**PoC and Patching - Validation Logic Disparity**

Now, we can go ahead and test our theory, and validate that we are able to book an exam slot even if there aren't any available ones.

**Proof of Concept**

First, we will restart our Docker container again to restore the original state of exam booking and remove any breakpoints we may have added to the code. For this demo, we have filled the remaining exam slots in the database to reflect what we will have on the real target (no available slots).

After that, all we need to do is send an exam booking request, like the one we saw when reviewing the front-end application, and only change the date value to any unavailable date, as follows:

Code: json

{

"id": 1,

"date": "2023-09-14T23:00:00.000Z"

}

Once we do, we do indeed get a booking confirmation:

A screenshot of a computer

Description automatically generated

Finally, we can refresh the /exams page, and we will see that our exam now says BOOKED, and we can view the exam:

A screenshot of a computer

Description automatically generated

As we can see, this logic disparity flaw allowed us to book an exam, even though all exam slots were fully booked. The same flaw applies to other scenarios, like allowing us to purchase an item that is unreleased or out of stock.

In this case, the missing validation test was on the back-end, and such flaws are always more serious as they allow us to modify data on the database. However, Logic Disparity issues may also be caused by missing validations on the front-end, as mentioned before. For example, and item may be in stock and available for purchases, and the back-end may be validating items correctly. However, an issue with front-end validation may show the item as out of stock, making customers unable to purchase it and leading to lost revenue, which is another common logic bug.

All of this should give us a very clear idea of how Validation Logic Disparity issues may arise and how to identify them and exploit them. Next, we will see tips on how to avoid such flaws.

**Note:** As this is a Hard module, the exercises will not match the same demo shown in the sections, and this specific vulnerability is patched on the real target. We will instead test your understanding of the content with a similar Logic Bug but in a different context. You may still use the same source code to test what is being shown in the section, as well as test and identify another logic bug, as discussed in the exercise.

**Patching**

We have proven that the vulnerability does exist and can be exploited, so let's see how we would remediate it by patching the code. The main thing that led to this vulnerability is a missing exam availability test on the back-end, as the application relied on the test done on the front-end.

So, we simply need to add this before proceeding with the exam booking. To do so, we can add the below code on line 191 in the /controllers/exam-controllers.js file:

Code: js

// ensure exam slot is available

const bookedExams = UserExam.find({

examId: exam.id,

date: new Date(date),

used: false,

});

if ((await bookedExams).length > 0) {

return next({

message: "Exam slot is not available.",

statusCode: 400,

});

}

This test will simply ensure that the selected exam slot is empty, before proceeding with the booking. Of course, this patch is for this specific case. In general, the remediation of a logic disparity is by bringing the logic back to parity on both ends, which means adding any missing tests on either end, like the one above that was missing on the back-end.

**Exercise:** Try to patch your code and then re-apply the same above PoC, as a way of confirming that fix. If it is still vulnerable, then the patch does not properly remediate the issue.