Web Programming
Login

Robert Ewald | University of Stavanger

Login

- This lecture: Simple login using sessions.
  - Has some security flaws
- Deployment alternatives:
  - Flask-Login
  - OAuth provider, e.g. firebase.google.com

Password Hashes

```
from werkzeug.security import generate_password_hash, check_password_hash
```

- Create a salted password hash to store

```
hash = generate_password_hash("Joe123")
```

```
"pbkdf2:sha256:150000$oMxlb0Oa$125a8c19b39e0fc7e903e7775a45e40667663ed01382f9b5adcb5e0eb3d80937"
```

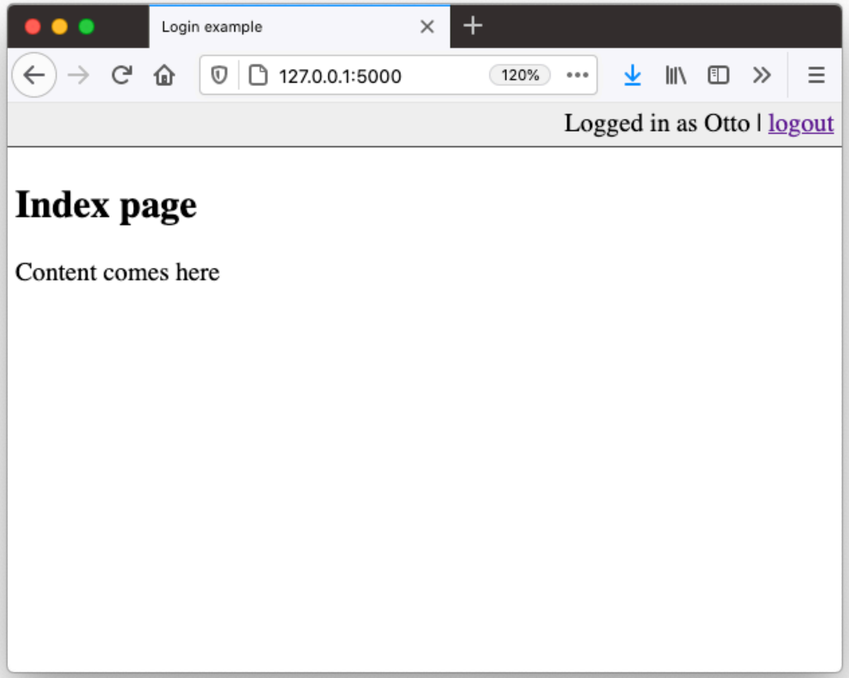
Includes a random **salt**, so no two passwords have the same hash

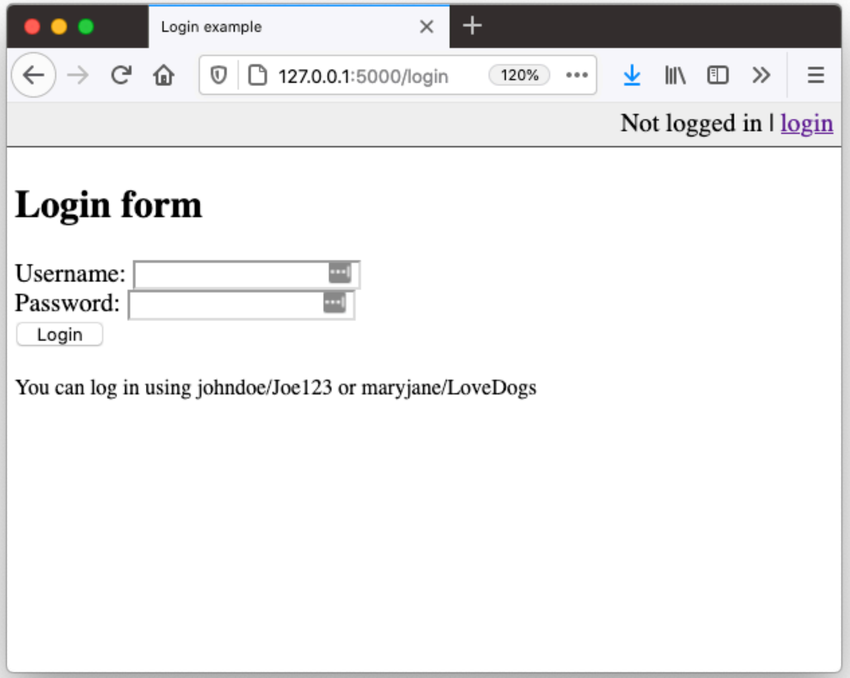
- Check password

```
ok = check_password_hash(hash,”Joe123")
```

Example

examples/python/flask/9_login/app.py





Hashing algorithms complexity

KDF	6 letters	8 letters	8 chars	10 chars	40-char text	80-char text
DES CRYPT	< $1	< $1	< $1	< $1	< $1	< $1
MD5	< $1	< $1	< $1	$1.1k	$1	$1.5T
MD5 CRYPT	< $1	< $1	$130	$1.1M	$1.4k	1.5×10^{15}
PBKDF2 (100 ms)	< $1	< $1	$18k	$160M	$200k	2.2×10^{17}
bcrypt (95 ms)	< $1	$4	$130k	$1.2B	$1.5M	$48B
scrypt (64 ms)	< $1	$150	$4.8M	$43B	$52M	6×10^{19}
PBKDF2 (5.0 s)	< $1	$29	$920k	$8.3B	$10M	11×10^{18}
bcrypt (3.0 s)	< $1	$130	$4.3M	$39B	$47M	$1.5T
scrypt (3.8 s)	$900	$610k	$19B	$175T	$210B	2.3×10^{23}

http://www.tarsnap.com/scrypt/scrypt.pdf

Example

examples/python/flask/9_login/app.py

- on login, check password hash and add username to session

```
@app.route("/login", methods=["GET", "POST"])
def login():
    username = request.form["username"]
    password = request.form["password"]

    if valid_login(username,password):
        session["username"] = username
        return redirect(url_for("index"))
```

Example

examples/python/flask/9_login/app.py

- on logout, remove username from session

```
@app.route("/logout")
def logout():
    session.pop("username")
    return redirect(url_for("index"))
```

Exercise #1, #2, #3

github.com/dat310-2022/info/tree/main/
exercises/python/flask5

Walkthrough in lecture video!

Limitation

- To further improve security session should include:
  - Unique token for every time you login
- Further, requests should contain CSRF token.
  - https://owasp.org/www-community/attacks/csrf
  - https://portswigger.net/web-security/csrf

Cross-site request forgery (CSRF)

- A web security flaw
- Attacker induces user to perform actions she didn’t intend

Preconditions for CSRF

- Relevant action (update password, order goods, etc. caused by a request)
- Cookie-based session handling
- All request parameters are predictable

Example HTTP request

```
POST /update-email HTTP/1.1
Host: flawed.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: session=iaeuHeklkeokEIkeji

email=my-new-email@example.com
```

Attackers website

```
<html>
   <body>
      <form action="https://flawed.example.com/update-email" method="POST">
         <input type="hidden"
                name="email"
                value="attacker@evil.example.com">
      </form>
      <script>
         document.forms[0].submit();
      </script>
   </body>
<html>
```

Defend against CSRF

- Include a CRSF token in the app
- The Flask-WTF extension provides mechanisms for that https://github.com/wtforms/flask-wtf/

- Use the SameSite:strict cookie attribute so that cookies are not sent with requests https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite