Học phần: Công nghệ web (CT275)

An ninh web

Bùi Võ Quốc Bảo | Khoa CNTT

An ninh web

- Một số lỗi bảo mật phổ biến
  - Cross-site Scripting (XSS)
  - SQL Injection
  - Cross-site Request Forgery (CSRF)
  - Insecure Direct Object Reference (IDOR)
- Xử lý đăng nhập người dùng
  - Đăng nhập người dùng bằng tài khoản cục bộ
  - Xác thực đa yếu tố (MFA)
  - Đăng nhập người dùng qua OAuth2/OpenID Connect

Một số lỗi bảo mật phổ biến

- Cross-site Scripting (XSS)
- SQL Injection
- Cross-Site Request Forgery (CSRF)
- Insecure Direct Object Reference (IDOR)

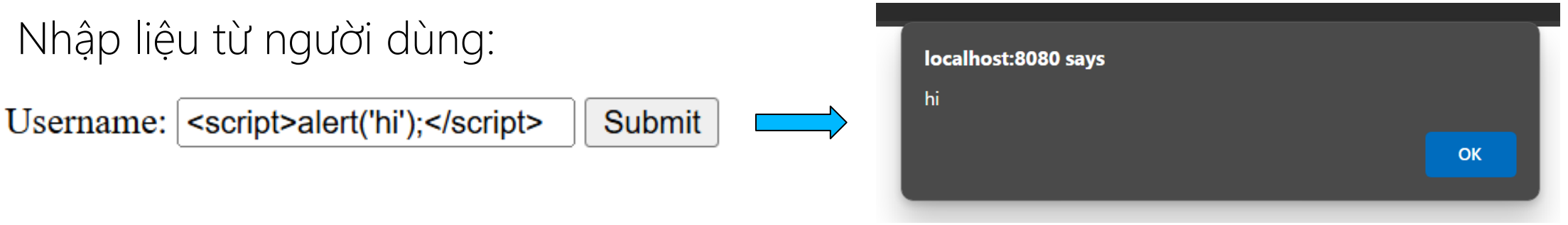
Cross-site Scripting (XSS)

- Website cho phép mã lệnh độc hại được chèn vào và thực thi phía người dùng
- Ví dụ: Cho đoạn mã lệnh xử lý form như sau

```php
<?php

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $username = $_POST['username'];
  echo 'Hello, ' . $username;
}
```

Nhập liệu từ người dùng:



Cross-site Scripting (XSS)

- Website cho phép mã lệnh độc hại được chèn vào và thực thi phía người dùng

- Cách đề phòng: *"làm sạch" (sanitisation)* dữ liệu trước khi xuất dữ liệu
  - Dùng hàm htmlspecialchars() hoặc một template engine để xuất và trình bày dữ liệu ra trang HTML

Cross-site Scripting (XSS)

- Website cho phép mã lệnh độc hại được chèn vào và thực thi phía người dùng

- Cách đề phòng: sử dụng header *Content-Security-Policy* trong câu trả lời HTTP
  - Header Content-Security-Policy ngăn trang web tải tài nguyên (JavaScript, CSS, ảnh, ...) từ nguồn bên ngoài
  - Có thể cấu hình máy chủ web tự động thêm header Content-Security-Policy vào câu trả lời hoặc dùng thẻ <meta http-equiv="Content-Security-Policy"> trong trang web

```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
  <meta http-equiv="Content-Security-Policy" content="default-src 'self' https://cdn.jsdelivr.net" />
```

SQL Injection

- Website cho phép mã lệnh SQL được đưa vào và thực thi phía server từ nhập liệu phía client
- Ví dụ: Cho đoạn mã lệnh xử lý form sau:

```
$userId = $_POST['userId'];
$SQL = 'SELECT * FROM Users WHERE id = ' . $userId;
```

Nhập liệu từ người dùng:

UserId: 105 OR 1=1 →

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Trả về tất cả người dùng trong CSDL

UserId: 105; drop table Suppliers; →

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;
```

Trả về thông tin người dùng có UserId = 105 và xóa bảng Suppliers

SQL Injection

- Website cho phép mã lệnh SQL được đưa vào và thực thi phía server từ nhập liệu phía client

- Cách đề phòng: *"làm sạch" (sanitisation)* dữ liệu trước khi đưa dữ liệu vào câu truy vấn SQL và thực thi
  - Dùng PDO với Prepared Statement để truyền tham số vào câu truy vấn (dùng tham số dạng *:varname* hoặc *?*)

Cross Site Request Forgery (CSRF)

- Website cho phép người dùng thực thi một hành động mà họ không có ý định thực hiện (do bị đánh lừa thực hiện)
- Ví dụ: Giao dịch chuyển khoản của một ngân hàng được thực thi với một yêu cầu POST như sau

```
POST http://acmebank.com/fundtransfer HTTP/1.1

acct=344344&amount=5000
```

Người dùng bị lừa submit form giả mạo sau (khi người dùng đang đăng nhập):

```
<form action="http://acmebank.com/fundtransfer" method="POST">
  <input type="hidden" name="acct" value="224224" />
  <input type="hidden" name="amount" value="50000" />
  <input type="submit" value="Click to get your free gift!" />
</form>
```

Cross Site Request Forgery (CSRF)

- Website cho phép người dùng thực thi một hành động mà họ không có ý định thực hiện (do bị đánh lừa thực hiện)
  - Người dùng bị đánh lừa submit một form "không chính chủ" về server
- Cách đề phòng: đảm bảo server chỉ chấp nhận form do chính server tạo ra
  - Đính kèm một CSRF token (thường sinh ra phía server tại mỗi phiên) vào form. Khi form submit về server thì kiểm tra sự hiện diện và tính hợp lệ của CSRF token trước khi thực thi hành động

```
<form action="/transfer.php" method="post">
  <input type="hidden" name="CSRFToken"
         value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZMGYwMGEwOA==">
  [...]
</form>
```

Cross Site Request Forgery (CSRF)

- Tạo token cho phiên làm việc:

```php
<?php

session_start();

if (! isset($_SESSION['sesskey'])) {
  $_SESSION['sesskey'] = bin2hex(random_bytes(32));
}
```

- Đính kèm sesskey vào form:

```php
<form action="/fundtransfer" method="POST">
  <input type="hidden" name="sesskey" value="<?= $_SESSION['sesskey']; ?>" />
  <input type="text" name="acct" value="0" />
  <input type="text" name="amount" value="0" />
  <input type="submit" value="Transfer" />
</form>
```

Cross Site Request Forgery (CSRF)

- Đính kèm sesskey vào form:

```
<form action="/fundtransfer" method="POST">
  <input type="hidden" name="sesskey" value="<?= $_SESSION['sesskey']; ?>" />
  <input type="text" name="acct" value="0" />
  <input type="text" name="amount" value="0" />
  <input type="submit" value="Transfer" />
</form>
```

- Kiểm tra sesskey trước khi xử lý form:

```
// ...
if (
    isset($_POST['sesskey']) &&
    hash_equals($_SESSION['sesskey'], $_POST['sesskey'])
) {
    // Proceed to process the form data
}
```

Insecure Direct Object Reference (IDOR)

- Website cho phép người dùng truy xuất hoặc cập nhật các đối tượng không được phép thông qua việc thay đổi định danh trên URL hoặc trong tham số yêu cầu
- Ví dụ 1: Bằng cách thay đổi tham số định danh trên URL https://example.org/profile.php?id=123 người dùng có thể truy xuất thông tin của người dùng khác
- Ví dụ 2: Bằng cách thay đổi tham số user_id trong form, người dùng có thể cập nhật thông tin của người dùng khác

```
<form action="/update_profile" method="post">
  <!-- Other fields for updating name, email, etc. -->
  <input type="hidden" name="user_id" value="12345">
  <button type="submit">Update Profile</button>
</form>
```

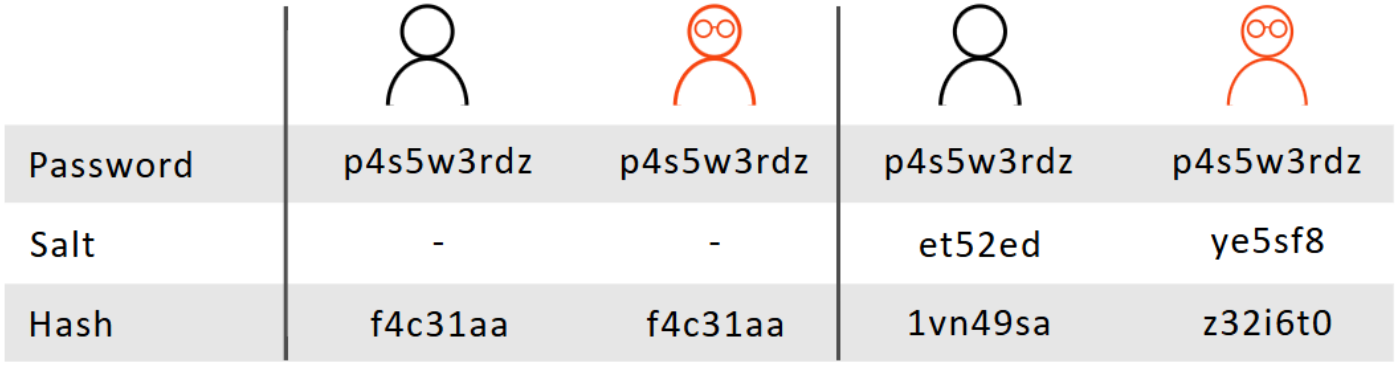
Insecure Direct Object Reference (IDOR)

- Website cho phép người dùng truy xuất hoặc cập nhật các đối tượng không được phép thông qua việc thay đổi định danh trên URL hoặc trong tham số yêu cầu

- Cách đề phòng:
  - Cài đặt điều khiển truy cập, kiểm tra quyền người dùng trên từng đối tượng được truy xuất
  - Có thể kết hợp điều khiển truy cập với việc sử dụng trường định danh dài, phức tạp, ngẫu nhiên (e.g., UUID) thay vì số tự động tăng

Xử lý đăng nhập người dùng

- Đăng nhập người dùng bằng tài khoản cục bộ
  - ✓ Không lưu mật khẩu người dùng dạng plain text, thay vào đó lưu giá trị băm
  - ✓ Thêm một giá trị ngẫu nhiên salt trước khi băm để tránh người dùng cùng mật khẩu có cùng giá trị băm



Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	1vn49sa	z32i6t0

Xử lý đăng nhập người dùng

- Đăng nhập người dùng bằng tài khoản cục bộ
  - ✓ Trong PHP, password_hash() sẽ tự động sinh giá trị salt

```
echo password_hash('123456789', PASSWORD_DEFAULT);
```

```
$2y$10$k/oMFbDkZ8UlnuXihOAloex84zQM.qgxKVKiL7iIQRvCR6.cXdTja
```

Giải thuật bcrypt

2^10 = 1024 lần lặp

Chuỗi salt

Chuỗi băm của mật khẩu

Xử lý đăng nhập người dùng

- Đăng nhập người dùng bằng tài khoản cục bộ
  - ✓ Dùng password_verify() để kiểm tra mật khẩu

```
$hash = '$2y$10$k/oMFbDkZ8UlnuXihOAloex84zQM.qgxKVKiL7iIQRvCR6.cXdTja';

if (password_verify('123456789', $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
}
```

Xử lý đăng nhập người dùng

- Đăng nhập người dùng bằng tài khoản cục bộ
  - ✓ Ngoài salt, nên thêm vào một giá trị ngẫu nhiên ***bí mật*** pepper để tăng cường tính bảo mật

```
function hash_password($password)
{
   $pepper = getenv('PASSWORD_PEPPER');
   $password_peppered = hash_hmac('sha256', $password, $pepper);
   return password_hash($password_peppered, PASSWORD_DEFAULT);
}
```

Đọc từ biến môi trường

Xử lý đăng nhập người dùng

- Đăng nhập người dùng bằng tài khoản cục bộ
  - ✓ Ngoài salt, nên thêm vào một giá trị ngẫu nhiên ***bí mật*** pepper để tăng cường tính bảo mật

```
function verify_password($password, $hashed_password)
{
   $pepper = getenv('PASSWORD_PEPPER');
   $password_peppered = hash_hmac('sha256', $password, $pepper);
   return password_verify($password_peppered, $hashed_password);
}
```

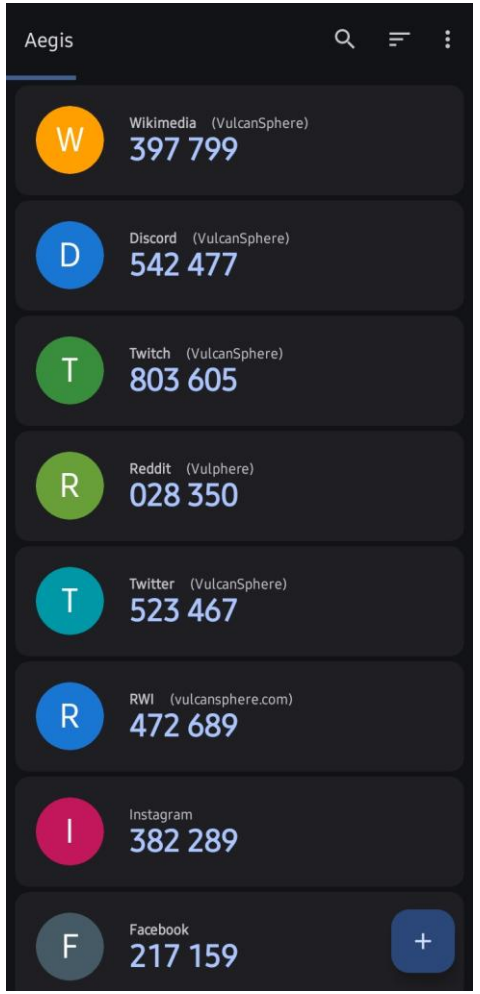
Đọc từ biến môi trường

Xử lý đăng nhập người dùng

- Xác thực đa yếu tố (MFA): Yêu cầu thêm các yếu tố xác thực ngoài cặp tên đăng nhập – mật khẩu thông thường
- Các yếu tố MFA phổ biến
  - ✓ Yếu tố kiến thức (Something you know): Mật khẩu, mã PIN
  - ✓ Yếu tố sở hữu (Something you have): Mã OTP (như TOTP) từ ứng dụng di động, khóa bảo mật vật lý
  - ✓ Yếu tố sinh trắc học (Something you are): Vân tay, nhận diện khuôn mặt

Xử lý đăng nhập người dùng

- TOTP (Time-based One-Time Password – mật khẩu một lần dựa trên thời gian) là một hình thức MFA sử dụng mã được tạo ngẫu nhiên làm mã thông báo xác thực bổ sung
  - ✓ Các mã TOTP thường được tạo thông qua ứng dụng trên điện thoại thông minh (ví dụ: Google Authenticator)
  - ✓ Mỗi mã TOTP chỉ có hiệu lực trong một khoảng thời gian rất ngắn và liên tục được làm mới (khoảng 30 giây)



Xử lý đăng nhập người dùng

- TOTP (Time-based One-Time Password) thực chất chỉ là một khóa bí mật kết hợp với thời gian hiện tại và được đưa qua một thuật toán (mặc định là HMAC-SHA1)
- Cả phía client (ví dụ: Google Authenticator) và phía server (ứng dụng PHP) đều tính toán cùng một công thức

$$\mathrm{OTP} = \mathrm{hmac\_sha1}(\mathrm{secret}, \mathrm{floor}(\mathrm{current\_time} \,/\, 30\mathrm{s}))$$

Cả hai phía phải thống nhất về khái niệm "thời gian hiện tại", ít nhất là trong một khoảng sai số nhỏ (mặc định thường là ±30 giây).

Xử lý đăng nhập người dùng

- Xác thực đa yếu tố (MFA)
  - ✓ Case study: Cài đặt xác thực MFA với thư viện TwoFactorAuth
  - ✓ Tài liệu tham khảo:
    - https://robthree.github.io/TwoFactorAuth/getting-started.html
    - https://robthree.github.io/TwoFactorAuth/improved-code-verification.html
    - https://robthree.github.io/TwoFactorAuth/optional-configuration.html

Xử lý đăng nhập người dùng

- Việc trực tiếp quản lý tài khoản người dùng thường đòi hỏi nhiều tác vụ khác: đăng ký, đổi mật khẩu, khôi phục mật khẩu, ...
  - ✓ Xem xét sử dụng một mô-đun chứng thực sẵn dùng như PHP-Auth

- Đăng nhập người dùng qua OAuth2/OpenID Connect
  - ✓ Ủy thác chứng thực người dùng cho một Identify Provider (IdP) như Google, Facebook, Github, ...
    - Thường dùng các chuẩn OAuth 2.0 và OpenID Connect
    - Các IdP mã nguồn mở/tự host: Keycloak, Authentik, Authelia, ...

Xử lý đăng nhập người dùng

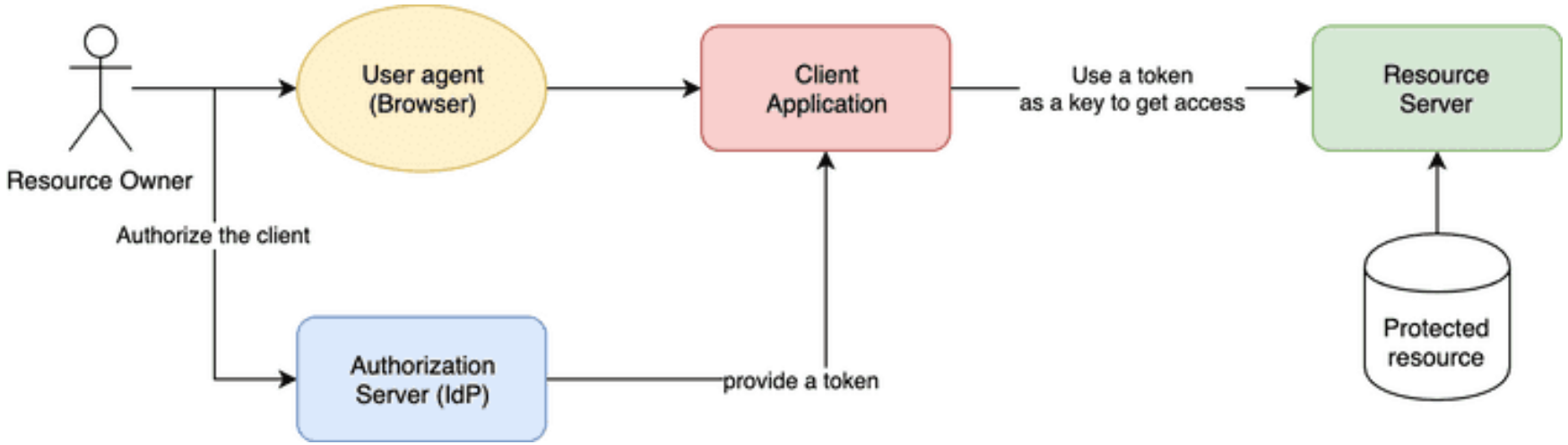
- Đăng nhập người dùng qua OAuth2/OpenID Connect
  - ✓ Các chuẩn giao thức cho chứng thực và phần quyền:
    - OAuth 2.0: một *giao thức phân quyền* cho phép một ứng dụng bên thứ ba truy cập vào một dịch vụ HTTP

      E.g, người dùng ủy quyền các ứng dụng thay mặt họ truy xuất các dịch vụ của Google
    - OpenID Connect: *một layer định danh* đơn giản trên nền tảng của OAuth 2.0 cho phép các ứng dụng bên thứ ba kiểm tra định danh của người dùng mà không tiết lộ thông tin mật khẩu cho ứng dụng

      E.g, người dùng dùng tài khoản định danh Google để đăng nhập vào ứng dụng khác

Xử lý đăng nhập người dùng

- Đăng nhập người dùng qua OAuth2/OpenID Connect
  - ✓ Các thành phần chính trong kiến trúc OAuth 2.0:



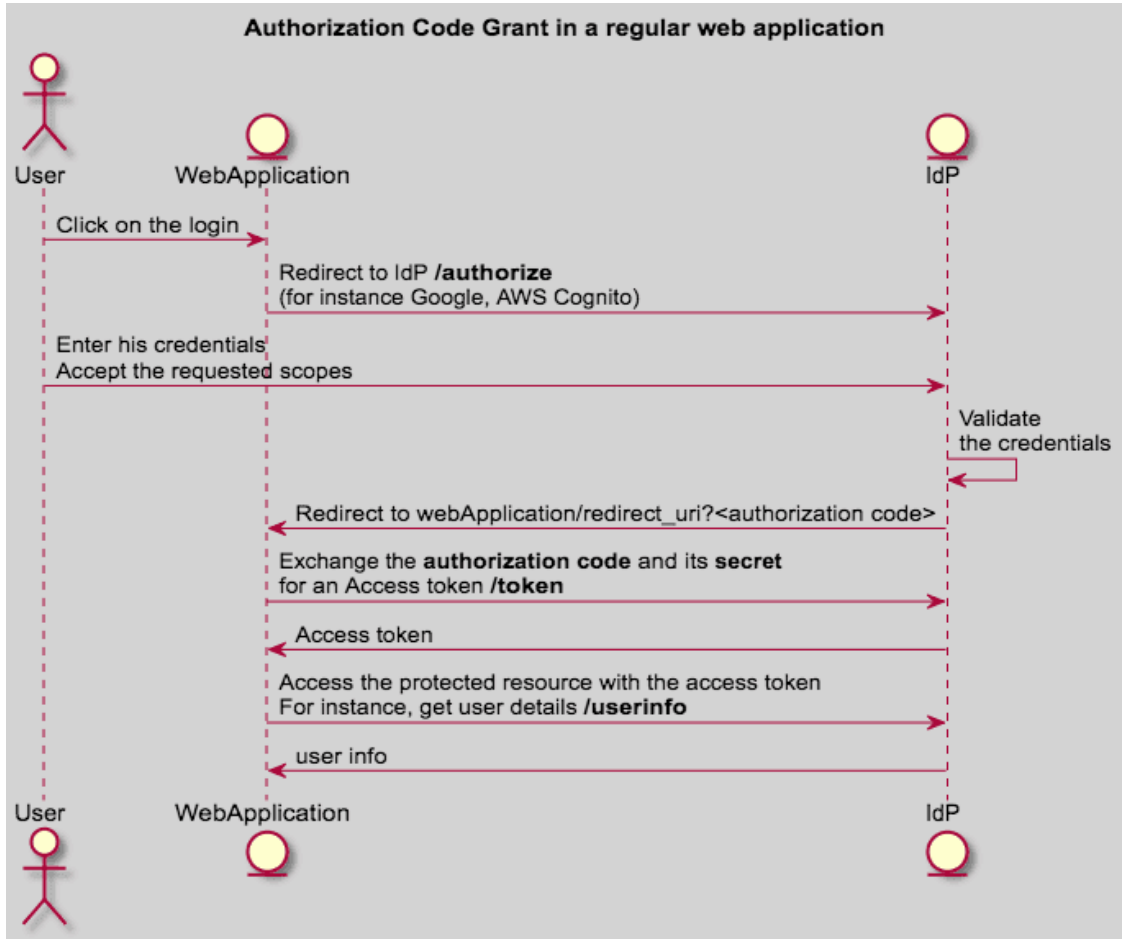
OAuth 2.0 dùng các luồng cấp quyền (grant flow) để cấp phát access token, (và refresh token), và với OpenID Connect có thêm Id token ở định dạng JWT

Xử lý đăng nhập người dùng

- Đăng nhập người dùng qua OAuth2/OpenID Connect

Authorization Code Grant

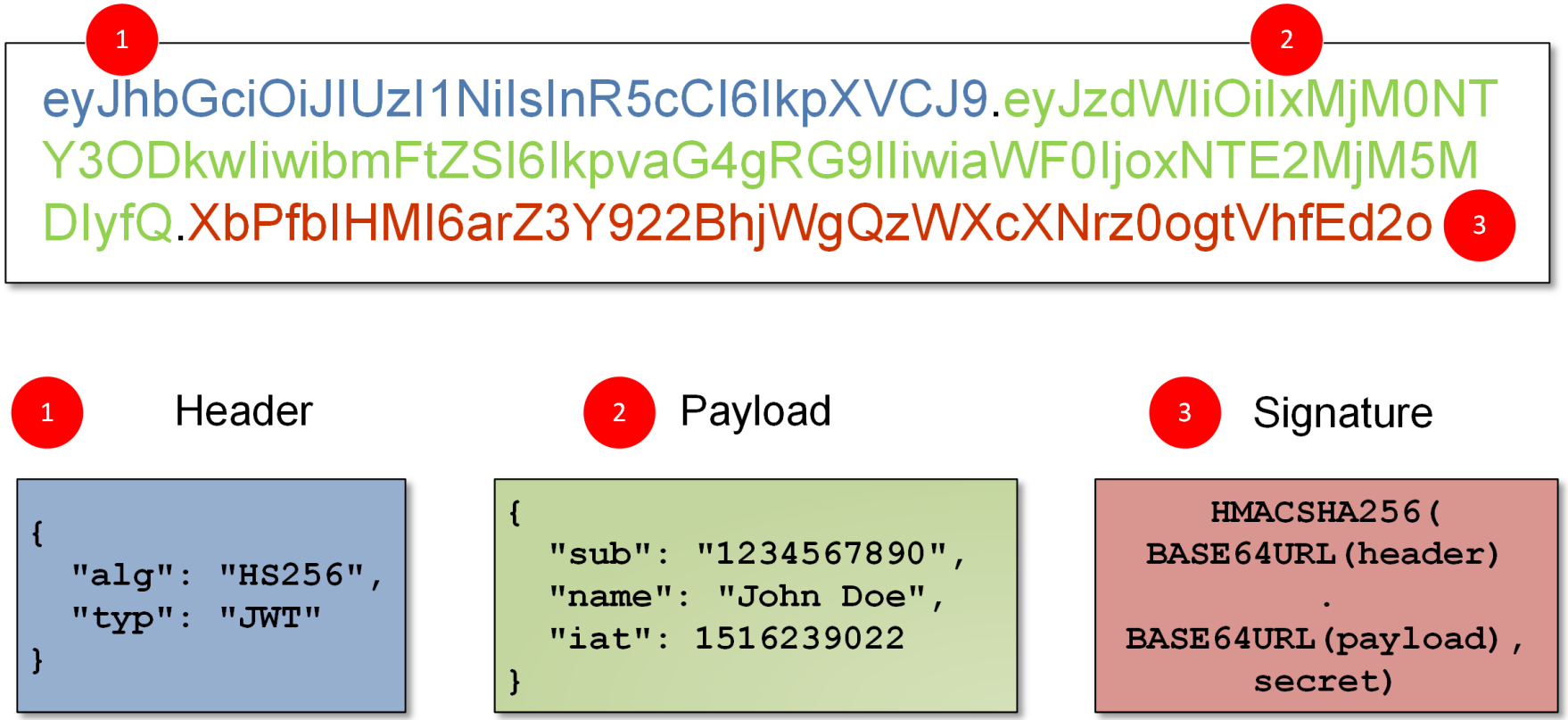
- Được sử dụng khi ứng dụng có khả năng che giấu một mã bí mật (e.g., có phần xử lý ở phía server)
- Cấp các token qua hai bước
- Sau khi người dùng ủy quyền, IdP xác nhận thành công sẽ chuyển hướng về cho server với một authorization code
- Server trao đổi authorization code và mã bí mật để lấy các token từ IdP



Xử lý đăng nhập người dùng

- Đăng nhập người dùng qua OAuth2/OpenID Connect

Định dạng JSON Web Token (JWT)



Xử lý đăng nhập người dùng

- Các giải pháp xử lý luồng cấp quyền của OAuth

1. Cài đặt các chức năng của một client OAuth bên trong website
   - league/oauth2-client, jumbojett/openid-connect-php, google-auth-library-php, ...
2. Đặt một OAuth proxy phía trước website
   - Proxy tự host: OAuth2-Proxy, lua-resty-openidc, caddy-security, ...
   - Proxy được quản lý bởi IdP: Authentik Proxy Provider (dùng OAuth2-Proxy)
3. Dùng một Auth-as-a-Service
   - Firebase Auth, Supabase Auth, PocketBase, ...

Xử lý đăng nhập người dùng

- Đăng nhập người dùng qua OAuth2/OpenID Connect
  - ✓ Case study: Làm việc với OpenID Connect của Google
  - ✓ Tài liệu tham khảo:
    - https://developers.google.com/identity/openid-connect/openid-connect
    - https://accounts.google.com/.well-known/openid-configuration
    - https://developers.google.com/oauthplayground/

An ninh web

- Câu hỏi?