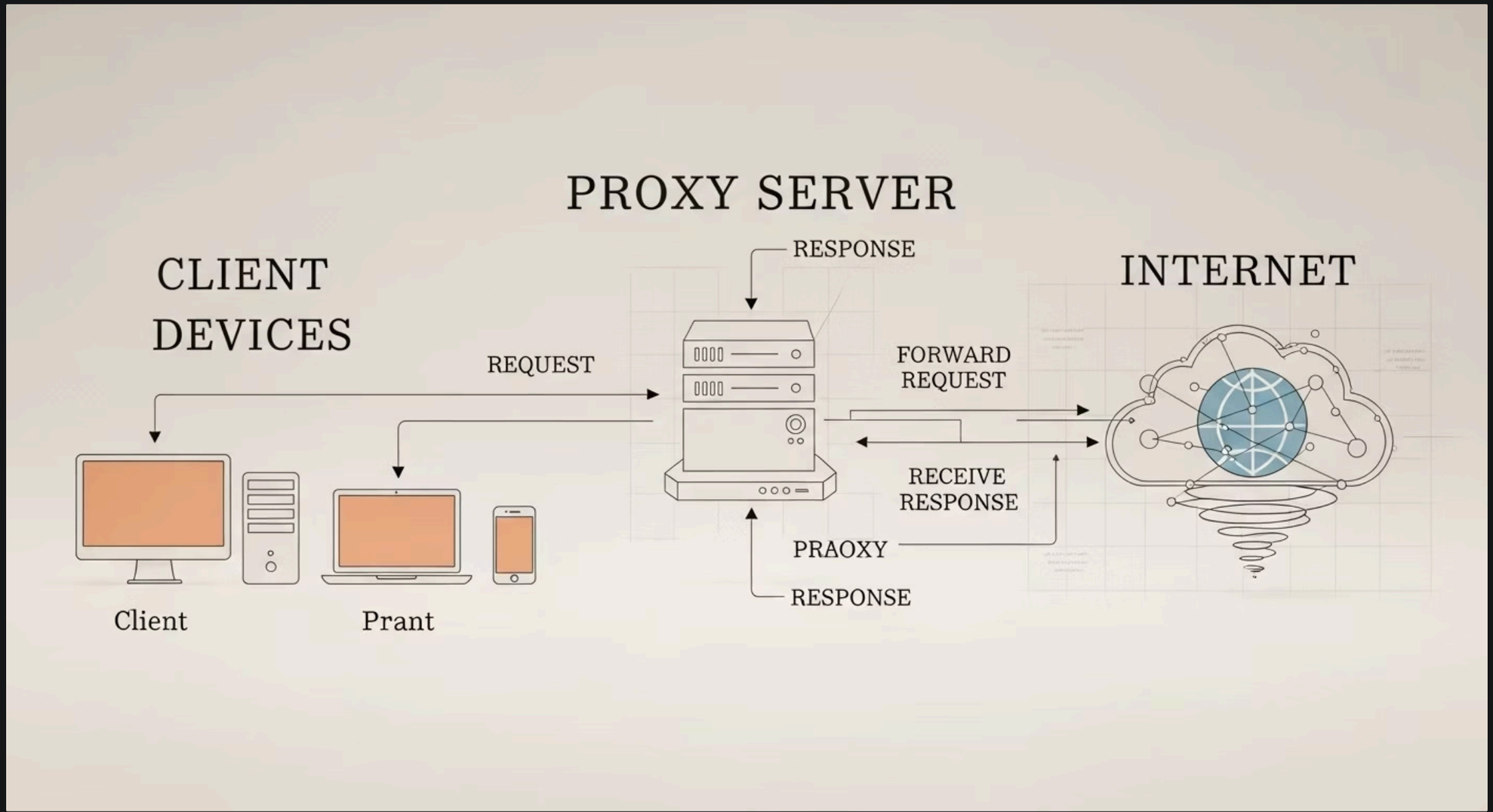
Understanding Proxy Servers and Squid ACLs

A comprehensive guide to proxy server technology, focusing on Squid proxy implementation and Access Control Lists for effective network management

What is a Proxy Server?



Intermediary Gateway

Acts as a middle layer between client devices and internet resources, intercepting all requests and responses

Content Caching

Stores frequently accessed web content locally to reduce bandwidth usage and accelerate content delivery

Security Enhancement

Filters malicious requests, masks client IP addresses, and provides an additional layer of network protection

Why Use a Proxy Server?

Proxy servers provide essential functions for modern networks, particularly in enterprise environments where security and efficiency are paramount.

Performance Optimisation

Intelligent caching reduces bandwidth consumption by up to 60% for frequently accessed content, dramatically improving response times.

Access Management

Centralised control over internet usage with comprehensive monitoring and detailed access logs for compliance requirements.

Privacy Protection

Client requests appear to originate from the proxy's IP address, shielding internal network structure and user identities from external sites.

Content Filtering

Granular control over accessible content based on URL patterns, domain names, or content categories to enforce acceptable use policies.

Introducing Squid Proxy Server

Squid has been the industry standard for proxy solutions since 1996, powering networks across government, education, and enterprise sectors worldwide.

Open-Source Excellence

Freely available, community-supported caching proxy solution for HTTP, HTTPS, FTP and other popular internet protocols

Cross-Platform Compatibility

Runs seamlessly on Unix/Linux distributions, macOS, and Windows environments with consistent performance

Enterprise-Grade Configuration

Highly adaptable with extensive access control capabilities, authentication integration, and logging options

Trusted by major ISPs and organisations for decades to manage terabytes of daily traffic with minimal resource requirements.

How Squid Works: Caching and Forwarding

Client Request

User's browser sends HTTP request to Squid proxy

Cache Check

Squid checks local cache for requested object

Origin Fetch

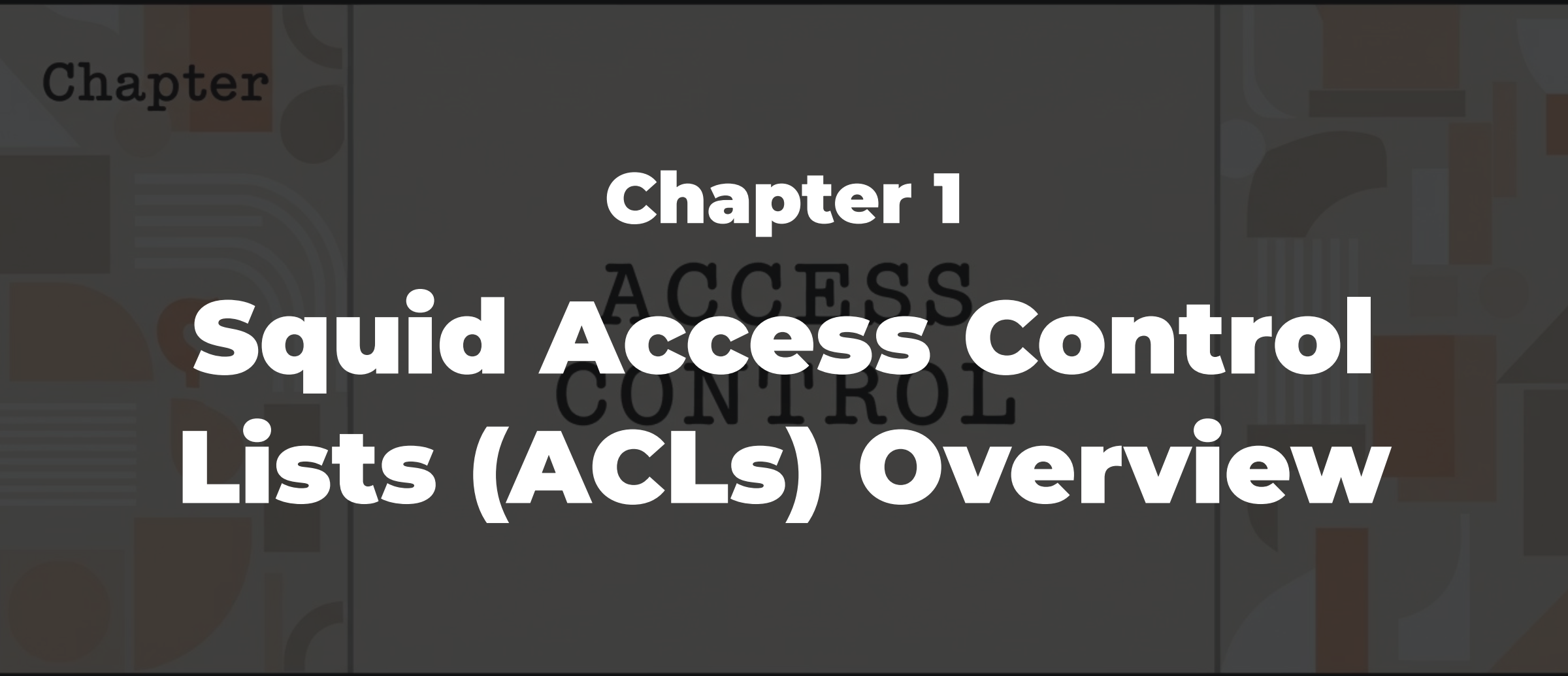
If not in cache, Squid retrieves from origin server

Store & Forward

Squid stores copy in cache, sends to client

Key Benefits:

- Reduces external bandwidth requirements by serving cached content locally
- Decreases page load times by up to 60% for cached resources
- Supports conditional requests to verify freshness of cached content
- Implements hierarchical caching for distributed environments
- Authenticates users through LDAP, Active Directory, or custom mechanisms



Chapter 1

Squid Access Control Lists (ACLs) Overview

What are ACLs in Squid?

Access Control Lists form the backbone of Squid's security model, enabling administrators to implement sophisticated access policies based on multiple criteria.

ACLs operate through a two-step process: defining criteria to match against, then creating rules that apply actions when those criteria match.

Core ACL Components:

- **ACL Elements:** Define specific criteria to test against (IP addresses, domains, times)
- **Access Rules:** Apply actions (allow/deny) based on matching ACL elements

Example Configuration:

```
# Define ACL for internal network
acl internal_network src 10.0.0.0/8

# Allow access from internal network
http_access allow internal_network

# Block everyone else
http_access deny all
```

This layered approach provides exceptional flexibility in controlling exactly who can access what resources and when.

Common ACL Element Types in Squid

Source-Based Controls

`src`: Matches client IP addresses or subnets

```
acl office_network src 192.168.1.0/24
```

`srcdomain`: Matches client domain names

```
acl local_clients srcdomain .example.org
```

Destination-Based Controls

`dst`: Matches destination server IP addresses

```
acl internal_servers dst 10.0.0.0/8
```

`dstdomain`: Matches destination domains

```
acl social_media dstdomain .facebook.com .twitter.com
```

Content-Based Controls

`url_regex`: Pattern matching with regular expressions

```
acl downloads url_regex -i \.(zip|exe|iso)$
```

`method`: HTTP methods like GET, POST

```
acl post_requests method POST
```

Time-Based Controls

`time`: Restricts access by time of day/week

```
acl business_hours time MTWHF 9:00-17:00
```

These core ACL types form the building blocks for comprehensive access control policies that can be combined for highly specific rules.

Advanced ACL Types

Authentication ACLs

`proxy_auth`: Verifies user credentials via external mechanisms like LDAP or Active Directory

```
acl authenticated proxy_auth REQUIRED
```

Client Attributes

`browser`: Matches against user-agent strings to identify specific browsers or devices

```
acl mobile browser -i mobile android iphone
```

Connection Properties

`port/myport`: Filters by destination port or local listening port

```
acl secure_ports port 443 8443
```

Connection Limiting

`maxconn`: Restricts the maximum number of simultaneous connections from a single client IP, effective for preventing resource exhaustion

```
acl heavy_users maxconn 10
```

Custom Extensions

`external`: Interfaces with external helper programs for custom validation logic beyond Squid's built-in capabilities

```
acl special_users external special_group
/etc/squid/check_group.sh
```

These advanced ACLs enable sophisticated policies that can adapt to complex organisational requirements.

Chapter 2

Using ACLs to Control Access

Defining and Applying ACLs in squid.conf

ACL Definition Syntax

```
acl <name> <type> <value>

# Examples:
acl localnet src 192.168.1.0/24
acl workstations src 10.0.1.0/24
acl blocked_sites dstdomain .example.com
```

ACL definitions create named patterns that can be referenced in access rules. They do not enforce any policies by themselves.

Access Rule Application

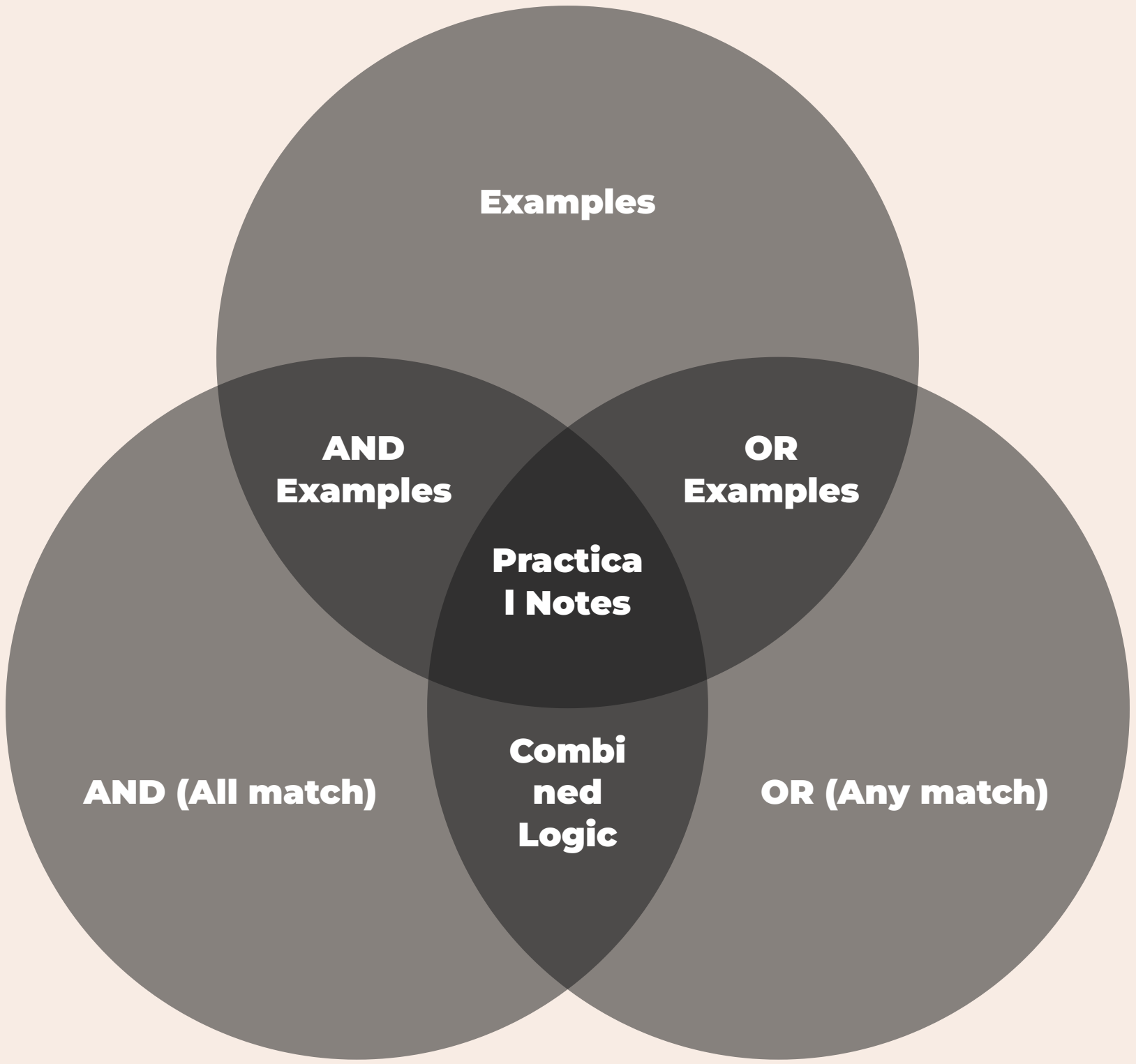
```
http_access <action> <acl_name> [<acl_name>...]

# Examples:
http_access allow localnet
http_access deny blocked_sites
http_access deny all
```

Important: Rule processing stops at the first match, so order is critical. Place more specific rules before general ones, with a default deny rule at the end for security.

The final rule should typically be `http_access deny all` to implement a "default deny" security posture, ensuring only explicitly allowed traffic passes through.

Combining ACLs: AND vs OR Logic



AND Logic: All Conditions Must Match

Multiple ACLs on a single http_access line create an AND relationship - all conditions must be satisfied for the rule to match.

```
# Allow local network during work hours only
acl localnet src 192.168.0.0/16
acl work_hours time MTWHF 9:00-17:00

http_access allow localnet work_hours
```

This rule only allows access when the client is both on the local network AND it's during work hours.

OR Logic: Any Condition Can Match

Separate http_access lines create an OR relationship - if any rule matches, its action is applied.

```
# Allow either administrators or users during work hours
acl admins src 10.0.0.0/24
acl users src 10.0.1.0/24
acl work_hours time MTWHF 9:00-17:00

http_access allow admins
http_access allow users work_hours
```

Admins can access anytime, while regular users can only access during work hours.

Organizing ACLs with External Files

For large-scale deployments, external files provide a clean way to manage extensive ACL lists without cluttering the main configuration file.

Benefits:

- Separates policy from configuration
- Enables easier updates to ACL lists
- Facilitates automation via scripts
- Improves readability of squid.conf

File-Based ACL Configuration

```
# In squid.conf:
acl allowed_sites dstdomain "/etc/squid/allowed_sites.txt"
acl blocked_sites dstdomain "/etc/squid/blocked_sites.txt"
acl internal_ips src "/etc/squid/internal_networks.txt"

http_access allow internal_ips allowed_sites
http_access deny blocked_sites
```

External File Format:

```
# /etc/squid/allowed_sites.txt
.gov.uk
.edu
.example.com
example.org
```

Note: After modifying external files, you must reload Squid for changes to take effect: squid -k reconfigure

Real-World Example: Restricting Access by Time and User

This comprehensive example demonstrates a typical enterprise configuration that restricts internet access based on authentication status and time of day.

Define Time Periods

```
acl work_hours time MTWHF 09:00-17:00
acl lunch_hours time MTWHF 12:00-13:00
acl after_hours time MTWHF 17:00-23:59
acl weekend time SA 00:00-23:59
```

Define Content Categories

```
acl social_media dstdomain
"/etc/squid/social.txt"
acl streaming dstdomain
"/etc/squid/streaming.txt"
acl business dstdomain "/etc/squid/business.txt"
```

1

2

Define User Groups

```
acl authenticated_users proxy_auth REQUIRED
acl managers external group
/etc/squid/check_group.sh managers
acl it_staff external group
/etc/squid/check_group.sh it
```

3

4

Apply Access Rules

```
# IT staff get full access
http_access allow it_staff

# Managers can access everything during work
hours
http_access allow managers work_hours

# All authenticated users can access business
sites
http_access allow authenticated_users business

# Social media allowed during lunch only
http_access allow authenticated_users
social_media lunch_hours

# Deny everything else
http_access deny all
```

This policy creates a balanced approach to internet access, granting privileges based on both role and time, while maintaining security through authentication.

Conclusion: Mastering Squid ACLs for Secure, Efficient Proxy Use

Performance Benefits

Proxy servers significantly reduce bandwidth consumption and accelerate content delivery through intelligent caching, particularly for organisations with limited internet connectivity.

Security Framework

Squid's flexible ACL system provides robust security controls, enabling precise management of who can access what resources and when, while maintaining detailed audit logs.

Implementation Strategy

Start with simple ACL configurations focused on essential security requirements, then gradually incorporate more sophisticated rules as your expertise grows.

Next Steps

Experiment with advanced ACL types and external helpers to extend Squid's capabilities for your specific network environment and security needs.

For further assistance or implementation guidance, contact our network operations team at netops@example.com