

# Security Issues in Web Programming (Part 2)

Copyright © 2025 by  
Robert M. Dondero, Ph.D.  
Princeton University

# Objectives

- We will cover:
  - Some web programming security attacks
  - Some ways to thwart them

# Agenda

- **Authentication & authorization**
- Cookie forgery attacks

# Authentication & Authorization

- **Problem:**

- PennyAdmin doesn't do **authentication** of its users
  - It doesn't know who its users are
- PennyAdmin doesn't do **authorization** of its users
  - Only certain users – the Penny bookstore owners – should be allowed to change the PennyAdmin DB
  - PennyAdmin doesn't enforce that restriction

# Authentication & Authorization

- ***Authenticate***

- Make sure the user is *authentic*
- Make sure the user is who he/she claims to be

- ***Authorize***

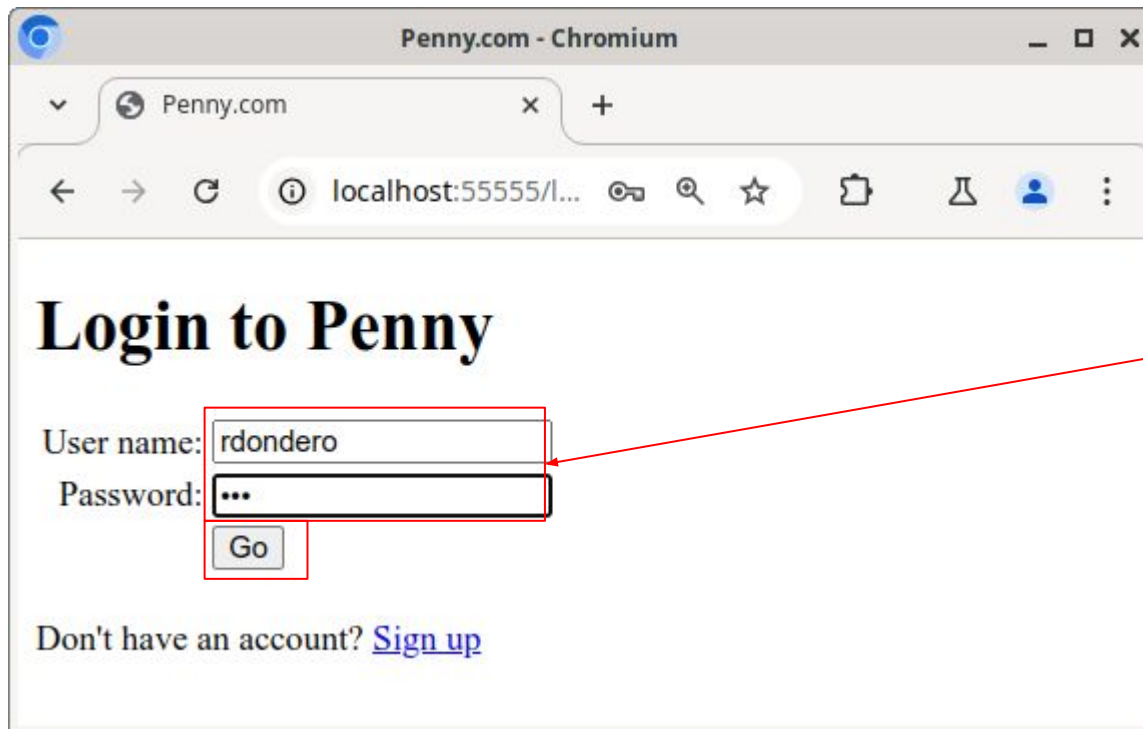
- Having authenticated the user, make sure the user has *authority* to use the app in the way the user wants

# Authentication & Authorization

- Suppose this is what we want...
  - **Show** books:
    - User must be **authenticated**
  - **Add & delete** books:
    - User must be **authenticated**
    - User must be **authorized**
      - Only rdondero and bwk have authority

# Authentication & Authorization

- See **PennyAdmin06Auth** app

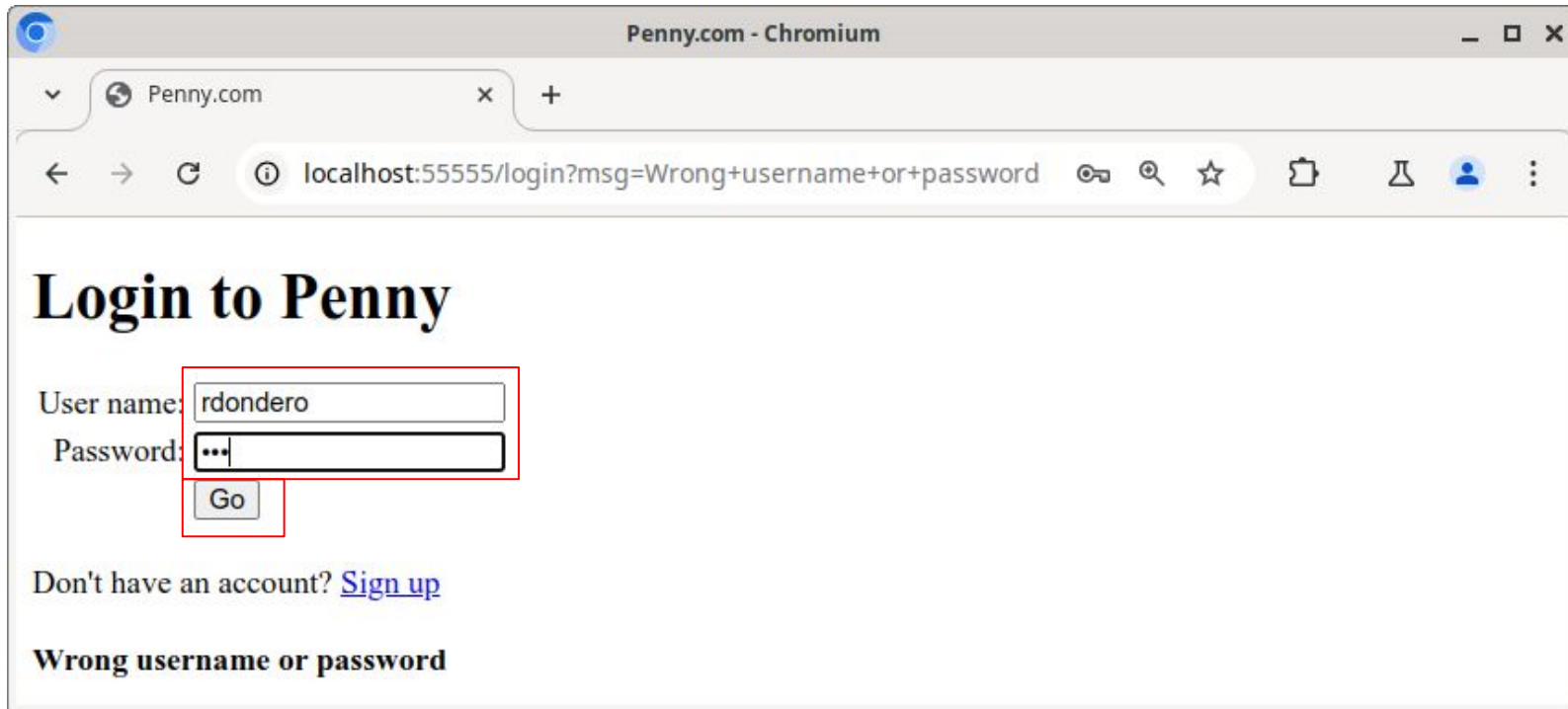


Login  
page

Wrong  
password

# Authentication & Authorization

- See **PennyAdmin06Auth** app

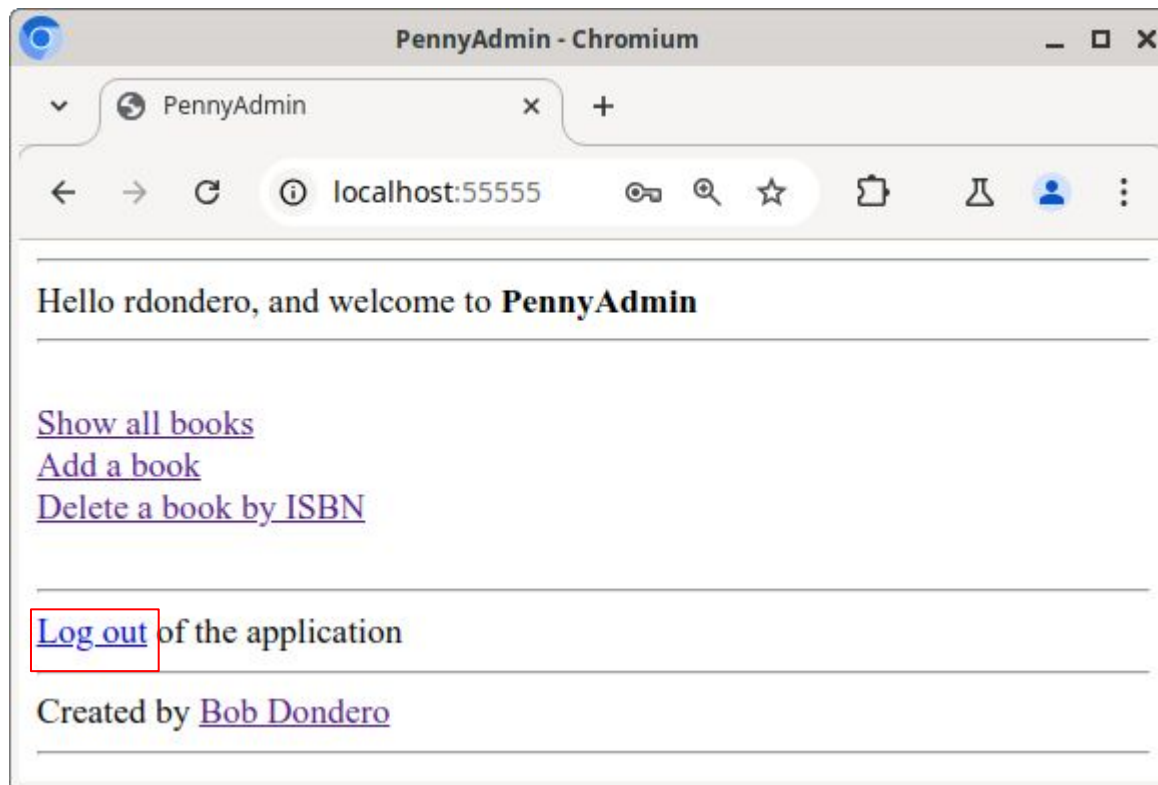


Login  
page



# Authentication & Authorization

- See **PennyAdmin06Auth** app



Index  
page

Username  
in  
header

Log out  
link in  
footer

# Authentication & Authorization

- See **PennyAdmin06Auth** app



Logged out  
page

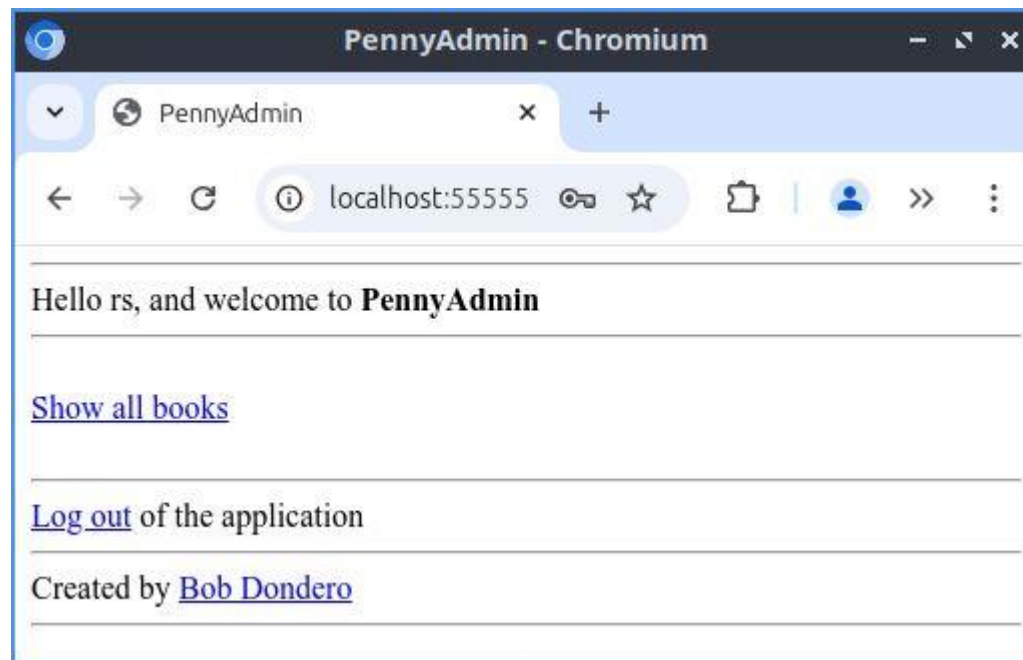
# Authentication & Authorization

- See **PennyAdmin06Auth** app



# Authentication & Authorization

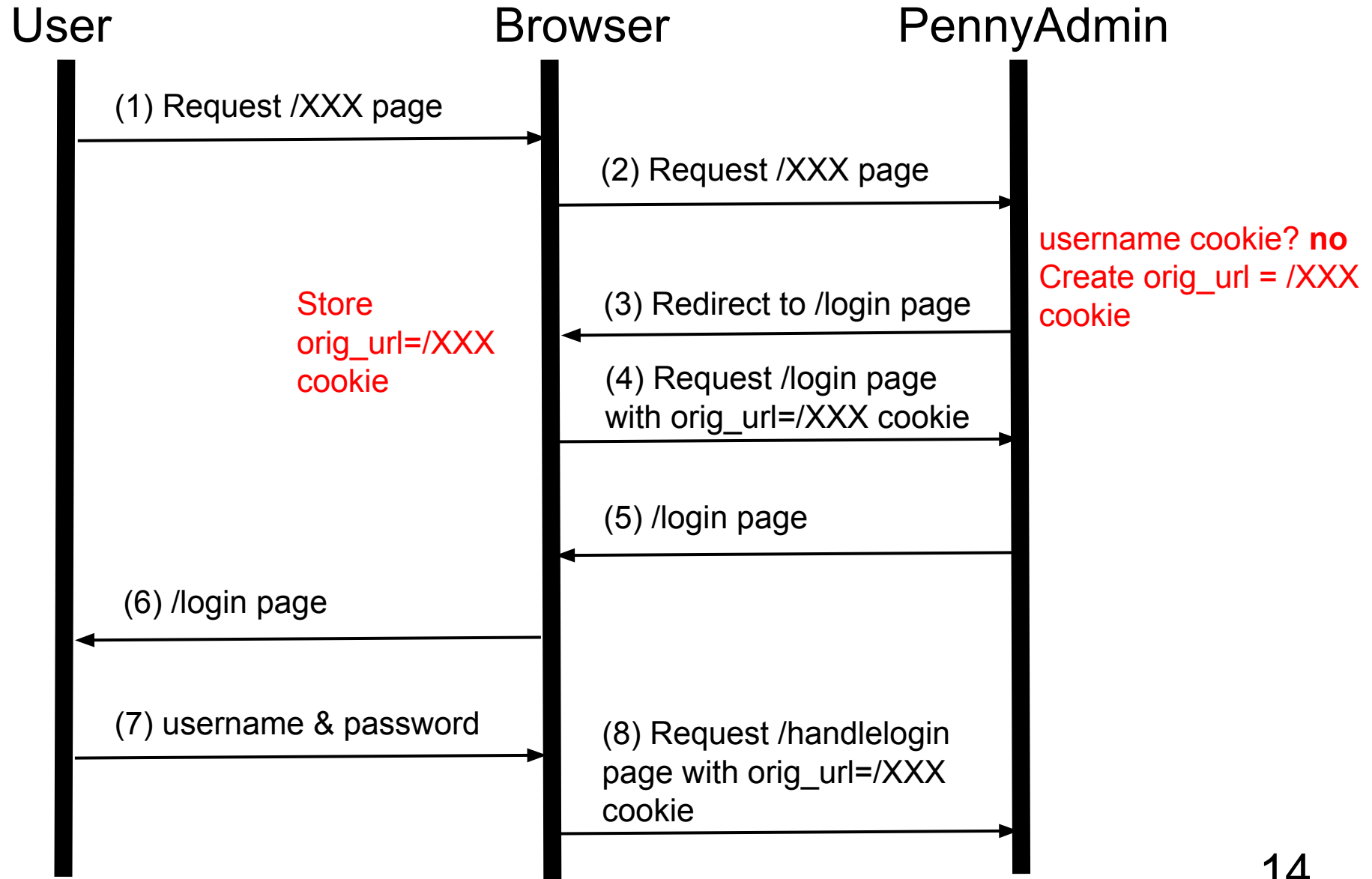
- See **PennyAdmin06Auth** app



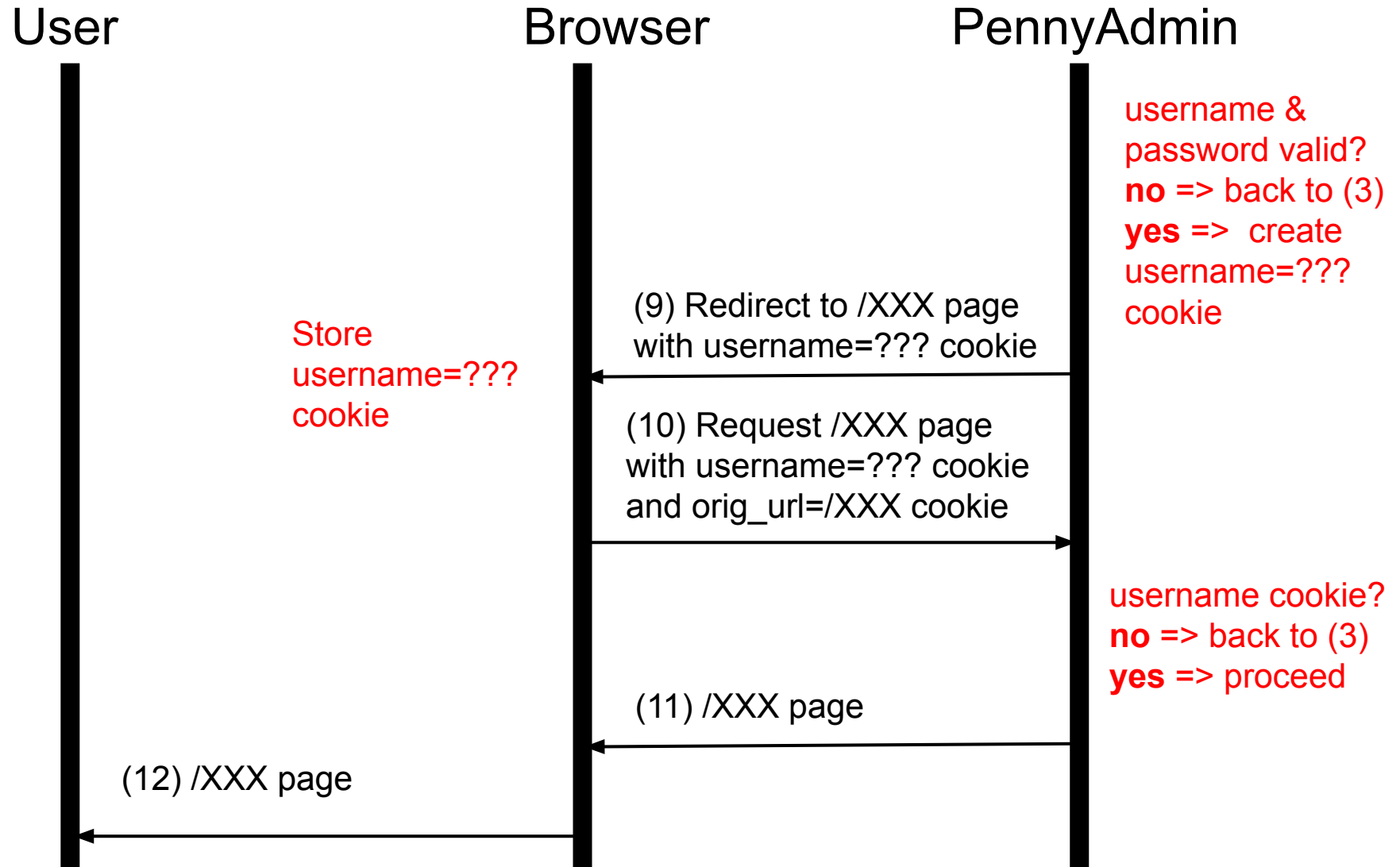
# Authentication & Authorization

- See **PennyAdmin06Auth** app
  - Authentication
    - The flow...

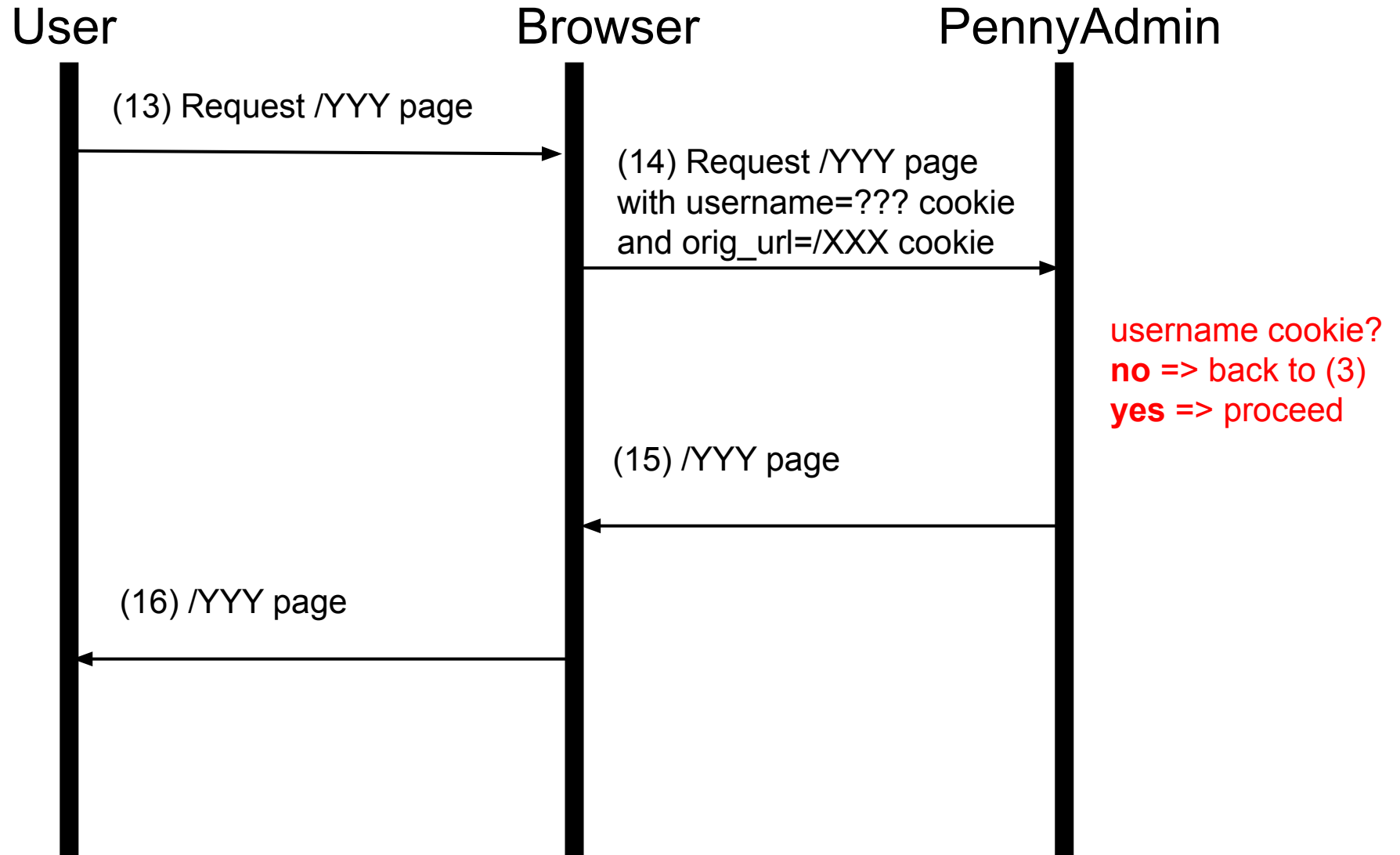
# Authentication & Authorization



# Authentication & Authorization



# Authentication & Authorization





# Authentication & Authorization

- See **PennyAdmin06Auth** app
  - Authorization
    - Handled separately from authentication
    - Straightforward...

# Authentication & Authorization

- See **PennyAdmin06Auth** app
  - runserver.py
  - **penny.sql**, penny.sqlite
  - **database.py**
  - **header.html**, **footer.html**
  - **index.html**, show.html,
  - add.html, delete.html, reportresults.html
  - **login.html**, **signup.html**, **loggedout.html**
  - **top.py**, **penny.py**, **auth.py**

# Authentication & Authorization

- PennyAdmin06Auth is realistic
  - Flow is similar with CAS and GoogleAuth
- But typically would be enhanced:
  - Password changing (easy)
  - Existing user un-registration (easy)
  - Email verification (hard)
  - Forgotten password recovery (hard)
  - Two-factor authentication (hard)
    - Email, phone call, text message, authenticator app

# Agenda

- Authentication & authorization
- **Cookie forgery attacks**

# Cookie Forgery Attacks

- **Problem:**
  - In PennyAdmin app, an attacker can forge the username cookie

# Cookie Forgery Attacks

- Recall **PennyAdmin06Auth** app
  - Example:
    - Attacker runs **cookieforgeryattack.py**
      - Sends forged username cookie to PennyAdmin app

# Cookie Forgery Attacks

- Recall PennyAdmin06Auth app
  - Example (cont.):

```
$ python cookieforgeryattack.py localhost 55555
HTTP/1.1 200 OK
Server: Werkzeug/3.0.3 Python/3.12.3
Date: Sat, 31 Aug 2024 18:56:30 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 780
Connection: close

<!DOCTYPE html>
<html>
  <head>
    <title>PennyAdmin</title>
  </head>
  <body>
    <hr>Hello rdondero, and welcome to <strong>PennyAdmin</strong><hr>
    <h1>Show All Books</h1>
```

App  
considers  
user  
to be  
logged  
in  
as  
rdondero

# Cookie Forgery Attacks

- Recall **PennyAdmin06Auth** app
  - Example (cont.):

```
123:
<strong>Kernighan</strong>:
The Practice of Programming<br>

234:
<strong>Kernighan</strong>:
The C Programming Language<br>

345:
<strong>Sedgewick</strong>:
Algorithms in C<br>

<br>
<a href="/index">Return to home page</a>
<br>
<hr>
<a href="logout">Log out</a> of the application</br>
<hr>
Created by <a href="https://www.cs.princeton.edu/~rdontero">
Bob Dondero</a>
<hr>
  </body>
</html>
```

App  
considers  
user  
to be  
logged  
in  
as  
rdontero



# Cookie Forgery Attacks

- **Solution 1:**

- ***Cookie encryption***

- Before server sends username cookie...
      - Server uses a ***secret key*** to **encrypt** the value of the username cookie
    - After server receives username cookie...
      - Server uses the same ***secret key*** to **decrypt** the value of the username cookie
      - Decryption fails => forgery

# Aside: Secret Keys

- **Question:** How to generate a secret key?
- **One answer:**

```
$ python
Python 3.12.3 (main, Jul 31 2024, 17:43:48)
[GCC 13.2.0] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> import os
>>> os.urandom(12).hex()
'████████████████████'
>>> quit()
$
```

Use  
this  
as  
secret  
key



# Aside: Secret Keys

- **Question:** Where to store secret keys?
- **Possible answers:**
  - Source code files? **No**
    - Attacker might gain access to GitHub repo
  - Some other file? **Maybe**
    - But must make sure the file is not in GitHub repo
  - Environment variables? **Yes**
    - The common way

# Aside: Secret Keys

To run subsequent versions of PennyAdmin:

Mac & Linux:

```
$ export APP_SECRET_KEY=yourappsecretkey  
$ python runserver.py 55555
```

MS Windows:

```
$ set APP_SECRET_KEY=yourappsecretkey  
$ python runserver.py 55555
```

Or use the `python_dotenv` package

```
$ cat .env  
APP_SECRET_KEY=yourappsecretkey  
$
```

# Cookie Forgery Attacks

- See **PennyAdmin07Encrypt** app
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - header.html, footer.html
  - index.html, show.html,
  - add.html, delete.html, reportresults.html
  - login.html, signup.html, loggedout.html
  - top.py, penny.py, **auth.py**

# Cookie Forgery Attacks

- See **PennyAdmin07Encrypt** app
  - Example:
    - Attacker runs **cookieforgeryattack.py**
      - Sends forged username cookie to PennyAdmin app

# Cookie Forgery Attacks

- See **PennyAdmin07Encrypt** app
  - Example (cont.):

```
$ python cookieforgeryattack.py localhost 55555
HTTP/1.1 302 FOUND
Server: Werkzeug/3.0.3 Python/3.12.3
Date: Sat, 31 Aug 2024 19:00:42 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 199
Location: /login
Set-Cookie: original_url=http://localhost/show; Path=/
Connection: close

<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL:
<a href="/login">/login</a>. If not, click the link.
$
```

App  
rejects  
username,  
redirects  
to  
login  
page

# Cookie Forgery Attacks

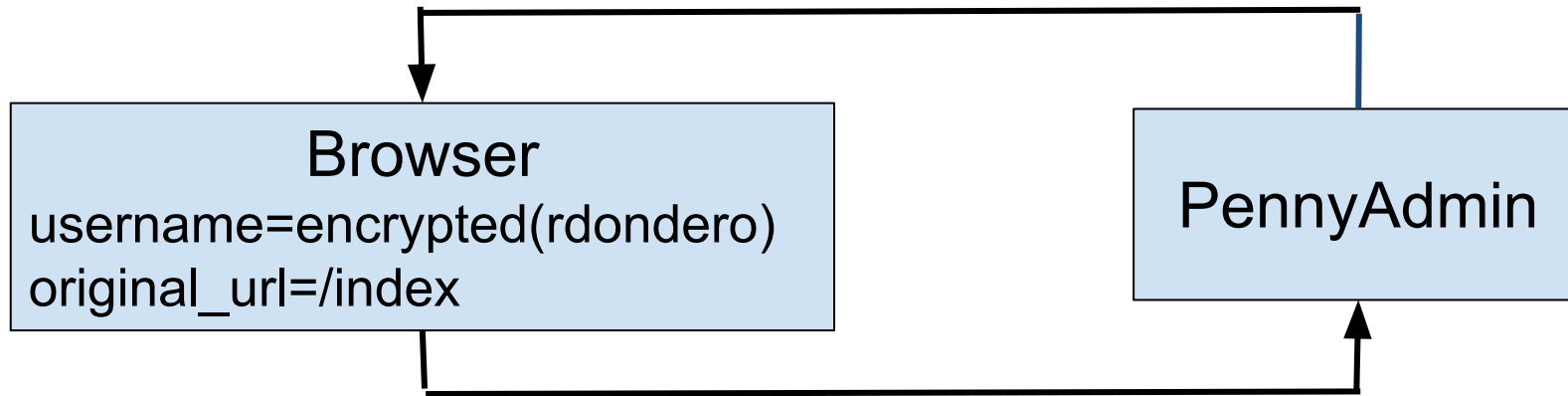
- **Solution 2:**
  - *Sessions*



# Cookie Forgery Attacks

Without sessions:

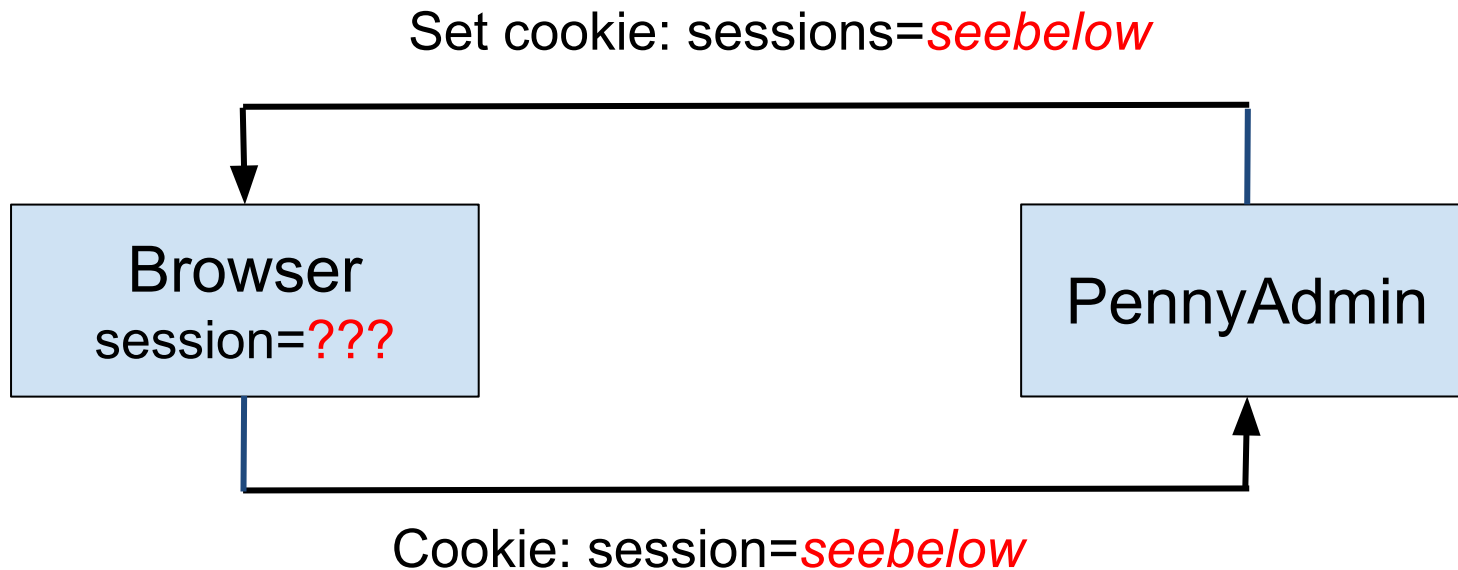
Set cookie: username=encrypted(rdondero)  
Set cookie: original\_url=/index



Cookie: username=encrypted(rdondero)  
Cookie: original\_url=/index

# Cookie Forgery Attacks

With sessions:



*eyJvcmlnaW5hbF91cmwiOiJodHRwOi8vbG9jYWxob3N0OjU1NTU1LyIsInVzZ  
XJuYW1lIjoicmRvbmRlcm8ifQ.YsDrnQ.fc45qAmc0Vk6pAr32bQnogTtx1c*

Using my secret key, decrypts to:

original\_url=/index; username=rdontero;

# Cookie Forgery Attacks

- See **PennyAdmin08Session** app
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - header.html, footer.html
  - index.html, show.html,
  - add.html, delete.html, reportresults.html
  - login.html, signup.html, loggedout.html
  - **top.py**, penny.py, **auth.py**

# Cookie Forgery Attacks

- See **PennyAdmin08Session** app
  - Example:
    - Attacker runs **cookieforgeryattack.py**
      - Sends forged session cookie to PennyAdmin app

# Cookie Forgery Attacks

- See **PennyAdmin08Session** app
  - Example (cont.):

```
$ python cookieforgeryattack.py localhost 55555
HTTP/1.1 302 FOUND
Server: Werkzeug/3.0.3 Python/3.12.3
Date: Sat, 31 Aug 2024 19:03:41 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 199
Location: /login
Vary: Cookie
Set-Cookie:
session=eyJvcmlnaW5hbF91cmwiOiJodHRwOi8vbG9jYXRob3N0L3Nob3cif
Q.ZtNpDQ.24_ouOA3YNt2oeAz9gEiz_sGZf0; HttpOnly; Path=/
Connection: close

<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL:
<a href="/login">/login</a>. If not, click the link.
```

App  
rejects  
username,  
redirects  
to  
login  
page

# Cookie Forgery Attacks

- Q: Project concern?
- A: **Yes!!!**
  - Iff your project app stores, in cookies, data that must not be forged

# Summary

- We have covered:
  - Authentication & authorization
  - Secret keys
  - Cookie forgery attacks