

Security Issues in Web Programming (Part 4)

Copyright © 2026 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

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

Agenda

- **Data storage attacks**
- Data comm attacks
- Third-party authentication (briefly)
 - CAS authentication
 - Microsoft Entra ID authentication
 - Google authentication

Data Storage Attacks

- **Problem:**
 - PennyAdmin app stores passwords in DB
 - If attacker gains access to DB
 - ... Then attacker learns passwords

Data Storage Attacks

- **Insight:**

- PennyAdmin doesn't really need to store passwords
- It's sufficient for PennyAdmin to know if a given password is correct

Data Storage Attacks

- **Solution:**
 - Store *password hash codes* instead of passwords
 - `hash_code = hash(password)`

Data Storage Attacks

- Which hash function?
 - *md5*?
 - `hash_code = md5(password)`
 - No! See <https://en.wikipedia.org/wiki/MD5>
 - *sha256*?
 - `hash_code = sha256(password)`
 - Yes! See <https://en.wikipedia.org/wiki/SHA-2>

Data Storage Attacks

- See **PennyAdmin7aHash** 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
 - penny.py, **auth.py**

Data Storage Attacks

- **Problem:**

- PennyAdmin app stores password hash codes in DB
- If attacker gains access to DB, then...
 - Attacker learns password hash codes
- If a password is common, then...
 - Attacker might find password hash code in a *rainbow table* (huge malevolent list of hash codes), and thereby learn the password

Data Storage Attacks

- **Example:**

- Password:

- xxx

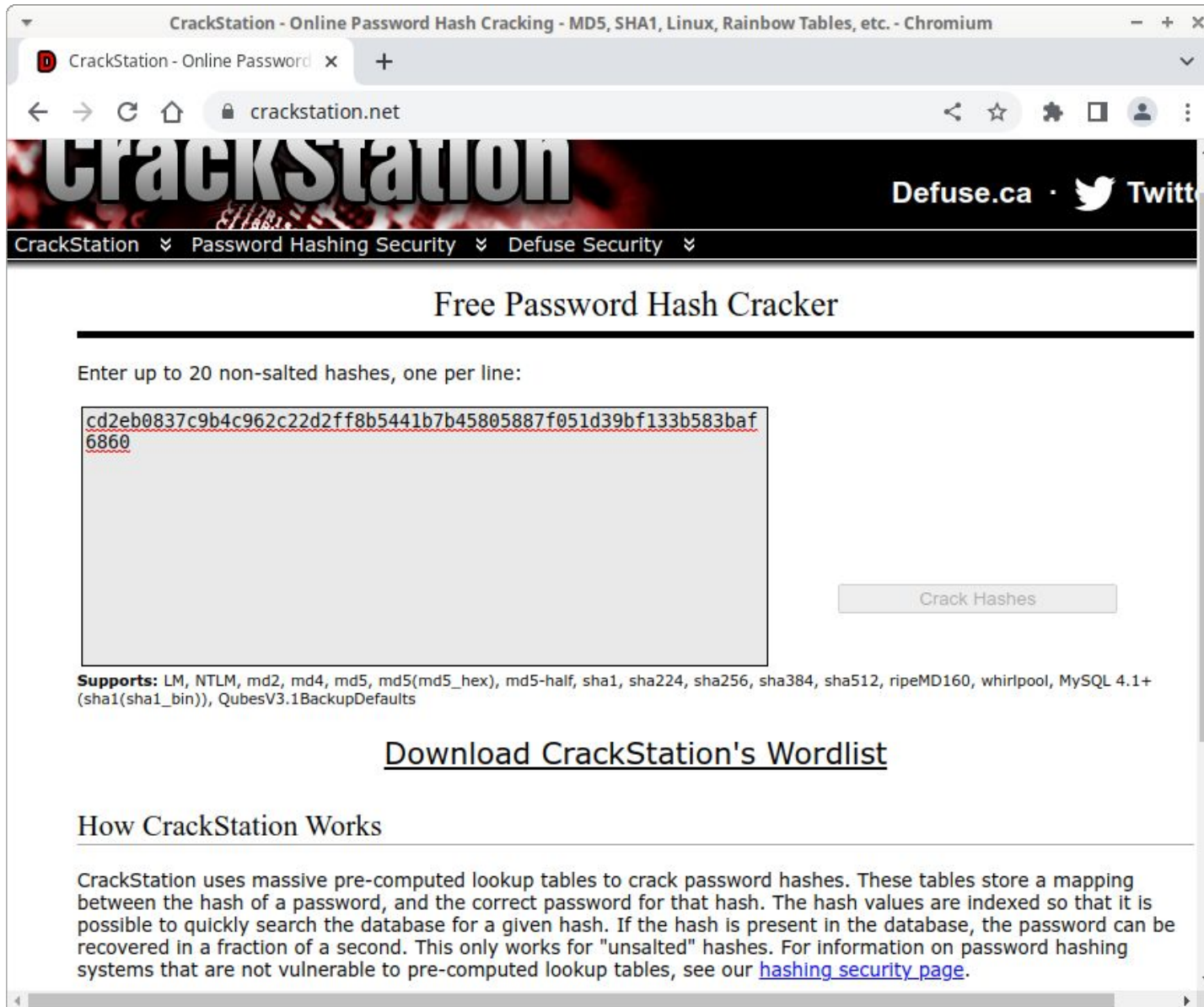
- sha256 hash code of that password:

- cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f
051d39bf133b583baf6860

- See <https://crackstation.net/>

- Can derive xxx

Data Storage Attacks



The screenshot shows a web browser window with the title "CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc. - Chromium". The address bar shows "crackstation.net". The website header features the "CrackStation" logo and navigation links for "Defuse.ca" and "Twitter". A main navigation bar includes "CrackStation", "Password Hashing Security", and "Defuse Security".

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f051d39bf133b583baf6860
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

[Download CrackStation's Wordlist](#)

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).


Data Storage Attacks

CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc. - Chromium

CrackStation - Online Password x +

crackstation.net

CrackStation


Defuse.ca ·  Twitter

CrackStation Password Hashing Security Defuse Security

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f051d39bf133b583baf6860
```

I'm not a robot  reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f051d39bf133b583baf6860	sha256	xxx

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Data Storage Attacks

- **Solution:**

- Store hash codes of *salted* passwords

- `hash_code =`

- `sha256('!@#$%^' + password)`

- hash codes of salted passwords will not be found in a rainbow table

Data Storage Attacks

- **Problem:**

- If an attacker learns the app's salt string, then the attacker (with lots of effort) might generate a rainbow table that contains hash codes for common passwords that are salted
 - ... And so might discover the app's (common) passwords

- **Solution:**

- Use a different salt string for each password

Data Storage Attacks

Salting and sha256 hashing in Python

```
$ python
>>> import werkzeug.security
>>> h = werkzeug.security.generate_password_hash('xxx', 'pbkdf2')
>>> h
'pbkdf2:sha256:600000$G8hNoAKf6ttD5iBa$262b04f2f287889ddffd77b0a735
b543491954d917d20bb36ae6ce2bd0ee5fde'
>>> werkzeug.security.check_password_hash(h, 'xxx')
True
>>> werkzeug.security.check_password_hash(h, 'yyy')
False
>>> quit()
$
```

Data Storage Attacks

Salting and sha256 hashing in Python

algorithm

salt

hashcode

```
pbkdf2 : sha256 : 600000 $G8hNoAKf6ttD5iBa $262b04  
f2f287889ddffd77b0a735b543491954d917d20bb36a  
e6ce2bd0ee5fde
```

Data Storage Attacks

- See **PennyAdmin7bSaltHash** 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
 - penny.py, **auth.py**

Data Storage Attacks

- Q: Project concern?
- A: **Yes**
 - If your app does its own checking of user passwords

Agenda

- Data storage attacks
- **Data comm attacks**
- Third-party authentication (briefly)
 - CAS authentication
 - Microsoft Entra ID authentication
 - Google authentication

Data Comm Attacks

- **Problem:**
 - Attacker may access data during comm between PennyAdmin app and browser
- **Solution:**
 - *Hypertext Transfer Protocol Secure (HTTPS)*

Data Comm Attacks

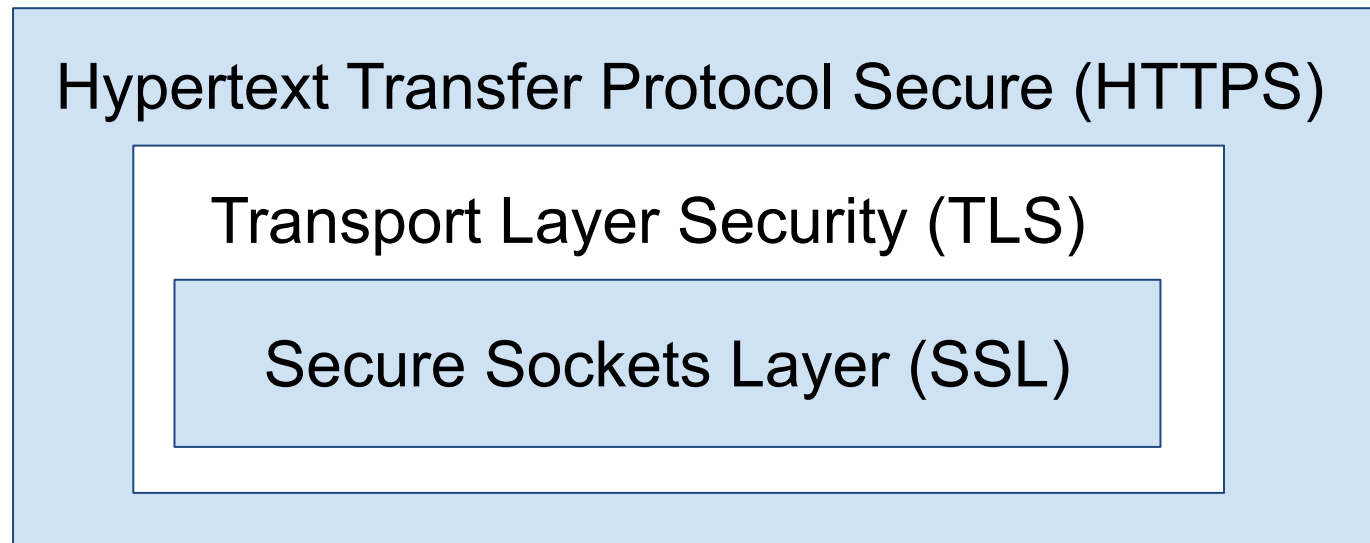
- **Technical** advantages of using HTTPS
 - Confidentiality
 - Prohibits *message eavesdropping attacks*
 - Integrity
 - Prohibits *message tampering attacks*
 - Authentication
 - Prohibits *message forgery attacks*

Data Comm Attacks

- **Business** advantages of using HTTPS
 - Increases user confidence/trust in website
 - Increases Google search rank of website

Data Comm Attacks

- How HTTPS works:



Data Comm Attacks

- How to use HTTPS:
 - Configure server & app to use (& require use of) HTTPS
 - Command browser to send request specifying HTTPS as protocol
 - `https://...`

Data Comm Attacks

- How to configure server & app to use (& require use of) HTTPS:
 - Depends upon server...

Data Comm Attacks

- **Render server**

- Already configured to use (& require use of) HTTPS
 - When server receives `http://something` request, it sends redirect for `https://something` request
- So:
 - **Server:** Do nothing!
 - **App:** Do nothing!

Data Comm Attacks

- **Heroku server**

- Already configured to use (but not require use of) HTTPS

- When server receives **https://something** request, it uses HTTPS

- When server receives **http://something** request, it uses HTTP

- So

- **Server:** (Regrettably) Do nothing!

- **App:** Small change...

Data Comm Attacks

- **Solution 1:**
 - App explicitly performs redirects

Data Comm Attacks

- See **PennyAdmin&Https** 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
 - **penny.py**, auth.py

Data Comm Attacks

- **Solution 2:**
 - *flask_talisman* module

Data Comm Attacks

- See **PennyAdmin8bHttps** 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
 - **penny.py**, auth.py

Data Comm Attacks

- Notes:
 - Good to design your app to require use of HTTPS even when the app server already forces use of HTTPS
 - flask_talisman implements some additional security measures
 - Need not configure Flask test server to use (or require use of) HTTPS
 - But if you want to...
 - Or if you're using Google authentication...

Data Comm Attacks

- How to configure Flask test server & app to use (& require use of) HTTPS:

Data Comm Attacks

- **Preliminary step:** Get a *certificate* for your app
- **Option 1:** Get a certificate that is signed by a *certificate authority*

Data Comm Attacks

Certificate authorities:

Rank	Authority	Market Share
1	Let's Encrypt	64.3%
2	GlobalSign	23.9%
3	Sectigo	5.6%
4	GoDaddy Group	3.8%
5	DigiCert Group	1.9%
6	Actalis	0.7%
7	Certum	0.6%
8	Secom Trust	0.3%

https://w3techs.com/technologies/overview/ssl_certificate

as of April 2024

Data Comm Attacks

- **Preliminary step:** Get a certificate for your app
- **Option 1:** Buy a certificate that is signed by a certificate authority
- **Option 2:** Create a ***self-signed certificate***

Data Comm Attacks

Linux, Mac, MS Windows Git Bash:

```
$ openssl req -x509 -newkey rsa:4096 -nodes -out cert.pem -keyout key.pem -days 365
Generating a RSA private key
.....++++
.....++++
writing new private key to 'key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]: US
State or Province Name (full name) [Some-State]: NJ
Locality Name (eg, city) []: Princeton
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Princeton University
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: localhost
Email Address []:
$
```

Output: cert.pem, key.pem

Data Comm Attacks

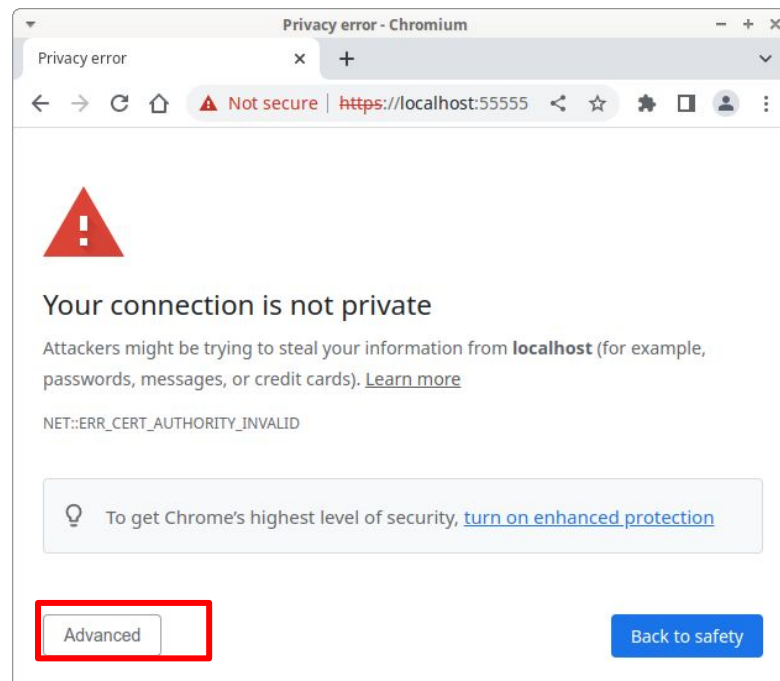
- Self-signed certificate
 - Confidentiality: yes
 - Integrity: yes
 - Authentication: no

Data Comm Attacks

- See **PennyAdmin08cHttpsLocal** 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
 - penny.py, auth.py

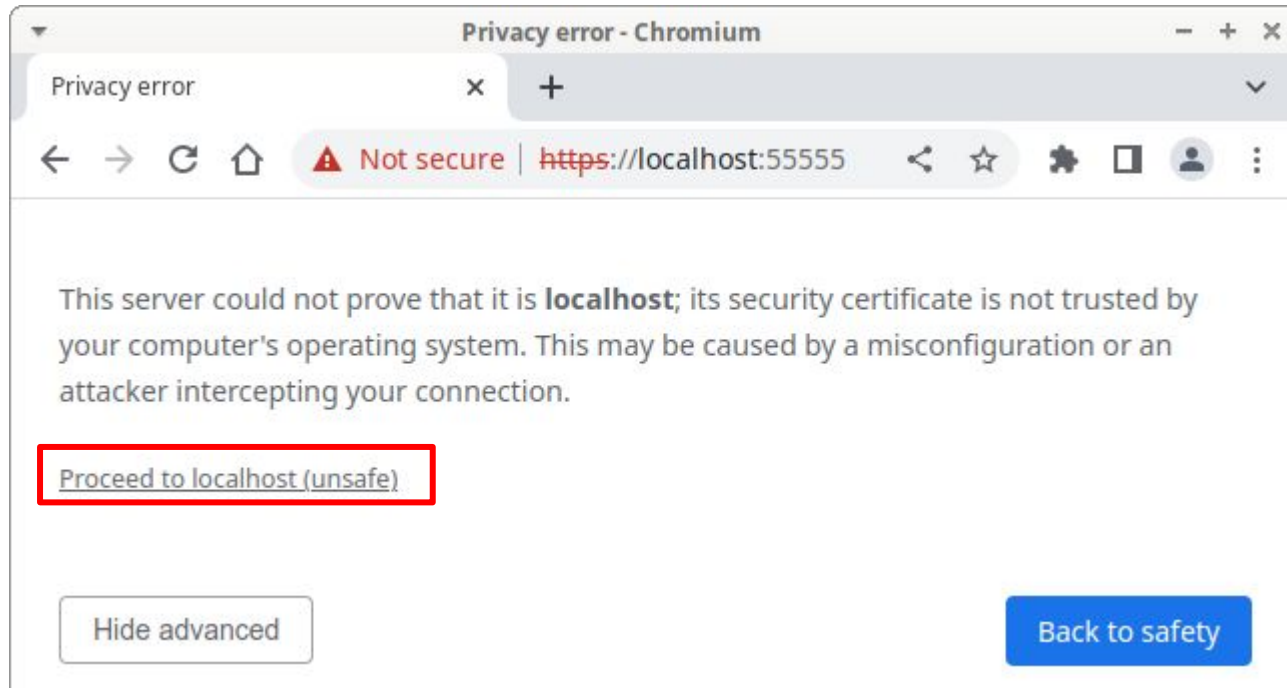
Data Comm Attacks

- See **PennyAdmin08cHttpsLocal** app
 - On local computer with Flask test server (using self-signed certif)



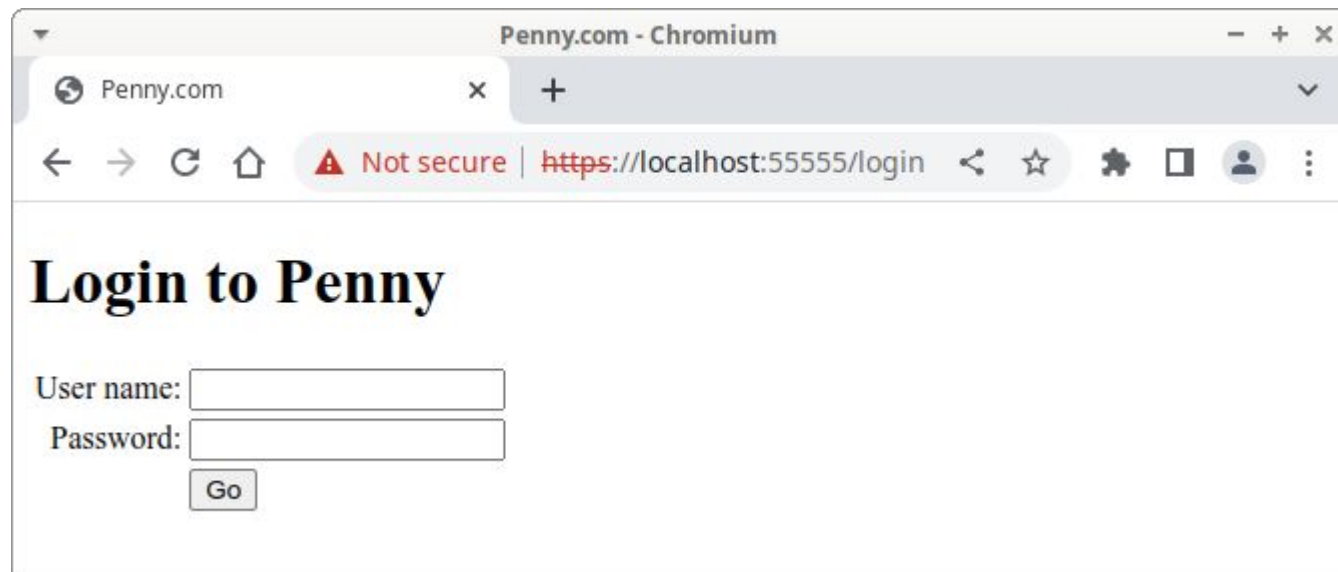
Data Comm Attacks

- See **PennyAdmin08cHttpsLocal** app
 - On local computer with Flask test server (using self-signed certif)



Data Comm Attacks

- See **PennyAdmin08cHttpsLocal** app
 - On local computer with Flask test server (using self-signed certif)



Data Comm Attacks

- Q: Project concern?
- A: **Yes**

Agenda

- Data storage Attacks
- Data comm attacks
- **Third-party authentication (briefly)**
 - CAS authentication
 - Microsoft Entra ID authentication
 - Google authentication

Agenda

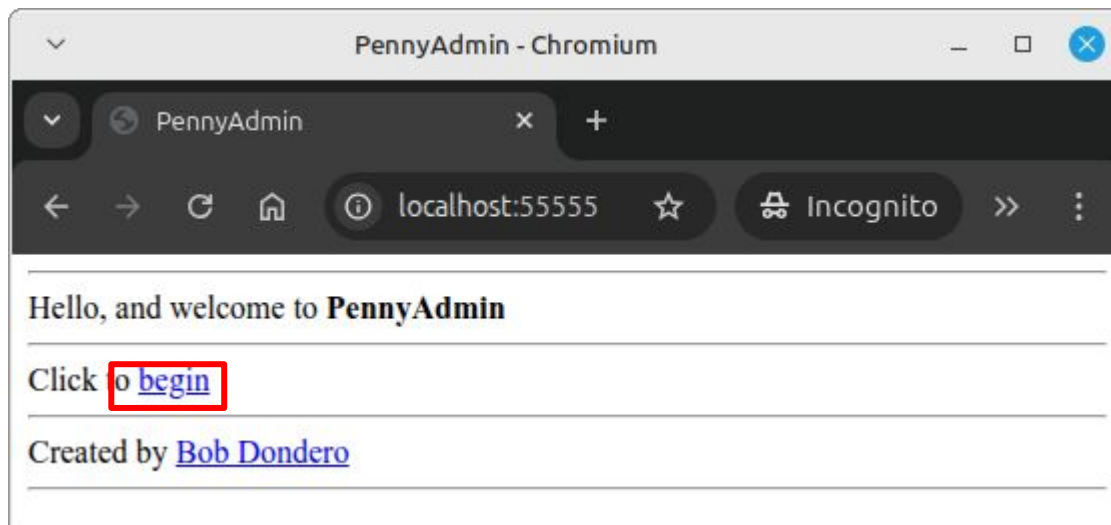
- Data storage attacks
- Data comm attacks
- Third-party authentication (briefly)
 - **CAS authentication**
 - Microsoft Entra ID authentication
 - Google authentication

CAS Authentication

- See **PennyAdminCas** app
 - Its behavior...

CAS Authentication

- See **PennyAdminCas** app (cont.)



CAS Authentication

- See **PennyAdminCas** app (cont.)

CAS - Central Authentication Service Login - Chromium

CAS - Central Authentication

fed.princeton.edu/cas/login?service=http%3A//localhost%3A5555/

PRINCETON UNIVERSITY
Central Authentication Service

NetID*

rdondero

Password*

.....

LOGIN

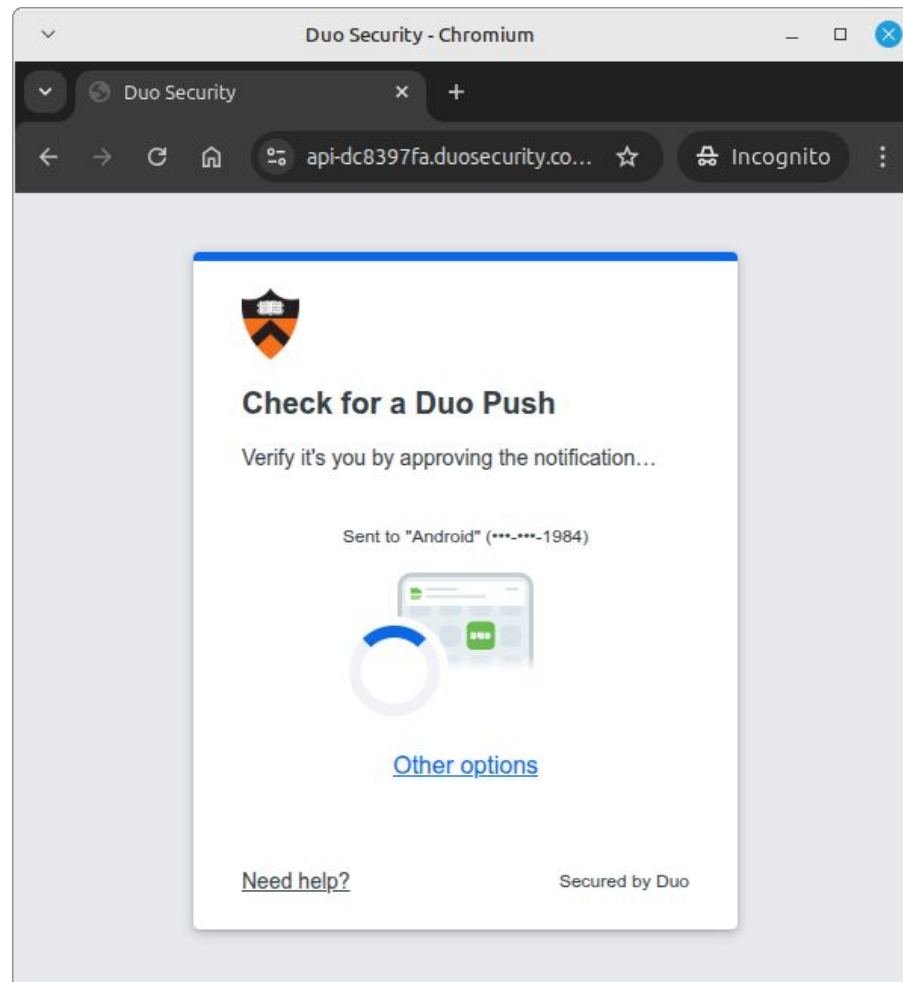
[Change my Password or Get Help](#)

By proceeding to access and use University computing and network resources through this sign-on, you agree to abide by applicable laws and University policies in your use of these resources. The University's right to access, preserve, and review information stored on or transmitted through these resources is described in the [Access to Accounts Policy](#).

© 2023 The Trustees of Princeton University

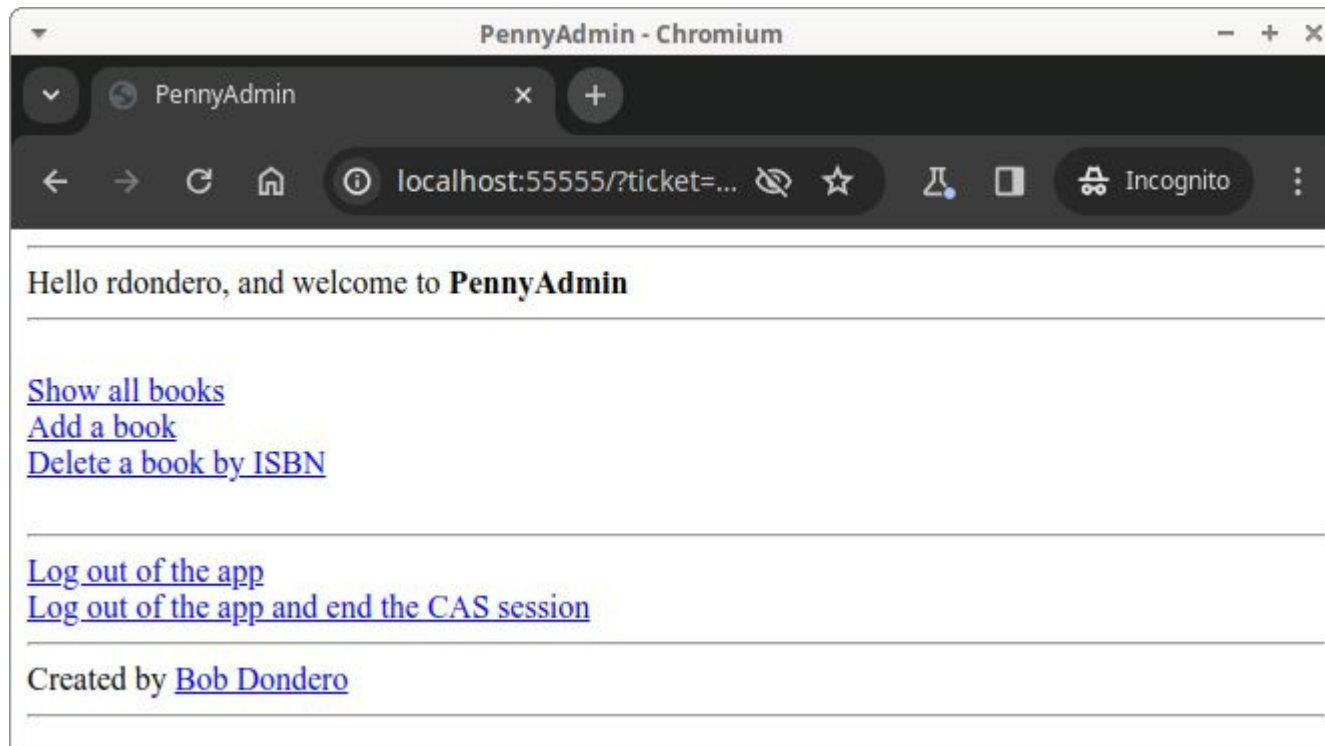
CAS Authentication

- See **PennyAdminCas** app (cont.)



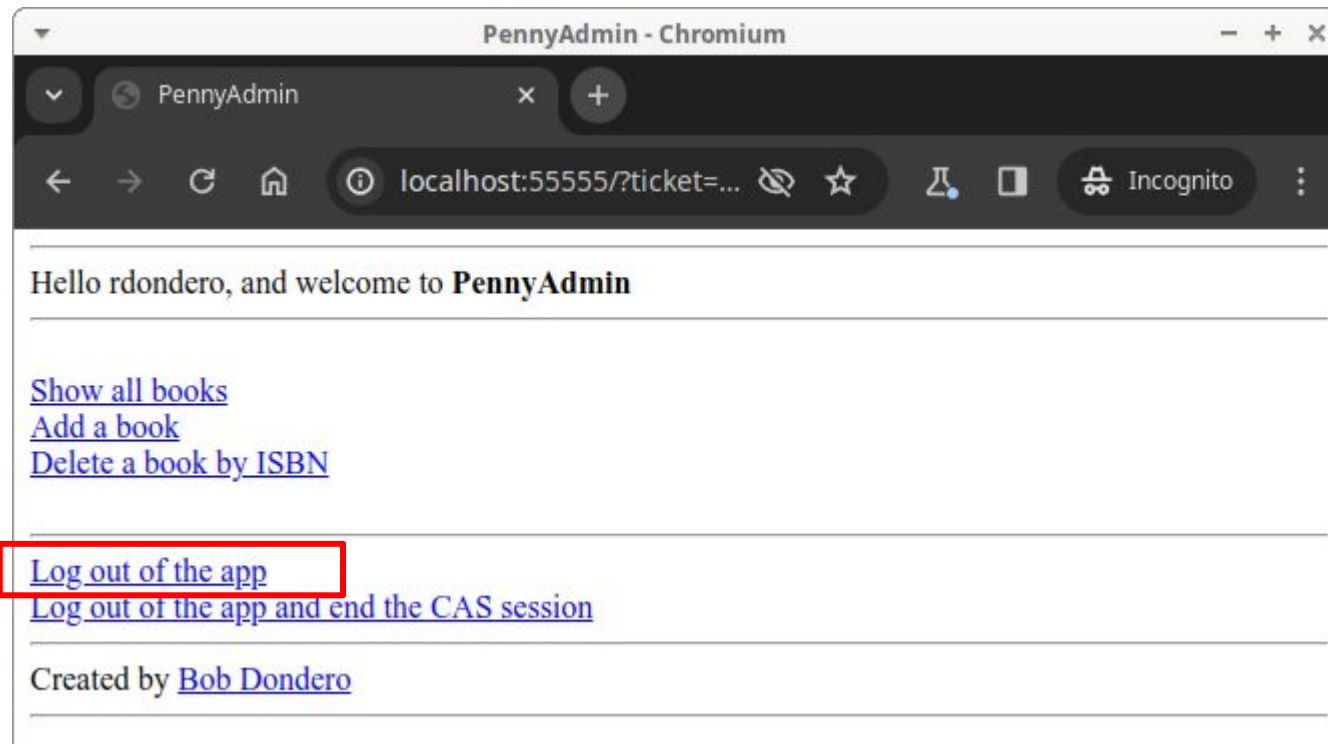
CAS Authentication

- See **PennyAdminCas** app (cont.)



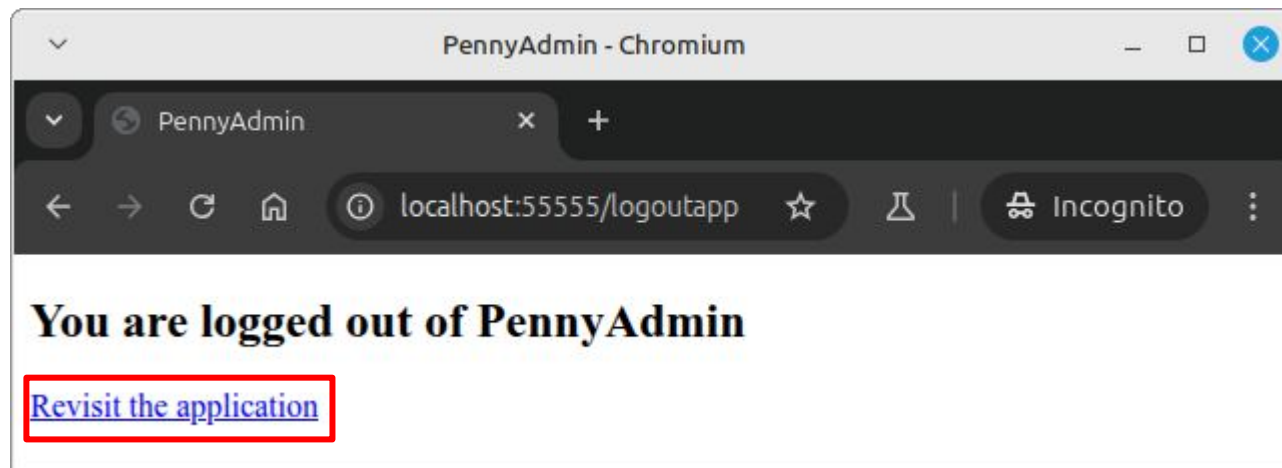
CAS Authentication

- See **PennyAdminCas** app (cont.)



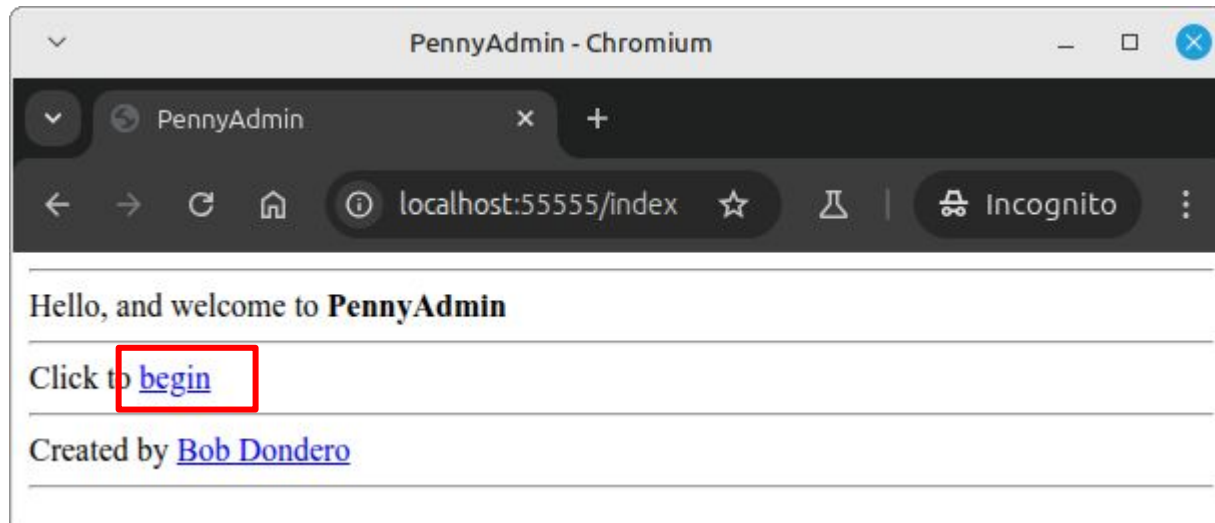
CAS Authentication

- See **PennyAdminCas** app (cont.)



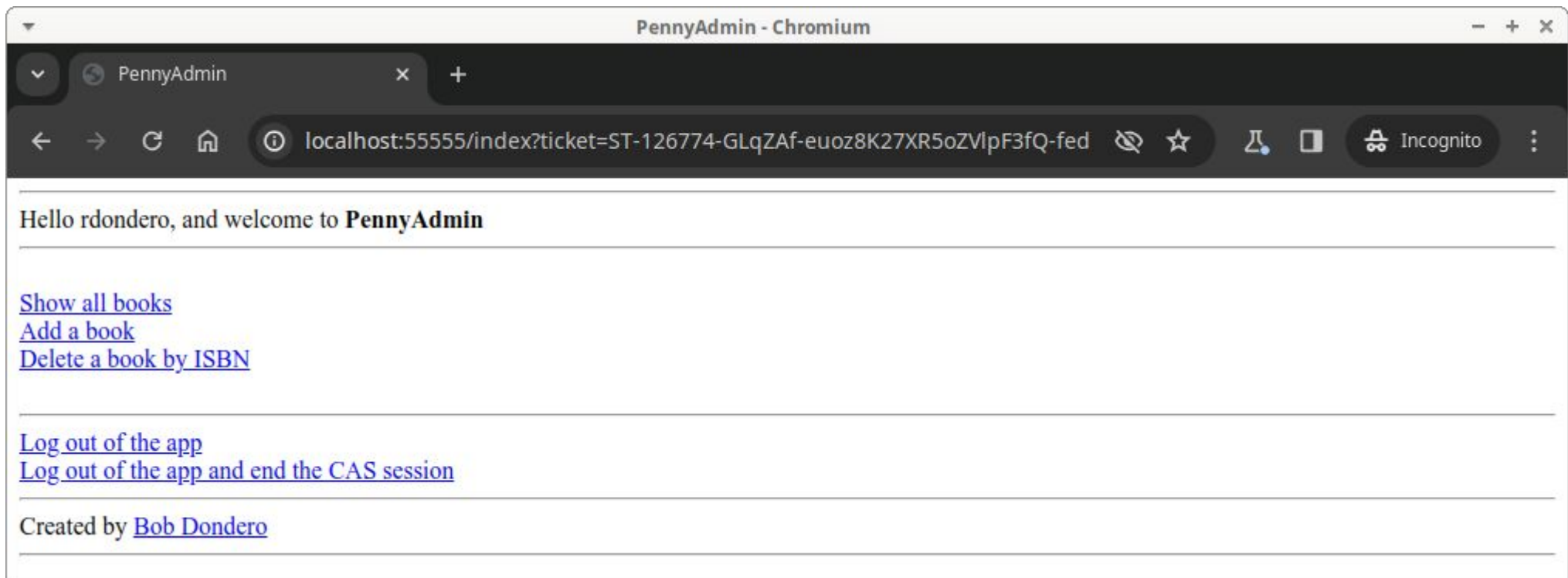
CAS Authentication

- See **PennyAdminCas** app (cont.)



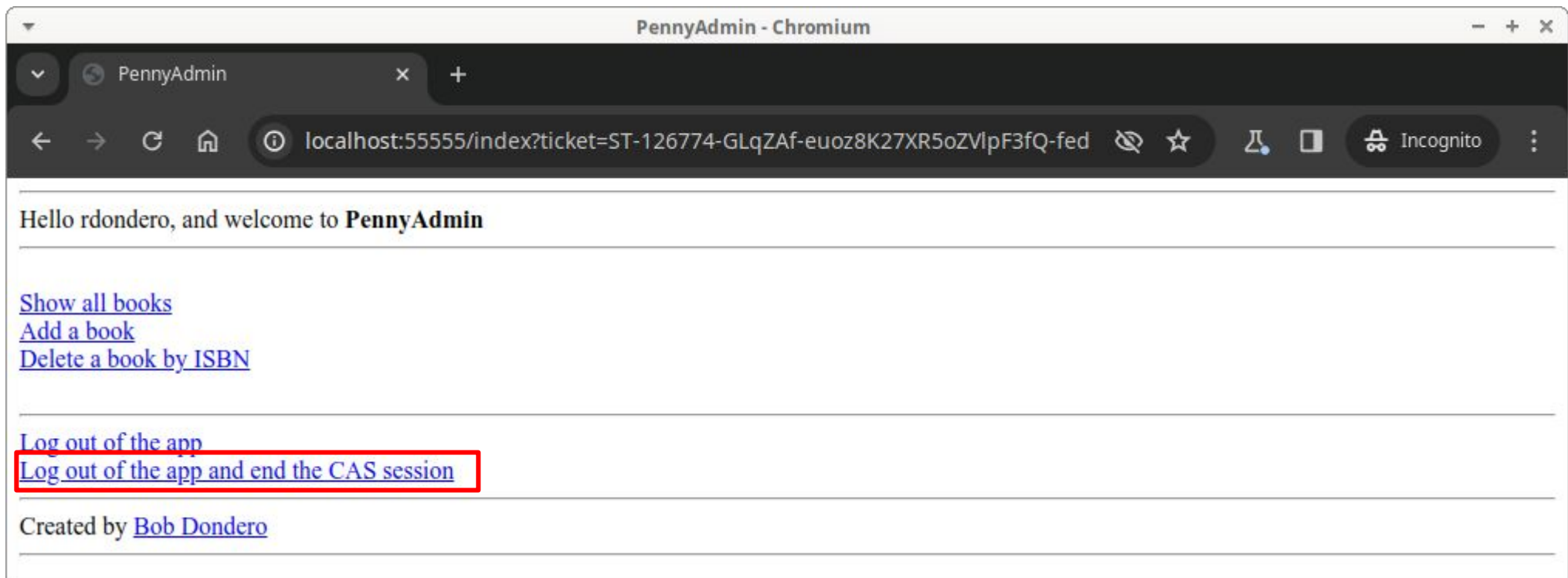
CAS Authentication

- See **PennyAdminCas** app (cont.)



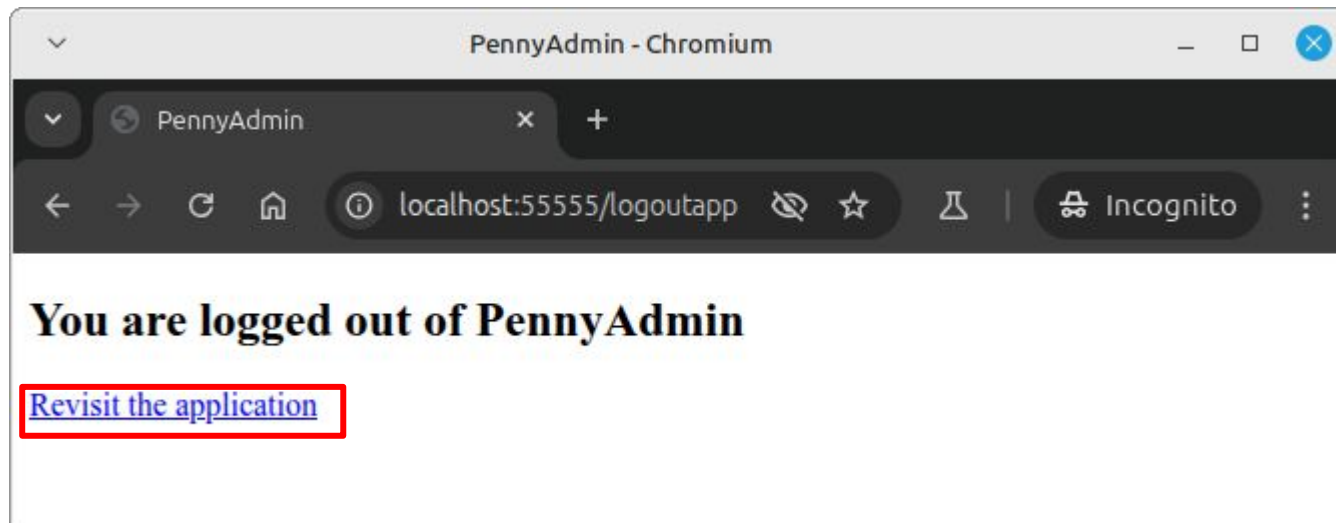
CAS Authentication

- See **PennyAdminCas** app (cont.)



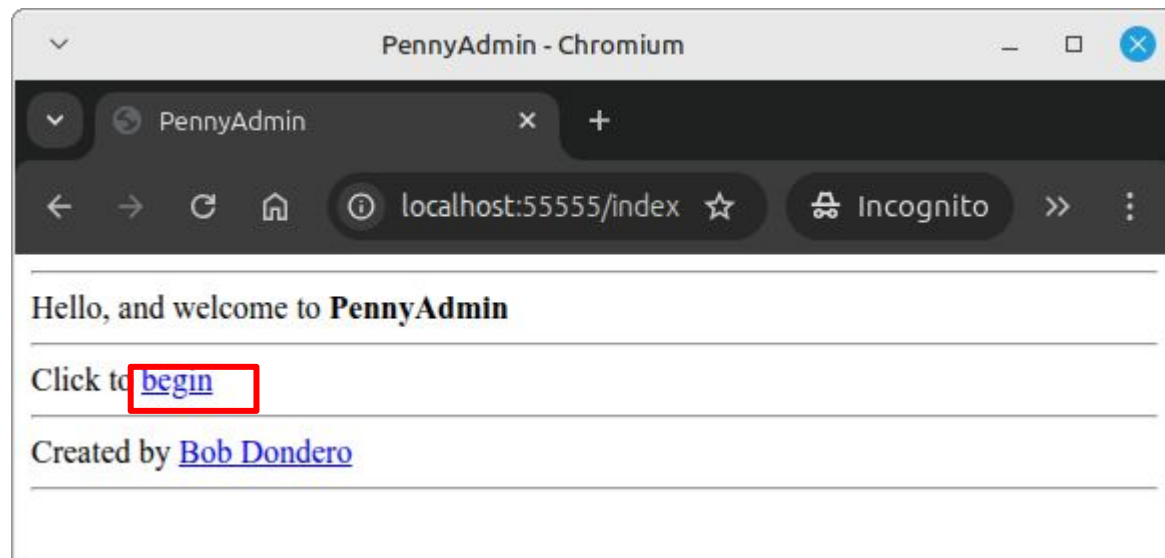
CAS Authentication

- See **PennyAdminCas** app (cont.)



CAS Authentication

- See **PennyAdminCas** app (cont.)



CAS Authentication


- See **PennyAdminCas** app (cont.)

CAS - Central Authentication Service Login - Chromium

CAS - Central Authentication

fed.princeton.edu/cas/login?service=http%3A//localhost%3A5555/index

Incognito

 PRINCETON UNIVERSITY

Central Authentication Service

NetID*

NetID is a required field.

Password:*

LOGIN

[Change my Password or Get Help](#)

By proceeding to access and use University computing and network resources through this sign-on, you agree to abide by applicable laws and University policies in your use of these resources. The University's right to access, preserve, and review information stored on or transmitted through these resources is described in the [Access to Accounts Policy](#).

© 2023 The Trustees of Princeton University

CAS Authentication

- See **PennyAdminCas** app (cont.)
 - See optional lecture material for:
 - The code
 - How to run it on your local computer
 - How to run it on Render
 - How it works

CAS Authentication

- **Pros**

- Application need not manage usernames or passwords
- Application ***cannot*** access passwords!
- Application is constrained to one user community
- Princeton OIT will support for the foreseeable future

CAS Authentication

- **Cons**

- Complex
- Adds overhead, but only during user's first visit to the app per browser session
- Application is constrained to one user community!
- Princeton OIT is phasing out
- Deployment requires (minimal) Princeton OIT cooperation

Agenda

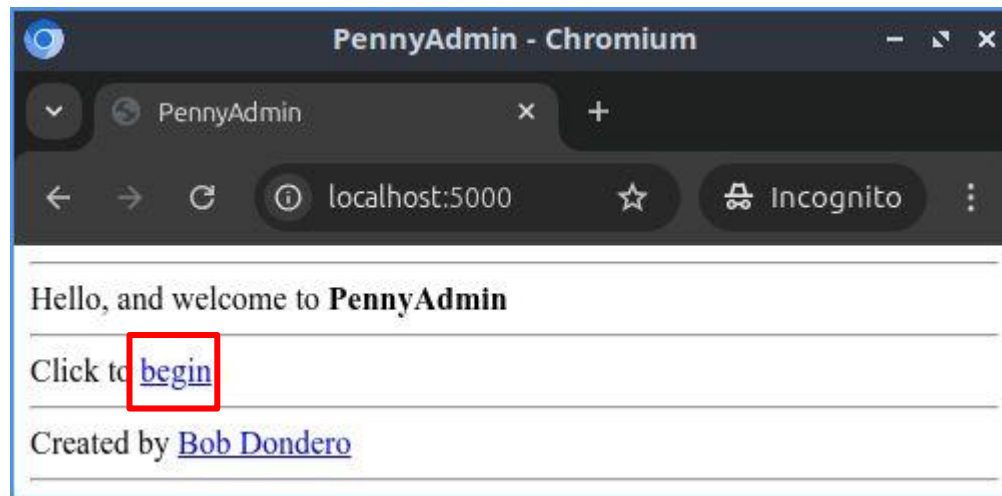
- Data comm attacks
- Third-party authentication (briefly)
 - CAS authentication
 - **Microsoft Entra ID authentication**
 - Google authentication

Entra ID Authentication

- See **PennyAdminEntraID** app
 - Its behavior...

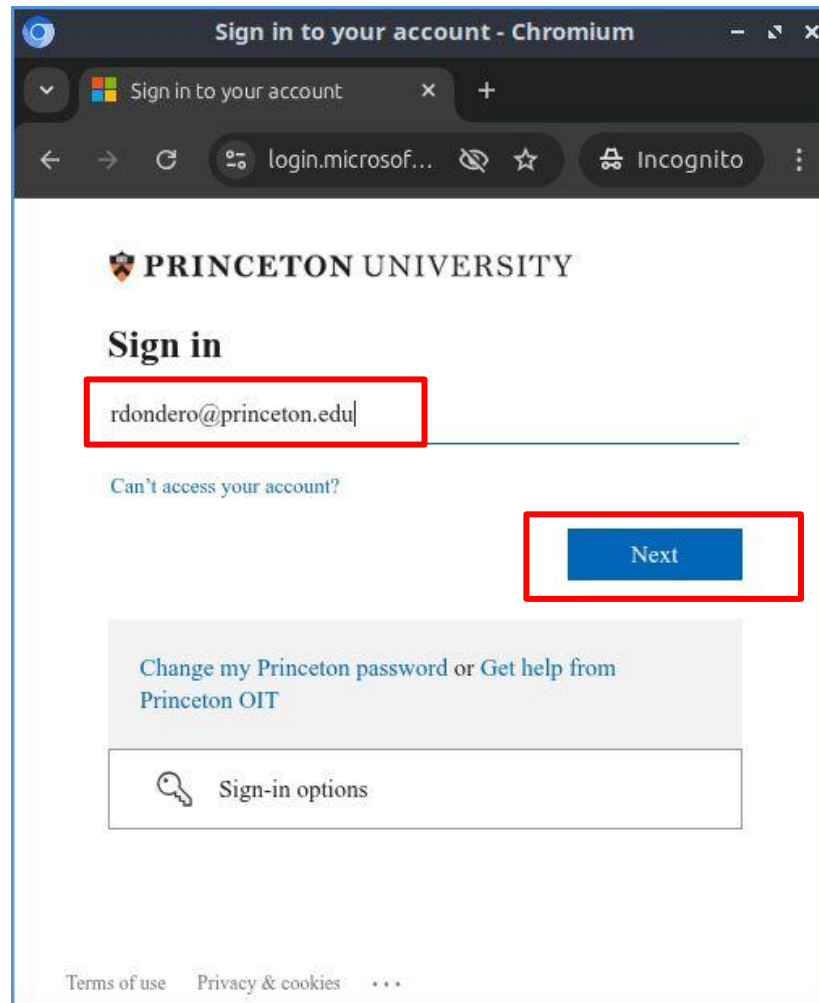
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



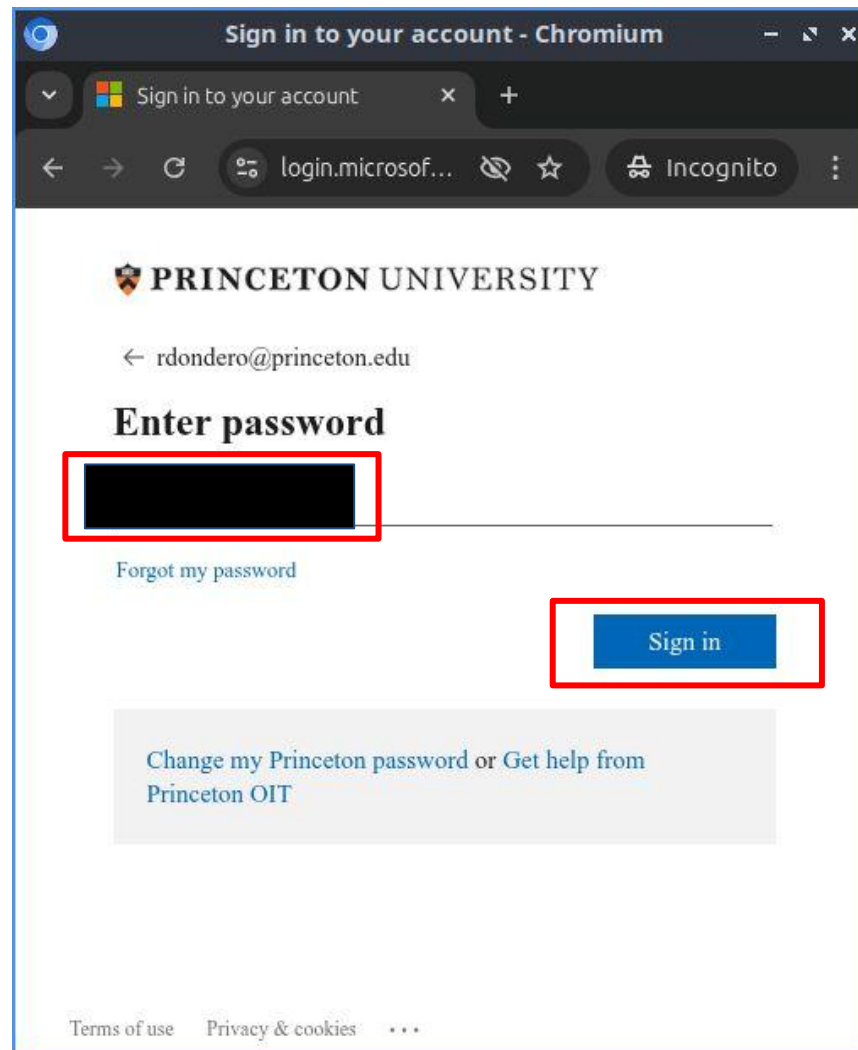
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



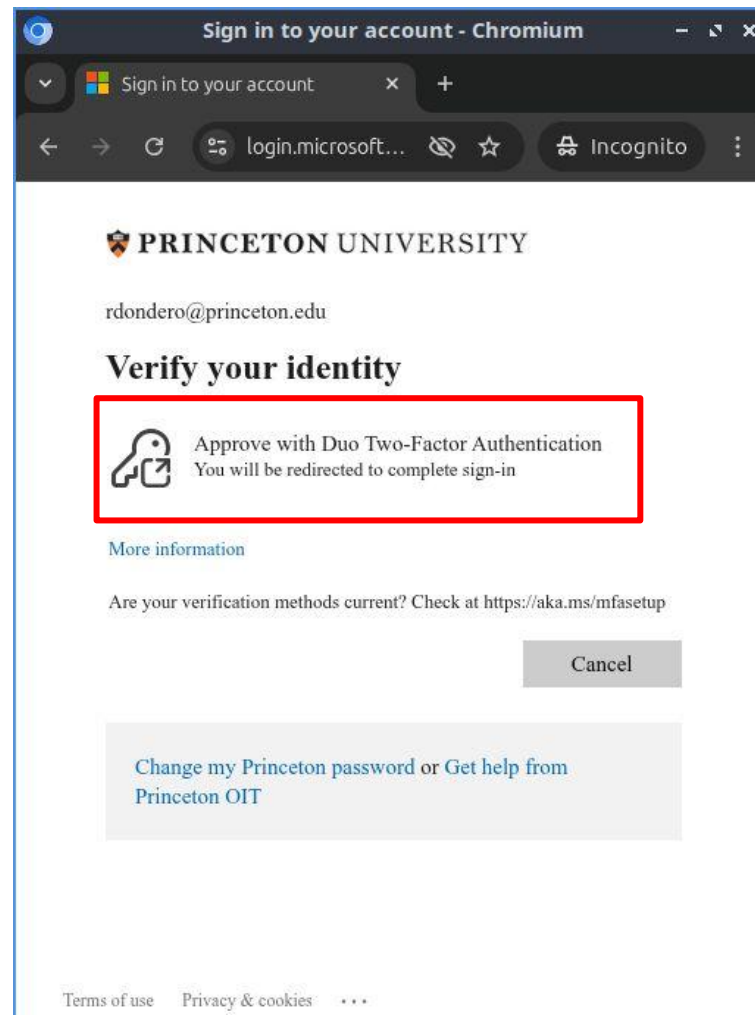
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



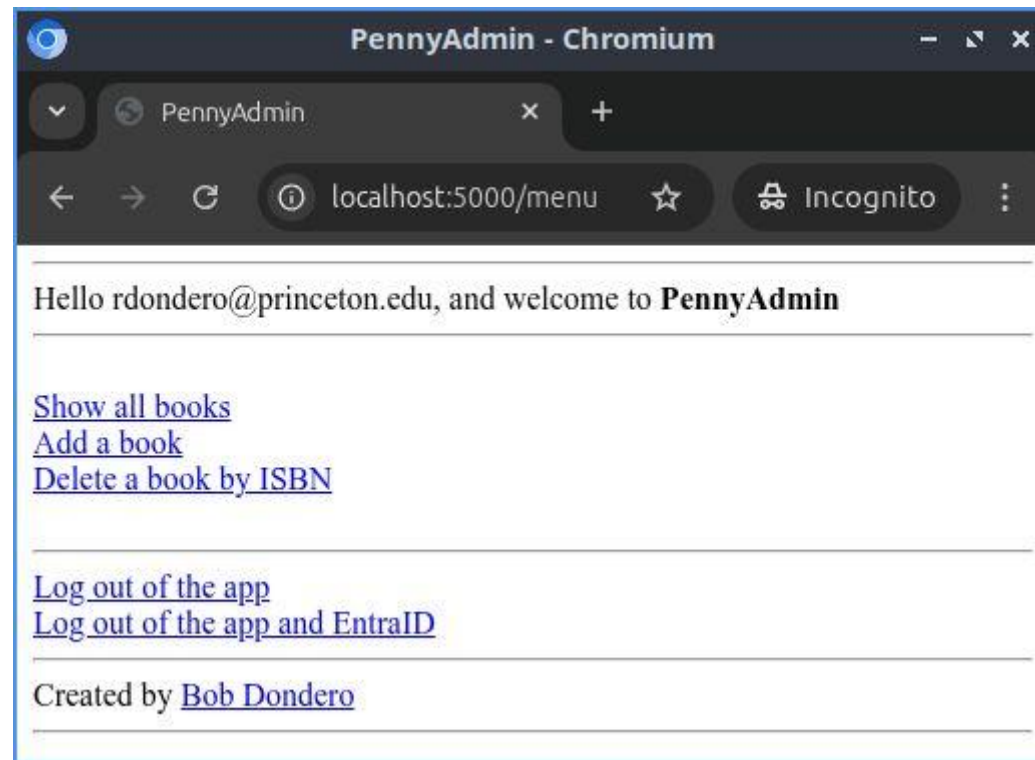
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



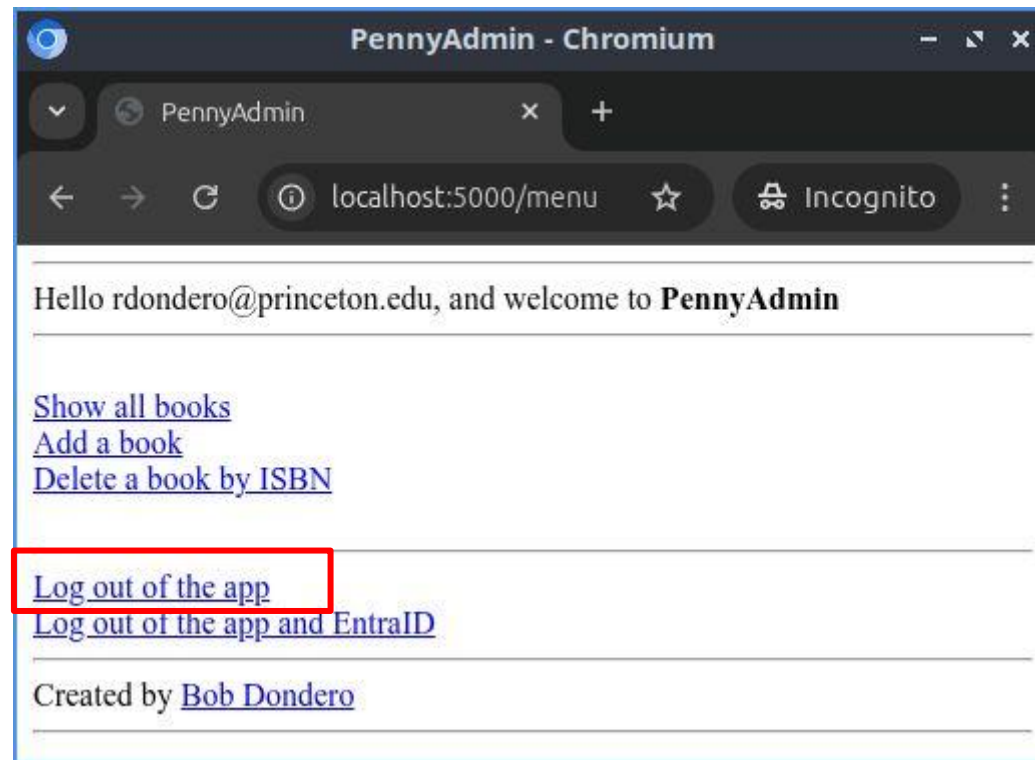
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



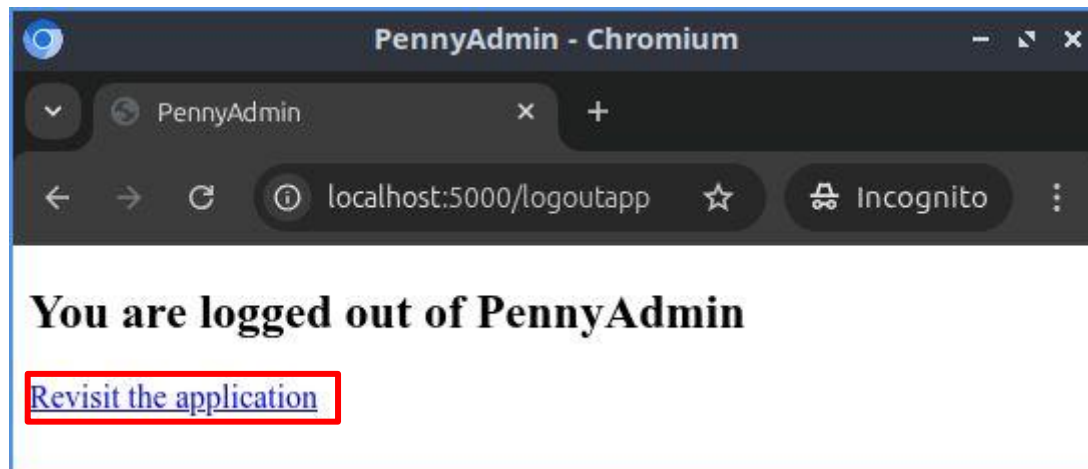
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



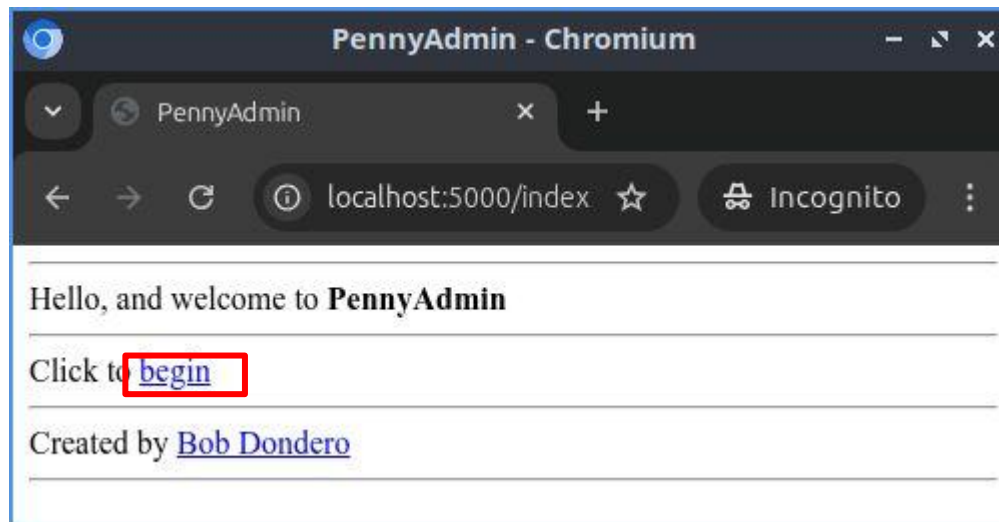
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



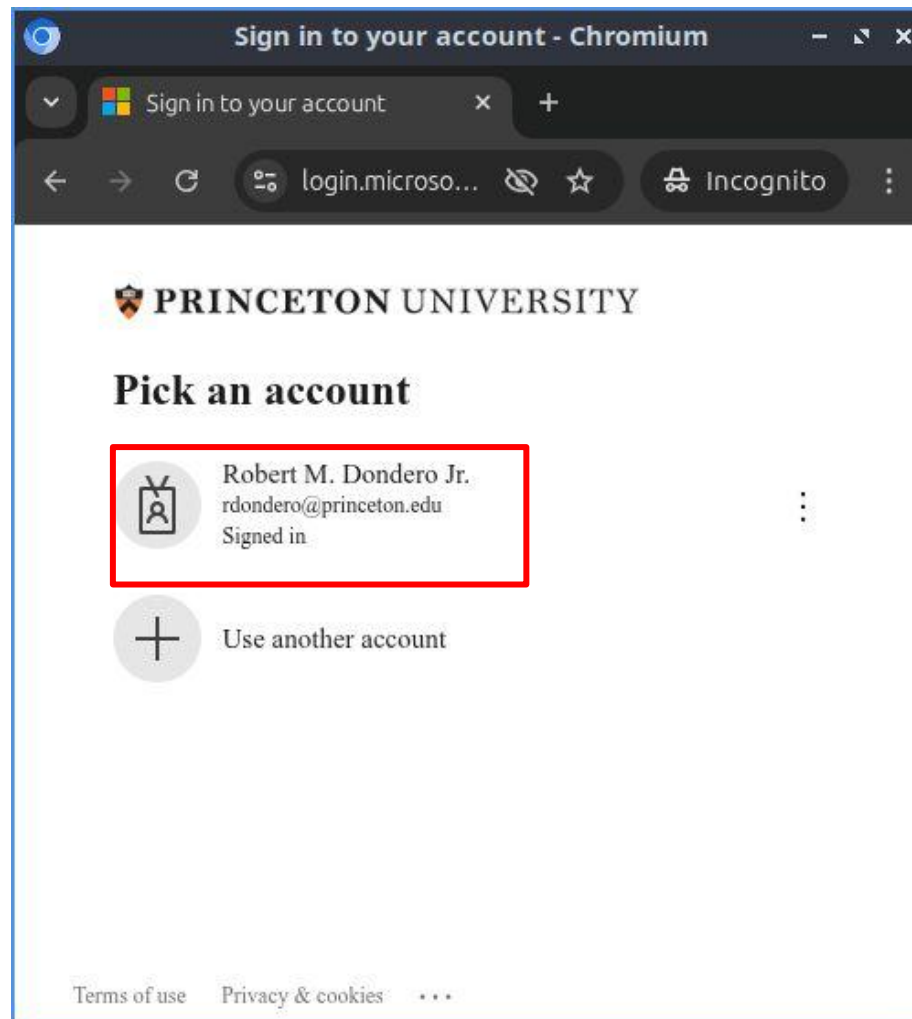
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



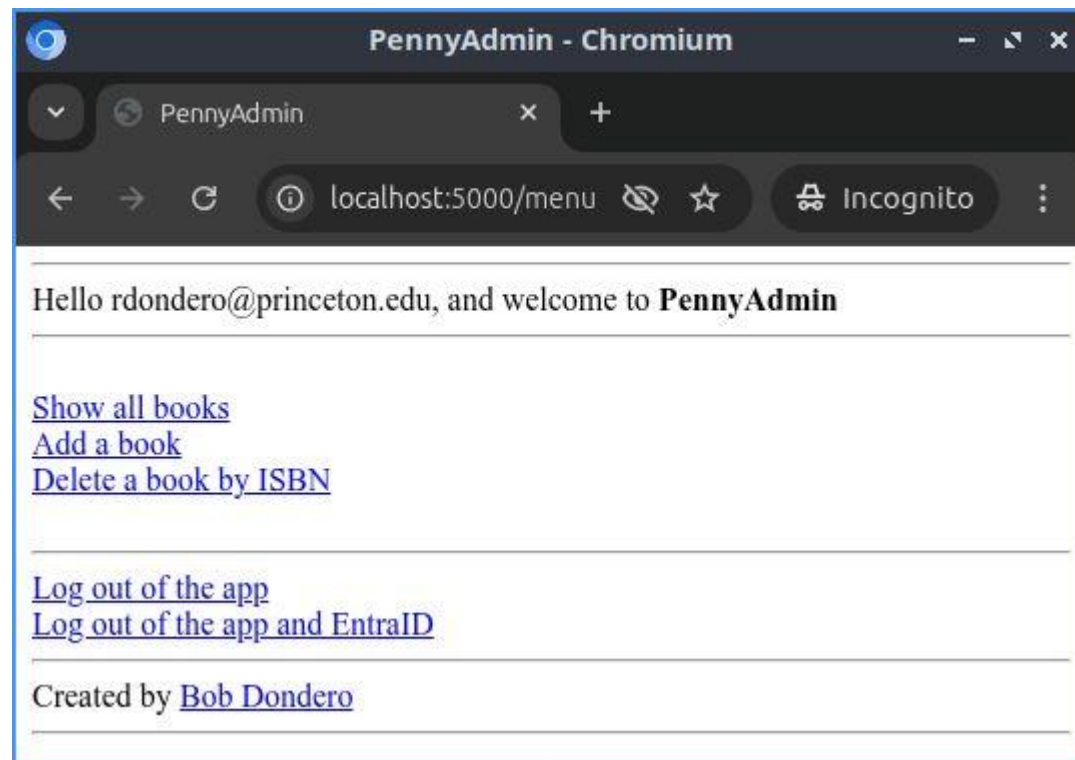
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



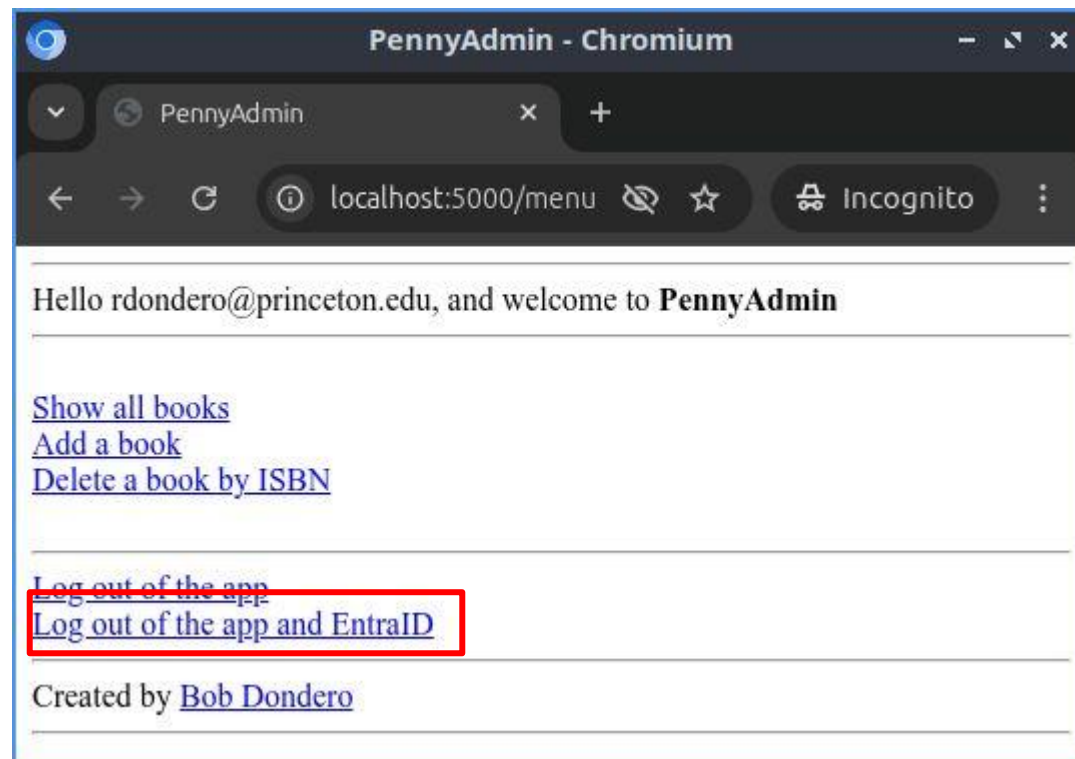
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



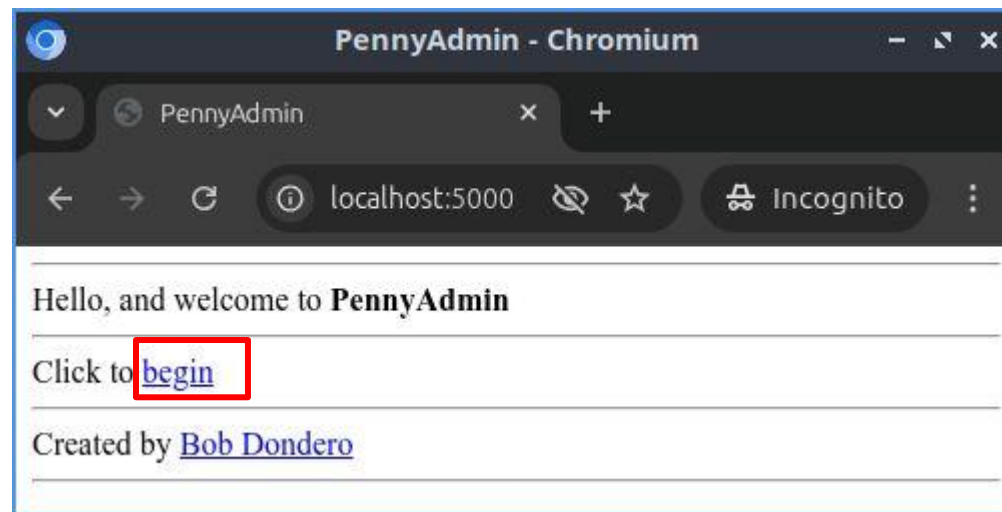
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



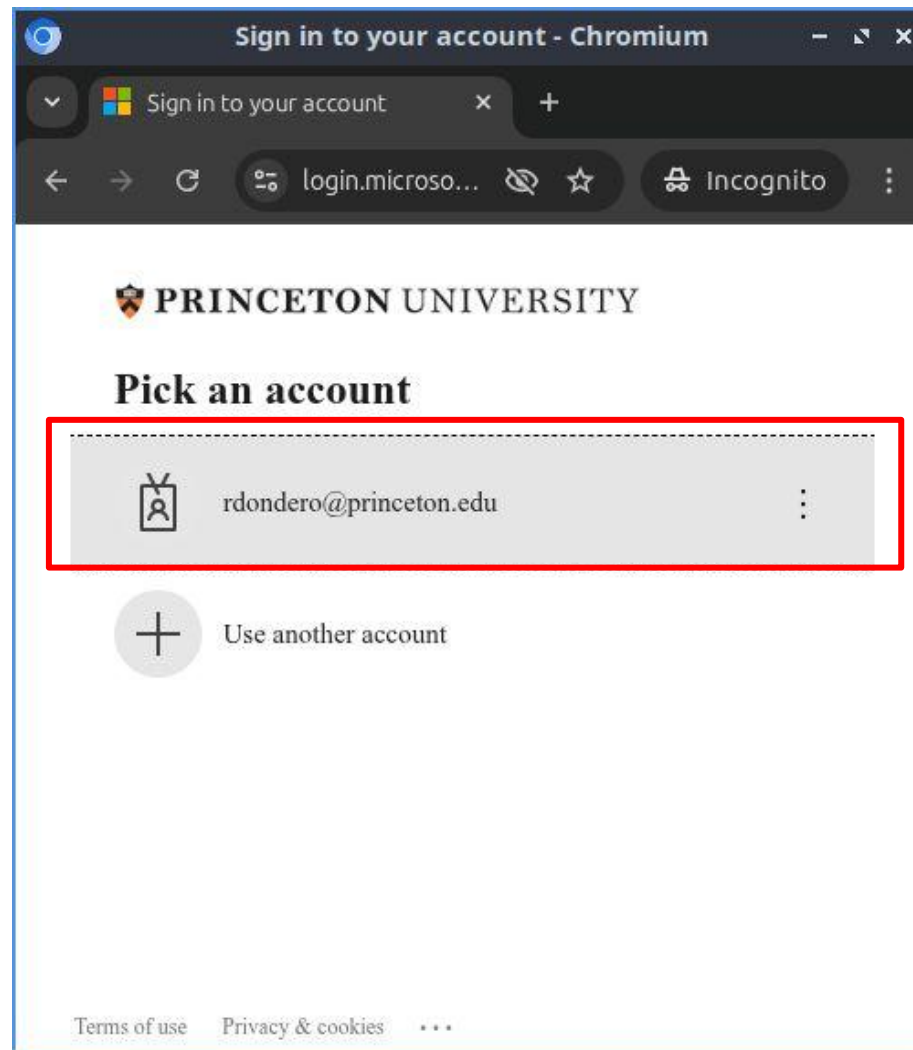
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



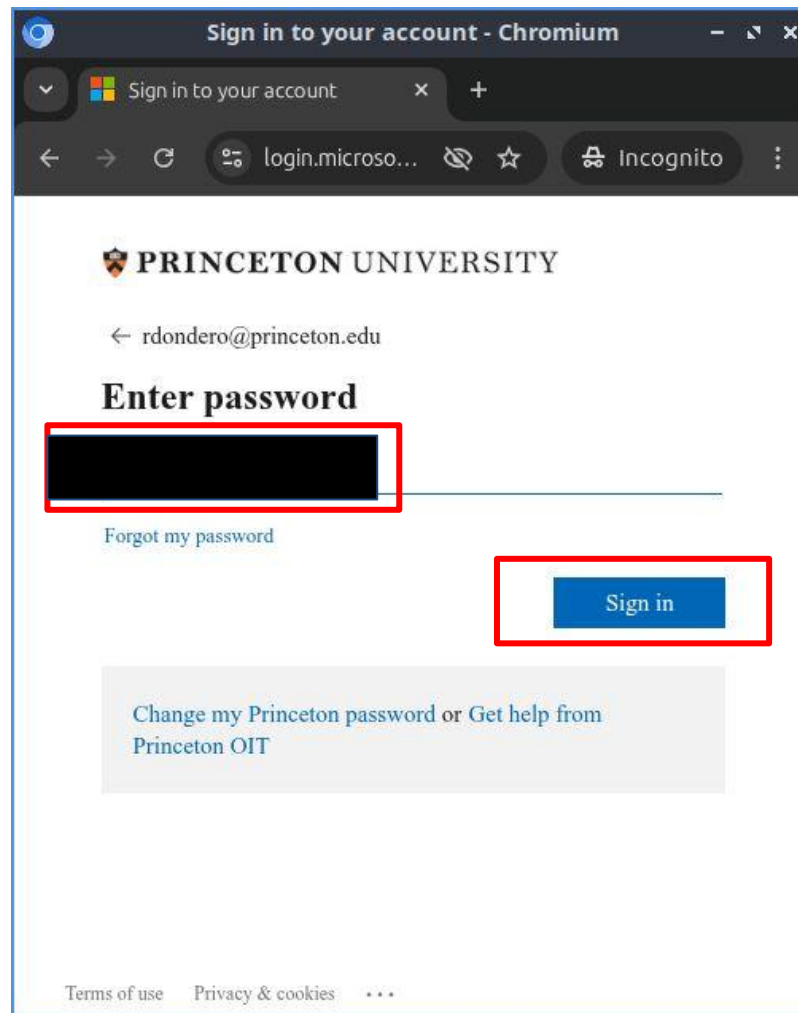
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



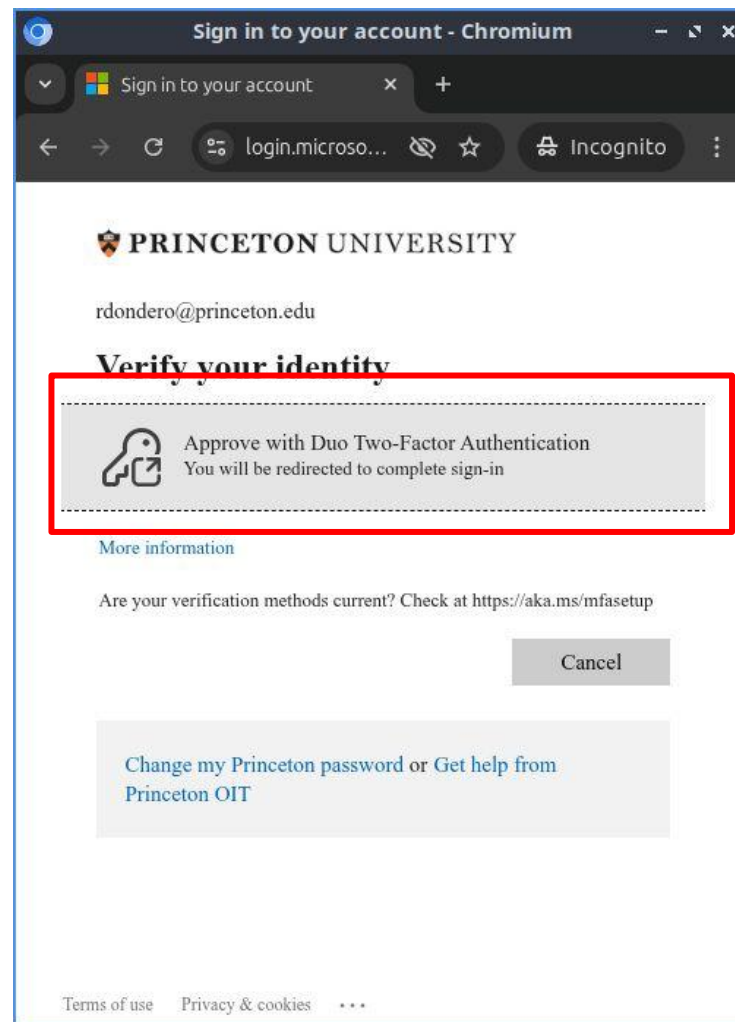
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



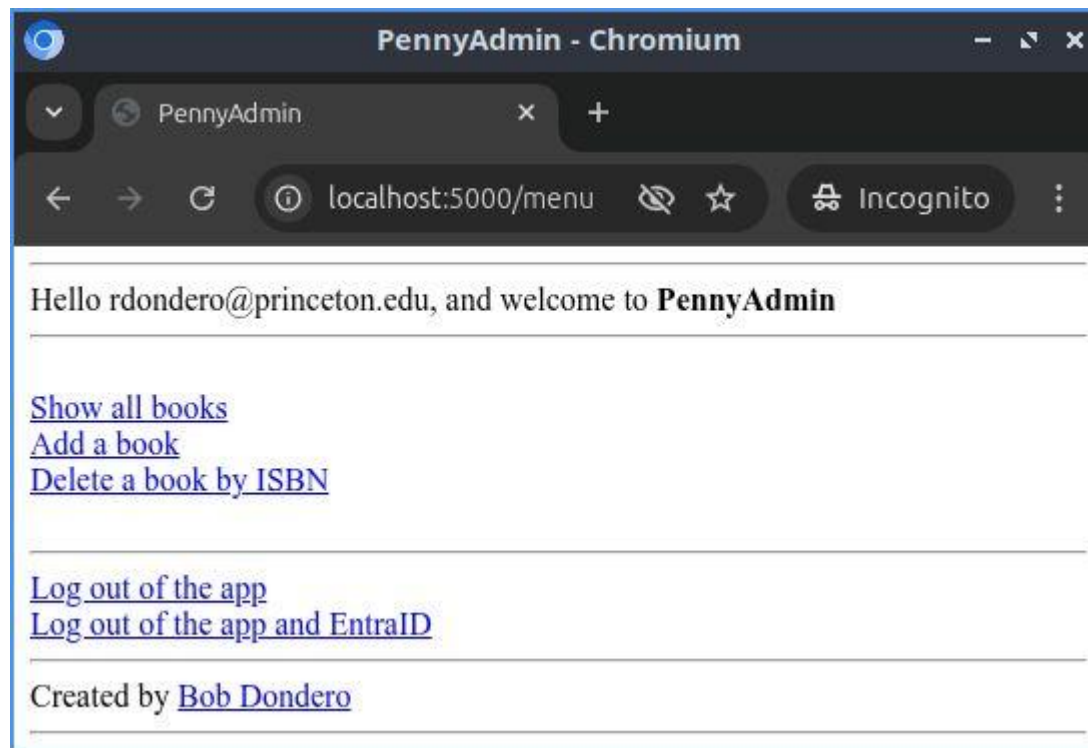
Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)



Entra ID Authentication

- See **PennyAdminEntraID** app (cont.)
 - See optional lecture material for:
 - The code
 - How to run it on your local computer
 - How to run it on Render

Entra ID Authentication

- **Pros**

- Application need not manage usernames or passwords
- Application ***cannot*** access passwords!
- Application is constrained to one user community
- More standards-based than CAS
- Princeton OIT is phasing in
- No need for Princeton OIT cooperation

Entra ID Authentication

- **Cons**

- Complex
- Adds overhead, but only during user's first visit to the app per browser session
- Application is constrained to one user community!

Agenda

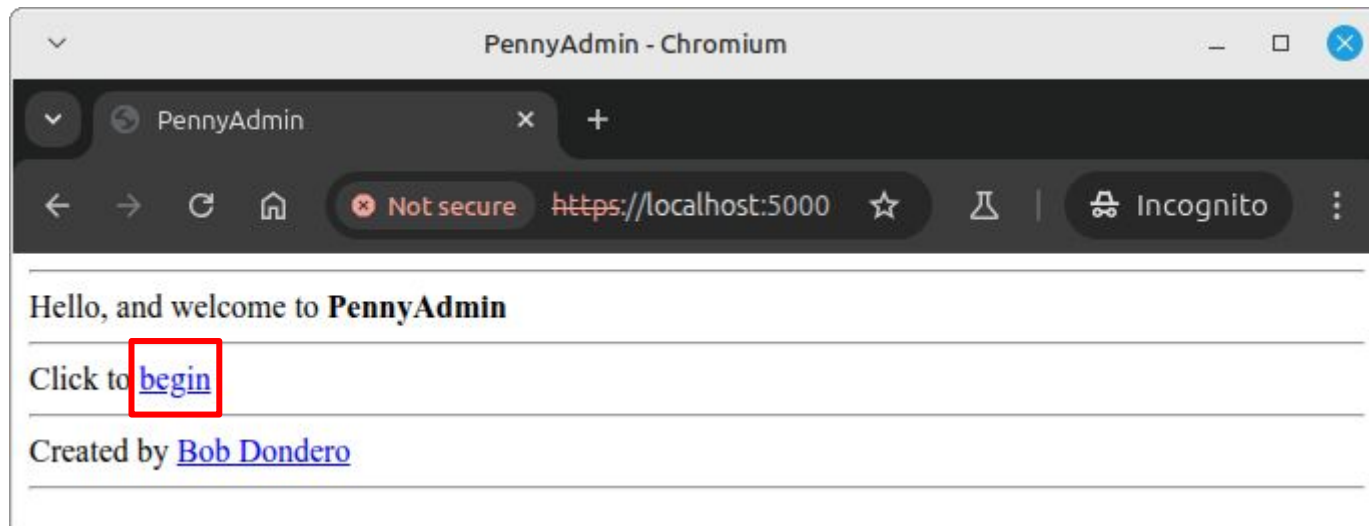
- Data comm attacks
- Third-party authentication (briefly)
 - CAS
 - Microsoft Entra ID authentication
 - **Google authentication**

Google Authentication

- See **PennyAdminGoogle** app
 - Its behavior...

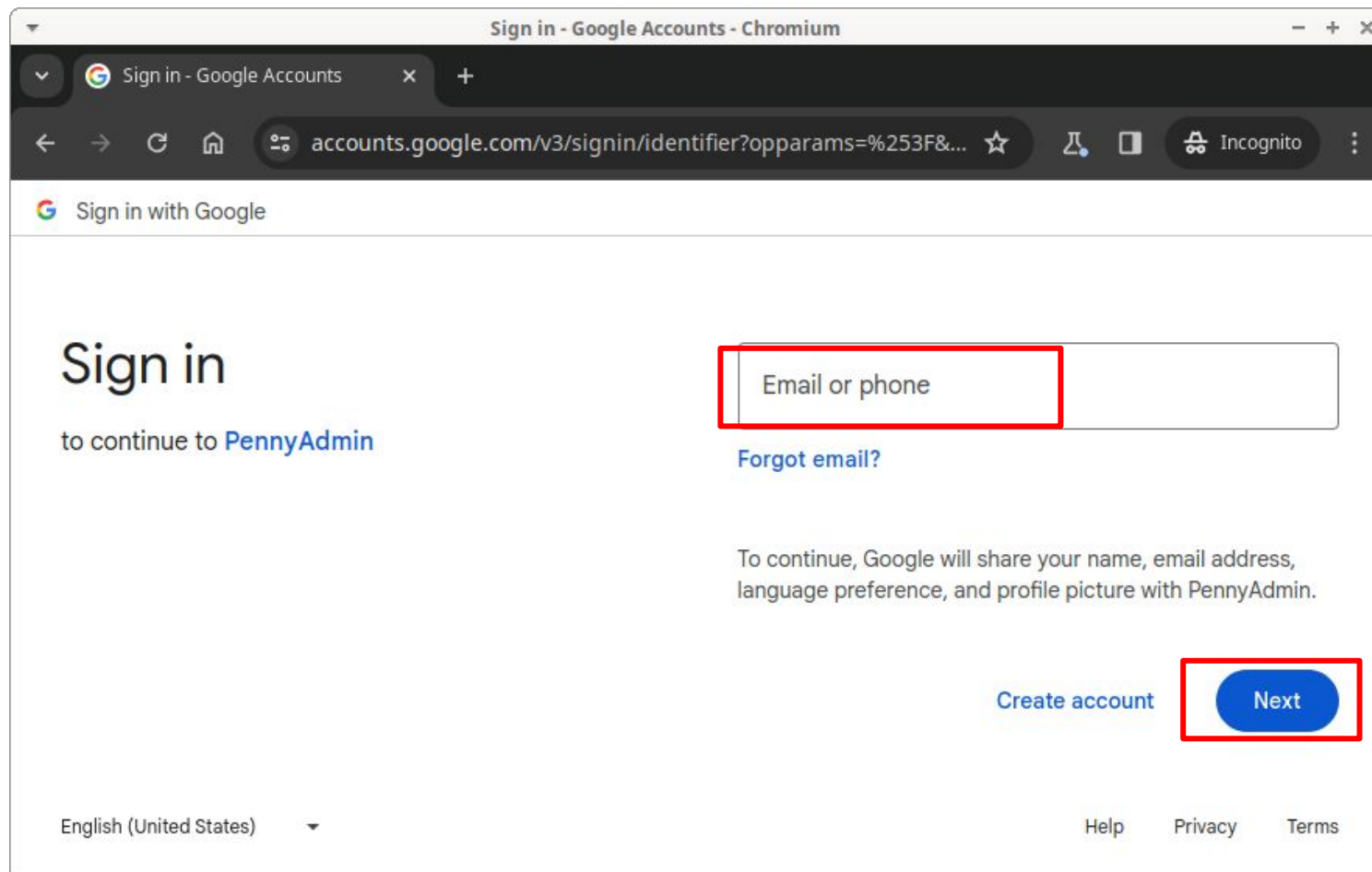
Google Authentication

- See **PennyAdminGoogle** app (cont.)



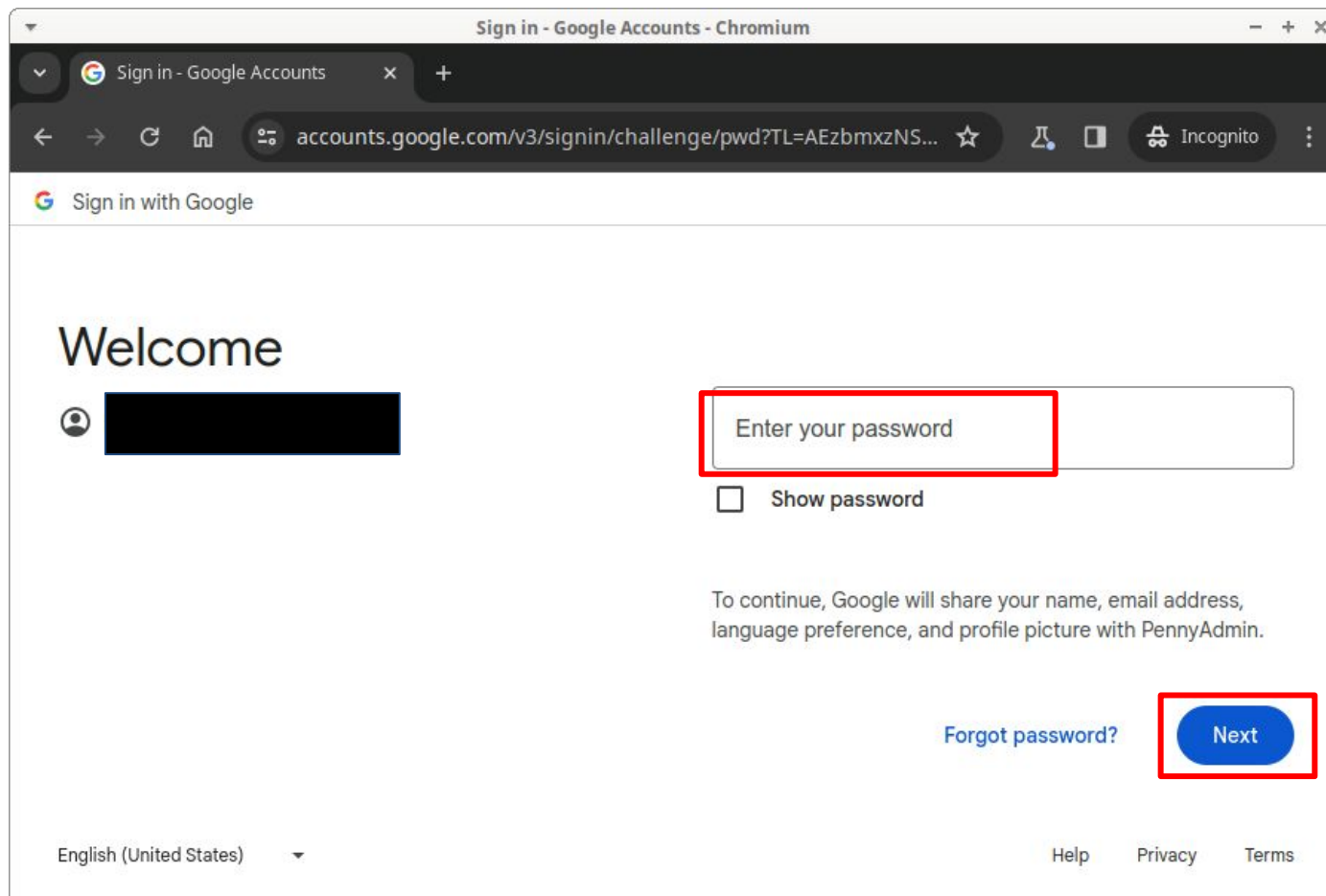
Google Authentication

- See **PennyAdminGoogle** app (cont.)



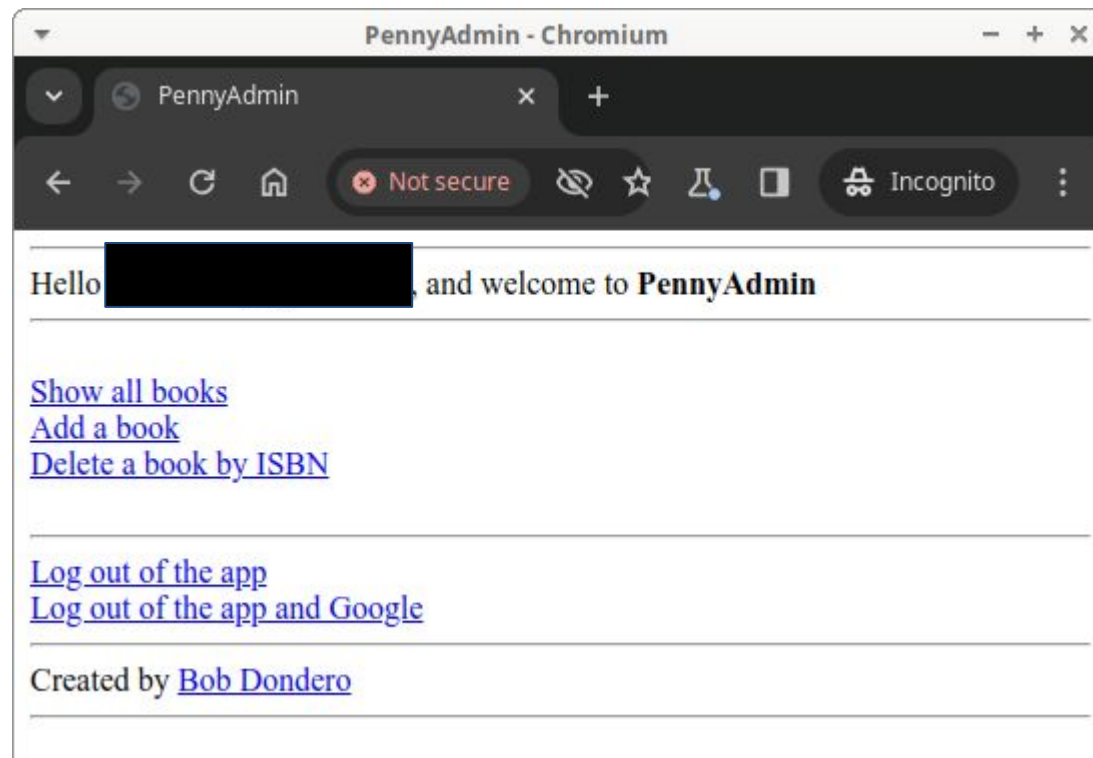
Google Authentication

- See **PennyAdminGoogle** app (cont.)



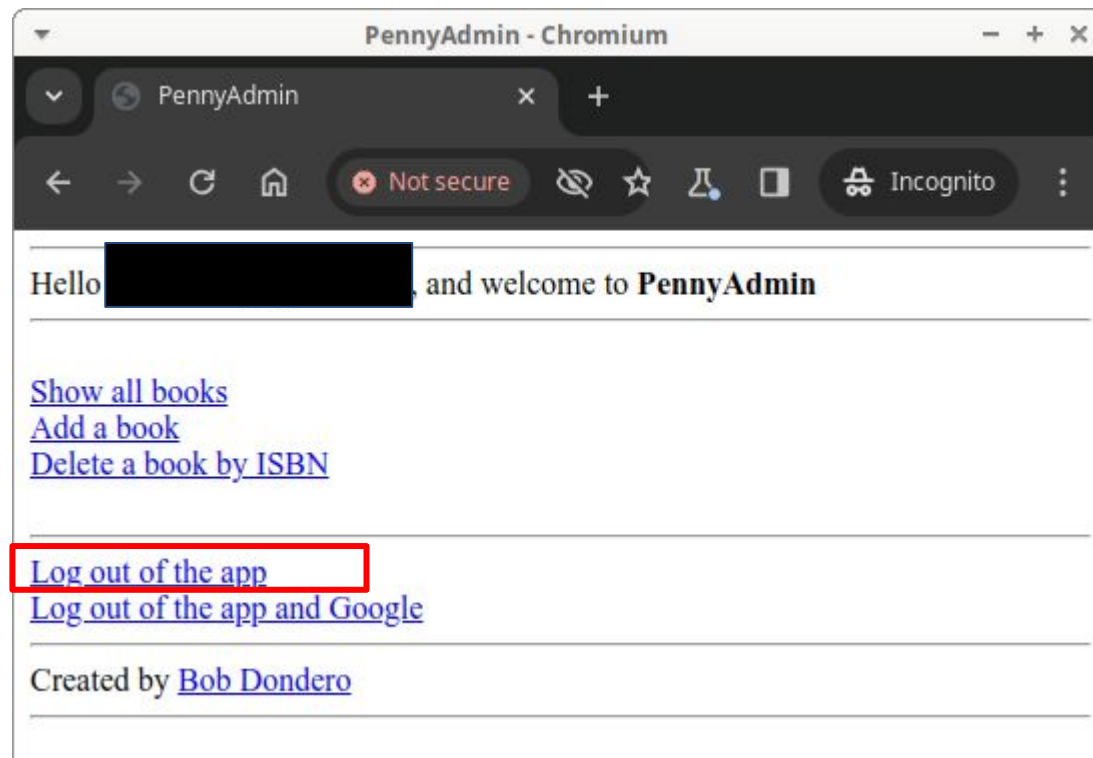
Google Authentication

- See **PennyAdminGoogle** app (cont.)



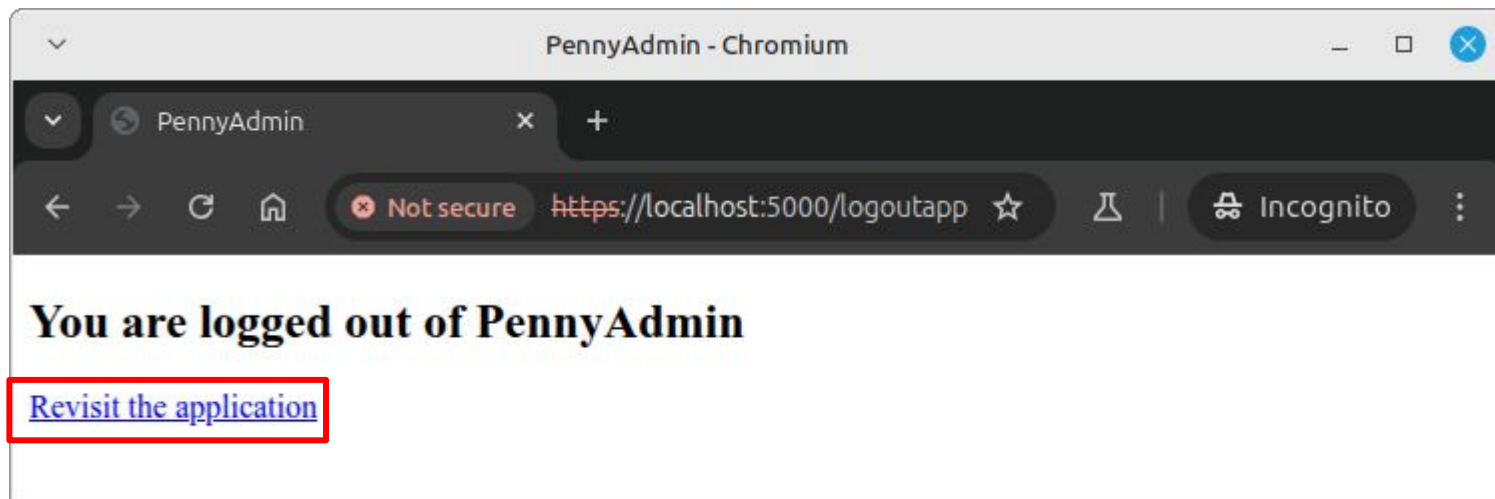
Google Authentication

- See **PennyAdminGoogle** app (cont.)



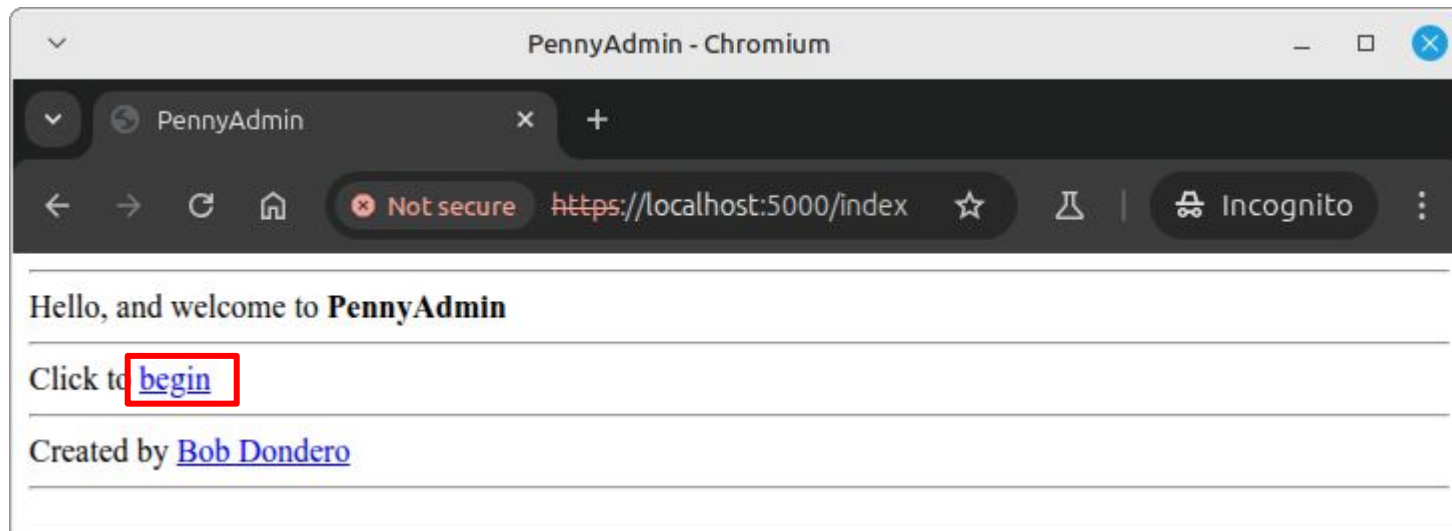
Google Authentication

- See **PennyAdminGoogle** app (cont.)



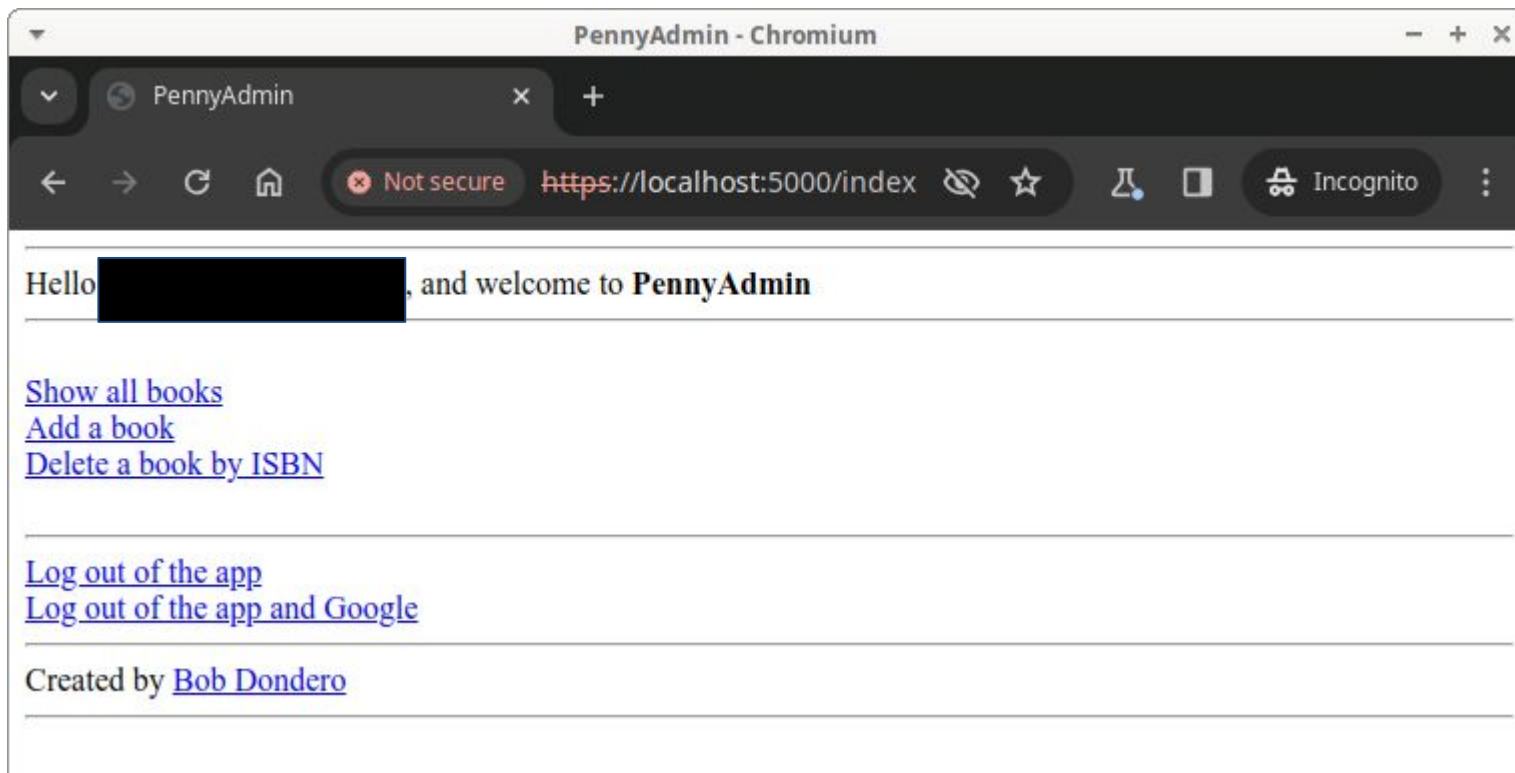
Google Authentication

- See **PennyAdminGoogle** app (cont.)



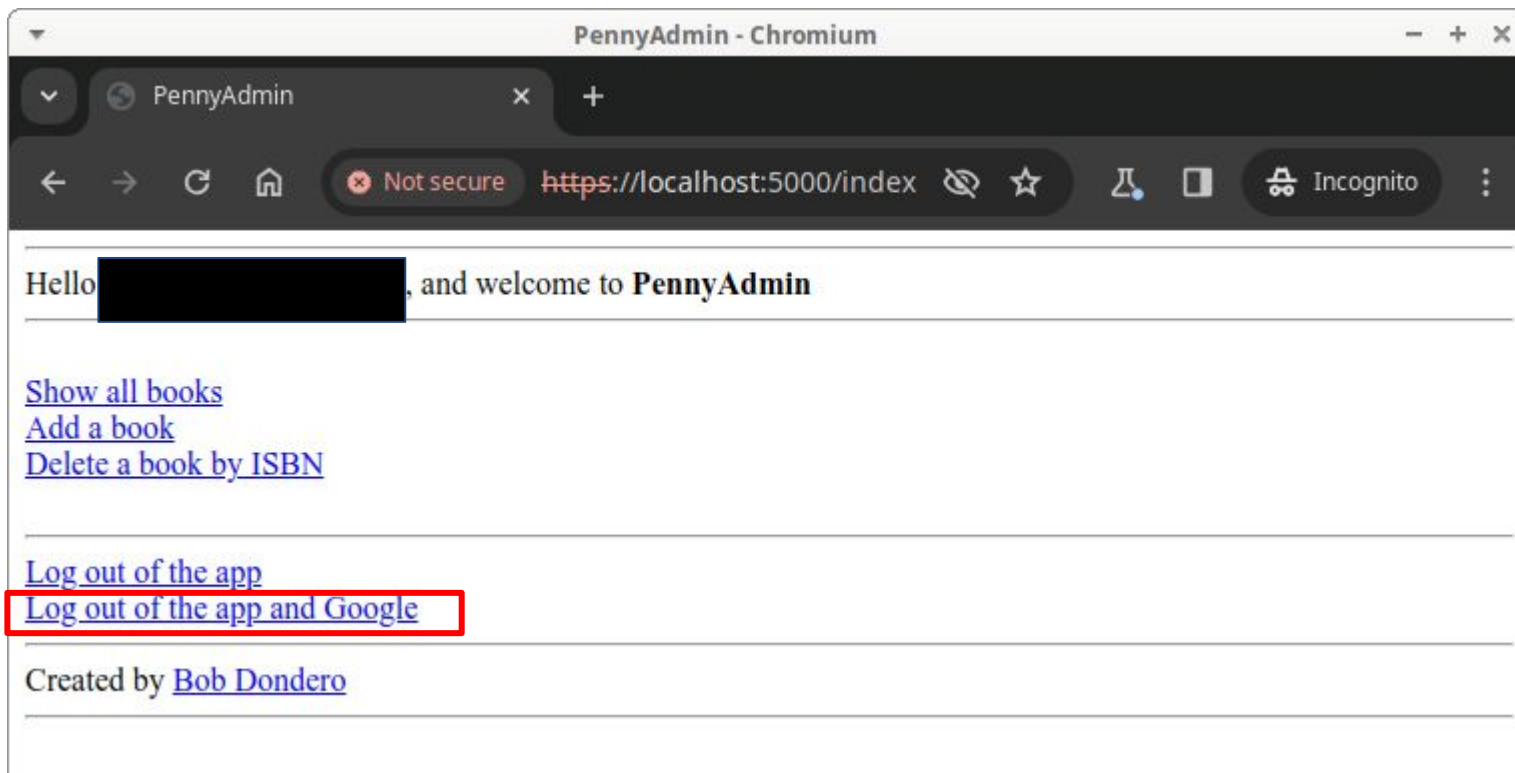
Google Authentication

- See **PennyAdminGoogle** app (cont.)



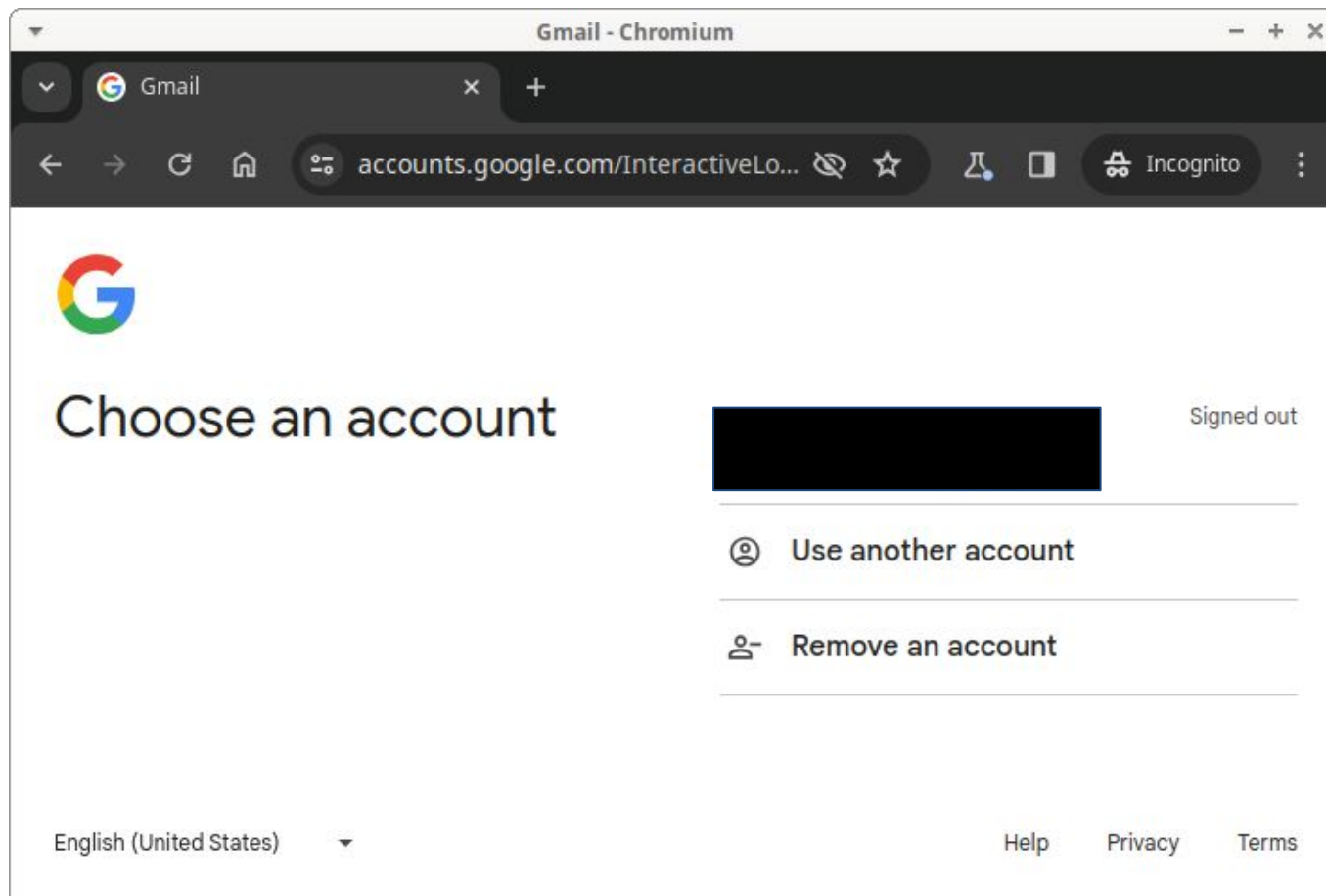
Google Authentication

- See **PennyAdminGoogle** app (cont.)



Google Authentication

- See **PennyAdminGoogle** app (cont.)



How to
show
loggedout
page?

Google Authentication

- See **PennyAdminGoogle** app (cont.)
 - See optional lecture material for:
 - The code
 - How to run it on your local computer
 - How to run it on Render
 - How it works

Google Authentication

- **Pros**

- Users need not remember (yet another) password
- Application need not manage usernames or passwords
- Application ***cannot*** access passwords
- Application can access profile info that user provided to Google
 - Given name, family name, picture, ...

Google Authentication

- **Cons**

- Complex
- Adds overhead, but mostly only during first user visit per browser session
- Application is constrained to users who have Google accounts
- Must use HTTPS with local server
- **If attacker learns user's password for Google, then attacker learns user's password for your app**

Lecture Summary

- In this lecture we covered:
 - Data storage attacks
 - Data comm attacks
 - Third-party authentication (briefly)
 - CAS authentication
 - Microsoft Entra ID authentication
 - Google authentication

Lecture Series Summary

- In this lecture series we covered:
 - SQL injection attacks
 - Cross-site scripting (XSS) attacks
 - Authentication & authorization
 - Cookie forgery attacks
 - Cross-site request forgery (CSRF) attacks
 - Data storage attacks
 - Data comm attacks
 - Third-party authentication (briefly)

More Information

- The COS 333 *Lectures* web page provides references to supplementary information