GRIDINSOFT
HELP CENTER

OFFICIAL SUPPORT GUIDE

R

Reverse Proxy: How It Works and How to Secure It

Learn how a reverse proxy routes requests to origin servers, how it differs from a forward proxy, and which security settings matter.

PUBLISHER	UPDATED	READ TIME
Gridinsoft Support	Oct 8, 2026	4 min

CONTENTS

In this guide

01 What is a reverse proxy?
02 How a reverse proxy works
03 Reverse proxy vs. forward proxy
04 What reverse proxies are used for
05 TLS and upstream security
06 Client IP and forwarded headers
07 Caching, streaming, and protocol details
08 Reverse proxy security checklist

A **reverse proxy** accepts requests on behalf of one or more origin servers, sends each request to an appropriate upstream service, and returns the response to the client. Users connect to the proxy’s public address and usually do not communicate directly with the application server.

Two related design checks are CDN SSL/TLS security, which covers the edge-to-origin connection, and web cache poisoning, where a cache serves attacker-influenced content to other users. A working HTTPS connection alone does not validate the cache policy.

This layer can simplify TLS, routing, caching, access control, and scaling. It also becomes a critical trust boundary: a configuration error can expose an origin, leak client data, or make every application behind the proxy unavailable.

01 What is a reverse proxy?

A reverse proxy is the client-facing intermediary for one or more servers. The client selects the public site, but the proxy decides which private or public origin receives the request. This is the opposite direction from a forward proxy, which is chosen or assigned on the client side to reach external destinations.

02 How a reverse proxy works

1. A browser resolves the site’s hostname and connects to the reverse proxy.
2. The proxy terminates or passes through the connection and applies routing and security rules.
3. It selects an upstream server based on hostname, path, health, or load-balancing policy.
4. The origin processes the request and returns a response to the proxy.
5. The proxy may buffer, compress, or cache the response before returning it to the client.

The origin can reside on the same host, a private network, a container platform, or another region.

03 Reverse proxy vs. forward proxy

A forward proxy represents clients that want to reach external destinations. A reverse proxy represents servers that receive incoming traffic. Users often choose the destination when using a forward proxy; they may not know which origin handled a request behind a reverse proxy.

A load balancer distributes traffic, while a reverse proxy may also perform routing, TLS termination, authentication, caching, and content transformation. Products often combine both roles. A CDN places distributed proxy caches near users, and an API gateway adds API-specific controls such as token validation, quotas, and request transformation.

04 What reverse proxies are used for

- TLS certificate management and HTTPS enforcement.
- Host- and path-based routing to multiple applications.
- Load balancing and health checks across redundant origins.
- Caching static or safe dynamic responses.
- Rate limiting, authentication, request-size limits, and web filtering.
- Keeping origin addresses off the public Internet.

05 TLS and upstream security

TLS termination at the proxy decrypts traffic before forwarding it. If the upstream crosses an untrusted or shared network, use HTTPS again between proxy and origin and validate the origin certificate. Restrict the origin firewall to trusted proxies, and consider mutual TLS for sensitive internal services. Never assume a hidden origin hostname prevents direct access.

06 Client IP and forwarded headers

Origins commonly receive the proxy's IP address as the network peer. The proxy can pass the original address in a standardized `Forwarded` header or fields such as `X-Forwarded-For`. The application must trust these headers only when the immediate peer is an approved proxy. Remove or overwrite untrusted incoming values; otherwise an attacker can spoof an IP address and bypass logging or access rules.

Forward the intended `Host` and scheme consistently. Incorrect values can cause bad redirects, insecure cookies, broken tenant routing, or password-reset links pointing to an attacker-controlled host.

07 Caching, streaming, and protocol details

Cache only responses whose authentication and privacy rules are understood. A mistaken cache key can serve one user's data to another. Streaming responses, server-sent events, WebSockets, large uploads, and gRPC may need adjusted buffering, connection-upgrade headers, and timeouts. Set explicit request and response size limits rather than accepting unlimited bodies.

08 Reverse proxy security checklist

1. Patch the proxy and restrict access to its control plane.
2. Block direct public access to origins and encrypt sensitive upstream traffic.
3. Define trusted proxies before using forwarded client addresses.
4. Set timeouts, body limits, rate limits, and health checks.
5. Test failover and monitor proxy and origin logs with a shared request ID.
6. Review cache rules after every authentication or routing change.
7. Use a web security gateway or equivalent controls when deeper request inspection is required.

NEED MORE HELP?

We are here when the guide is not enough.

Visit the Gridinsoft Support Center for current product guidance or submit a support request with the details of your case.

Open this guide online at support.gridinsoft.com