Information security and privacy

Pierre Colson

Monday 03 January

Contents

Vocabulary **1**

Basic properties **2**
- Security 2
- Privacy 2

Cyber Threats **3**

Web application vulnerabilities **3**

Software vulnerabilities **3**

Crypto **4**

TLS and HTTPS **5**

Dtatabase Security **5**

Password Storage **5**

Access Control **6**

Authentication **6**

Network and Operational Security Practices **7**

Trusted Computing **8**

Markdown verion on *github*
Compiled using *pandoc* and *`gpdf` script*

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) $\Rightarrow$ 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

Dtatabase Security

- **Access control** : Least privilege
- Granularity at the row level can be achieved by defining *views*
- SQL databases also support *role based* acces control
- To limit the impact of SQL injection, use different DB users for different accesses

Layer	**Function**	**Protect against**
Hardware / OS	Data is encrypted when read/write to disk	Stealing/cloning virtual machines
Database	BD encrypts when read/write to file	Access by OS users/admins
Network	DB encrypts when read/write to network (e.g. TLS)	Hackers connot sniff data in transit
Application	Application encrypts when read/write to the DB	Access by admins, memory dumps by OS admins

- If the data is encrypted in the database then the DB cannot
  - search with wildcards (e.g. `WHERE name='Pete%'`)
  - sort, compare or aggregate data
  - ⇒ the BD is pretty useless

Password Storage

- Classic way : use **salt** and **iterations**
- Modern way : use a **memory hard function**
- **Time-memory trade-offs** :
  - We create a *reduction* function r : it takes a hash as input and produces a password from our set
  - We build chains : $p_1 \overset{hash}{\rightarrow} h_1 \overset{reduce}{\rightarrow} p_7 \overset{hash}{\rightarrow} h_7 \overset{reduce}{\rightarrow} \dots \overset{hash}{\rightarrow} h_3$
  - We only keep the first and last element of each chain
    * this is where we save memory
    * we pay for this with more time to crack the password
  - We build a table with several chains
  - Hellman's original trade-off becomes inefficient when there are too many chains in a single table
    * For each collision of the reduction function, we end up with two identical chains
  - **Rainbow table** solve the collison problem by using a different reduction function in each column
- If yous search through all columns of all tables :
  - Hellman : t^2 memory look-ups, t^2 hash operations
  - Rainbow : t memory look-ups, $\frac{1}{2}t^2$ hash operations
- Adding a random value (**hash**) to the hash function prevents :

- cracking multiple hasesh with a single hash calculation
  - calculating the hashes in advance
- Another simple way to slow the attacker is to apply the hash functions multiple times
- **Memory hard function**
  - the function run through many steps
  - intermediate steps results are stored in memory
  - each step depends on results from previous steps

Access Control

- **Access control** defines and enforce the operations that can do an objects
- Principle of **least privilege**
  - subjects should have the minimum rights necessary to their job
- Multiple level of access control :
  - Network
  - Operating system
  - Application
  - Within an entreprise
- Multiple approaches to access control
  - **Role-based Access Control** (RBAC)
    * Simplifies the specification of premission by grouping users into roles
    * A role can contains multiple permissions
  - **Discretionary Access Control** (DAC)
    * Access control is at the discreton of the object owner
    * Owner specifies policies to acces resources it owns
  - **Mandatory Acces Control** (MAC)
    * Tries to ensure that even someone with acces cannot leak the data
    * Depends on the trusted software and admins
    * no write down
- **ACL** vs **Capabilities**
  - Think of a door protected by a bouncer vs a door protected by a lock
  - ACL :
    * The bouncer knows exactly who can get in
    * People don't know where they can get in and where thay cannot
  - Capabilities :
    * Doors do not know who will show up with a key
    * People know exactly for which doors they have a key
- Modern OSes make use of all of these types :
  - DAC with ACLs for file and most objects
  - DAC and capabilities for privileged operations
  - Using groups to implement RBAC
  - Mac for protectiond the integrity of a system

Authentication

- Access control only makes sense if we can *authenticate* subjects
- **Password**
- **Something you own** : hardware/software token
- **OATH** is a standard that describes
  - How **OTPs** are generated from a seed
  - An XML fomat for importing the seeds into a authentication server
- **Biometrics**
  - no hashing is possible

    - it is impossible to change a stolen finger
- **Challende-response** : Rather then sending the password to the server
    - The server sends a random challenge to the client
    - The client uses the hash of the password to create a reponse
- **Kerberos** uses a three steps approach
    - An anthentication server (AS) authenticates the client and delivers a ticker granting ticket (TGT)
    - The client can then present the TGT to the tocket granting server (TGS) to get a ticket for the service he wants to use
    - The client can access the service
- **Oauth2** is a protocol used for delegated authentication on the internet
    - Facebook, Google, Twitter etc. can be used to authenticate and access other application

Network and Operational Security Practices

- Secure communication, outside of our network
    - TLS, IPsec, VPNs
- **Network segmentation** : Break down the network based on system and data classification or into functional zones
    - Access from zone to zone can be managed by access control list (ACLs) in router or firewalls
    - Prevents all-at-once compromise of facilities
    - Protects the data center form external threats
    - Containment zones aims at stpopping attacks from spreading between zones
- **Demilitarized Zone** (DMZ)
    - A physical or logical subnet that contains and exposes an organization's external-facing services to an auntrusted network
    - An external network node can access only what is exposed in the DMZ
- **Zero trust network**
    - Do note trust anybody, not even internal machines
    - More work for configuring machines
    - Less work on configuring the network
    - Greatly reduces the impact if on machine is compromised
- **Virtual private network**
    - Encryption and encapsulation keep the network private
    - Before a packet is sent over the public network, it is encrypted and encapsulated with and IP header with the public address
    - Let remote workers access the internal company network
    - Interconnecting remotes sites for a company
- **Firewalls**
    - Enforce network level access control
    - Firewalls operate at the network layer
    - Firewalls should aussi be present within the network
    - Principle of default deny
- **Proxies**
    - They operate at the application level
    - **(Direct) proxies** : between the client and internet
        * Protect our users when they access servers on the internet
    - **Reverse proxies** : between internet and the server
        * Protect our servers when accessed by users from the Internet
    - **Web proxies** protect users by
        * Analyzing all data downloaded from the web with anti-virus software
        * Blocking access to dangerous sites
- **Web application firewall** (WAF)
    - It stands in front of your web server and receives the requests from the internet.
    - It analyses the request, and if it deems them safe, it forwards thel to the real server

- **Instructions detection systems**
  - Inspects traffic for all application to detect protential intrusions
  - **Signatures based** system
    * Network traffic is compared to signature form a pattern database
    * *Snort* is an example of signature based IDS
  - **Anomaly based** system
    * IDS creates traffic profile during normal operation to callibrate
    * Looks for unusual packets
  - Possible issues
    * False posititves (too many alarms)
    * False negatives (too many sucessful attacks)
- Keeping **audit trails** (logs) is an important part of network security
- Good way to protect data : **Backups**
  - We also need restorations tests, to check if we are actually able to restore data from backups
  - We also need a **Disaster Recovery Plan** (DRP) that explains in details how to rebuild each system in case of a major failure

Trusted Computing

- **Trusted hardware** : A piece of hardware can be trusted if it always behaves in the expected manned for the intended purpose
- **Attestation** : It can be prove that it does what you think it does
  - Attest there is secure hardware
  - Attest the state of the OS
  - Attest state of the code
  - **Secure boot**
- **Sealing** : It can store secrets in unprotected memory
  - The device derives a key that is tied to its current status and stores the encrypted data
  - Data can only be decrypted by a device with the same status
- **Isolation** : It is not possible to *peek* inside
  - Requires protection against side channel attacks
  - Trusted hardware offers one well identified entry-point to interact with the software
  - **Tamper resistance** : hard to open
  - **Tamper evident** : You can see if it has been opened
  - **Tamper responsive** : Delete keys when attacked
  - **Resitance to side channel attacks** and physical probing
- **Trusted Execution Environments** (TPM) : Isolated processing environment in which applications can be securely executed irrespective of the rest of the application
  - Dedicated devices : Strong physical protections
  - Secure enclaves : Prtected regions of memory
  - Enable processes to run while being protected from attacks perpetrated by the OS, hypervisor, firmware, drivers, or remote attackers
- **Non-volatile Storage**
  - **Endorsement key** (EK)
    * Created at manufacturing time
    * Signed by manufacturer
    * Cannot be changed
    * Used for *attestation*
  - **Storage Root Key** (SRK)

    * Used for encrypted storage
  - **OwnerPassword**
  - They are private and never leave the TPM
- **Platform Configuration registers** (PCR lol)
- **Side channels** : Determine the secret key of a cryptographic device by measurign its execution time, its power consumption, or its electromagnetic field
  - Learn how the system's secret by observing how different computations are
  - Difficult to create trusted hardware resitent to side channel
- For trusted hardware we need to **trust the manufacturer**