Computer Security

Elia Ravella

March 23, 2021

Contents

Part I
Introduction

1 Security

What is security? First, we need to distinguish *security* itself and *safety*; the first represents a system that protects from the outside, the second a system that does not harm when it's used.

1.1 What is a secure system and how to engineer one

Famous triangle problem, this time with CIA properties

- Confidentiality: only authorized entities can access information
- Integrity: only authorized entities can modify information;
- Availabilty: data must be available to all authorized entities in a determined time constraint.

"You can take only two" problem: A conflicts with C and I.
System security is an engineering problem: it's crucial to find an *optimal tradeoff* between properties to be ensured, and costs to be faced when securing such properties.

1.2 Entities in a secure system

- Vulnerability: something that allows to violate one of the CIA properties. It can be something as a design errors, a construction error, or a use defect. Even an *inherent property* of a system can be a vulnerability.
- Exploit: a specific way to use a vulnerability (or more) to violate one of CIA properties.
- Asset: values of a system, properties that must be protected. Depending on the context a system operates in, different assets can be defined (soldier - armor - life).
- Threat: **possible / potential** violation of CIA constraints. A threat is linked to a threat agent, that's the actual physical person / system that performs the potential attack.
- Attack: **intentional** use of an exploit to violate CIA properties.
- Risks: combination of assets, vulnerabilities and threats. Formally (using economic value) they're A*V*T. Risk management takes care of the first two properties. The security problem is to manage the reduction of vulnerabilities and the recovery / mitigation plan.

A vulnerability can be present without any exploits. An exploit depends from a set of vulnerabilities.

Security vs Cost Balance As usual, costs can be separated in direct and indirect costs. In this scenario, direct costs are related to the actual put-in-place of a solution. Indirect costs, instead are all the user experience costs, all those experience-heaving procedures, for example. So EVERY SINGLE SECURITY SOLUTION ADDS A COST.

Trust and Boundaries The security problem must have boundaries, otherwise it will cover over the universe. So, subparts must become *trusted* and be *assumed secure*. All the system that are beyond (below) the boundary are assumed secure too. The trusted element is the one to be tested most.

Part II
Cryptography

2 Cryptography, cryptoanalisys, cryptology

Cryptography is the building of secure mathematical systems. Cryptoanalisys is the discipline of analyzing such systems, and trying to break them. Cryptology is the union of both.
Cryptography *as a discipline* was formalized by Claude Shannon in 1949. The (still actual) formalization of a cryptosystem is composed of 3 main components:

1. the *plaintext*, the message to be encoded
2. the reversible *key*
3. the *cyphertext*, the encoded message

2.1 Confidentiality and Integrity

"Why a cryptosystem"? The two main features that a cryptosystem must have are

1. confidentiality: only the recipients can read and utilize the information in the message.
2. integrity: data can't be altered in the message passing. If so, the recipient can detect the change.

The Kerchoff's principles states that the security of a cryptosystem relies *solely* on the secrecy of the key, rather than the secrecy of the algorithm. This, in fact, should remain public. This means that a secure system is *transparent* inherently.
The key must be the trusted element of the entire cryptographic system.
Formally, a perfect cypher does not leak any information about the message if and only of

$$P(M = m \,|\, C = c) = P(M = m) \tag{1}$$

As theorized by Shannon. So "the probability of observing message m given cypher c is indipendent from the cypher given". Shannon also proved that perfect cyphers exists: they're *one time non reusable key systems, with the lenght of the key equal to the lenght of the message* (also known as the **Shannon Theorem)**.
The only perfect cypher implementation is OTP, One Time Pad. Classic XOR based encryption with a one time password (pre shared) that is long as the message. The cool thing about perfect cyphers such as OTP is that bruteforcing the cyphertext simply returns all the possible combination of plaintext, rendering so useless the brutaforce approach.

2.2 Cryptoanalysis

All the "brutaforce" fuss is about the problem of determining "how much a system is breakable". Bruteforce "always works", but it's a costly approach. So, it formally represents an *upper bound* to the decyfering provedure complexity for that cypher. Other (more smart) attacks are

- Cyphertext attack - only cyphered text are available, a key/algorithm leak of information must be exploited
- Known plaintext attack - comparation between the plain and crypted texts should lead to the key
- Chosen plaintext attack - reverse engineering of the algorithm

distinguished by the materials available to the attacker, and orderd in increasing complexity.

2.3 Remarks on Cryptosystems

1. Security of a CS is based on the robustness of the *algorithm*
2. No algorithm is invulnerable to brute force attacks, exceptionmade for *one time pads*
3. An algorithm is said to be broken if there's at least one way *faster than bruteforce* to bypass it
4. THe only way to prove a cypher is robust is to try to break it

3 Symmetric Encryption

Classic: Alice uses key K to encrypt message M into cypher C, then Bob uses the very same key K to decypher cypher C into message M. Problem: exchanging keys. We need a *trusted separated channel* to exchange the key. What is a trusted channel? It generally is a different channel that both the recipients *consider trusted* and generally *requires a different attack to be violated.*
We so must try to find a way to send to all recipents the key *in a secure way.*
Another issue with symmetric encrypted communication is scalability, each pair of user in a network should have a unique key.

3.1 How to build a secure symmetric system

Three characteristics are fundamental to ensure the robustness of a symmetric cypher:

1. *Multi Alphabetical Cyphers*: to mask structure and repetition in the plaintext several target alphabets can be (must be) used at the same time.
2. *Transposition*: also called diffusion, it consists of "swapping the value of bits", messing with the order in which the plaintext is translated.

Keyspace With keyspace is intended the *set of possible keys*. There's an exponential correlation between the size (rank) of this set and the temporal complexity of the brutaforce effort. This means that more possible keys (generally translated in "how many bits is the key made of") more the time needed to bruteforce the cypher. *This is valid in the current silicon architecture computers. Quantum computations could change this.*

4 Asymmetric Encryption

Cyphers with *two* directional keys instead of one. What is encrypted with key A can be decrypted with key B and viceversa. What you can't do is

- decrypt the key with the *same* key you used to encrypt the key
- deduce (calculate) one key from another

This approach enables "public key scenarios" due to the asymmetricity of the encryption mechanism. The robustness of this methods relies on the function used to encrypt the plaintext, that must be easy to apply *but difficult to reverse*. To make a public key scenario works we must add to the set of trusted elements (that contains the private keys) a *trusted assumption*: "only Bob knows Bob's private keys".
In asymmetric encryption the key lenght is an important measure of safeness. However, while in symmetric cypher the lenght of the key measure the number of decription attempts, in asymmetric cypher it measures the *number of key-breaking attempts*. That implies that asymmetric cyphers are "easier" to crack because of the approach chosen to encrypt. This also means that asymmetric and symmetric cyphers cannot be compared only using the key lenght.

4.1 Hash Functions

A hash function is a function that maps an *arbitrary lenght string* in input on a *fixed lenght string* at the output. Usually, input strings lenght are much longer than output ones $\Rightarrow$ *collisions*. A good hash function satisfies three properties:

1. preimage attack resistance: the function is *hard to invert*.
2. second preimage attack resistance: it must be difficult to find a input string that have the same hash to another one *given* (if the function is not preimage resistant it's not second preimage resistant too. The opposite is not true tho).
3. collision resistance: it must be hard to find 2 inputs with the same output.

4.1.1 Attacks to Hash Functions

Hash function may be broken. This means that it's possible to find the original text from the cyphertwxt *faster than bruteforcing*, just as the definition of broken cryptosystem. Two methods are used to attack a hash function:

Arbitrary Collision This is a first/second preimage attack: the attacker tries to deduce $x \mid H(x) = h$ or, respectively, $y \mid y \neq x \wedge H(x) = H(y)$.
Again, an hash function is broken if this can be completed *faster than bruteforcing it.*

Simplified Collision Attack Exactly the same approach as the arbitrary collision attack, but exploiting some probabilistic aspects of the collision (birthday paradox).

4.1.2 Digital Signature

Classic digital signature mechanism: an hash function produces a digest of a document, that's then encrypted with the private key of the sender. The receiver (to both check for identity of the sender and the integrity of the document) does the same: hashes the document and produces a digest, that's then compared with the decrypted version of the received one.
A big issue in this case is "who to trust". To estabilish a list of trusted elements the current infrastructure relies on Certification Authorities, a tree-like structure (a hierarchical group) of entities that each signs with a trusted key (a trusted role) the certificates.

Part III
Authentication

5 Identification and Authentication

What's the difference between identification and authentication? The first represents the act of "telling your identity". Authentication means to *proof* the stated identity. Authentication must be *bidirectional* or *mutual*: both entities must acknowledge each other identity and the proof.

5.1 Authentication factors

To identify someone, three methods can be used:

1. Authentication by "to know" factor (passwords, pins)
2. Authentication by "to have" factor (cards, ids)
3. Authentication by "to be" factor (fingerprint, voice)

Multi factor authentication uses more than one factor to authenticate.

5.1.1 To Know Authentication

To Know authentication systems relies on something the user willing to authenticate remembers, or is supposed to know. These systems are low cost, are easy to deploy and easy to use (due to the spread of such systems). Systems basing on to know auth are systems that rely on passwords, pins, login data combination etc.
On the other hand, the secrets the user is asked to keep can be easily stolen or guessed, or even cracked (bruteforced).

Countermeasure Against Password Wearing Three different attacks can moved against a passwrod-secured system, each one has a valid countermeasure:

1. against snooping, it's valid to change password often;
2. against cracking, it's useful to enforce complexity of the password;
3. against guessing, it's useful to ask for a nonsense password, or at least not easy to pin to the user;

A system cannot put in place all these countermeasure, because they are *costly*: they heavy the use of a system (a system that asks for a 64 mixed characters every 2 days).

Secure Password Exchange, Challenge Response Authentication phase, exchange of messages between two entities A and B that mutually authenticate each other

1. A to B: This is my identity. Let me authenticate
2. B to A: nice. Compute hash(random-data + secret) + random-data
3. A to B: hash(random-data + secret)
4. A to B: nice. Compute hash(random-shit + secret) + random-shit
5. B to A: hash(random-data + secret + random-shit)

Where secret is the password, generally. The supposition is that both entities *know* the secret.

Secure Password Storage Passwords are the secret upon it's based the security of a system (an authenticating one, at least). How to enforce password security?
To protect a file (data in general) the techniques are always the same:

- Encyption
- Salting (modifying stored passwords to mitigate dictionary attack)
- Access Control
- Password Recovery procedures that not disclose sensitive infos

5.1.2 To Have Authentication

To Have factor relies on *a phisycal device / object* that must be associated to a person willing to authenticate. The problem remains the human factor: this time the security of the system depends on how the key (secret) is handled by the person associated to it.
In any case, to have authentication system are low cost and offer a good level of security. On the other hand, they're hard to deploy, and the tokens / cards are easy to be lost or destroyed / compromised.

5.1.3 To Be Authentication

To be authentication asks the person willing to authenticate to prove to be who s/he declares to be by biometrically satisfy a costraint (like having a certain fingerprint, hand geometry, iris pattern...).
Although this kind of authentication is very effective (it's *really* difficult to emulate someone else's fingerprint) the list of disadvantages for biometric security systems is quite long:

1. They're hard to deploy and set up. Moreover, usually during the setup phase an invasive test must be performed
2. The matching of the feature selected is non-deterministic: accepting or rejecting a person depends on a threshold of accuracy, and this can act differently from time to time
3. They're not *entirely* secure: a fingerprint can be cloned
4. Biometric features *change* over time
5. Privacy issues: people *may not want* to distribute personal features involved in a biometric security system

5.1.4 Single Sign On

Single sign on was proposed as a system to countermeasure to password reusing. This system is based on a 1-2 factors authentication and *a single trusted host*. This is simply the "delegating authentication to someone else" method of authentication, the *external auth* method. Two main approaches:

- SAML approach: an external service serves as authentication factor
- OAuth approach: the external service *just helps the authentication*, giving the user a code in order to sign on to the principal service

This is also complicated to implement: an app should (in order to implement OAuth or SAML) complicate a lot its flow of usage (user experience) in order to just authenticate a user.
Moreover, the trusted factor is a SPOF.

Part IV
Web security

Part V
Exercises and Examples

6 Exercises

6.1 Identity theft

Which are the risk components in the scenario of identity theft, like in social media platforms?

1. Reputation threathened both of the victim and the company
2. Social engineering attacks
3. Malicious spread of information

And the assets?

1. reputation
2. money
3. personal information

And who are the threat agents?

1. People around you
2. Government
3. etc

So *everybody* with a malicious intent
Why is this scenario possible, and how to mitigate it?

1. 2FA
2. trusted device based auth
3. enforce credentials strongness

6.2 Ransomware attack

A small company specialized in a particular object is infected by a ransomware
What are the threats, the asset at risk and the possible countermeasures?
First:

- data loss
- data of the company

- offline backups

Second:

- data leakage
- data of the company
- encryption

Third:

- denial of service
- production
- redundancy

Who are the possible threat agents? Cuncurrent companies, ex internal employees, ransomers...

6.3 Auto driving vehicles

Consider auto driving cars used as taxies.
Which are the most valuable assets in this scenario?

1. People inside the car
2. People outside the car
3. the vehicle itself

What are the attack surfaces on the vehicles?

1. The VEHICLE itself, you enter there!
2. the connectivity capabilities of the vehicle
3. the company network, or the internet itself (it was an hypothesis)