

Security

i web programming

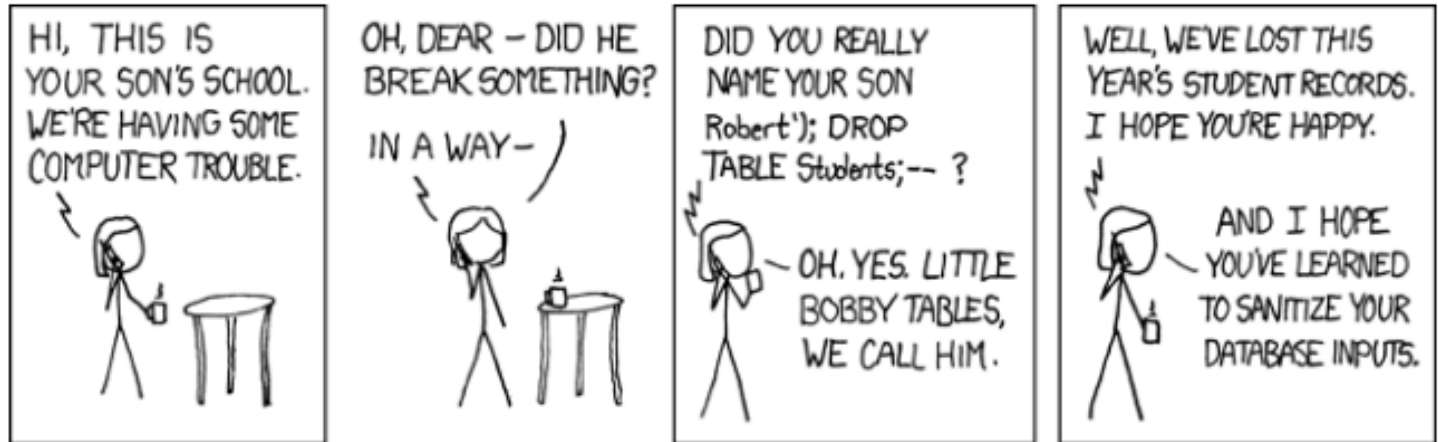
Referencer:

<http://martinfowler.com/articles/web-security-basics.html>

Mål

- Kende til trusler og forholdsregler
- Kunne anvende forholdsregler i egen software udvikling

Emner:



1. Form input validering
2. Placeholders til database input
3. Man in the middle — (encrypt data transfer: HTTPS)
4. Hash and Salt Passwords
5. Authentication -> Authorization

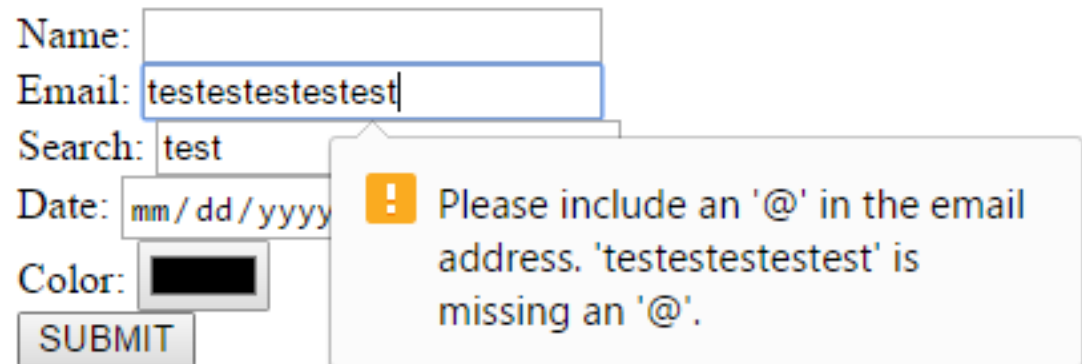


Form validating - client

- HTML5

- email
- url
- tel
- number
- range
- date
- month
- week
- time
- datetime
- color

```
<form action="#" method="post">  
  Name:   <input type="text" name="name" />  
  Email:  <input type="email" name="email" />  
  Search: <input type="search" name="search"/>  
  <input type="datetime" name="entry-day-time"/>  
  <input type="color" name="color" />  
  <input type="submit" value="SUBMIT">  
</form>
```



The screenshot shows a web form with the following fields and values:

- Name: (empty text input)
- Email: testtesttesttest (email input, highlighted with a blue border)
- Search: test (search input)
- Date: mm/dd/yyyy (datetime input)
- Color: (color input, showing a black swatch)
- SUBMIT (submit button)

A validation error message is displayed in a white box with a yellow exclamation mark icon:

! Please include an '@' in the email address. 'testtesttesttest' is missing an '@'.

Form validation - client

```
var form = document.getElementById("form");
form.onsubmit = function() {
    var email = document.getElementById("email").value;
    if(validateEmail(email))
        return true;
    else return false;
};

function validateEmail(email) {
    var re = /^(([^<>()\\[\]\\. ,;: \s@"]+)(\.[^<>()\\[\]\\. ,;: \s@"]+))*$/;
    return re.test(email);
}
```

- Regular expressions



```
private static boolean emailIsValid(String email) {  
    Pattern ptr = Pattern.compile("(?:(?:\\r\\n)?[ \\t])*?(?:[(?![^()<>@"]
```

PreparedStatement

- Modgå sql-injection
- Adskille indhold fra kommando

```
private void insertImg(String pathname, Kayak kayak){
    String sql = "UPDATE kayak SET image = ? WHERE id = ?";
    try {
        PreparedStatement pstmt = DB.getConnection().prepareStatement(sql);
        InputStream in = new FileInputStream(pathname);
        pstmt.setBlob(1, in);
        pstmt.setInt(2, kayak.getId());
        if(pstmt.executeUpdate() == 0)
            throw new IOException("Image was not persisted in db");
        in.close();
    }
```



HTTPS

- Kryptere trafik fra klient til server
- Beskyt mod 'man in the middle-attack'
 - Sniffing af passwords
 - Session hijacking (stjæle sessions-id)
 - Forhindre anden læsning af data
- Bruge HTTP over TLS

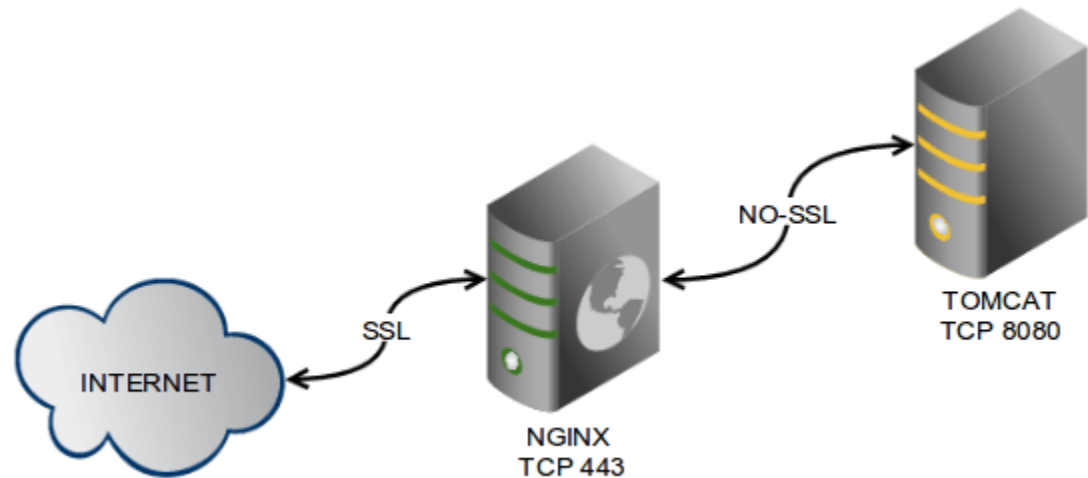
HTTPS





Java web application

- Host som Digital Ocean
- NGINX webserver med gratis certifikat
- Tomcat bagved modtager ukrypteret trafik



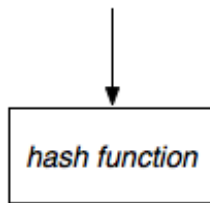
Hashing og Salting

Reference: <https://crackstation.net/hashing-security.htm>

- Passwords skal aldrig ligge i db i klartekst
- Først salting:
 - Undgå rainbow table opslag
 - Tilføj randomiseret text til password før hash
 - Gem random salt med pwd i db tabel
- Så hashing:
 - One way function
 - Fixed length fingerprint
 - En bestemt text fører altid til samme hash

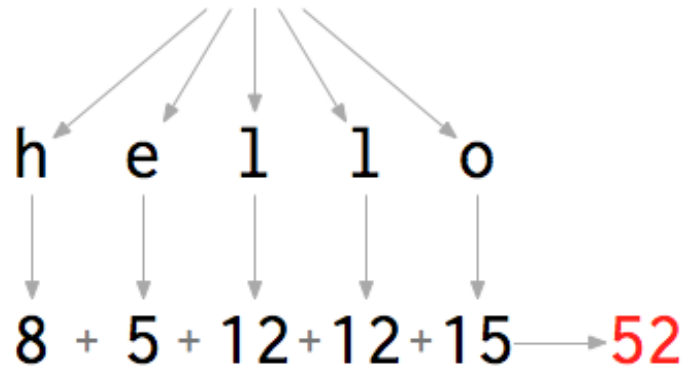
Hashing:

hello



52

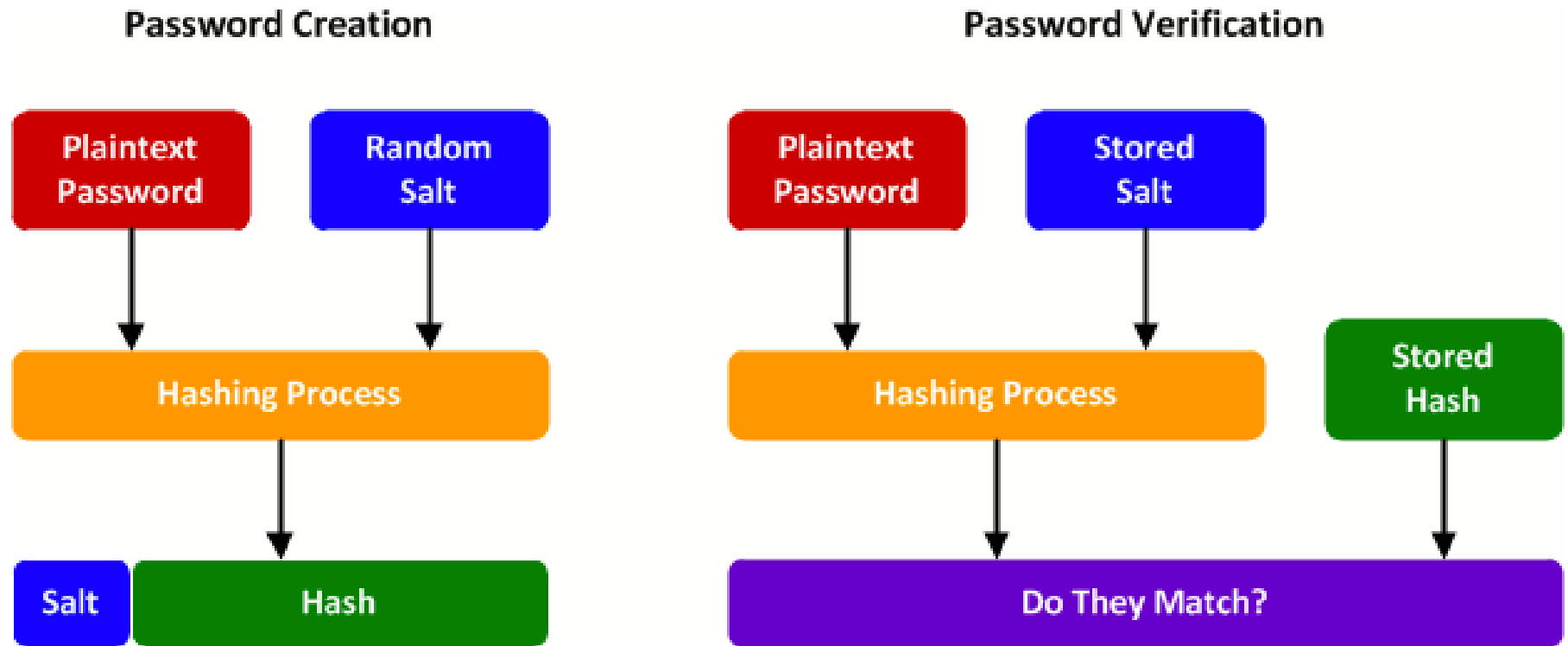
hello



52

id	username	password	passwordHint
1	admin	7E7274BAC45E467C5AB832170F12E418	k3wl dud
2	pumpkin22	5377DBF76D995CC213ED76924A31CB13	my favorite holiday
3	johndoe	512239D9AE0C3B5567DE188739F689F2	Freddie Mercury's band
4	alexa45	2FE5421E49061F8225C2FB7CB81980FD	password
5	guy	ABE35E2827DDA834C9612FE9E9C92CE0	NULL
6	maryjane	198670893B2781C83F3DA5D45150123D	I'm one!
7	dudson123	59E2113217E65B9885F9DA73FDC5697B	scary movie!

Salting



Gem hashed password sammen med den random salt for at kunne verificere

Demo: Hashing og Salting

Lav hashing og salting af password i databasen:

Hent filen: [PasswordStorage.java](#)



PasswordStorage.java

- Gemmer salt i databasen i formen:
- Password er concateneret af:
 - Sha1 +
 - Iterations + (64000 encryptions: computational extra work)
 - Hashsize +
 - base64 encoded salt +
 - base64 encoded hash

sha1:64000:18:fHXOSM9KX80+JH/0BHe3QZmO0QRsIIW9:2CqWJ7d4Vy45NhhRD82/Kfww