Android Security

Android Security

- Like any operating system or application platform Android has paid significant focus to the security of the system
  - An insecure system is unlikely to adopted by users
- In this lecture we will cover a range of design decision that attempt to secure the system
  - Things like sandboxing, permissions, use of the linux kernel as a base etc

Android is a linux based system

- The first decision that was made was to base the Android OS off the linux kernel
  - Has been in existance and use since 1991
- Is an open source piece of software
  - Meaning anyone can observe the code and suggest patches
  - Also Android adopts a lot of linux centric security measures by default

Open source nature of the OS

- The open source nature of the kernel leads to increased security of the system
  - The reason for this is that code and patches are completely visible to everyone
- Meaning that all patches can be analysed and if they show suspicious behavior or introduce bugs then they are rejected
  - Thus any patches that are accepted will be of sufficient quality to remove bugs or security issues.

How open source enhances security

- Android itself is an open source project
  - If a security issue is spotted then contributors outside of google and suggest patches to fix errors
- Will go through a checking process by the main development team
  - Any approved patches will be brought into the mailine

User ids and group ids

- Linux usually imposes an id on every user and group within a system.
  - A user is an individual while a group is a collection of users. Anyone outside this is considered “other”
- Permissions for access to files including read write and execute operations are managed by these permissions
  - So a permission list like rwxr-xr-- states that the user that owns the file can read write and execute it while those in the group can read and execute but outsiders can only read.

How user ids and group ids are assigned

- Android uses this system of uids and gids in relation to applications
  - Whenever an application is installed a new user and a new group is generated and assigned to that application
- These permissions will then be used to determine if what files and directories that application will have access to
  - Used heavily in the sandboxing process

Sandboxing

- All applications are sandboxed in android
  - A sandbox is simply a container that houses all of the applications code and data
  - In android this takes the form of a single directory
  - Applications are permitted to make any changes they wish to their sandbox

Sandboxing

- To enforce this Android will set the uid and gid for everything in that directory to be the same uid and gid of the owning application
  - And will set all permissions to rwx------
  - This means that each application is isolated from each other
  - One application cannot modify or interact with the data of another

Sandboxing

- All other files on the system are either
  - Owned by another application in a seperate directory
  - Or files outside of this are owned by the root user
    - And other users have limited access to touch or modify those files

File system structure

- To complicate matters further the filesystem is divided into two regions
  - They are internal storage and external storage
  - Internal storage covers data belonging to an application
  - External storage covers data shared with all applications

Internal storage

- Internal storage contains everything within the sandbox of the application
  - This data is kept private to the application
  - Is not visible or modifiable by other applications
  - Generally if you have any data that should not be shared with another application it should be stored here (user information etc)

External Storage

- Data that is stored outside the sandbox and is ment to be shared with other applications
  - Generally all types of media will be stored here including images and audio.
  - For example a camera will put all images here as they are intended to be used by other applications
  - Only non sensitive data should be stored here

Permissions for sensors and access

- The reason for enforcing such isolation between applications and the system is to force applications to declare the resources that they require in order to function
  - Thus android needs to be informed prior to app installation what permissions the app requires in order to function
  - This is to stringently control permissions from app to app
  - And also inform users what the app has access to on the device

Role of the manifest file

- The manifest of an app is where the permissions will be stated.
  - Every permission required must be listed here
  - If the user accepts the required permissions the application will be installed and will be given those permissions by the android OS
  - If at this point an app tries to access permissions it does not have android will immediatly kill it as it is considered a security risk

Default permissions of every application

- The bare minimum permissions that every application has are the following
  - The right to update the screen and generate sound
  - The right to interact with user input
  - The right to read and write to internal storage
  - The right to use notifications and vibrations
  - Amongst other very basic permissions

Whitelisting and blacklisting

- What you see here is an example of the whitelisting approach
  - Whitelisting as applied to permissions states that all apps are given no permissions to begin with and are allocated permissions as required
  - Blacklisting as applied to permissions states that all apps are given all permissions to begin with and permissions are recinded as required
  - From a permissions perspective whitelisting makes much more sense than blacklisting

Why whitelisting is used

- The whitelist approach is used to encourage developers to use the minimum set of permissions required for an application
  - As the list of permissions grows users start to become skeptical of what the application does.
    - Particularly when sensitive data is involved
- Also encourages users to read through the list of permissions and spot suspicious applications.

Advice about permissions

- Always use the absolute minimum amount of permissions possible
  - If you don't need a permission get rid of it
- If you require a permission give the user clear and consise information about why you are using this information
  - Especially when user data is involved as data protection law states that users are allowed to query the data collected on them and the purpose of that data

Why the human is the weakest link in the security chain

- Unfortunately all of these security measures are useless particularly if the human who owns the device is not the best with regards to permissions
  - But as time goes on users are getting better at recognising potential threats to their devices
  - There are a number of tricks that applications and unscrupulos developers will use in order to try and get an application installed with lots of permissions
    - Particularly through the use of social engineering

Social engineering

- A common trick of viruses and scams. Here applications will attempt either immediately to get access to all permissions or else will slowly ask for more over time
  - Remember there is no patch available for human stupidity
  - Social networks are one of the best examples of this. Initially started with little or no permissions but over a few years has grown to nearly the full permission list
  - If you can get past the user you circumvent the whole security system.

App security

- While android does provide a series of mechanisms to prevent security issues it does not mean you are free to write apps without security considerations
  - If you can learn about the topic of secure programming
    - Will vastly help in reducing security issues in your code
  - Also there are other pieces of advice that Google offers that I will talk about here on the subject of security

Networking Security

- If your application uses any form of networking you should consider using secure forms of transport
  - Particularly around login/logout and transmission of sensitive user data
  - HTTPS and SSL sockets are readily available in the API
    - Gives you some level of encryption and protection against man in the middle attacks

Telephony

- Communication gets particularly sensitive around the issue of using phone services and SMS services
  - A lot of rogue apps send SMS messages or make calls to premium rate numbers as a means of illegaly acquiring money
  - Be very careful if you require either of these services and explain in detail why you require them.

Credentials

- With regards to credentials try to ask for them as little as possible
  - Try not to store username/password combinations and if possible use authorisation tokens for authentication
  - The less a user has to provide username/password information the less the information will be transmitted over the network
    - And subsequently eavesdropped on

Cryptography

- In the cases where you have to store user sensitive information encrypt it
  - The API provides a number of ciphers (including AES and RSA) in its Cipher class
  - A secure random number generator is also provided in the SecureRandom class
  - Don't implement your own encryption mechanism.
    - Likely to be much weaker than what is already provided

Input Validation

- One of the most common methods that attackers will use is to inject code into forms that are not properly validated
  - Particularly if there is SQL or JavaScript involved.
  - All input that you recieve (user or network) should be validated stringently to ensure it adheres to the datatype that you are expecting
    - If it looks suspicious in any way reject it immediately