UNIVERSITÀ DELLA CALABRIA
il Campus per eccellenza

Why design matters for security

Mario Alviano

Exercise

We have to develop a booking system for a theater with 1000 seats grouped in 25 rows of 40 armchairs each. Every armchair is identified by a letter and a number. For a person, we are interested in fiscal code and name.

Think to how you usually approach to coding.

Can you implement such a system?

Security is often a design issue

- Any issue here?
- Can PIN be shielded?
- Can you guess the PIN?

Common scenario (1)

- You have to complete a software project
- You have a team of developers, testers and domain experts
- Important features identified with stakeholders
  - performance, security, maintainability, and usability
- Priority goes to business logic to reduce release time and costs
  - Users start to use whatever is released
- Security on a second phase
  - Nobody thanks for security because it is transparent
  - Anyhow you opt for using some security library

Common scenario (2)

The software is ready to be released (to go in production)

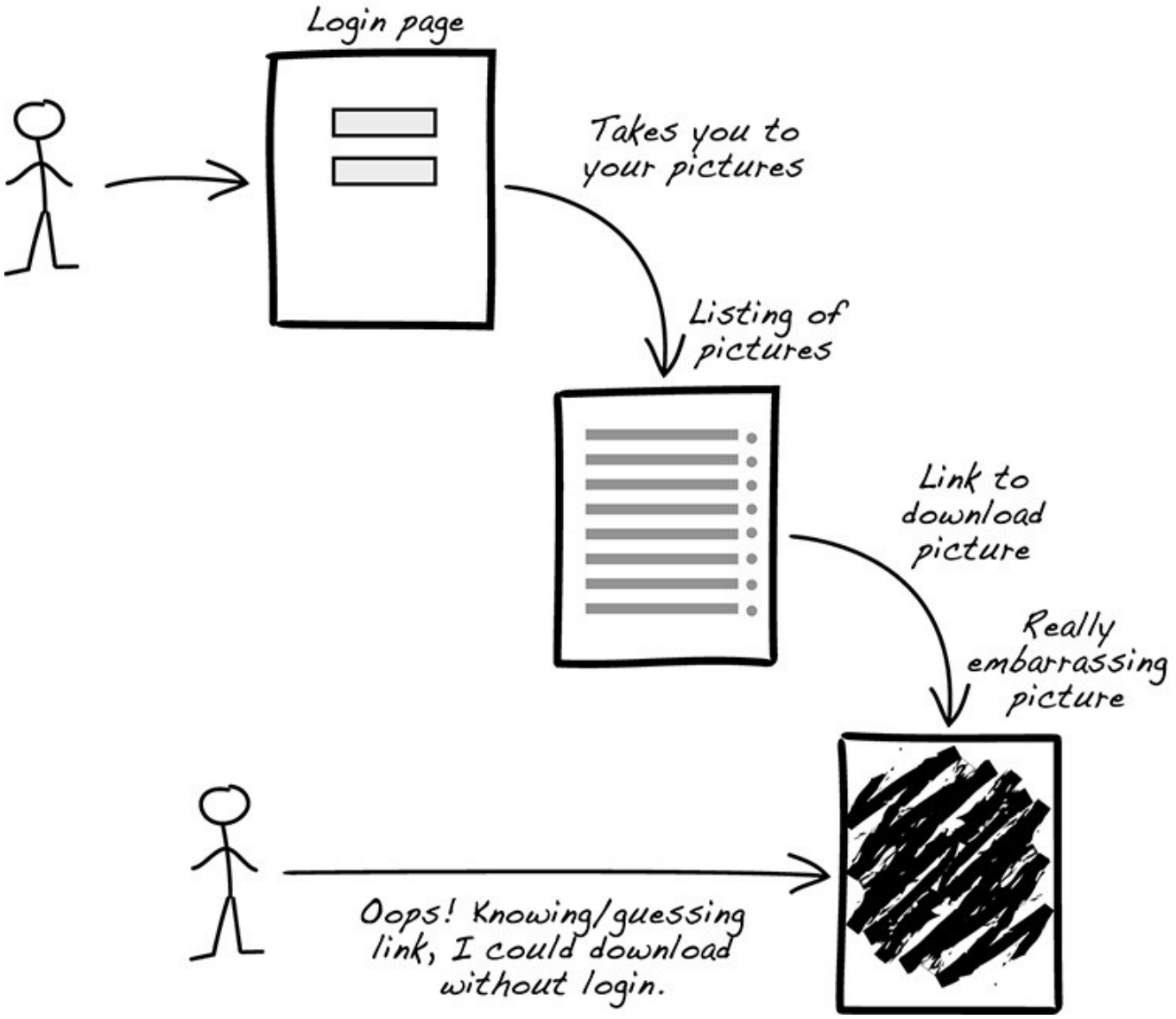
1) Time for a security check and penetration testing
   - Many vulnerability are discovered
   - The release is delayed
   - Extreme case: fixing the vulnerabilities requires to completely rewrite the software
2) Skip any check, just go in production
   - Users start to use your software
   - You are on all newspaper because your software was hacked and sensible data were stolen
   - Goodbye users (and welcome Europen Union lawyers)

A concern, not a feature

- Security is something that must make us anxious
- However, often it is described as a set of features
  - Home alarm: sensors, sirens, calls and SMS
  - Is it sufficient to “keep thieves out”?
  - How is the alarm activated and deactivated?
  - Do I always activate it before leaving home?
  - Do I leave a remote control somewhere were thieves can find it?
  - Is is easy to tamper sensors and sirens?

Security features and concerns (1)

- The problem with seeing security as features is that the focus is on “what the system does”
- Example, website with authentication to store pictures
  - “As a user, I want a login page to access my pictures.”
- Is it sufficient to implement a login page?
  - This is the required feature
  - But the user has also a concern about security, which is not satisfied!


Login page
Takes you to your pictures
Listing of pictures
Link to download picture
Really embarrassing picture
Oops! Knowing/guessing link, I could download without login.

Security features and concerns (2)

- The login page is useless if pictures can be accessed via direct links
- We want to provide access to pictures only to the owner
- ~~“As a user, I want a login page to access my pictures.”~~
- “As a user, I want access to my uploaded pictures to pass through a login page so that my pictures stay confidential.”
- “As a user, I want all access to my uploaded pictures to be protected by authentication so that my pictures stay confidential.”

Security features and concerns (3)

- The user concern is the confidentiality of their pictures
- Protecting **one path** to the pictures is not sufficient (the login page)
- We have to protect **all paths** to the pictures

Security concerns: CIA-T

- Confidentiality
  - Keep things secret that shouldn’t be made known to the public (eg. healthcare record)
- Integrity
  - The information doesn’t change or is only allowed to change in specific, authorized ways (eg. counting election results)
- Availability
  - Ensure access to information when needed (eg. being able to place an offer in an online auction before it expires)
- Traceability
  - Know who changed or accessed data
  - Required by GDPR for sensible data

Traditional approach to SSD

Worklist

- [x] Attack vectors
- [x] Zero day exploits
- [x] Web vulnerabilities
- [x] OWASP

Apparently not sufficient!

Example. User with id and name

```
public class User {
   private final Long id;
   private final String username;

   public User(final Long id, final String username) {
      this.id = id;
      this.username = username;    ①
   }

   // ...
}
```

Using String is too permissive. Possible XSS vulnerability:
my username is *<script>alert(42);</script>*

Example. Validation to avoid specific attacks

```
import static
com.example.xss.ValidationUtils.validateForXSS;
import static org.apache.commons.lang3.Validate.notNull;

public class User {
   private final Long id;
   private final String username;

   public User(final Long id, final String username) {
      notNull(id);     ①
      notNull(username);     ①

      this.id = notNull(id);
      this.username = validateForXSS(username);     ②
   }

   // ...
}
```

Issues

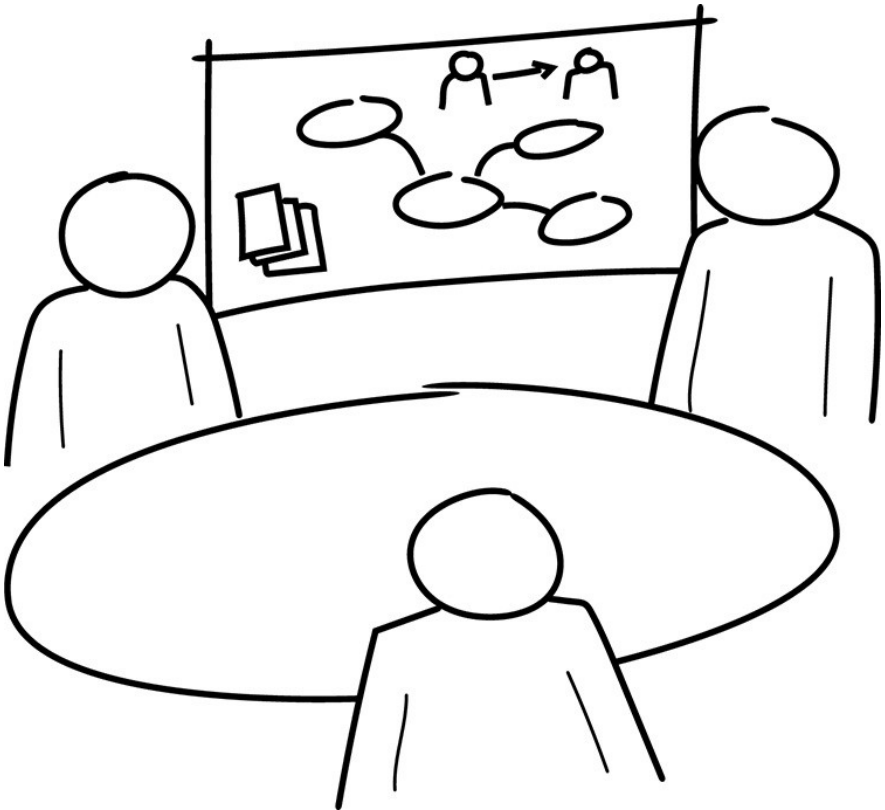
Not all developers are security experts

Focus is usally on business logic

In the future there will be new attacks

Example. Safe design

- What is a username for my application?
  - Contains only characters [A-Za-z0-9_-]
  - Contains at least 4 characters
  - Contains no more than 40 characters
- XXS is not possible anymore
  - Characters < and > are not allowed
  - We didn’t think to XXS
  - We just modeled the domain



```
import static org.apache.commons.lang3.Validate.*;

public class User {
   private static final int USERNAME_MINIMUM_LENGTH = 4;
   private static final int USERNAME_MAXIMUM_LENGTH = 40;
   private static final String USERNAME_VALID_CHARACTERS =
         "[A-Za-z0-9_-]+";

   private final Long id;
   private final String username;

   public User(final Long id, final String username) {
      notNull(id);
      notBlank(username);

      final String trimmed = username.trim();
      inclusiveBetween(USERNAME_MINIMUM_LENGTH,     ①
                       USERNAME_MAXIMUM_LENGTH,     ①
                       trimmed.length());     ①
      matchesPattern(trimmed,     ①
                     USERNAME_VALID_CHARACTERS,     ①
                     "Allowed characters are: %s",     ①
                     USERNAME_VALID_CHARACTERS);     ①

      this.id = id;
      this.username = trimmed;
   }

   // ...
}
```

Username is validated on creation

Is it the only point where the concept *username* is used?

Better to encapsulate all the knowledge about *username* in a *Username* class (we will call it a *domain primitive*)

```java
import static org.apache.commons.lang3.Validate.*;

public class Username {    ①
   private static final int MINIMUM_LENGTH = 4;
   private static final int MAXIMUM_LENGTH = 40;
   private static final String VALID_CHARACTERS = "[A-Za-
z0-9_-]+";

   private final String value;

   public Username(final String value) {
      notBlank(value);

      final String trimmed = value.trim();
      inclusiveBetween(MINIMUM_LENGTH,
                       MAXIMUM_LENGTH,
                       trimmed.length());
      matchesPattern(trimmed,
                     VALID_CHARACTERS,
                     "Allowed characters are: %s",
VALID_CHARACTERS);
      this.value = trimmed;
   }

   public String value() {
      return value;
   }
}
```

```java
public class User {
   private final Long id;
   private final Username username;    ②

   public User(final Long id, final Username username) {
      this.id = notNull(id);
```

Advantages of security by design

- Software is designed anyhow, so developers do not perceive extra work
- Business logic and security have the same priority
  - Differently, priority is on business logic
- Even non-experts developers write secure code
- Many security problems are implicitly fixed

Homework

- Write an application for the theater
- Prefer a terminal application, no need for a complex framework

Questions