



JÖNKÖPING UNIVERSITY

*School of Engineering*

---

# SAME-ORIGIN POLICY & CROSS-ORIGIN RESOURCE SHARING

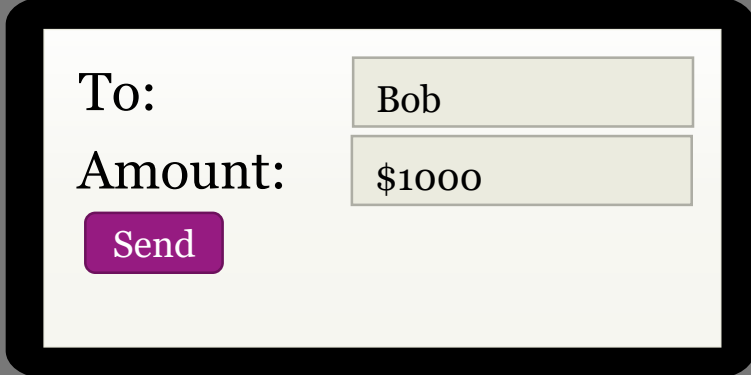
**Peter Larsson-Green**

Jönköping University

Autumn 2018

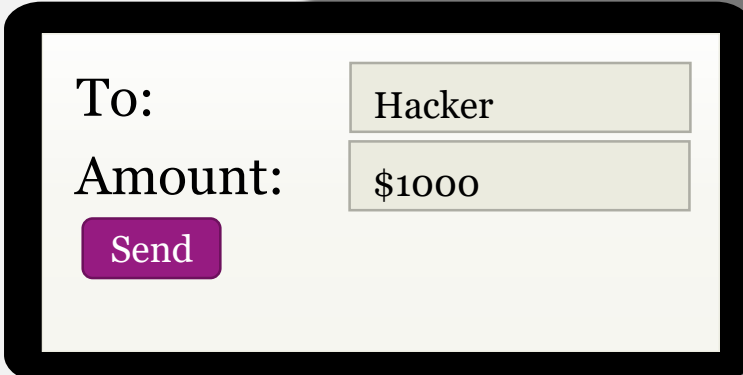
# CROSS-SITE REQUESTS

1. Alice login at bank.com.
  - bank.com creates a session remembering Alice has logged in.
2. Alice visits a hacker's website:



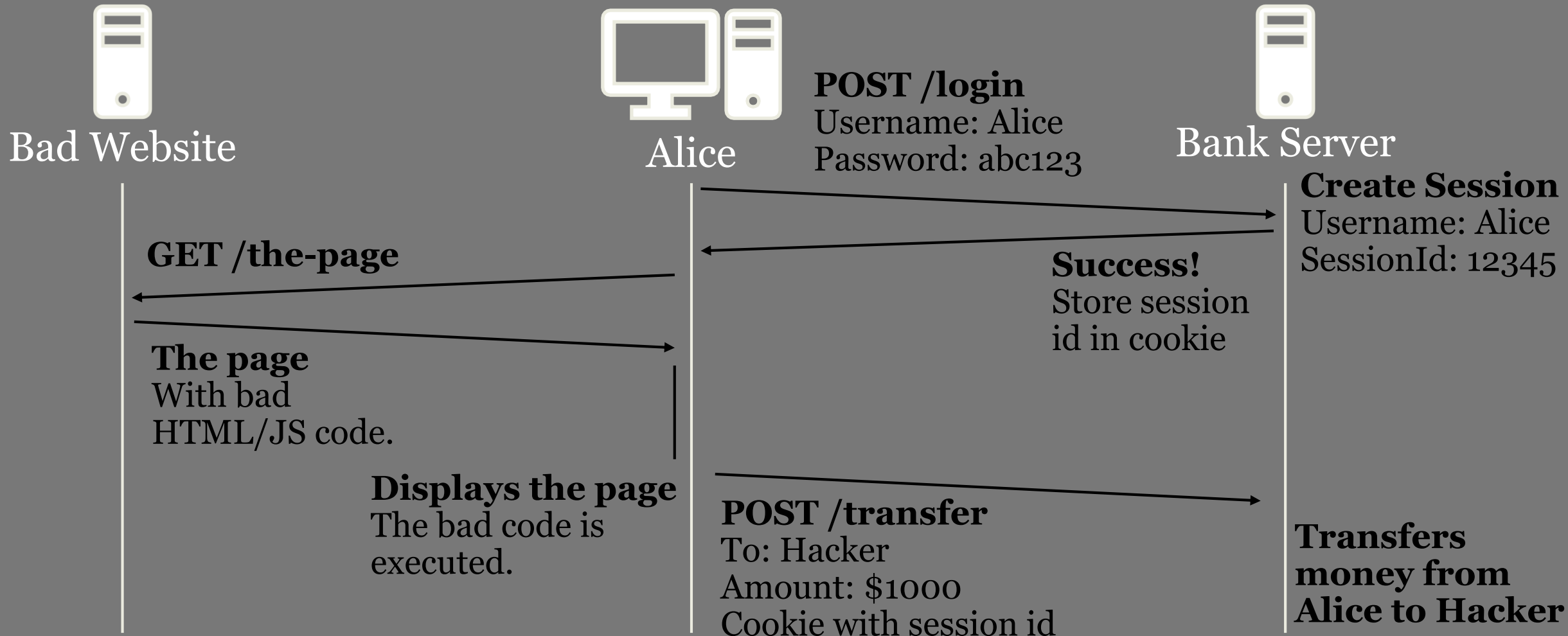
A screenshot of a web browser showing a transfer form. The form has two input fields: 'To:' with the value 'Bob' and 'Amount:' with the value '\$1000'. Below the fields is a purple 'Send' button.

```
<form method="POST" action="http://bank.com/transfer">  
  To: <input name="to" value="Hacker">  
  Amount: <input name="amount" value="$1000">  
  <input type="submit" value="Send">  
</form>  
<script>document.querySelector("form").submit()</script>
```



A screenshot of a web browser showing a transfer form. The form has two input fields: 'To:' with the value 'Hacker' and 'Amount:' with the value '\$1000'. Below the fields is a purple 'Send' button.

# CROSS-SITE REQUESTS



# CROSS-SITE REQUESTS

Is it really this bad?

- Yes!

# THE XMLHTTPREQUEST OBJECT

```
const request = new XMLHttpRequest()
request.open("POST", "http://bank.com/transfer")
request.setRequestHeader(
  "Content-Type",
  "application/x-www-form-urlencoded"
)
request.send("to=Hacker&amount=$1000")
request.addEventListener('load', function(event) {
  const responseBody = request.responseText
})
```

# THE SAME-ORIGIN POLICY

The HTML/JS code on one webpage may send requests to any other website it wants.

- The HTML/JS code may not read the responses.

```
const request = new XMLHttpRequest()
request.open("GET", "http://facebook.com/profile")
request.send("")
request.addEventListener('load', function(event) {
  const responseBody = request.responseText
})
```

# THE SAME-ORIGIN POLICY EXAMPLE

Practical demonstration.



# CROSS-SITE REQUESTS



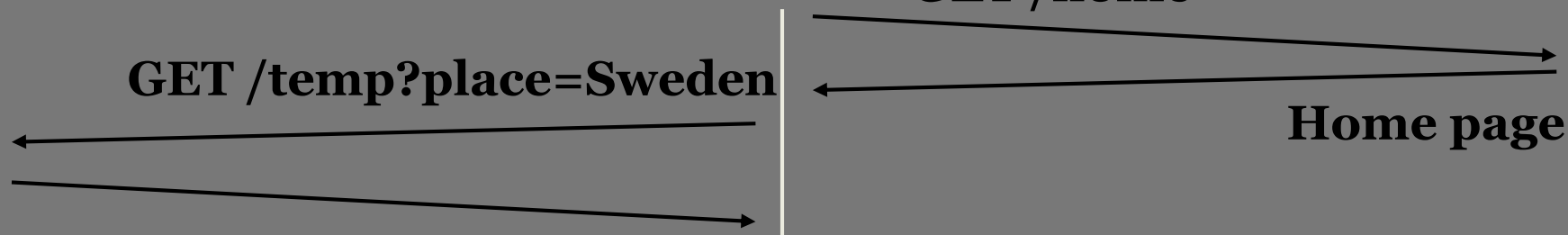
openweathermap.org



User



Your Web App



**GET /temp?place=Sweden**

**GET /home**

**Home page**

**Same-origin policy  
forbids this ☹️**

**Unless  
openweathermap.org  
supports CORS 😊**

# CROSS-SITE REQUESTS

Start: HTML & JS can send HTTP requests through forms

Same-Origin Policy: JS can only read responses to requests sent to the website the JS code comes from

Cross-Origin Resource Sharing: Websites can tell web browser to allow Cross-Site Requests to their website

# CROSS-ORIGIN RESOURCE SHARING

```
Access-Control-Allow-Origin: *
```

```
Access-Control-Allow-Origin: http://test.com
```

# CORS EXAMPLE

Practical demonstration.

# CORS DETAILS

## Simple Requests

- Method is:
  - GET
  - POST
  - HEAD
- Custom headers are only:
  - Accept
  - Accept-Language
  - Content-Type:
    - `application/x-www-form-urlencoded`
    - `multipart/form-data`
    - `text/plain`

## Preflighted Requests

- All other requests.

# CROSS-SITE REQUESTS



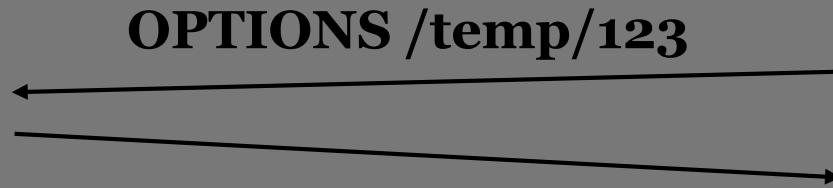
openweathermap.org



User



Your Web App



**OPTIONS /temp/123**

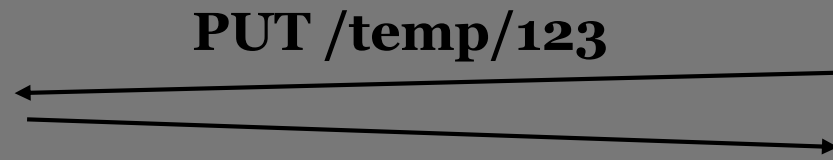


**GET /home**



**Home page**

```
200 OK
Access-Control-Allow-Origin: http://test.com
Access-Control-Allow-Methods: GET, PUT, POST, DELETE
Access-Control-Allow-Headers: Content-Type
```



**PUT /temp/123**