WebSec21 Artemis Project Phase

Eric Armbruster, Florian Freund

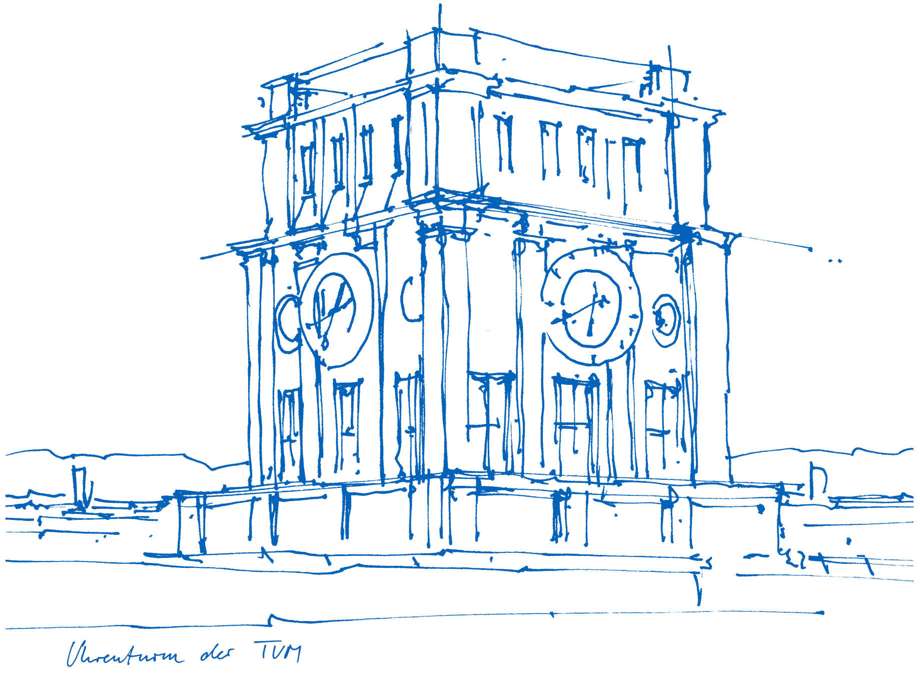
Team btw we use Arch

Technische Universität München

Fakultät für Informatik

Lehrstuhl für IT-Sicherheit

12. Juli 2021



Outline

- Coverage
- Overview of Findings
- Vulnerabilities and Live Demo
- Conclusion

Coverage

- Assessment
- Attachment
- Complaint
- Course
- File*
- Notification
- SystemNotification

- Mostly (but not only) checked for:
  - Path traversal
  - Access control
  - Stored XSS

Overview of Findings

- Authorization Bypass in Lecture Attachments
- Limited File Overwrite
- Arbitrary File and Folder Deletion
- Arbitrary Notification

- Not: weaknesses and bugs

Authorization Bypass in Lecture Attachments

- Type: missing access control
- Affected endpoints: createAttachment, updateAttachment, getAttachment, getAttachmentForLecture in AttachmentResource.java
- Description:
  - Role check with @PreAuthorize(hasRole(<role>)) is done
  - However, check for specific course is missing

Authorization Bypass in Lecture Attachments

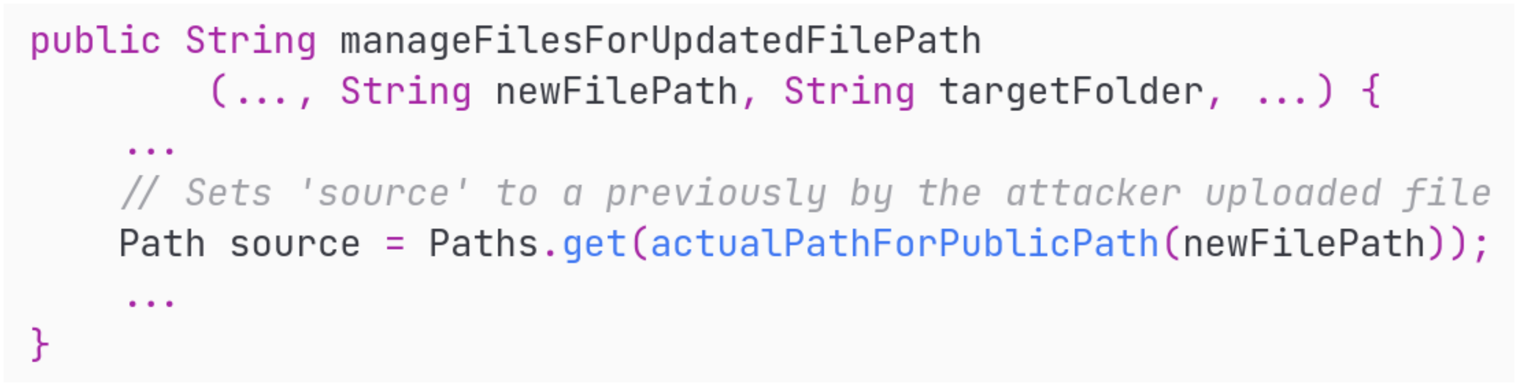
- Type: missing access control
- Affected endpoints: createAttachment, updateAttachment, getAttachment, getAttachmentForLecture in AttachmentResource.java
- Description:
  - Role check with @PreAuthorize(hasRole(<role>)) is done
  - However, check for specific course is missing
- Impact:
  - Severity: low
  - Limitation: editor role required, editors are rather trustworthy subjects
- Workarounds and Fixes: Add the missing check (see deleteAttachment)
- Note: another team also discovered missing access control, we discovered this first, however they discovered it for many more endpoints

Limited File Overwrite

- Type: file overwrite, deletion, creation
- Affected endpoints: createAttachment(), updateAttachment() in AttachmentResource.java

Attachment Upload Mechanism

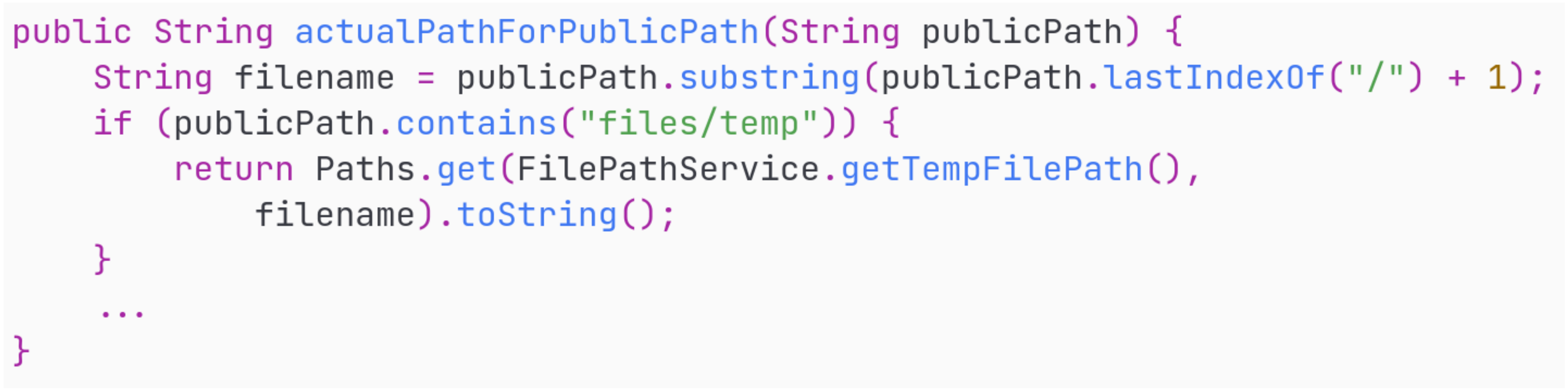
1. handleSaveFile() stores file in <temp upload>/images/temp/ (Request 1)
2. manageFilesForUpdatedFilePath()
   moves temp file into destination dir (Request 2)



```
public String manageFilesForUpdatedFilePath
        (..., String newFilePath, String targetFolder, ...) {
    ...
    // Sets 'source' to a previously by the attacker uploaded file
    Path source = Paths.get(actualPathForPublicPath(newFilePath));
    ...
}
```

Attachment Upload Mechanism

1. handleSaveFile() stores file in <temp upload>/images/temp/ (Request 1)
2. manageFilesForUpdatedFilePath()
   moves temp file into destination dir (Request 2)
   i. actualPathForPublicPath() generates temp path based on the request



```java
public String actualPathForPublicPath(String publicPath) {
    String filename = publicPath.substring(publicPath.lastIndexOf("/") + 1);
    if (publicPath.contains("files/temp")) {
        return Paths.get(FilePathService.getTempFilePath(),
            filename).toString();
    }
    ...
}
```

Attachment Upload Mechanism

1. handleSaveFile() stores file in <temp upload>/images/temp/ (Request 1)
2. manageFilesForUpdatedFilePath()
   moves temp file into destination dir (Request 2)
   i. actualPathForPublicPath() generates temp path based on the request
   ii. generateTargetFile() generates **destination path** based on the request

```
public String manageFilesForUpdatedFilePath
        (..., String newFilePath, String targetFolder, ...) {
    ...
    // Sets 'source' to a previously by the attacker uploaded file
    Path source = Paths.get(actualPathForPublicPath(newFilePath));
    // Inject arbitrary path into 'targetFile'
    File targetFile = generateTargetFile(newFilePath, targetFolder, keepFileName);
    ...
}
```

Attachment Upload Mechanism

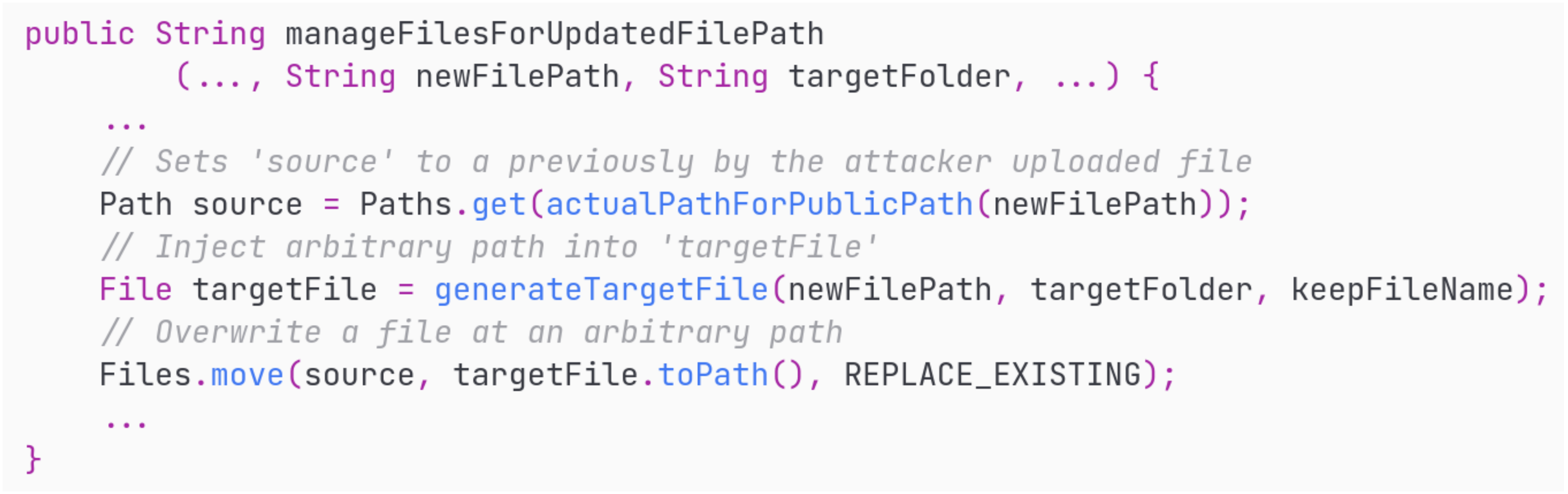
1. handleSaveFile() stores file in <temp upload>/images/temp/ (Request 1)
2. manageFilesForUpdatedFilePath()
   moves temp file into destination dir (Request 2)
   i. actualPathForPublicPath() generates temp path based on the request
   ii. generateTargetFile() generates **destination path** based on the request

```java
private File generateTargetFile(String originalFilename,
    String targetFolder, ...) throws IOException {
    ...
    var path = Paths.get(targetFolder, originalFilename).toString();
    File newFile = new File(path);
    newFile.delete();
    newFile.createNewFile();

    return newFile;
}
```

Attachment Upload Mechanism

1. handleSaveFile() stores file in <temp upload>/images/temp/ (Request 1)
2. manageFilesForUpdatedFilePath()
   moves temp file into destination dir (Request 2)
   i. actualPathForPublicPath() generates temp path based on the request
   ii. generateTargetFile() generates **destination path** based on the request
   iii. Move the file



```java
public String manageFilesForUpdatedFilePath
        (..., String newFilePath, String targetFolder, ...) {
    ...
    // Sets 'source' to a previously by the attacker uploaded file
    Path source = Paths.get(actualPathForPublicPath(newFilePath));
    // Inject arbitrary path into 'targetFile'
    File targetFile = generateTargetFile(newFilePath, targetFolder, keepFileName);
    // Overwrite a file at an arbitrary path
    Files.move(source, targetFile.toPath(), REPLACE_EXISTING);
    ...
}
```

Limited File Overwrite

- Type: file overwrite, deletion, creation
- Affected endpoints: createAttachment(), updateAttachment() in AttachmentResource.java
- Impact:
  - Severity: low-to-medium
  - Overwrite only if file ending: {png, jpg, jpeg, svg, pdf, zip} (see handleSaveFile in FileResource.java)
  - But arbitrary file deletion possible
  - Limitation: EDITOR role required, editors are rather trustworthy subjects

Limited File Overwrite

- Type: file overwrite, deletion, creation
- Affected endpoints: createAttachment(), updateAttachment() in AttachmentResource.java
- Impact:
  - Severity: low-to-medium
  - Overwrite only if file ending: {png, jpg, jpeg, svg, pdf, zip} (see handleSaveFile in FileResource.java)
  - But arbitrary file deletion possible
  - Limitation: editor role required, editors are rather trustworthy subjects
- Workarounds and Fixes:
  - Equip all endpoints that accept a file path with sanitization (e.g. call removeIllegalCharacters on the file path)
  - Sanitize newFilePath inside of manageFilesForUpdatedFilePath and generateTargetFile
  - Long term: two endpoints, allowing different file paths is not the best design

Arbitrary File and Folder Deletion

- Type: file and folder deletion
- Affected endpoints:
  - updateAttachment, deleteAttachment in AttachmentResource.java
  - updateCourse, deleteCourse in CourseResource.java
  - updateQuizExercise, deleteQuizExercise in QuizExerciseResource.java

Arbitrary File and Folder Deletion

- Type: file and folder deletion
- Affected endpoints:
  - updateAttachment, deleteAttachment in AttachmentResource.java
  - updateCourse, deleteCourse in CourseResource.java
  - updateQuizExercise, deleteQuizExercise in QuizExerciseResource.java
- Impact:
  - Severity: low-to-medium
  - Limitation: editor role required, editors are rather trustworthy subjects

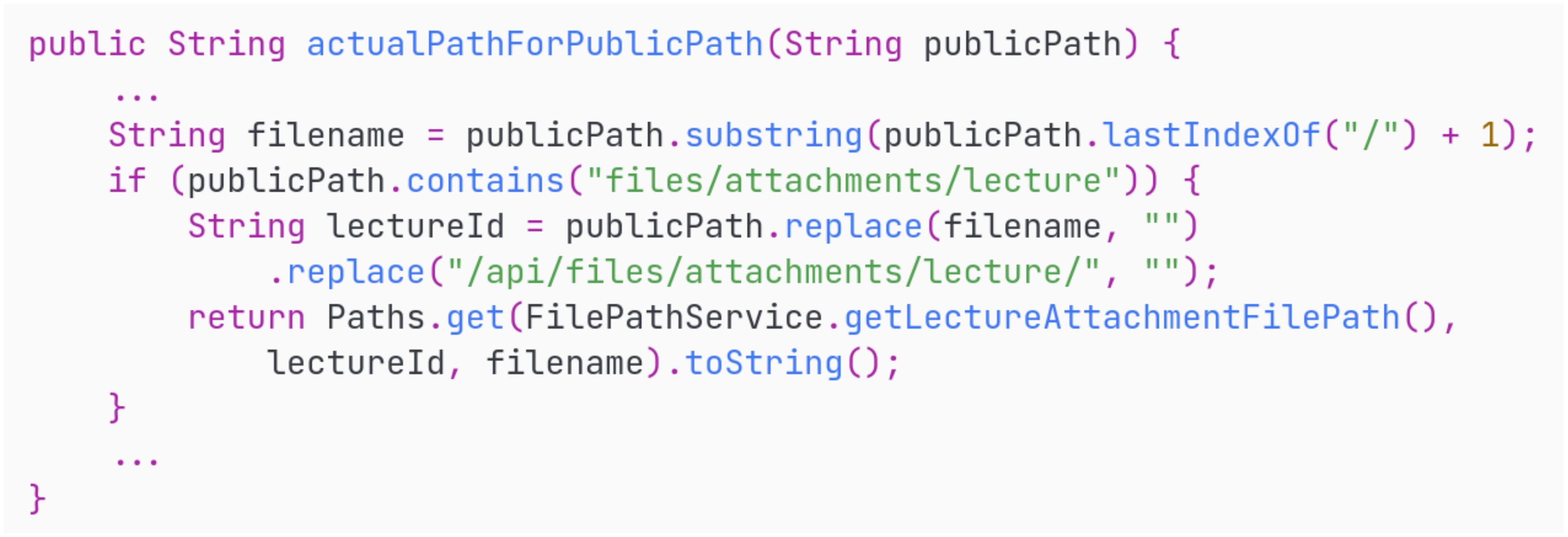
Arbitrary File and Folder Deletion

- oldFilePath is attacker controlled

```java
public String manageFilesForUpdatedFilePath(String oldFilePath, ...) {
    ...
    File oldFile = new File(actualPathForPublicPath(oldFilePath));
    FileSystemUtils.deleteRecursively(oldFile);
    ...
}
```

Arbitrary File and Folder Deletion

- oldFilePath is attacker controlled
- actualPathForPublicPath as before

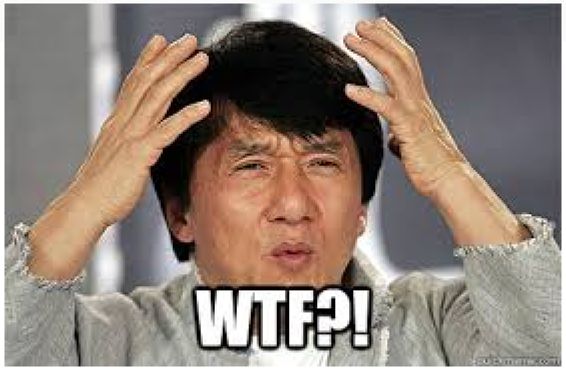


```java
public String actualPathForPublicPath(String publicPath) {
    ...
    String filename = publicPath.substring(publicPath.lastIndexOf("/") + 1);
    if (publicPath.contains("files/attachments/lecture")) {
        String lectureId = publicPath.replace(filename, "")
            .replace("/api/files/attachments/lecture/", "");
        return Paths.get(FilePathService.getLectureAttachmentFilePath(),
            lectureId, filename).toString();
    }
    ...
}
```

Arbitrary File and Folder Deletion

- oldFilePath is attacker controlled
- actualPathForPublicPath as before

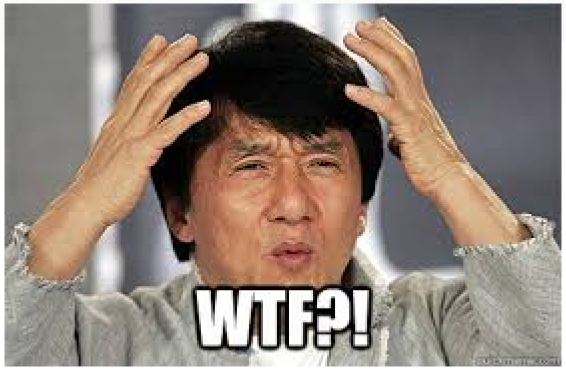
```java
public String manageFilesForUpdatedFilePath(String oldFilePath, ...) {
    ...
    File oldFile = new File(actualPathForPublicPath(oldFilePath));
    FileSystemUtils.deleteRecursively(oldFile);
    ...
}
```

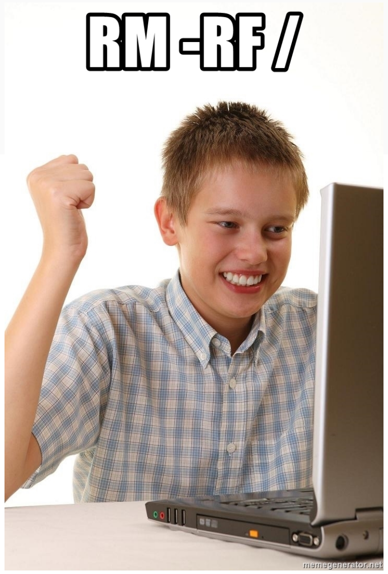


Arbitrary File and Folder Deletion

- oldFilePath is attacker controlled
- actualPathForPublicPath as before

```
public String manageFilesForUpdatedFilePath(String oldFilePath, ...) {
    ...
    File oldFile = new File(actualPathForPublicPath(oldFilePath));
    FileSystemUtils.deleteRecursively(oldFile);
    ...
}
```





Arbitrary File and Folder Deletion

- Type: file and folder deletion
- Affected endpoints:
  - updateAttachment, deleteAttachment in AttachmentResource.java
  - updateCourse, deleteCourse in CourseResource.java
  - updateQuizExercise, deleteQuizExercise in QuizExerciseResource.java
- Impact:
  - Severity: low-to-medium
  - Limitation: editor role required, editors are rather trustworthy subjects
- Workarounds and Fixes:
  - Pro tip: Use Files.delete instead of Files.deleteRecursively for file deletion
  - Furthermore:
    - Equip all endpoints that accept a file path with sanitization (e.g. call removeIllegalCharacters on the file path)
    - Sanitize oldFilePath inside of manageFilesForUpdatedFilePath

Live Demo

Arbitrary Notification

- Type: identity spoofing enables notification creation, overwrite, deletion
- Affected endpoints: createNotification, updateNotification, deleteNotification, getNotification

Arbitrary Notification

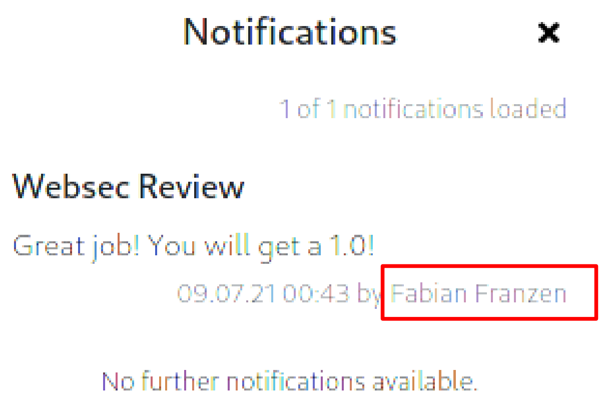
- Type: identity spoofing enables notification creation, overwrite, deletion
- Affected endpoints: createNotification, updateNotification, deleteNotification, getNotification

```java
public ResponseEntity<Notification> createNotification(Notification notification) {
    restrictSystemNotificationsToAdmin(null, notification);
    Notification result = notificationRepository.save(notification);
}
```

```java
public class SingleUserNotification extends Notification {
    private String title;
    private String text;
    private ZonedDateTime notificationDate;
    private User author;
    private User recipient;
}
```

Arbitrary Notification

- Type: identity spoofing enables notification creation, overwrite, deletion
- Affected endpoints: createNotification, updateNotification, deleteNotification, getNotification


Notifications

1 of 1 notifications loaded

Websec Review

Great job! You will get a 1.0!

09.07.21 00:43 by Fabian Franzen

No further notifications available.

Arbitrary Notification

- Type: identity spoofing enables notification creation, overwrite, deletion
- Affected endpoints: createNotification, updateNotification, deleteNotification, getNotification
- Impact:
  - Severity: low-to-medium
  - Limitation: instructor role required, instructors are trustworthy subjects
- Workarounds and Fixes: Add the following lines to the above endpoints:
  - User currentUser = userRepository.getUserWithGroupsAndAuthorities();
  - if (currentUser != notification.author) throw ...

Conclusion

- Found a total of four vulnerabilities
- The crux for webapp security in complex applications like Artemis often lies in detail and requires thorough analysis
- For the Future: The Artemis team needs someone that thinks endpoints through from an attacker perspective (e.g. notification endpoints, but also unsanitized file paths)

Conclusion

- Found a total of four vulnerabilities
- The crux for webapp security in complex applications like Artemis often lies in detail and requires thorough analysis
- For the Future: The Artemis team needs someone that thinks endpoints through from an attacker perspective (e.g. notification endpoints, but also unsanitized file paths)