CEN 5016: Software Engineering

Fall 2025

University of Central Florida

Dr. Kevin Moran

Week 8 - Class I: Software Security + Midterm Exam Review

Administrivia

- *Assignment 4*
  - Due Monday, October 13th
  - Exploring Static Analysis Tools and CI with a simple Python app
  - Accept the Assignment on GitHub Classroom
- *SDE Project Part 2*
  - Due Tuesday, October 15th (updated deadline!)
  - You should have already received feedback on your plan!
  - Two parts:
    - Process & Implementation Snapshot
    - Checkpoint Presentation

Midterm Exam Format

- 2 Parts, In-class exam, closed book, 200 points total
  - ***Part 1:*** Multiple Choice
    - 12-15 questions
    - Will test basic knowledge of concepts, select the best answer for each question
  - ***Part 2:*** Short Answer Questions
    - 4-5 questions
    - Concepts from class, SE scenarios, answer in a paragraph
  - Covers material from Weeks 1-7
  - You will have the ***entire*** class period to complete the exam
  - Please bring your UCF ID to the exam

Example Multiple Choice Questions

- Which of the following is NOT a tenant of Agile?
  - (a) Incremental Design/Development
  - (b) Inspect and Adapt Cycles
  - (c) Ignoring the Customer
  - (d) Collaborative workflows
- What is the name of the concept where someone looks for something where they think it will be?
  - (a) the spotlight effect
  - (b) the streetlight effect
  - (c) The candle effect
  - (d) the software effect

Example Short Answer Questions

- *Consider the following scenario: You are working on a development team that seems to have a lot of issues with reoccurring bugs in your codebase. Describe some concepts from class that might aid in this situation. Be sure to use at least two separate concepts.*

Security & Privacy

Security Requirements for Web Apps

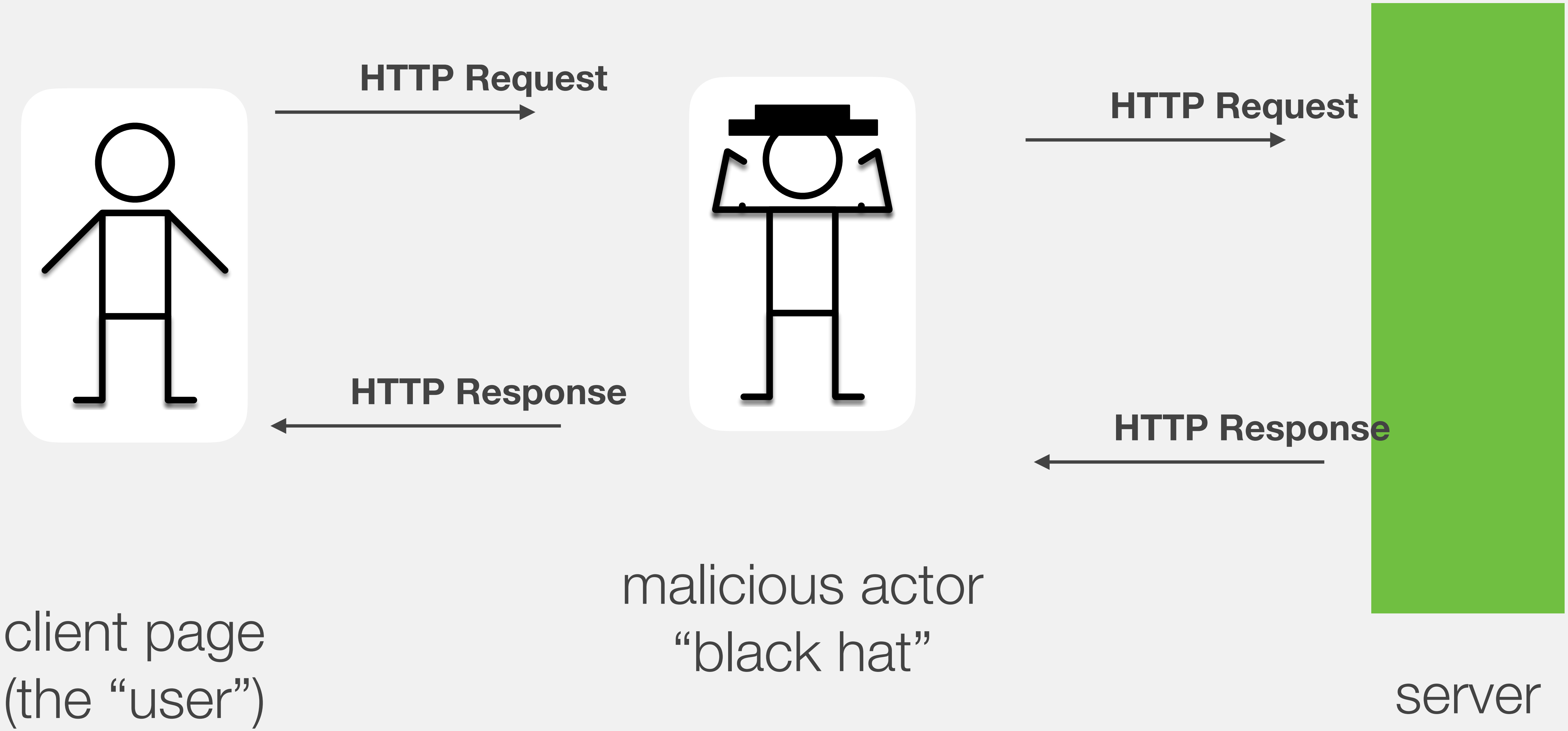
1. Authentication
    - Verify the ***identify*** of the parties involved
    - Who is it?
2. Authorization
    - Grant ***access*** to resources only to allowed users
    - Are you allowed?
3. Confidentiality
    - Ensure that ***information*** is given only to authenticated parties
    - Can you see it?
4. Integrity
    - Ensure that information is ***not changed*** or tampered with
    - Can you change it?

Security Requirements for Web Apps

- What is being defended?
  - What resources are important to defend?
  - What malicious actors exist and what attacks might they employ?

- Who do we trust?
  - What entities or parts of system can be considered secure and trusted
  - Have to trust **something**!

Web Threat Models: Big Picture



HTTP Request

HTTP Response

HTTP Request

HTTP Response

client page
(the "user")

malicious actor
"black hat"

server

Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

Security Requirements for Web Apps

1. Authentication
   - Verify the ***identify*** of the parties involved
   - Threat: Impersonation. A person pretends to be someone they are not.
2. Authorization
3. Confidentiality
   - Ensure that ***information*** is given only to authenticated parties
   - Threat: Eavesdropping. Information leaks to someone that should not have it.
4. Integrity
   - Ensure that information is ***not changed*** or tampered with
   - Threat: ***Tampering***.

Man in the Middle

- Requests to server intercepted by man in the middle
  - Requests forwarded
  - But… response containing code edited, inserting malicious code
- Or could
  - Intercept and steal sensitive user data

HTTPS: HTTP over SSL

- Establishes secure connection from client to server
  - Uses SSL to encrypt traffic
- Ensures that others can’t impersonate server by establishing certificate authorities that vouch for server.
- Server trusts an HTTPS connection iff
  - The user trusts that the browser software correctly implements HTTPS with correctly pre-installed certificate authorities.
  - The user trusts the certificate authority to vouch only for legitimate websites.
  - The website provides a valid certificate, which means it was signed by a trusted authority.
  - The certificate correctly identifies the website (e.g., certificate received for “https://example.com" is for "example.com" and not other entity).

Using HTTPS

- If using HTTPS, important that all scripts are loaded through HTTPS
    - If mixed script from untrusted source served through HTTP, attacker could still modify this script, defeating benefits of HTTPS
- Example attack:
    - Banking website loads Bootstrap through HTTP rather than HTTPS
    - Attacker intercepts request for Bootstrap script, replaces with malicious script that steals user data or executes malicious action

Authentication

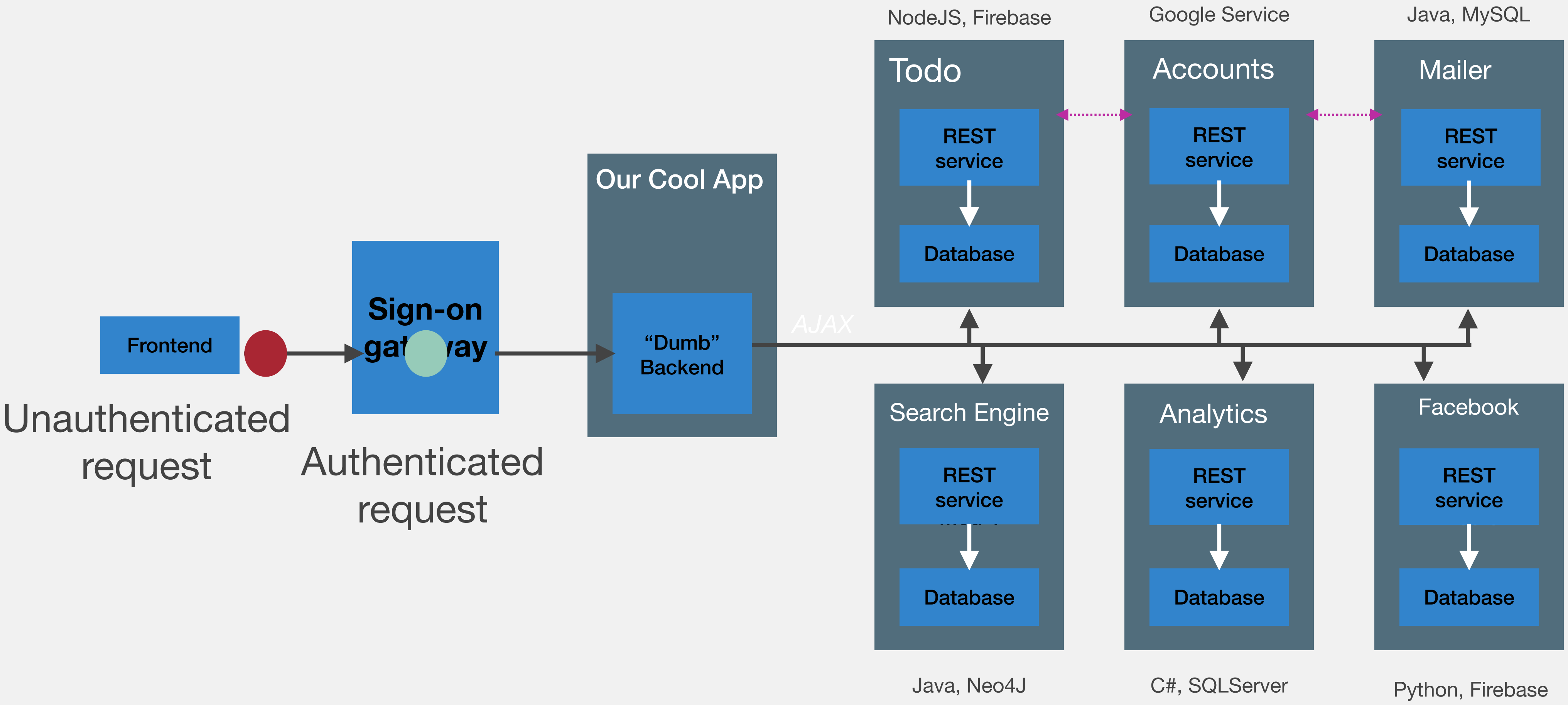
- How can we know the identify of the parties involved
- Want to customize experience based on identity
  - But need to determine identity first!
- Options
  - Ask user to create a new username and password
    - Lots of work to manage (password resets, storing passwords securely, …)
    - Hard to get right (#2 on the OWASP Top 10 Vulnerability List)
    - User does not really want another password…
  - Use an authentication provider to authenticate user
    - Google, FB, Twitter, Github, …

Authentication Provider

- Creates and tracks the identity of the user
- Instead of signing in directly to website, user signs in to authentication provider
  - Authentication provider issues token that uniquely proves identity of user

Sign On Gateway

- Can place some magic “sign-on gateway” before out app - whether it’s got multiple services or just one



Frontend

Unauthenticated request

Sign-on gateway

Authenticated request

Our Cool App

“Dumb” Backend

AJAX

NodeJS, Firebase

Todo

REST service

Database

Google Service

Accounts

REST service

Database

Java, MySQL

Mailer

REST service

Database

Search Engine

REST service

Database

Java, Neo4J

Analytics

REST service

Database

C#, SQLServer

Facebook

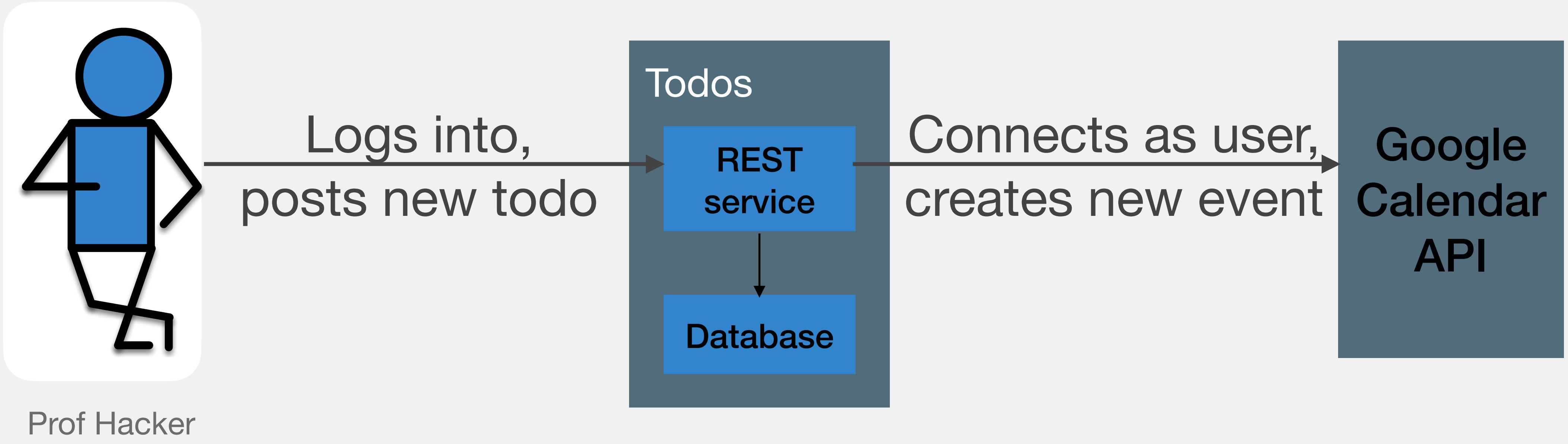
REST service

Database

Python, Firebase

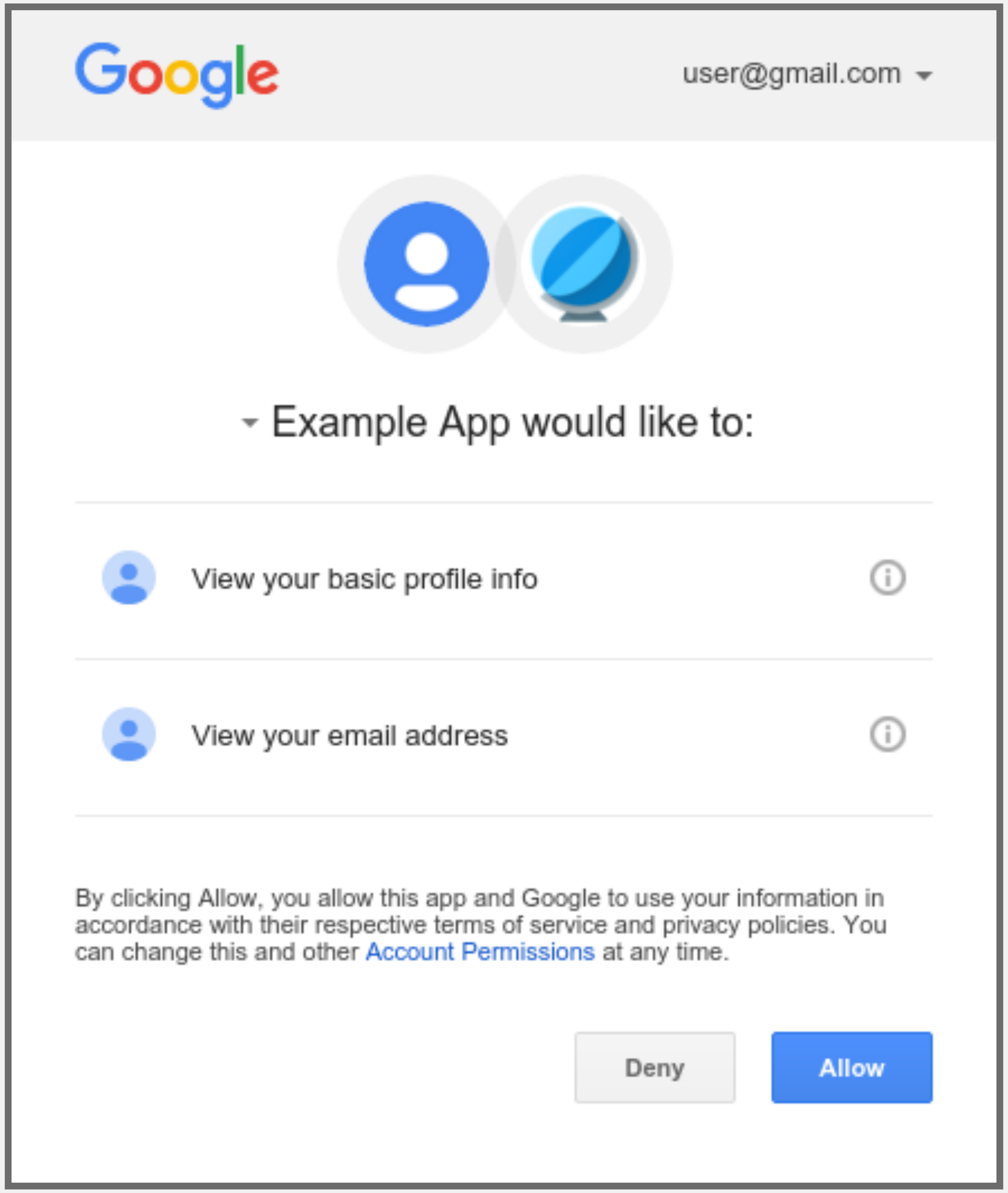
Authentication with Multiple Service Providers

- Let's consider updating a Todos app so that it can automatically put calendar events on a Google Calendar


Prof Hacker
Logs into, posts new todo
Todos
REST service
Database
Connects as user, creates new event
Google Calendar API

How does Todos tell Google that it's posting something for Prof Hacker? Should Prof Hacker tell the Todos app her Google password?

We've Got Something for that



Example App would like to:

View your basic profile info

View your email address

By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

OAuth

- OAuth is a standard protocol for sharing information about users from a “service provider” to a “consumer app” ***without*** them disclosing their password to the consumer app
- 3 key actors:
  - User, consumer app, service provider app
  - E.x. “Prof Hacker,” “Todos App,” “Google Calendar”
- Service provider issues a ***token*** on the user’s behalf that the consumer can use
- Consumer holds onto this token on behalf of the user
- Protocol could be considered a conversation…

Top 3 Web Vulnerabilities

- OWASP collected data on vulnerabilities
  - Surveyed 7 firms specializing in web app security
  - Collected 500,000 vulnerabilities across hundreds of apps and thousands of firms
  - Prioritized by prevalence as well as exploitability, detectability, impact

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

#3 - XSS: Cross Site Scripting

- User input that contains a *client-side* script that does not belong
    - A todo item:

```
/><script>alert("LASAGNA FOR PRESIDENT”);</script>
```

- Works when user input is used to render DOM elements without being escaped properly
- User input saved to server may be served to other users
    - Enables malicious user to execute code on other’s users browser
    - e.g., click ‘Buy’ button to buy a stock, send password data to third party, …

#2 - Broken Authentication and Session Management

- Building authentication is hard
  - Logout, password management, timeouts, secrete questions, account updates, …
- Vulnerability may exist if
  - User authentication credentials aren't protected when stored using hashing or encryption.
  - Credentials can be guessed or overwritten through weak account management functions (e.g., account creation, change password, recover password, weak session IDs).
  - Session IDs are exposed in the URL (e.g., URL rewriting).
  - Session IDs don't timeout, or user sessions or authentication tokens, particularly single sign-on (SSO) tokens, aren't properly invalidated during logout.
  - Session IDs aren't rotated after successful login.
  - Passwords, session IDs, and other credentials are sent over unencrypted connections.

#1 - Injection

- User input that contains *server-side* code that does not belong
- Usually comes up in context of SQL (which we aren’t using)
  - e.g.,
  - `String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";`
- Might come up in JS in context of eval
  - `eval(request.getParameter(“code”));`
  - Obvious injection attack - don’t do this!