# CS 253: Web Security

## Cookies, Sessions

# Recall: Cookies

# Server sends a cookie with a response

Set-Cookie: theme=dark;

Header Name

Cookie Name

Cookie Value

# Client sends a cookie with a request

Cookie: theme=dark;

Header Name | Cookie Name | Cookie Value

# Sessions

- **Cookies** are used by the server to implement **sessions**
- **Goal:** Server keeps a set of data related to a user's current "browsing session"
- Examples
  - Logins
  - Shopping carts
  - User tracking

# Ambient authority

- **Access control** - Regulate who can view resources or take actions
- **Ambient authority** - Access control based on a **global and persistent property** of the requester
  - The alternative is explicit authorization **valid only for a specific action**
- There are four types of ambient authority on the web
  - **Cookies** - most common, most versatile method
  - **IP checking** - used at Stanford for library resources
  - **Built-in HTTP authentication** - rarely used
  - **Client certificates** - rarely used

# Demos: Sessions

# Demo: Insecure Session 1

```
<!doctype html>
<html lang='en'>
  <head>
    <meta charset='utf-8' />
    <title>My Cool Site</title>
  </head>
  <body>
    <h1>Bank login:</h1>
    <form method='POST' action='/login'>
      Username:
      <input name='username' />
      <br />
      Password:
      <input name='password' type='password' />
      <br />
      <input type='submit' value='Login' />
    </form>
  </body>
</html>
```

# Demo: Insecure Session 1

```
const express = require('express')
const { createReadStream } = require('fs')
const cookieParser = require('cookie-parser')

const app = express()
app.use(cookieParser())
app.use(express.urlencoded({ extended: false }))

// Routes go here!

app.listen(8000)
```

# Demo: Insecure Session 1

```
const USERS = { alice: 'password', bob: '50505' }
const BALANCES = { alice: 500, bob: 100 }

app.get('/', (req, res) => {
  const { username } = req.cookies
  if (username) {
    res.send(`
      <h1>Welcome, ${username}</h1>
      <p>Your balance is $${BALANCES[username]}</p>
    `)
  } else {
    createReadStream('index.html').pipe(res)
  }
})
```

# Demo: Insecure Session 1

```
app.post('/login', (req, res) => {
  const { username } = req.body
  const { password } = req.body
  if (password === USERS[username]) {
    res.cookie('username', username)
    res.redirect('/')
  } else {
    res.send('fail!')
  }
})

app.get('/logout', (req, res) => {
  res.clearCookie('username')
  res.redirect('/')
})
```

*First HTTP request:*

```
POST /login HTTP/1.1
Host: example.com

username=alice&password=password
```

*HTTP response:*

```
HTTP/1.1 200 OK
Set-Cookie: username=alice
Date: Tue, 24 Sep 2019 20:30:00 GMT

<!DOCTYPE html ...
```

*All future HTTP requests:*

```
GET /page.html HTTP/1.1
Host: example.com
Cookie: username=alice;
```

# END FOR NOW