# Authentik Forward Auth and Nginx: SSO Protection for Web Radio and Streams (Without Timeouts)

# Authentik Forward Auth and Nginx: SSO Protection for Web Radio and Streams (Without Timeouts)

When securing a corporate web application behind an Identity Provider like **Authentik**, the quickest solution is usually to deploy a **Proxy Standalone Outpost**. In this setup, the Outpost acts as a shield (Reverse Proxy) in front of the application: all traffic flows through the Outpost container before reaching the actual web server.

This approach works beautifully for the vast majority of websites and APIs. However, it turns into a **total disaster** if the application in question runs a continuous media stream, such as a **Web Radio** (e.g., AzuraCast, Icecast).

Let's look at why audio streaming breaks with traditional proxies, how to solve this permanently using **Forward Auth** mode with Nginx, and how to overcome HTTP Host header rewriting issues introduced by Cloudflare Tunnel.

---

## The Problem: Go Proxy Timeouts

Authentik outposts are written in Go. Web servers written in Go use strict internal timeouts to manage TCP connections (specifically `ReadTimeout` and `WriteTimeout`).

When a user connects to an audio stream (e.g., `/listen/radio.mp3`), the established HTTP connection uses *infinite chunked transfer*: the connection remains open indefinitely to continuously stream audio packets to the listener's ears.

After a few minutes of playback, the audio stream abruptly stops. This happens because the connection exceeds the Go outpost's write timeout.

The outpost interprets the infinite session as a potential attack or a hung socket and forcefully shuts it down. Increasing these timeouts indefinitely is not a good idea, as it exposes the server to Denial of Service (DoS) vulnerabilities.

---

## The Solution: Nginx Forward Auth (`auth_request`)

To combine SSO security with the stability of infinite streaming, we must reverse the roles and use **Forward Auth (Option B)**.

In this scenario:

1. **Nginx** becomes the public entry point (port 80).
2. When a request arrives, Nginx uses the `auth_request` module to make a very fast, internal subrequest to the outpost to check if the user has the correct session cookie.
3. If the outpost responds with `200 OK`, Nginx serves the content.
4. If it responds with `401 Unauthorized`, Nginx redirects the user to log in.

The magic for the Web Radio is that **Nginx handles the audio stream directly to the client**, completely bypassing the outpost container for the heavy data transfer. Because Nginx is optimized for infinite asynchronous connections and does not suffer from Go's strict write timeouts, the stream will remain stable forever while still being protected by authentication at the edge!

```
sequenceDiagram
    autonumber
    actor Client
    participant Cloudflare as Cloudflare Tunnel
    participant Nginx as Nginx (Port 80)
    participant Outpost as Authentik Outpost (Port 9000)
    participant AzuraCast as AzuraCast (Port 24443)

    Client->>Cloudflare: Audio request (/listen/stream.mp3)
    Cloudflare->>Nginx: Forward request
    Nginx->>Outpost: Verify Cookie (auth_request)
    Note over Outpost: Checks session
    Outpost-->>Nginx: 200 OK (User authenticated)
    Nginx->>AzuraCast: Proxy audio stream
```

```
    AzuraCast-->>Nginx: Send audio packets (infinite)
    Nginx-->>Client: Direct audio stream (Infinite stability)
```

---

## Step 1: Central Authentik Configuration

In your central Authentik administration dashboard:

1. **Create the Proxy Provider**:
    - Go to *Directory -> Applications -> Providers -> Create*.
    - Select **Proxy Provider**.
    - Set the mode to **Forward Auth (Single Application)**.
    - Set the external host to `https://radio.example.org`.
    - Set the internal host to `http://192.168.5.3` (the local IP of the radio server).
2. **Create the Remote Outpost**:
    - Go to *Directory -> Applications -> Outposts -> Create*.
    - Select **Proxy** as the type and assign the application created above.
    - Set the *Integration* field to **No Integration**.
    - Click save, open the details of the newly created outpost, and copy the generated **Outpost Token**.

---

## Step 2: Running the Outpost on Docker (Remote Server)

On the machine running the Web Radio (`192.168.5.3`), we create a `docker-compose.yml` file to run the local proxy outpost.

We need the outpost container to talk to the central Authentik server via the internal VPN (NetBird) on port `9000`, but instruct client browsers to perform the login using the public HTTPS URL. To do this, we set the `AUTHENTIK_HOST_BROWSER` variable:

```
services:
  authentik_proxy:
    image: ghcr.io/goauthentik/proxy:2026.5.3
    container_name: authentik_proxy
    ports:
      - "9000:9000"
```

```
      - "9443:9443"
    environment:
      # Internal connection to central Authentik API via NetBird
      AUTHENTIK_HOST: "http://auth.example.org:9000"
      # Public URL for client browser redirects
      AUTHENTIK_HOST_BROWSER: "https://auth.example.org"
      AUTHENTIK_TOKEN: "<YOUR_AK_OUTPOST_TOKEN>"
      AUTHENTIK_INSECURE: "true"
    restart: unless-stopped
```

Start the container with `docker compose up -d`.

---

## Step 3: Nginx Configuration with `auth_request` and Cloudflare Tunnel Troubleshooting

Nginx must act as the gatekeeper. However, there is a catch: **Cloudflare Tunnel overrides the request's `Host` header by default**, setting it to the internal destination hostname (e.g., `docker` or `localhost`). This breaks Authentik's validation, which expects to see the external host `radio.example.org` and the redirect URL in HTTPS.

To solve this issue in a robust manner, **we hardcode the Host to `radio.example.org` and the protocol to `https` directly inside Nginx's Authentik location blocks**.

Here is the complete `/etc/nginx/sites-available/azuracast.conf` file:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;

    client_max_body_size 250M;

    # Optimized buffers for stream
    proxy_buffers 8 16k;
    proxy_buffer_size 32k;

    root /var/www/azuracast;
    index index.html;

    # --- AUTHENTIK FORWARD AUTH INTEGRATION ---
```

```
    # 1. Outpost location block for login, callback, and auth check
    location /outpost.goauthentik.io/ {
        auth_request off; # Disable recursive check on itself

        proxy_pass http://127.0.0.1:9000;
        proxy_ssl_verify off;

        # Override Cloudflare Tunnel Host header rewriting
        proxy_set_header Host radio.example.org;
        proxy_set_header X-Original-URL https://radio.example.org$request_uri;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Forwarded-Host radio.example.org;
        proxy_set_header X-Forwarded-URI $request_uri;

        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }

    # 2. Intercept 401 errors to redirect users to login page
    error_page 401 = @goauthentik_to_login;
    location @goauthentik_to_login {
        # Relative redirect on the local domain managed by the outpost
        return 302 /outpost.goauthentik.io/start?
rd=https://radio.example.org$request_uri;
    }

    # --- APPLICATION PROTECTION ---

    # Protected document root
    location / {
        auth_request /outpost.goauthentik.io/auth/nginx;
        try_files $uri $uri/ =404;
    }

    # Protected APIs
    location /api/ {
        auth_request /outpost.goauthentik.io/auth/nginx;
        proxy_pass https://127.0.0.1:24443/api/;
        proxy_ssl_verify off;

        proxy_set_header Host radio.example.org;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
```

```
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }

    # --- PROTECTED AUDIO STREAM PASSED DIRECTLY BY NGINX (STABLE) ---
    location /listen/ {
        # Check if the user has a valid session on Authentik (Instant
subrequest)
        auth_request /outpost.goauthentik.io/auth/nginx;

        # If OK, Nginx proxies the audio stream directly (No Go timeouts!)
        proxy_pass https://127.0.0.1:24443/listen/;
        proxy_ssl_verify off;

        proxy_buffering off;
        proxy_cache off;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_read_timeout 36000s;

        proxy_set_header Host radio.example.org;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
    }
}
```

Reload Nginx using `sudo systemctl reload nginx`.

---

## Step 4: Cloudflare Tunnel Configuration

Now that Nginx acts as the entry point and orchestrates the authentication, we must configure the Cloudflare Tunnel to point to **Nginx (port 80)** instead of the outpost (port 9000).

In your Cloudflare Zero Trust dashboard, update the route for `radio.example.org`:

- **Service Type**: `HTTP`
- **URL**: `http://localhost:80` (or the internal IP on port 80).

---

## Conclusion

By combining Cloudflare Tunnel, Nginx (`auth_request`), and Authentik Forward Auth, we achieved:

- A web radio portal fully secured by corporate SSO.
- Complete verification ensuring that no one can fetch or listen to the `/listen/` audio stream from the outside without authenticating.
- An infinitely stable media stream, freed from the write timeouts imposed by Authentik's Go outpost proxy.