AUTHORIZED ASSESSMENT

HTTP BOUNDARIES

SSRF, shared caches, and request interpretation

A companion field guide for authorized assessment.
Trace every interpretation. Prove only the boundary that the evidence supports.

RESEARCH CUTOFF 06 SEPTEMBER 2026
AUDIO COMPANION HTTP BOUNDARIES



FIELD MAP

Contents

Nine concise sections move from policy definition to controlled proof, cleanup, and evidence labeling.

1. Map the complete chain . . . 2
2. Trace a server-side fetch . . . 4
3. Correlate blind observations . . . 6
4. Map the shared cache . . . 7
5. Prove cache deception . . . 8
6. Prove cache poisoning . . . 9
7. Follow HTTP message boundaries . . . 10
8. Run the field loop . . . 11
9. Source register and evidence limits . . . 12

> ***Name the policy. Trace every interpretation. Prove the boundary crossing. Stop at the strongest evidence.***

Scope and safety boundary

Use this guide only for systems that you own or may assess. Read the current program policy before each test.

Use controlled accounts, synthetic markers, private listeners, and isolated proxy laboratories. Do not scan internal ranges or retrieve real cloud credentials.

Ask for separate approval before a test can affect shared cache objects, shared connections, or other users.

This guide uses ASD-STE100 writing principles. Sentences are short, terms stay consistent, and each paragraph has one main topic.

The guide does not claim full controlled-dictionary certification.

The central rule

> ***Name the policy. Trace every interpretation. Prove the boundary crossing. Stop at the strongest evidence.***

An unusual input is not a finding. A parser difference is not a finding by itself. A cache hit is not a disclosure by itself.

The finding is the proved security effect. Examples include a forbidden peer, the wrong reader, or an incorrect response assignment.

Use one synthetic marker

This guide uses the marker **ALICE-LAB-731**. Alice owns this marker. Bob must not receive it without a permitted reason.

The marker contains no personal information. Replace it with a new, nonsecret value for each controlled test.

01 Map the complete chain

One string can have many readers

Each component can interpret the same request differently. Record the representation that each component receives and uses.

```
Entered URL
  -> transmitted request target
  -> policy parser or cache classifier
  -> key or destination decision
  -> forwarded target
  -> transport parser or origin route
  -> network peer or stored object
  -> response consumer or independent reader
```

Do not replace a hidden component with a guessed product. Mark it as unknown until evidence identifies it.

Start with the security property

Area	Example security property
Server-side fetch	A user-controlled value must not select a forbidden destination.
Credential binding	A trusted credential must go only to an approved destination.
Cache deception	Bob must not retrieve Alice's private response from shared storage.
Cache poisoning	A normal reader must not receive a response selected by an omitted input.
HTTP parsing	Each response must remain attached to the correct request and client.

Write one property before you design the experiment. A test without a property can produce many weak observations.

Separate observation from conclusion

Observation	Supported statement	Unsupported jump
Two parsers return different hosts	A parser disagreement exists.	A forbidden request occurred.
A controlled name was resolved	A domain lookup occurred.	HTTP occurred or a response was used.
A header says HIT	Some cache state can exist.	Alice's response reached Bob.
Two responses arrive	Two responses arrived.	Request smuggling occurred.
One request is slow	A timing difference occurred once.	A protected operation ran.

Evidence ladder

Move only one rung when the next observation supports it.

```
Component difference
  -> observable protocol event
  -> controlled response or object
  -> boundary bypass
  -> sensitive effect
```

Record the strongest supported rung. Also record the smallest missing fact.

02 Trace a server-side fetch

Server-side request forgery is usually called SSRF. SSRF is a destination or trust-boundary defect.

A feature is not vulnerable only because it fetches a public URL. First state the intended destination policy.

URL-flow worksheet

Stage	Record
Source	Exact field, controller, and prerequisite
Composition	Base join, decoding, normalization, and relative resolution
Policy parser	Library, version, scheme, user information, host, and port
Resolution	Every supplied address, alias, and observation time
Continuation	Each redirect, pagination URL, retry, or document reference
Transport parser	Library, version, effective scheme, host, and port
Route or proxy	Direct route, configured proxy, and connection authority
Connection	Observed peer address and port
Authority	Host, Server Name Indication, headers, cookies, and credentials
Consumption	Returned marker, body use, timing use, or state change

Server Name Indication is called SNI. Record it when Transport Layer Security uses a named server.

Compare exact parser tuples

Do not use a general bypass list. Compare the exact libraries and versions used by the supplied system.

For each controlled candidate, predict this tuple for both parsers:

```
(scheme, effective host, effective port)
```

Change one relevant component. Preserve every fact that reached the suspected gate.

Treat user information, host, port, path, query, and fragment as separate components. Record unknown values as unknown.

Address decisions

A host can return more than one Internet Protocol address. Record all supplied version four and version six results.

Do not reduce the decision to one `private` Boolean value. Special-purpose classes include loopback, link-local, shared, multicast, and mapped addresses.

The address at policy time can differ from the address at connection time. Record the peer selected by the actual connection.

Redirects and continuations

Each redirect is a new destination decision. A server-returned pagination URL is also a new decision.

Repeat destination, address, method, header, and credential checks for every hop. Initial validation does not cover a later hop.

Credential binding

Trace destination authority separately from credential authority. A trusted token can become unsafe when untrusted input selects its destination.

Record where the client adds cookies, tokens, signed headers, certificates, or proxy credentials. Check whether each item survives an origin change.

Cloud metadata claims

Do not equate a general fetch with cloud credential access. Metadata services can require special methods, headers, or session tokens.

Use an inert local service to model these capabilities. Do not request real credentials without explicit authority and a safe plan.

SSRF evidence ladder

Rung	What it supports
1	Two components interpret the input differently.
2	A controlled name was resolved.
3	A Transmission Control Protocol, TLS, or HTTP event reached the listener.
4	The application used a controlled response marker.
5	The observed peer violated the destination policy.
6	A sensitive controlled effect crossed the boundary.

A lower rung does not prove a higher rung. A blind defect can still be real when no response body returns.

03 Correlate blind observations

Out-of-band application security testing is called OAST. It uses an external observation to reveal hidden behavior.

Correlation record

```
label
  -> originating request
  -> changed field
  -> submission time
  -> DNS, TCP, TLS, or HTTP event
  -> callback time
  -> response marker consumed
```

Use one unique, nonsecret label per trial. Preserve delayed events and retries.

Minimum controls

- Calibrate the controlled listener directly.
- Keep the known-valid request shape.
- Change only the URL-bearing leaf.
- Include an ordinary permitted destination.
- Include an ordinary rejected destination.
- Include a no-fetch or inert control when it reaches the same gate.

A missing callback does not always disprove a fetch. Network policy, resolver caches, retries, or client behavior can hide the event.

A callback does not always identify the application. Link scanners, browsers, security products, and delayed jobs can create similar events.

Case-study limits

The local Weblate parser report proved a parser and prepared-connection difference. It did not send the complete pagination request on the wire.

The lesson is evidence discipline. Component proof must not become a claim of end-to-end SSRF.

A separate Weblate report used a controlled listener to capture a dummy token. That case shows credential-to-destination binding.

It is adjacent to SSRF, but its main failure is trusted authority bound to an untrusted destination.

04 Map the shared cache

Keep four functions separate.

Function	Question
Eligibility, E	Can this response enter shared storage?
Key, K	Which cache key receives the response?
Forwarding, F	Which request target reaches the origin?
Routing, R	Which origin route handles the forwarded target?

Do not use the phrase “the CDN parsed it differently” as the complete explanation.

Record the input and result for every function. The cache can normalize a path for its key but forward another representation.

Shared-cache diagram

```
Alice request
  -> eligibility E
  -> key K
  -> forwarded target F
  -> origin route R
  -> response stored under K

Bob request
  -> key K
  -> stored response or origin request
```

Record identity partitions, cookie handling, tenant keys, query behavior, time to live, and purge behavior.

A query value is not a safe cache buster until evidence shows that the value participates in the key.

05 Prove cache deception

Cache deception is mainly a confidentiality failure. The origin creates Alice's private response, but a shared cache exposes it.

Three required conditions

1. The changed path still selects private content at the origin.
2. A shared storage rule accepts the response.
3. A less privileged client can request the same cache key.

Break one condition and the chain stops.

Controlled sequence

Step	Client	Expected result
1	Anonymous cold client	Denial or public response without ALICE-LAB-731
2	Alice	Private response with ALICE-LAB-731
3	Independent Bob or anonymous client	Same candidate key, without Alice's credentials
4	Cleanup	Purge or expire the reserved object
5	Replay	Repeat with a fresh reserved key

The important observation is Bob receiving ALICE-LAB-731. A HIT header is supporting evidence only.

Normalization questions

Keep these representations separate:

- URL entered by the tester.
- Request target transmitted by the browser.
- Path used for cache classification.
- Path used for the cache key.
- Target forwarded to the origin.
- Route selected by the origin.

Browsers can remove dot segments or retain fragments locally. Capture the actual request bytes.

Common false positives

- Alice and Bob used the same cookie store.
- The response was a generic application shell.

- The cache stored a denial, not Alice's response.
- A tenant partition remained correct.
- The query changed a separate key.
- The candidate never selected the private origin route.

06 Prove cache poisoning

Cache poisoning is mainly an integrity or availability failure. Request A changes the origin response, but request B uses the same cache key.

```
K(A) equals K(B)
Origin(A) differs from Origin(B)
```

Controlled sequence

1. Prove that one controlled input changes the origin response.
2. Reset or isolate the cache object.
3. Populate the object with request A and marker ALICE-LAB-731.
4. Retrieve with ordinary request B, which omits the influencing input.
5. Verify that B receives the marker.
6. Reset the object and run B before A.
7. Record key, partition, time to live, and intended audience.

Reflection under a separate key is not poisoning of the normal resource. The normal reader must receive the changed response without the special input.

Input categories

Investigate only categories supported by evidence from the actual stack.

- A header that changes origin output but is absent from the key.
- A query value that changes output but is omitted or normalized in the key.
- A path that routes one way but maps to another key.
- A method or override that changes the stored representation.

Do not spray unrelated headers. Each candidate needs a predicted E, K, F, and R result.

Case-study lessons

A public Cloudflare report described header-count truncation that removed cache-rule and key dimensions. The report is complete public source material.

A Mozilla report described method override behavior that stored an empty representation. It is useful because the researcher limited unsupported claims.

A Shopify Linkpop report described path normalization that associated one product with another lookup. It was resolved with a low severity.

An Algolia record describes a CSS-looking URL and private information. The available local detail is summary-only, so the missing sequence must stay unknown.

07 Follow HTTP message boundaries

HTTP version one connections can carry several messages. Each component must agree where each message ends.

Common labels include content-length then transfer-encoding, transfer-encoding then content-length, and zero-body variants.

The label describes observed parser behavior. Header presence alone does not prove a disagreement.

Private-lab record

Item	Record
Client bytes	Exact bytes and line endings
Front-end boundary	Byte offset after each request
Origin boundary	Byte offset after each request
Connection	Client and upstream identifiers
Request assignment	Origin request identifier and route
Response assignment	Client request that received each response

Minimum comparison

1. Send two ordinary valid requests. Confirm normal pipelining.
2. Apply one framing change through controlled client A.
3. Send a stable request through controlled client B.
4. Compare B's marker and response assignment.
5. Disable upstream reuse and repeat.
6. Send a correctly framed candidate and repeat.

Two requests and two responses can be normal pipelining. A timeout can mean buffering, rejection, throttling, or connection closure.

Record the protocol at each hop. Public HTTP version two does not prove that the upstream connection also uses version two.

Repair direction

- Reject ambiguous message framing.

- Use consistent parsing between components.
- Drain request bodies safely before connection reuse.
- Close a connection when its boundary becomes uncertain.
- Keep translated requests isolated when required.

08 Run the field loop

Before testing

- Confirm scope and current program policy.
- Record the permitted identities and destinations.
- Write one security property.
- Choose a new nonsecret marker.
- Define stop conditions, cleanup, time, and attempt budgets.
- Obtain separate approval for shared-state validation.

Experiment record

Field	Value
Attempt identifier	
Security property	
Baseline request and response	
One changed property	
Predicted component results	
Positive control	
Negative control	
Falsifier	
Actual observation	
Strongest evidence rung	
Missing fact	
Result-conditioned next step	

Stop conditions

Stop when one of these conditions applies:

- The security property is disproved at the permitted evidence rung.
- The current authority does not permit the next effect.
- The attempt or time budget ends.
- Controls cannot distinguish the proposed cause.
- The instrument is not calibrated.
- The next step would repeat an equivalent failed attempt.

Finding structure

1. State the violated security property.
2. Show the affected component chain.
3. Show baseline, candidate, control, and independent replay.
4. State the strongest evidence rung.
5. Separate observed facts from research background.
6. State missing facts and alternative explanations.
7. Recommend repair at the broken boundary.
8. Include cleanup and regression evidence.

Final review

- Does each important claim point to captured evidence?
- Did a lower evidence rung become a higher claim?
- Did the test use two truly independent identities where required?
- Did the report distinguish a complete source from a summary?
- Did the test stay within the current policy?
- Did cleanup finish?

09 Source register and evidence limits

Primary standards and guidance

- OWASP SSRF Prevention Cheat Sheet.
- OWASP Web Security Testing Guide: Server-Side Request Forgery.
- RFC 3986, URI Generic Syntax.
- RFC 9110, HTTP Semantics.
- WHATWG URL Standard.
- IANA IPv4 and IPv6 special-purpose address registries.
- Go `net/http`, Python `urllib.parse`, and Python `ipaddress` documentation.

- AWS, Google Cloud, and Microsoft Azure metadata documentation.

Main research

- Martin Doyhenard, *Gotta cache 'em all*, Black Hat USA and DEF CON 2024.
- James Kettle, *HTTP/1.1 must die*, 2025.
- James Kettle, *HTTP Terminator*, Black Hat USA 2026.
- Microsoft Project Dusseldorf, controlled OAST infrastructure.

Local synthesis used

- `research/ssrf-harness-notes.md`, reviewed 6 September 2026.
- `web-api-hacking-techniques/Content/05-cache-deception.md`.
- `web-api-hacking-techniques/Content/06-cache-poisoning.md`.
- `web-api-hacking-techniques/Content/07-http-desync.md`.
- `web-api-hacking-techniques/Content/08-crlf-desync.md`.
- `web-api-hacking-techniques/Content/13-feedback-fuzzing-oast.md`.
- `harness/skills/bb-oast-fetch-paths/SKILL.md`.
- `harness/skills/bb-cache-deception/SKILL.md`.
- `harness/skills/bb-cache-key-collisions/SKILL.md`.

Disclosure evidence labels

Use these labels when you reuse the local disclosure archive:

- **Complete public report:** full public body is stored and reviewed.
- **Partial technical extract:** important sections are present, but the report is incomplete.
- **Platform summary:** the platform created a short description from limited public data.
- **Metadata only:** title, program, state, severity, and source link can be available without technical detail.
- **Local reproduction:** a controlled test independently reproduced the behavior.

Do not present a platform outcome as independent reproduction. Do not invent a missing exploit sequence from a title or summary.

Pocket rule

> ***Name the policy. Trace every interpretation. Prove the boundary crossing. Stop at the strongest evidence.***