



Broken Authentication and Session Management

Cybersecurity

Web Vulnerabilities and Hardening Day 3



Class Objectives

By the end of class, you will be able to:



Exploit broken access controls by executing a client-side JavaScript validation bypass attack.



Exploit broken authentication by executing attacks on insecure login forms attacks, logout management, and administrative portals.



Use WebScarab and CyberChef to manipulate cookies and execute command injection.



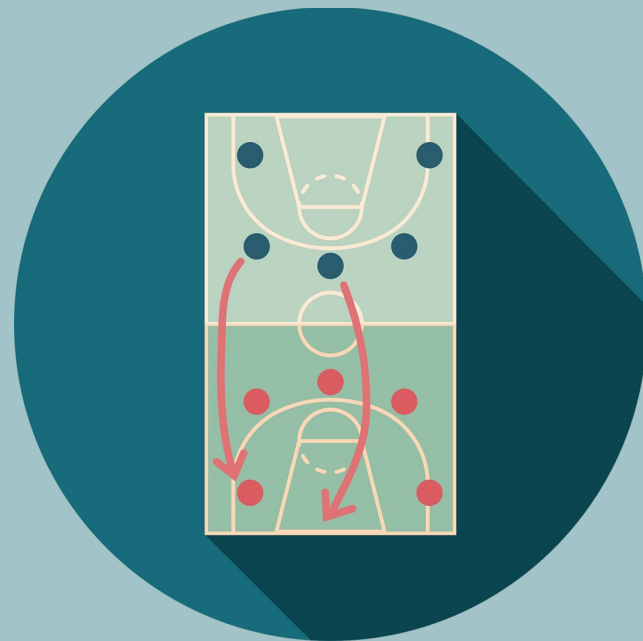
Provide mitigation strategies for all attacks executed.

Offense Informs Defense

Throughout this unit, we will act out examples of malicious attacks to show how various hacks and exploits work and how we can better defend against them.

It is important to note that the skills we learn in offensive security units should only be used ethically and with permission.

The actions and intents of criminal hackers, hacktivists and other malicious actors that we mimic for demonstrations are in no way condoned or encouraged.





Any actor or group
can be a threat
agent for exploiting
access controls.



1. Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

2. —OWASP Top 10

Validation Bypass

Security code analysis tools and vulnerability scanning tools can detect a lack of access controls.



Validation Bypass

Weak access control mechanisms result from a lack of automated detection and a lack of proper testing by application developers.

Adobe leaves Creative Cloud database open, 7.5 million users exposed

Doug Olenick



An unsecured Elasticsearch database left exposed the account information of about 7.5 million Adobe Creative Cloud users.

Comparitech, in association with security researcher Bob Diachenko, found the Adobe database, which could be accessed without a password or any login credentials. The company was notified on October 19 and the database was locked down that day.

Validation Bypass

Manual testing can also detect missing or ineffective access controls such as proper HTTP GET and PUT methods.

HTTP Method	Description
GET	Requests data <i>from</i> a server.
POST	Sends data <i>to</i> a source, often changing or updating a server.
PUT	Replaces current data with the new value.
DELETE	Deletes a specified resource.
CONNECT	Establishes a tunnel to the server.
OPTIONS	Lists the communication options for target resource.

Validation Bypass

Some common access control vulnerabilities include bypassing access control checks by:



Manipulating the URL.



Modifying the HTML code on a webpage.



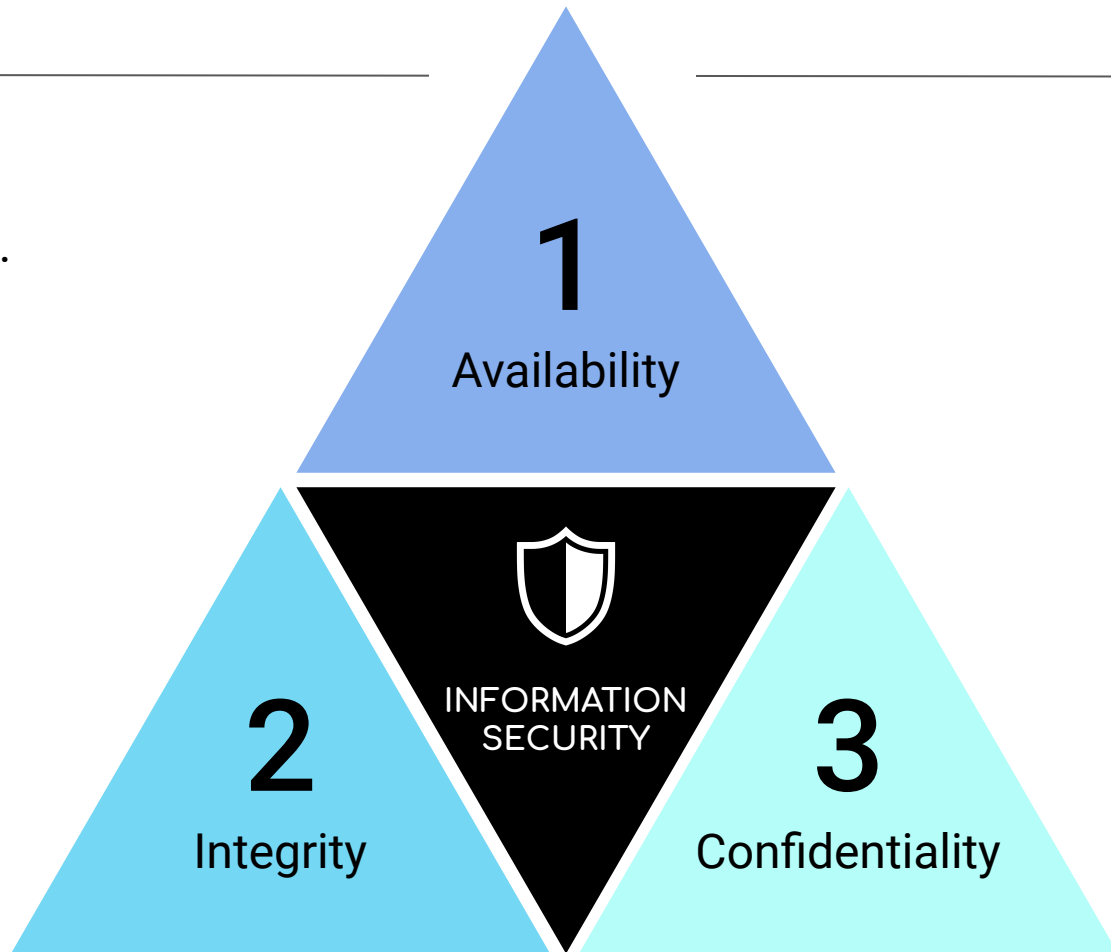
Using a custom API attack tool.



Manipulating hidden form fields or cookies.

Validation Bypass

Validation bypass exploits mostly affect confidentiality and integrity of the CIA triad.



Validation Bypass

Attackers can use validation bypass breaches to assume the identity of users or administrators with escalated privileges, allowing them to create, access, update, or delete data.



Validation Bypass

Command vulnerability bypass attacks can also result in a complete takeover. The degree of impact depends on the business needs of the data and the application it runs on.



2 Million IoT Devices Vulnerable to Complete Takeover

By Lindsey O'Donnell

Millions of security cameras, baby monitors and “smart” doorbells are open to hijack—and no solution is currently available.

Over 2 million IP security cameras, baby monitors and smart doorbells have serious vulnerabilities that could enable an attacker to hijack the devices and spy on their owners—and there's currently no known patch for the shared flaws.



Now we'll demonstrate
how to exploit access
control vulnerabilities
on client-side JavaScript.



Instructor Demonstration

Client-Side JavaScript Validation Bypass

Mitigation Strategies: Validation Bypass

Access controls are only effective on the server side, where attackers cannot modify control checks and metadata.



Mitigation Strategies: Validation Bypass

Alternative methods of prevention include:



Implicitly deny all (except public resources).



Reuse access control mechanisms after implementation.



Disable web server directory listings, ensure backup files aren't present in web roots.



Ensure logging of all access control failures and immediate generation of alerts.



Activity: Validation Bypass

In this activity, you will bypass input validation and then perform a SQL injection to exploit a vulnerability that you've discovered in a company's web server.

Suggested Time:
15 Minutes





Times Up! Let's Review.

Broken Authentication and Session Management



Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

—OWASP Top 10

Broken Authentication

Flaws in the implementation of user authentication and session management have serious implications for businesses. A security breach can expose confidential information and create backdoors.



Broken Authentication

The following implementation issues can lead to vulnerable web applications:

- Automated attacks that allow brute-force or credential stuffing.
 - Allowing the use of well-known passwords, weak or default passwords.
 - Inefficient password recovery mechanisms.
 - Unencrypted passwords or weak hashes.
 - Absence of multifactor authentication.
 - Exposure of session ID in the URL.
 - Non-rotation of session IDs after each use.
 - Improper session invalidation or timeouts.
 - Inadequate security built into web application software.
-

Broken Authentication Mitigation Strategies

We'll demonstrate how to perform a broken authentication and session management attack with the following scenario:



Factor 1

Standard login inputs
(passwords, PIN,
cognitive questions)



Factor 2

Physical keys
(smartcards, hard
tokens)



Factor 3

Biometrics
(Iris/retina scans,
hand geometry)



Factor 4

Location
(GPS detection, callback
to a home phone
number)

Broken Authentication Demo

We'll demonstrate how to perform a broken authentication and session management attack with the following scenario:



We'll play the **role of a junior security administrator** working at FireFly Inc., a web application software developer.



We've been asked by the lead project engineer to do a few **penetration tests against their website** and provide **remedial recommendations**.

Broken Authentication Demo

We'll cover the following attacks:

01

Insecure login forms existing in web page forms as a result of insecure web application programming.

02

Logout management, which allows previously logged out users to log back in by hitting the Back button.

03

Administrative portals, which reveal authentication mechanisms in the URL.



Instructor Demonstration

Broken Authentication

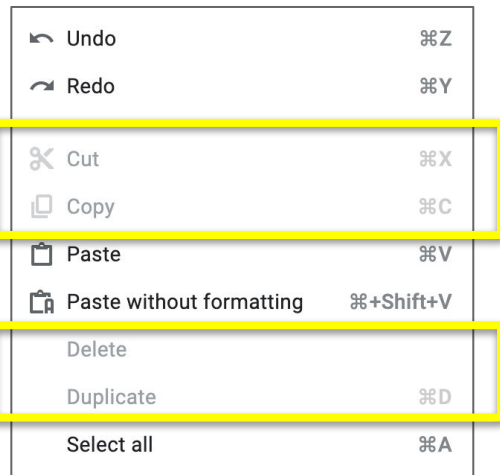
Broken Authentication Mitigation Strategies

Constrained user interface restricts what users can see and do based on their privileges.

Request edit access

You can only view this document. To make changes, ask the owner for edit access.

Context-dependent controls regulate activity-based functions, such as your ability to perform tasks like editing a document.



Content-dependent controls regulate the content of an object, such as grayed-out menu items.

Broken Authentication Mitigation Strategies

01

Polyinstantiation

Assigns the storage of information at different classification levels, preventing hackers from understanding the information without the missing pieces.

02

API Keys

Like passwords, should be treated as very sensitive information. They should always be stored in secure locations and transmitted only over encrypted communications channels.



Activity: Broken Authentication

In this activity, you will exploit a company's broken authentication system and provide mitigation strategies.

Suggested Time:
20 Minutes





Times Up! Let's Review.





Countdown timer

15:00

(with alarm)

Encoded Cookies



Cookies are typically sent using a type of encoding called **base64**.

Encoded Cookies

Base64 is an encoding scheme that represents a binary set of data in an ASCII string format.

01

ASCII characters

Base64 encoding schemes are most common when data needs to be stored and transferred over media that is designed to deal with ASCII characters, such as web-based datastreams like URLs.

02

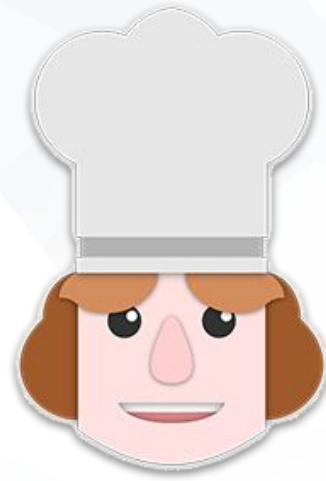
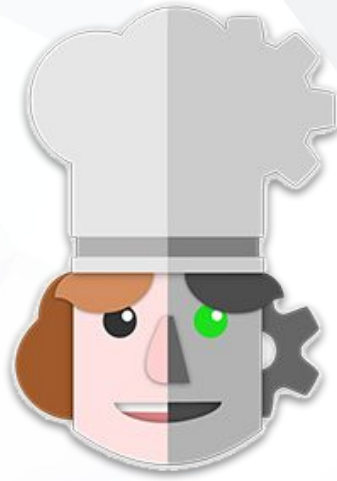
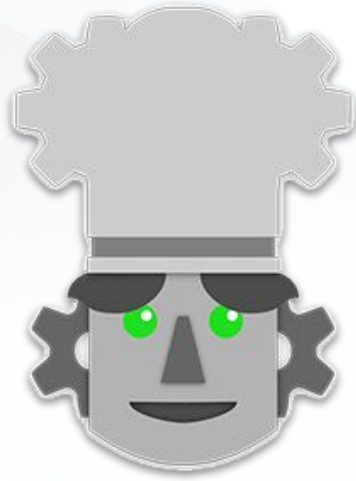
Transmission

This encoding ensures that data remains intact throughout transmission without being modified.

Cookie Encoding

Once encoded, a username or password can be transmitted inside the URL in an ASCII string. This is what the underlying HTTP protocol expects to see.

```
user="eW91YXJldGhld2Vha2VzdGxpbms=";  
security_level=0;  
JSESSIONID=A77E31270D12B1FF01F773F8575B4D64;  
acopendivids=swingset,jotto,phpbb2,redmine;  
acgroupswithpersist=nada
```



We can decode this cookie using
an online tool called CyberChef.

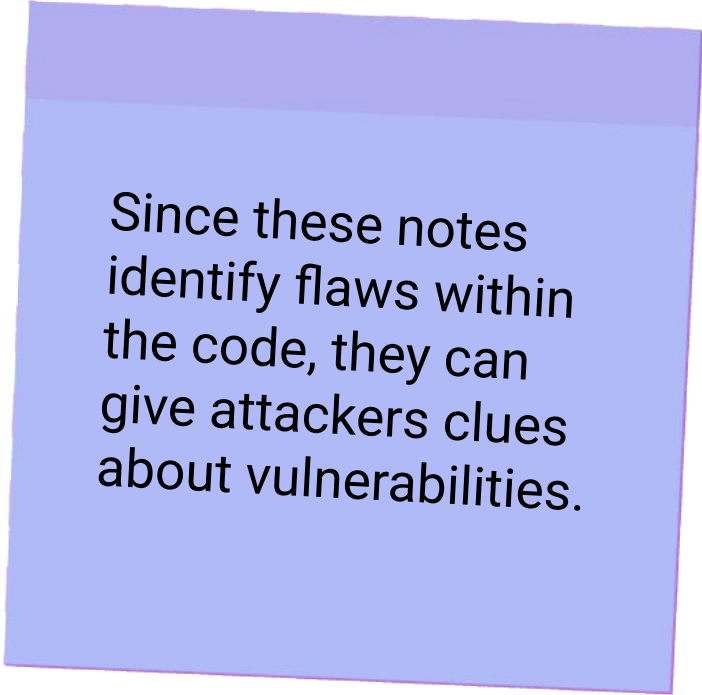


Instructor Demonstration

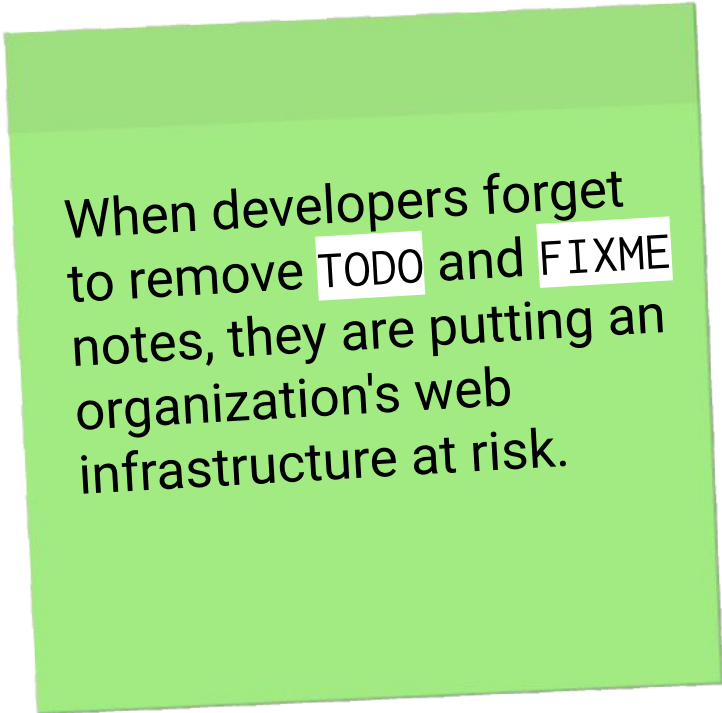
CyberChef

Code Quality

Developers often make notes such as `TODO` and `FIXME` within the application code throughout the software development life cycle (SDLC).



Since these notes identify flaws within the code, they can give attackers clues about vulnerabilities.



When developers forget to remove `TODO` and `FIXME` notes, they are putting an organization's web infrastructure at risk.

Code Quality

...

```
<div id="lessonContent"><form accept-charset="Unknown"
method="POST" name="form" action="attack?Screen=40&menu=700"
enctype="<!-- FIXME: admin:adminpw --><!-- Use Admin to
regenerate..."
</div>
```

...

Code Quality Demo

In this demonstration, we'll perform a code quality attack using the following scenario:



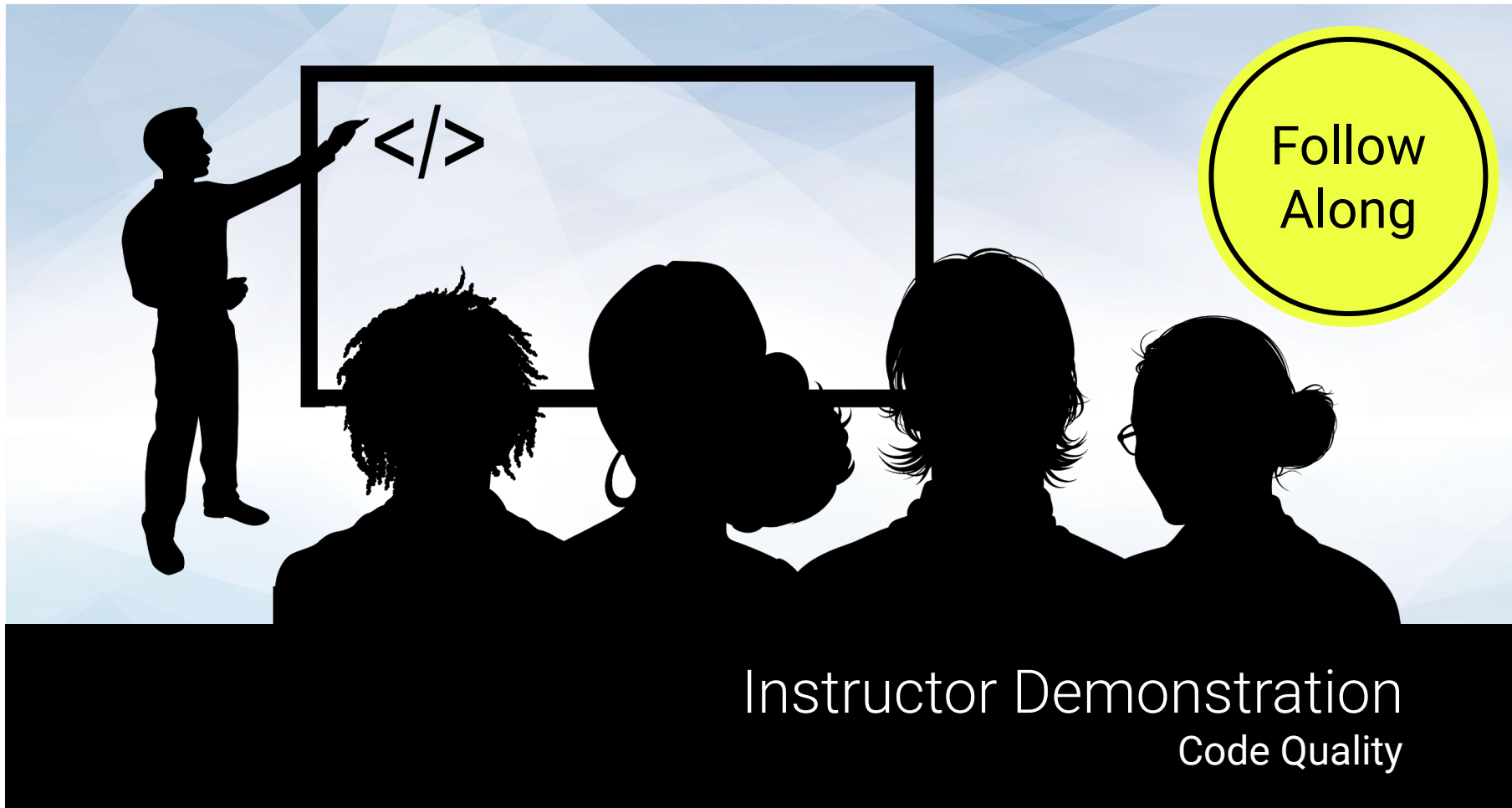
A **hacker** wants to **hack into a web server** to deface an organization's home page.



After doing some research, they discover that the **developers are known to leave evidence of quality flaws** in their web applications.



They will exploit these flaws by **examining the underlying HTML code** for any orphaned web developer notes.



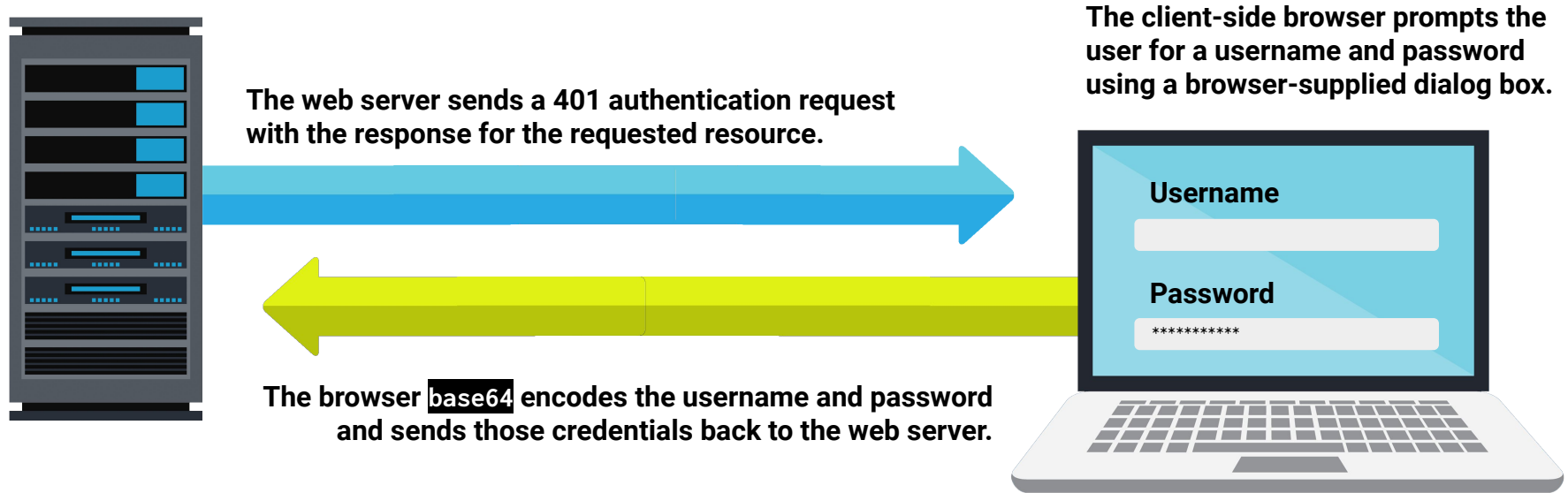
Follow
Along

Instructor Demonstration
Code Quality

Basic Authentication

Basic Authentication

Basic authentication is used to protect server side resources as follows:



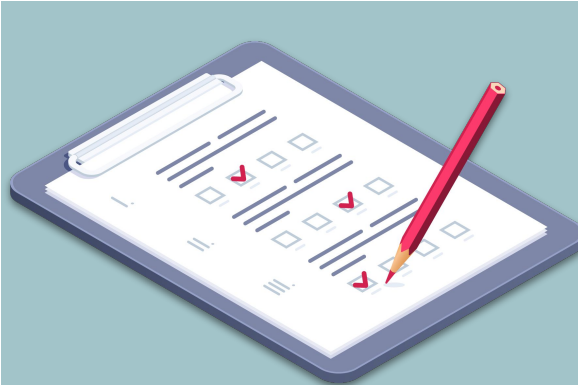
These credentials are automatically reset for each page protected with this mechanism, without requiring the user to enter their credentials again.

Code Quality Demo

In the next demonstration, we'll extract passwords that are sent to the server during a 401 authentication request. We'll use CyberChef to decode them from base64-encoded into clear human-readable text. We'll use the following scenario:



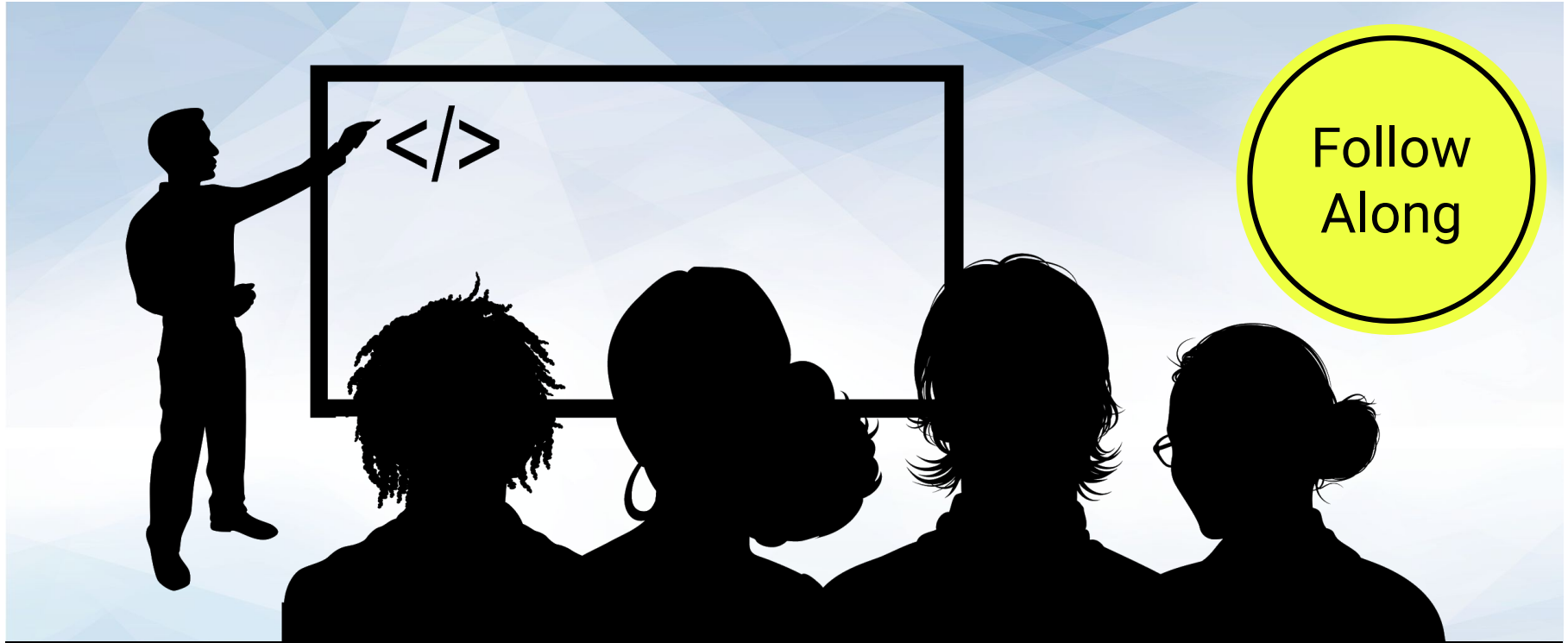
An organization has recently experienced a **large number of unauthorized logins** outside of business hours.



You were hired as a pentester to **identify the origin** of these user authentication issues.



You believe this may be related to **how cookies are handled during user authentication.**



Follow
Along

Instructor Demonstration

Basic Authentication



Activity: The Challenge

In this activity, you will play the role of a pentester hired by a bank to test the security of the bank's authentication scheme, sensitive financial data, and website interface.

This activity is part of this week's homework and requires the use of tools and knowledge covered throughout the week.

Suggested Time:
Until End of Class (*submit with the Unit 15 homework*)

