Information security and privacy

Pierre Colson

Monday 03 January

Contents

Markdown verion on *github*

Vocabulary

- **Virus** : a malware that infects a file and replicates by infecting other files
- **Worm** : a piece of malware that propagate automatically
- **Trojan** :
  - a malware hidden in a usefull software or file
  - a malware that stays on the victim's computer and communicate with a control center to carry out malicious activity
- **Rootkit** : hides the precense of a malware on a computer
- **Ransomware** : encrypts the files and request payment for decryption
- **Vulnerability** : weekness in the logic, the software or hardware of a system (bugs)
- **Exploit** : method/tool to make advantage of a vulnerability
- Vulnerability can be fixed by **patching** a system
- **Zero day** exploit : exploit for which no patch exists yet

Basic properties

Security

- Protects the data of data owners against attacks

- **Confidentiality** :

  - keep informations secret
  - give read access only to those who need to know
  - tools : access control, isolation, encryption

- **Integrity** :

  - keep information correct
  - prevent modification of the data

  - detect modification
  - tools add a hash, a MAC or a signature, make public

- **Availability** :
  - keep information available/systems running
  - tools : make copies, duplicate/distribute systems, prevent intrusions

- **Authenticity** :
  - demonstrate the authenticity of information
  - prevent fake information
  - detect modification
  - tools : add keyed hash (MAC) or a signature

- **Non repudiation** :
  - prevent denial of a statement
  - tool : add a signature as proof of origin

Privacy

- Protects the data *subject* against abuse

- **Confidentiality** :
  - keep information of *the data subject* secret
  - give access only to those who need to know
  - tools : access control, encryption, absence of data

- **Anonymity** :
  - prevent a link between data and a subject
  - reduce/modify information until no correlation is possible
  - tools : k-anonymity, defferential privacy

- **Absence of information** :
  - prevent revealing information
  - do not request, or delete information that is no longer needed
  - work on encrypted information
  - tools : homomorphic encryption, private information retrieval, zero knowledge proofs

Cyber Threats

- A **threat** is a potential unwanted action that creates impact

- **Cyber attack lifecycle** :
  - Preparation
  - Gain access
  - Maintain access
  - Complete mission
  - Cover tracks

- **Commodity threats** :
  - Non targeted
  - Fully automated
  - Low risk to attackers
  - Short term financial gains

- **Hacktivism** :

  - Politically motivated hacking
  - Variant of (anarchic) civil disobediance

Web application vulnerabilities

- **OWASP** : **O**pen **W**eb **A**pplication **S**ecurity **P**roject
  - Documentation on the top 10 critical security risk of web application
- **Injection** :
  - Context can be : HTML, JavaScript, JSON, SQL
  - Special character sequences in user inputs can trigger an action in the context
- **Injection protection** :
  - Refuse characters you do not want
  - Escape (encode) specila characters when you use them
- **Direct object reference** : When a user-submitted parameters is a direct reference to a resource, a user may try to change it to access other resources

Software vulnerabilities

- **Buffers overflows** : while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory location
- **Buffers overflows protection** :
  - **Stack canaries** :
    - Push a random value on the top of the stack at the beginning of a funcion
    - Before returning, verify that the value has not been modified
  - **Non executable memory** :
    - Do not want to set execution permission on a page that can be written while the program is running
  - **Address space randomization** (ASLR) :
    - Every time the program is started, it is load at a random address
    - Every time the system boot, the OS is load at random address

Crypto

- **Symmetric Crypto** : Encryption and decryption is done with the same key
  - Solve the problem os transferring large amount of confidential data
  - Creates the problem of transferring a symmetric key
- **Stream cypher** : Use the ey and a pseudo random generator to generate a stream of random bits
- **Block cypher** : Encrypt fixed blocks of data
  - a *padding scheme* is used to fille the last block
  - a *mode of operation* is used to combine multiple block
  - DES (collisions and brute force) ⇒ AES
- **Mode of operation** :
  - ECB :
    - Encrypt each block separately with the same key
    - Same cleartext clock results in same ciphertext block
  - CBC :
    - Introduces the use of an *initialization vector* (IV) for the first block
    - Each ciphertext block acts a IV of the next block
    - Decryption is the opposite of encryption
    - Does not reveal any structure
    - Malleability : flipping one bit in a cyphertext block flips the same bit in th next cleartext block and mangles the current block
    - The last block must be padded to obtain the correct block size, if not carefully implemented, validation fo padding can lead to leakage of the cleartext
- **Hash** function take an arbitrary length input and generate a fixed length output

    - *Pre-image resistance* : Given an hash h, it is difficult to find a message m for which $h = hash(m)$
    - *Second pre-image resistance* : Given a message m_1 it is difficult to find a second message m_2 such that $hash(m_1) = hash(m_2)$
    - *Collision resistance* : It is difficult to find two arbitrary messages that have the same hash
    - SHA-3 : no weakness known
- **Messages authentication codes** (MAC) :
    - Like a hash function, but involves a symmetric key
    - The same key is used to generate the MAC and to validate it
    - If the key is know only to the two parties of an exchange, a correct mac proves
        - that the message was not created by a third party (authentication)
        - that the message was not been modified (integrity)
- **Public-key Crypto** : Uses a pair of public (encryption) and private key (decryption)
    - Solves the problem of having to agree and on a pre-shared symmetric key
    - No need to keep the public key secret (as the name suggest ^^)
- Assymetric is powerful but orders of magnitude slower than symmetric crypto
- Assymetric is typically used to exchange a symmetric key
- All these algo are only safe if you use keys that are long enough
    - symmetric : 128 to 256 bits
    - asymmetric : RSA 2048 bits, ECC 256 bits
    - has function : 256 bits
- With public key crypto the puclic key does not have to be secret but it still has to be authentic (e.g. man in the middle atk)
    - We need a trusted third party to distribute the public keys
    - The Certification Authority certifies the keys by signing them
        - If we trust the key of the CA, we can trust all keys signed by the CA
    - A signed key is a certificate. It contains at least :
        - The identity of the holder
        - The validity date of the certificate
        - The public key of the subject
        - The signature by the CA

TLS and HTTPS

- **TLS** Transport layer Security : provide a secure channel between two communicating peers
    - The server is authenticated with a cetificate
    - It proves its identity by signing some information received from the client with its private key
    - Client and server create a symmetric key using asymmetric crypto
    - They use a symmetric cipher to encrypt data:
    - They use HMAC to guarantee integrity
- Let's Encrypt ⇒ free certificates
- A **Public key infrastructure** (PKI) ditributes public keys usign certificates
- HSTS and Certificate transparency protect against MITM and fraudulent CAs