Security Engineering Notes

Security Engineering

- CIA
  - Confidentiality
  - Integrity
  - Availability
- Some other attributes include,
  - Authenticity
  - Privacy
  - Accountability,
  - Non-reputability
  - Receipt-freeness (for voting systems)
- Methods of attack
  - Brute Force
  - Deception/Social Engineering
  - Calculated attack on a vulnerability
  - Insider (both intentional and unintentional/accidental)
    - Mitigation techniques: Keeping logs and conducting audits could. Also least privilege to prevent accidental insider attacks.
  - Chance (right place at the right time)
  - Accidental
- Reliable systems (protect against random failures)
- Secure systems (protect against targeted deliberate attacks)
- Assets which need to be protected
  - Tangible (eg. documents)
  - Non-tangible (eg. reputation)
- Security Engineering: Protecting **assets** against **threats**.
- Kerckhoff's Principle – "the security of the cryptosystem must not depend on keeping secret the crypto-algorithm. It must depend only on keeping secret the key."
  - Because, its often easier to change the key after a compromise or possible compromise than changing the algorithm,
  - and it's usually also harder to keep the algorithm secret (can be reverse-engineered or bought) compared to the key

Crypto

- **Code** (hidden meaning, ie. A spy using a phase that seems normal, but actually has some other hidden meaning)
- Plaintext -> E(encryption key) Ciphertext -> D(decryption key)
- **Symmetric-key cryptography** – Encryption key can be calculated from the decryption key and vice versa (usually they are the same though). The sender and receiver must agree upon the keys first. Two types,
  - **Stream Ciphers** – operate one bit at a time (eg. RC4)
  - **Block Ciphers** – operate on a group of bits at a time (eg. DES and AES/Rijndael)
    - Block chaining is used to make each block of cipher text depend on **all** the previous blocks of plaintext.
- **Public-key Cryptography** (asymmetric-key) – public and private keys where you cannot (or is really hard to) calculate one from the other. (eg. RSA)

- One problem is that an attacker can use an adaptive chosen plaintext attack, where they encrypt a large number of messages using the target's public key. A fix is to pad messages with a long random string.
- Can be used to do **digital signatures**. (encrypt your message M (though in practice a one way hash is used instead) using your private key, verifiers just decrypt using your public key as only someone with the private key (ie. you) could have successfully encrypted M)
- **Birthday attack** – find two messages that have the same hash but only differ by say spaces at the end of the line, and some other key part like dollar amount. Then when you get Alice to sign document A, the signature will be the same for document B (as Alice is signing the hashes and the hashes are the same)
- **Public Key Infrastructure** – how to ensure that Bob's public key really belongs to the Bob you want to communicate with (tries to solve the key distribution problem).
- Public Key Certificates
- Certificate distribution (X.509, GPG)
- Most cryptosystems use public-key crypto just to exchange the symmetric key shared key, and then both parties this shared key to communicate with symmetric-key crypto. These systems are known as **hybrid cryptosystems**. This is done for mainly two reasons,
- public-key algorithms are usually around 1000 times **slower** than symmetric algorithms.
- public-key algorithms are **vulnerable to chosen-plain text attacks**.
- Cryptanalysis is about determining the plaintext from the key.
- Authentication
  Extra notes from Schinder 2nd Ed.

- Types of attacks,
- cipher-text only
- known plain text
- chosen plain text
- adaptive chosen plain text (you can modify your next choice based on the results of the previous choice)
- chosen cipher text
- known key relationship
- rubberhose (get the person who knows the key to reveal it)
- **Substitution ciphers**
- Simple (monoalphabetic cipher)
- A → 2
  B → 3
  C → 1
  …
- Homophonic (where multiple options available such as 5,2 one is chosen at random)
- A → 5,2
  B → 1,3
  C → 4
  …
- Polygram
- ABC → 819
  ABB → 234
  ABA → 567
  ACA → 587
  …

- Polyalphabetic (simple substitution cipher, but the mapping also depends on the position of the input character in the plain text), examples include,
  - Caesar cipher – shift each letter forward or back by K. eg. if K = 1, A → B, B → C …
  - rot13
  - **Vigenere cipher** – **align the repeated key K over the plaintext**
- These can be attacked by statistical cryptanalysis of the letter frequency
- **Transposition Ciphers** – the order of the characters is changed (rotor machines such as the Enigma did this)
- eg. write across, but read down

```
HELLO
WORLD
THISI
```

```
SSOME
```

encrypts to HWTSEOHSLRIOLLSMODIE

- **One time pads**
- **One-way hash function** (also known as message digest, fingerprint, cryptographic checksum)
- pre-image → *f(x)* → hash value
- A hash function is collision free if it is hard (ie. at best becomes brute force) to generate two pre-images with the same hash value.
- Also a single bit change in the pre-image should on average change half of the bits in the hash value.
- Types of **random sequences**,
- Pseudo-random (looks random, good for applications like writing an random AI game agent)
- Cryptographically secure pseudo-random (unpredictable)
- True randomness (cannot be reproduced with the same inputs)
- **Zero-knowledge proofs** – proving to someone that you know something without revealinfg them what that something is. Two types,
- Interactive
- Non-interactive

RSA

Choose two primes *p* and *q* and let *n* = *pq*. Choose *e* such that *e* and (*p* − 1)(*q* − 1) are relatively prime (ie. no common factor and both prime numbers). Let d be a solution of ed $\equiv$ 1 mod (p-1)(q-1).
Public key $K = (e, n)$, private key $K^{-1} = (d, n)$.

$$E_K(M) = M^e \mod n$$

$$D(M) = E_{K^{-1}}(M) = M^d \mod n$$

Access Control

- "An access control policy is a description of **who** may access **what** (and **how**)."
- **Principle of least privilege** – "The powers that an agent in the system is given should be the minimum that they need to have to play their assigned role in the system."
- AC can be done at many levels (eg. hardware, os, database, network, application…)
- Managing **Subjects**, **Objects** and **Operations**. (can use Groups or Roles to classify subjects)

- **Access Control Matrix** (central database of all objects, subjects and operations permitted)
- **Access Control List** (store permissions with object)
- **Capabilities** (store permissions with subject)
- **Mandatory AC** – central entity sets the AC.
- Multi-level (eg. top secret, secret, unclassified labels…)
  - Bell-LaPadula – no read up, no write down.
- **Discretionary** – each object has an owner, and the owner sets the AC.
- Commercial Policies
- Chinese Wall – eg set up rules like if a subject can access X they cannot access Y.
- Dual Control – need two people together to modify X, they cannot do it on their own.
- Reference Monitor – some entity that sits between the subjects and objects to implement some policy (relies on trusted entity)

Elections

You want your elections to be,

- verifiable – so the voter can check that their vote was indeed counted (and if possible check that everyone else's vote was counted correctly)
- correct – votes are counted correctly and the results match calculated correctly
- secret (ie. no one knows how you vote, so that you cannot be coerced)
- coercion free (so you want to be recite free)
  (additional notes, but don't really need to know for exam)

- True Democracy (like the Greek's assembly)
- Representative Democracy (where we vote in someone to represent us)
- Participatory Democracy (where we represent ourself an all have equal say)
- Australian Federal Elections are subject to the Mafia problem. Because we have preferential voting that is you can order your preferences that means that for n candidates there are n! ways of voting. So if there are enough candidates and few enough people you can coerce a person into using a specific voting pattern which you can then verify that they indeed did vote that way.
- An alternative attack would be to obtain a blank voting paper, fill it in the way you want it to be give it to someone and tell them to submit it and return you a blank one. Although the attacker won't know if the victim has scribbled it out to make it invalid, but at least the attacker has prevented someone from voting.

Security Architecture

Security Design Principles:

- Economy of Mechanism – keep things simple
- Fail-safe defaults
- Complete Mediation – every access request checked
- Open design, ie. Kerckhoffs' Principle: Keep the key secret, don't rely on keeping the algorithm secret.
- Separation of Privilege
- Least Privilege
- Least common mechanism
- Psychological acceptability
  Defence in depth – use many layers of security (or many security perimeters in layers)

Human Factors

- People are trusting
- People are lazy (to read the manual or change factory defaults)
- People are greedy (will exploit the system if given the chance)
- People are forgetful (forget to revoke access when terminating employees)
- People are selfish
- People are stick-beaks (eg. Medicare employees accessing client data for personal interest)
  Some strategies for reducing the risk,

- logging of insider's actions
- honeypots for insiders (eg. fake web server running on the network, or fake data in the database)
- security policy for staff to follow

Risk

Risk is not something I would have thought to be in a security course, but now that I think about it there are few if any bullet proof systems, so there is always some risk.

Whether it be secure communication (there is always some risk that an eavesdropper cracks your cryto and reads the message, so its important to weigh up those risks to decide if its worth sending the message in the first place), or be it running a web server (you cannot be sure that your web server doesn't have bugs, or even if you have verified the code to be matching the vulnerability free specification other things can happen like, has your CPU been verified to be bug free, are you sure that a cosmic ray won't flip a bit and subsequently create a vulnerability). So weighing up the risks involved is important to decide how much time and effort you devote to securing a system, or how the system is designed to work.

Business Risk Concepts

- Exposure – what is the worst case scenario (or loss)
- Volatility – how predictable is the loss
- Probability – how likely is that loss

Degree of Risk

```
       <-- Probability -->
High Exposure/  |  High Exposure/         /\
Low Probability |  High Probability       ||
---------------------------------    Exposure
Low Exposure/   |  Low Exposure/          ||
Low Probability |  High Probability       \/
```

Exposure – how widespread is the system?
Probability – how easy is it to exploit the system, and how great is the incentive to do so (which relates to how valuable the assets you are protecting are)?

Risk management

- Example. The risk of viruses; the mitigation strategy may be to use anti-virus detection, but the residual risk is zero-day attacks.
- **Carry the risk** – just accept it.

- **Balancing the risk** – eg. data backups at multiple locations, distributed servers at multiple locations (perhaps even running different systems, so for instance if you want to keep your web site up and running you may load balance between both a Linux box running Apache, and a windows box running IIS, so that an attack found in one is not likely to be present in the other so the attacker can't take both offline with the one attack.)
- **Mitigate it** – reduce the risk
- **Transfer the risk** – eg. in finance, get insurance

Source: http://andrewharvey4.wordpress.com/2010/07/31/security-engineering-notes/