CS304
Software Engineering

TAN, Shin Hwei

陈馨慧

Instructor: 胡春风

Southern University of Science and Technology

Reference: https://www.youtube.com/watch?v=T807GT0XCg4

Security Engineering

- Watch the video “Security risks and common mistakes in mobile application development” in lab folder (From: https://www.youtube.com/watch?v=T807GT0XCg4)
- Accept the invitation links at:
  - https://classroom.github.com/a/RwgDwl38

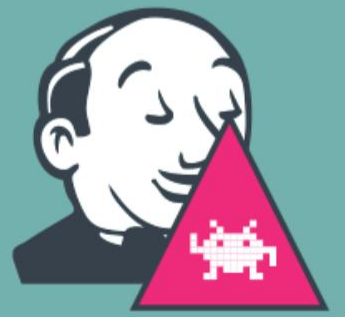
Security Engineering

Top 10 Java Vulnerabilities

1 JUnit

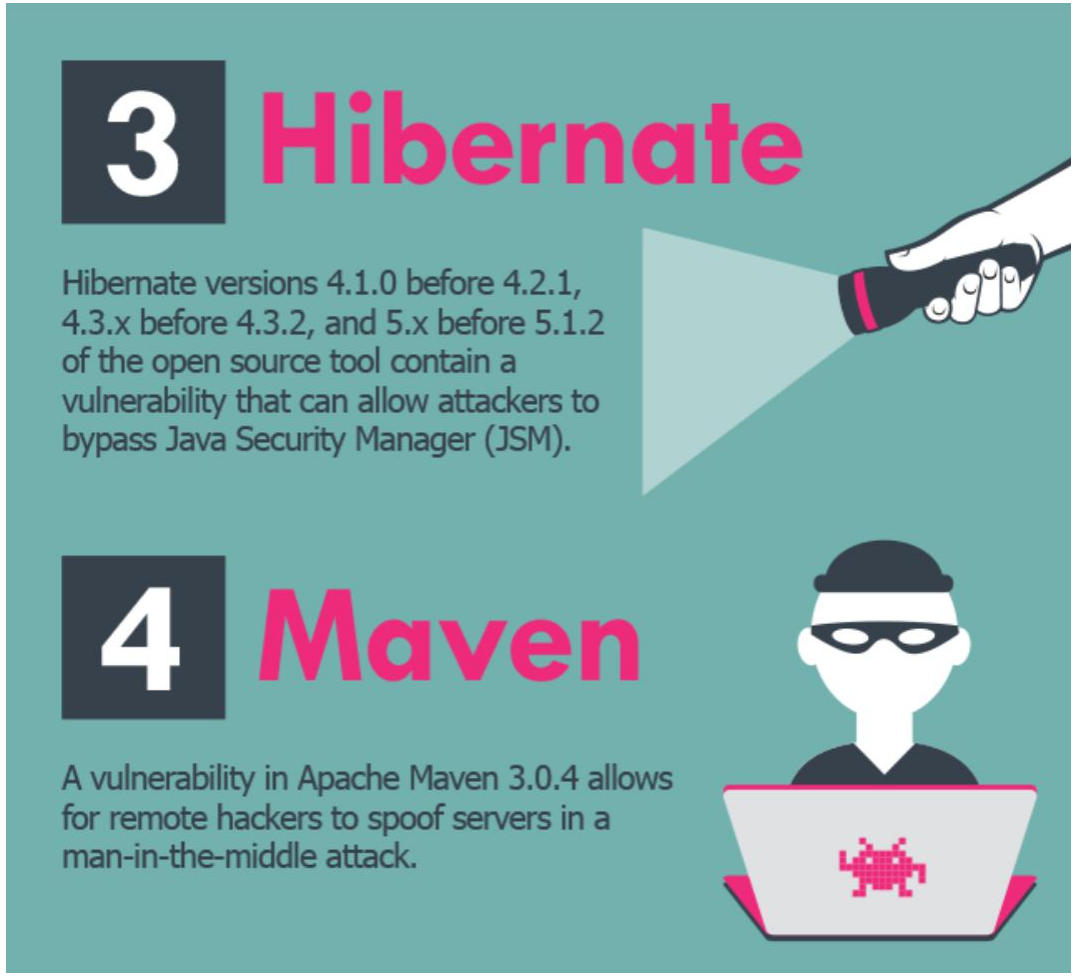
JUnit files that come with other applications can harbor vulnerabilities. For example, versions of the Google Web Toolkit (GWT) before 2.5.1 RC contain multiple cross-site scripting (XSS) vulnerabilities.

2 Jenkins

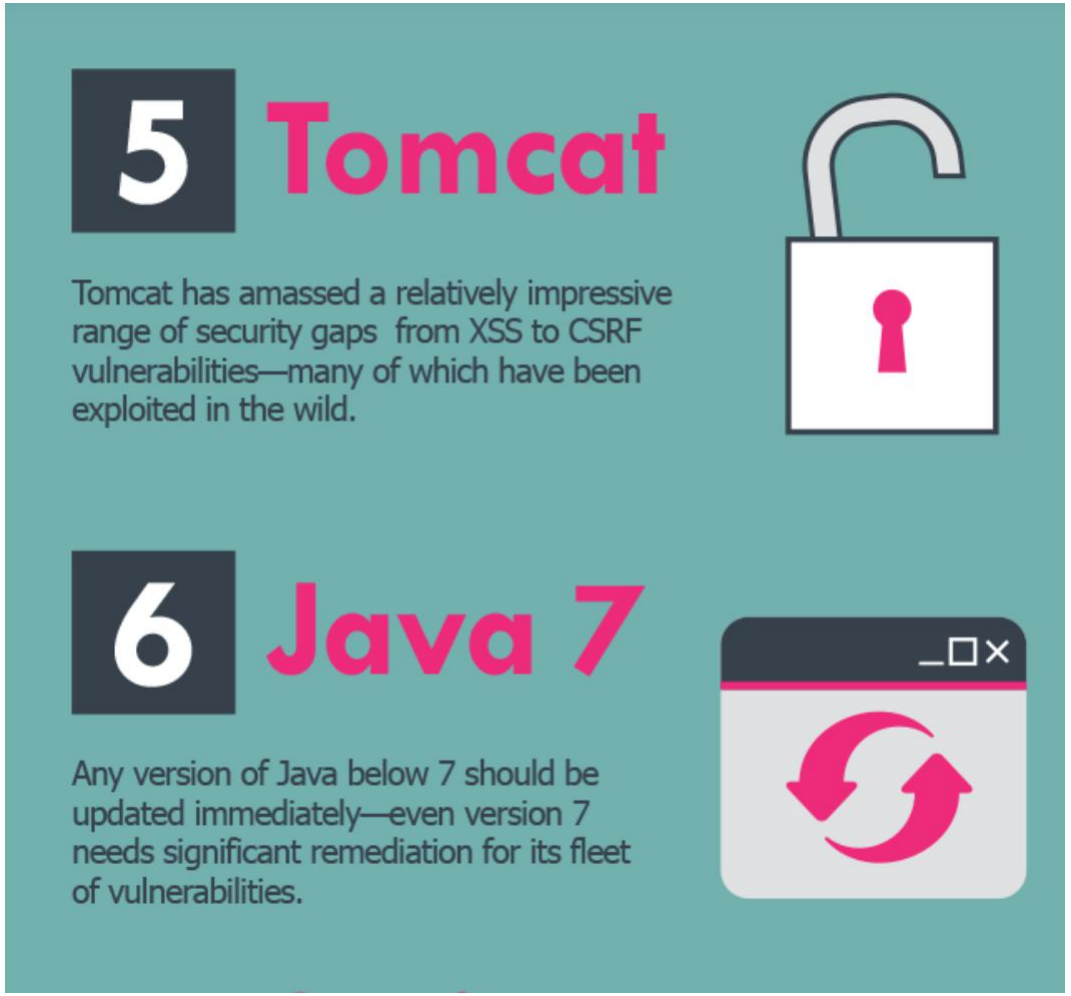


Popularity as continuous integration tool usually means more vulnerabilities and exploits—and in Jenkins' case, multiple XSS, cross-site request forgery (CSRF), and denial-of-service (Dos) vulnerabilities exist.

Top 10 Java Vulnerabilities



Top 10 Java Vulnerabilities



Top 10 Java Vulnerabilities

7 Spring Framework

As an open source project, Spring is not without its fair share of documented vulnerabilities.

8 JavaServer Faces

A vulnerability in Apache MyFaces Core 2.0.x before 2.0.12 and 2.1.x before 2.1.6 can give remote attackers the ability to read arbitrary files.

Top 10 Java Vulnerabilities

9 Eclipse IDE

Certain versions of Eclipse IDE help files are vulnerable to XSS-related exploits.

10 Vaadin

An XSS vulnerability in the Vaadin framework can allow remote attackers to inject arbitrary scripts into the pages.

What is cross-site scripting (XSS)?

- Injection attack.
- When an attacker uses a trusted web site to send malicious code to an unsuspecting user, generally in the form of a JavaScript or HTML browser-side script.

A benign user enters a comment as intended.

Show me an example of XSS

Your site has this:

```
<b> <%= username %> </b></br>
<div> <%= profileText %> </div>
```

A benign user enters a comment as intended.

```
http://www.example.com/saveComment?
comment=Great+Site!
<h3> Thank you for your comments! </h3>
You wrote:
<p/>
Great Site!
<p/>
```

An attacker uses the field to send malicious code to your site.

```
http://wwww.example.com/saveComment?comment=
<script>alert('xss');</script>
<h3>Thank you for your comments!</h3>
You wrote:
<p/>
<script>alert('xss');</script>
<p/>
```

What is cross-site scripting (XSS)?

Show me an example of XSS

Your site has this:

```
<b> <%= username %> </b></br>
<div> <%= profileText %> </div>
```

A benign user enters a comment as intended.

A benign user enters a comment as intended.

```
http://www.example.com/saveComment?
comment=Great+Site!
<h3> Thank you for your comments! </h3>
You wrote:
<p/>
Great Site!
<p/>
```

Benign (good) users

An attacker uses the field to send malicious code to your site.

```
http://wwww.example.com/saveComment?comment=
<script>alert('xss');</script>
<h3>Thank you for your comments!</h3>
You wrote:
<p/>
<script>alert('xss');</script>
<p/>
```

Attackers

XSS Prevention

Getting it right: This is the easiest case. Just HTML escape the dynamic content:

```
<div>
  Hello ${cov:htmlEscape(name)}!
</div>
```

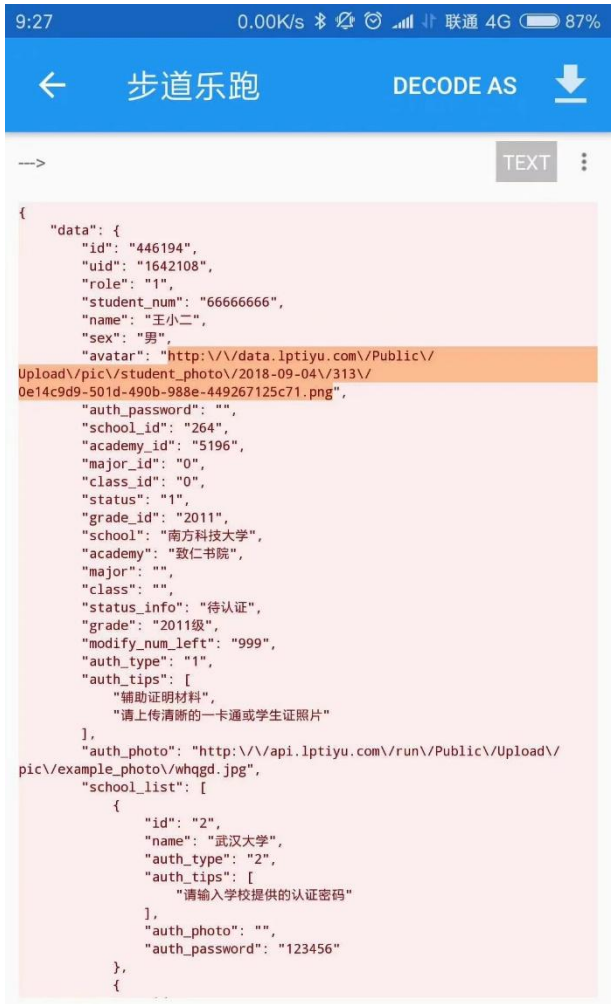
```
+
+ }
```

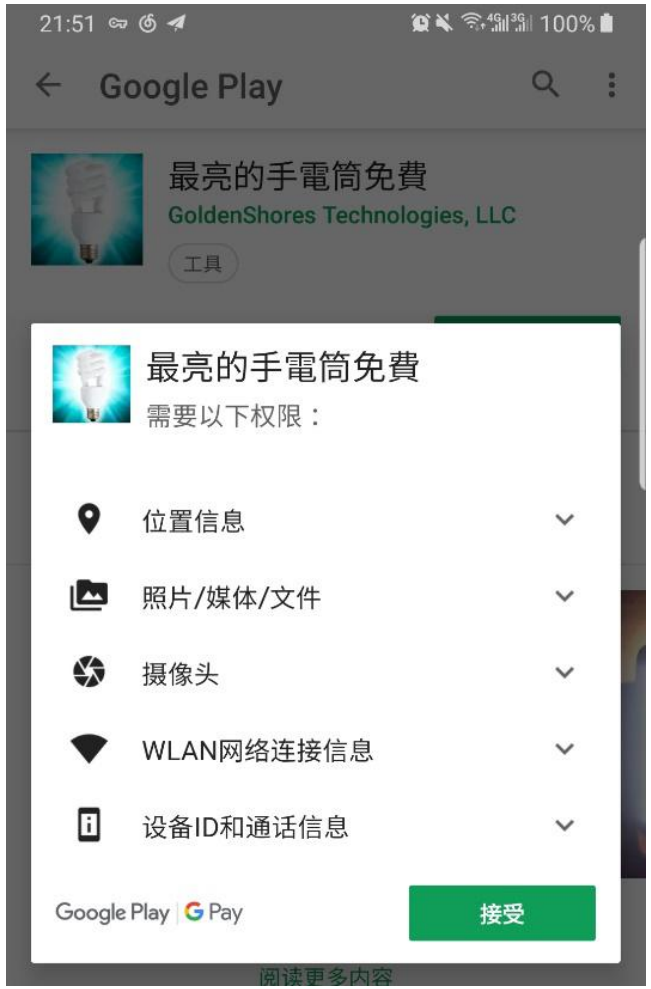
18 ...marconivr/jacopo/microblog/security/services/implementations/SimpleSanitationService.java

```
@@ -0,0 +1,18 @@
+ package edu.marconivr.jacopo.microblog.security.services.implementations;
+
+ import org.springframework.stereotype.Service;
+ import org.springframework.web.util.HtmlUtils;
+
+ import edu.marconivr.jacopo.microblog.security.services.ISanitationService;
+
+ @Service
+ public class SimpleSanitationService implements ISanitationService
+ {
+
+     @Override
+     public String escapeHTML(String original)
+     {
+         return HtmlUtils.htmlEscape(original);
+     }
+
+ }
```

Have you ever encountered a security problem in apps?

- For example, 步道乐跑





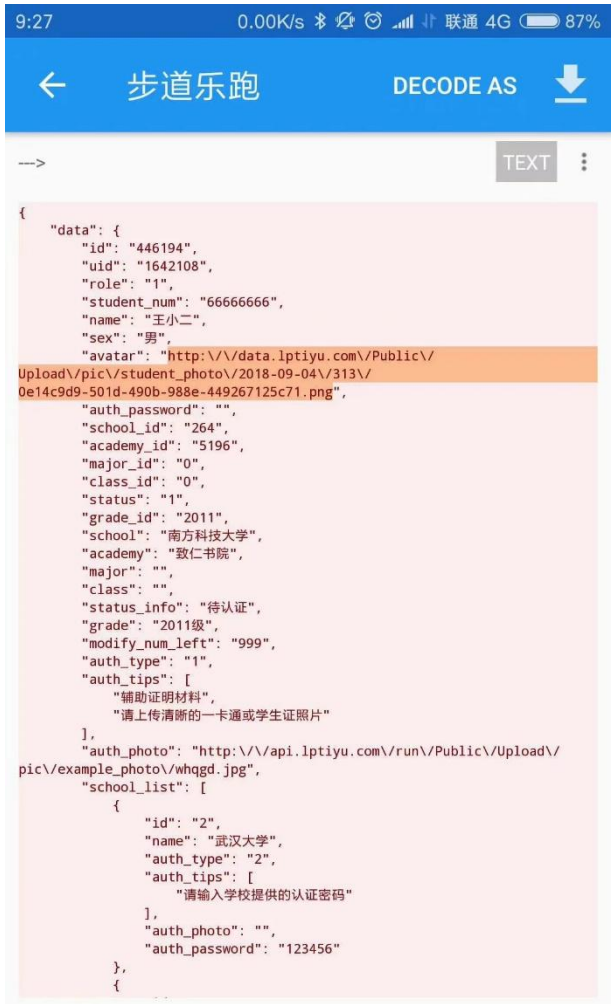
Question: How many security risk types are there?

- Weak Server Side Controls
- Insufficient Transport Layer Protection
- Poor authentication and Authorization
- Improper Session Handling
- Sensitive Information Disclosure

Weak Servier Side Controls

- What will you do when you find out a company handle your username and password in plain text?
- What will happen if their database was attacked?
- Question: How many of you store data in plain text in your App?
- You should encrypt the sensitive information in the server database, including username, password, location, tokens, etc.

Insufficient Transport Layer Protection



- If a hacker attacks your network, will your app becomes insecure?
- This picture again.
- Question: How many of you transfer data like 步道乐跑?
- Do not be like them!

Poor Authentication and Authorization

- Misusing already built in authentication libraries.

- Develop security logic to be managed and checked in the API and not in the APP itself. Because the client can be DECOMPILED!

Improper Session Handling

- Session Must TIMEOUT in case of inactively!
- 15 minutes for high security apps
- 30 minutes for medium security apps
- 1 hour for low priority apps

Sensitive Information Disclosure

- APIs can be crawled by search engine bots and hackers
- Expose only required information to your API!
- Carefully design your API, if someone maliciously attacks your public API, it should be robust enough to defence itself!

Answer the questions in Github Classroom README.md

Link:https://classroom.github.com/a/RwgDwl38

What kind of security problems you have in your app?

What you can do to induce the risk and improve security?

Reminder

- **Project Final Presentation Uploaded:**
  - **due on 22 May 2021, 11.59pm**
- **All lab exercises due on 24 May 2021,11.59pm**
  - **coverage lab: https://classroom.github.com/a/S8hiiL1J**
  - **junit lab(Pair programming): https://classroom.github.com/g/trrjQPYs**
  - **metrics lab: https://classroom.github.com/a/qMOJKfSm**
  - **reverse engineering lab: https://classroom.github.com/a/4Dfhej1A**
  - **ui-ci: https://classroom.github.com/a/Yuzgn5gJ**
  - **security (this week): https://classroom.github.com/a/RwgDwl38**