OWASP Top 10:

Cryptographic Failures in Cybersecurity

03.12.2025

Nikhil Kumar

Cryptographic failures (formerly known as "Sensitive Data Exposure" in OWASP Top 10) refer to vulnerabilities and security lapses related to the protection of sensitive data. Inadequate or improper cryptographic measures can lead to data breaches, unauthorized access, and exploitation by attackers. Below are examples and simple explanations for the OWASP Top 10 cryptographic failures:

---

1. Use of Weak or Outdated Cryptographic Algorithms

- **Description**: Using algorithms that are old and vulnerable (e.g., MD5, SHA-1, DES) makes it easier for attackers to break the encryption.
- **Example**: Imagine locking your house with an old, rusty lock. Thieves (attackers) can easily pick it open. Similarly, MD5 hashed passwords can be cracked using rainbow tables.
- **Fix**: Use strong, modern algorithms like AES-256 for encryption and SHA-256 for hashing.

---

2. Hardcoded or Weak Cryptographic Keys

- **Description**: Storing encryption keys directly in code or using simple keys (e.g., "12345") is like leaving your house keys under the doormat.
- **Example**: A mobile app developer writes `String key = "mypassword"` in their app's code. If hackers get the app, they get the key too.
- **Fix**: Store keys securely in vaults or environment variables, and rotate keys regularly.

---

3. Missing or Weak Encryption for Sensitive Data

- **Description**: Not encrypting sensitive data like credit card numbers or passwords allows attackers to read it if intercepted.
- **Example**: Sending login credentials over HTTP instead of HTTPS is like sending a postcard instead of a sealed envelope.
- **Fix**: Always use HTTPS and encrypt data at rest with AES.

---

4. Improper Implementation of Cryptographic Functions

- **Description**: Misusing cryptographic libraries can weaken security.
- **Example**: Using AES in ECB mode is like encrypting pages of a book but keeping the same words in the same position, making patterns easy to spot.
- **Fix**: Use CBC or GCM mode with AES and follow the library's recommended guidelines.

---

5. Lack of Encryption for Backups

- **Description**: If backups are unencrypted, anyone with access to them can read sensitive data.
- **Example**: Losing a USB drive with an unencrypted database backup is like losing a diary with all your secrets written in plain text.
- **Fix**: Encrypt backups and store them securely.

---

6. Failure to Use Secure Random Number Generators

- **Description**: Using predictable random numbers for encryption-related tasks makes systems vulnerable.
- **Example**: A lottery system uses a random number generator that always starts with the same seed. Hackers predict the numbers and win the lottery.
- **Fix**: Use secure generators like `SecureRandom` in Java.

---

7. Storing Passwords in Plaintext

- **Description**: If passwords are stored in plain text, attackers who breach the database get instant access.
- **Example**: A website saves users' passwords as "password123" instead of hashing them. If leaked, hackers can directly log in as users.
- **Fix**: Hash passwords using bcrypt, Argon2, or PBKDF2, and add a salt to make them harder to crack.

---

8. Poor Key Management Practices

- **Description**: Not securely managing encryption keys is like hiding your house key under a flowerpot.
- **Example**: Sharing encryption keys over email or not rotating them when employees leave.
- **Fix**: Use secure key management services (e.g., AWS KMS) and enforce strict key rotation policies.

---

9. Lack of Forward Secrecy in Communication

- **Description**: Without forward secrecy, if the private key is stolen, past communications can be decrypted.
- **Example**: Using an old RSA-based HTTPS setup without ephemeral keys allows attackers to decrypt recorded traffic.
- **Fix**: Use protocols like TLS with DHE or ECDHE for forward secrecy.

---

10. Vulnerable Storage of Secrets in Client-Side Applications

- **Description**: Storing sensitive data like API keys in mobile apps or browser JavaScript makes it easy for attackers to extract them.
- **Example**: A weather app stores its API key in the app code. Hackers reverse-engineer the app and misuse the API key.
- **Fix**: Perform sensitive operations on the server side and avoid storing secrets on clients.

Conclusion

Understanding these cryptographic failures and their real-world examples helps developers and security professionals secure sensitive data effectively. Following best practices, such as using modern encryption standards, securely managing keys, and encrypting backups, ensures robust protection against cryptographic vulnerabilities.