Objectives

- JSTL Review
- Designing for Security
- Lightning Talk
  - Feedback

JSTL Review

- What is the purpose of JSTL?
  - What is the goal of JSTL?
  - What is JSTL?
- What are some common tags in the JSTL Core library?
  - How do they work?
  - What do they do?

Use when appropriate in your project

Google's AI is writing poetry

- No.
  he said.
  "no," he said.
  "no," i said.
  "i know," she said.
  "thank you," she said.
  "come with me," she said.
  "talk to me," she said.
  "don't worry about it," she said.
- it made me want to cry.
  no one had seen him since.
  it made me feel uneasy.
  no one had seen him.
  the thought made me smile.
  the pain was unbearable.
  the crowd was silent.
  the man called out.
  the old man said.
  the man asked.
- he was silent for a long moment.
  he was silent for a moment.
  it was quiet for a moment.
  it was dark and cold.
  there was a pause.
  it was my turn.
- there is no one else in the world.
  there is no one else in sight.
  they were the only ones who mattered.
  they were the only ones left.
  he had to be with me.
  she had to be with him.
  i had to do this.
  i wanted to kill him.
  i started to cry.
  i turned to him.

http://www.androidauthority.com/google-ai-poetry-692231/

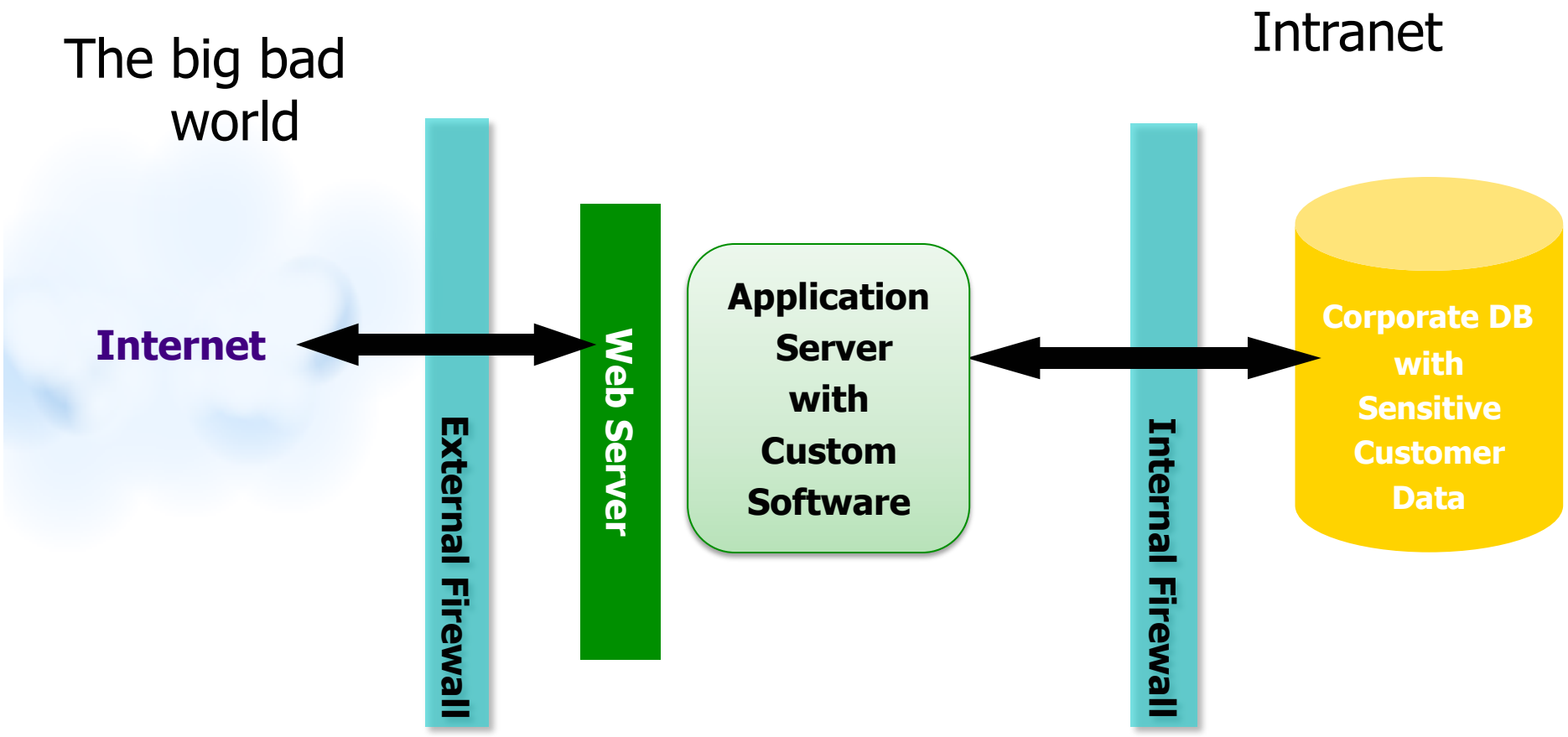
Security

"A few lines of code can wreak more havoc than a bomb."

--Tom Ridge
(Former) Secretary of the U.S. Department of Homeland Security

Discussion: Why is the Web a Target?

Web Application Architecture



"75% of cyber attacks and Internet security violations are generated through Internet Applications" - Gartner Group

Why is the Web a target?

- New environments: lots of vulnerabilities
  - Constantly new technologies to exploit
- Many different targets
  - Web browsers, Web servers, databases, personal computers, network traffic, humans
- Efficient, lucrative
  - Can get a lot done quickly
  - Personal info, credit card info, databases
  - Businesses: could lose millions of dollars/hour

What are some Web vulnerabilities?
What are ways to address those vulnerabilities?

2013 OWASP Top 10 https://www.owasp.org/index.php/Top_10_2013-Introduction

Most Critical Vulnerabilities

Open Web Application Security Project

Injection	←
Broken Authentication and Session Management	
Cross-Site Scripting (XSS)	←
Insecure Direct Object References	
Security Misconfiguration	
Sensitive Data Exposure	
Missing function level access control	
Cross-site request forgery (CSRF)	
Using components with known vulnerabilities	
Unvalidated Redirects and Forwards	

Security Requirements

- Authentication
  - Verify the user's identity
- Authorization
  - Give access to resources only to allowed users
- Confidentiality
  - Ensure that information is passed only to allowed users
- Integrity
  - Ensure that data is not inappropriately changed

Security Design Principles

- Least Privilege
- Defense in Depth
- Secure Weakest Link
- Fail-safe Stance
- Secure By Default
- Simplicity
- Usability

Security Design Principle:

Principle of Least Privilege

- Give just enough permissions to get the job done
  - Minimize damage if compromised
- Common world example: Valet Keys
  - Can start the car but can't access glove compartment, trunk
- A web server should only be given access to set of HTML files that web server is to serve
  - If web server is compromised, attacker can only read HTML files

Security Design Principle:

Defense in Depth

- Also called redundancy / diversity
- Common world example: Banks
  - What security does a bank provide so that bad guys won't steal money?
- Passwords:
  - Require users to choose strong passwords
  - Monitor web server logs for failed login attempts
  - If 3 incorrect password attempts, lock account for some period of time

Discussion of Passwords

- What makes a good password?
  - To the software?
  - To a user?
- How do passwords and their restrictions affect software?
  - How do they affect those who try to break them?
- How often should users change their passwords?
  - What are the tradeoffs in frequency?

Security vs. Usability

- Hard to design passwords that are **both secure** and **easy to remember**
- If users have to change their password more than every 6 months, security decreases
  - Come up with schemes to make passwords easier to remember

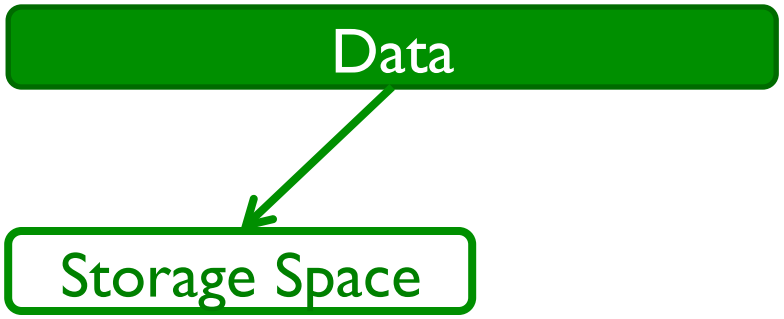
Security Design Principle:

Secure the Weakest Link

- Common Weak Links:
  - Weak Passwords - Crack
  - People - Social Engineering Attacks
    - "I am owed an inheritance; you can get a cut if you just give me your bank account number…"
  - Buffer Overflows
    - Slapper Worm: exploited OpenSSL, Apache to create peer-to-peer networks → DDOS
    - Not a problem in Java

Buffer Overflows

- An error caused when a program tries to store too much data in a temporary storage area
  - Exploited by hackers to execute malicious code
- Not an issue in Java, Python



Security Design Principle:

Fail-Safe Stance

- Even if 1 or more components of system fail, there is some security
- Common world example: Elevators
  - If elevator power fails, grip the cable
- System failure should be expected (and planned for)
  - If firewall fails, let **no** traffic in
  - Deny access by default

Security Design Principle:

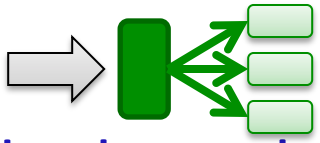
Secure By Default

- "Hardening" a system: All unnecessary services off by default
  - Only enable the 20% of the products features that are used by 80% of the user population
  - Ex: don't enable insecure networking apps by default
    - If choose to use those apps, know what you're doing
- More features enabled → more potential exploits → less security!

Security Design Principle:

Simplicity

- Complex software is more likely to have security holes
  - More code that wasn't tested thoroughly
  - More opportunities for buffer overflows, etc.
- Use **choke points** to keep security checks localized



  - centralized piece of code through which control must pass
- Generally: good rule of thumb to keep software simple

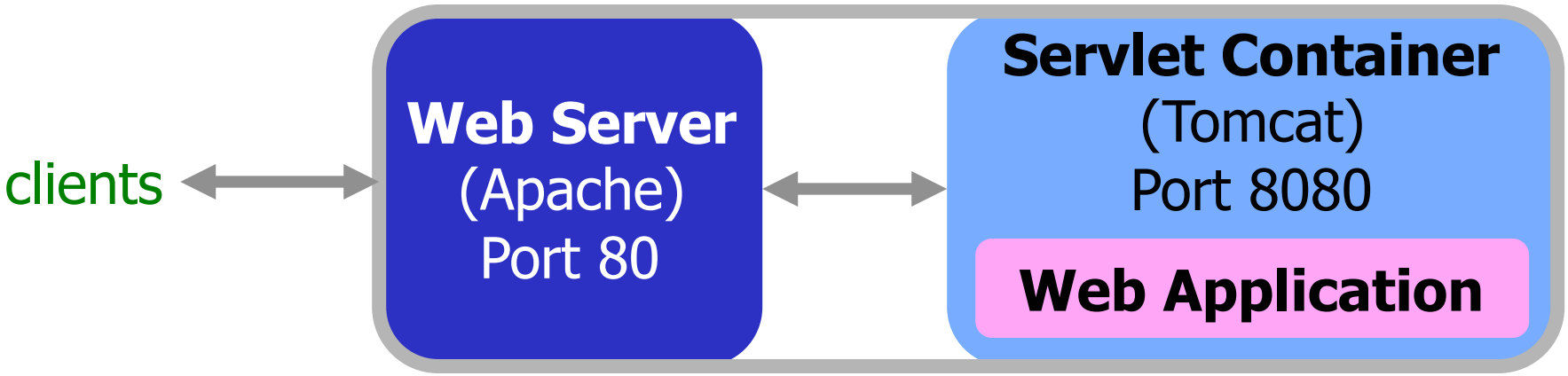
Security Design Principle:

Usability

- Users typically do not read documentation
  - Enable security by default
- Users can be lazy
  - Assume: They ignore security dialogs
- **Secure-by-default features in software forces users and vendors to be secure**
- More on usability in a bit…

Applying Security

- Can be applied at multiple levels:
  - Web server (e.g., Apache)
  - Servlet container (e.g., Tomcat, Resin)
  - Web application
- Each level offers different amounts of security and flexibility



Security Application Levels

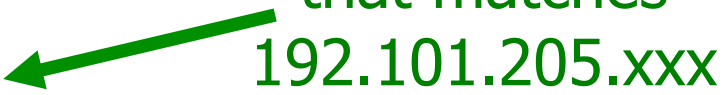
- Web server
  - HTTP authentication
  - Authorization of users/ groups
  - Authorization of domains
  - HTTPS: a secure version of HTTP
- Servlet container
  - HTTP authentication (basic, digest)
  - Form-based authentication
  - Authorization of users/ groups
  - SSL capabilities
- Application
  - Authorization of users
  - User information kept on server in a session

Apache's User-Level HTTP Passwords

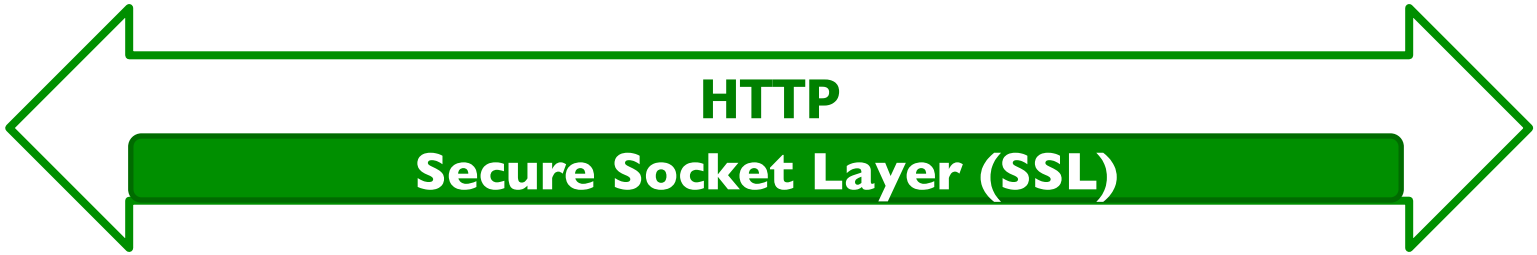
- Protect a Web-accessible directory with `.htaccess` file
  - Stored in the directory you want to protect
  - Special format
- Requires users to enter a username, password to access files in the directory
- Can also create authorized *groups*

Apache: Access Control

- Allow or deny requests by their domain
  - Syntax:
    `{allow/deny} from address`
    - `deny from 11.22.33.44`
    - `deny from hostname`
    - `deny from 192.101.205`
    - `deny from exampleone.com exampletwo.com`



HTTPS:
Hypertext Transfer Protocol Secure



- HTTP over Secure Socket Layer (SSL)
- Encrypt every HTTP message to and from the web server using standard PKI (public key infrastructure) technology
- De-facto standard used for secure web-based transactions
  - Otherwise, bad guys can get access to your session
- Default URL `https://some.domain.com` with default port number 443

Effective exception handling is essential in designing for security

CAREFUL EXCEPTION HANDING

Careful Exception Handling

- Error messages and observable behavior can tip off an attacker to vulnerabilities
  - Be careful not to provide more information to a user than is needed
- Exploit vulnerabilities using **Fault Injection**
  - Provide an application with input that it does not expect and observe its behavior
  - Bad guys use info to figure out application's vulnerabilities
    - Server may have fatal exception

Error Pages

- What you don't want
  - Show the Server's default exception page
  - Show the stack trace
    - Looks like your application is broken to user
- Want error page to look like the rest of your site
  - While you're debugging, that's useful,
    but when you're ready for deployment…

JSP Error Pages

- Defining error pages in deployment descriptor (web.xml)

Can have user-defined exceptions

```
<error-page>
      <exception-type>java.lang.Throwable</exception-type>
      <location>/error-page.jsp</location>
</error-page>
```

Can be a path to a servlet

- A JSP Error Page includes

```
<%@ page isErrorPage="true" %>
```

Allows use of exception object

- A JSP page can define its error page
  - All thrown exceptions go here, regardless of web.xml

```
<%@ page errorPage="errorPage.jsp" %>
```

TODO

- Project
  - Tues a.m. client meetings
    - 10:30: Logic
    - 11: AGP
  - Tues p.m. – screen shots from each project
    - For poster for Friday
  - Final implementation deadline: Fri – Spring Fest, noon-2
  - Bug analysis: Friday
  - Documentation: Saturday, 5 p.m.