Cryptographic Failures

November 10, 2023

Emmanuel Benoist | BFH-TI

Table of Contents

Presentation

Cryptographic Failure

- **OWASP TOP 10 A02:2021**
- **Where sensitive data can be accessed due to lack of encryption**
  - Local Storage
  - Database
  - Transit (LAN)
- **Notable Common Weakness Enumerations (CWEs)**
  - CWE-259 : Use of Hard-coded Password
  - CWE-327 : Broken or Risky Crypto Algorithm
  - CWE-331 : Insufficient Entropy
  - …

Exploitability

- **Attackers typically don't break crypto directly**
  - Break something else
  - Steal keys
  - man-in-the-middle
- **Steal clear text data**
  - on the server
  - in transit
  - from user's browser

Hard to exploit ?

- **Simple: no encryption at all**
  - the most common flaw
- **When crypto is employed**
  - weak key generation
  - weak key management
  - weak algorithm usage
- **Difficult to detect server side flaws**
  - limited access
  - hard to exploit

Impact is Severe

- **Compromises sensitive data**
  - Health records
  - credentials
  - personal data
  - credit cards
  - …
- **Impact on your business**
  - Value of data for competitors
  - Reputation
  - Compliance

Examples of attacks

NSA vs. Google

- **Wikileaks showed that NSA was spying on google mail**
  - Tons of mails were readable
  - NSA listened to communication between datacenters
  - Communication was cleartext
- **Google strengthtened its systems**
  - HTTPS for any client to gmail
  - encryption of data between servers

Heartbleed

- **Disovered in 2014**
  - Bug in the library Open SSL
  - Implementation of Heartbeat by a PhD student
  - Heartbeat : extension for TLS (Transport Layer Security)
- **Principle**
  - Each heartbeat exposed up to 64kB of memory
- **Victims**
  - Canada Revenue Agency : theft of 900 taxpayers Social Insurance Numbers
  - in UK, Mumsnet had accounts hijacked (including CEO's one)

Scenario: encrypted database

- **A Database is stored encrypted**
  - Automatic encryption is done,
  - Data stored on the disk are unreadable,
- **Possible Attack**
  - SQL uses the proper key to access data,
  - SQL injection will succeed in reading data,
  - Data can be exported to another computer.

Scenario: Unsecure Cookie

- **A site has HTTP and HTTPS pages**
  - Login occures in HTTPS pages
  - Rest of pages are HTTPS
  - Another part of site is HTTP
- **If cookie is not "secure"**
  - SessionID is sent also for HTTP pages / resources (images, css, …)
  - Can be spied by third party (Wireshark, ...)
  - Can be used for inpersonating the victim

E-Commerce Web Site

- **Suppose we manage a e-shop**
  - We sell goods and clients pay using their credit cards
  - We have to store the address and references of all our clients for the legal issues.
  - Data stored: name, address, e-mail, phone, Credit Cards Numbers
- **Our web site is attacked**
  - Attackers access to our Database
  - They can harvest the whole content of our customer clients
- **Assets stollen**
  - Credit Card numbers can be stollen,
  - Credentials (usernames and passwords) also,
  - Identifiable information can be added.

E-Commerce Web Site (Cont.)

Damages?

- **For the Clients**
  - Use of Credit Cards Number by attackers
  - Privacy violation
  - Identity Theft
  - …
- **For The Web Site**
  - Reputation
  - Clients data stolen (can be resold to a competitor)
  - Business secrets stolen
- **For the Credit Card Company**
  - Reputation

Keys stored clear text

- **You operate a web server**
  - On this we server you offer a service;
  - This service needs another service (for instance a Bitcoin Wallet);
  - A key is used as a credential for the service;
  - Key is stored on the server.
- **Attack**
  - Web site is compromized
  - Attacker takes the control of the user running the web site
  - Attacker reads the key and gets it
  - Key is leaked.

Are you exposed?

Are you exposed?

- **First define which data are sensitive**
  - Health data
  - Credit Card information
  - personal information
  - Credentials (passwords, keys, …)
- **Are they stored clear text?**
  - Including Backup
- **Are they transmitted clear text?**
  - On the internet
  - Inside the internal network (between load balancers, web servers or back-end systems)

Are you exposed? (Cont)

- **Do you use weak or old crypto?**
  - Some algorithms are proven weak,
  - Can be used by default,
  - Can be found in older code.
- **Do you use the browser correctly?**
  - security directives or headers missing?

Insecure Cryptographic Storage

- **Data and Credentials are rarely protected with cryptographic functions**
  - Data collected can be used by attackers
  - For Identity Theft
  - or other crimes like Credit Card Fraud
- **Most common problems**
  - Not encrypting sensitive data
  - Using home grown algorithms
  - Insecure use of strong algorithms
  - Continued use of proven weak algorithms (MD_5, SHA-1, RC_3, RC_4, etc.)
  - Hard coding keys, and storing keys in unprotected stores

Define your assets

Which assets should be protected?

- **Passwords of users**
  - *Clear-text :* accessible by SQL injection, or insiders
  - *Hashed :* can be verified, but not read
    Problem : Easy to check using lists of hashed passwords (dictionary attack)
  - *Hashed with the same salt :* Attackers need to find the salt
  - *Hashed using a generic salt and a specific salt*
- **Credit Card Numbers**
  - Ruled by the Credit card industry (see later)
- **Private keys**
  - Should always been stored encrypted
  - At least protected using a passphrase

Regulatory framework I

- **Private Data must be protected**
  - Private data (GDPR EU's General Protection Regulation and Swiss FADP Federal Act on Data Protection aka "*Bundesgesetz über den Datenschutz*" or "*Loi fédérale sur la protection des données*")
  - Any personal data (that can be linked to a person) must be protected
  - Art 7 of FADP:
    "*Personal data must be protected against unauthorised processing through adequate technical and organisational measures.*"

Regulatory framework II

- **Sensitive data:** Swiss legislation defines the following data as "*sensitive personal data*" (FADP art.3)
  - "*religious, ideological, political or trade union-related views or activities,*"
  - "*health, the intimate sphere or the racial origin,*"
  - "*social security measures,* "
  - "*administrative or criminal proceedings and sanctions;*"
- Private persons holding such data must declare them to the Data Commissioner.
- Goal of the data must be approved
- Protection must be presented to the data commissioner.

Cryptographic tools

- **Encryption**
  - If you need to read and write data: symmetric encryption (e.g. DES, AES)
  - If reading and writing are done by different entities: asymmetric encryption (e.g. RSA, El-Gamal, ED25519, ...)
- **One-way hash functions**
  - One input has always the same output
  - Impossible to go from the output back to the input
  - No collision can be generated (two inputs having the same output)
  - Example : SHA-512

Recommendations

Use only Strong Crypto Algorithms

- **Do not create cryptographic algorithms**
  - Only use approved public algorithms such as:
  - AES, RSA public key cryptography and SHA-512 or better (eliptic curves for instance)
- **Do not use weak algorithms**
  - MD5 / SHA1 hash functions have been proven weak
  - Favor safer alternatives such as SHA-256
- **Use TLS with Perfect Forward Secrecy**
  - To protect the future if a private key is leaked.
  - The communication is done using a session key that can not be found, even it key is leaked later.

Handle Keys with extra Care

- **Generate keys offline and store private keys with extreme care**
  - Never transmit private keys over insecure channels
- **Store if possible your private key encrypted**
  - Using a pass-phrase
  - Or in a Password Manager

Handle Keys with extra Care (Cont.)

- **Do not let your front user access the key directly**
  - Store the key encrypted
  - Store the key such that it can only be accessed from one user
  - Create a REST API to access to functions requiring the key.
- **Do not store the key for decrypting the key where it can be read by the front user**
  - Enter it manually each time the server is restarted
  - or store as an environment variable (only Root can read starting scripts - for instance `/etc/systemd/system/` on Linux)
  - ...

Protect Infrastructure Credentials

- **Data Base credentials**
  - Use tight file system permissions and controls
  - Encrypt securely credentials
- **Encrypted data should not be easy to decrypt**
  - database encryption,
  - useless if database connection pool provides unencrypted access

Store only sensitive data that you need

- **Credit Card number**
  - Many shops use third party payment providers
  - *No need to store Credit Card numbers*
  - Receives a transaction number certified by the bank
- **Examples**
  - Zalando uses “Verified by Visa” or “MasterCard secure code”
  - The bank verifies the valid use of the card (often with 2FA)
- **Data that you don’t have can not be stollen**

Ensure that random numbers are cryptographically strong

- **Random generators are used for**
  - random numbers
  - random file names
  - random userID’s or sessionID’s
  - random strings
- **Should be generated in a cryptographically strong fashion**
  - No one should guess
  - Seeded with sufficient entropy
- **Bad example**
  - Seed = current time in milliseconds or microseconds
  - Very easy to know (or brute force)

Use only widely accepted implementations of crypto algorithms

- **Do not implement an existing algo on your own**
  - No matter how easy it appears
  - Example : heartbleed
- **Ensure that implementation involved crypto specialists**
  - For the design
  - and for the review
- **If possible: implementation should be FIPS 140-2 certified**

Always ensure data integrity and authenticity

- **Encryption must be combined with data integrity**
  - Otherwise the ciphertext can be changed
  - Especially over an untrusted chanel (e.g. URL or cookie)
- **Use cryptographic cipher modes that offer both confidentiality and authenticity**
  - CCM, GCM, OCB
- **If not combine encryption in cipher-block chaining mode CBC with MAC (Message Authentication Code)**
  - CBC = Cipher Block Chaining
  - Message Authentication Code : HMAC, UMAC
  - Do not use ECB mode (Electronic codebook)

Store passwords hashed and salted

- **Clear text**
  - Vulnerable to SQL injection and the like
- **Just hashed**
  - Present in rainbow tables
- **Hashed with a user specific salt**
  - A salt is an information added to the value before passing to the hash function.
  - Attack must be conducted for each of the users
- **Use hash functions with work factors**
  - `Argon2`, `scrypt`, `bcrypt` or `PBKDF2`

Ensure that any secret key is protected from unauthorized access

- **Define a key lifecycle**
- **Store unencrypted keys away from the encrypted data**
- **Use independent keys when multiple keys are required**
- **Protect keys in a key vault**
- **Document concrete procedures for managing keys through the lifecycle**
- **Build support for changing keys periodically**

PCI Data Security Standard

PCI Data Security Standard

- **Payment Card Industry Data Security Standard**
  - Developed by major credit card companies (e.g. Visa, Mastercard, American Express)
  - to help organizations preventing credit card fraud
- **Must be implemented by any merchant using Credit Cards**
  - A company processing, storing or transmitting payment card data must be PCI DSS compliant
  - Risk: losing their ability to process credit card payment
- **Compliance must be validated periodically**
  - Validation conducted by auditors (Qualified Security Assessors (QSAs)
  - Smaller companies just fill a self-assessment questionnaire.

PCI-DSS Requirements I

- **Build and Maintain a Secure network**
  - Install and maintain a firewall
  - Do not use vendor-supplied default password and other security parameters
- **Protect Card-holder Data**
  - Protect stored card-holder data
  - Encrypt transmission of card-holder data across open, public networks
- **Maintain a Vulnerability Management Program**
  - Protect all systems against malware and use and regularly update anti-virus software or programs
  - Develop and maintain secure systems and applications
- **Implement Strong Access Control Measures**
  - Restrict access to card-holder data by business need-to-know
  - Identify and authenticate access to system components
  - Restrict physical access to card-holder data

PCI-DSS Requirements II

- **Regularly Monitor and Test Networks**
  - Track and monitor all access to network resources and card-holder data
  - Regularly test security systems and processes
- **Maintain an Information Security Policy**
  - Maintain a policy that addresses information security for all personnel

PCI DSS - Storage of data

- **Card-holder Data**
  - Primary Account Number (PAN, a.k.a. credit card number)
  - Card-holder name
  - Service Code
  - Expiration Date
  - *Can be stored*
  - *Require protection*
- **Sensitive Authentication Data**
  - Full Magnetic Stripe
  - CVC2/CVV2/CID
  - PIN
  - ***Can in no case be stored***

Store only necessary data

- **Develop a data retention and disposal policy**
  - Limit storage and retention time to which is required
  - for business, legal, and/or regulatory
- **Protect PAN**
  - Truncate card-holder data if full PAN is not needed
  - Never send PAN in unencrypted e-mails
  - Mask PAN when displayed
- **Render PAN unreadable anywhere it is stored**
  - Strong one-way hash functions
  - Truncation
  - Index tokens and pads (pads must be securely stored)
  - Strong cryptography with associated key management processes and procedures

Conclusion

Conclusion

- **Define which data are sensitive**
  - Depends on regulation
- **Protect data**
  - Use cryptography
- **Discard data as soon as possible**
  - Any non existent data can not be stollen!

Sources

- **OWASP Top 10, A3:2017**
- **Wikipedia Heartbleed**
- **Swiss Federal Act on Data Protection**
  `https://www.fedlex.admin.ch/eli/cc/1993/1945_1945_1945/fr`
- **EU's General Data Protection Regulation**
  `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504&qid=1532348683434`
- **OWASP Cryptograhic storage cheat sheet**
  `https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet`
- Payment Card Industry Data Security Standards (PCI DSS)
  `https://www.pcisecuritystandards.org/`