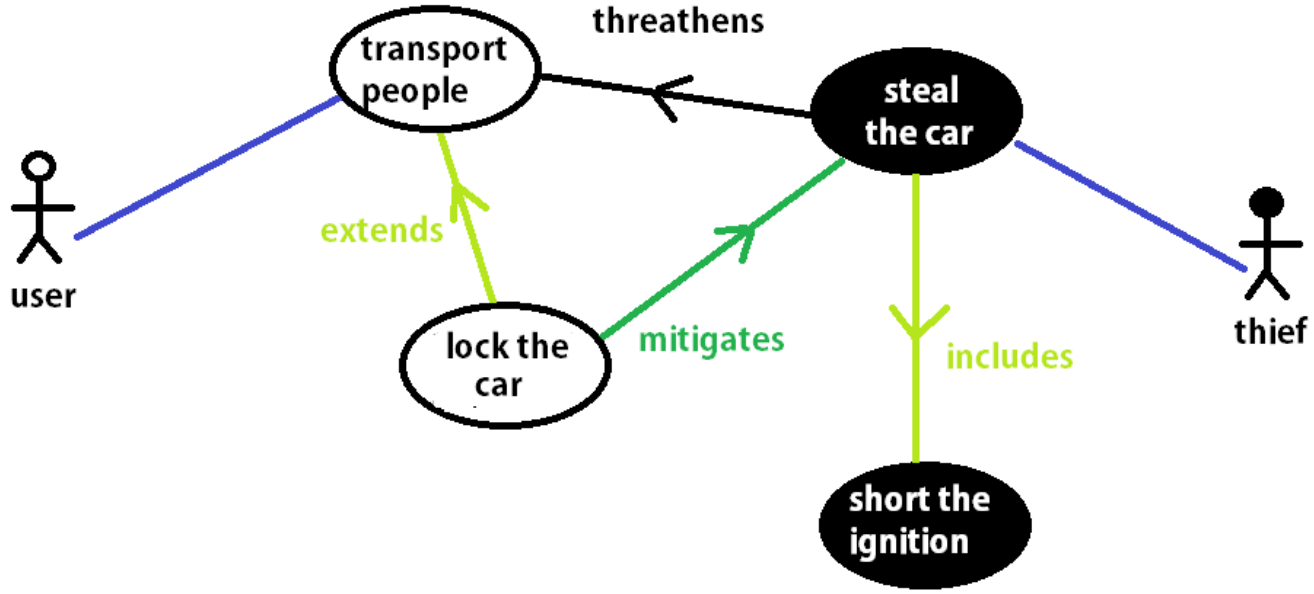
SD3 Secure Web Application Development

Password Protection

Introduction

- Security flaws can work their way into web applications in a myriad of ways.
- Security can be neglected during requirements gathering.
- Customers may not demand security from the outset.
- Misuse cases are just as important as use cases.



Introduction

- Even when developers do consider security, they are covering only the basics: authentication, authorization, access control and encryption.
- They often do not provide comprehensive input validation to prevent the likes of SQLi.
  - As a result, developers can leave security loopholes in their code.
- The primary focus of developers is to build functional and available applications.
  - Attention is on performance and ease-of-use.
- Developers can often approach the application from the end users pov.
  - What about a hackers pov?
- Many web applications are rich in functionality but vulnerable to exploitation and intrusion.

Addressing security in the SD Lifecycle

- Security needs to be addressed in all phases of software development.
  - Requirements Gathering.
    - Will the application hold government or industry-regulated information?
    - Will the application have access to, or reside on, the same network or servers as highly confidential information?
  - Design.
    - Develop misuse cases and threat models.
    - Does the application connect to a classified database?
    - If so, would stronger authentication be necessary?

Addressing security in the SD Lifecycle

- Build.
  - Enforce secure coding practices.
    - Validate inputs.
    - Adhere to least privilege of processes.
    - Consult OWASP.
- Secure code reviews.
  - In addition to quality and functional code reviews, security defect reviews need to be incorporated throughout development.
  - Software inspection software can help to automatically find and fix security-related defects
- Test.
  - Use an automated scanner that integrates into your development environment and delivers fast scanning capabilities, broad security assessment coverage and accurate results

Addressing security in the SD Lifecycle

- Deployment.
  - Your app is installed with all secure defaults enabled.
  - All file permissions are set appropriately.
  - Emerging threats need to be evaluated.
    - Vulnerabilities need to be prioritized and managed.
- Production.
  - Changes to web applications create risk.
    - What once was secure can become vulnerable.
- If security is a one-time activity, a vulnerability that enters the system after the audit can go undetected.

Usability, Performance, Cost and Security

- The web is risky.
  - Allows numerous anonymous users to request services from your machines.
  - Most of these requests are legitimate.
  - Connecting your machine allows people to attempt other types of connections.
- If you want the ultimate security.
  - Turn off your computer and disconnect it from the internet.
- To make your machines usable and available some relaxation of security is required.

Usability, Performance, Cost and Security

- A trade-off needs to be made between security, usability, cost and performance.
- Running security services can reduce the performance of your servers.
  - Virus scanners, intrusion detection software, extensive logging etc uses resources.
  - This can be countered by buying faster machines.
- A compromise is required.

Security Policy Creation.

- A security policy is a document that describes: The general philosophy toward security in your organisation.
  - The items to be protected – software, hardware, data.
  - The people responsible for protecting these items.
  - Standards for security and metrics, which measure how well those standards are being met.
- The policy shouldn't address specific implementations or solutions. It should describe the goals and security requirements in your environment.
  - It shouldn't need to be updated very often.

Authentication Principles

- Authentication attempts to prove that somebody is actually who they claim to be.
- Techniques include:
  - Passwords.
  - Digital signatures.
  - Biometrics.
  - Smart cards.
- Biometrics and hardware solutions aren't suitable for the web.
- Passwords are simple to implement, simple to use and require no special devices.
  - Are they suitable on their own for authentication?
  - On their own they have a number of potential weaknesses.

Authentication Principles

- Many passwords are easily guessed.
- If left to choose their own password 50% of users will choose an easily compromised password.
- At the expense of usability users can be forced to select passwords that have numbers or are case sensitive.
  - Making passwords difficult to remember increases the likelihood that people will write passwords down.

Cryptography

- The process of taking readable data and changing it in some way so it is no longer readable as the original text.
- One of the simplest forms is a simple substitution cypher.
  - Substitute different symbols for characters to produce a coded message.
  - Anyone with the substitution key can decode the message.
- Computer cryptography is more complex but the basic aim is the same.
- There are two types of cryptography:

Cryptography

1.One Way Hashing:

- Pass a message through an algorithm that produces a scrambled version of it.
- Its not possible to recover the original message from the scrambled version.
- Useful for storing passwords in a DB.
  - You never need to reverse the encrypted password as long as the users know their password.
  - When a user logs in, encrypt it, and check it matches what's in the DB.
- Its not useful for say emailing someone a document they need to read.
  - A reversible form of encryption is needed.

Cryptography

2.Reversible Encryption:

- Can be reversed.
- The data is encrypted using a key.
- In its simplest form you use one key to encrypt the data.
  - Anyone who has the key can decrypt the data.
- Useful for tasks such as storing secure data in a database and transferring secure data over the Internet.

Common Cryptographic Algorithms

Algorithm	Security Level
DES, MD5	Weak
RC4, SHA-1	Legacy
3DES	Baseline
AES-128, SHA-256	Standard
AES-192, SHA-384	High
AES-256, SHA-512	Ultra

Digital Signatures

- A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents.
- A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity).
- Digital signatures are a standard element of most cryptographic protocol suites, and are commonly used for software distribution, financial transactions, contract management software, and in other cases where it is important to detect forgery or tampering.

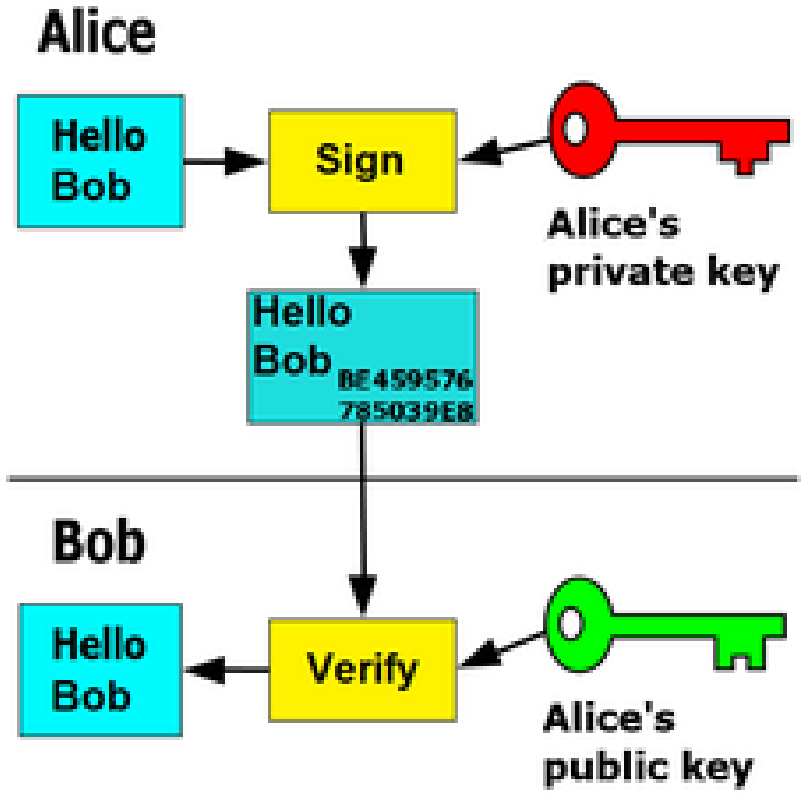
Digital Signatures

- Digital signatures employ asymmetric cryptography.
- In many instances they provide a layer of validation and security to messages sent through a non-secure channel: Properly implemented, a digital signature gives the receiver reason to believe the message was sent by the claimed sender.

Digital Signatures

- In this example the message is only signed and not encrypted.

1. Alice signs a message with her private key.
2. Bob can verify that Alice sent the message and that the message has not been modified.



Common Password Attacks

Method	Description
Social engineering	An attacker tricks a user into revealing his login credentials, or learns everything he can about the user to determine likely passwords.
Dictionary attacks	An attacker simply tries different passwords until they find one that works. Typically, this is done using an automated program and an electronic dictionary. As a result, words found in the dictionary are particularly vulnerable to this type of attack.
Rainbow table attacks	Similar to a dictionary attack, except a pre-computed lookup table is used that contains the hashes for the words. This allows an attacker who has access to the hashed passwords to crack them much more efficiently and quickly.

Salting Passwords

- Hashing passwords in your DB is a good first step to preventing attackers from being able to use them should they gain access to the passwords.
  - It doesn't go far enough.
  - Hashed passwords are vulnerable to a rainbow table attack.
  - If two users are using the same password they will hash to the same value.
    - If an attacker can crack one password, the can access more than one account.
- To prevent rainbow attacks you can use a technique called salting.
  - When a password is created, you generate a random string of characters and append it to the end of the password before hashing.

Salting Passwords

- The hash of the password + the salt is what you store as the password in the DB.
- You also store the salt value in the database because you will need it later when the user logs in.
- This technique makes sure that the rainbow attacks wont work because even weak passwords will hash to an entirely different value thanks to the salt.
- If two users have the same passwords but different salts, their passwords will not have the same hash.
  - Prevents attackers from discovering that two or more users have the same password.

Salting Passwords

- PHP has four (reasonably simple) functions for hashing passwords:

1. **password_hash()** –used to hash the password.
2. **password_verify()** –used to verify a password against its hash.
3. **password_needs_rehash()** –used when a password needs to be rehashed.
4. **password_get_info()** –returns the name of the hashing algorithm and various options used while hashing.

Enforcing Password Strength Requirements

- Its good practice to prevent users from choosing weak passwords.
- The following code only checks is a password isn't null, isn't empty and is not less than eight characters long.
- A much better solution is to use a regular expression.
  - Check min length, mix of upper and lower case chars, mandatory special chars etc.