth shutdown patterns that preserve BGP reachability while cutting off practical access.

---

1. Background

1.1 Topology and centralization

Iran's Internet topology is highly centralized:

- **Upstream control:** International connectivity is funneled through a small number of upstream providers and gateways.
- **Domestic aggregation:** Commercial ISPs and mobile operators typically rely on these upstreams, creating a small number of natural chokepoints.
- **National intranet:** A large private addressing domain (e.g., 10.0.0.0/8) is used internally for services, middleboxes, and block pages.

From an engineering perspective, this centralization implies:

- A relatively small set of locations where middleboxes can be deployed to enforce policy.

- High consistency of filtering behavior across different ISPs, because they share the same upstream points of control.

1.2 Evolution of censorship mechanisms

The censorship stack evolved in phases:

1. **IP-based filtering:** Static blocking of IP prefixes associated with disfavored services.
2. **DNS interference:** Interception and forging of DNS responses, particularly for high-profile domains.
3. **HTTP content filtering:** DPI on HTTP requests, matching Host headers and URL paths.
4. **TLS/SNI inspection:** Blocking based on TLS ClientHello SNI, without decrypting application payload.
5. **Protocol whitelisting:** Dropping or throttling traffic that is not recognized as DNS/HTTP/HTTPS.
6. **Traffic engineering:** Throttling, prioritization of domestic services, and sometimes selective reachability to international networks.

This layered approach allows the censor to choose between coarse and fine-grained control, depending on the target and the political or operational context.

---

2. Architecture of the filtering system

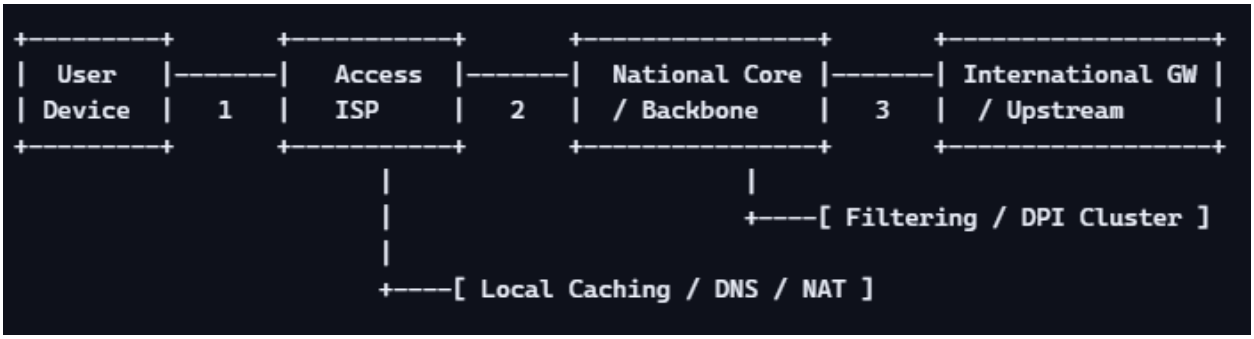
2.1 High-level data path

At a high level, traffic from end users traverses:

1. **Access network:** Last-mile ISP (mobile operator, DSL provider, WISP, etc.).
2. **Aggregation and BRAS/BNG layers:** Regional aggregation and authentication points.
3. **National backbone:** Carriers and peering nodes that connect regions.
4. **International gateways:** Limited set of upstream links to foreign networks.

Filtering middleboxes can be located in any of these layers, but in practice the most impactful are at or near national backbones and international gateways.

A simplified ASCII view:



2.2 Placement of censorship middleboxes

Measurements and traceroute-based localization (including TTL-limited probes and private address observations) suggest that:

- Filtering nodes are often located **just upstream of the last domestic hop before international transit**.
- The same private address ranges (e.g., certain 10.x.x.x hops) appear across multiple ISPs when traffic is blocked, indicating shared infrastructure.
- Some DNS poisoning is performed **close to or at the recursive resolver's upstream**, not at the user's local gateway.

The practical implication: a single policy change at the central cluster propagates instantly to all downstream ISPs.

---

3. Technical analysis of censorship mechanisms

3.1 DNS interference

3.1.1 Poisoning and redirection

A common censorship pattern is to respond to DNS queries for blocked domains with:

- **Private IPv4 addresses**, e.g.:
  - 10.10.34.34
  - 10.10.34.35
  - 10.10.34.36
- **Private IPv6 addresses**, e.g.:
  - d0::11

These addresses typically host block pages or terminate traffic at controlled endpoints. From the client perspective:

- The DNS response is syntactically valid (proper header, flags, and answer count).
- The TTL is often unusually low, suggesting dynamic or inline generation.
- Answers for the same domain may differ between "clean" resolvers (e.g., DoH/DoT) and local resolvers.

3.1.2 Injection behavior

Typical behaviors:

- **UDP DNS queries** are intercepted in-path and receive forged answers, even when sent to public resolvers (e.g., 8.8.8.8 or 1.1.1.1).
- **TCP DNS queries** may bypass interference in some configurations, because they are less commonly handled by the same injection logic.
- Remote resolvers may receive **spoofed TCP RSTs** to terminate connections from the censor's side, preventing the legitimate response from returning to the client.

3.1.3 Detection signatures

From an engineering perspective, you can look for:

- **Inconsistent answers** between:
    - UDP vs TCP queries to the same resolver.
    - Local resolvers vs DoH/DoT endpoints.
- **Presence of known censorship IPs** (e.g., 10.10.34.34) in answers.
- **Abnormal TTLs** and unusually fast responses when querying "far" servers.

3.2 HTTP Host and keyword filtering

3.2.1 Host header filtering

DPI devices inspect HTTP request headers, particularly:

- Host: header
- Request path and query string

If the target domain or path matches a blocklist:

- The censor may inject a **HTTP 403** response.
- The TCP connection may be terminated with a **RST** packet.
- A redirect or block page may be served from a private IP.

Some filters exhibit **case sensitivity** in string matching, which can lead to partial or inconsistent bypasses (e.g., Host: YouTube.com vs Host: youtube.com).

3.2.2 Inline injection

When a request hits a blocked domain:

- The user sends GET / HTTP/1.1 with Host: blocked.example.
- The censor's middlebox immediately injects a 403 or block page before the upstream server replies.
- The original request may still reach the upstream, but the client's TCP stream is already closed or poisoned.

Signs in packet captures:

- The block page or 403 response arrives **faster** than a round-trip to the real server would allow.
- The source IP of the injected response may be:
    - The actual server IP (spoofed), or
    - A private or intermediate IP if the censor does not fully spoof.

3.3 TLS SNI-based blocking

3.3.1 SNI inspection

For TLS connections, the censor inspects the **ClientHello** message, specifically the **Server Name Indication (SNI)** extension. If the SNI matches a blocked domain:

- The censor can send a **TCP RST** to terminate the connection.
- Alternatively, it may silently drop the packet, causing a timeout.

No decryption of application payload is needed; the SNI is in cleartext in the initial handshake.

3.3.2 Observable patterns

Common field observations:

- The connection is reset **immediately after the ClientHello** is sent.
- Changing the SNI to a permitted domain (or using domain fronting where possible) restores connectivity.
- Using IP-only connections without SNI may bypass some rules but often fails due to certificate name mismatch.

3.4 Protocol whitelisting

3.4.1 Allowed vs blocked protocols

The system tends to **whitelist** known, “ordinary” protocols:

- DNS over UDP/53
- HTTP over TCP/80
- HTTPS over TCP/443

Other protocols are often:

- Silently dropped (no RST, no ICMP, just no response), or
- Throttled to unusable levels.

Examples:

- SSH (TCP/22) connections that never complete.
- OpenVPN over UDP/1194 or TCP/443 that stalls during TLS or key exchange.
- WireGuard (UDP/51820) that never receives responses.
- QUIC (UDP/443) that is blocked or heavily throttled.

3.4.2 Engineering impact

For operators and engineers:

- VPN and tunneling protocols that do not closely mimic allowed traffic are highly fragile under such policies.

- Generic VPNs on non-standard ports are easier to detect and drop than HTTPS-camouflaged transports.

3.5 Traffic throttling

3.5.1 Differential treatment of HTTP vs HTTPS

Measurements often show:

- HTTP (plain) traffic achieving significantly higher throughput than HTTPS.
- Under certain events, HTTPS is capped at a fraction of the normal rate, while HTTP remains near normal.

This creates incentives for users to fall back to unencrypted services, which can then be monitored and filtered more easily.

3.5.2 Detection via throughput comparison

Throttle detection strategies:

- Download the **same asset** over HTTP and HTTPS from a neutral CDN.
- Measure per-protocol throughput in parallel.
- Compute the HTTP/HTTPS bandwidth ratio.

If HTTP is consistently much faster (for example, >1.5×), and the path otherwise looks healthy, targeted throttling is likely.

---

4. Measurement methodology

This section outlines how to instrument and detect the above mechanisms using field-deployable tools.

4.1 DNS measurement

4.1.1 Multi-protocol DNS comparison

Tests:

1. **UDP DNS** to:
    - Local resolver (ISP DNS).

    - Public resolvers (8.8.8.8, 1.1.1.1).
2. **TCP DNS** to the same resolvers.
3. **DoH and DoT** to major providers.

Compare:

- IPs returned.
- TTL values.
- Error codes or timeouts.

4.1.2 Hex and header inspection

A dig-style parser (as in your code) can:

- Print raw hex of the DNS response.
- Decode header fields:
    - Transaction ID.
    - Flags (QR, AA, RA, RCODE).
    - Counts (QDCOUNT, ANCOUNT, NSCOUNT, ARCOUNT).
- Parse question and answer RRs, displaying types, TTLs, and RDATA.

This allows detection of:

- Truncated or malformed responses.
- Suspicious RCODEs.
- In-line forging patterns.

4.2 HTTP and HTTPS measurement

4.2.1 Censorship checks

For each target domain:

- Perform plain HTTP and HTTPS GET requests.
- Use realistic User-Agent strings to avoid trivial blocking.
- Optionally override:

    - Host header (for host-based probing).
    - Destination IP (to distinguish name-based from IP-based blocking).

Observe:

- HTTP status codes (403, 301, 302, 503, etc.).
- Connection errors:
    - Timeouts.
    - Resets.
    - TLS handshake failures.

4.2.2 Host and SNI manipulation

By decoupling:

- TCP/IP destination (IP),
- HTTP Host header,
- TLS SNI hostname,

you can test:

- If censorship keys on IP (destination-based).
- If censorship keys on Host and/or SNI (name-based).
- If fronting patterns (e.g., clean SNI but censored Host) behave differently.

4.3 TCP connectivity and reset probing

Using raw TCP tests:

- Attempt handshakes to specified ports.
- Distinguish:
    - **RST during handshake** → active blocking.
    - **Timeout** → potential blackhole or silent drop.
    - **Immediate acceptance but failure after data** → application-layer interference.

A simple connect() loop using socket with custom timeouts and per-destination logging is sufficient to detect these patterns.

4.4 Throttling and performance measurement

4.4.1 Controlled throughput tests

Select:

- A stable asset hosted on a large CDN (e.g., a JS library served over HTTP and HTTPS).

Run:

- Parallel downloads over HTTP and HTTPS.
- Measure:
    - Total bytes transferred.
    - Duration.
    - Derived throughput (KB/s or Mbps).

4.4.2 Interpreting results

Let:

- v_{HTTP} = throughput over HTTP.
- v_{HTTPS} = throughput over HTTPS.
- $R = \frac{v_{\text{HTTP}}}{v_{\text{HTTPS}}}$.

If:

- $R \approx 1$: no clear differential.
- $R > 1.5$: HTTPS likely throttled in relation to HTTP.
- $R < 0.67$: unusual; HTTPS faster than HTTP may indicate caching or CDN differences.

Multiple runs and targets should be used to avoid conflating censorship with congestion or server-side limits.

---

5. Case study: CensorTrace-style diagnostic tool

Your CensorTrace CLI is a production-grade example of how to systematically test for censorship from an end host.

5.1 Core components

Based on your code, key capabilities include:

- **DNS multi-protocol testing:**
    - UDP and TCP to multiple resolvers.
    - DoH and DoT queries.
    - Recognition of known censorship IPs such as 10.10.34.34, 10.10.34.35, 10.10.34.36, d0::11.
- **Dig-style DNS diagnostics:**
    - Hex dump of raw responses.
    - Header parsing and RR decoding.
- **HTTP/HTTPS censorship checks:**
    - Configurable domain, Host header, SNI, and target IP.
    - Differentiation between timeout, reset, and normal responses.
- **TCP reset probe:**
    - Handshake detection and classification into PASS/FAIL/WARN.
- **Throttling check:**
    - Parallel HTTP vs HTTPS downloads of the same asset.
    - Ratio-based decision on throttling.
- **Path diagnostics:**
    - Optional traceroute (UDP / TCP).
    - ICMP ping and TCP “ping” via SYN.
- **Packet export:**
    - Hex dump of DNS queries/responses to a file for offline analysis.

5.2 User interaction and reporting

The tool also includes:

- Interactive prompts for domain selection.
- Colorized console output with status tags:
    - [OK], [WARN], [FAIL], [FILTERED].
- JSON output mode for scripting and integration.
- Summary section that classifies overall network health.

This structure is well aligned with best practices for censorship measurement tools: interactive enough for technicians, scriptable for researchers, and explicit in detection logic.

---

6. Engineering recommendations

6.1 For ISP and network operators

Monitoring and detection:

- **Log DNS behavior:**
    - Track unusual TTLs and private-IP answers from upstream.
    - Watch for inconsistencies between UDP and TCP responses.
- **Inspect TCP anomalies:**
    - Monitor sudden spikes in RST packets on user flows to certain ports/domains.
- **Correlate events:**
    - Link changes in user-visible reachability with routing, configuration, or known policy events.

Hardening and transparency:

- Deploy **DNS over TLS** between forwarders and upstream resolvers.
- Minimize reliance on external resolvers that may be easily intercepted.
- Provide clear status pages for users in case of routing or internal filtering decisions that are not censorship-related, to avoid misattribution.

6.2 For measurement and research teams

- Use **multi-layer probes**:
    - DNS, HTTP, HTTPS, raw TCP, and performance tests.
- Run measurements from **diverse vantage points**:
    - Different ISPs, regions, and access technologies.
- Store **raw packet captures**, where permitted, for detailed offline analysis.
- Document **test cases and heuristics** clearly so that results can be reproduced and independently verified.

6.3 For tool developers

- Expose both **interactive** and **non-interactive (JSON)** modes.
- Allow configuration of:
    - Target domains.
    - Timeouts.
    - Concurrency and test selection.
- Make detection logic explicit:
    - Why a particular test is labeled PASS/FAIL/WARN.
    - Which exact observations led to a “filtered” verdict.

---

7. Future trends in censorship techniques

Looking forward, engineers and researchers should expect increasing sophistication:

- **Encrypted DNS interference:**
    - Blocking or throttling DoH/DoT endpoints.
    - SNI-based blocking against DNS-over-HTTPS hostnames.
- **More advanced fingerprinting:**
    - Recognizing obfuscated VPNs by traffic shape.
    - Classifying protocols using ML-based DPI.
- **Application-layer content manipulation:**

    - Selective interference with specific API endpoints or content types rather than whole domains.
- **More granular throttling:**
    - Per-AS, per-prefix, or per-service traffic shaping under load or during events.

For resilience, tools and systems need to be adaptable, with pluggable transports and dynamic evasion strategies.

---

8. Countermeasures and resilience design

8.1 Transport and protocol strategies

- Use **HTTPS-based transports** that are indistinguishable from ordinary web traffic as much as possible.
- Consider **domain fronting-like** designs where legal and operationally permissible (understanding that many CDNs now limit or forbid it).
- Implement **TLS camouflage**:
    - uTLS-style libraries that mimic popular client fingerprints.
    - Rotating cipher suites and handshake parameters.

8.2 Redundancy and multi-pathing

- Use multiple upstream paths where possible.
- Leverage both IPv4 and IPv6 connectivity.
- Employ multi-homing or tunnel fallback mechanisms (e.g., failover from UDP-based VPN to TLS-based obfuscated proxy).

8.3 Operational practices

- Test regularly from within filtered environments, not just from outside.
- Maintain a library of **known filtered indicators** (e.g., specific DNS IPs, block page signatures, characteristic RST patterns).
- Automate periodic runs of tools like CensorTrace to establish baselines and detect changes early.

---

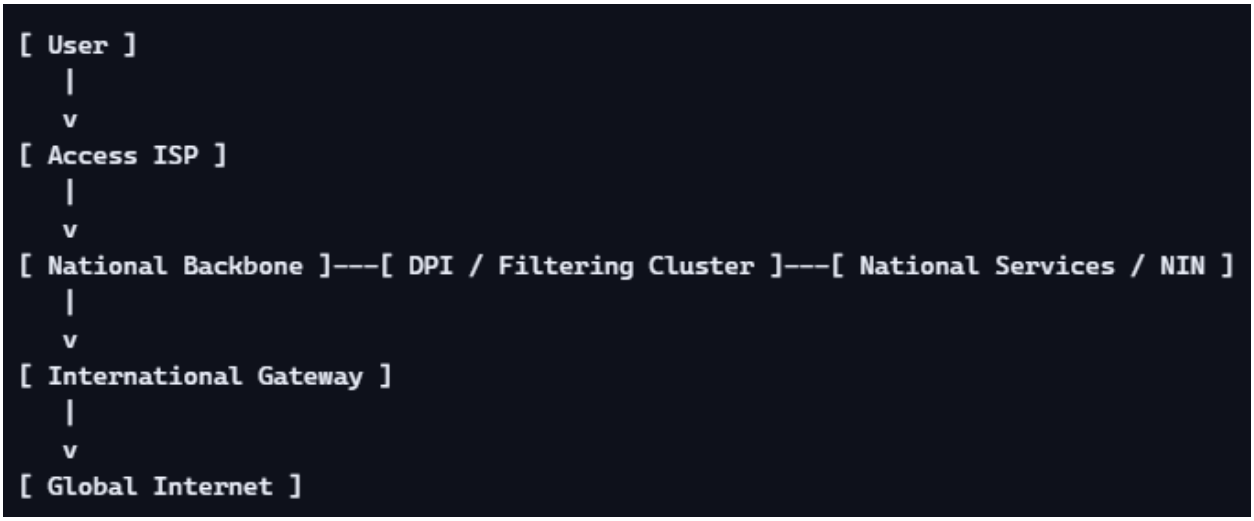
9. Glossary of technical terms

- **BGP (Border Gateway Protocol):** Routing protocol used to exchange reachability information between autonomous systems.
- **DPI (Deep Packet Inspection):** Technology that inspects packet headers and payloads beyond the basic IP/TCP/UDP headers.
- **DNS (Domain Name System):** System that maps human-readable domain names to IP addresses.
- **DoH (DNS over HTTPS):** DNS transported over HTTPS to protect queries from interception.
- **DoT (DNS over TLS):** DNS transported over TLS on a dedicated port (usually 853).
- **HTTP (Hypertext Transfer Protocol):** Application protocol for the Web, usually on port 80.
- **HTTPS:** HTTP over TLS, usually on port 443.
- **SNI (Server Name Indication):** TLS extension indicating the hostname the client is trying to reach.
- **TTL (Time To Live):** DNS field specifying how long a record can be cached.
- **RST (Reset):** TCP flag used to abruptly terminate a connection.
- **Protocol whitelisting:** Strategy where only explicitly allowed protocols are permitted; others are blocked or dropped.

---

10. ASCII diagrams

10.1 Simplified censorship data flow



10.2 DNS interception model

---

11. Conclusion

This whitepaper has outlined a realistic, multi-layer censorship architecture with emphasis on:

- Centralized deployment at a small number of national chokepoints.
- DNS poisoning and redirection to known private IPs.
- HTTP and TLS filtering via Host and SNI inspection.
- Protocol whitelisting that favors DNS/HTTP/HTTPS while suppressing VPNs and other protocols.
- Selective throttling as a subtle form of control.

- A concrete measurement tool design (CensorTrace-style) integrating DNS, HTTP(S), TLS, and performance tests.

For network engineers, ISPs, and measurement researchers, the key takeaway is that reliable detection and understanding of censorship requires:

- Multi-layer testing.
- Careful observation of edge cases (TTL behavior, resets vs timeouts, protocol-specific anomalies).
- Reproducible tooling that makes its logic explicit.