CS-061 · Application Security

Server-Side Request Forgery (SSRF)

An attack that makes a server send requests to targets the attacker chooses.

Executive Summary

Server-side request forgery, or SSRF, is an attack in which an application is tricked into making requests to a destination the attacker controls or chooses, including internal systems the attacker could not reach directly. Because the request comes from a trusted server, it can bypass network boundaries and reach sensitive internal services. The core defenses are strict allowlists for outbound requests and network controls that limit what the server can reach.

What It Is

SSRF occurs when an application takes a user-supplied address or resource identifier and fetches it on the server side without properly restricting where it can go. Applications legitimately fetch remote resources for many reasons, such as loading a URL, importing a file, or calling a webhook. If the destination is not tightly controlled, an attacker can point it at internal addresses, cloud metadata services, or other systems that trust requests coming from inside the network. The server becomes a proxy that reaches places the attacker cannot reach from the outside. SSRF has become especially significant in cloud environments, where an internal metadata endpoint may expose sensitive configuration if it can be reached.

Why It Matters

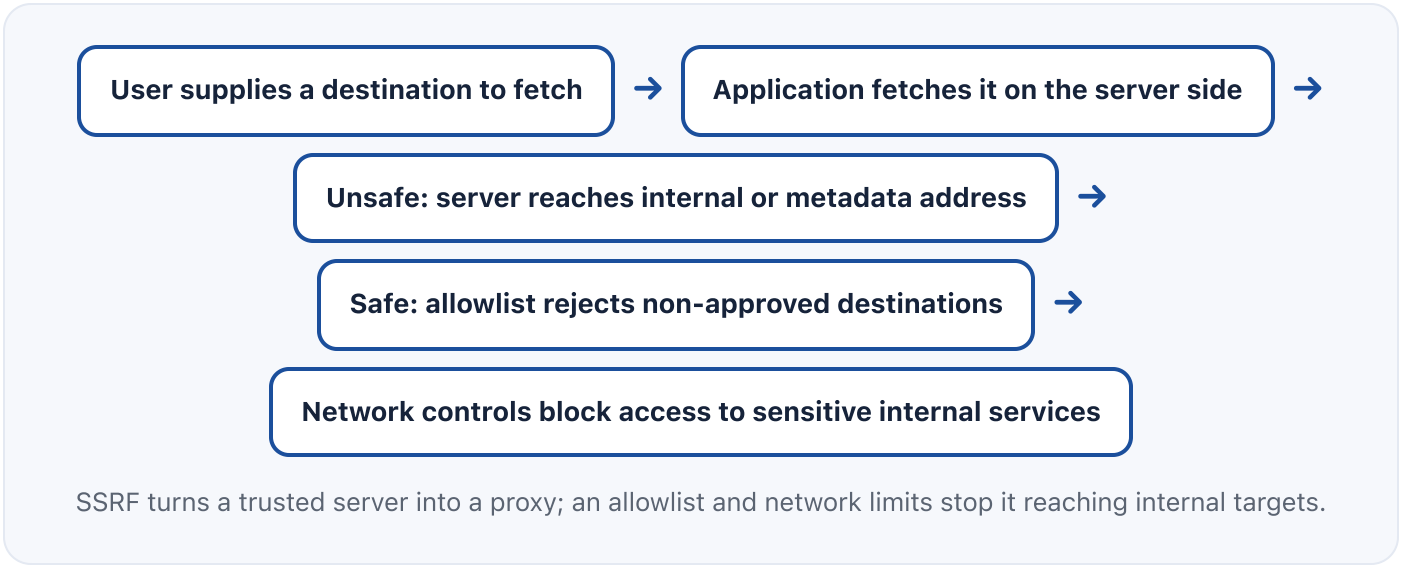
SSRF is dangerous because it defeats the assumption that internal systems are safe simply because they are not directly exposed to the internet. By abusing a trusted server, an attacker can scan internal networks, reach administrative interfaces, and in cloud environments potentially retrieve credentials or configuration from internal metadata services. That can escalate from a single flawed feature to broad access. SSRF has been recognized as a distinct,

high-priority web application risk, which means anyone building features that fetch remote resources needs to understand and prevent it.

How It Works

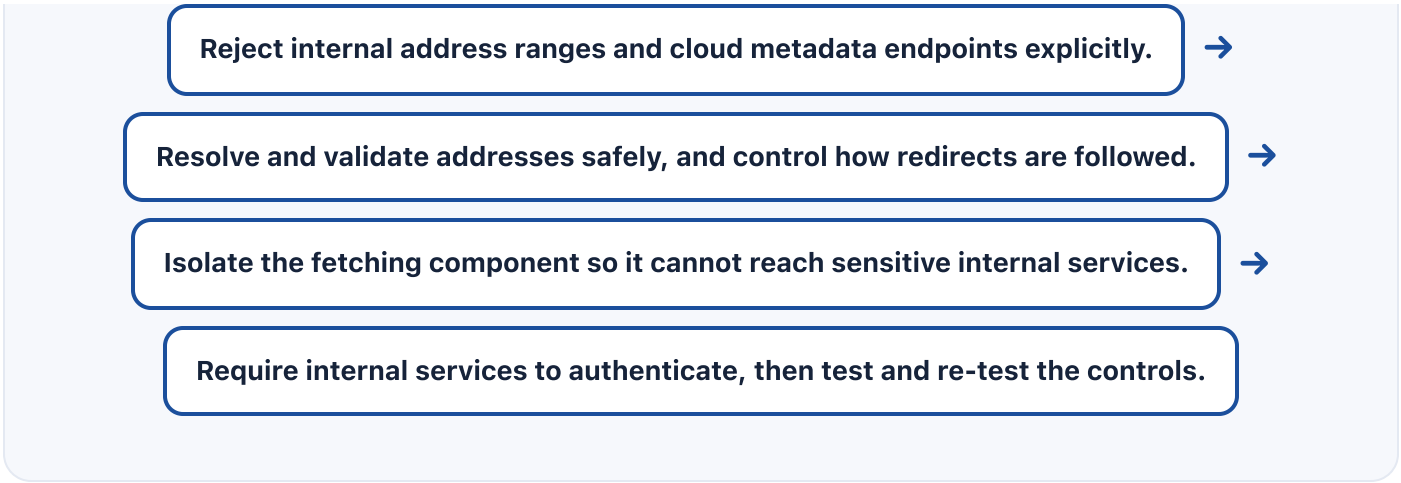
At a conceptual level, SSRF works because the application trusts a destination that the attacker can influence. The strongest defense is to restrict outbound destinations to a strict allowlist of known, required hosts rather than trying to block bad ones, since blocklists are easily bypassed through alternate encodings and redirects. Validate and resolve the destination safely, and reject requests to internal address ranges and metadata endpoints. Network controls add a critical layer: place the fetching component where it cannot reach sensitive internal services, and require internal services to authenticate rather than trusting any request from inside. Handling redirects carefully and not returning raw fetched responses to the user further reduce what an attacker can learn or reach.

Architecture Diagram



SSRF turns a trusted server into a proxy; an allowlist and network limits stop it reaching internal targets.

Visual Workflow



Common Attacks

- Pointing a fetch feature at internal-only systems the server can reach
- Reaching a cloud metadata endpoint to retrieve configuration or credentials
- Scanning or mapping internal networks through the trusted server
- Bypassing weak filters with alternate encodings, hostnames, or redirects
- Abusing webhooks, importers, or previewers that fetch arbitrary destinations

Common Mistakes

- Relying on a blocklist of bad addresses instead of an allowlist of good ones
- Trusting that internal services are safe because they are not internet-facing
- Following redirects blindly so an approved host redirects to an internal one
- Returning the raw fetched response to the user and leaking internal data
- Leaving the fetching component able to reach metadata and admin services

Best Practices

- Restrict outbound destinations to a strict allowlist of hosts and protocols
- Explicitly block internal address ranges and cloud metadata endpoints
- Validate and resolve destinations safely and control redirect handling
- Isolate the fetching component with network segmentation and egress controls
- Require authentication on internal services rather than trusting internal requests
- Avoid returning raw fetched content and log outbound requests for review

Quick Checklist

- [x] Outbound fetch destinations limited to an allowlist
- [x] Internal ranges and metadata endpoints explicitly blocked
- [x] Redirects handled safely so they cannot reach internal targets
- [x] Fetching component isolated by network segmentation and egress rules
- [x] Internal services require authentication, not implicit trust
- [x] SSRF tested on every remote-fetch feature before release

Recommended Tools

Egress network controls
Limit which destinations a server component can reach outbound

Dynamic application security testing (DAST)
Probes remote-fetch features for SSRF behavior

Static application security testing (SAST)
Flags user-influenced destinations passed to fetch operations

Network segmentation
Separates fetching components from sensitive internal services

Industry Standards

OWASP Top 10
SSRF is recognized as a distinct high-priority web application risk

OWASP Application Security Verification Standard (ASVS)
Testable requirements for controlling server-side requests

CWE-918
The Common Weakness Enumeration entry for server-side request forgery

Career Relevance

SSRF is a priority topic for application security engineers, penetration testers, and secure code reviewers, especially in cloud-heavy environments where the impact can be severe. Developers building importers, webhooks, and preview features are expected to prevent it with allowlists and isolation, and cloud and platform engineers enforce the network controls that contain it. For the AI-Governance-Jobs.com audience, SSRF has become a frequent modern interview topic.

Interview Questions

- What is SSRF, and why is a request from the server so powerful?
- Why is SSRF especially dangerous in cloud environments?
- Why is an allowlist preferred over a blocklist for outbound destinations?
- How do network segmentation and egress controls contain SSRF?
- How can an attacker bypass naive SSRF filters, and how do you counter that?

Related Certifications

GIAC Web Application Penetration Tester (GWAPT)

Offensive Security Web Assessor (OSWA)

CompTIA Security+

Further Reading

- OWASP: Server-Side Request Forgery
- CWE-918: Server-Side Request Forgery
- OWASP Top 10

Key Takeaways

- SSRF makes a trusted server send requests to attacker-chosen destinations.
- It can reach internal systems and cloud metadata that are not directly exposed.
- Allowlists beat blocklists because blocklists are easily bypassed.
- Network segmentation and egress controls are essential containment.

- Internal services should authenticate rather than trust internal requests.

FAQ

Related Careers

RELATED ROLES

- Application Security Engineer
- Penetration Tester
- Secure Code Reviewer
- Security Engineer

RELATED CERTIFICATIONS

- GIAC Web Application Penetration Tester (GWAPT)
- Offensive Security Web Assessor (OSWA)
- CompTIA Security+

CURRENT OPENINGS

Live roles are on the job board.

Browse all jobs

GUIDES & SALARY

- Career guides
- Role roadmaps
- Salary data
- Career tools

SUGGESTED LEARNING PATH

1. Ground the basics with CS-001 Cybersecurity
2. Study this sheet: **Server-Side Request Forgery (SSRF)**
3. Go deeper: OWASP Top 10
4. Go deeper: SQL Injection

5. Validate it: work toward **GIAC Web Application Penetration Tester (GWAPT)**
6. Find the role: browse current openings

RELATED SHEETS

CS-057 OWASP Top 10 | CS-058 SQL Injection | CS-063 Authorization | CS-064 Secrets Management | CS-001 Cybersecurity