M10: Insufficient Cryptography

Threat Agents

Application Specific

Threat agents who exploit insecure cryptography in mobile applications can undermine the confidentiality, integrity, and authenticity of sensitive information. These threat agents include attackers who target cryptographic algorithms or implementations to decrypt sensitive data, malicious insiders who manipulate cryptographic processes or leak encryption keys, state-sponsored actors engaged in cryptanalysis for intelligence purposes, cybercriminals who exploit weak encryption to steal valuable data or conduct financial fraud, and attackers who leverage vulnerabilities in cryptographic protocols or libraries.

Attack vectors

Exploitability AVERAGE

The attack vector for insecure cryptography in a mobile application involves exploiting vulnerabilities in the cryptographic mechanisms used to protect sensitive information. Adversaries may employ various techniques, such as cryptographic attacks, brute force attacks, or side-channel attacks, to exploit weaknesses in encryption algorithms, key management, or implementation flaws. By targeting insecure cryptography, attackers aim to decrypt encrypted data, manipulate cryptographic processes, or gain unauthorized access to sensitive information. This can lead to data breaches, unauthorized access to user accounts, compromised confidentiality, or the ability to forge or tamper with data.

Security weakness

Prevalence COMMON

Detectability AVERAGE

Insecure cryptography in a mobile application introduces security weaknesses that can undermine the effectiveness of cryptographic measures and compromise the confidentiality and integrity of sensitive data. These weaknesses may include using weak encryption algorithms or inadequate key lengths, poor key management practices, improper handling of encryption keys, insecure random number generation, flawed implementation of cryptographic protocols, or vulnerabilities in cryptographic libraries or frameworks. Attackers can exploit these weaknesses to bypass encryption, perform cryptographic attacks, manipulate data, or gain unauthorized access to encrypted information. Insecure hash functions and cryptographic algorithms pose significant security weaknesses in mobile applications. These vulnerabilities can lead to serious data breaches and unauthorized access to sensitive information. When outdated or weak hash functions are used, attackers can exploit the flaws to reverse-engineer hashed data, revealing the original content. To protect mobile applications from these

OWASP Global AppSec EU 2027 - Vienna, Austria

- June 21-25, 2027

OWASP Global AppSec USA 2027 - Atlanta, GA

- September 20-24, 2027

OWASP Global AppSec EU 2028 - Vienna, Austria

- June 19-23, 2028

security risks, it is essential to adopt strong and modern hash functions and cryptographic algorithms, as well as follow best practices in encryption and key management to ensure data integrity and confidentiality. Regular security audits and updates are also crucial to maintain the highest level of protection against potential threats.

Technical Impact

Impact SEVERE

This vulnerability will result in the unauthorized retrieval of sensitive information from the mobile device.

Business Impacts

Impact SEVERE

Insufficient cryptography or insecure hash functions in a mobile application can have significant business impacts. Here are some potential consequences:

Data Breach: Weak or insufficient cryptography can make it easier for adversaries to compromise the confidentiality of sensitive data stored or transmitted by the mobile application. This can result in a data breach, leading to the exposure of sensitive customer information, such as personal identifiable information (PII), financial details, or intellectual property. Such breaches can lead to legal liabilities, regulatory penalties, loss of customer trust, and reputational damage.

Loss of Intellectual Property: Inadequate cryptography can jeopardize the protection of proprietary algorithms, trade secrets, or other intellectual property embedded within the mobile application. If adversaries are able to decrypt and extract this valuable information, it can be exploited for competitive advantage by rival companies or sold on the black market.

Financial Losses: Insufficient cryptography can lead to financial losses in multiple ways. For instance, if payment transactions or financial data are improperly encrypted, it can expose customers to fraud and unauthorized access to their funds. Additionally, the costs associated with investigating and remediating security breaches, compensating affected customers, and addressing legal ramifications can be substantial.

Compliance and Legal Consequences: Many industries have specific data protection and privacy regulations that mandate the use of strong encryption for sensitive information. Inadequate cryptography can result in non-compliance with these regulations, leading to legal consequences, fines, or sanctions imposed by regulatory authorities.

Am I Vulnerable To ‘Insufficient Cryptography’?

There are several ways in which insecure cryptography and insecure hash functions can manifest in a mobile application:

Weak Encryption Algorithms: The mobile app may use encryption algorithms that are known to be weak or vulnerable to attacks. These algorithms may have known weaknesses, be outdated, or lack the necessary level of security to protect sensitive data effectively.

Insufficient Key Length: Inadequate key length can weaken the encryption strength. If the mobile app uses short or easily guessable encryption keys, it becomes easier for attackers to decrypt the encrypted data through brute-force or other cryptographic attacks.

Improper Key Management: Poor key management practices, such as storing encryption keys insecurely or transmitting them in plain text, can expose the keys to unauthorized access. Attackers who gain access to the keys can decrypt the data without difficulty.

Flawed Encryption Implementation: The encryption/decryption process itself may be implemented incorrectly or contain programming flaws. These implementation errors can introduce vulnerabilities that attackers can exploit to bypass or weaken the encryption protections.

Insecure Storage of Data/Encryption Keys: If the encryption keys are stored insecurely on the mobile device, such as in plain text or in easily accessible locations, attackers with physical or unauthorized access to the device can retrieve the keys and decrypt the protected data.The mobile application utilizes a weak encryption algorithm or uses encryption incorrectly, such as using a weak key or failing to properly encrypt all sensitive data. This can result in compromised data if the encryption is easily bypassed or decrypted by an attacker.

Lack of Secure Transport Layer: When transmitting encrypted data over networks, it is crucial to use secure transport layer protocols like HTTPS. If the mobile app fails to implement secure transport protocols, encrypted data may be vulnerable to interception or tampering during transmission.

Insufficient Validation and Authentication: Inadequate validation and authentication of parties involved in the encryption process can weaken the overall security. Without proper validation, attackers can impersonate legitimate entities, intercept encrypted data, and manipulate it without detection.

Lack of Salting: Salting, the process of adding random data to the input before hashing, is essential for enhancing the security of passwords. Insecure hash functions may not support salting or may use weak salting methods, making password hashes susceptible to attacks like pre-computed tables or brute-force attacks.

How Do I Prevent 'Insufficient Cryptography'?

To prevent "insufficient cryptography" vulnerabilities in the mobile application, consider the following best practices:

Use Strong Encryption Algorithms: Implement widely accepted and secure encryption algorithms, such as AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), or Elliptic Curve Cryptography (ECC). Stay updated with current cryptographic standards and avoid deprecated or weak algorithms.

Ensure Sufficient Key Length: Select encryption keys with an appropriate length to ensure strong cryptographic strength. Follow industry recommendations for key lengths, considering the specific encryption algorithm being used.

Follow Secure Key Management Practices: Employ secure key management techniques, such as using key vaults or hardware security modules (HSMs) to securely store encryption keys. Protect keys from unauthorized access, including restricting access to authorized personnel, encrypting keys at rest, and using secure key distribution mechanisms.

Implement Encryption Correctly: Carefully implement encryption and decryption processes in the mobile application, adhering to established cryptographic libraries and frameworks. Avoid custom encryption implementations, as they are more prone to errors and vulnerabilities.

Secure Storage of Encryption Keys: Ensure encryption keys are securely stored on the mobile device. Avoid storing keys in plain text or easily accessible locations. Consider using secure storage mechanisms provided by the operating system or utilizing hardware-based secure storage options.

Employ Secure Transport Layer: Use secure transport layer protocols, such as HTTPS (HTTP Secure), for transmitting encrypted data over networks. Implement proper certificate validation and ensure secure communication channels between the mobile app and backend systems.

Validate and Authenticate: Implement strong validation and authentication mechanisms to verify the integrity and authenticity of parties involved in the encryption process. Perform proper validation of certificates, digital signatures, or other mechanisms used for authentication.

Regularly Update Security Measures: Stay informed about security updates, patches, and recommendations from cryptographic libraries, frameworks, and platform providers. Keep the mobile application and underlying cryptographic components up to date to address any identified vulnerabilities or weaknesses.

Conduct Security Testing: Perform thorough security testing, including cryptographic vulnerability assessments, penetration testing, and code reviews. Identify and remediate any weaknesses or vulnerabilities discovered during the testing process.

Follow Industry Standards and Best Practices: Stay updated with industry standards and best practices related to cryptography. Organizations like NIST (National Institute of Standards and Technology) and IETF (Internet Engineering Task Force) provide guidelines and recommendations for secure cryptographic practices.

Use Strong Hash Functions: Choose widely recognized and cryptographically secure hash functions like SHA-256 or bcrypt. These algorithms are designed to resist attacks and provide a high level of security.

Implement Salting: Always use a strong random salt when hashing passwords. Salting adds an extra layer of security by making it harder for attackers to use precomputed tables or rainbow tables to crack passwords.

Use Key Derivation Functions (KDFs): For password hashing, use Key Derivation Functions like PBKDF2, bcrypt, or scrypt. These functions are specifically designed for securely deriving cryptographic keys from passwords and provide additional security features like iteration counts to slow down brute-force attacks.

Example Attack Scenarios

Scenario #1: Man-in-the-Middle (MitM) Attacks - An attacker intercepts the communication between the mobile application and the server. Weak cryptography can enable attackers to decrypt the intercepted data, modify it, and re-encrypt it before forwarding it to the intended recipient. This can lead to unauthorized access, data manipulation, or the injection of malicious content.

Scenario #2: Brute-Force Attacks- Attackers systematically try various combinations of keys until they find the correct one to decrypt the data. Weak cryptography can shorten the time required for such attacks, potentially exposing sensitive information.

Scenario #3: Cryptographic Downgrade Attacks - Mobile applications may support multiple encryption protocols or algorithms to establish secure connections. If weak cryptography is allowed as a fallback option, attackers can exploit this weakness and force the application to use weak encryption. As a result, they can decrypt the intercepted data more easily and launch subsequent attacks.

Scenario #4: Key Management Vulnerabilities - Weak key management practices can undermine the security of the cryptographic systems used in mobile applications. For example, if encryption keys are stored insecurely or are easily guessable, attackers can gain unauthorized access to the keys and decrypt the encrypted data. This can result in data breaches and privacy violations.

Scenario #5: Crypto Implementation Flaws - Weak cryptography can also stem from implementation flaws in the mobile application itself. These flaws may include incorrect usage of cryptographic libraries, insecure key generation, improper random number generation, or insecure handling of encryption-related functions. Attackers can exploit these flaws to bypass or weaken the encryption protections.

References

- OWASP
  - OWASP
- External
  - External References

Edit on GitHub

Spotlight: Cobalt

Cobalt combines talent and technology to provide end-to-end offensive security solutions that enable organizations to remediate risk across a dynamically changing attack surface. The innovators of Pentest as a Service, Cobalt empowers businesses to optimize their existing resources, access an on-demand community of trusted security experts, expedite remediation cycles, and share real-time updates and progress with internal teams to mitigate future risk.

Corporate Supporters

Become a corporate supporter

HOME PROJECTS CHAPTERS EVENTS ABOUT

PRIVACY SITEMAP CONTACT