Security day 2

Encryption, auth, autz and oauth

Encryption basic

What is encryption?

- It is the process of encoding a message or information in such a way that only the allowed party can access it.

It is reversible

Encryption basic

Symmetric key / Private key/ a password

- The same key is used both to encrypted message and to decrypt it again.
- The security is based on the “password”

Encryption basic

Public key

- There is a public key that can be used to encrypt the message.
- There is a private key that can be used to decrypt the message

Common uses of encryption

SSH/sftp

HTTPS

Encrypting emails

Hashing

A hash function is a function that can create a “sum” of that data

The md5 sum for hello world is:

278bf123abaea7b8d661d337e8112dfc

Sha1 for the same is

ce9b71ef7a2eb18be6d24dafb0845cdc1623d1d4

Sha256

03457c36995bc52df8fa0430393535842195c72ffff555a9febdf2098c959ce8

Common uses for hashing

Check if a downloaded file is correct

Store sensitive information like passwords and social security numbers

Creating unique keys for data and create fast lookup in key/value databases

Authentication

It is confirming the identity of a client

- Username/password
- Public / private key eg SSH
- Fingerprint/facial

Authentication

Single factor

- Username/Password
- SSH with public/private key
- SSH with password

Authentication

Multifactor / 2 factor

- Something that the user has eg a bankcard/nemid
- Something that the user knows, a pin code, a password

A lot of services supports 2 factor

- Google
- Facebook
- Github
- Twitter

Authorization

Authorization is giving the client access rights to specific resources related to that client.

- Files
- Resources
- Pages
- Hardware

Authorization vs Authentication

Authentication makes sure that the user is the correct user (login password)

Authorization gives the user the privileges that the user should have according to the system

Different types of services

Ldap

Wayf

Oauth

Attacks against password

- Bruteforce
- Attacking hashes

Access control example

The Linux file system

Session management

- Make sure that your sessionID is unique and random on every login
- If someone can guess or steal the sessionID it is possible to impersonate that person on your site(eg using MIT)
- Make sure that a session times out

Session management

Php sends a unique id for the session

PHPSESSIONID cookie is sent to the user

A file is created on the server with the session information

Session management

- session_start()
- $_SESSION
- session_destroy();

Exploits/security problems

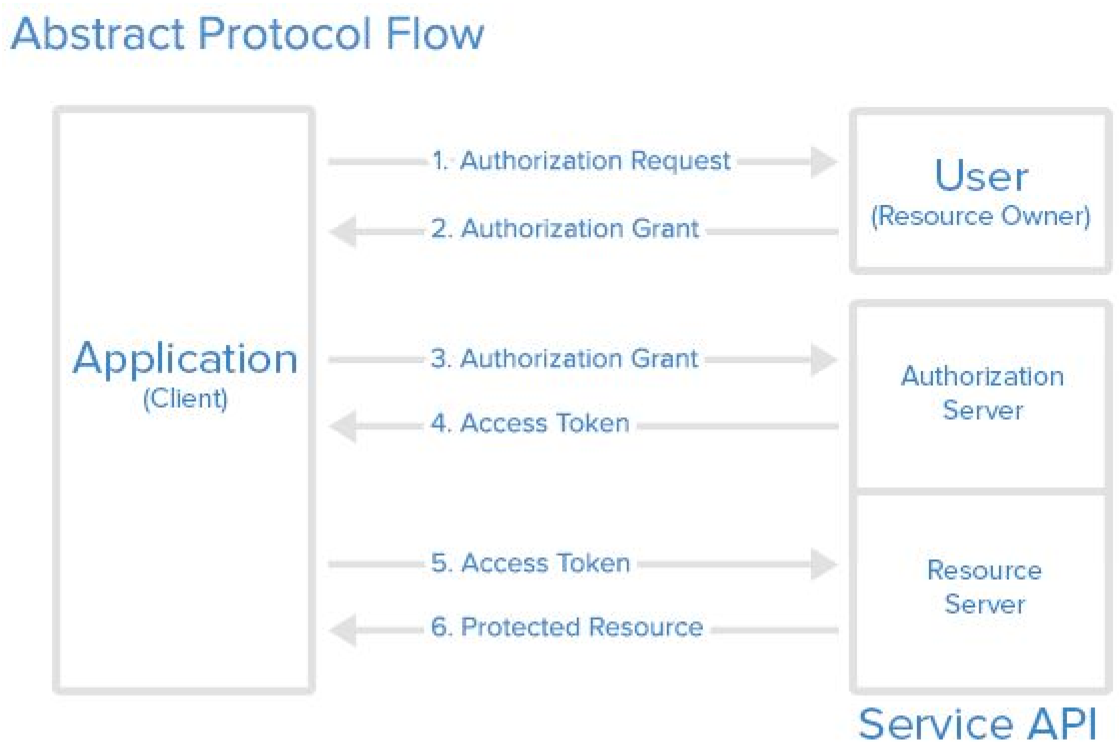
- Forceful browsing
- Problems with input validating in cookies, input fields, parameters
- Manipulating the http header
-

Oauth

OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, GitHub, and DigitalOcean. It works by delegating user authentication to the service that hosts the user account, and authorizing third-party applications to access the user account. OAuth 2 provides authorization flows for web and desktop applications, and mobile devices.

Source: digitalocean

Oauth


Abstract Protocol Flow
Application
(Client)
1. Authorization Request
2. Authorization Grant
3. Authorization Grant
4. Access Token
5. Access Token
6. Protected Resource
User
(Resource Owner)
Authorization Server
Resource Server
Service API

Oauth

1. The application requests authorization to access service resources from the user.
2. If the user authorized the request, the application receives an authorization grant.
3. The application requests an access token from the authorization server (API) by presenting authentication of its own identity, and the authorization grant
4. If the application identity is authenticated and the authorization grant is valid, the authorization server (API) issues an access token to the application. Authorization is complete.
5. The application requests the resource from the resource server (API) and presents the access token for authentication
6. If the access token is valid, the resource server (API) serves the resource to the application

Limits of Oauth

Only works on http(s) traffic

Google Oauth

Google's OAuth 2.0 APIs can be used for both **authentication and authorization.**

Task 1, hashing

Try creating a small php program that takes a password and compare it to a hashed value, stored in a database or as an predefined var.

Use password_hash() and password_verify() to create the program, try to echo the password_hash result and run your program a couple of times, the hashed result should change

Ask yourself why does the hash of the password that is getting hashed using password_hash change every time you run it?

How is it possible to verify a pass

Task 2, Session

Create a form with login and password (login.html).

Create login.php that handles the login

Create a form with input fields to register a user's password and username

Create register.php that handles the creation of a user

Create secret.php, this is the page that is password protected.

Task 2 session

Flow:

a user is trying to get a access to protected.php and gets an error

The user creates a user at register.html that uses register.php to create a user

- The username is saved to the db

- The password is saved to the db

The user then logins using login.html

 - The password and username is validated agains the one saved in the datase.

 - A session i created

After the login the user is redirected to protected.php

- protected.php checks if there is a valid session

Task 2 session

remember to hash passwords

remember to validate input

Task 3, Oauth

Create a small program that authenticate against google.com

Use the following guide:

https://developers.google.com/+/web/samples/php