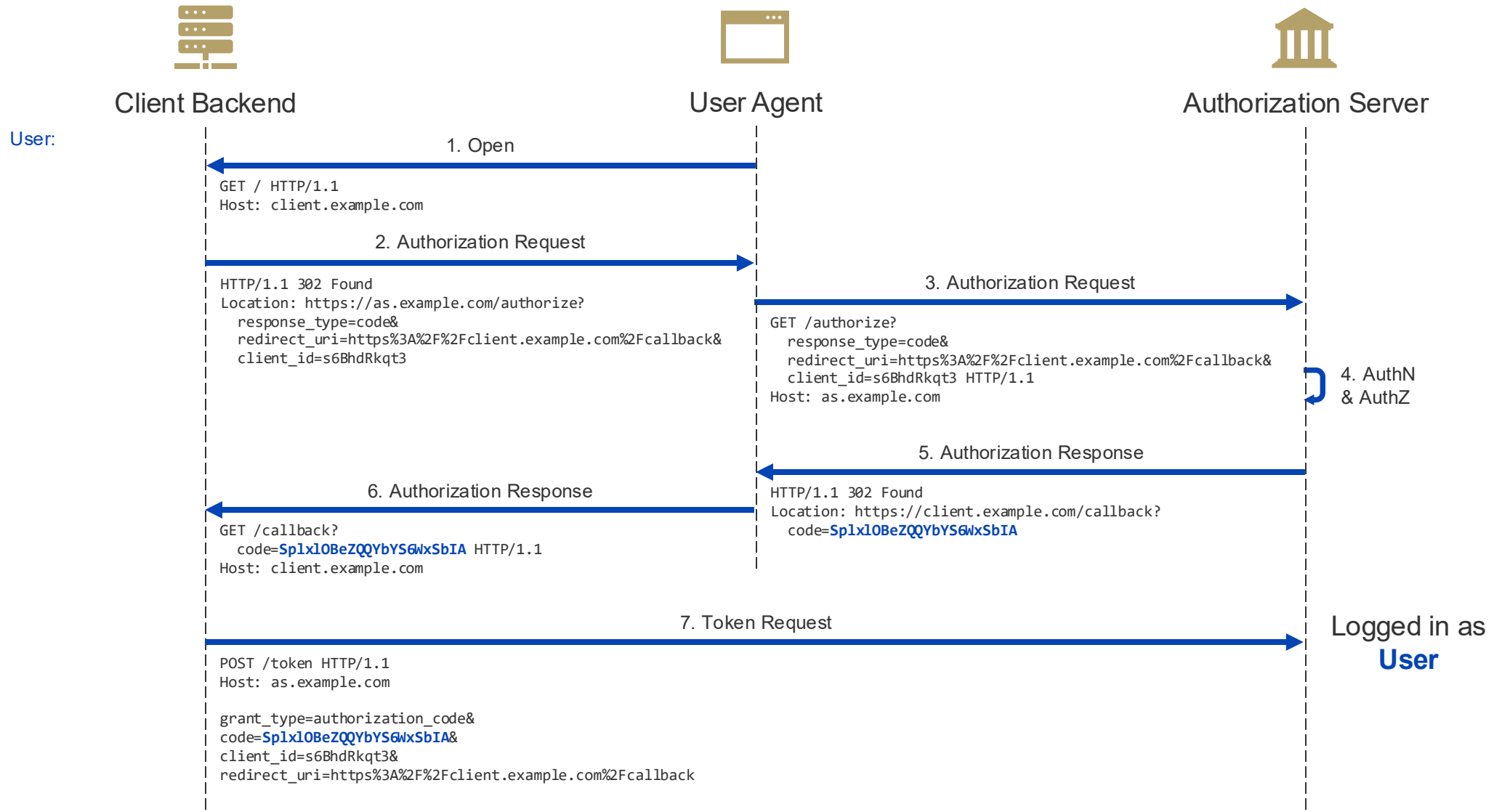
Browser-Swapping Attacks

By Jonas Primbs, SySS GmbH | University of Tübingen

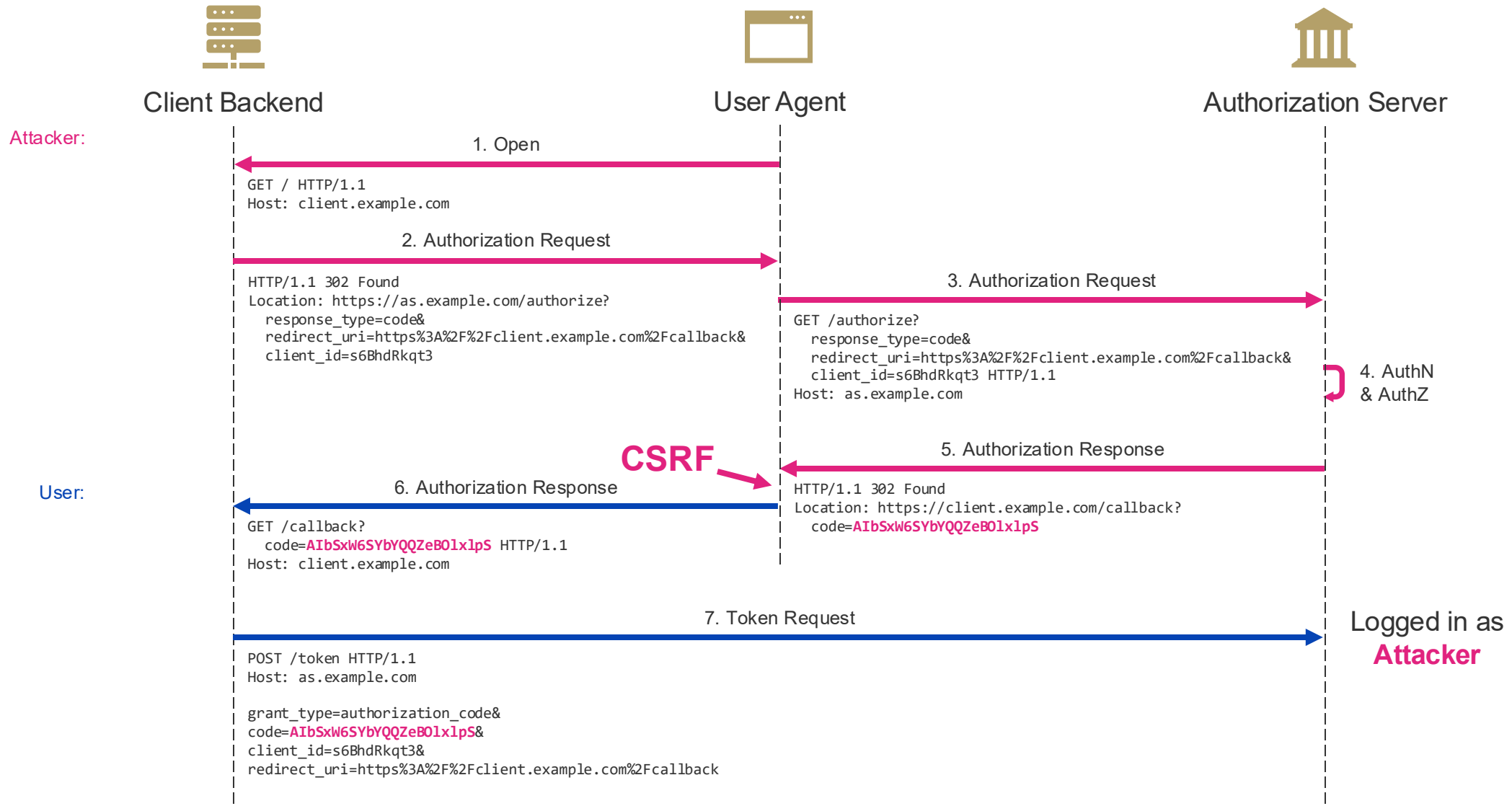
/jonasprimbs

https://www.syss.de | https://kn.cs.uni-tuebingen.de

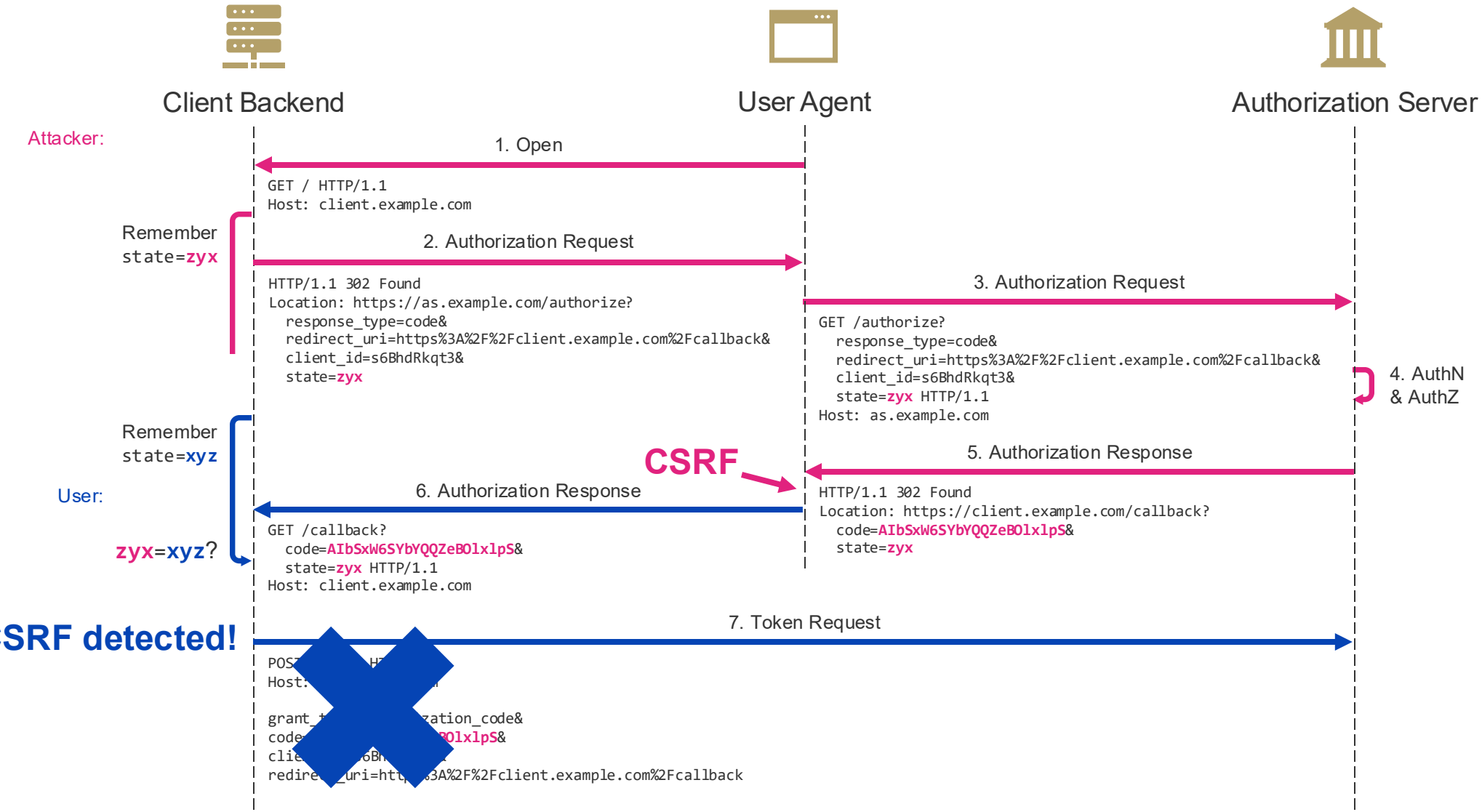
Authorization Code Flow



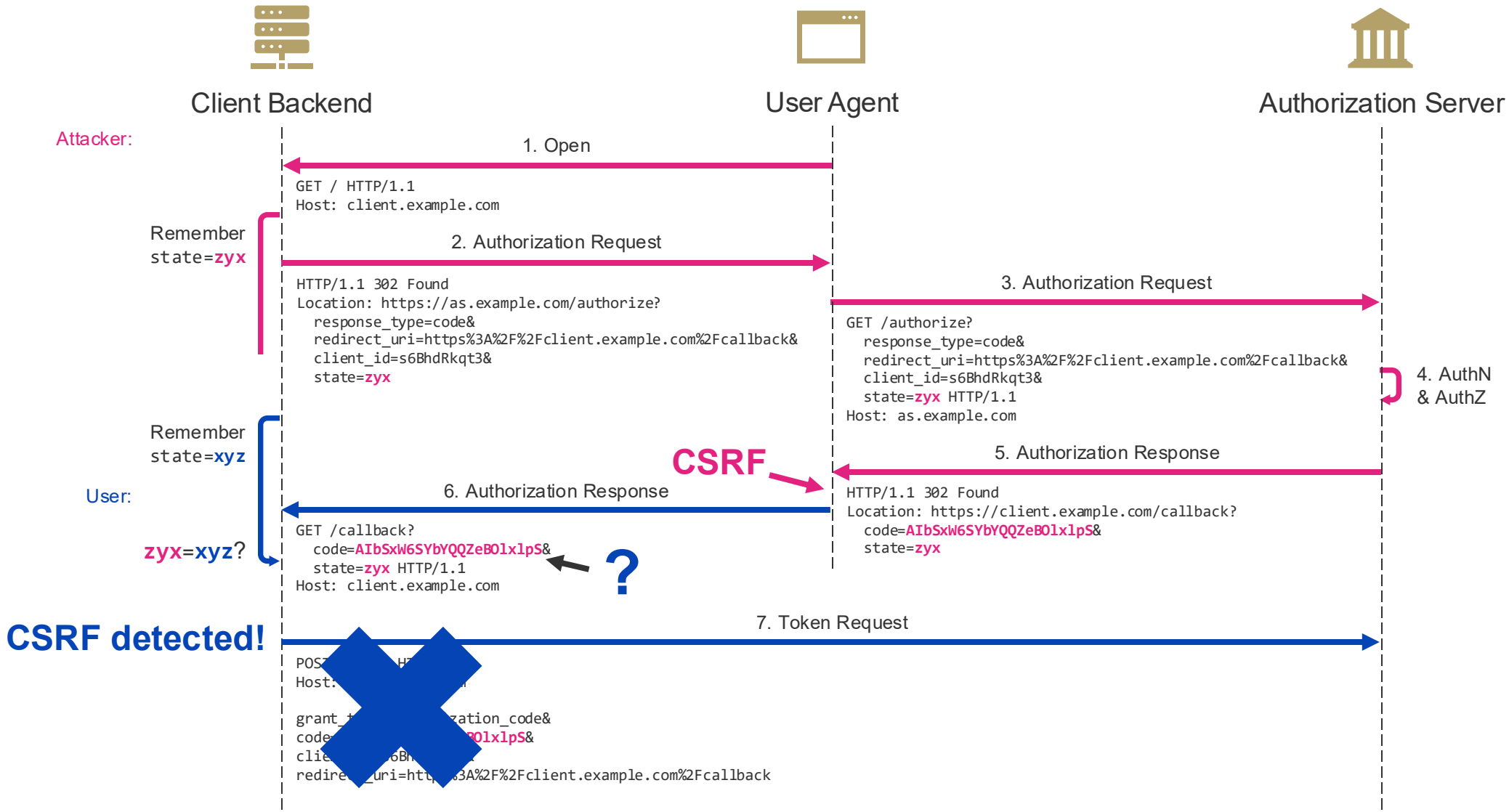
Cross-Site Request Forgery (CSRF) Attack



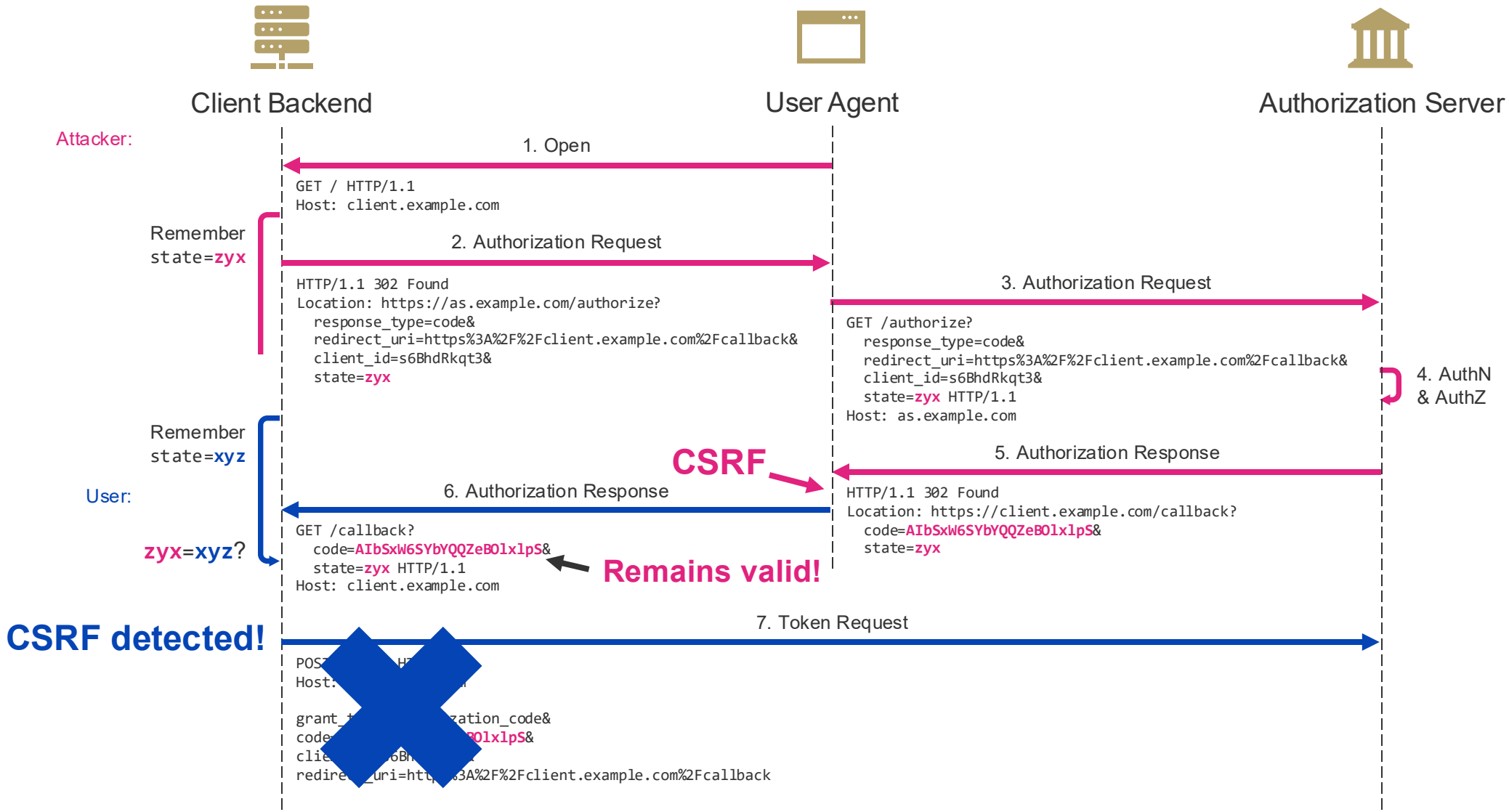
Cross-Site Request Forgery (CSRF) Prevention



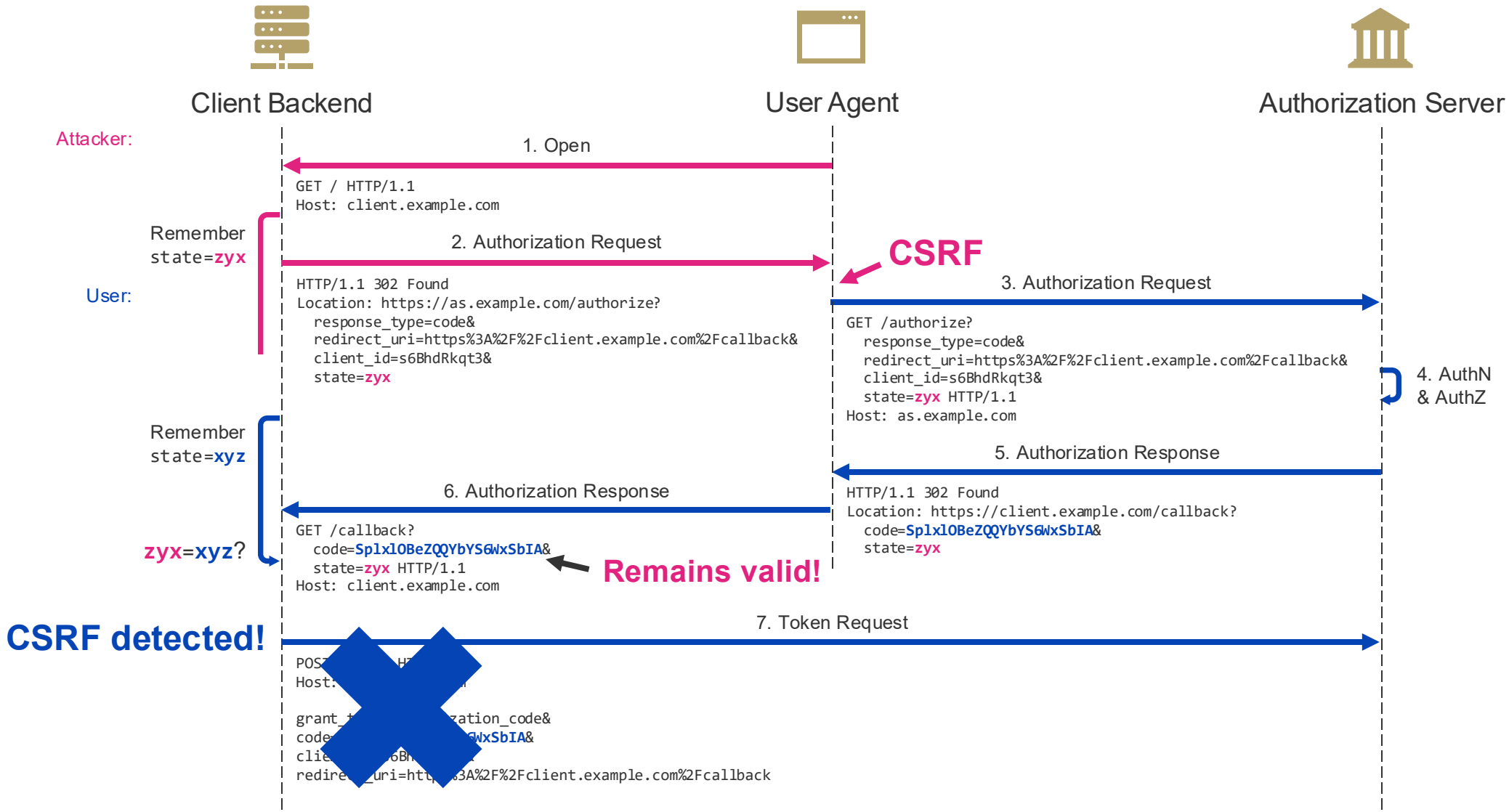
What happens to the Authorization Code?



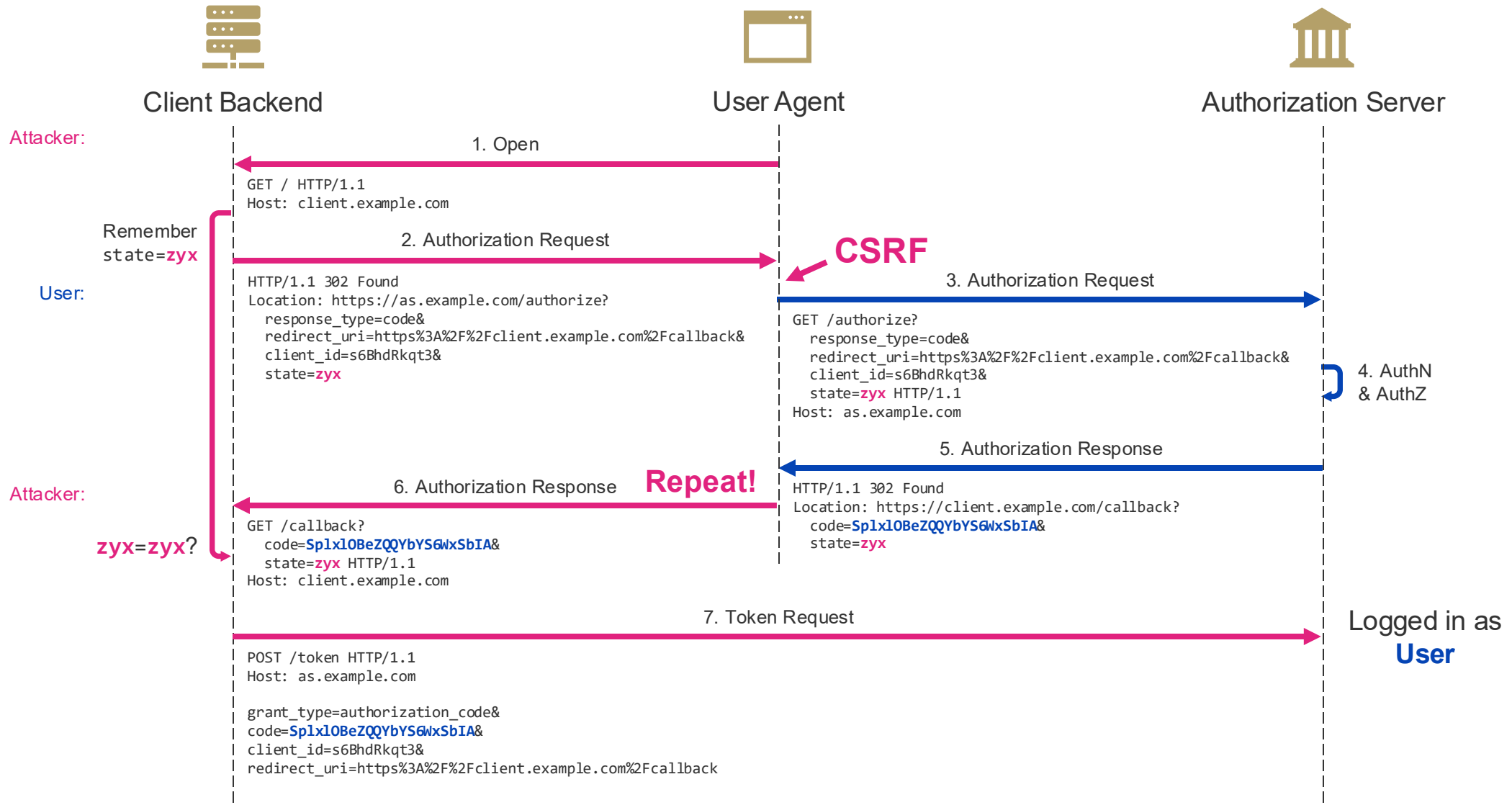
The Authorization Code remains valid!



Browser-Swapping Attack – User Steps

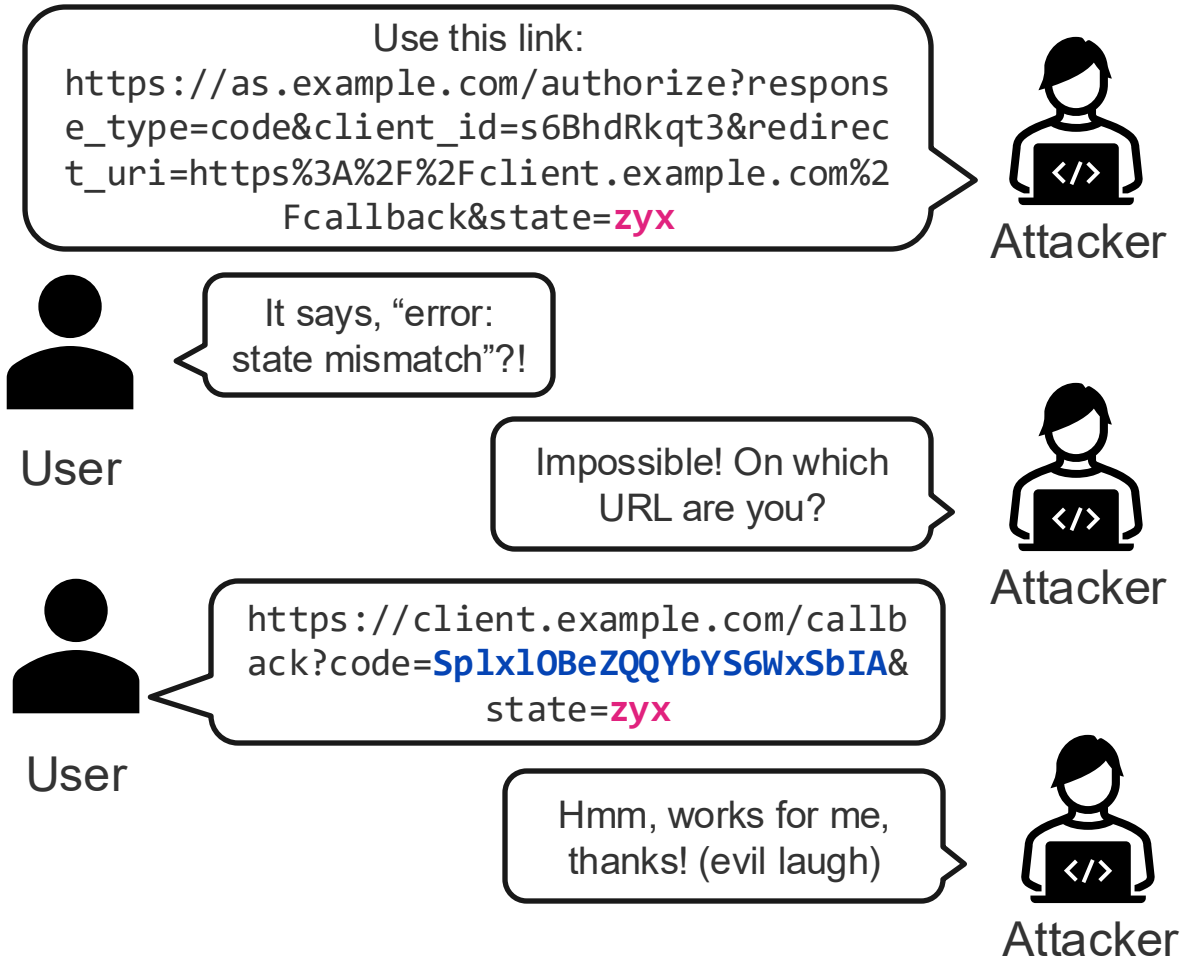


Browser-Swapping Attack – Attacker Steps



How can the Authorization Code be leaked?

- **URL sharing**
  - Users might share their full URL, including query parameters
  - **Solution**: Redirect to sanitized URL after login
    - **Especially on error!**



How can the Authorization Code be leaked?

- **URL sharing**
  - Users might share their full URL, including query parameters
  - **Solution**: Redirect to sanitized URL after login
    - **Especially on error!**

- The `Referer`-header
  - Sent to third-party websites in on link-navigation or when requesting external content
  - **Solution**: `Referrer-Policy: no-referrer`

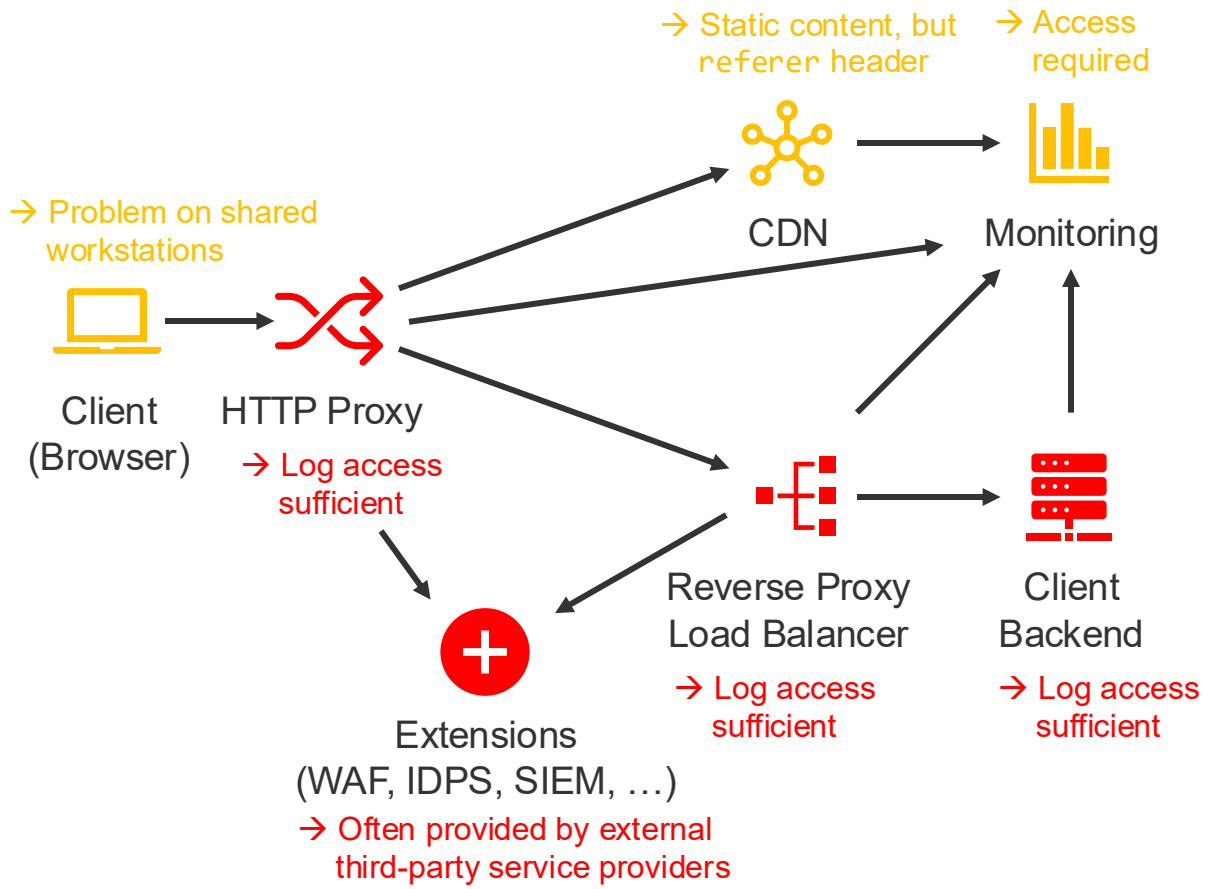
```
GET /favicon.ico HTTP/1.1
Host: external.example.com
Referer: https://client.example.com/callback?code=SplxlOBeZQQYbYS6WxSbIA&state=zyx
```

How can the Authorization Code be leaked?

- **URL sharing**
  - Users might share their full URL, including query parameters
  - **Solution**: Redirect to sanitized URL after login
    - **Especially on error!**
- The **`Referer`-header**
  - Sent to third-party websites in on link-navigation or when requesting external content
  - **Solution**: `Referrer-Policy: no-referrer`
- **Analytics tools**
  - Google Analytics & Facebook Pixel record full URLs
  - **Solution**: No analytics tools on redirect URI

How can the Authorization Code be leaked?

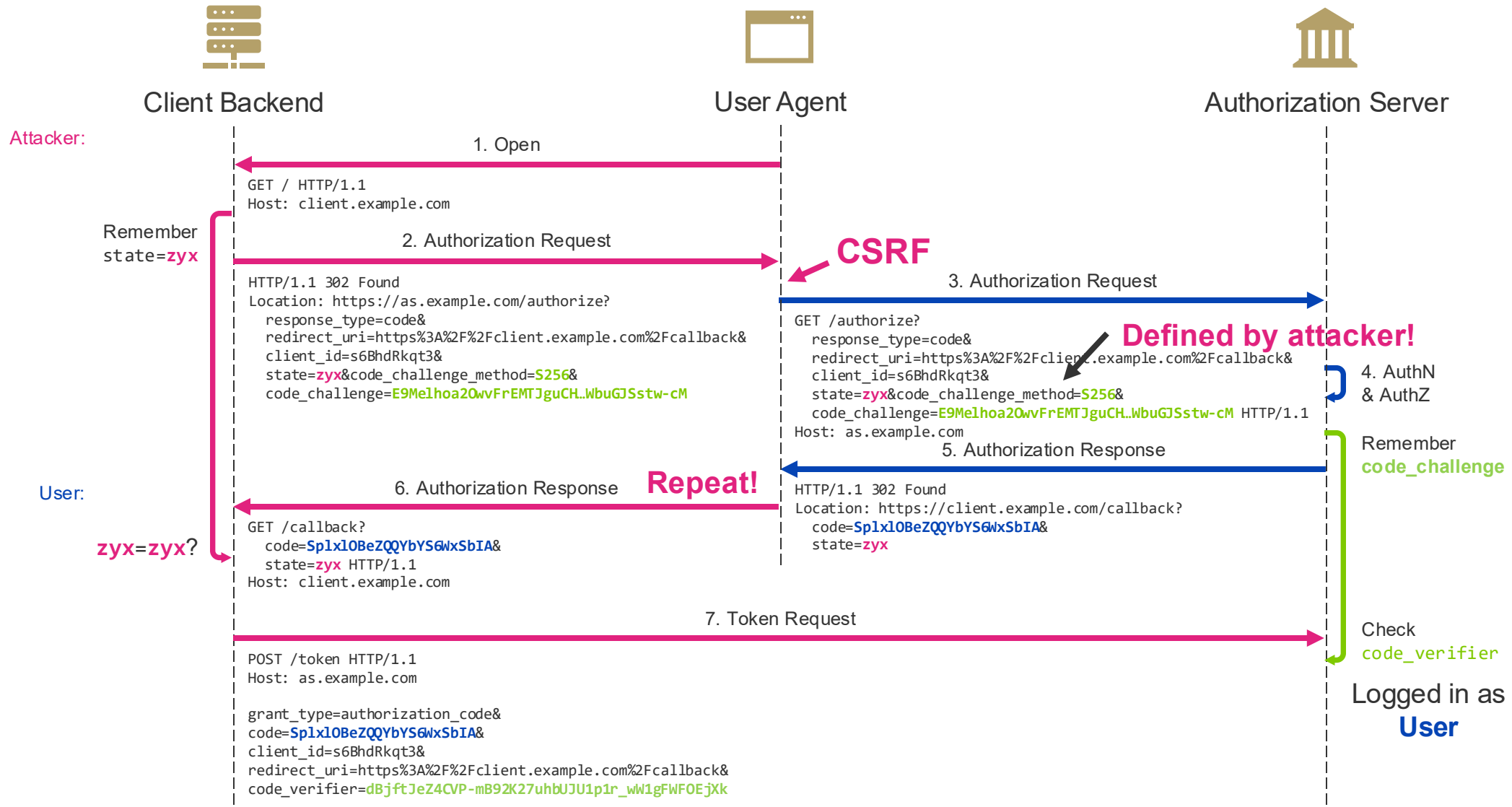
- **URL sharing**
  - Users might share their full URL, including query parameters
  - **Solution**: Redirect to sanitized URL after login
    - **Especially on error!**
- The `Referer`-header
  - Sent to third-party websites in on link-navigation or when requesting external content
  - **Solution**: `Referrer-Policy: no-referrer`
- **Analytics tools**
  - Google Analytics & Facebook Pixel record full URLs
  - **Solution**: No analytics tools on redirect URI
- **Logging**
  - Clients, proxies and servers typically log full URLs
  - Logs are reported to centralized monitoring systems



Browser-Swapping Prevention

- Proof Key for Code Exchange (PKCE)?

Browser-Swapping Prevention: PKCE



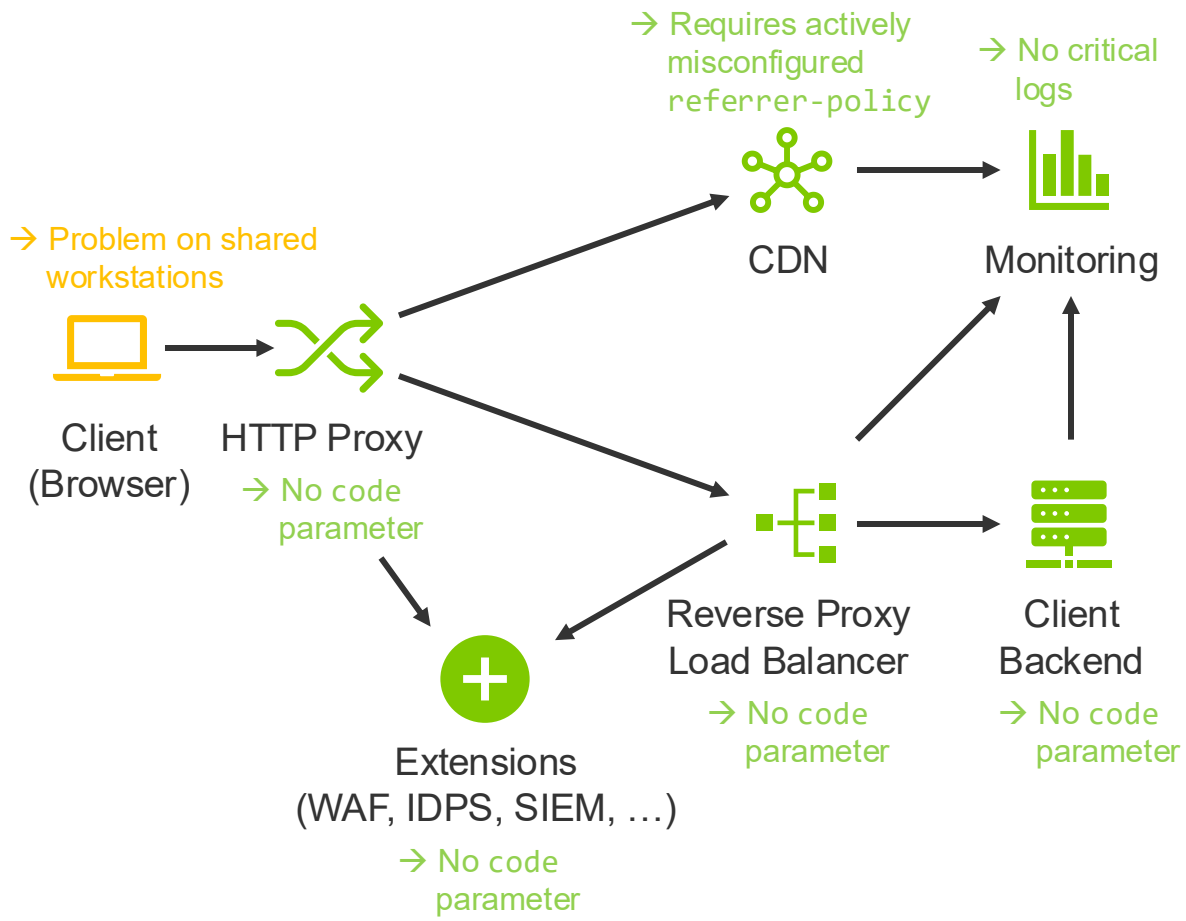
Browser-Swapping Prevention

- Proof Key for Code Exchange (PKCE) → No!
  - PKCE protects the attacker!

Browser-Swapping Prevention

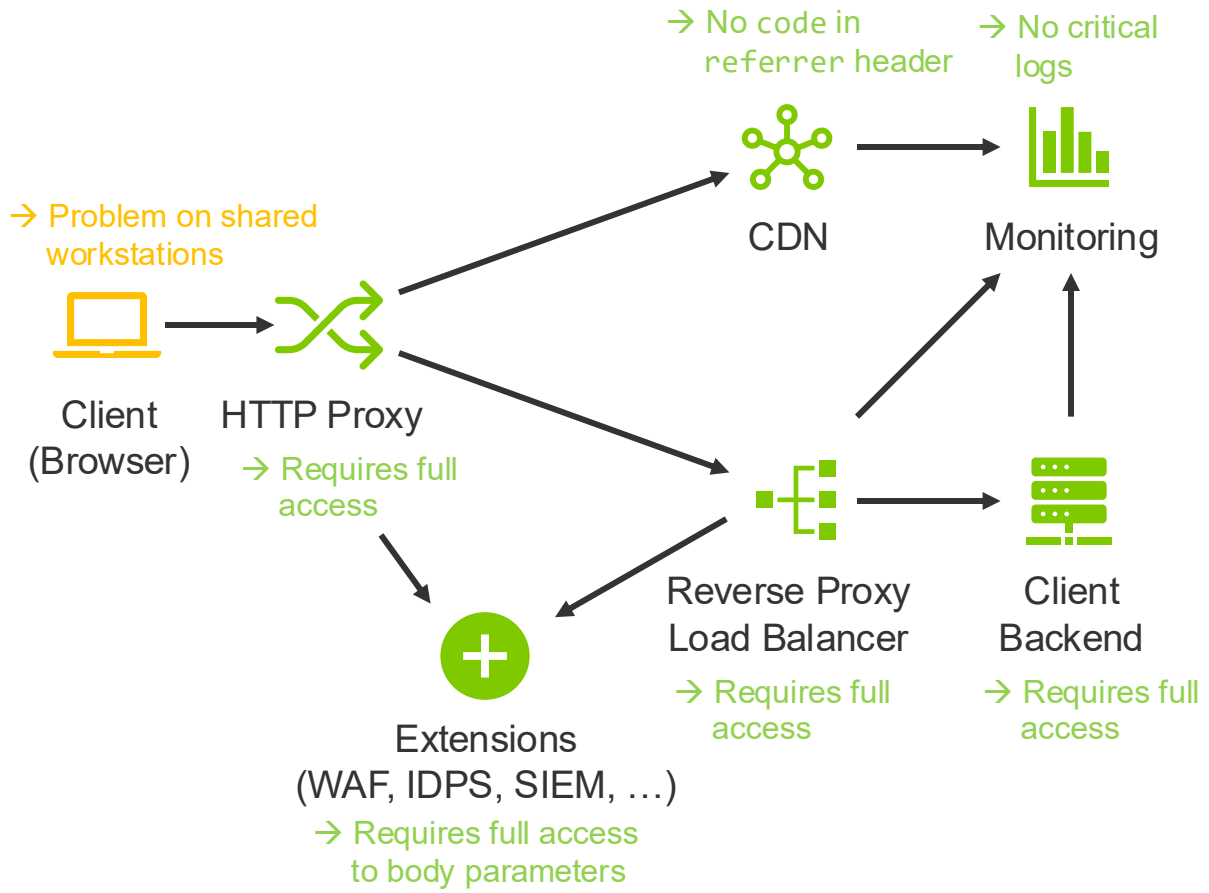
- Proof Key for Code Exchange (PKCE) → No!
  - PKCE protects the attacker!

- `response_mode=fragment`
  - Works for user-side (public) clients
    - Web Apps → `code` parameter must be removed on errors
    - Mobile Apps



Browser-Swapping Prevention

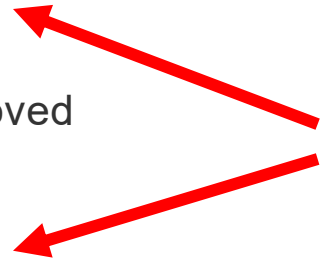
- Proof Key for Code Exchange (PKCE) → No!
  - PKCE protects the attacker!
- `response_mode=fragment`
  - Works for user-side (public) clients
    - Web Apps → `code` parameter must be removed on errors
    - Mobile Apps
- `response_mode=form_post`
  - Works for server-side (confidential) clients



Browser-Swapping Prevention

- Proof Key for Code Exchange (PKCE) → No!
  - PKCE protects the attacker!

- `response_mode=fragment`
  - Works for user-side (public) clients
    - Web Apps → `code` parameter must be removed on errors
    - Mobile Apps
- `response_mode=form_post`
  - Works for server-side (confidential) clients



Can be downgraded to `response_mode=query`!

Browser-Swapping Prevention

- Proof Key for Code Exchange (PKCE) → No!
  - PKCE protects the attacker!

- `response_mode=fragment`
  - Works for user-side (public) clients
    - Web Apps → `code` parameter must be removed on errors
    - Mobile Apps
- `response_mode=form_post`
  - Works for server-side (confidential) clients

- Additional: Invalidate authorization codes suspected of CSRF attacks
  - Clients should always notify the authorization server of used authorization codes
  - OAuth 2.1 v14 only specifies invalidation after a successful token request

Recommendations for OAuth 2.1

1. **Deprecate `response_mode=query`**
   - No secrets (authorization code) in query parameters!

Recommendations for OAuth 2.1

1. **Deprecate `response_mode=query`**
   - No secrets (authorization code) in query parameters!

2. **Response mode enforcement must be mandatory for authorization servers**
   - Prevents downgrading attacks

Recommendations for OAuth 2.1

1. **Deprecate `response_mode=query`**
   - No secrets (authorization code) in query parameters!

2. **Response mode enforcement must be mandatory for authorization servers**
   - Prevents downgrading attacks

3. **Specification of an authorization code invalidation mechanism**
   - Provides clients with the ability to invalidate used authorization codes at the authorization server