Web Apps Security (Part 2)

Cryptography, HTTPS, Best Practices, Common Threat Vectors

Refer to the Security Chapter from the Textbook to answers the following questions

Note: You can find a PDF version of Ch16: Security under (Exercises > 08 SEC > Fundamentals of Web Dev - 3rd - Chapter 16 Security.pdf)

1. What type of cryptography addresses the problem of agreeing to a secret symmetric key?
2. What is a cryptographic one-way hash?
3. What does it mean to salt your passwords?
4. What is a Certificate Authority, and why do they matter?
5. What is a DoS attack, and how does it differ from a DDoS attack?
6. What can you do to prevent SQL injection vulnerabilities?
7. How do you defend against cross-site scripting (XSS) attacks?
8. What features does a digital signature provide?
9. What is a self-signed certificate?
10. Why are slow hashing functions like bcrypt recommended for password storage? What is a downgrade attack and how can you protect a site against it?

Answers:

1. Type of cryptography for agreeing to a secret symmetric key:
   Public Key Cryptography or Asymmetric Cryptography, specifically the Diffie-Hellman key exchange protocol, is used to agree on a secret symmetric key over an insecure channel.

2. Cryptographic one-way hash:
   A one-way hash function takes an input (or 'message') and returns a fixed-size string of bytes. The output is unique to each unique input and is designed to be irreversible, making it computationally infeasible to reverse the output hash to retrieve the original input.

3. Salting passwords means:
   Adding a unique, random string of characters known as a 'salt' to each password before it is hashed can prevent attackers from using precomputed tables (rainbow tables) to crack the password.

4. Certificate Authority (CA):
   A CA is an entity that issues digital certificates. These certificates verify the ownership of a public key by the named subject of the certificate. This is important for establishing trust in digital communications.

5. DoS vs. DDoS attack:
   A DoS (Denial of Service) attack aims to make a machine or network resource unavailable to its intended users. A DDoS (Distributed Denial of Service) attack comes from multiple sources, often a botnet, making it harder to defend against.

6. Preventing SQL injection vulnerabilities:
   Use prepared statements and parameterized queries, employ stored procedures, validate user inputs, and escape special characters.

7. Defend against XSS attacks by:
   Validating and sanitizing all user inputs, using Content Security Policy (CSP), encoding data on output, and using appropriate response headers.

8. Features provided by a digital signature:
   Authenticity, integrity, and non-repudiation. It ensures that a message or document was sent by the purported sender and that it has not been tampered with.

9. Self-signed certificate:
   A certificate not signed by a trusted CA but signed by its own creator. It provides the same level of encryption as a CA-signed certificate but does not provide the same level of trustworthiness.

10. Slow hashing functions and bcrypt:
    Slow hashing functions like bcrypt are recommended for password storage because they take more time and resources to produce a hash, making brute-force attacks less feasible.
    A downgrade attack is when an attacker forces a system to fall back to a less secure version of a protocol. Protection involves enforcing strong protocols and cipher suites, and using HSTS (HTTP Strict Transport Security).

🔥 Hands-On

Exercise 1: Encrypting and decrypting a message manually

Follow this tutorial to learn how to encrypt and decrypt data using node.js. Use it to encrypt your name and submit your encrypted name to Blackboard.

Exercise 2: Brute-force

Go to the (Web-Dev GitHub Page > Demos > 08 SEC > brute-fource.js) and explore different passwords and see how hard/easy are they to be cracked!

References

- https://www.tutorialspoint.com/encrypt-and-decrypt-data-in-nodejs