Security

Read from the textbook

- Ch16.2 Program Threats
- Ch16.3 System and Network Threats
- Ch16.6 Implementing Security Defenses
- Ch16.7 An Example: Windows 10

Knowledge - Passwords

Extremely common way of gaining unauthorised access to computer systems.

Password guessing

- brute force – try every possible combination
    - The system should spend several seconds before replying and deny access after a few attempts.
- The greater the number of symbols and the length of the password the harder it is.
    - Many Unix systems used to have a maximum password length of 8.
- intelligent search
    - try default passwords (unfortunately common)
    - sometimes the user hasn't even set a password
    - words associated with the user – names of friends, relatives, pets, dates, hobbies, phone numbers, and the same things backwards (Facebook is great for this)
    - common passwords – "sesame", "password", …
    - dictionary attacks – word lists

Most commonly done when password files are extracted from a site. Can find a very high percentage of hits. Rainbow tables (precomputed).

- Substituting numbers for similar letters e.g. 3 for E.
- Typing one row higher or lower.

Stealing Passwords

- Shoulder surfing
- Video cameras
- Snooping on a network for plain text passwords
- Keyloggers
- Trojan horse login screen
- Keyboard sniffing
  - wireless keyboards
  - but also
    - can use the electromagnetic radiation emitted when keys are pressed on wired keyboards

Making Passwords Safer

1. Don’t write them down.
2. Use mixed upper and lower case letters with numbers and symbols.
   Better to use the first letters of a phrase
3. Change them occasionally (but not too often).
   - This can be enforced by the system, along with other common password requirements.
   - Usually prevent the user from using an earlier version.
   - Unfortunately this makes it harder to remember and hence the user is more likely to write it down.
4. Have system produced passwords.
   - Random but pronounceable
   - If people forget their passwords they need the sys admin to give them a new one.
   - This also requires authentication.
   - Many passwords have been bullied out of sys admins over the phone

- See http://xkcd.com/936/ “correct horse battery staple”

Password Files

- The password has to be kept somewhere.
- Either keep the password file secret or one-way encrypt its data (preferably use both).
- If the file is readable they can be broken by dictionary attacks or rainbow tables
- One-time passwords
  - a new one produced at the end of a session
  - security tokens - time based / algorithm based
  - challenge/response
- Rather than memorising a password an algorithm can be the secret.
- System issues a challenge e.g. an integer.
- The user responds with the value of using that integer as input to the algorithm.
- Can be made one-time by using secret seeds that are generated each use. In this case the user needs the algorithm (and seed) on another protected computer or smart card.
- Two-factor authentication - e.g. PIN and message sent to your phone

Program Threats

When **a user runs a program written by another user** there is always the potential for misuse.

- Trojan horse
  - A program that has hidden side-effects.
  - Trojan horses can be hidden in search paths.
- Spoofing attacks and phishing e.g. man-in-the-middle or presenting fake login screens.
  - This can be stopped with non-trappable key sequences or reporting the number of unsuccessful login attempts.
- Backdoors
  - Leaving hidden access to the programmer.
  - Disgruntled employees.
  - Compromised compilers - producing compromised compilers.
- Logic bombs
  - Goes off under particular circumstances.
  - If I don’t login every week wipe all files.
- Root-kits
  - Replace standard commands with versions which hide the presence of the kit. Usually used to keep access hidden.

Mobile Security

Extra problems.

- Portable devices
  - Always with us, easy to lose/steal
- Convenience required
  - We don’t want to make using the device difficult
- How can we make it safe?
  - minimise the attack surface
  - code signing
  - user permissions to do things
  - sandboxing

Attack Surface

- What code can be run without authentication?
- The less code which is reachable from outside the better.
- Many security risks have been found in 3rd party layers such as Java and Flash environments in browsers, and components in pdf readers.
  - Getting rid of these immediately increases the difficulty that attackers have to find exploits (bugs which give them access that should not be allowed).
  - Removing this type of functionality is reducing the attack surface.

Checking Before Running

- Only running code which has been checked strongly reduces the chance of running malware.
- Apple’s App Store and Google’s Google Play
- They perform checks on submitted apps (playing the role of an anti-virus program).
- Many AV programs are ineffective on mobile devices (and not allowed on iOS)

Code Signing

- Code signing is used for very different purposes on iOS and Android.
- In iOS Apple signs every app (all apps must come from the App store)
  - This means that you can be sure the app you install and run is the same code that Apple verified. It has not been altered.
  - The app is only installed if properly signed and when the program runs each page of memory is checked to make sure it has not been altered.
- In Android apps are signed by the developer (there is no need to use a CA but it is a good idea). Apps can come from anywhere (much safer if you only get them from Google Play).
  - The code signing is so that updates can be verified to come from the same developer and establishing trust between apps.
  - Jailbreaking removes most of the checks on code signing (actually allows other signers, including self signed certs).

Permissions

- Usually not a good idea to allow all apps access to all of the services on the device. e.g. Granting read access to messages to an app that doesn't need them.
- Android before version 6 asks the user at install time.
  - Presents a list of privileges it wants. Remember that the app is only checked by Google if you use Google Play.
  - Supposed to be minimal but it was up to the developer.
  - No explanations were given.
  - Most users just clicked "accept".
    - The program was not installed if rejected.
  - After version 6 asks at runtime.
- Apple didn't use to ask users except for location information
  - If an app was passed by the App Store it could do anything it wanted.
  - No longer true - now as apps request access to photos or messages the user is asked to allow or not.
  - If the user turns down a permission the program is still supposed to run (but obviously it can't do everything).

Sandboxing

- If an app goes bad or allows an external exploit the last protection is the sandbox.
- In Android the default is files in internal storage are only accessibly by the creating app. (Can be made accessible to others.)
- Files on external storage (SD cards) are globally readable and writable. Not so on iOS (trick question, why not?).
- In iOS all files can only be created in the sandboxed area and there is very limited ability to pass information from one app to another. (Now there are controlled ways since iOS 8 to share data.)

Before Next Time

Read from the textbook

- Ch12.3 Application I/O Interface
- Ch12.5 Transforming I/O Requests to Hardware Operations