Dept of Computer Science
Ahmadu Bello University

WEB APPLICATION ENGINEERING II

Lecture #8

Umar Ibrahim Enesi

Dept of Computer Science
Ahmadu Bello University

Objectives

- Understand the various principles to adhere to when designing web application
- Know the various best practices that help prevent various kind of attacks.

Dept of Computer Science
Ahmadu Bello University

Introduction to Web App. Security

- About 70% of attacks are on web application.
- $400 million estimate financial loss from many compromised records of various firms.

-2015 Verizon Data Breach Investigation Report

- *informationisbeautiful.net* for more statistical information on software breaches.
- Most organisations focus on spending more on firewalls and network security.
- But most breaches are caused by logic flaws in code at application level

Dept of Computer Science
Ahmadu Bello University

Introduction to Web App. Security (Cont.)

- Applying security features is important but not as important as writing secure features.
- Ultimately the best way of securing web applications is by taking into consideration well known security principles, adhering to standards and applying best practices throughout the life cycle of the application.
- Most of these principles are described in various standards organisations:
  - **OWASP** – Open Web Application Security Project
  - **WASC** – Web Application Security Consortium

Dept of Computer Science
Ahmadu Bello University

Security Principles

- There are various principles for securing web application:
  - Server-Side security principles
  - Client-Side security principles
  - Database security principles
  - File security principles
  - Development and deployment security principles

Dept of Computer Science
Ahmadu Bello University

SERVER-SIDE SECURITY PRINCIPLES

Access Control Mechanism

- A mechanism for regulating access to web resources.
- Web resources can be data and functionality
- Two processes are involved:
  - Authentication
  - Authorisation

Authentication

- Authentication provides a way for subjects to be identified.
  (subjects are actors such as persons, software, other web app. etc)
- An authentication is successful (authenticated) if the subject can prove it is what it claim to be.
- It is the first step in any access control mechanism
- Factors of authentication:
  - Knowledge factor
  - Inherent factor
  - Ownership factor

Factors of Authentication

- **Knowledge factor** :
  - Something you know
  - Eg password, PIN, pass phrase, security question etc
- **Ownership factor**:
  - Something you have
  - Eg ID card, credit card, security token, cell phone etc
- **Inherent factor**:
  - Something you are
  - Eg fingerprint, retina pattern, face detection etc

Factors of Authentication(cont.)

- A web application might utilize one or more of any of the factors during authentication:
  - Single-factor authentication: Only one of the factor is used for authentication
    - facebook and gmail uses only knowledge factor (username and password) for login into the application.
  - Two-factor authentication: authentication is performed using two factors
    - WhatsApp Web (login from mobile app and QR scan with mobile phone),
    - Accessing your account via ATM with pin and debit card
  - Three-factor authentication: authentication is performed using three factors

Authentication in Web Applications

- Almost all web application use password-based (knowledge factor) authentication mechanism.
  - HTTP Authentication:
    - Basic access authentication
    - Digest access authentication
    - Not secure , prone to man-in-the-middle attack.
  - Custom Authentication Systems:
    - Developer code their own logic to process and manage credentials.
    - Most common.
  - Single Sign-on Authentication:
    - Login in one application, access other independent/unrelated web application.
    - Authentication via third-party web application (eg Oauth)
    - More convenient (manage less credentials)
    - Single point of intrusion is an issue

Common Attacks Against Passwords

- **Dictionary attack:** search using words in dictionary
- **Brute Force attack**: key search using all possible set of allowed password.
- **Precomputed Dictionary Attack**: search using precomputed dictionary or brute force search keys.
- **Rubber-hose:** using physical coercion.

Dept of Computer Science
Ahmadu Bello University

Password Management Best Practices

- Require minimum password length (8 - 12 recommended)
- Enforce password complexity:
  - Allow for different chars
  - Enforce combination of characters. Eg. Numbers + Alphabets + special chars
  - Detect and reject weak passwords
- Enforce Rotation of passwords: force users to change password periodically
- Avoid password reuse by the same user.
- Password should never be the same as username or other profile info

Dept of Computer Science
Ahmadu Bello University

Password Management Best Practices

- Allow users to disable their account if they see the need to do so.
- Store password properly:
  - Secure database
- Use multiple hashing
- Include salt.

Authentication Best Practices

- Know when to perform authentication:
  - When access right change
  - The need to access protected resources or functionality
- Use secure transmission (SSL/TLS)
- Lock down account if anomalous activity is detected.
  - Temporary lock down account.
  - Limit the number of successive authentication failure
  - Use security questions
  - Use CAPTCHA

Authentication Best Practices

- Use “Remember Me” feature with caution
- No hardcoding of credentials
- Allow accounts to be disabled by users

Dept of Computer Science
Ahmadu Bello University

Authorization

- Comes second after Authentication
- It is the process of determining, using policy data, whether a subject has sufficient permission to perform certain operation on a web resource.
- Authorization (and Authentication) greatly rely on session.
- Authorisation ensures that users only perform actions within their privilege level.

Dept of Computer Science
Ahmadu Bello University

Review of Session

- A mechanism to track a single user's interaction with a web application.
- At the start of a session, browser is assigned a unique token.
- The browser presents the token along with each subsequent request.
- Information stored during the interaction constitute the session state.
- At the end to the session, the token (along with the session state) is destroyed.

Dept of Computer Science
Ahmadu Bello University

Session State Storage Strategies

- **Cookies** – not secure, limited in size
- **Client Side Storage** – not secure, limited in support (HTML5)
- **Hidden form field/URL Parameters** – not secure
- **Server side storage** – may rely on cookies or URL parameters.

Dept of Computer Science
Ahmadu Bello University

Attack Against Session

- **Tampering (Cookie Poisoning)**: An attacker modifies the session state from the client in order to lie to the server.
- **Session Theft**: Stealing the session id so that it can be used to impersonate the authenticated user.
- **Session Hijacking**: Stealing the session of an authenticated user and preventing the user from using the session.
- **CSRF**: fooling the browser to make request to a URL.

Session Best Practices

- **Enforce absolute session timeout**: Set a strict maximum lifetime for every session (1-4 hours recommended).
- **Enforce idle session timeout**: explicitly log out users that are idle (10 -20 minutes recommended).
- **Limit session concurrency**: Users should be limited to the number of sessions that can be used at a time.
- Use `Secure` flag when setting cookies
- Use `HttpOnly` flag only when setting cookies
- Session IDs should be unpredictable: apply strong cryptography

Session Best Practices

- Invalidate session in the server
- Destroy all session related cookies in the client
- Delete all session related cookies when the browser closes
- Err on the practice of storing most session state in the server
- Make the logout functionality easy to access in the UI
- Never reuse Session IDs

Attacks Against Authorisation

- **Forceful Browsing**: Accessing a URL that is not supposed to be accessed directly.
  - Careless assumption in the design process.
  - Not authorising every request.
- **Input/Parameter tampering**: modifying input values before it gets to the server.
  - Tools like firebug can be used to modify input values
- **Header Manipulation**: Trusting header field values can lead to security breaches
- **CSRF:** Tricking the browser(and user) into making unintended requests to a certain URL.

Authorisation Best Practices

- **Fail securely:**
  - If anything goes wrong, fall into a state that is safe from attack.
  - Invalidate users session and redirect to homepage
  - Force users to restart the process.
  - Hide application/database errors from users
- **Operate with least privileges:**
  - Web app should run as a restricted user.
  - Users should have just enough privilege to perform an operation.
- Perform authorisation on every request
- Centralize authorisation code
- Never trust header field values
- Avoid client side authorisation credentials (cookies and the like)

Other Good Practices

- Input Validation:
  - Use whitelist (not blacklist) validation technique
  - Use regular expressions for complex validation
    - regexlib.com
  - Always validate on the server side before use
- Attack Surface Reduction:
  - Disable seldom-used or noncritical features by default
  - Allow users to opt in/out of extra features
  - Always use principle of least privileges

CLIENT-SIDE SECURITY PRINCIPLES

Dept of Computer Science
Ahmadu Bello University

Same Origin Policy

- It is a policy agreement by browser manufacturers.
- Essentially, same-origin policy states that scripts running on a web page should be able to read from and write to the content of another web page if both pages have the same ***origin***.
- Origin of a web page is determined by:
  - Application Layer Protocol eg HTTP, HTTPS, etc
  - TCP Port eg 80, 8080
  - Domain Name
- It is important to note that this restriction is only effective on the client side – code at the server side can access other domains

Dept of Computer Science
Ahmadu Bello University

Exceptions to Same-Origin Policy

`<script src="">...</script>`
- Loading third-party script library can be potentially dangerous

`<iframe src="">...</iframe>`
- Scripts cannot access contents of iframe and vice versa
- But scripts in both ends can agree to explicitly lower their domain value
- Lowering of domain value is only possible if both pages are from sub-domain of the same root domain

- AJAX with CORS
  - Set the response header field "`Access-Control-Allow-Origin`"
  - Set the response header field "`Access-Control-Allow-Credentials`"
  - IE 8 and later: `XDomainRequest`

Dept of Computer Science
Ahmadu Bello University

Same-Origin Best Practices

- Load third-party scripts with caution.
- Avoid using javascript `eval()` function to parse JSON strings
- Consider using trusted libraries like JQuery to parse JSON strings
- Lower web page domain value with caution.
- `XDomainRequest` object is safer than CORS but limited to IE

Dept of Computer Science
Ahmadu Bello University

Cross-Site Scripting (XSS)

- XSS is a vulnerability that allows an attacker to include his own script code into a web applications pages.
- An attacker may also inject HTML code into vulnerable web pages.
- **Root cause**: web application accepts user's input and display it as-is.

XSS Attack Prevention

- Encode output
  - Eg. htmlSpecialCharacters()
- Sanitise input
- Use reduced mark-up language (checkout Wikipedia edit page)
- Set sensitive cookie with `'HttpOnly'` flag
- Apply Mozilla's Content Security Policy (CSP)

Cross-Site Request Forgery (CSRF)

- When a user is authenticated, trust exist between the user and the web application.
- CSRF is a situation where an attacker exploits the trust between users and web application.
- An attacker inject his content in a way that the browser assumes it comes from the web server. The web server may assume all input is from the user.
- These contents may contain code that causes the browser to make HTTP request to web server.

CSRF Prevention

- Rely on POST where necessary

DATABASE SECURITY PRINCIPLES

Attack on Database - SQL Injection

- SQL injection is one of the easiest vulnerabilities to discover and exploit by attackers.
- SQL injection mostly target the integrity of the application data.
- **Method**: Attacker will supply data in a way that it changes the execution path of the query. Consequently the query will do something it was never intended to do.
- This can be laborious but attackers actually use automated tools to exploit databases.

SQL Injection Prevention

- Avoid outputting error messages to users. Use generic message
- Validate every input and return appropriate error message to the user.
- Use parameterised queries to escape input.
- Apply principle of least permission on database account
- Use stored procedures and triggers.
- Avoid single database account security
- If role-based authorisation is employed, create different account for different roles.

Dept of Computer Science
Ahmadu Bello University

Attack on Database -Insecure Direct Object Reference

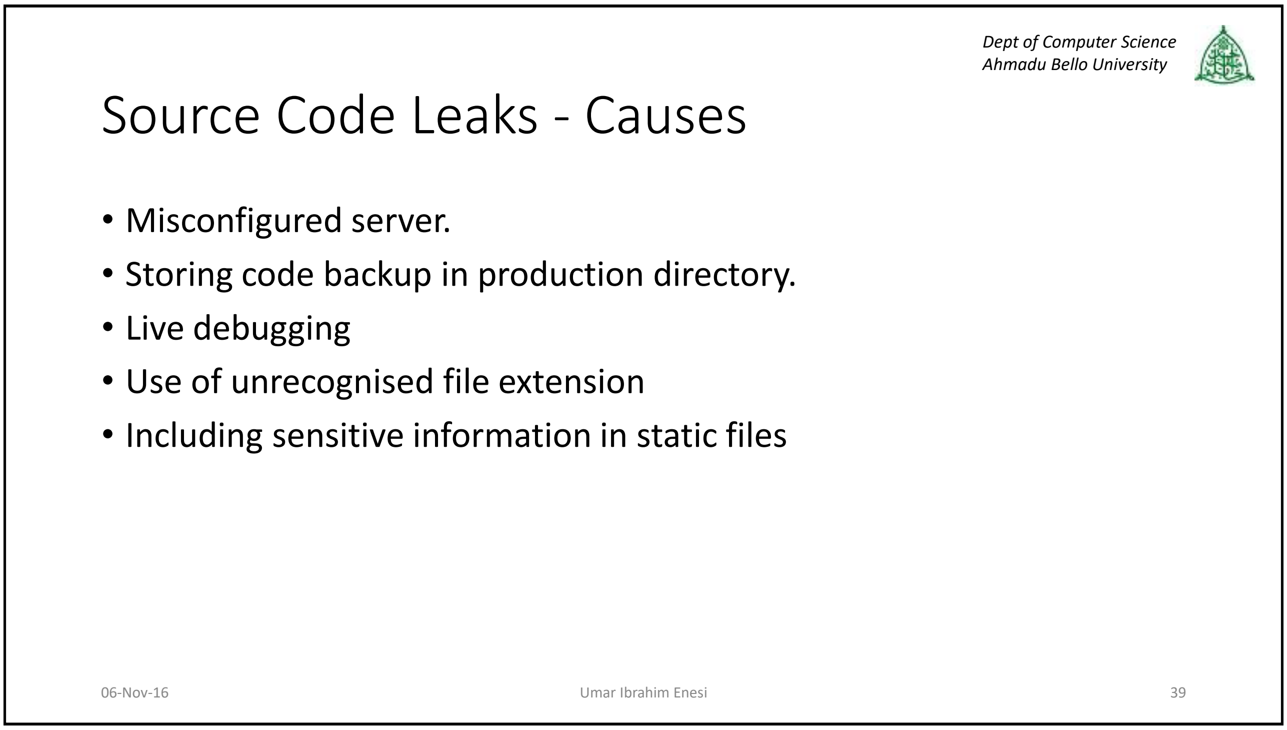
- This can occur when there is an authorisation flaw that can compromise data integrity.
- An authorised user can tweak browser's query parameter and gain access to sensitive data.

Prevention:

- Pre- or Post- request authorisation check.

Dept of Computer Science
Ahmadu Bello University

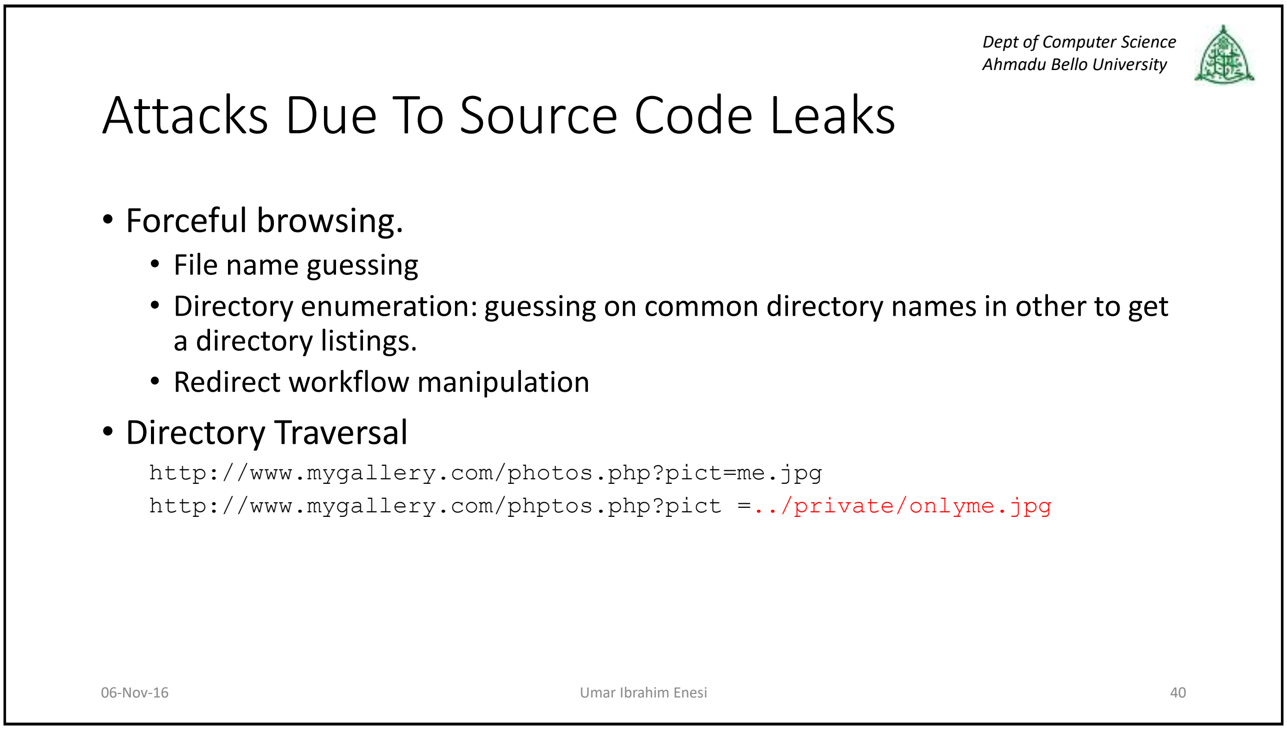
FILE SECURITY PRINCIPLES



Dept of Computer Science
Ahmadu Bello University

Source Code Leaks - Causes

- Misconfigured server.
- Storing code backup in production directory.
- Live debugging
- Use of unrecognised file extension
- Including sensitive information in static files



Dept of Computer Science
Ahmadu Bello University

Attacks Due To Source Code Leaks

- Forceful browsing.
  - File name guessing
  - Directory enumeration: guessing on common directory names in other to get a directory listings.
  - Redirect workflow manipulation
- Directory Traversal

```
http://www.mygallery.com/photos.php?pict=me.jpg
http://www.mygallery.com/phptos.php?pict =../private/onlyme.jpg
```

Preventing (the effect of) Source Code Leak

- Avoid hard-coding sensitive credentials (eg database credentials) in the source code.
- Never keep backup copies in production directory.
- Include files should be named with the same extension as main source code files.
- Keep no sensitive information or logic in static files/directories.
- Configure web server to disable directory listings.
- Replace status code error messages with custom error pages.
- Always perform server-side check when strict workflow is required.

Key Point

- Many of software attacks is on web application.
- The best way to secure web application is by
  - Taking into consideration, various web security principles.
  - Adhering to standards,
  - Applying best practices.

Reference

- PHP Documentation Manual
- Murach PHP and MySQL
- http://www.w3schools.com/php/php_form_validation.asp
- http://info.varonis.com/web-security-fundamentals-course