جامعة الملك سعود
King Saud University
College of Computer & Information Sciences
Information Technology Department

INPUT VALIDATION AND AUTHENTICATION

Prepared by: Dr. Alia Alabdulkarim

Input Validation

- Never trust the user
  - They are the reason we create applications
  - We ask for their emails, credit cards,…etc.
  - They trust us not to spam them
- **Blacklist validation:** List all inputs that are invalid, then block anything that matches that list.
- **Whitelist validation:** List out and match only what should be allowed.

Blacklist validation

- List all inputs that are invalid, then block anything that matches that list.
- It is extremely difficult to list out every possible unwanted or malicious input value.
  - The list will change over time (and change often).
- Example: restaurant dress code
- Can you list all possible inappropriate clothing?

- No shorts
- No T-shirts
- No jeans
- No sweatpants
- No hoodies

- No shorts
- No T-shirts
- No jeans
- No sweatpants
- No hoodies
- No bare feet

Blacklist validation

- Example: apostrophe (‘).
- What if a user needs to use it.

%27	(URL encoded)
'	(HTML encoded)
'	(XML encoded)
'	(HTML hex encoded)
0x27	(UTF-8 hex)
0x0027	(UTF-16 hex)
0x00000027	(UTF-32 hex)
%2527	(double-URL-encoded)

Blacklist validation

- More difficult when we want to prevent the access of certain files or URLs.
- Consider the page: www.site.cxx/my page.html

- http://www.site.cxx/my page.html
- http://www.site.cxx/My Page.html
- http://www.site.cxx/MY PAGE.HTML
- http://www.site.cxx/my%20page.html
- http://www.site.cxx:80/my page.html
- http://www.site.cxx/./my page.html
- http://1.2.3.4/my page.html
- http://16909060/my page.html

Whitelist Validation

- Listing out and matching only what should be allowed.
- Any input that doesn't match the list or the pattern is rejected.
- How we can use whitelist validation in the restaurant example?

Whitelist Validation

- When using Drop-down lists or Radio buttons, do you need validation? Why?
- Example:
  - Car configuration application
  - Exterior color options: Midnight Blue, Sunset Red, Canary Yellow
  - Validation logic: compare these the input against those three options

```
POST /buildcar.php HTTP/1.1
Host: www.sportscar.cxx
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

exteriorColor=Midnight+Blue
```

```
POST /buildcar.php HTTP/1.1
Host: www.sportscar.cxx
Content-Length: 31
Content-Type: application/x-www-form-urlencoded

exteriorColor=Shimmering+Silver
```

Whitelist Validation

```
POST /buildcar.php HTTP/1.1
Host: www.sportscar.cxx
Content-Length: 143
Content-Type: application/x-www-form-urlencoded

exteriorColor=';EXEC+xp_cmdshell+'…'
```

Exploit the database access logic

Whitelist Validation

- It is impossible to defend the server-side logic of a web application by implementing defenses on the client side.
- Any validation logic that you put into client-side code can be completely bypassed by an attacker
  - Even when you constrain the user input through user interface objects (e.g. Dropdown lists)

Regular Expression (Regex)

- To validate a choice of color of a new car you can check the incoming value against a predefined list.
  - This method cannot be used with all types of inputs (e.g. email address)
- Regular expressions are good whitelist validation method to handle more complicated whitelist validation logic.
  - Email address
  - Phone number
- Client side validation using regular expression can improve performance.
  - But not only on the client side.

Regular Expression (Regex)

- Email:
  - ^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,63}$
- Date:
  - ^(19|20)\d\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])$

	Metacharacter	Metacharacter name	Meaning
1	^	caret	denote the beginning of a regular expression
2	$	Dollar sign	denote the end of a regular expression or ending of a line
3	[]	Square bracket	check for any single character in the character set specified in []
4	()	Parenthesis	Check for a string. Create and store variables.
5	?	Question mark	check for zero or one occurrence of the preceding character
6	+	Plus sign	check for one or more occurrence of the preceding character
7	*	Multiply sign	check for any number of occurrences (including zero occurrences) of the preceding character.
8	.	Dot	check for a single character which is not the ending of a line
9	|	Pipe symbol	Logical OR
10	\	Escaping character	escape from the normal way a subsequent character is interpreted.
11	!	Exclamation symbol	Logical NOT
12	{}	Curly Brackets	Repeat preceding character

More Validation Practices

- **Two important questions: What input to validate? AND Where to validate it?**
- **What input to validate?** any untrusted input. What should be considered untrusted?
  - Any input you get directly from a user.
    - Any attacker controllable input.
    - E.g. Web forms, Query strings parameters, Cookie values, Header value.
  - Any data you pull from your database.
    - Where did that data in the database come from? Can you trust it? (No)
    - User input, other company or organization
  - Real-World example: Asprox SQL injection worm.

More Validation Practices

- **Where is the best place to validate input?**
  - Right as it comes into the system.
    - Before it gets stored
  - Right before you use it.
- The defense-in-depth approach:
  - Use both options (as it comes in and right before it's used).
    - May impact performance

Attack Surface Reduction

- Attack surface = choke points = points of failure.
- **Attack surface:** all of your application code and functionality that can be accessed by any untrusted user.
  - Can't trust users, so simpler definition: all of your application code and functionality.
- Every time you add a new feature to your application ➔ adding a potential point of failure
- Don't remove features ➔ users love features. Instead, give them the option to activate\deactivate features.
- **Attack surface reduction:** an effective defense against the known attacks of today and a hedge against any new attacks that you might face tomorrow.

Attack Surface Reduction Rules of Thumb

- First: principle of least privilege
  - Give users minimum permissions
  - Perform authorization checks
  - Use different access accounts for different types of users
  - Doesn't impact legitimate users
- Second: minimize the capabilities of the programming calls and objects that you use in your code.
  - DataSet object vs. DataTable object

AUTHENTICATION

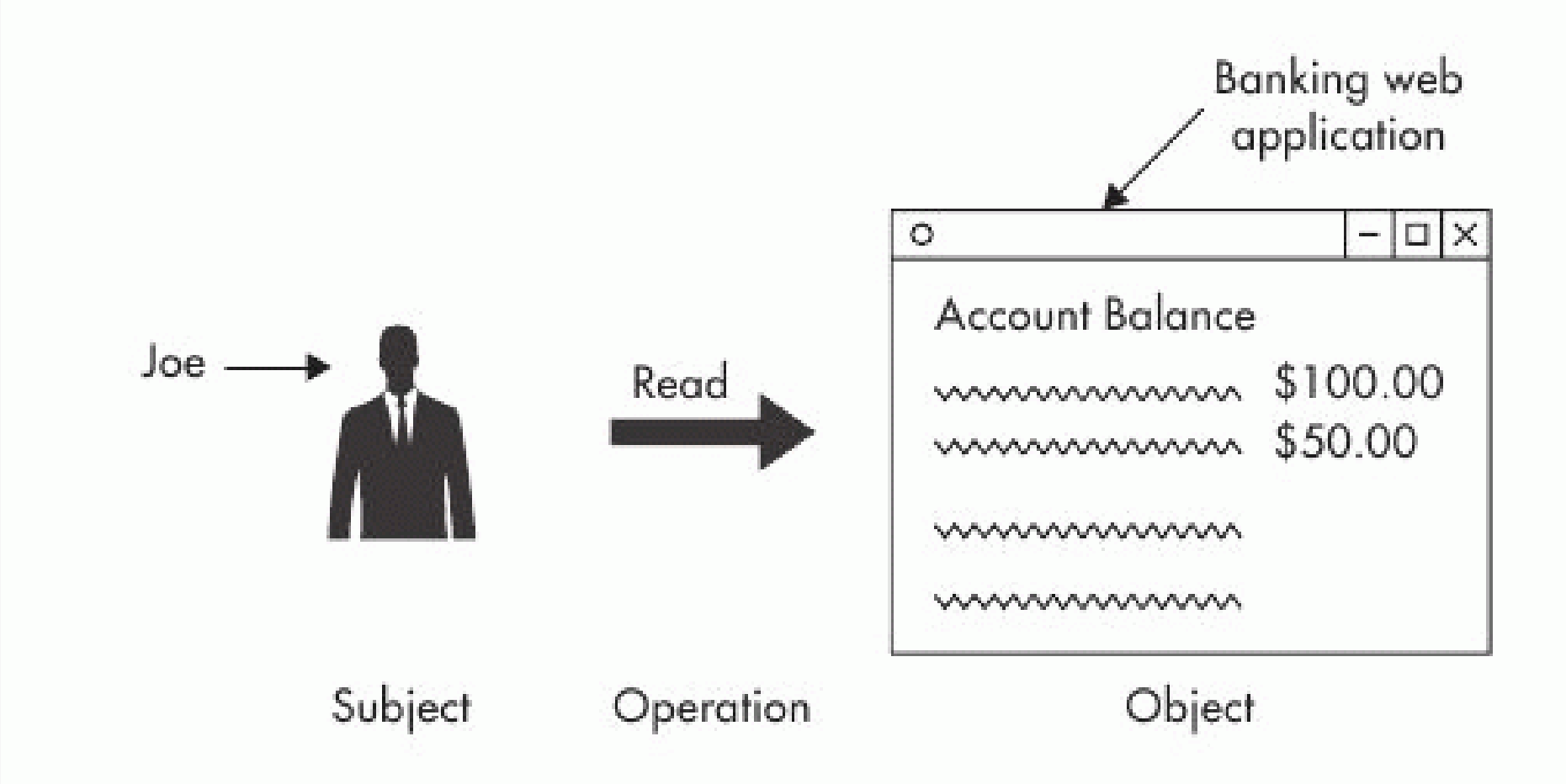
Access Control Overview

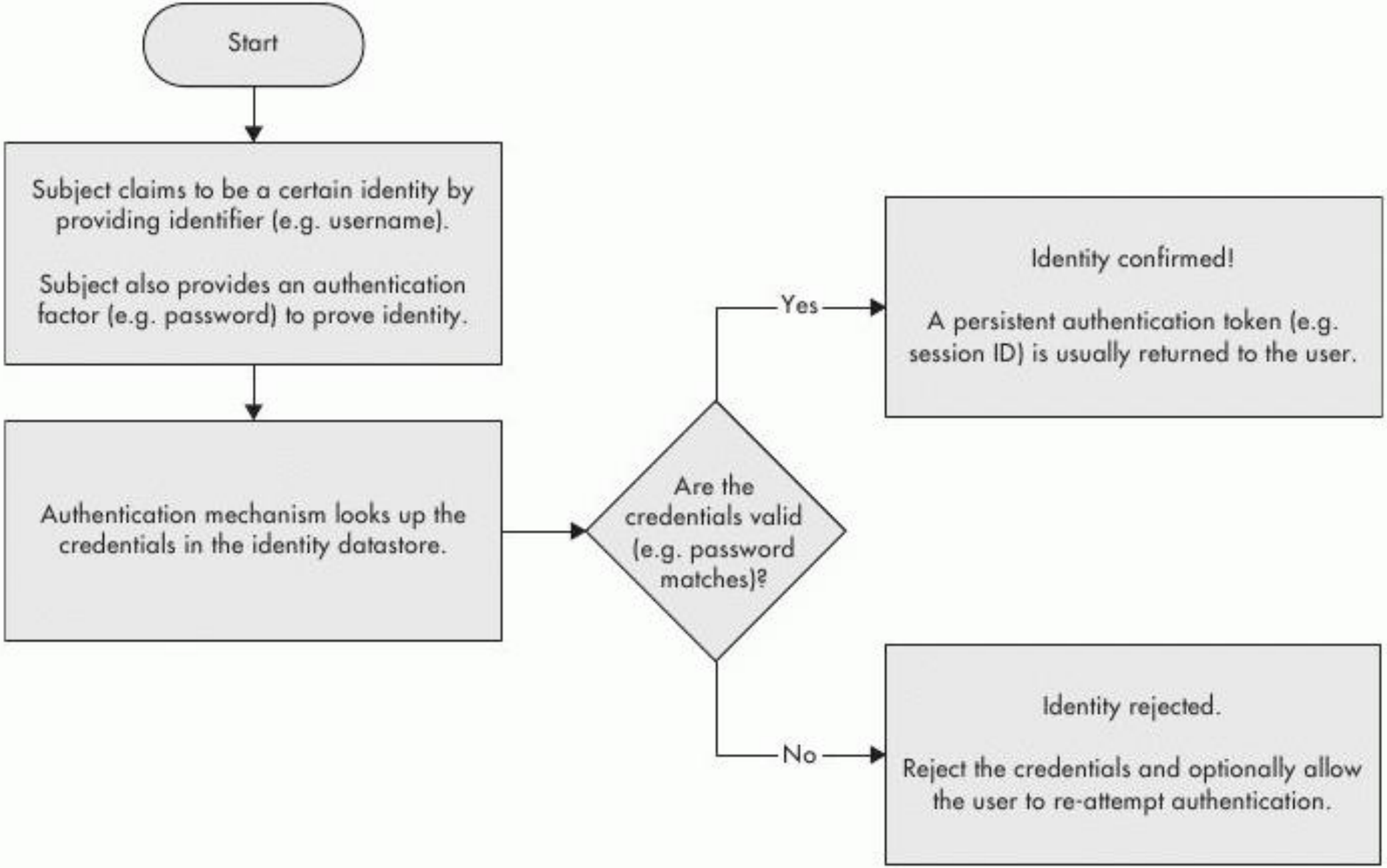
- In many web applications it is important that only certain users be permitted to access protected resources.
- Enforcing this kind of control means that you need to have a strong access control system.
- An ***access control*** is a mechanism that regulates access to data or functionality by determining whether a **subject** is permitted to perform an **operation** on a target **object**.

Access Control Overview

- Two related processes: Authentication and Authorization.
- ***Authentication:*** is essentially proving that you are who you claim to be.
  - Username and password
- ***Authorization:*** is the process of determining whether the validated identity has the rights to do what they want to do.
  - Permissions

Access Control Overview




Start
Subject claims to be a certain identity by providing identifier (e.g. username).
Subject also provides an authentication factor (e.g. password) to prove identity.
Authentication mechanism looks up the credentials in the identity datastore.
Are the credentials valid (e.g. password matches)?
Yes
Identity confirmed!
A persistent authentication token (e.g. session ID) is usually returned to the user.
No
Identity rejected.
Reject the credentials and optionally allow the user to re-attempt authentication.

Authentication Fundamentals

- The process involves two steps: **identification** and **authentication (confirmation)**.
- If an application doesn't perform proper authentication, then anyone with my username could impersonate me.
- **Without authentication, we can't perform authorization.** You can have authentication without authorization, but not vice-versa
- Because authorization looks up permissions based on a confirmed identity, it must follow after authentication
- A well-designed access control mechanism will **first perform authentication** and **then perform authorization** whenever access is requested to any protected resource.

Proving Your Identity

- Three classes:
  - Something you ***know***
    - Passwords, PIN, passphrases….
  - Something you ***are***
    - Fingerprint, hand geometry, topography of your face…
  - Something you ***have***
    - Digital certificate, smart card…

Which class does it belong to?

Two-Factor and Three-Factor Authentication

🔒**Two-factor (2FA):** using factors from two of the three categories.

🔒**Three-factor (3FA):** using factors from each category.

🔒**Two-step validation (2SV):** using factors from one category.

Using multiple factors from the same class doesn't increase the factors.

Web Application Authentication

- **Usernames** and **passwords** are the standard for authenticating users to web application.
- A **second factor** such as hardware or software security token may be used to **increase the security** of the authentication process.
- The use of biometrics is almost unheard of for a web application.

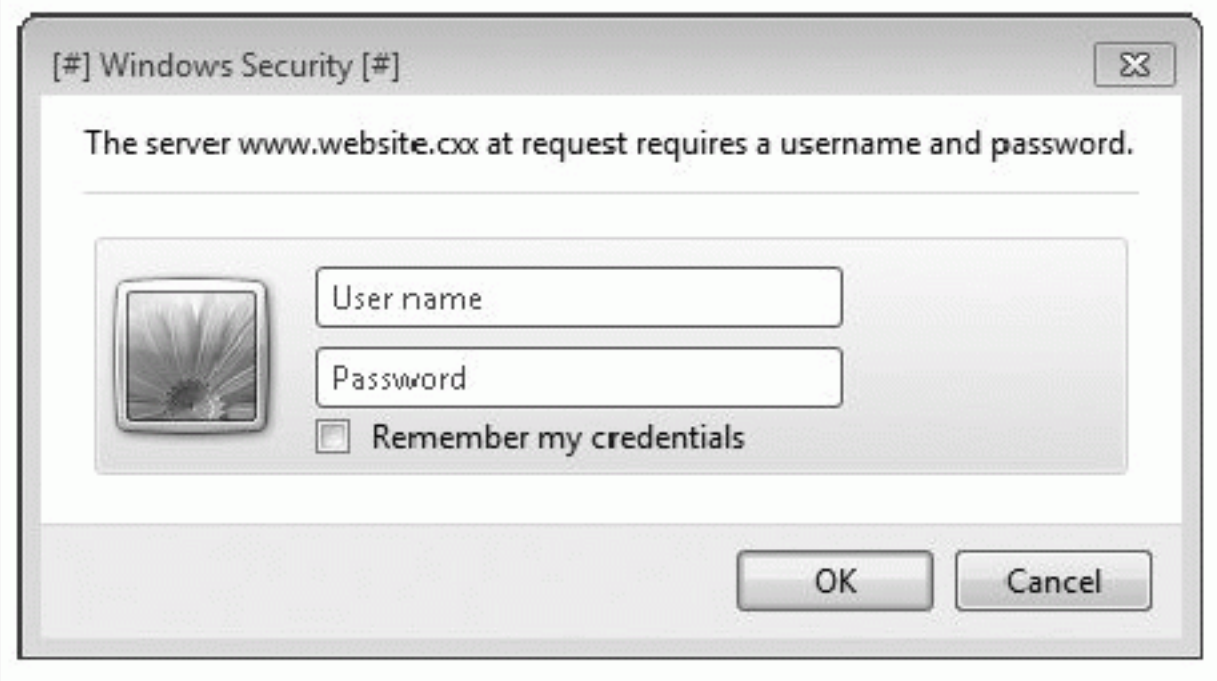
Password-Based Authentication Systems

- A number of different username and password systems exist for web apps:
  - **Built-in HTTP authentication**
  - **Single Sign-On (SSO) authentication**
  - **Custom authentication systems** (Reading Assignment)

Built-in HTTP Authentication

- **Basic access authentication:** doesn't use encryption.
- **Digest access authentication:** uses MD5 hashing, vulnerable to Man-in-the-middle attack.

Both of them have significant weaknesses and they're not recommended for use under any circumstances.

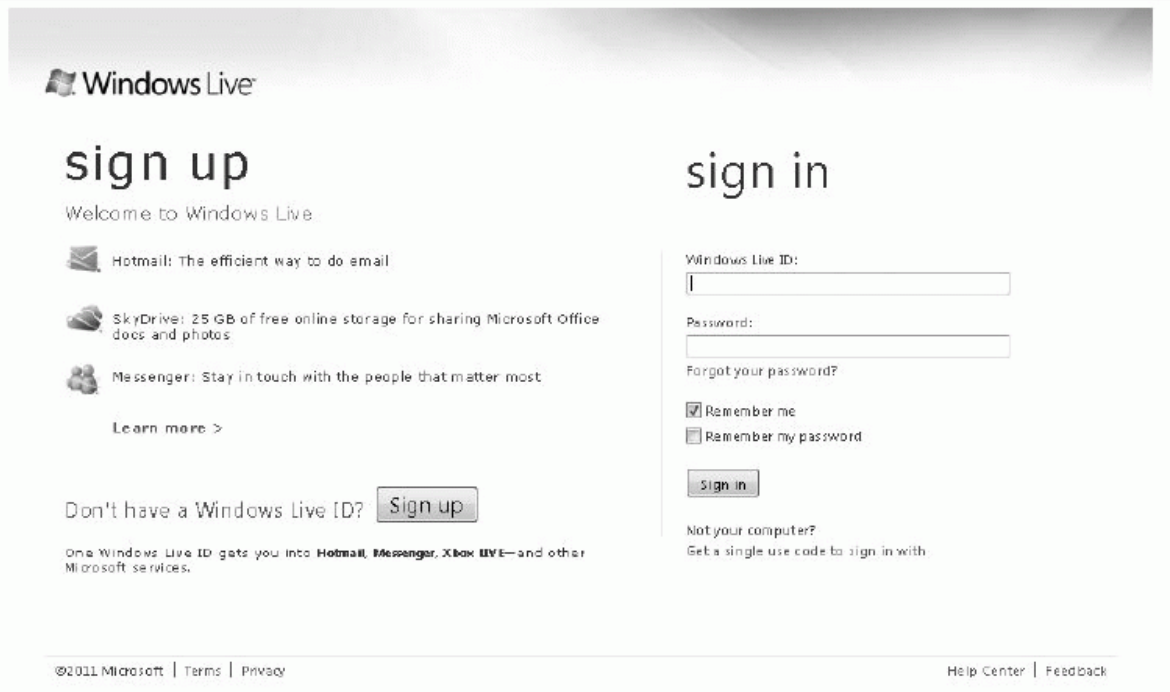


Single Sign-On Authentication

- Allow a user to log in to a single interface and gain access to multiple independently secured systems.
  - Intranets
  - Internet SSO

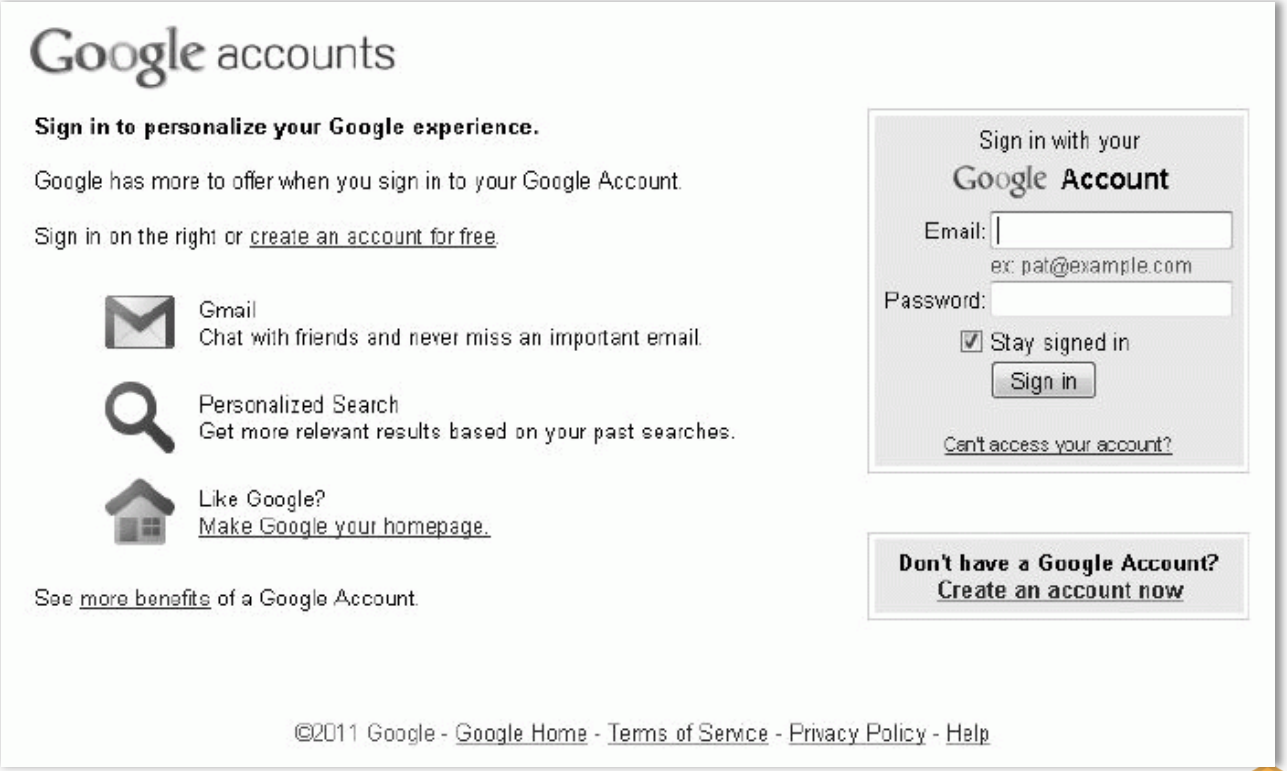
Single Sign-On Authentication

- Example: Microsoft's Live ID
  - Allows for third-party integration: developers can use the Windows Live ID Web Authentication SDK to leverage the Live ID authentication system in their sites



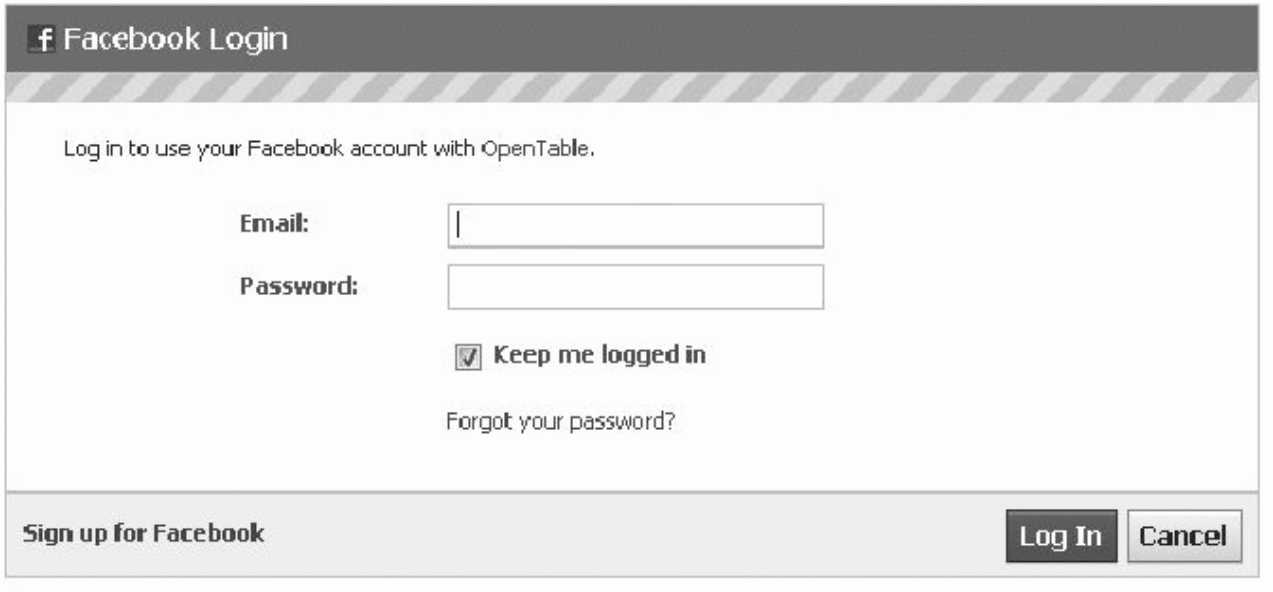
Single Sign-On Authentication

- Example: Google Accounts
  - Internet SSO
  - Are Third-party web applications supported?



Single Sign-On Authentication

- Example: Facebook Connect system
  - Connect API allows third-party developers to leverage the authentication system in addition to being able to connect with other users as they would on Facebook.
  - Examples of sites using Facebook Connect: CNN, Vimeo, Huffington Post



Single Sign-On Authentication

- **Pros:** reduce the number of credentials that must be remembered by users
- **Cons:** Single-point of failure

Validating Credentials

- There are several ways
- Depends on: **the location of comparison logic** and **how the passwords are being stored** in the back-end system

1. Comparison logic in the application with plaintext passwords
2. Comparison logic in the database with plaintext passwords
3. Comparison logic in the application with hashed passwords
4. Comparison logic in the database with hashed passwords

SQL Injection

Validating Credentials

```
$result = sql_query('SELECT users.password FROM users WHERE userId = %i', $userId); if ($result['password'] == $userPassword) { print 'access granted'; } else { print 'wrong credentials'; }
```

```
$result = sql_query('SELECT (users.password = %s) AS passwordOk FROM users WHERE userId = %i', $userPassword, $userId); if ($result['passwordOk'] == 1) { print 'access granted'; } else { print 'wrong credentials'; }
```

Validating Credentials

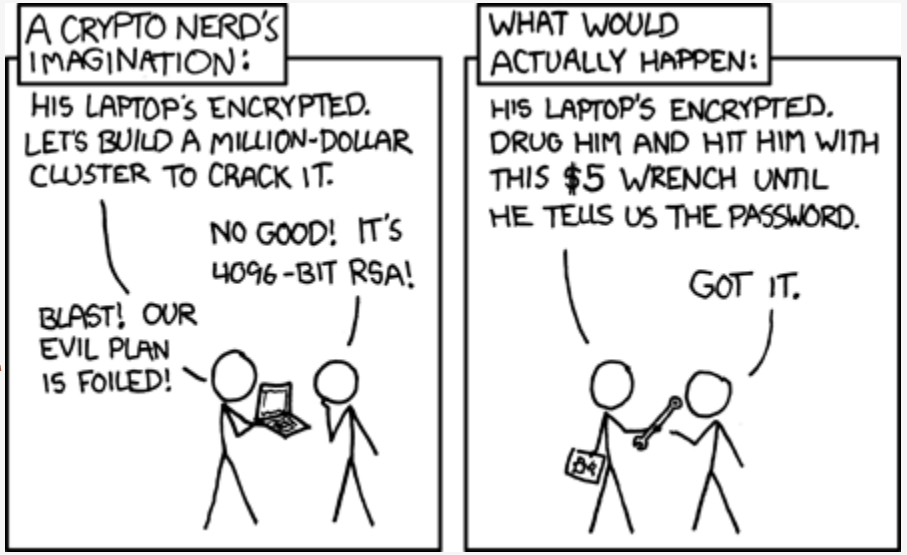
- It is important to understand the different forms of credential validation
- It is the baseline against which you can understand how the attacks work against your system
  - Example: SQL injection

Securing Password-Based Authentication

- Most popular way of confirming your identity
- Attackers will attempt to break them
- How to defend against their attacks?

Attacks Against Passwords(Reading Assignment)

- They all come down to guessing
- Either on live system (online) or hashed or encrypted passwords (offline)
- Common Variations:
  - Dictionary attack
  - Brute-force attack
  - Pre-computed dictionary attack
  - Rubber hose attack
    - Video



The Importance Of Password Complexity

- Our goal is to make it harder for attackers to guess the passwords, by making the key space (domain) larger
- By "difficult" we mean to make it take as long as possible to exhaustively search the potential key space
  - Minimum length
  - Mixed character set
  - Change them regularly

Password best practices

- Require minimum password length
- Enforce minimum password complexity
- Rotate passwords
- Require password uniqueness
- Password cannot equal username
- Disable accounts
- Properly store passwords

Secure Authentication Best Practices

- Authentication plays a fundamental role in access control.
- Must explore best practices.
  1. **When and where to perform authentication**
  2. **Securing web authentication mechanisms**

When And Where To Perform Authentication

- It's important to keep in mind that even after the most obvious authentication step of providing a username and password, the web application continues to authenticate the user.
  - **Cookies**
  - Browsers **append** associated **cookie** values to **all HTTP requests**
  - **Session ID** = persistent authentication token that you ***have***
  - Unless cookie expires

http cookies

https://networkencyclopedia.com/http-cookie/

When And Where To Perform Authentication

- **Session ID** must be **validated** in your code **with every request**.
- Your code also should perform authorization (more details in the next chapter).
- Examples of Re-authentication:
  - Amazon (before you place your order)
  - Changing password
  - Increase privileges → update cookie or issue a new one

When And Where To Perform Authentication

- The rule is to perform authentication every time that a **request** is made to **access a protected resource:**
  - When a user's access level or rights change
  - With every request to protected data or functionality
  - When accessing an outside or third-party resource

Securing Web Authentication Mechanisms

- Secure the transmission
  - SSL/TLS
- Allow account lockout
  - After certain number of failed attempts
  - Counter measure against online password attacks

- How many failed attempts should trigger the lockout?
- Within what timeframe are we counting failed attempts?
- How long do we lock out the account until it automatically resets?

	Number of Attempts	Window of Measurement	Lockout Period
Minimum Security Requirements	10	60 minutes	30 minutes
High Security Requirements	5	30	indefinite

Securing Web Authentication Mechanisms

- Allow account lockout (cont.)
  - May cause DoS ➔ admins accounts should never be locked-out
  - Sometimes not feasible → flooding customer support with requests to unlock accounts
  - Alternatives:
    - Increasing time out values → not common
    - CAPTCHA → works against brute-force attack

- Allow accounts to be disabled
  - Reduce attack surface
- No default accounts

Securing Web Authentication Mechanisms

- Don't hard-code credentials
  - They can be extracted with little effort
  - Recommendation: use keys or credential management system, or use a properly secured configuration file
- Avoid remember me
  - Classic example of security vs. convenience tradeoff
  - Provides users with authentication for long periods of time
  - For high security apps → never use
  - Standard security apps → only remember username
  - Never be a default

References

- Web Application Security: A Beginner's Guide
  - Chapter 2
  - Chapter 3