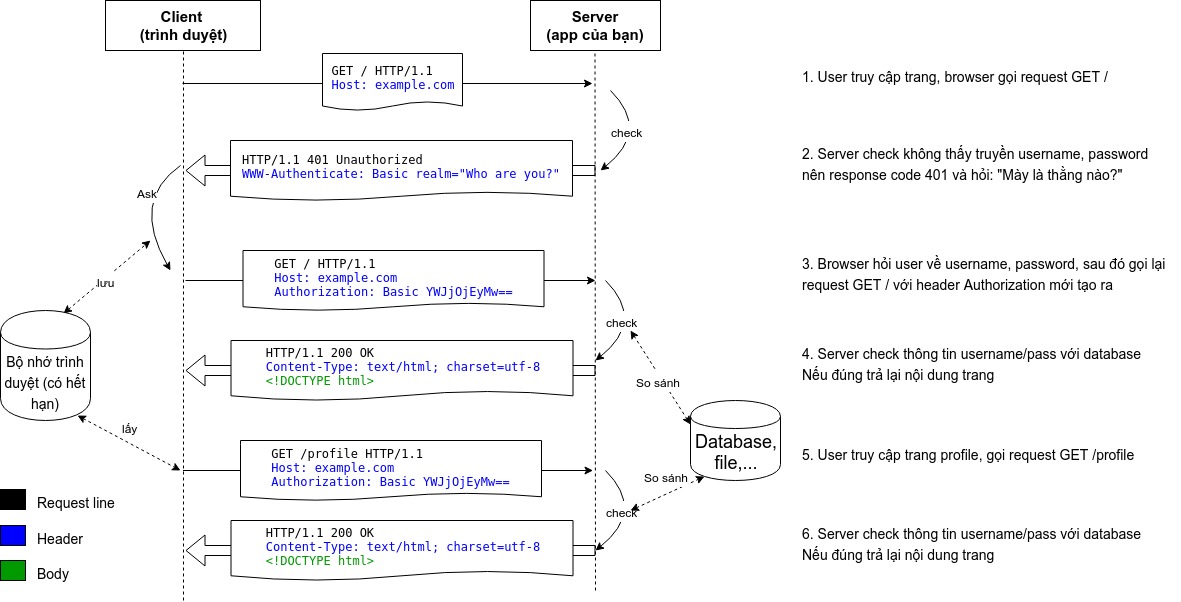
1. **Authentication.**
   1. **HTTP basic**

* Basic Auth là cơ chế xác thực đơn giản nhất của một ứng dụng web, việc ứng dụng nó rất dễ dàng và đã được nhiều phần mềm / máy chủ tích hợp và xử lý một cách tự động.
* Cách hoạt động của Basic Auth là gửi chính username + password của người dùng theo mỗi request.
  + Dấu hiệu: Chuỗi username:password đã được mã hóa Base64. Ví dụ có username là abc, password là 123 thì ta tạo chuỗi mã hóa: abc:123 --Base64--> YWJjOjEyMw==
  + Lưu trữ dấu hiệu:
    - Tại server: Máy chủ web sẽ lưu lại username, password trong database, file (htpasswd),...
    - Tại client: Sau khi hỏi người dùng nhập username và password lần đầu, browser sẽ lưu lại 2 giá trị này trong bộ nhớ được quản lý bởi mỗi trình duyệt (và chúng ta không thể tiếp cận bộ nhớ này bằng code trên trang) để tránh phải liên tục hỏi chúng ta username, password. Tuy nhiên thời gian lưu thường là có giới hạn.
    - Truyền tải: chuỗi đã mã hóa base64 phía trên sẽ được truyền trong HTTP request trong Authorization header với từ khóa Basic phía trước: Authorization: Basic YWJjOjEyMw==
  + Kiểm tra dấu hiệu: Với mỗi request gửi lên kèm thông tin username/password trên, server sẽ so sánh username/password với database, config file,... để kiểm tra tính hợp lệ.
* Flow của Basic Auth:  
  
* Ưu điểm:
  + Đơn giản, do đó được hầu hết các trình duyệt, webserver (nginx, apache,...) hỗ trợ. Có thể dễ dàng config cho webserver sử dụng Basic Auth với 1 vài dòng config.
  + Dễ dàng kết hợp với các phương pháp khác. Do đã được xử lý mặc định trên trình duyệt và webserver thông qua truyền tải http header, các bạn có thể dễ dàng kết hợp phương pháp này với các phương pháp sử dụng cookie, session, token,...
* Nhược điểm:
  + Username/password dễ bị lộ. Do mỗi request đều phải truyền username và password nên sẽ tăng khả năng bị lộ qua việc bắt request, log server,... Không tránh được tấn công replay, mitm, …
  + Không thể logout. Vì việc lưu username, password dưới trình duyệt được thực hiện tự động và không có sự can thiệp của chủ trang web. Do vậy không có cách nào logout được người dùng ngoại trừ việc tự xóa lịch sử duyệt web hoặc hết thời gian lưu của trình duyệt.
  + Không thân thiện với người dùng. Việc hiển thị hộp thoại đăng nhập cũng như thông báo lỗi của trình duyệt, như các bạn đã biết là vô cùng nhàm chán, không chứa đựng nhiều thông tin cho người dùng.
  1. **HTTP digest**
* Sự khác biệt giữa digest authen và basic authen là digest authen không gửi username và password dạng cleartext trên đường truyền. Nó cũng chống được tấn công mitm, replay.
* Cách hoạt động:
  + Client gửi request truy cập đến server.
  + Client nhận về 1 giá trị nonce từ the server và 401 authentication request
  + Client gửi lại 1 array gồm (username, realm, generate\_md5\_key(nonce, username, realm, URI, password\_given\_by\_user\_to\_browser))
  + The server giữ lại username and realm (plus it knows the URI the client is requesting) và kiểm tra password của username này. Then it goes and does its own version of generate\_md5\_key(nonce, username, realm, URI, password\_I\_have\_for\_this\_user\_in\_my\_db)
  + It compares the output of generate\_md5() that it got with the one the client sent, if they match the client sent the correct password. If they don't match the password sent was wrong.
* Ví dụ:
  + Client request (no authentication)

|  |
| --- |
| GET **/dir/index.html** **HTTP**/1.0  Host: localhost |

* + Server response

|  |
| --- |
| **HTTP**/1.0 401 **Unauthorized**  Server: HTTPd/0.9  Date: ​Sun, 10 Apr 2014 20:26:47 GMT  WWW-Authenticate: ​Digest realm="testrealm@host.com",  ​qop="auth,auth-int",  ​nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",  ​opaque="5ccc069c403ebaf9f0171e9517f40e41"  Content-Type: ​text/html  Content-Length: ​153  <!DOCTYPE htm**l>**  **<**html  **​<**head>  **​<**meta charset="UTF-8" />  **​<t**itle>Err**or</t**itle  **​</**head  **​<**body>  ​<h1>401 Unauthorize**d.**</h1  **​</**body**>**  **</html>** |

* Gửi lại phương thức authen: digest, giá trị của biến realm, giá trị của nonce
  + Client request (username "Mufasa", password "Circle Of Life")

|  |
| --- |
| GET **/dir/index.html** **HTTP**/1.0  Host: localhost  Authorization: Digest username="Mufasa",  realm="testrealm@host.com",  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",  uri="/dir/index.html",  qop=auth,  nc=00000001,  cnonce="0a4f113b",  response="6629fae49393a05397450978507c4ef1",  opaque="5ccc069c403ebaf9f0171e9517f40e41" |

* The "response" value is calculated in three steps, as follows. Where values are combined, they are delimited by colons.
  + The MD5 hash of the combined username, authentication realm and password is calculated. The result is referred to as HA1.
  + The MD5 hash of the combined method and digest URI is calculated, e.g. of "GET" and "/dir/index.html". The result is referred to as HA2.
  + The MD5 hash of the combined HA1 result, server nonce (nonce), request counter (nc), client nonce (cnonce), quality of protection code (qop) and HA2 result is calculated. The result is the "response" value provided by the client.

HA1 = MD5( "Mufasa:testrealm@host.com:Circle Of Life" )

= 939e7578ed9e3c518a452acee763bce9

HA2 = MD5( "GET:/dir/index.html" )

= 39aff3a2bab6126f332b942af96d3366

Response = MD5( "939e7578ed9e3c518a452acee763bce9:\

dcd98b7102dd2f0e8b11d0f600bfb0c093:\

00000001:0a4f113b:auth:\

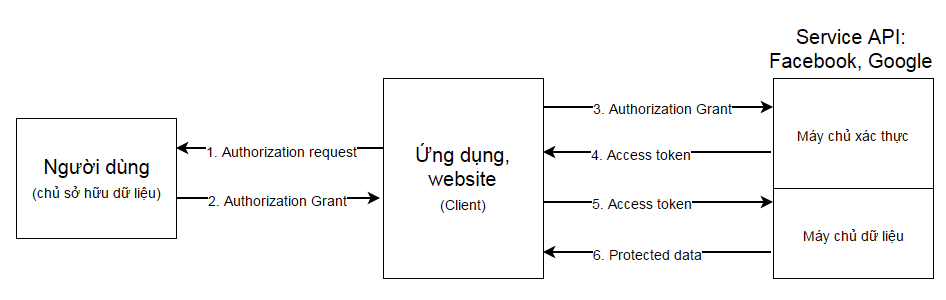
39aff3a2bab6126f332b942af96d3366" )

= 6629fae49393a05397450978507c4ef1

* + Tại thời điểm này, máy khách có thể đưa ra một yêu cầu khác, sử dụng lại giá trị nonce của máy chủ (máy chủ chỉ đưa ra một nonce mới cho mỗi phản hồi "401") nhưng cung cấp một nonce máy khách mới (cnonce). Đối với các yêu cầu tiếp theo, bộ đếm yêu cầu hệ thập lục phân (nc) phải lớn hơn giá trị cuối cùng mà nó đã sử dụng - nếu không kẻ tấn công có thể chỉ cần "phát lại" một yêu cầu cũ với cùng thông tin xác thực. Máy chủ đảm bảo rằng bộ đếm tăng lên đối với từng giá trị nonce mà nó đã đưa ra, từ chối bất kỳ yêu cầu xấu nào một cách thích hợp. Rõ ràng việc thay đổi phương thức, URI và / hoặc giá trị bộ đếm sẽ dẫn đến một giá trị phản hồi khác.
  + Máy chủ nên nhớ các giá trị nonce mà nó đã tạo gần đây. Nó cũng có thể nhớ khi mỗi giá trị nonce được phát hành, chúng sẽ hết hạn sau một khoảng thời gian nhất định. Nếu một giá trị đã hết hạn được sử dụng, máy chủ sẽ phản hồi với mã trạng thái "401" và thêm stale = TRUE vào tiêu đề xác thực, cho biết rằng máy khách nên gửi lại với nonce mới được cung cấp mà không cần nhắc người dùng nhập tên người dùng khác và mật khẩu.
  + Máy chủ không cần giữ bất kỳ giá trị nonce nào đã hết hạn - nó có thể đơn giản giả định rằng mọi giá trị không được nhận dạng đã hết hạn. Máy chủ cũng có thể chỉ cho phép mỗi giá trị nonce được trả về một lần, mặc dù điều này buộc máy khách phải lặp lại mọi yêu cầu. Lưu ý rằng máy chủ hết hạn ngay lập tức sẽ không hoạt động, vì máy khách sẽ không bao giờ có cơ hội sử dụng nó.
* Ưu điểm:
  + Không có username hoặc password được gửi đến máy chủ trong bản rõ, làm cho kết nối không phải SSL an toàn hơn yêu cầu HTTP Basic không được gửi qua SSL. Điều này có nghĩa là SSL không bắt buộc, điều này làm cho mỗi request nhanh hơn một chút.
* Nhược điểm:
  + Yêu cầu server lưu cleartext.
  1. **Oauth2**
* Oauth2 là viết tắt của Open với Authentication hoặc Authorization. OAuth ra đời nhằm giải quyết vấn đề trên và xa hơn nữa, đây là một phương thức chứng thực giúp các ứng dụng có thể chia sẻ tài nguyên với nhau mà không cần chia sẻ thông tin username và password.
* Ứng dụng:
  + Authentication: xác thực người dùng.
  + Authorization: người dùng ủy quyền cho ứng dụng truy cập tài nguyên của họ.

Nếu một website (hay phần mềm) nào đó cho phép đăng nhập bằng tài khoản Facebook hay Google mà không cần tạo nick mới thì đó chính là OAuth.

* Cách hoạt động:
  + Khi bạn đăng nhập bằng Facebook hay Gmail, website sẽ dẫn bạn đến trang (hoặc phần mềm) Facebook và liệt kê những quyền mà nó cần phải có để cho phép bạn đăng nhập và sử dụng dịch vụ.
  + Nếu bạn đồng ý thì lúc này Facebook sẽ phát cho website một cái token Token này chứa một số quyền hạn nhất định giúp cho website có thể xác minh bạn là ai cũng như giúp cho website có thể hoạt động được.
  + Nếu website này bị hacker tấn công thì nó chỉ lấy được thông tin hay hoạt động của bạn trên website đó mà không ảnh hưởng đến những website khác mà bạn đang sử dụng.
* Sơ đồ luồng hoạt động của OAuth2:



* + Ứng dụng (website hoặc mobile app) yêu cầu ủy quyền để truy cập vào Resource Server (Gmail,Facebook, Twitter hay Github…) thông qua User
  + Nếu User ủy quyền cho yêu cầu trên, Ứng dụng sẽ nhận được ủy quyền từ phía User (dưới dạng một token string)
  + Ứng dụng gửi thông tin định danh (ID) của mình kèm theo ủy quyền của User tới Authorization Server
  + Nếu thông tin định danh được xác thực và ủy quyền hợp lệ, Authorization Server sẽ trả về cho Ứng dụng access\_token. Đến đây quá trình ủy quyền hoàn tất.
  + Để truy cập vào tài nguyên (resource) từ Resource Server và lấy thông tin, Ứng dụng sẽ phải đưa ra access\_token để xác thực.
  + Nếu access\_token hợp lệ, Resource Server sẽ trả về dữ liệu của tài nguyên đã được yêu cầu cho Ứng dụng.
* Lỗ hổng của Oauth2:
  + Vulnerabilities in the client application
    - Improper implementation of the implicit grant type
    - Flawed CSRF protection
  + Vulnerabilities in the OAuth service
    - Leaking authorization codes and access tokens
    - Flawed scope validation
    - Unverified user registration
    1. **Improper implementation of the implicit grant type (Áp dụng sai implicit flow)**
* Trong trường hợp này, access token được gửi từ Oauth service đến client app thông qua trình duyệt người dùng bằng URL fragment. Sau đó, client app lấy access token bằng JavaScript. Vấn đề là, nếu client app muốn duy trì phiên sau khi người dùng đóng trang, nó cần phải lưu trữ current data (thường là UserID và access token).
* Để giải quyết vấn đề này, client app thường sẽ gửi dữ liệu này đến máy chủ trong 1 POST request và gán cho người dùng 1 phiên trong cookie.
* Attacker có thể tác động tới POST request này, dẫn đến một lỗ hổng nếu client app không kiểm tra chính xác accessken token có trùng khớp với dữ liệu khác trong request hay không. Attacker có thể thay đổi tham số gửi đến server để mạo danh bất kì user nào.
  + 1. **Flawed CSRF protection (Thiếu việc bảo vệ tấn công CSRF)**
* Tấn công Cross-Site Request Forgery là kiểu tấn công mà kẻ tấn công lừa người sử dụng truy cập tới một trang web độc hại (hacker có thể kiểm soát), thông qua trang đó kẻ tấn công có thể tấn công một trang khác (Cross-site) mà người dùng đã xác minh (thông thường là đã có cookie). Để chống lại kiểu tấn công này, thông thường là các request quan trọng sẽ có thêm một csrf token để ngăn các tấn công cross site.
* Trong triển khai OAuth, tham số state được tính là optional tuy nhiên lại rất quan trọng khi nó được sử dụng như một csrf token để bảo vệ trang web. Việc thiếu hoặc sử dụng tham số này sai cách sẽ cho phép kẻ tấn công sử dụng CSRF tấn công ứng dụng web.
  + 1. **Leaking authorization codes and access tokens**
* Tùy thuộc vào cách cấp quyền, mã hoặc token được gửi qua trình duyệt nạn nhân đến /callback endpoint trong redirect\_uri parameter của authorization request.
* Tham số redirect\_uri là tham số quan trọng để server có thể sử dụng để gửi trả về authorization code. Việc không xử lí tham số này có thể cho phép kẻ tấn công lợi dụng để đánh cắp authorization code để truy cập trái phép tới tài khoản người dùng.
  + 1. Flawed scope validation
* x
  + 1. Unverified user registration
* X

1. **Access Control**
   1. **Acces Control là gì**?

* Access control (or authorization) là việc áp dụng các ràng buộc ai có thể làm được việc gì? Trong ngữ cảnh các ứng dụng web, access control phụ thuộc vào xác thực và quản lý phiên.
  + Authentication định danh người dùng và xác nhận họ là chính họ.
  + Session management xác định các HTTP request tiếp theo nào đang được thực hiện bởi chính người dùng đó.
  + Access control xác định liệu người dùng có được phép thực hiện hành động họ đang định thực hiện hay không.
* Broken access controls là một lỗ hổng bảo mật thường gặp và nghiêm trọng. Thiết kế access control phải được thực hiện bởi con người, do đó khả năng xảy ra sai sót là rất lớn.
  1. **Các loại Access Control**
* **Vertical Access Control**: là cơ chế quản lí truy cập tới các chức năng (functions) quan trọng trong hệ thống. Những nhóm người dùng khác nhau có quyền sử dụng những chức năng khác nhau trong hệ thống.
  + Unprotected functionality: Xảy ra khi giao diện ứng dụng hiển thị khác nhau cho từng đối tượng khác nhau khiến người lập trình chủ quan cho rằng các chức năng đặc biệt không thể bị sử dụng bởi user thông thường theo các đường link “chính thống” trên giao diện và không thực hiện kiểm tra phân quyền khi có yêu cầu truy cập tới các chức năng quan trọng.
  + Parameter-based access control methods: Một vài trang web xác minh vai trò của người dùng và lưu trữ vào các biến mà người dùng có thể thay đổi được.
  + Broken access control resulting from platform misconfiguration: Trong một vài trường hợp, tùy vào vai trò của user, ứng dụng sẽ ngăn chặn việc truy cập của user đó vào một số đường link nhất định. Thay đổi phương thức gửi request hoặc sử dụng các headers cho phép ghi đè URL gốc của request như X-Original-URL và X-Rewrite-URL có giúp user truy cập vào đường link bị cấm.
* **Horizontal Access Controls**: Cơ chế ngăn chặn truy cập vào tài nguyên không được cho phép. VD: Tài khoản ngân hàng thì mình chỉ được coi số dư của mình thôi, k được coi của người khác.
  + Horizontal privilege escalation: một số trang web hiện thực phân quyền thông qua các biến dễ bị thay đổi bởi chính người dùng => dễ dàng truy cập vào tài nguyên không được phép.
  + Context-dependent access controls: Quản lí truy cập dựa vào trạng thái và các tương tác của user với website => Hạn chế việc user thực hiện các thao tác sai thứ tự. VD: Mua hàng online thì phải bỏ vào giỏ hàng, xác minh số lượng rồi thanh toán (k cho thanh toán rồi thay đổi số lượng hàng hóa ).
  1. **Lỗi IDOR**
* IDOR (INSECURE DIRECT OBJECT REFERENCE) là một lỗ hổng bảo mật mà trong đó người dùng có thể truy cập và thay đổi dữ liệu của bất kỳ người dùng nào khác có trong hệ thống.
* Để kiểm tra lỗi IDOR trên 1 website, ta sử dụng 2 tài khoản khác nhau, đăng nhập 2 tài khoản trên 2 trình duyệt khác nhau. Rồi sử dụng các chức năng của tài khoản này, thử thay đổi các param truyền vào nhưng lại ứng với tài khoản kia xem có được không.