SOME TERMINOLOGIES...

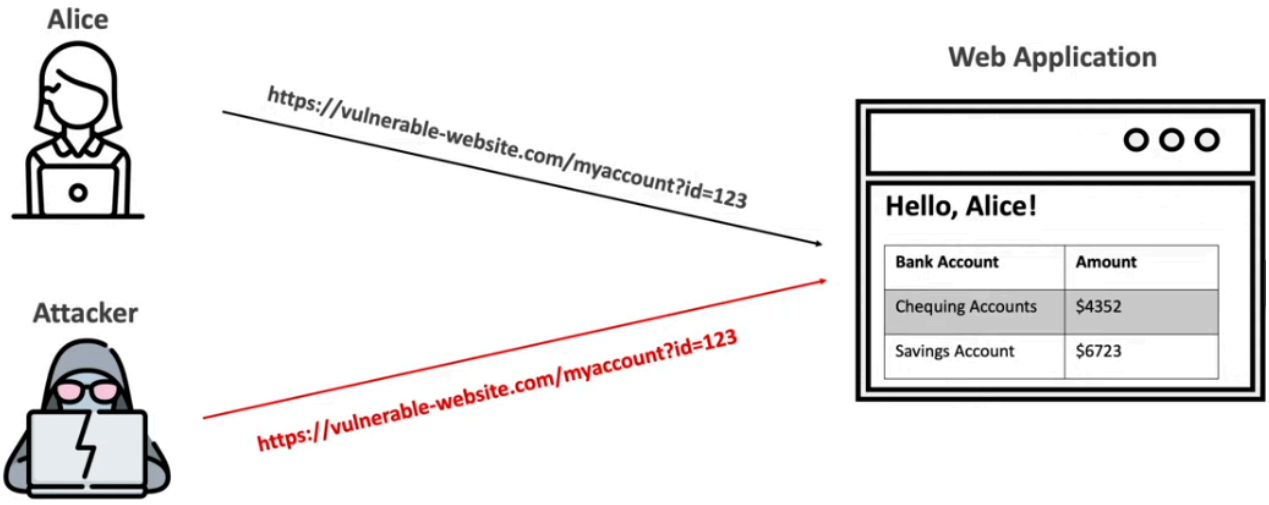
- **Authentication** identifies he user and confirms that they say who they say they are.
- **Session Management** identifies which subsequent HTTP requests are being made by each user. (This generate a unique token that it tied to your username that acts as a short term password and that token verifies your identity every time you make a request).
- **Access Control** determines whether the user is allowed to carry out the action that they are attempting to perform(Your session token is passed to the application and the back-end checks to which user ID is the session token has assigned to and check the access control rules).
  Three different types of access control:
- **Vertical Access Control** is used to restrict access to functions not available for other users in the organization(ex: A regular user cannot perform the admin functions climbing up the ladder).
- **Horizontal Access Control** enables different users to access similar resource types(ex: users on the same privilege level cannot access another users data).
- **Context-Dependent Access Control** restricts access to functionality and resources based on the state of the application or the users interaction with it(ex: When you want to delete a user and you select delete the specific user, you get another window asking to confirm the action you are about to perform, if your select yes and confirm, then only the user gets deleted).

BROKEN ACCESS CONTROL

> Broken Access Control Vulnerabilities arise when users can act outside of their intended permissions This typically leads to sensitive information disclosure, unauthorized access and modification or destruction of data.

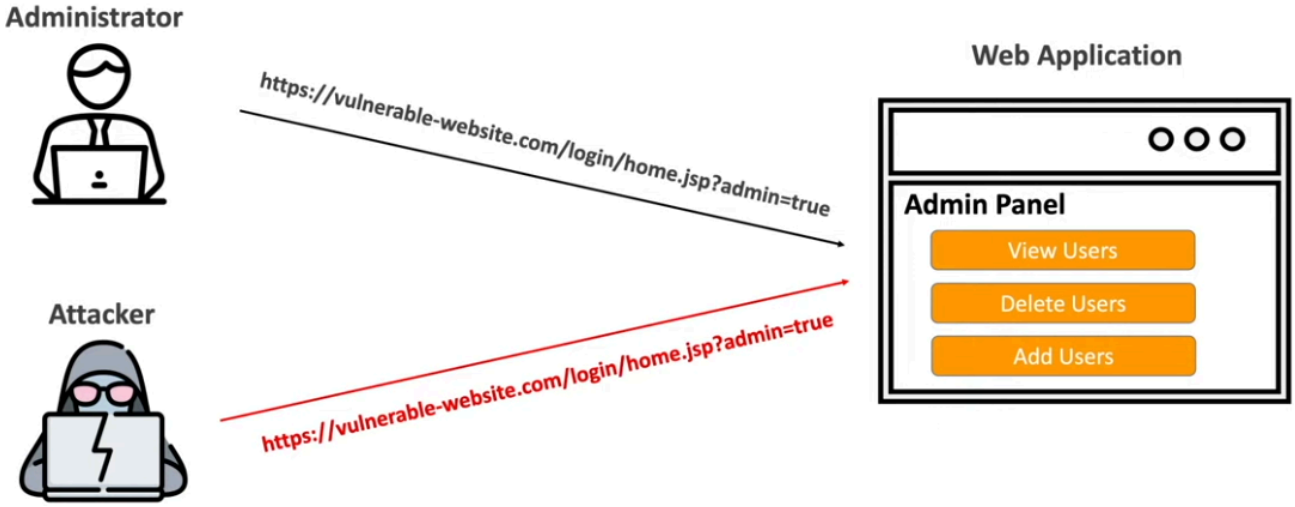
Horizontal Privilege Escalation

- Horizontal privilege escalation occurs when an attacker gains access to resources belonging to another user of the same privilege level.



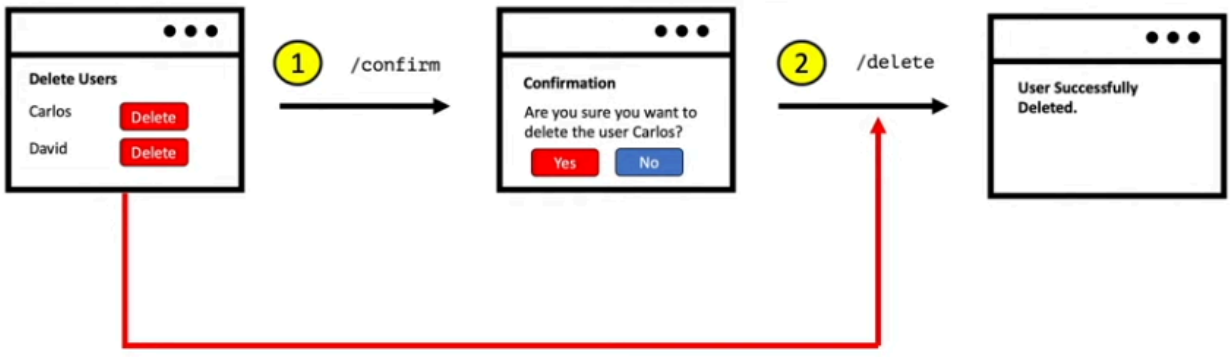
Vertical Privilege Escalation

- Vertical privilege escalation occurs when an attacker gains access to privileged functionality that they are not permitted to access.



Access Control Vulnerabilities in Multi-Step Processes

- These type of vulnerabilities occurs when access control rules are implemented on some of the steps, but ignored on others.



Other Access Control Examples

- Bypassing access control checks by modifying parameters in the URL or HTML page.
- Accessing the API with missing access controls on the POST, PUT and DELETE methods.
- Manipulating metadata, such as replaying or tampering with JSON Web Tokens(JWT) or a cookie.
- Expploiting CORS misconfiguration that allows API access from unauthorized/untrusted origins.
- Force browsing to authenticated pagees as an unauthenticated user.

SPOT THE VULNERABILITY?

```
1 public boolean deleteOrder(Long id) {
2   Order order = orderRepository.getOne(id);
3   if (order == null) {
4     log.info("No found order");
5     return false;
6   }
7   User user = order.getUser();
8   orderRepository.delete(order);
9   log.info("Delete order for user {}", user.getId());
10  return true;
11 }
```

Answer: No verification is performed on line 8 to see if the order has been made by the currently logged-in user.

How do you fix this code?

Answer: Ensure the object id of the order belongs to the user making the request.

Code Source: : https://www.securecodewarrior.com/blog/coders-conquer-security-owasp-top-10-api-series-broken-object-level-authorization

HOW TO FIND ACCESS CONTROL VULNERABILITIES?

- Map the application
  - Identify all instances where the web application appears to be interacting with the underlying operating system.
- Understand how access control is implemented for each privilege level.
- Manipulate paramters that are potentially used to make access control decisions in the back-end.
- Automate testing using extensions like Autorize.

EXPLOITING ACCESS CONTROL VULNERABILITIES

- Depends on the type of Access control vulnerability.
- Usually just a matter of manipulated the vulnerable field / parameter.
  Refer the Labs...

Preventing Access Control Vulnerabilities

- Use a security-centric design where access is verified first and ensure all requests go through an access control check.
- Except for public resources, deny access by default.
- Apply the principal of least privilege throughout the entire application.
- Consider using attribute or deature-based accesses control checks instead of role-based access control.
  - NIST Attribute Based Access Control
  - NIST Special Publication 800-162.
- Access control checks should always be performed on the server side.