What is server-side request forgery?

Server-side request forgery (usually written SSRF) is the class of vulnerability where an attacker convinces an application to fetch a URL of the attacker's choosing. Because the request comes from inside the application's network, it can reach private services that have no public exposure, including cloud metadata endpoints that hand out access credentials. SSRF is the failure mode behind several of the largest cloud breaches on record.

HOW IT WORKS

01 What does SSRF have to do with the Capital One breach?

In 2019, an attacker exploited a misconfigured web application firewall in front of Capital One's AWS environment to perform SSRF against the EC2 instance metadata endpoint at `169.254.169.254`. The metadata service returned temporary IAM credentials, which the attacker used to read approximately 100 million customer records from S3. The settlement reached $190 million.

This chain (SSRF -> metadata -> credentials -> data) is one of the most common cloud-era exploit patterns. AWS introduced IMDSv2 specifically to defend against it. Many production AWS environments still permit IMDSv1 on at least some instances.

02 What do attackers do with SSRF?

- Steal cloud credentials. Hit the metadata endpoint (`169.254.169.254` on AWS, `169.254.169.254` on Azure, `metadata.google.internal` on GCP), get the temporary credentials assigned to the instance, use them against the cloud provider's API.
- Scan and exploit internal services. Enumerate internal hostnames and ports, find admin interfaces, exploit them directly.
- Bypass authentication. Some internal services trust requests that come from inside the network without authentication. SSRF turns the application server into an authentication bypass.

HOW TO DEFEND

- URL allowlist. The application should only fetch URLs that match an explicit allowlist of hosts. Block everything else by default. Validate after DNS resolution, not just on the input string, to catch DNS-rebinding tricks.
- Network-level controls. The application server should not be able to reach the metadata endpoint or internal admin services at all. Egress firewalls, network segmentation, and IMDSv2 (which requires a session token, defeating basic SSRF) close most of the post-exploitation paths.
- Response handling. Do not return the raw response of the fetched URL to the user. Return a status or a transformed value. Limits how much the attacker can read even when the fetch succeeds.
- Disable response redirects. Most SSRF libraries follow HTTP redirects by default. An allowlist that accepts only `https://images.example.com` is bypassed by a redirect to the metadata endpoint.

SOURCES

[1] OWASP A10:2021 Server-Side Request Forgery

[2] OWASP SSRF Prevention Cheat Sheet

[3] CWE-918: Server-Side Request Forgery

[4] AWS IMDSv2 documentation

- Pivot into databases and caches. Read or write to internal Redis, Memcached, Elasticsearch instances that were never meant to be exposed.
- Data exfiltration. Use the SSRF response to leak data the application normally would not return to the user.

03 How does SecureLayer7 test for SSRF?

Every application engagement maps the places the server makes outbound HTTP requests on the user's behalf. For each one we test:

- Direct SSRF. Can we point the fetch at an internal hostname or IP?
- Metadata-endpoint reachability. Can we reach the cloud metadata service, and does it return credentials?
- DNS-rebinding bypass. Can we register a hostname that resolves to a public IP at validation time and an internal IP at fetch time?
- Redirect-based bypass. Can we redirect from an allowlisted host to an internal target?
- Blind SSRF. When the response is not returned to us, can we still confirm the request fired (via timing, DNS callback, or out-of-band) and reach an internal target?

Deliverable includes the realistic blast radius for each finding: what internal service is reachable, what credentials are extractable, what data is at risk.

Test for SSRF before your cloud metadata service gets owned.
securelayer7.net/learn/application-security/ssrf

Open online