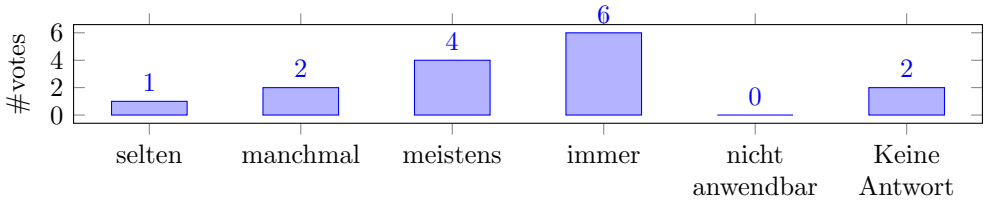
Modul: Secure Software Engineering	**Semester:** WS 22/23

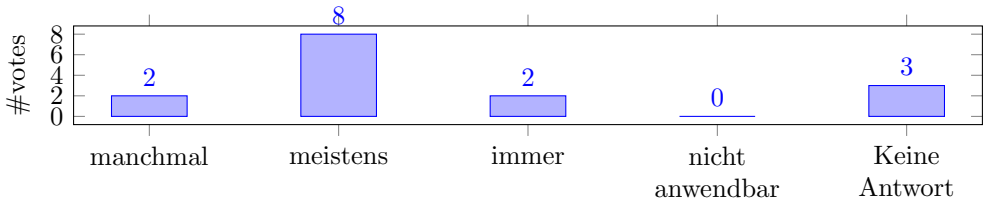
Ergebnis der Online-VLU. Die Umfrage fand in den letzten beiden Vorlesungswochen statt.

1 Bewertung der Vorlesung

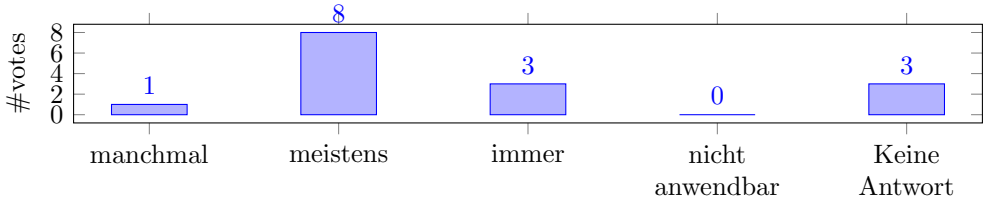
Wie oft hast du die Vorlesung besucht?



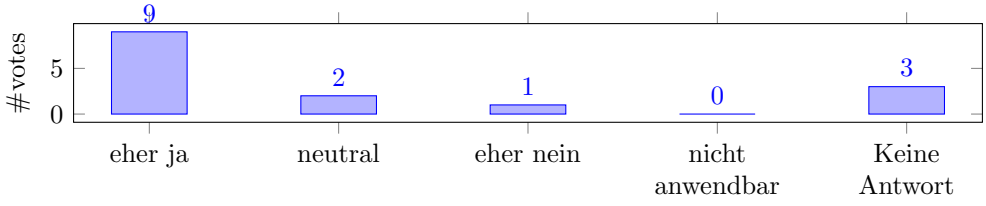
Wurden Themen durch Beispiele veranschaulicht?



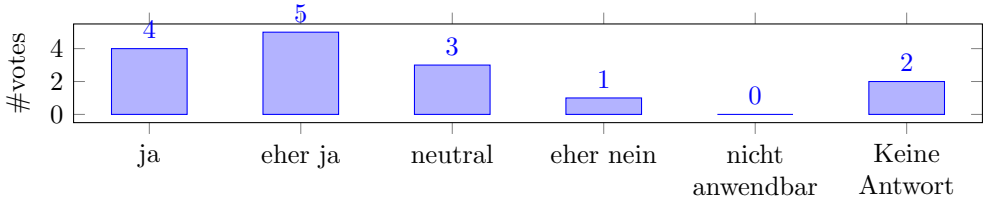
Wurden die Themen ausführlich genug erklärt?



War die Struktur der Vorlesung klar zu erkennen?

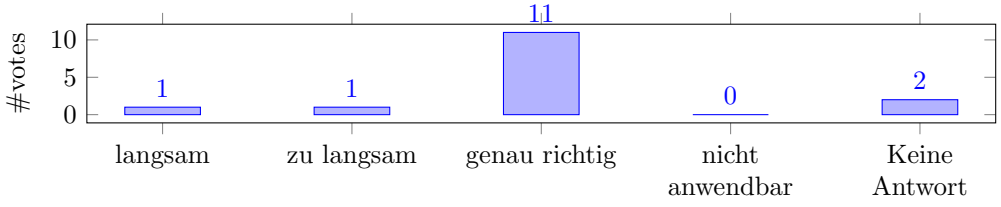


Waren die Folien/das Skript hilfreich?

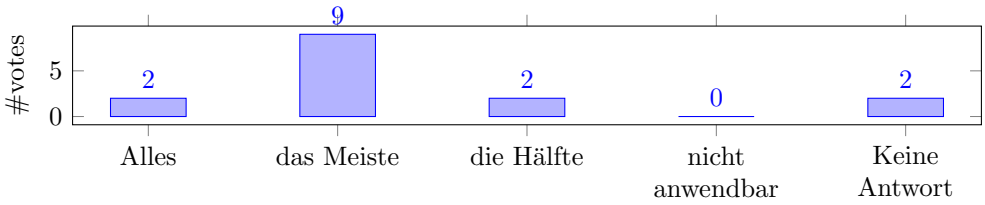


2 Bewertung der Dozierenden

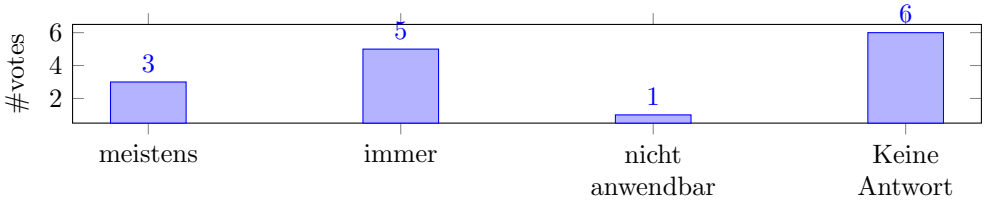
Die Geschwindigkeit der Vorlesung war...



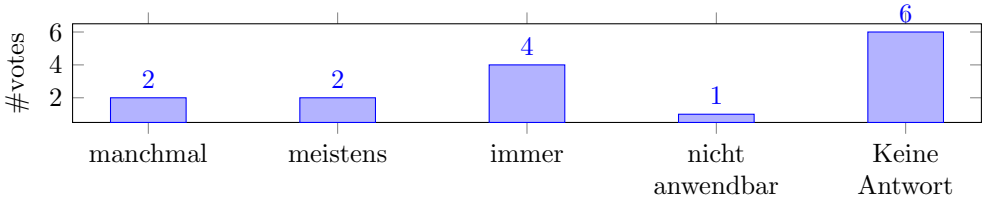
Wie viel verstehst du während der Vorlesung?



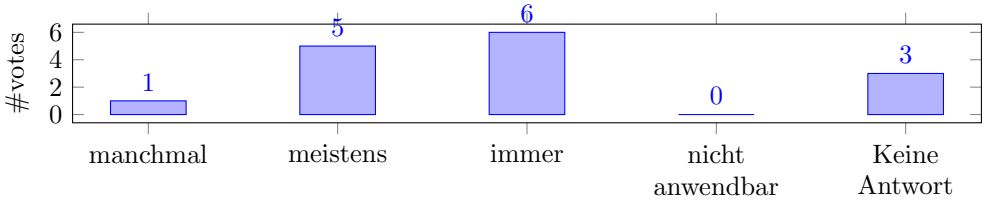
Ist der Dozent/die Dozentin gut auf Fragen eingegangen?



War der Dozent/die Dozentin außerhalb der Vorlesung für Fragen etc. erreichbar?

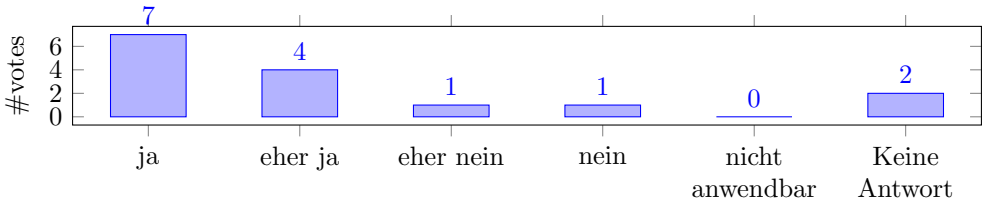


War die Dozentin / der Dozent akustisch gut zu verstehen?

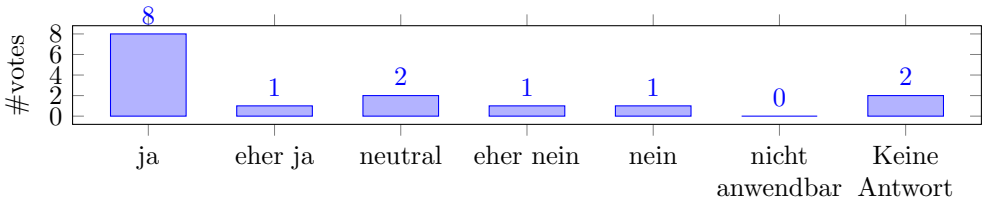


3 Bewertung des Moduls

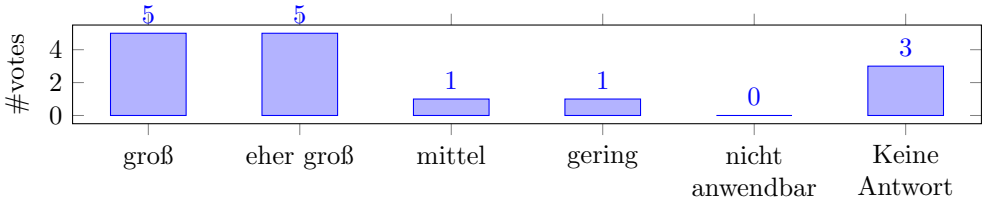
Findest du die verlangten Studienleistungen für dieses Modul angemessen?



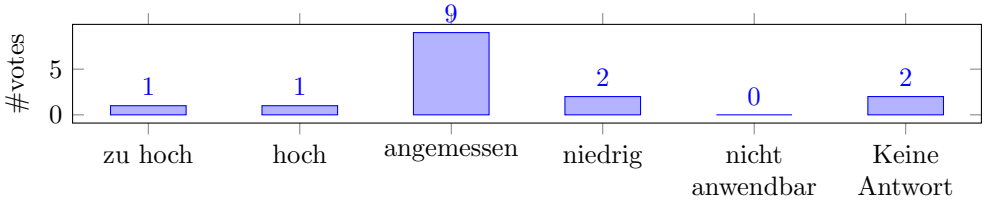
Würdest du das Modul weiterempfehlen?



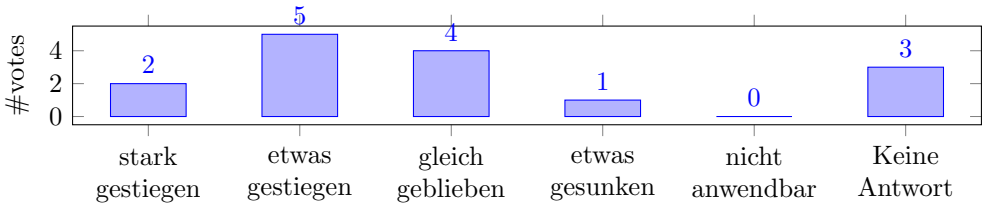
Der Praxisbezug war...



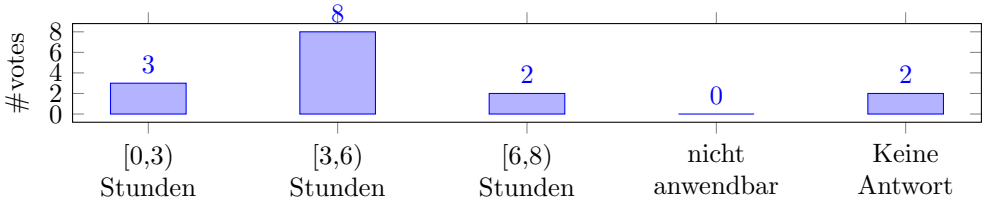
Ist der Arbeitsaufwand für dieses Modul im Hinblick auf die LP-Zahl angemessen?



Dein Interesse für dieses Thema ist...

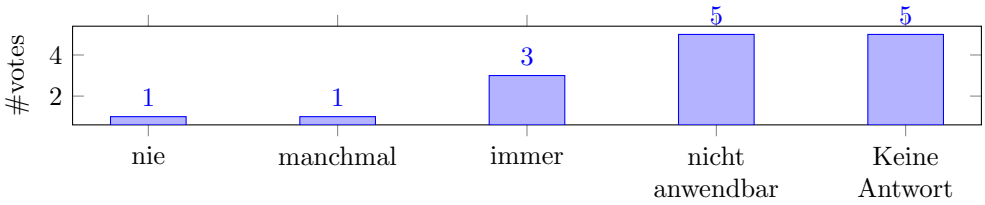


Wie viele Stunden hast du insgesamt, inkl. Vorlesung, Übung, Übungsaufgaben..., pro Woche für dieses Modul aufgewendet?

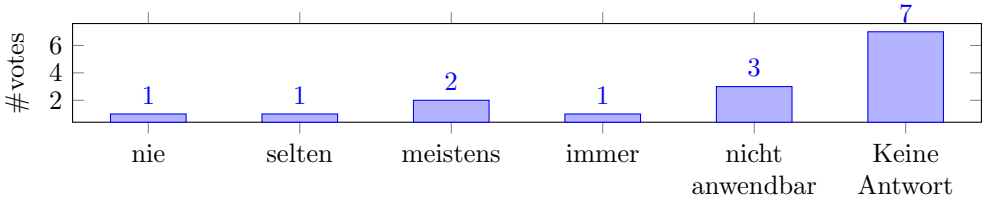


4 Bewertung der Übungsaufgaben

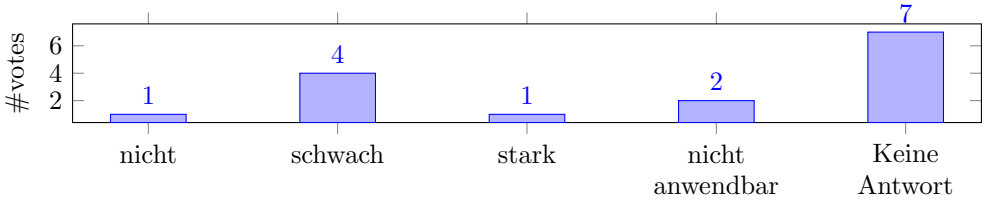
Wie oft hast du die Übungen besucht?



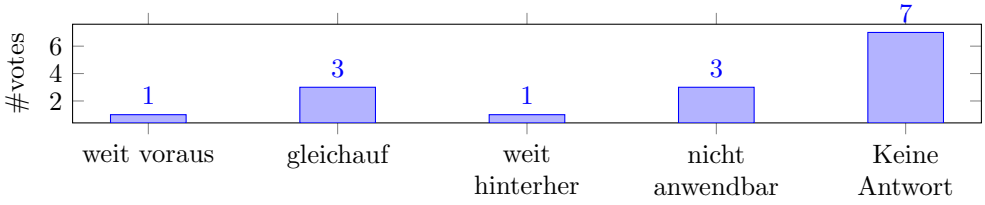
Wurden die Übungsaufgaben rechtzeitig zur Verfügung gestellt?



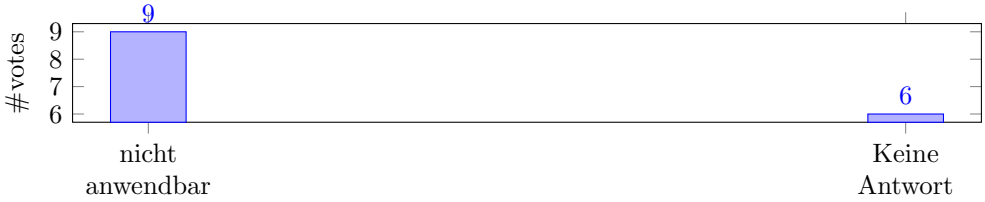
Die Schwierigkeit der Übungsblätter schwankte...



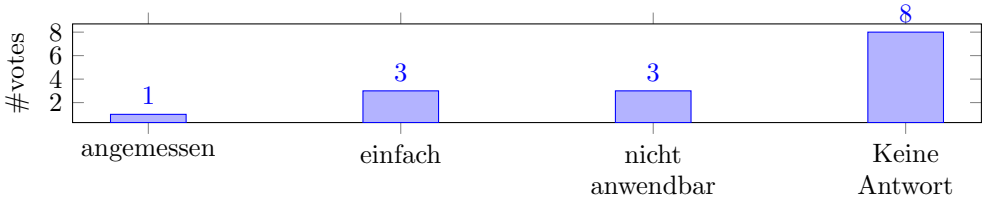
Die Vorlesung war...



Die Übungsgruppe war...

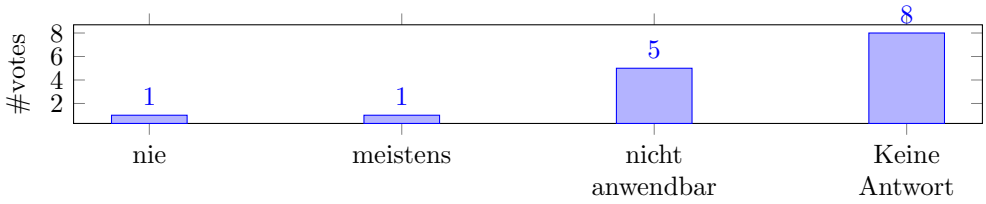


Die Übungsaufgaben waren meistens...

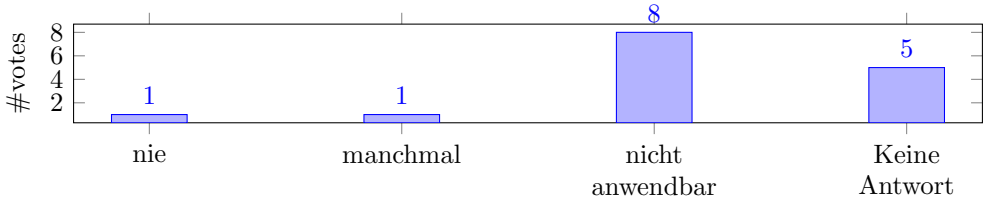


5 Bewertung des Tutoriums

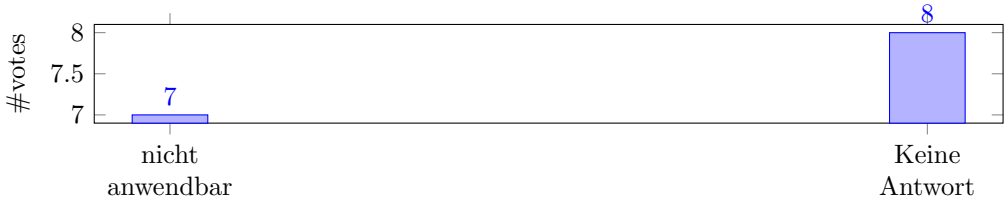
War der Tutor/die Tutorin außerhalb der Übung für Fragen etc. erreichbar?



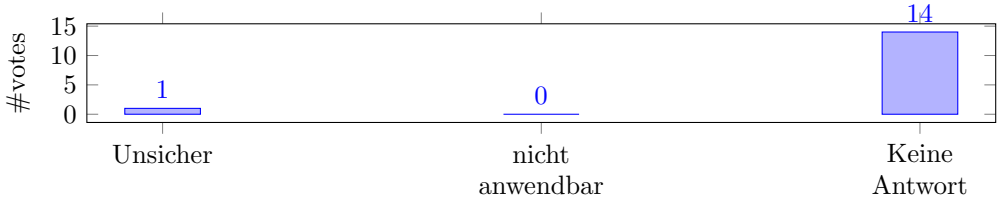
Waren die Korrekturen des Tutors/der Tutorin nachvollziehbar?



Wurde der Tutor/die Tutorin mit dem Stoff der Übung fertig?

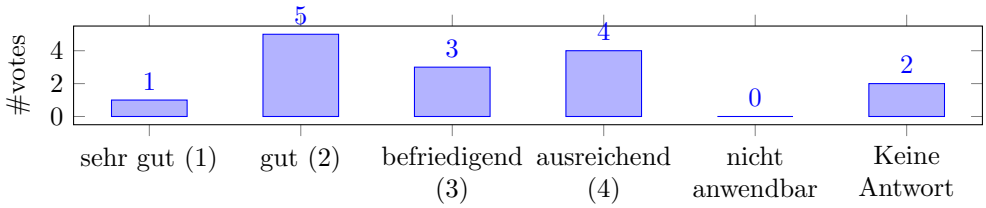


Lohnt sich der Besuch des Tutoriums?



6 Abschließende Bewertung des Moduls

Note:



6.1 Hälst du die Vorlesung der Dozent:in für Lehrpreiswürdig und falls ja, warum?

Nein
Yes

7 Freitextkommentare

7.1 Was hat dir an dieser Lehrveranstaltung gefallen?

The practical aspect of doing the coding challenge (bibifi)
Very interesting lecture, a lot of intuitive practices I already used now have proper names and procedures
Hybride Veranstaltung. Die Idee mit den möglichen Klausurfragen fand ich mega cool. Hochgeladene Solutions zu den exercises sind auch mega gut.

build it, break it, fix it, practical relevance, vulnerability of the week
Interesting content; I liked the idea of the bibifi contest
I really liked the vulnerabilities of the day, they gave me a nice overview of important attacks that are commonly used.
- CVE were always a good start into the lecture - some topics are good explained

7.2 Was könnte noch besser gemacht werden?

Lecture should be structured better: Less topics, but more in depth. Also have more exercise sheets that accommodate the lecture, such that the students get an idea what is expected from the exam. Even though this is such a practical lecture, which I think is great, parts are off the reality: To give an example, software development and code reviews are usually not done as presented. Git flow is often used different, and there are multiple ways of doing it, trunk based, feature based, etc..
There post probably a need to add content. It had by far the sparsest information amount of all modules I took in my masters.
The contents of the lecture often feel shallow, not going into depth like other lectures do. Often there is blah blah on the slides, mostly a long motivation for some quickly explained technique.

The bibifi platform must work flawlessly, otherwise it is not fair for the students and impossible to properly manage the time. This includes the grading, but also the oracle. Also I would like to have a more open communication about how the bibifi system works (not access to the code, but written explanation). For example: how are the scores calculated, what do the tests consist of, what is the oracle, how are the break submissions checked in the backend...
The bibifi project is very interesting and was fun to work on, but the exercises should be part of the exam prep as well and not be a side task (i.e. correction and on time solutions). Also writing the reports and therefor gaining or not gaining the exam admission should be different as it currently is a win or lose scenario (was no problem for me, but if someone wrote a report that was "not good enough", they would not have a chance to get feedback and improve upon it, which is unfair). If there would be exercises with points you could see that your solution is not good enough and improve on the next sheet.
Another thing is that the amount of work for bibifi varied very strongly. Not only but also due to downtime of the platform. So there were weeks, mostly in the beginning, where we would spend 20+ hours on the project, but then weeks (mostly after in the deadline extension period of the build it phase and in the break it phase) where would had nothing to do at all. Also, while it was fun, I have the feeling that the time spent on the project does very very little to prepare me for the exam. It is more of a preparation for teamwork / group projects and management for life (which is good and strangely seldomly experienced at the university..., but does not help me for the exam).
The concept in the current form (three phases over the complete semester) with non mandatory uncorrected exercises and two mandatory reports is not viable in my opinion. My proposal would be weekly exercises that are split in week A and B and each have a duration of two weeks. In week A there are theoretical tasks that prepare the student for the exam and in week B there are practical tasks that take the students more by the hand in the bibifi project. I would publish all tasks at the beginning of the course as this would allow to progress faster in the project if needed, but the weekly tasks and tutorial would lead to more spread out engagement with the topic.
I would incorporate the reports in the theoretical exercise tasks (A-weeks). Each spread out over multiple sheets. So for example in the first week there should be a description of the threat-model, but nothing about implementation. Then in the third week, after the groups are formed and started working on their concept (libraries, security mechanisms, who does what) everyone should write a few paragraphs about the security considerations and decisions they had to make.
On the project side of things, I think that a more structured approach would open opportunities to better apply lessons from the lecture, like testing and reviewing. But this could also be done on example code: write some tests in one week based on specifications for a function / class (names and parameters given) and evaluate the next week on some faulty example functions given by the lecture. This in turn could be used in the break it phase to find mistakes of other teams.
A drawback would be that a too hands on approach would lead to fewer mistakes in the build it phase and therefore fewer possible exploits and bugs in the break it phase. But this could be countered with faulty example implementations that are available in the break it phase to exploit for the groups. This would also guarantee that there are bugs and exploits to find in the break it phase, which could also be accompanied with theoretical tasks or knowledge from the lecture before.
How would grading look like? The theoretical tasks are pretty straight forward: there are points for correct solutions and part solutions. The tasks derived from the reports would need to have some criteria in the formulation of the task about which aspects are important to include. The programming tasks would be evaluated by the bibifi system. For the build it phase there would be a maximum amount of points which is reached if the space and time scores are better or equal to the reference implementation. Otherwise the reached percentage of points is gained (timescore team: 500, ref impl: 400 -> 400/500= 80% of half of the points gained, same for space, better calculation needed, maybe some non-linear scaling). After the phase is concluded the best commit is taken and the points are added to the global total. In the break it phase there should be just an upper limit on how many exploits a graded and this can then be combined with a theoretical task again for the report side of things. Grading of fix it does not make sense, but there could be practical tasks where you have to fix faulty example code and write an explanation.
All in all, I think that this module has the potential to be one of the most valuable modules offered. I have no idea what it would take to make it a 9 LP module, but considering the potential depth of the software development lifecycle and the amount of time invested into the module by me already, I would think it is worth it.

Die 3 Phasen haben leider nicht gut funktioniert. Der Server war häufig nicht erreichbar und am Ende wurden nicht einmal alle Phasen bearbeitet aus Zeitgründen. Die Klausur, auch wenn noch nicht geschrieben, findet in einem für mich wesentlich zu weit entferntem Teil statt. Zu der normalen Uni habe ich bereits eine 1.5h Anreise. Zu dem Venusberg sind es nochmal mind. 30 Minuten mehr, dafür dass die Klausur am Computer "vermutlich über ecampus" geschrieben wird. Wieso nicht direkt online? Es ist eine Hybride Veranstaltung, könnte man da nicht beides einrichten, mir entzieht sich der Sinn so lange zu fahren um am Ende ein schlechteres Setup (Umfeld, Lautstärke, etc.) zu haben, als wenn ich von Zuhause via. Zoom die Klausur schreiben könnte.
Improve the bibifi, lecture slides
Sometimes the structure of the slides could be better
The vulnerabilities of the day. I think it would be nice to see some example mitigations, especially in the context of modern web applications. For example, CSRF mitigations would look different for modern Single Page Apps and classic template-based apps.
1. Bibifi - let people please choose their programming language, or give a variety of languages (maybe C/C++, Rust, Java, Python) to choose from - or if java is mandatory, PLEASE give a proper build system and EXPLAIN how to LINK STUFF - is does not make ANY sense to use MAKE FILES with an OFFLINE DOCKER CONTAINER ==> it took me more time to link a external lib then to actually program (coming from a C++/Python background and NOT knowing java) - and maybe it would be great to use all the parts of the lecture in bibifi instead of throwing a problem at one (software projects which has more stages, more near to the lecture) 2. Tutorials/Question sheets - the sheets did not cover nearly enough of the lecture and had no impact at all (covered so little of the lecture) ==> make sheets OR bibifi, both make no sense - the "solutions" were really bad sometimes (like instead of answering the question with context; look at lecture X slides Y, Z)