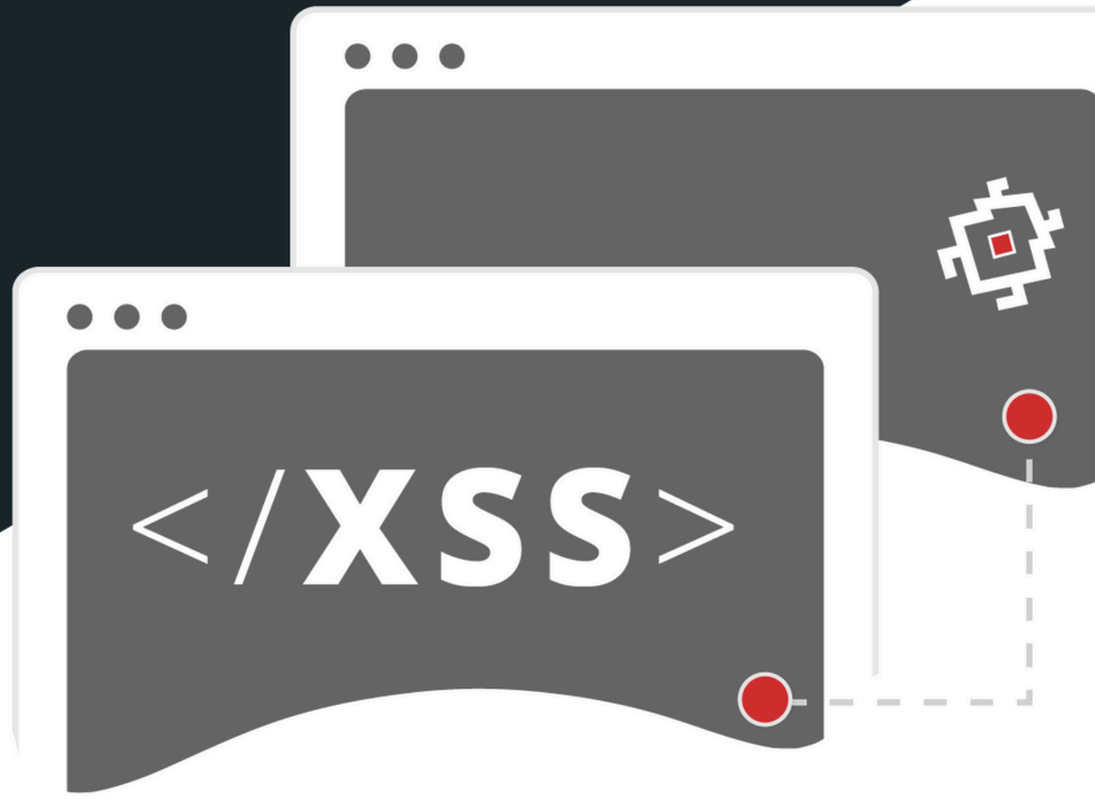
YouCode
POWERED BY SIMPLON

SUCURI GUIDES

What is Cross Site Request Forgery (CSRF)?



Prepared by: BOUCHTA MOHAMED

Introduction

Cross Site Request Forgery (CSRF)

CSRF is a web security vulnerability that tricks a user into performing an unwanted action on a trusted website where they are authenticated.

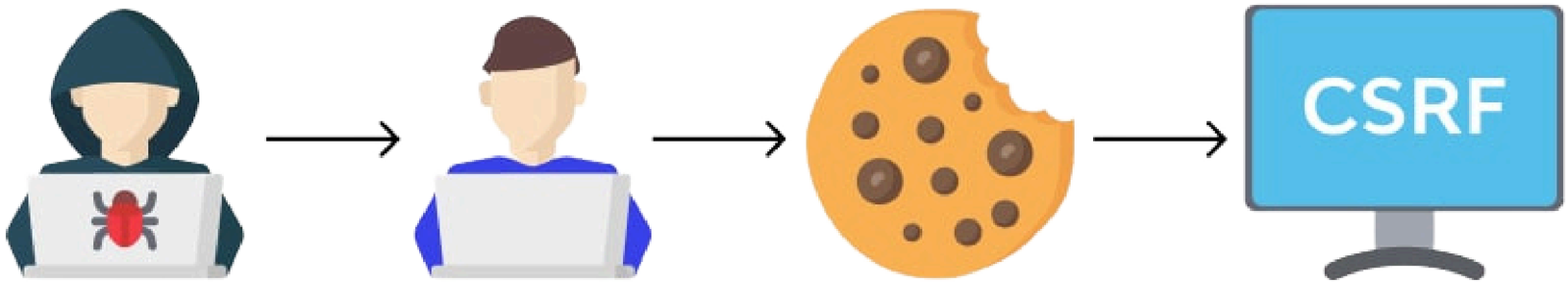
- Key Points:
- Exploits the trust that a website has in the user's browser.
- Commonly targets actions like changing account details, transferring funds, or other sensitive requests.

Why is CSRF Dangerous?

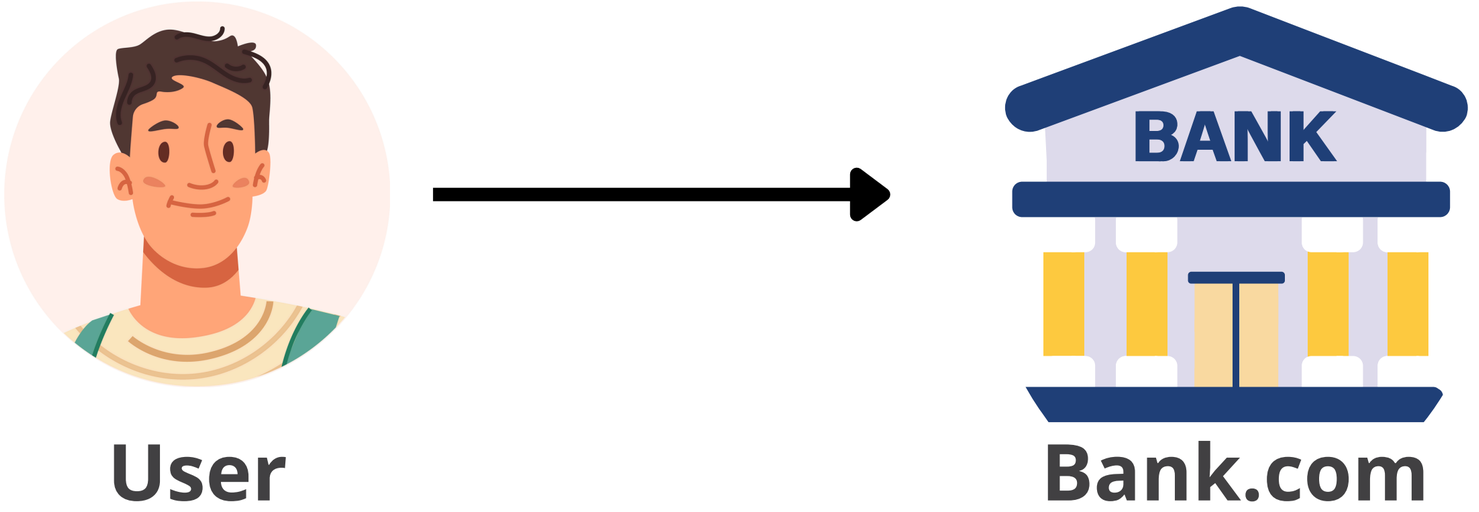
CSRF attacks exploit the implicit trust between users and websites, making them both stealthy and harmful.

1. ***Unauthorized transactions.***
2. ***Data theft or account compromise.***
3. ***Changing user settings (e.g., email, password).***
4. ***Potential escalation to more severe attacks.***

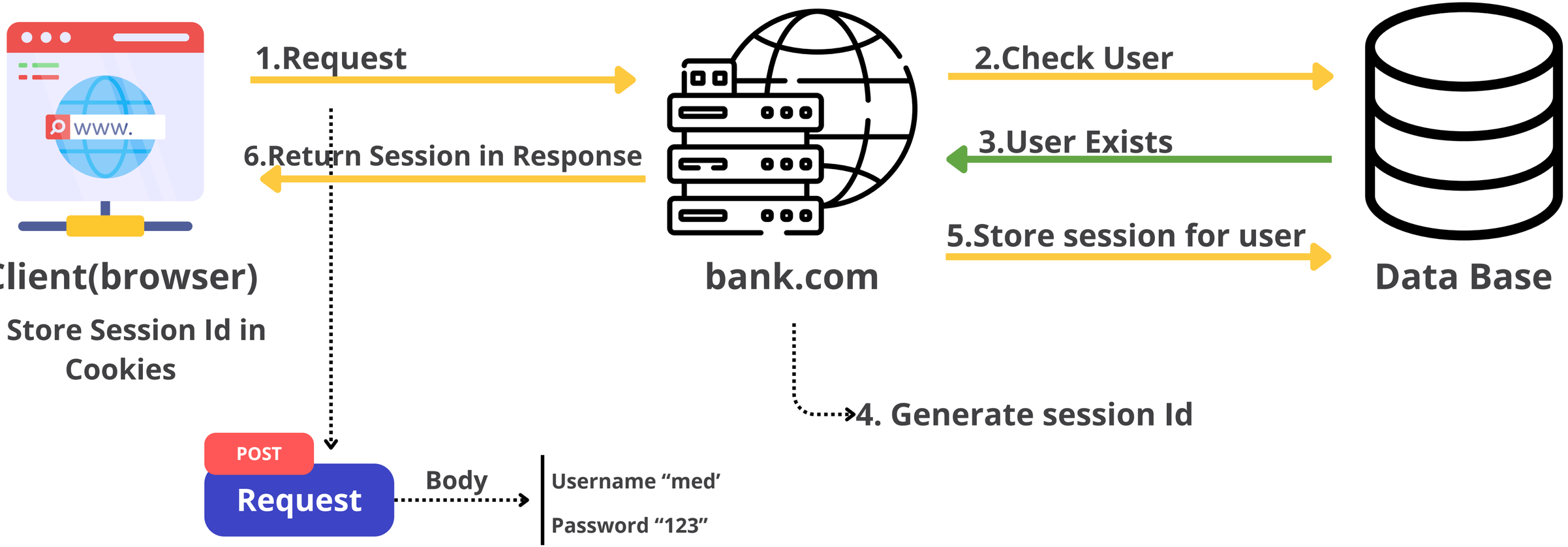
CSRF - Cross site request forgery attack



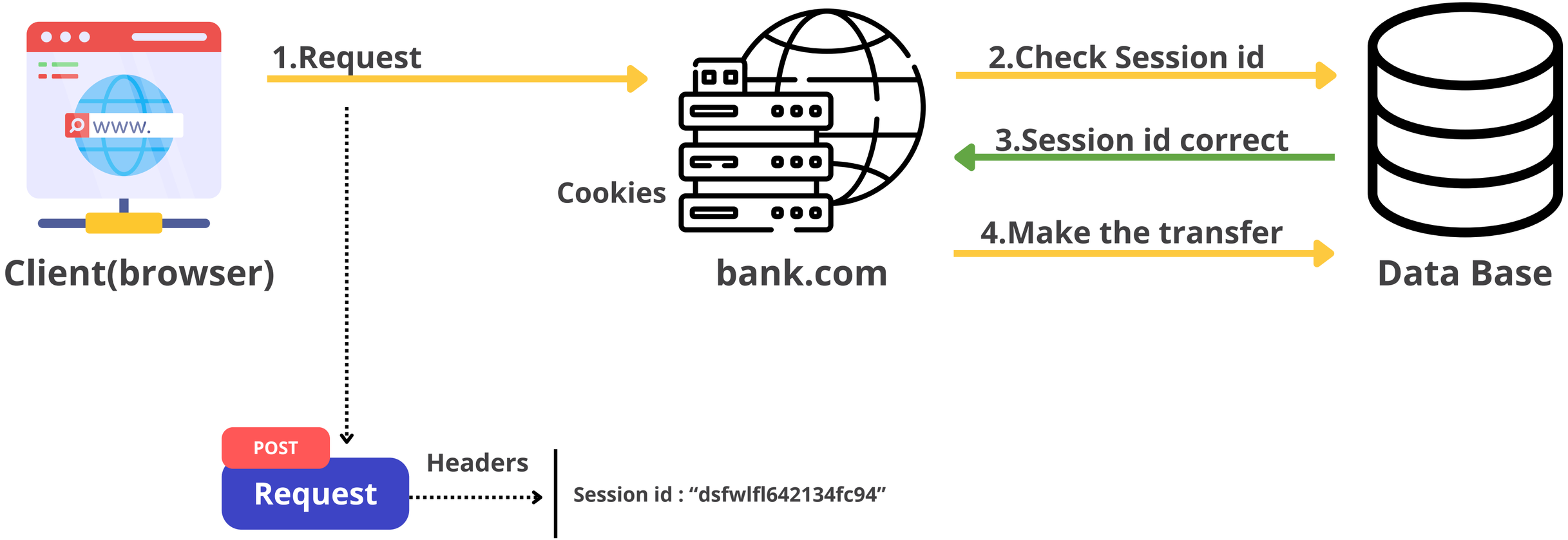
Example :



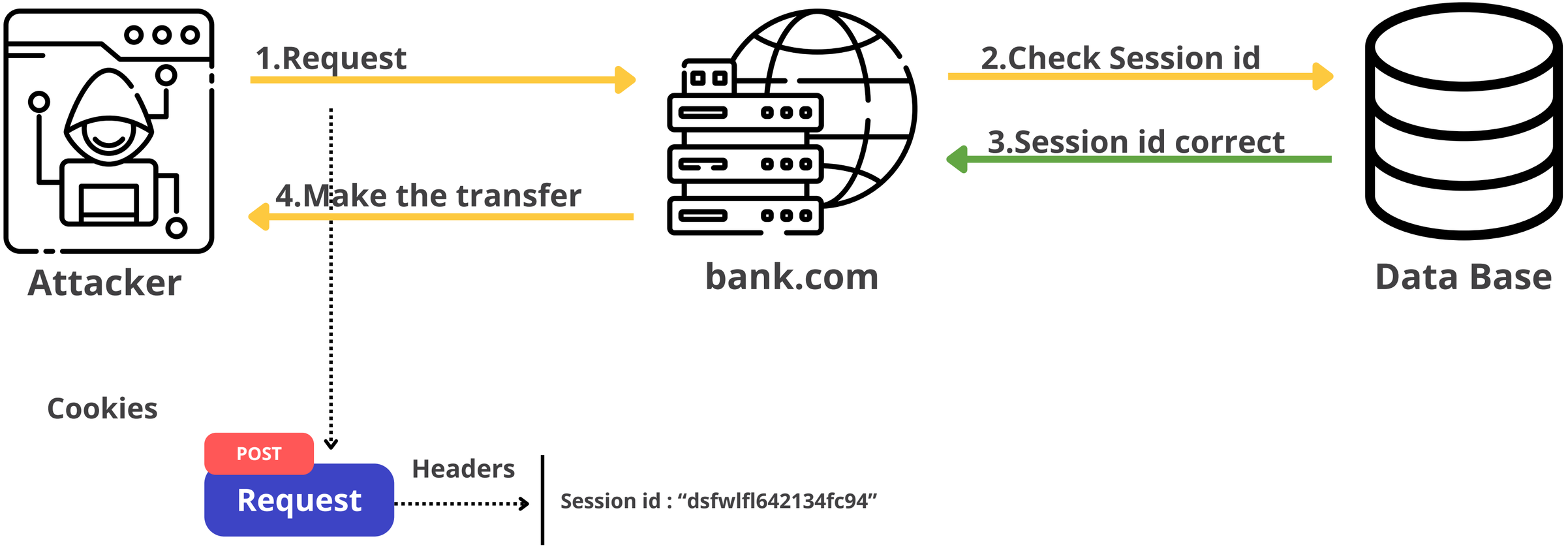
Example :


www.
1.Request
6.Return Session in Response
2.Check User
3.User Exists
5.Store session for user
Client(browser)
bank.com
Data Base
7. Store Session Id in Cookies
4. Generate session Id
POST
Request
Body
Username “med’
Password “123”

Example :


www.
1.Request
Cookies
2.Check Session id
3.Session id correct
4.Make the transfer
Client(browser)
bank.com
Data Base
POST
Request
Headers
Session id : “dsfwlfl642134fc94”

```
// Function to fetch account data
async function fetchAccountData() {
  try {
    const response = await fetch('https://bank.com/myAccount');

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    console.log('Account data:', data);


    sendDataToHackerEmail(data);

    return data;
  } catch (error) {
    console.error('There was a problem fetching the account data:', error);
  }
}

// Call the function
fetchAccountData();
```


1.Request
2.Check Session id
3.Session id correct
4.Make the transfer
Attacker
bank.com
Data Base
Cookies
POST
Request
Headers
Session id : “dsfwlfl642134fc94”

Preventing CSRF Attacks

a. **CSRF Tokens: Include a unique token with every sensitive request to validate the request's origin.**
b. **SameSite Cookies: Configure cookies to be sent only with same-site requests.**
c. **User Authentication Revalidation: Require re-entering credentials for critical actions.**
d. **CORS Policies: Restrict cross-origin requests.**
e. **Check Referrer Header: Ensure requests originate from trusted domains.**

Merci pour votre attention