

# Secure Mobile Application Development Introduction

**Unit 5**

- 1 Basic Principles of Secure Development
  - Secure Software Development Life Cycle
  - S-SDLC in agile environments
- 2 Good Practices in Secure Development
- 3 Good Practices in Mobile Development
  - Protection of the application
  - Sensitive data management
  - Logs and sensitive data leakage
  - HTML components in mobile applications
  - Communication between applications
  - Authentication in mobile applications
- 4 Laboratory
  - Android
  - iOS
- 5 Research exercises
  - Assessment test



A photograph of a workspace with multiple Dell monitors. The left monitor shows a collage of images including a laptop and a desk with a clock. The middle monitor displays a starry night sky. The right monitor shows a document with text. In the foreground, there is a white keyboard, a white mug with a blue Twitter logo and the text "#milknosugar", and a blue pen. A blue semi-transparent banner is overlaid across the middle of the image, containing the title text.

# Basic Principles of Secure Development

# ◆◆◆ Principles of Secure Development

## Introduction

- Vulnerabilities and security problems may affect a software product during the whole development process if:
  - Security requirements are not described during the analysis stage.
  - Designs created have security failures.
  - Vulnerabilities are created during the implementation stage.
  - The software is deployed inappropriately.
  - Security incidents occurred have not a proper response.
- Such problems directly affect the developed software and information stored, but it may also affect:
  - Other applications that are executed in the shared environment.
  - The user's system (including mobile devices).
  - Other systems that interact with the software to develop.



A photograph of a brick wall with a window. A black 'ONE WAY' street sign with a white arrow pointing right is mounted on the wall. The sign is partially obscured by a semi-transparent dark grey rectangle containing the text 'Secure Software Development Life Cycle'.

# Secure Software Development Life **ONE WAY** Cycle

# ◆◆◆ Secure Software Development Life Cycle

## Introduction

- The Secure Software Development Life Cycle (**S-SDLC**) is the software development process that implements security as a transversal element during the whole development cycle.
- The implementation of security as a transversal element of the software development is called “Security by Design”.
- Most vulnerabilities **may be solved in the application development stage**, since many of such vulnerabilities are created when development processes and their associated controls are not implemented properly.
- The S-SDLC takes into consideration all the security aspects that may be involved in the software development from the beginning of the process.
  - It allows developers to detect vulnerabilities during early stages of development.
    - It saves costs in vulnerabilities detected in systems that are in production.
  - It allows developers to take into consideration requirements according to different regulations and standards from the beginning of the development process.
    - It saves costs in the implementation of new requirements or functionalities related to compliance.

# ◆◆◆ Secure Software Development Life Cycle

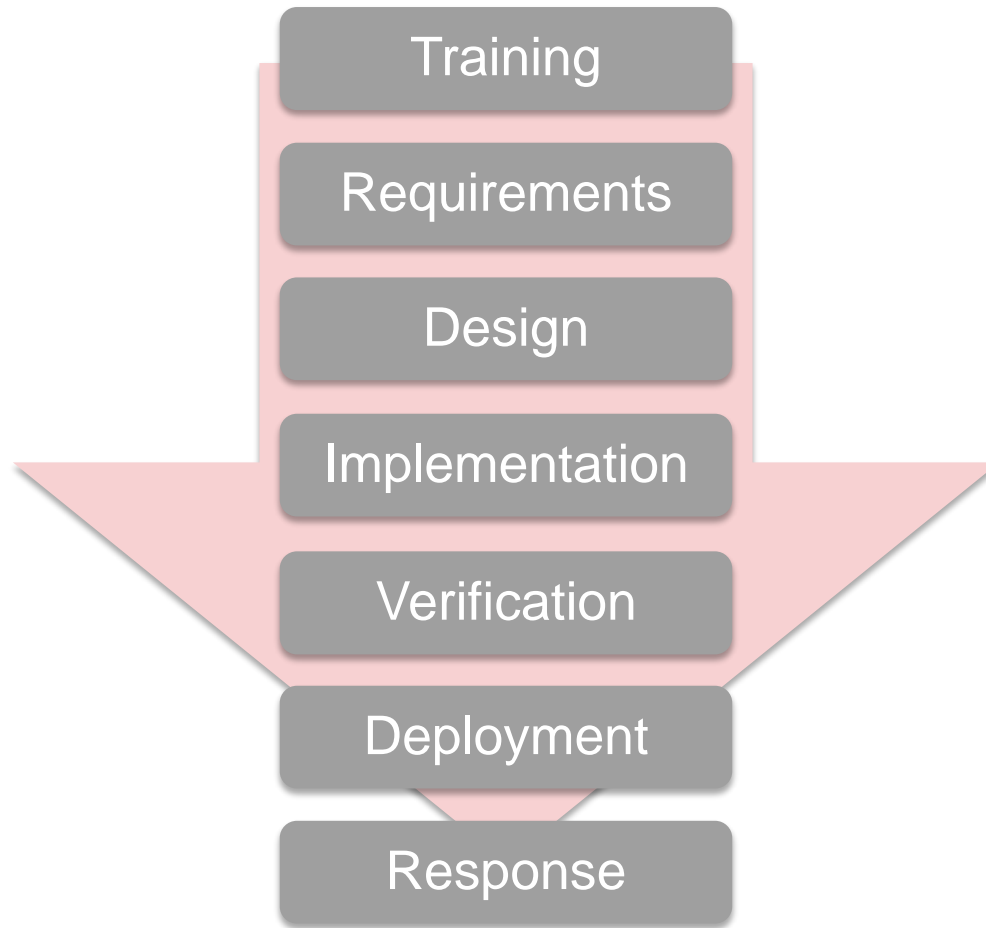
## Basic Characteristics of S-SDLC

- There are processes similar to the S-SDLC:
  - OWASP CLASP (Comprehensive, Lightweight Application Security Process):
    - [https://www.owasp.org/index.php/Category:OWASP\\_CLASP\\_Project](https://www.owasp.org/index.php/Category:OWASP_CLASP_Project)
  - Microsoft Secure Development Life Cycle:
    - Developed by Microsoft, but not applicable to all types of development.
    - <https://www.microsoft.com/en-us/sdl/>
  - Digital's Security Touchpoints:
    - Developed by Gary McGraw.
    - <http://www.swsec.com/resources/touchpoints/>
  - NIST 800-64:
    - Set of security considerations suggested by the NIST that should be taken into consideration during the SDLC.
    - <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-64r2.pdf>
- Generally, all the models include a series of security activities for the development life cycle.



# ◆◆◆ Secure Software Development Life Cycle

## Stages



# ◆◆◆ Secure Software Development Life Cycle

## Training

- It is not an S-SDLC stage strictly, but it is essential to perform it.
- The technical staff involved in the development of the project have to be able to perform all the additional tasks that the S-SDLC implies.
- They should be aware of the following concepts:
  - Secure architectures.
  - Threat Modelling.
  - Secure encryption.
  - Penetration testing.
  - Security and privacy practices.
- This stage is essential for the different roles involved in a development process to know their responsibilities from the point of view of security.



# ◆◆◆ Secure Software Development Life Cycle

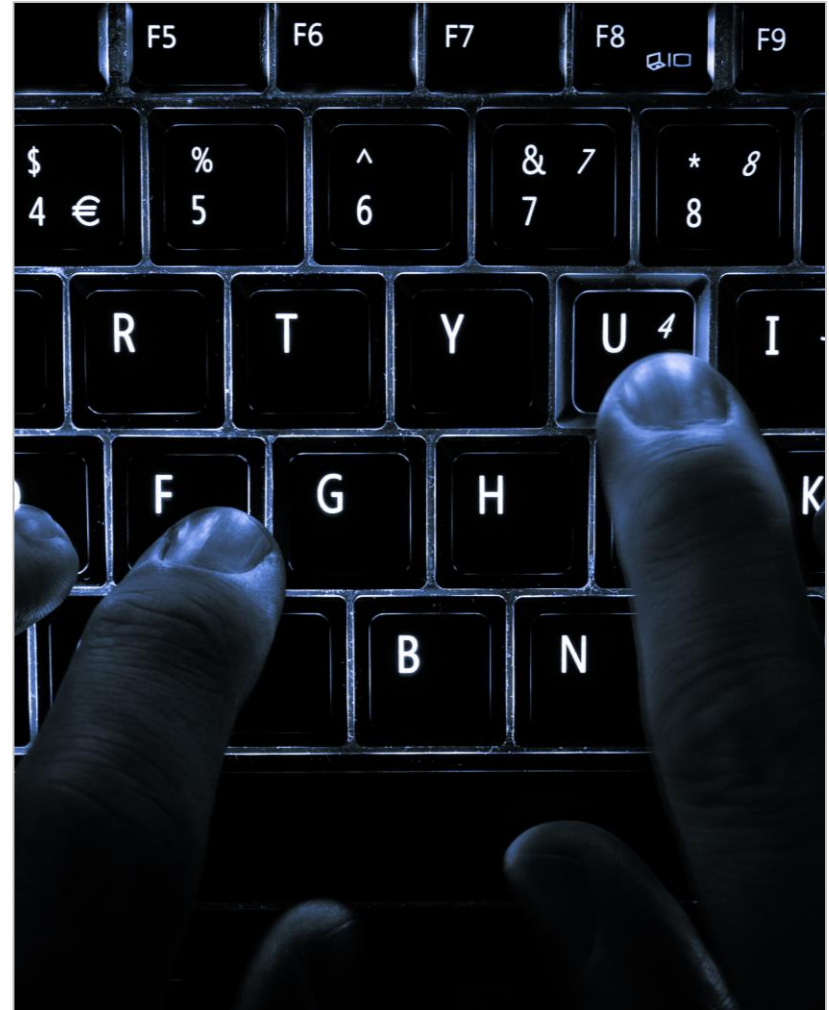
## Requirements

- During this stage, apart from traditional functional requirements of the application, further security, privacy and regulatory requirements should also be taken into consideration.
  - A set of minimum security requirements should be defined.
  - As in the case of the rest of requirements, it is important to implement the necessary measures to follow its development during the SDLC.
- Another possible mechanism would imply defining a set of security metrics that should be maintained during all the development stages:
  - Establish security levels for vulnerabilities.
  - Explain the acceptable maximum levels for each development stage.
    - E.g.: a product cannot pass the launch stage if it includes any critical vulnerability.
- In order to facilitate the identification of requirements, the following processes are carried out:
  - Identification of roles, capabilities and resources of the application.
  - Risk analysis.
  - Definition of abuse cases.

# ◆◆◆ Secure Software Development Life Cycle

## Design

- During this stage, the security solutions that will cover the security requirements explained in the previous stage should be described.
- Furthermore, in this phase, functional details that have not been specified during the requirements stage should be described.
  - Example: cryptographic algorithms to use.
- The S-SDLC adds a set of principles to this stage that should be followed for the design of the system. Such principles should be taken into consideration transversally during the design of the system.





# ◆◆◆ Secure Software Development Life Cycle

## Design – Principles of Secure Design I

- **Defence in depth:**
  - It implies creating different security layers so that, in case one of them fails, the system will not be compromised.
  - It requires the design of different defence strategies for the same threat.
- **Fail securely:**
  - It means that all the fails will take the system to a status that is considered as secure (without losing confidentiality, integrity or availability).
- **Least privilege:**
  - Each user or process should only have the least amount of privilege required to perform the necessary tasks.
  - Privileges should also be granted for the shortest possible time.
- **Separation of duties:**
  - The performance of any critical activity on the system should require the participation of two or more entities.
  - It is aimed at eliminating single points of failure.

## Design – Principles of Secure Design II

- **Keep security simple:**
  - When two solutions provide the same security level, generally, the less complex one should be used.
  - In general, the simpler, the smaller attack surface.
- **Supervision:**
  - During the execution of any task (access, writing, modification), it is important to verify that the user or process that is executing it has the proper authorisation.
  - In order to avoid synchronisation issues, it is recommended not to use authorisation caches.
- **Open design:**
  - The details of the system design should be open in order to avoid security by obscurity.
  - This principle is useful to create secure systems from the design.
  - It ensures that the design publication or revision will not imply directly a serious security incident.

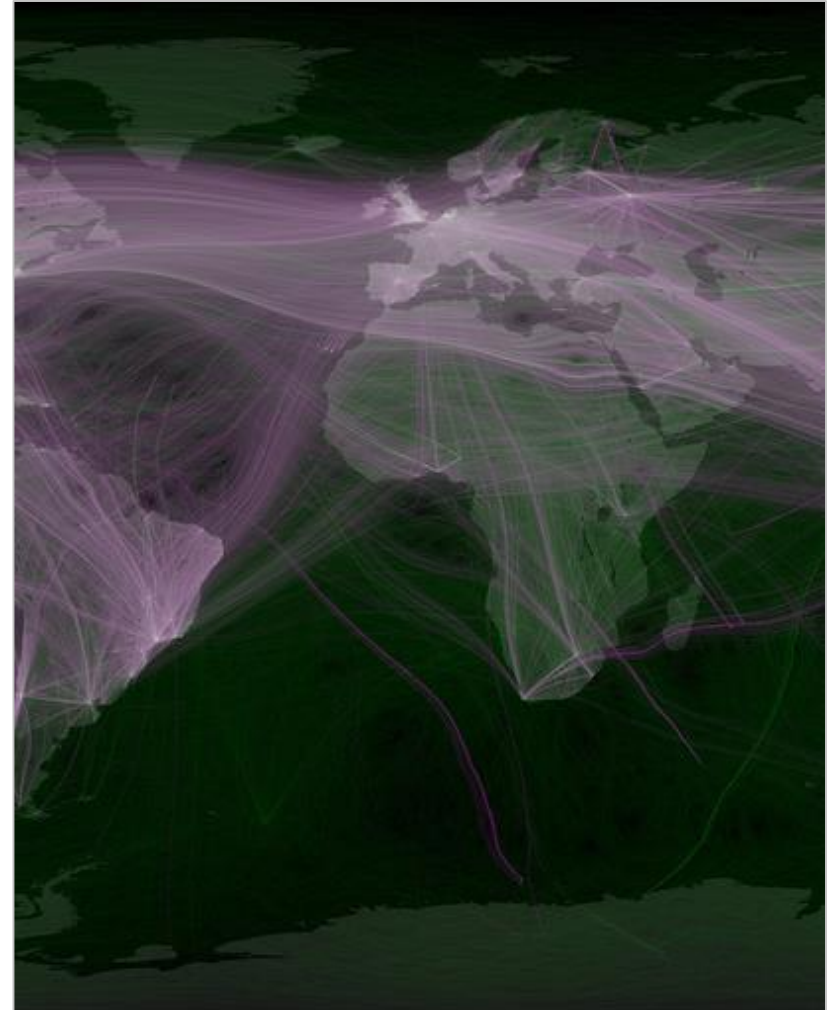
## Design – Principles of Secure Design III

- **Least common mechanism:**
  - This principle advises not to use the same security mechanism, even if it is common to various processes or users if they have different levels of privilege.
- **Acceptability:**
  - The security mechanisms of the system should be designed taking into consideration the acceptability of users.
  - If users face difficulties when using security characteristics, they will look for different mechanisms to avoid them, rendering them useless.
- **Weakest points:**
  - The security of an entire system will depend on its weakest point.
- **Reuse:**
  - It is preferable to use already existing and verified components than the creation of new ones that may increment the risk of vulnerabilities and the attack surface.

# ◆◆◆ Secure Software Development Life Cycle

## Design – Surface of Attack

- It implies specifying the entry points to the system in an structured way. This task should be performed by the designer.
- The entry points of the application can be divided into:
  - Network.
  - File System.
  - User.
- For each entry point, the following elements should be identified:
  - Resources accessible through it.
  - Roles that have access to such access point.
- It allows users to identify resources leakages to roles that should not have the required privileges.

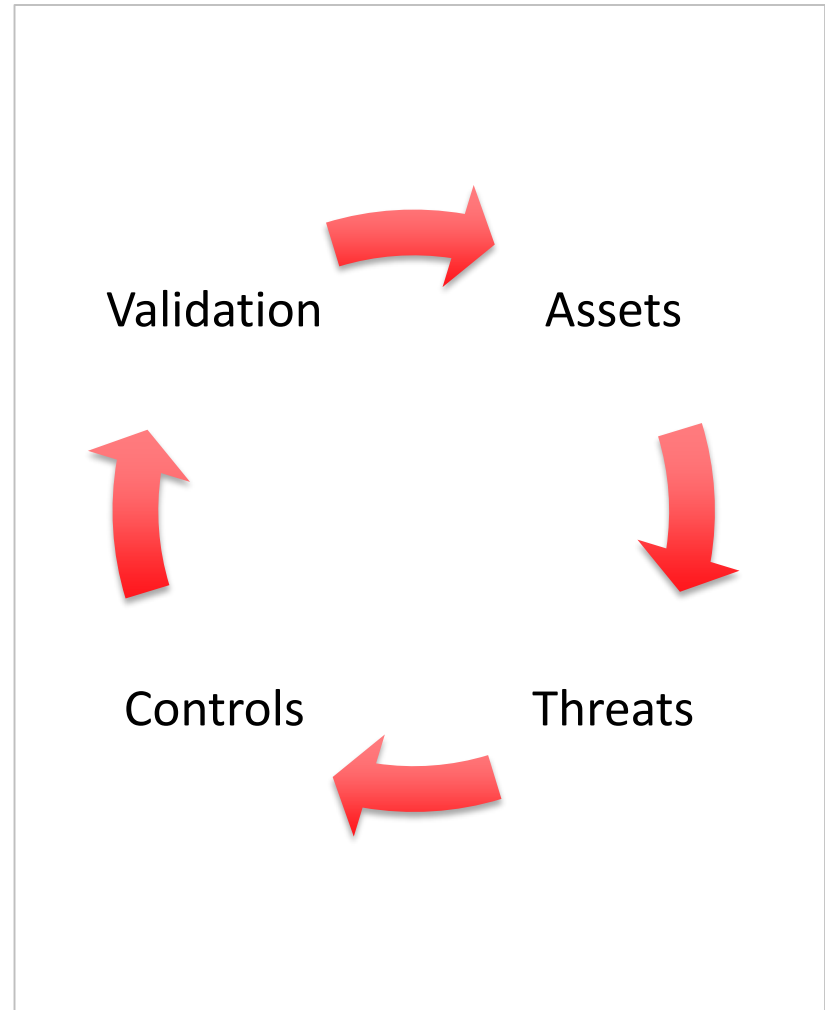




# ◆◆◆ Secure Software Development Life Cycle

## Design – Threat Modelling

- As explained in unit 3, risks and threats that may affect a given system should be catalogued and assessed.
  - It will be the first task carried out by an attacker.
  - If it is not performed, the protection of systems is reduced.
- Threat != vulnerability. Threats are persistent.
- The threat modelling is an iterative process that implies:



# ◆◆◆ Secure Software Development Life Cycle

## Design – Threat Modelling

- Identify assets or capabilities existing on the system by using a diagram. It is important to check whether they coincide with the ones identified in the documentation.
- For each asset or capability, identify potential threats.
  - This task requires the analyst to have some creative skills.
- For each threat, assess the risk existing:
  - Use threat trees that describe the different steps that should be followed by the attacker in order to materialize threats.
  - Measure factors such as: impact, reproducibility, exploitability and affected users.
- For each threat, identify controls that can be implemented to mitigate it.
- At the end of the process, as many threats as possible should be covered.

# ◆◆◆ Secure Software Development Life Cycle

## STRIDE

- STRIDE is a threat model that groups them into six categories:
  - **Spoofing:** a system or user is masqueraded.
    - Example: a person or program tries to act as the system administrator.
  - **Tampering:** modification of data or code.
    - Example: modification of the source code of the application used to deactivate protections.
  - **Repudiation:** denial of a specific action to have been carried out.
    - Example: “I did not send that message”.
  - **Information Disclosure:** access to a piece of information by an entity that has no credentials to do it.
    - Example: personal information leaked to the public.
  - **Denial of Service:** blocking or degrading a service.
    - Example: block of servers due to a high number of requests.
  - **Elevation of Privilege:** increase of capabilities without the proper authorisation.
    - Example: a user becoming administrator.

# ◆◆◆ Secure Software Development Life Cycle

## STRIDE - Controls

- In order to mitigate the possible impact of a security breach, the develop controls below are established for such threats.

Threats	Security control/service
Spoofing	Authentication
Handling	Integrity Controls
Repudiation	Non-repudiation methods
Information disclosure	Confidentiality mechanisms
Denial of service	Availability
Elevation of privilege	Authorisation



# ◆◆◇ Secure Software Development Life Cycle

## Implementation

- In the traditional SDLC, this stage implies the encryption of the different functions of the software product to develop.
- S-SDLC activities are aimed at helping developers to implement the required functionalities as securely as possible.
- The main contributions of the S-SDLC to this stage are guides and good practices on secure encryption (they will be covered in-depth in the following sections).
- Furthermore, during the implementation stage, an S-SDLC should consider the following activities:
  - Secure configuration of the development environment.
  - Revision of the application's source code.
  - Revision of third parties' elements.



# ◆◆◆ Secure Software Development Life Cycle

## Implementation - Secure Development Environment

- An official configuration should be defined for the development environment that will be used during the implementation of the software product.
- The configuration should specify:
  - Valid operative system or systems (including versions).
  - Tools supported in the development (including versions):
    - IDE.
    - Versions control system.
  - Remote access restrictions.
- The required mechanisms to apply and restrict the configuration of work stations to the accepted level should be included:
  - User accounts restrictions.
  - Pre-installed environments.

# ◆◆◆ Secure Software Development Life Cycle

## Implementation - Secure Development Environment

- XcodeGhost is a clear example of problems that can be caused if such guidelines are not followed.
- A group of cyber-criminals modified and made public a version of the XCode IDE for iOS and Mac OS on a Chinese server.
- The modification injected malicious code in the compiled version of iOS applications.
- Due to the slow downloading speed from US servers, multiple Chinese developers decided to download the version available on their country's servers.
- When using the modified version to develop their applications, many developers created malicious applications unintentionally that were then published on the App Store.
  - <https://developer.apple.com/news/?id=09222015a>
  - [https://www.incibe.es/technologyForecastingSearch/CERT/Bitacora\\_de\\_ciberseguridad/ataque\\_appstore](https://www.incibe.es/technologyForecastingSearch/CERT/Bitacora_de_ciberseguridad/ataque_appstore)

# ◆◆◆ Secure Software Development Life Cycle

## Implementation – Code Review I

- The revision of the application's source code allows users to identify vulnerabilities that are introduced during the implementation stage.
- Automatic tools for static analysis, such as the one reviewed in unit 3, make this task easier, but they are not able to find some vulnerabilities that require a manual revision.
- The review of code is an additional task to the execution of unit and integration tests that are defined within the traditional SDLC. It is not equivalent and should never substitute it.
- The review of the source code may be carried out:
  - Lightly during the implementation process.
  - Formally, once a part of the implementation process has ended.

# ◆◆◆ Secure Software Development Life Cycle

## Implementation – Code Review II

- Regarding lightweight reviews of the source code, it is possible to perform them in the following ways:
  - **Pair-programming techniques:** two people develop code together in the same machine, supervising the written code mutually.
  - **External review:** the author explains the code to other developer that verifies it.
  - **Assisted review:** developers use semi-automatic tools that enable the identification of code problems during the programming.
  - **Commit review:** when there is a commit on the version control system, a tool is launched in order to:
    - Send the element automatically via email to the reviewers.
    - Conduct an analysis of the commit with an analysis tool integrated with the version control.
      - Example: <https://www.pullreview.com/> or <https://codeclimate.com/>

# ◆◆◆ Secure Software Development Life Cycle

## Implementation – Third Parties' Elements

- During the implementation stage, it may be necessary to use third parties' tools and libraries.
- Controls made on our own code should also be implemented on these kind of libraries.
- Specifically, the following activities will be performed:
  - If the source code is available, perform an analysis process as the one described before.
  - Review vulnerabilities or possible security problems related to the library version used:
    - Secure storage.
    - Encrypted communications.
    - Validation of input data.
    - Problems of misconfiguration or data exposure by default.
  - Check the use of functions or elements that have been declared to be deprecated by developers.



# ◆◆◆ Secure Software Development Life Cycle

## Verification

- On the traditional SDLC, this stage includes all the activities aimed at verifying that the software product works as it is described in the requirements.
- S-SDLC tasks during the verification stage allow users to perform security verifications directly on software elements that have been implemented in the previous stage:
  - **Dynamic Analysis:** studied in Unit 3. It enables the verification of the system's security properties and their behaviour by executing it.
  - **Fuzzing:** it is a part of the dynamic analysis. It is checked whether controls implemented in entry points of the system control the possible entries properly .
  - **Revision of the attack surface:** once the code is finished, it is possible to verify that the attack surface identified in previous stages of the S-SDLC corresponds to the real one.

# ◆◆◆ Secure Software Development Life Cycle

## Deployment I

- In this stage, the software product is prepared for its deployment.
- Regarding the S-SDLC, this stage includes activities to cover security aspects of the product beyond its launch date.
- Incident response plan:
  - It allows users to mitigate the scope of security incidents, reduce risks, and costs of an incident.
  - It should clearly identify events to consider when declaring the existence of a security incident.
  - For each incident, actions to carry out should be described in detail.
  - The roles of each response team member and their contact information should be included as well.
  - This task is essential in order to response quickly against any security incident.
  - The response plan is not a static document. It progresses according to modifications of the system or the appearance of new threats that were not taken into consideration.

## Deployment II


- **Final security revision:**
  - Before the launch, it should be verified that all the security tasks planned to perform the S-SDLC have been completed.
  - Furthermore, it is advisable to carry out a revision of each task in order to ensure that no failures have been committed during the performance.
- **Certification:**
  - It allows users to ensure that the product comply with certain security regulations/standards.
- **Storage:**
  - It implies saving a copy of all the elements involved in the software version that is going to be launched.
  - It will be one of the elements to be considered in case of security incident.
- Tasks such as dynamic analysis, fuzzing and other security revisions are still executed during this stage.

# ◆◆◆ Secure Software Development Life Cycle

## Response

- This stage is only activated in response to events that have been declared as generators of an incident in the incident response plan.
- Once activated, the guidelines established by the plan should be followed, including:
  - Staff to notify and order of notification.
  - Data capture for the later analysis of the incident.
  - Execution of tasks for the mitigation of the threat.
  - Execution of tasks for the re-establishment of the service (if needed).



A close-up photograph of a person's hands with red-painted fingernails typing on a black laptop keyboard. A semi-transparent grey rectangular box is overlaid on the center of the image, containing the title text. In the foreground, a pair of black-rimmed glasses and an orange pen with silver accents are resting on a dark surface. The background is softly blurred, showing white flowers with yellow centers.

# S-SDLC on Agile Development Environments



## ◆◆◇ S-SDLC on Agile Development Environments

- Agile methodologies are an alternative process to traditional methodologies that are based on the development through smaller iterations used to include functionality (<http://agilemethodology.org>).
- Tasks and activities normally established by the S-SDLC assume the traditional life cycle (waterfall) during the software development.
- Generally, systems and products for mobile environments are developed via agile methodologies.
- The execution of S-SDLC, as we have studied, requires adaptations to be applied on agile methodologies.
- In such cases, the S-SDLC activities are executed with three different frequencies:
  - By **sprint**: activities that should be executed for each completed release.
  - By **bucket**: activities that should be executed for each set of sprints.
  - By **project**: activities that are executed only once during the whole project.



# ◆◆◇ S-SDLC on Agile Development Environments

- **Activities by sprint:**
  - Threat modelling of the functionality included on the sprint.
  - All the activities related to the S-SDLC implementation stage.
  - Final security review by sprint.
  - Certification and storage.
- **Activities by bucket:**
  - Definition of the security metrics that will be used to assess the bucket.
  - Dynamic analysis, fuzzing and review of the attack surface tasks.
- **Activities by project:**
  - Define the security requirements.
  - Risk analysis.
  - Define the attack surface.
  - Create an incident response plan.

The background is a dark, textured surface with a grid-like pattern. Overlaid on this are several glowing, pixelated text elements in shades of blue and green. The word 'infected' is prominent in the upper right. Below it, the word 'payload' is visible, partially obscured by a semi-transparent blue rectangle. To the left, there's a large, stylized arrow pointing right, and the word 'include' is partially visible. A human hand is shown in the lower left, with fingers extended towards the center of the image, as if interacting with the digital elements.

# Good Practices in Secure Development

# ◆◆◆ Good Practices in Secure Development

## Introduction

- In the previous section, the different stages and activities involved in an S-SDCL were reviewed.
- In this section, a series of general guides and practices for the secure encryption of applications will be studied.
- Such guides are independent from the programming language and the target platform.
- In the case of mobile applications, they should be used where applicable, whether in the back-end or the end-point (mobile application).
- Its objective is to provide the developer a set of practices in order to implement software in a secure way without needing to know concepts related to vulnerabilities security and exploitation in depth.
- Following secure encryption good practices is not enough to ensure that an S-SDLC is being performed; it is only a part of the whole process.

# ◆◆◆ Good Practices in Secure Development

## List

- The following set of practices will be covered within this section:
  - Input validation.
  - Output elements encryption.
  - Password authentication and management.
  - Session management.
  - Access control.
  - Error management.
  - Data protection.
  - Security in communications.
  - System configuration.
  - Security in databases.
  - Memory management.
  - Other general considerations.
- <https://www.youtube.com/watch?v=CIT1VJqJXO0>.



# ◆◆◆ Good Practices in Secure Development

## Input Validation I

- All inputs to the system are considered as malicious: all input is evil:
  - Text fields.
  - URL.
  - Cookies and other HTTP fields.
- Input data validation should always be performed in a reliable system, generally, in the back-end. The mobile device is not considered a reliable element.
- All data validation should focus on a specific part of the application.
- Before carrying out the validation of data, it is advisable to unify the encryption of data for all the verifications to be performed with the same encryption.



# ◆◆◆ Good Practices in Secure Development

## Input Validation II

- It is recommended:
  - To validate data ranges and length.
  - To use white lists to verify that all the elements of an input are valid.
  - Check that all the received data correspond to what is expected:
    - HTTP headers should include only ASCII characters.
    - If an image is expected to be received in a specific format, verify that it is correct.
    - URL text and parameters fields should include the type of data expected in the application.
- In case that there are characters or strings that could be considered as dangerous `< > . / . \ % \ ( ) “ ‘ ’ \` specific additional controls should be added for calls that may include them.
  - In the case of interpretable inputs (such as HTML code), avoid redirects that may render controls ineffective.

# ◆◆◆ Good Practices in Secure Development

## Output Elements Encryption

- Just like input validation, the output data encryption should be performed in a reliable system such as the application's back-end.
- Reinventing the wheel should be avoided. Multiple libraries and methods for output encryption that have been widely tested and accepted by the community are available:
  - The following NSString class method can be used on iOS:  
*stringByAddingPercentEncodingWithAllowedCharacters*.
  - On Android, [URLEncoder](#) or [DatabaseUtils](#) are available.
- Output data should be encrypted according to the way they are going to be used in the application:
  - In case the output is going to be interpreted by a web browser, avoid the creation of interpretable elements in HTML CSS, Javascript, etc.
  - If the output is going to be interpreted by other system, avoid the possible creation or modification of commands to them (SQL, XML, LDAP, etc.).



## Password Authentication and Management I

- All the pages, except for those strictly defined as public, should require the authentication of users.
- The following recommendations should be followed for the implementation of authentication controls:
  - Authentication controls should always be performed on a reliable system (back-end).
  - All the authentication controls should be centralised on a unique module, including libraries able to perform calls to external authentication services.
  - The authentication logic should not be connected to the accessed resource's logic.
  - Authentication requests should always be performed via properly encrypted (SSL) HTTP POST connections.

# ◆◆◆ Good Practices in Secure Development

## Password Authentication and Management II



- The following practices are recommended for the authentication process:
  - The validation of authentication data should only be performed if all the necessary information has been introduced (user and password).
  - In case there is an authentication failure, no details (either visual nor in the source code used) regarding the specific authentication failure (incorrect password, incorrect user, etc.) should be provided.
  - The authentication process should fail in a secure way.
  - Regardless the access method, the password field should not display the elements typed.

# ◆◆◆ Good Practices in Secure Development

## Password Authentication and Management III

- Under no circumstances should passwords be stored in the non-encrypted application.
- All passwords should be stored summarised by using a secure cryptographic function, using a salt in order to complicate brute-force attacks via rainbow tables.
- The application should obligate users to use passwords with a minimum level of complexity:
  - Minimum length of 8 characters, but longer passwords are recommended.
  - Alphanumeric characters, punctuation marks and numbers.
- If the application creates passwords by default, oblige the user to change it during the first access.
- If various failed access attempts occur, deactivate access for a period of time, long enough to avoid brute-force attacks, but not as long as to cause a denial of service.

# ◆◆◆ Good Practices in Secure Development

## Password Authentication and Management IV

- Regarding the reset of passwords:
  - When possible, the use of security questions should be avoided. In case they are necessary, questions with a predictable or common answer should be avoided:
    - Incorrect example: What is the name of your first pet?
    - Correct example: Name of the street your mother lived in.
  - The email to which the reset request is sent should be verified to be registered in the system.
- In case any critical operation is going to be performed in the system, such as the change of password itself, the user should be asked to authenticate again.
- If possible, implement a double authentication factor via:
  - Password + mobile application that creates passwords that can only be used once ([Google authenticator](#)).
  - Password + biometric element.

# ◆◆◆ Good Practices in Secure Development

## Session Management

- If the session control included in the framework server on which the application is developed provides sufficient guarantees, it is advisable to use it.
  - [iOS](#)
  - [Android](#)
  - [Java EE](#)
  - [.Net](#)
  - [Django](#)
  - [Ruby on Rails](#)
- Identifiers should be created by a reliable system (generally, the back-end), with libraries that ensure that they are random enough.
- Every re-authentication should create a new session identifier and remove the old file.
- The user should be able to logout easily.
- Sessions should expire after a minimum period of inactivity.
- The exposition of information regarding sessions or cookies to third parties (logs record, use of GET parameters, etc.) should be avoided.
- According to budget constraints, a system that enables the control and logout of active sessions should be provided to the user.

# ◆◆◆ Good Practices in Secure Development

## Access Control

- Access control decisions should be taken according to information coming from reliable systems.
- As in the case of input, output, and authentication validation, it is advisable that the access control system is centralised and separated from the rest of logic, in a unique element of the system.
- The access control should be performed for all the requests, including those carried out via technologies such as AJAX.
- Unauthorised users should not be allowed to access elements such as:
  - Application and services data.
  - URLs only accessible by authorised users, including images.



# ◆◆◆ Good Practices in Secure Development

## Error Management

- When an error occurs, it should be avoided to disclose sensitive information such as system details, sessions identifiers or accounts information.
- The application should manage all the errors and never depend on system errors by default.
- When an error occurs, the policy used by default regarding the task performed should be the denial.
- Logs should register relevant events of the system:
  - Failures on input validation.
  - Failed authentication attempts.
  - Connection attempts with expired sessions.
  - Changes in the configuration of critical elements.
  - Exceptions in the system and other errors occurred during the execution.



# ◆◆◆ Good Practices in Secure Development

## Data Protection

- Passwords, authentication tokens and other sensitive information should be stored encrypted.
- The storage of credentials within configuration files or the application's source code itself should be strictly avoided. If open repositories such as GitHub are used, access credentials to services could be published.
- Configure the applications server for files of the back-end application's source code not to be downloaded.
- Remove documentation and configuration files that are installed by default.



# ◆◆◆ Good Practices in Secure Development

## Security in Communications

- Since most connections include authentication tokens, sessions or sensitive information in order to access services of the application, connections between clients and the server should be encrypted.
- Applications should verify the validity of the certificate provided and even use certificate pinning techniques to mitigate attacks to the PKI system.
- Resources accessible via secure connections should not be available through insecure connections (downgrade to non-encrypted connections).
- The implementation of such technologies on mobile applications will be reviewed more in depth in the following section.



# ◆◆◆ Good Practices in Secure Development

## System Configuration

- Ensure that the versions of the third parties' elements required for the execution of the application are the ones approved during the design.
- It is also necessary to check that no vulnerabilities that may affect the approved versions have been registered. In such case, they should be reviewed and chosen again.
- The system in process of production should not include the source code and resources files that have been used to perform tests and verifications during the development process.
- In case part of the application is provided by a web server, use the “robots.txt” corresponding file to avoid indexing. It is important to keep this aspect in mind since it may cause the providing of extra information to the attacker.
- It is advisable to use a system for versions control during the development of the software.

# ◆◆◆ Good Practices in Secure Development

## Security in Databases

- The access to the database should be made via parametrised queries, regardless the database type.
- Parameters used and results obtained in queries should go through their corresponding encryption and validation processes (escaping and filtering).
- Applications and systems should use as least privileges as possible to access databases' tables.
- Roles with different access levels should access by using different users to ensure the separation of privileges.
- The connection to the database should be maintained only as long as it is strictly necessary to complete the requests needed.
- As in the case of the rest of subsystems (applications server, etc.), all the unnecessary files and installation configuration created by default should be removed.

# ◆◆◆ Good Practices in Secure Development

## Memory Management I

- Some programming languages are responsible for the management of memory through their execution environments (Java, Javascript, Python, Swift, etc.) in an automatic way. However, others such as C (that can be used for the development of mobile applications on Android or iOS) have a manual memory management system.
- In such cases, it is essential to carry out a good memory management. Most critical vulnerabilities are caused due to memory management issues (buffer overflow).
- The most critical aspects regarding memory management are related to:
  - Buffer copies between memory addresses.
  - Space reserved in memory for indefinite-length variables.
  - Freed up memory that had previously been freed up.

# ◆◆◆ Good Practices in Secure Development

## Memory Management II

- It is advisable to follow a series of guidelines that limit the arising of memory management issues as much as possible:
  - When using functions that accept the amount of bytes to copy (such as `strncpy`), users should take into consideration the fact that the target buffer may not end in zero (not all the source bytes are copied).
  - When creating copies between buffers, users should verify that sizes are correct and there is not possibility of writing when the space reserved for each buffer is exceeded (end-of-copy condition well defined).
  - Maximum sizes should be defined for all the buffers used.
  - When a variable that we reserved memory space for is not needed anymore, the resource should be released or closed. Users should not rely on the garbage collector function.
  - If possible, users should avoid the use of dangerous functions such as `strcat`, `strcpy`, etc.

# ◆◆◆ Good Practices in Secure Development

## Other General Considerations I

- Regardless the element to program, if there already is a tested and verified code that performs such operation, using it would always be the best option.
- When a task related to the operative system has to be performed, it should be executed through the API provided by it. Under no circumstances should commands be sent to the operative system through the console directly.
- Whenever a code that has not been included in the initial deployment of the application (dynamic execution) is going to be executed, its integrity should be verified.
- Synchronisation mechanisms existing should be used in the operative system in order to avoid the appearance of race conditions.
- All the variables and sources of data should be initialized before first use.



# ◆◆◆ Good Practices in Secure Development

## Other General Considerations II

- The numerical representation of the programming language should be taken into consideration in order to avoid errors when carrying out calculations.
  - Specifically, the following aspects should be taken into account: accuracy of operations, types of signed/unsigned data, conversions, castings, and the way that the programming language handles numbers over and under the limits of representation.
- If the application will implement automatic updating mechanisms, it should be verified that the code received during such updating comes from a reliable source.
  - Code signing mechanisms can be used to this aim, as the ones used in mobile applications shops/markets. Once the code has been downloaded and before the update, its signature should be verified.



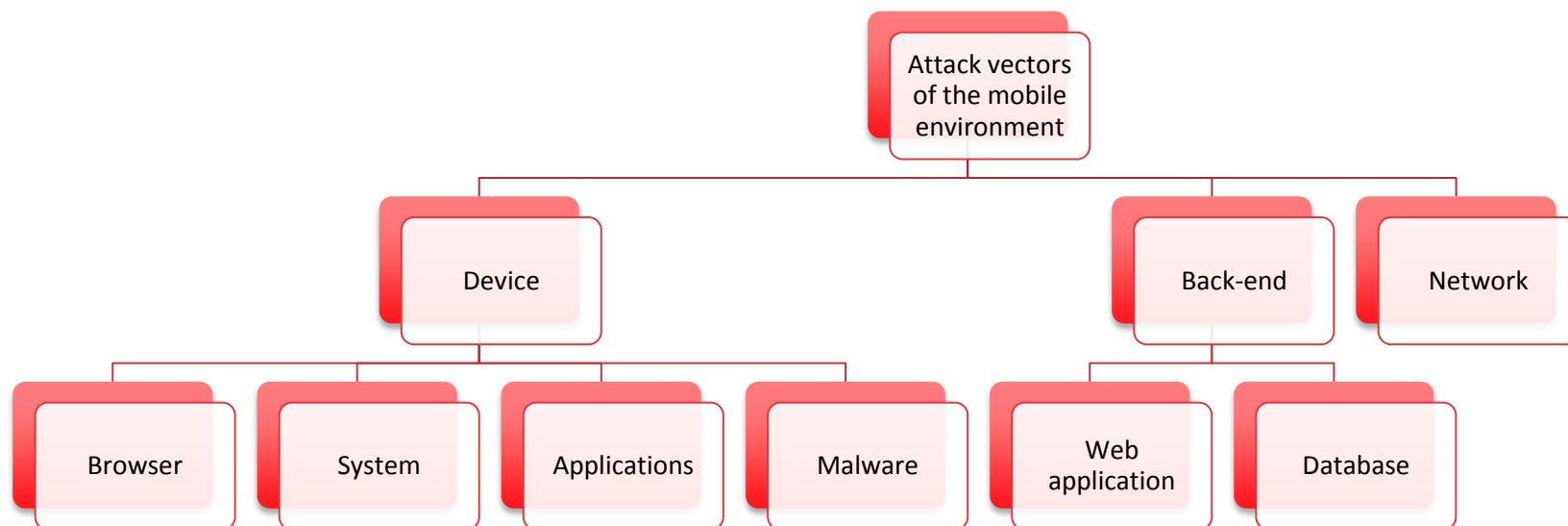


# Good Practices in Mobile Development

# ◆◆◆ Good Practices in Mobile Development

## Introduction

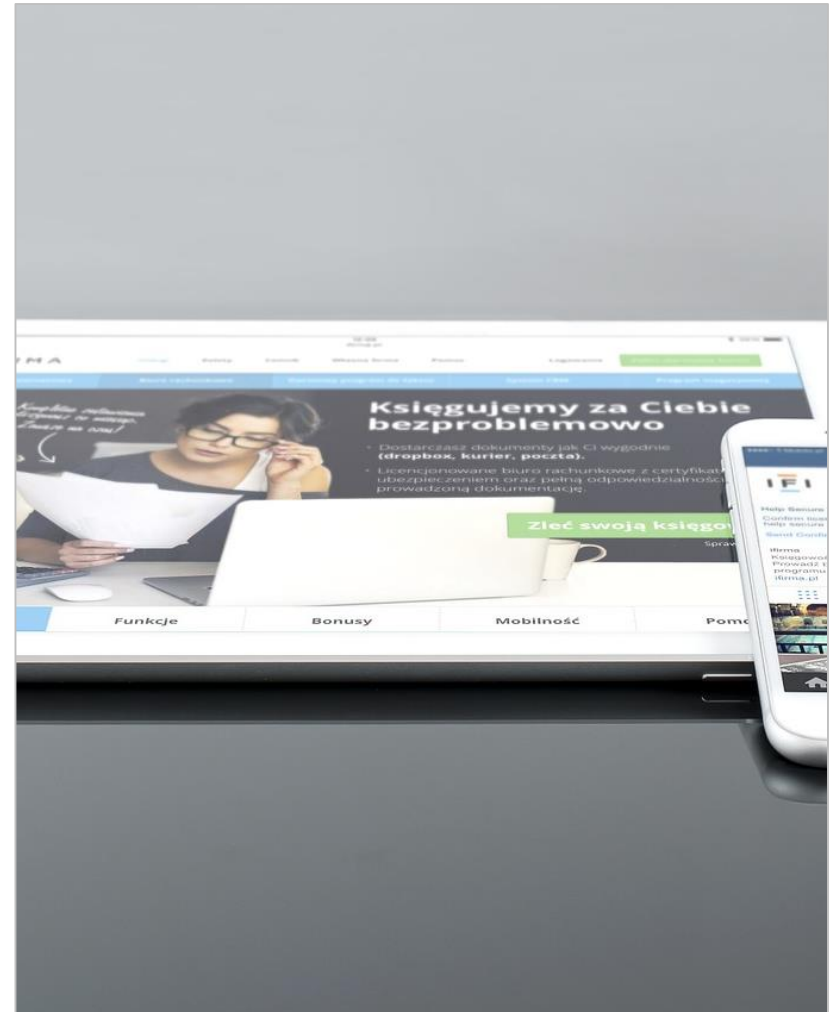
- Mobile applications share characteristics with web applications (many users, fast development, continuous network connectivity), as well as with desktop systems (shared data storage, malware, existence of vulnerable applications such as the browser) in a mobility environment.
- The attack surface on a mobile device is a combination of elements that affect both types of applications.



# ◆◆◆ Good Practices in Mobile Development

## Specific Aspects

- During the rest of the section, the security aspects that should be taken into consideration when developing of applications, in order to mitigate threats created by attack vectors of the mobile environment will be studied:
  - Protection of the application's code.
  - Sensitive data management.
  - Logs and sensitive data leakage.
  - Sensitive data management.
  - HTML components in mobile applications.
  - Communication between applications.
  - Authentication in mobile applications.





# Application Protection



## Code Obfuscation

- As it was verified in unit 3, reverse engineering techniques may provide quite useful and valuable information on the functioning of applications.
- Increasing the complexity of the code by using obfuscation techniques will make it difficult for the attacker to identify vulnerabilities and failures that have been unnoticed during the revision processes.
  - ProGuard
  - DexProtector
- In order to make reverse engineering tasks difficult, the following techniques may be used:
  - Restriction of the use of debuggers.
  - Traces detection.
  - Debugger optimisation.
  - Destruction of information on binary symbols.

# ◆◆◆ Application Protection

## Restriction of the use of debuggers



- Through the operative system, applications may restrict the use of debuggers in order to inspect their execution.
- Even though such techniques can be avoided using repackaging techniques, it requires an extra effort from the attacker.
- On Android, it can be specified with the `android:debuggable="false"` attribute in the application tag within the manifest.
- On iOS, the following call may be introduced at the beginning in the execution of the application for it to close in case a debugger is attempted to be added:
  - `ptrace(PT_DENY_ATTACH, 0, 0, 0);`



# ◆◆◆ Application Protection

## Detection of Traces

- Since repackaging techniques can be used to debug the application, it is advisable to add controls in order to verify if it is being debugged.
- If the application is detected to be connected to a debugger, there are different possibilities:
  - Notify the back-end.
  - Remove sensitive data from the application.
- It can be identified on Android with the following line:

```
boolean depuracion= ( 0 != ( getApplicationInfo().flags &
ApplicationInfo.FLAG_DEBUGGABLE ) );
```
- On iOS, the following code provided by Apple can be implemented:  
<https://developer.apple.com/library/ios/qa/qa1361/index.html>

## Debugger Optimisation

- Debugger optimisations modify the code for it to be faster when executing on the processor; however, they also make its reading and understanding more difficult.
- On Android, two options are possible:
  - A part of the application can be programmed in C in order to use it as native libraries.
  - ProGuard removes the code that is not used in the application and modifies the name of methods, variables, classes and packets in order to complicate their reading. ProGuard documentation is available at:  
<http://developer.android.com/tools/help/proguard.html>
- Some tools similar to ProGuard can be used on iOS:
  - iOS Class Guard: <https://github.com/Polidea/ios-class-guard>
  - LLVM obfuscator: <https://github.com/obfuscator-llvm/obfuscator>

# ◆◆◆ Protection of the Application

## Destruction of Binary Symbols

- The destruction of binary symbols (or binary stripping) eliminates the binary symbols table.
- This table is a structure of data created by the compiler in order to identify the names of variables and methods used in the binary. Its elimination complicates the read and understanding of the code during its static analysis and debugging.
- On Android, it is possible to:
  - Use an executable file compressor such as UPX (<http://upx.sourceforge.net>).
  - Use console utilities such as `strip`.
- On iOS, it is possible to configure the project on the Build Settings tab and section **Deployment Postprocessing**

► <b>Deployment Postprocessing</b>	Yes ⇅
Strip Debug Symbols During Copy	Yes ⇅
Strip Linked Product	Yes ⇅
Strip Style	All Symbols ⇅

# ◆◆◆ Application Protection

## Application Integrity

- Besides reverse engineering attacks, the repackaging can be used for malicious code injection.
- The application is able to verify the integrity of its components via summaries or digital signatures of its components.
- If a modification of the application is detected, the user can notify the back-end or remove the application's sensitive data.
- On Android, the `PackageManager` allows users to access the information of the current signature of the application.
  - ```
PackageInfo packageInfo =  
    context.getPackageManager().getPackageInfo(context.getPackageName(),  
    PackageManager.GET_SIGNATURES);
```
  - `packageInfo.signatures`
- On iOS, it is possible to verify the validity of the purchase receipt of the application provided by the system.
  - <https://developer.apple.com/library/mac/releasenotes/General/ValidateAppStoreReceipt/Introduction.html>
- Such verifications can be removed via repackaging techniques.

# ◆◆◆ Application Protection

## Jailbreak and rooting detection

- Devices jailbreak or rooting remove many of the security measures of the system for apps protection such as sandboxing.
- The detection can be made by executing some of the tasks that can only be executed under these conditions or by locating files installed during the project.
- On Android, users may try to locate the supersu application or busybox tools:
  - The supersu application has a packet name `eu.chainfire.supersu`
  - Executing a command via `Process su = Runtime.getRuntime().exec("command");`
- On iOS, users can detect jailbroken devices.

```
+ (BOOL)isJailbroken{
    if ([[NSFileManager defaultManager] fileExistsAtPath:@"Applications/Cydia.app"]){
        return YES;
    }else if ([[NSFileManager defaultManager] fileExistsAtPath:@"Library/MobileSubstrate/MobileSubstrate.dylib"]){
        return YES;
    }else if ([[NSFileManager defaultManager] fileExistsAtPath:@"bin/bash"]){
        return YES;
    }else if ([[NSFileManager defaultManager] fileExistsAtPath:@"usr/sbin/sshd"]){
        return YES;
    }else if ([[NSFileManager defaultManager] fileExistsAtPath:@"etc/apt"]){
        return YES;
    }
    return NO;
}
```
- Such types of techniques can be avoided by modifying the location or files names, or through the repackaging of the application.



A composite image featuring a tablet, a spiral notebook, a USB drive, and glasses. The tablet, which has a Windows logo on its bezel, displays a lock screen with a vibrant, multi-colored abstract background of curved lines. The time '15:51' is shown in large white digits, with 'Donnerstag, 15. Jänner' below it. The tablet is placed on a dark blue, textured spiral-bound notebook. To the left of the notebook is a silver Intenso USB drive. A pair of red-rimmed glasses with gold-colored temples is resting on the notebook in the foreground. The entire scene is set against a light-colored wooden surface.

# Sensitive Data Management

# ◆◆◆ Sensitive Data Management

## Sensitive Data of an Application

- Mobile applications handle sensitive data of different types during their execution.
- The different possible states of a device's data are as follow:
  - Data at rest: data located in the persistent storage of the device such as memory cards or files internal system.
  - Data in transit: data sent and received from the back-end and other mobile devices.
  - Data in memory: data on which operations are being executed and, therefore, are stored in the device's memory.
- In order to ensure the proper management of data during all their life cycle, users should follow a series of guidelines to handle them in each of the possible states.



# ◆◆◆ Sensitive Data Management

## Secure Data Storage I

- Data considered as sensitive require a specific protection when they are at rest on the device.
- The use of encrypted file systems limits access to the device if it is turned off for an external attacker; however, it does not prevent other applications or processes of the device from performing data readings through the file system (if the corresponding permissions allow it).
- In order to prevent access by third parties, users should avoid the use of external storage systems (SD cards, etc.) to store sensitive information.
- On Android, the use of the attribute `android:installLocation` (that allows users to use external storage to install the application) should be avoided.

# ◆◆◆ Sensitive Data Management

## Secure Data Storage - Android

- Furthermore, for data stored in the internal memory, it is advisable to implement an additional encryption layer.
- Android does not include specific libraries for the encryption of files at rest (except for certificates and keys). There are two options to implement such type of encryption:
  - Create from scratch using the standard encryption libraries existing in the Java API.
  - Use a library for the secure storage of data such as sqlcipher (<http://sqlcipher.net>).
- Two APIs can be used for the storage of keys:
  - KeyChain API used to store credentials that will be used through the whole the system (e.g.: root certificate of a CA).
  - Keystore used to store keys that will be used by the application. From Android 6.0, it is possible to store symmetric encryption keys.

# ◆◆◆ Sensitive Data Management

## Secure Data Storage - KeyStore

- Keys created via Keystore include two security measures:
  - No application has direct access to keys. The KeyStore API is responsible for all the operations.
  - In the case of devices that have a secure execution environment or a “secure element” (SE), the key is stored inside the SE itself. In this case, applications request the SE to perform the cryptographic operations. Applications will only be able to use the SE for such cryptographic options compatible with the SE (encryption algorithms, signature, verification, etc.).
- Furthermore, each key stored in the KeyStore may be configured in a way that can only be used if the user has unlocked the phone via pattern, PIN, password or biometric element (from Android 6.0).



# ◆◆◆ Sensitive Data Management

## Secure Data Storage - iOS

- On iOS, data can be encrypted with an additional encryption layer by using the Data Protection API.
- Every time a file is created via `NSFileManager` on iOS, four different protection classes can be specified:
  - `NSFileProtectionComplete`: the file will be encrypted with a key derived from the lock code. The key is only accessible with the device unlocked and is discarded 10 seconds after the lock.
  - `NSFileProtectionCompleteUnlessOpen`: the same protection as in the previous case, but, in case the file is open when the device is locked, the key is not discarded until the file is closed.
  - `NSFileProtectionCompleteUntilFirstUserAuthentication`: the encryption key is obtained after the first authentication and is not forgotten until the device is turned off.
  - `NSFileProtectionNone`: it does not provide any additional protection.

# ◆◆◆ Sensitive Data Management

## Secure Data Storage - iOS

- Furthermore, iOS provides the Keychain, an encryption container (that also has a key derived from the lock code) for credential storage by applications.
- Elements added to the Keychain include a series of attributes that specify its type (user, password, certificate, etc.), type of authentication they can be used for and conditions to access them.
- In addition to conditions described for files, some new ones are added:
  - `kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly`
  - `kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly`
  - `kSecAttrAccessibleAlwaysThisDeviceOnly`
  - `kSecAttrAccessibleWhenUnlockedThisDeviceOnly`
- Aimed at avoiding the replication of elements introduced into other devices that share the same iCloud account (and have the Keychain activated on the cloud).

# ◆◆◆ Sensitive Data Management

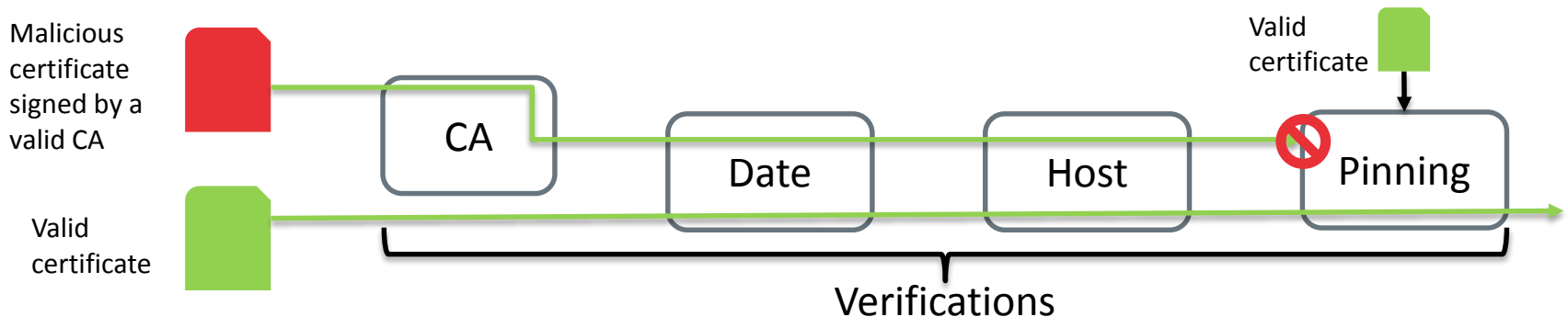
## Security in Data Transport – SSL

- All the connections that include any kind of sensitive information should be performed via SSL connections that have been completely validated.
- To this end, it is necessary to verify a series of properties of the certificate during the establishment of the connection.
  - The certificate should have been signed by a valid certificate authority. Authorities that are included in the list of valid authorities of the device can be considered as valid authorities; furthermore, the application itself may establish a valid authority for its connections by including the corresponding CA certificate.
  - The certificate should not have expired or be in a black list of certificates.
    - Android has a black list of insecure certificates that can be updated remotely.
    - On iOS this list is maintained up to date with the operative system's updates.
  - The name of the certificate's server should be the same as the requested one.

# ◆◆◆ Sensitive Data Management

## Security in Data Transport – Pinning

- The pinning certificate is a technique that takes advantage of a mobile application generally connecting to a limited number of servers.
- Apart from the previous verifications, the application will check that the certificate provided by the server corresponds to a certificate already known by the application.
- The certificate authorities committed do not affect the application's security. If pinning is implemented, the signature of a reliable CA is not required.
- The pinning can be deactivated via repackaging or if the device has been rooted or jailbroken.





# ◆◆◆ Sensitive Data Management

## Security in Data Transport – Pinning

- Android allows the implementation of Certificate Pinning by defining a specific `TrustManager`.
- To this end, it is necessary to include the CA certificate among the resources of the application.

```
// A KeyStore that includes the certificate of our CA is created
Certificate ca = //the certificate of a file is read
keyStoreType = KeyStore.getDefaultType();
KeyStore keyStore = KeyStore.getInstance(keyStoreType);
keyStore.load(null, null);
keyStore.setCertificateEntry("ca", ca);
// A TrustManager that relies only on our CA is created
String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
tmf.init(keyStore);
// A context that includes our TrustManager is created (and it will be used to create the
HttpsURLConnection).
Context context = SSLContext.getInstance("TLS");
context.init(null, tmf.getTrustManagers(), null
```

# ◆◆◆ Sensitive Data Management

## Security in Data Transport – Pinning on iOS

- On iOS, the pinning is performed via the `NSURLConnection` delegate that includes the method: `willSendRequestForAuthenticationChallenge`.

```
...
SecTrustRef serverTrust = challenge.protectionSpace.serverTrust;
SecCertificateRef certificate = SecTrustGetCertificateAtIndex(serverTrust, 0);
NSData *remoteCertificateData = CFBridgingRelease(SecCertificateCopyData(certificate));
NSData *localCertData = [NSData dataWithContentsOfFile:[NSBundle mainBundle]
pathForResource:@"file_name" ofType:@"cer"];
if ([remoteCertificateData isEqualToData:localCertData]) {
    NSURLConnection *credential = [NSURLCredential credentialForTrust:serverTrust];
    [[challenge sender] useCredential:credential forAuthenticationChallenge:challenge];
}else {
    // ERROR MESSAGE
    [[challenge sender] cancelAuthenticationChallenge:challenge];
}
```

- It may also be implemented through the [TrustKit](#) library.

# ◆◆◆ Sensitive Data Management

## Variables in Memory



- When the content of a variable that includes sensitive information (keys, authentication tokens, cookies, etc.) is not necessary anymore, it should be removed from the memory.
- It is advisable that such kind of values are stored in bytes arrays instead of in objects such as strings.
  - The reallocation of an object (as, for example, a string) generally reserves a new space in memory for the new reference, but it does not remove the old one until the garbage collector passes.
  - The use of a byte array allows users to overwrite the content of the variable at any time.

# ◆◆◆ Sensitive Data Management

## Caches

- Whenever it is possible, avoid the storage of sensitive data in caches, including:
  - Cookies.
  - Files.
  - SQLite databases.
  - Websites cache.
  - On iOS, it is possible to limit websites cache by returning `null` in the `willCacheResponse` method of the `NSURLConnection` delegate.

```
- (NSCachedURLResponse *)connection:(NSURLConnection *)connection  
willCacheResponse:(NSCachedURLResponse *)cachedResponse {  
    return nil;  
}
```

- On Android, it is possible to deactivate HTTP connections cache by using the `setUseCaches` method of `URLConnection` objects.

```
URLConnection connection = miURL.openConnection();  
connection.setUseCaches(false);
```

A photograph of a workspace in a modern office. In the foreground, a laptop is open, displaying lines of code on its screen. To the left of the laptop is a white coffee cup on a saucer. In front of the laptop are two spiral-bound notebooks. The notebook on the left contains a hand-drawn diagram with arrows and text. The notebook on the right is open to a page with handwritten notes and a pen lies on it. The background shows a blurred office environment with large windows and modern furniture.

# Logs and Sensitive Data Leakage



# ◆◆◆ Logs and Sensitive Data Leakage

## Removal of Logs

- Every time it is possible, it is advisable to remove the maximum amount of logs created by the application in the device.
- On Android, it is possible to configure ProGuard in order to remove logs by adding proguard.cfg to its configuration file.

```
-assumenosideeffects class android.util.Log {  
    > public static *** d(...);  
    > public static *** v(...);  
    > public static *** i(...);  
    > public static *** e(...);  
}
```

- On iOS it is possible to use a macro that removes the log sentence in development versions in production process.

```
#define NSLog(s,...)
```

# ◆◆◆ Logs and Sensitive Data Leakage

## Keyboard Cache

- Users should avoid the operative system to store sensitive data typed by the user in the keyboard cache.
- Specific password fields should be used for passwords. Information typed on such fields is not stored in the device's cache.
- For the rest of fields that may store sensitive data, the option to store data typed in the cache should be deactivated.
- On Android, the only consistent way of performing such operation for it to work on all devices, is to define the field as a password field with visible text. It can be made by using the text field attribute `android:inputType="textVisiblePassword"`.
- On iOS, it may be performed by configuring the `UITextField autocorrectionType` property with the following value: `UITextAutocorrectionTypeNo`.



# ◆◆◆ Logs and Sensitive Data Leakage

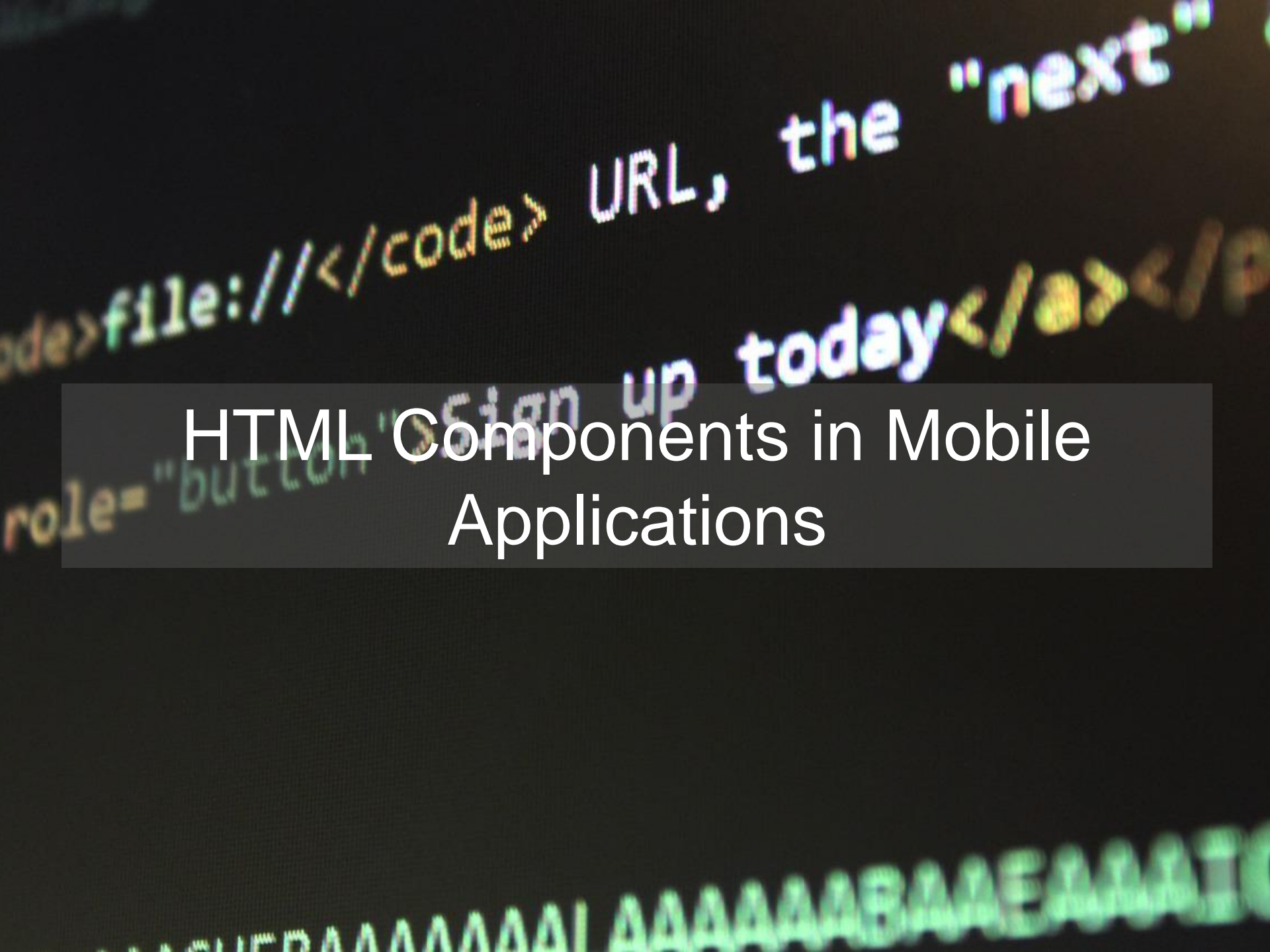
## Clipboard

- The information stored in the system's clipboard may be accessed by other applications without needing any special permission.
- On Android, it is necessary to create a subclass of the EditText class and re-implement `isSuggestionsEnabled` and `canPaste` methods for them to return false.
- There are two possibilities on iOS:

- By

```
-(BOOL)canPerformAction:(SEL)action withSender:(id)sender {  
    if (action == @selector(copy:) || action == @selector(cut:)) {  
        return NO;  
    }  
    [super canPerformAction:action withSender:sender];  
}
```

- When `copy` and `cut` functions are executed, the `pasteboardWithUniqueName` method is called in order to obtain the specific clipboard of the application.



# HTML Components in Mobile Applications

# ◆◆◆ HTML Components in Mobile Applications

## Introduction

- All mobile operative systems allow applications to display HTML content through web views.
- The use of such views implies certain security risks that should be taken into account when using them.
- In order to reduce the attack surface created by such views and, regardless the platform used, it is recommended:
  - Not to load remote content via unencrypted connections (without SSL).
  - To make sure that an SSL encryption secure connection is used and the SSL certificate provided by the server is completely validated.
  - To forbid the access to the system to device files from the web view.
  - To deactivate Javascript and any other plug-in when possible.
  - To verify that the web view only loads URLs of the required domains.
  - Not to expose native methods through Javascript.
  - Not to allow the load of URL via communication mechanisms with other applications.

# ◆◆◆ HTML Components in Mobile Applications

## How to Secure Android Components

- Deactivate Javascript.

```
WebView webview = new WebView(this);  
webview.getSettings().setJavaScriptEnabled(false);
```

- Deactivate access to the file system.

```
WebView webview = new WebView(this);  
webview.getSettings().setAllowFileAccess(false);
```

- Verify URLs loaded on the web view (WebView extension).

```
private class MiWebViewClient extends WebViewClient {  
    @Override  
    public boolean shouldOverrideUrlLoading(WebView view, String url) {  
        //VERIFICATIONS OF THE INITIAL URL  
    }  
    @Override  
    public WebResourceResponse shouldInterceptRequest(final WebView view, String url){  
        //VERIFICATION OF ALL THE REQUESTS  
    }  
}
```

# ◆◆◆ HTML Components in Mobile Applications

## How to Secure Android Components

- In order to avoid the load of URL from external elements of the application, it is necessary to remove from the manifest all the exported attributes of activities defined on the application that have a web view as main view.
- Data of the web view cache should be removed once it has stopped being used (`onPause()` method of the activity).

```
@Override
protected void onPause() {
    ...
    webview.clearCache();
    ...
    super.onPause();
}
```

- Finally, users should avoid exposing the visible native code by using the `addJavaScriptInterface()` method.

# ◆◆◆ HTML Components in Mobile Applications

## iOS - UIWebView

- From iOS 9, there are three types of web views.
- UIWebView whose use is not recommended from iOS8, but may be used on applications.
- It uses a non-optimised rendering motor, therefore, web pages load more slowly.
- It only allows configuration through the delegate methods and, among them, the only relevant one is `webViewShouldStartLoadWithRequest`, which enables

```
- (BOOL)webView:(UIWebView*)webView shouldStartLoadWithRequest:(NSURLRequest*)request
navigationType:(UIWebViewNavigationType)navigationType {
    NSURL *url = request.URL;
    //VERIFICATIONS

    return ...;
}
```

- Web view of such kind do not allow the deactivation of the Javascript motor.



# ◆◆◇ HTML Components in Mobile Applications

## iOS - WKWebView

- WKWebView is the web view used from iOS 8 by default.
- During its initialisation, a series of parameters can be defined via an object such as WKWebViewConfiguration.
- For example, the following code can be used to deactivate Javascript on the

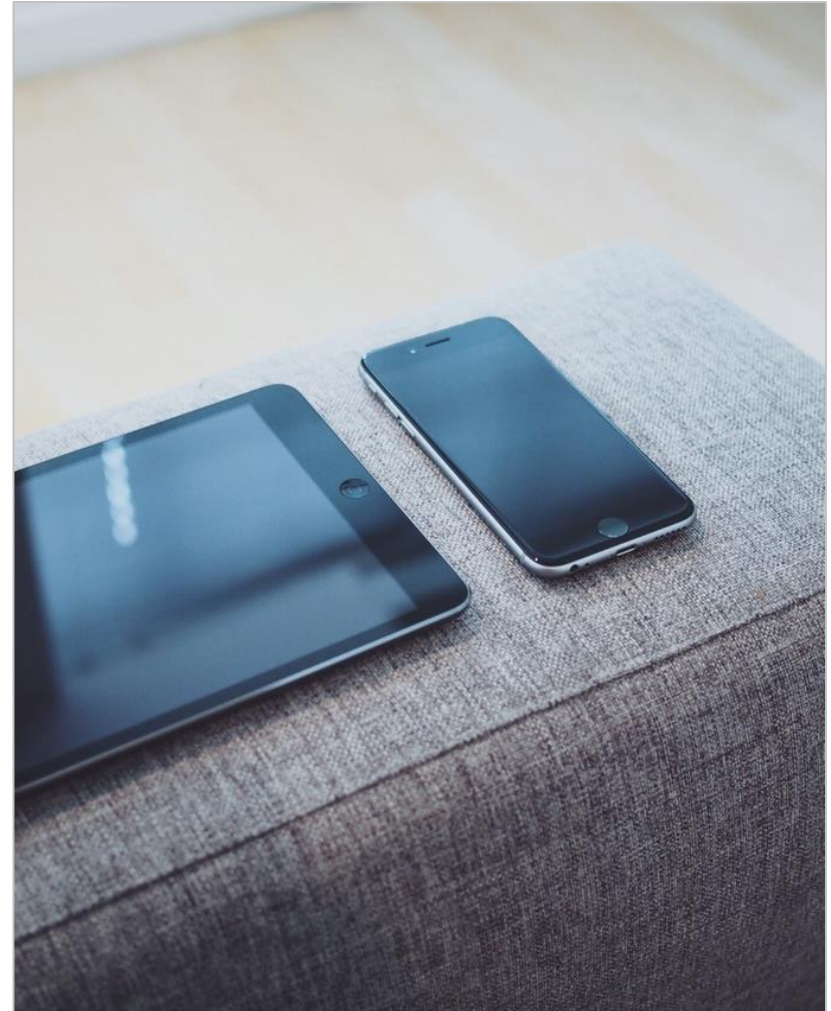
```
WKWebViewConfiguration *conf = [[WKWebViewConfiguration alloc] init];  
conf.preferences.javaScriptEnabled = NO  
WKWebView *webView = [[WKWebView alloc] initWithFrame:self.view.frame  
configuration:conf];
```

- It allows users to perform the certificate pinning through the delegate as it was explained in the Sensitive Data Protection section.
  - There is an error on the delegate of iOS 8, therefore, pinning should be carried out reviewing the certificates included in the `certificateChain` property.

# ◆◆◆ HTML Components in Mobile Applications

## iOS - SafariViewController

- SafariViewController provides a specific controller that enables web navigation from inside the application.
- Such controller is executed on a different process from the application that invokes it.
- The user may access all Safari's functionalities, including password auto-complete function, button to execute actions and address bar in read mode.
- The application cannot access places navigated from the view or data introduced into it by the user.
- If the application wants a greater control on the content displayed, it should use the `WKWebView` view.







# Communication Between Applications

# ◆◆◆ Communication Between Applications

## Communication Between Applications on Android

- The communication between applications on Android is an additional attack vector that should be controlled.
- First of all, all the application elements that are not specifically used to communicate with other applications should be marked in the manifest with the following property: `android:exported=false`.

```
<activity android:name=".MyActivity" android:label="Etiqueta"
android:exported=false >
  <intent -filter>
    <action android:name="accion.intent"></action>
  </intent>
</activity>
<service android:enabled="true" android:name=".MyService"
android:exported=false ></service>
<receiver android:enabled="true" android:exported=false
  android:label="My Broadcast Receiver"
  android:name=".MyBroadcastReceiver"
</receiver>
```

# ◆◆◆ Communication Between Applications

## Intents

- Users should avoid sending sensitive information via `BroadcastIntents` since malicious activities could define the same filter with a bigger priority to capture sensitive information.
- All the information received from an intent should be validated as any other input.
- If the activity is not public, the user should avoid defining `IntentFilters`. Therefore, the only way to access them is through the name of the component.

```
//INTENT WITH SENSITIVE INFORMATION: IF IT IS CAPTURED, DATA ARE COMPROMISED
public static Intent crearIntent(Context context, Usuario user) {
    Intent i = new Intent(context, DetallesUsuario.class);
    i.putExtra(EXTRA_USUARIO, user);
    return i;
}
//INTENT THAT ONLY SENDS THE USER'S ID
public static Intent crearIntent(Context context, String userId) {
    Intent i = new Intent(context, DetallesUsuario.class);
    i.putExtra(EXTRA_USER_ID, userId);
    return i;
}
```

# ◆◆◆ Communication Between Applications

## Content Providers

- ContentProviders provide the application data to third parties' applications.
- ContentProviders may require permissions to read (`readPermission`), write (`writePermission`), or both actions (`permission`) to other applications through the provider in the application's database.

```
<provider android:name="MiProvider"
    android:authorities="com.miapp.provider.MisDatos"
    android:readPermission="permiso.de.lectura"
    android:writePermission="permise.de.escritura"
    android:permission="permiso.para.ambos">
</provider>
```

- Permissions defined within the provider should have been previously declared in the application via elements such as `<permission>`.
- A provider should validate all the requests received to avoid SQL code injection to access to unauthorised files.
- It is performed via the `android:grantUriPermissions="true"` attribute. An application may also provide occasional access to ContentProvider elements to applications that request it, even if they have not defined any specific permission.



# ◆◆◆ Communication Between Applications

## Services

- Services may also expose functionalities or sensitive information to third parties' applications that should be properly protected.
- A service may validate permissions of an application, method by method, in order to access a specific functionality through the method checkPermission() of the PackageManager.
- It is also possible to define the permissions required to communicate with the service through the application's manifest.

```
<permission android:name="com.mipermiso" android:label="mi_permiso"
android:protectionLevel="dangerous"></permission>`
<service android:name="com.MiServicio" android:permission="com.mipermiso">
    <intent-filter>
        <action android:name="com.MI_ACCION"/>
    </intent-filter>
</service>
```

# Authentication in Mobile Applications



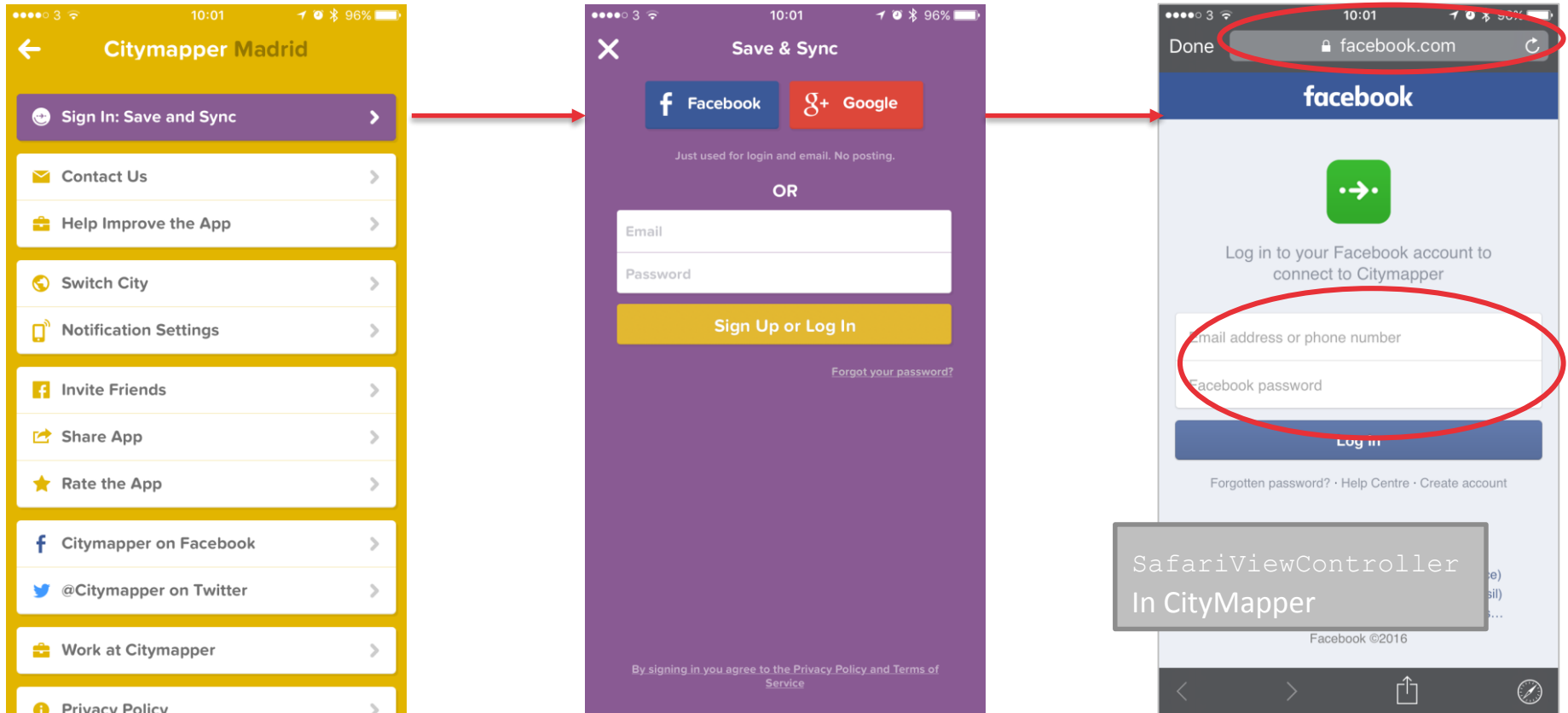
# ◆◆◆ Authentication in Mobile Applications

## Introduction to OAuth

- OAuth is a standard for the access authorisation to resources/services through the web.
- From the point of view of an end user, OAuth provides the following advantages:
  - Use of only one account to access multiple services.
  - Less passwords to remember.
  - It is not necessary to share credentials (generally, user and password) with an unreliable external service.
  - It is possible to revoke provided authorisations easily.
- From the point of view of the developer, it provides the following advantages:
  - The handling and the amount of sensitive information to store is simplified.
  - Password management or renovation systems are not required.
  - Its implementation is performed via widely tested libraries.
  - There are multiple identity and other services providers that already use OAuth.

# ◆◆◆ Authentication in Mobile Applications

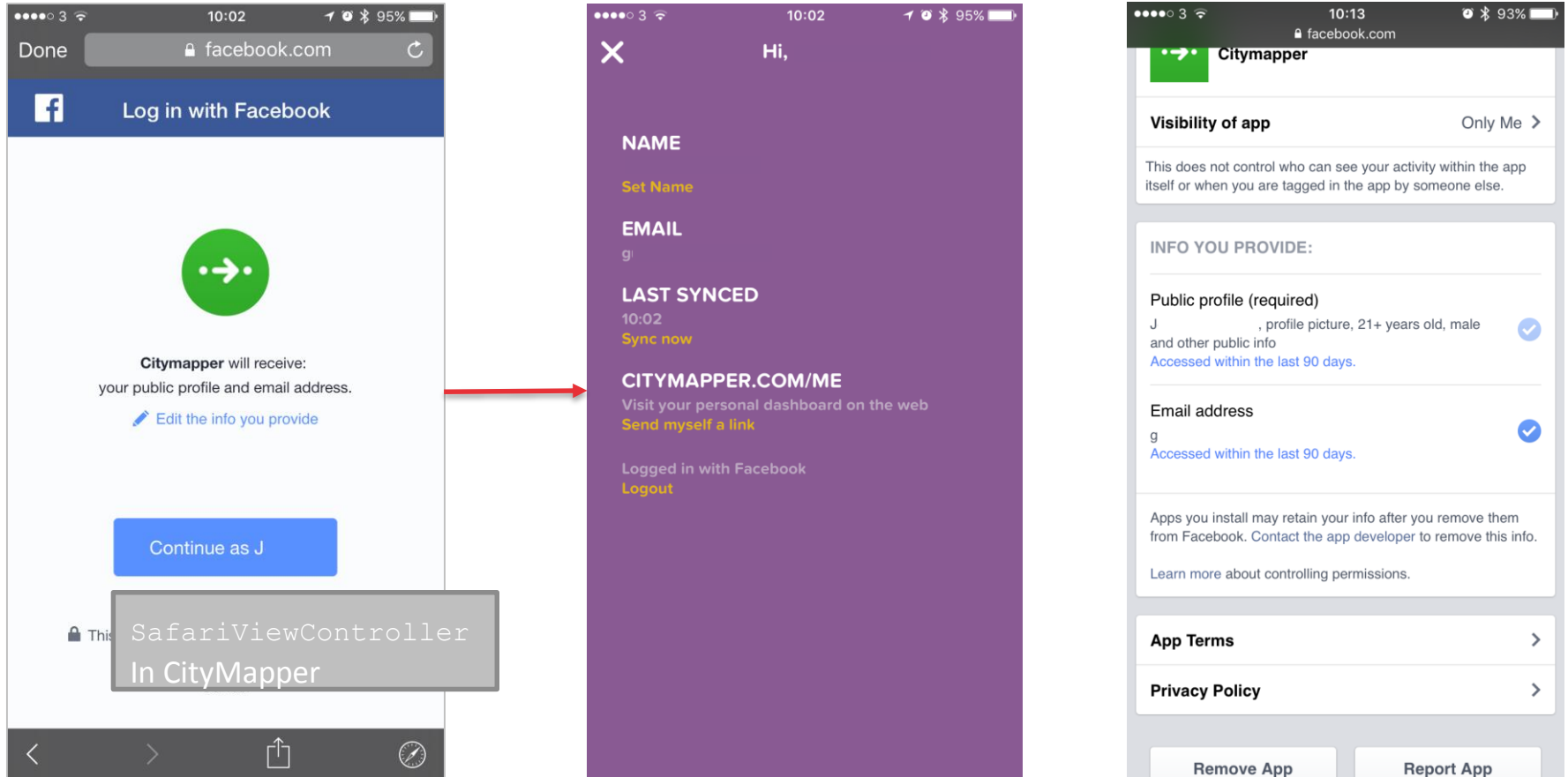
## With OAuth



Credentials are only sent to the authorisation provider

# ◆◆◆ Authentication in Mobile Applications

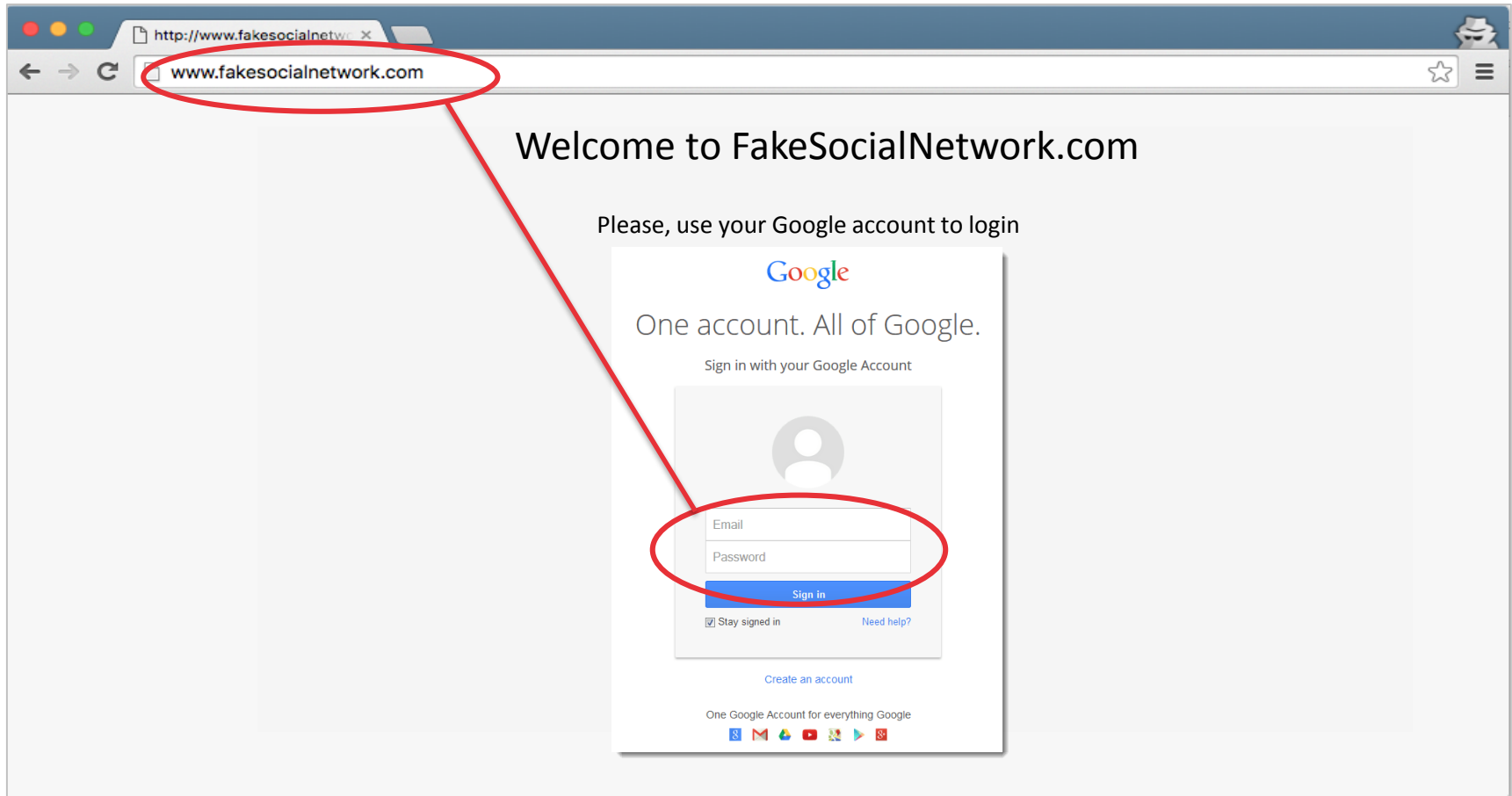
## With OAuth



The user authorises the information to share and may modify it at any moment through its provider.

# ◆◆◆ Authentication in Mobile Applications

## Authentication before OAuth



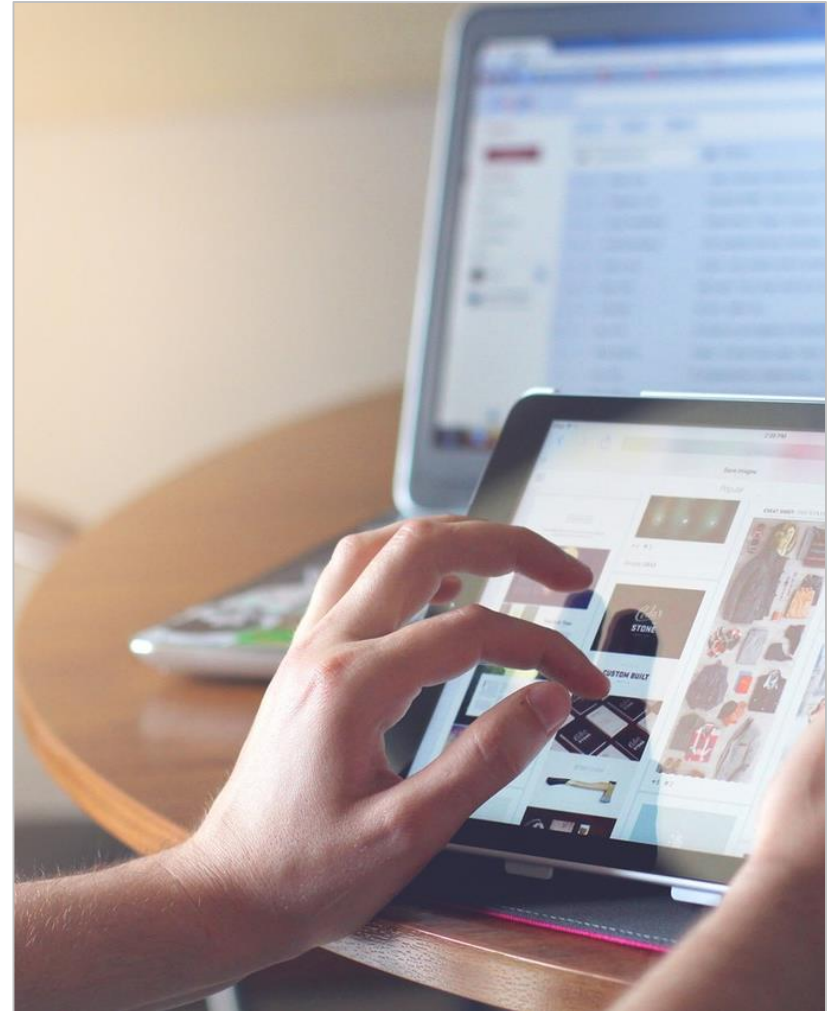
For users to authorise or authenticate in a web, credentials of the authorisation provider should be introduced in an unreliable web.



# ◆◆◆ Authentication in Mobile Applications

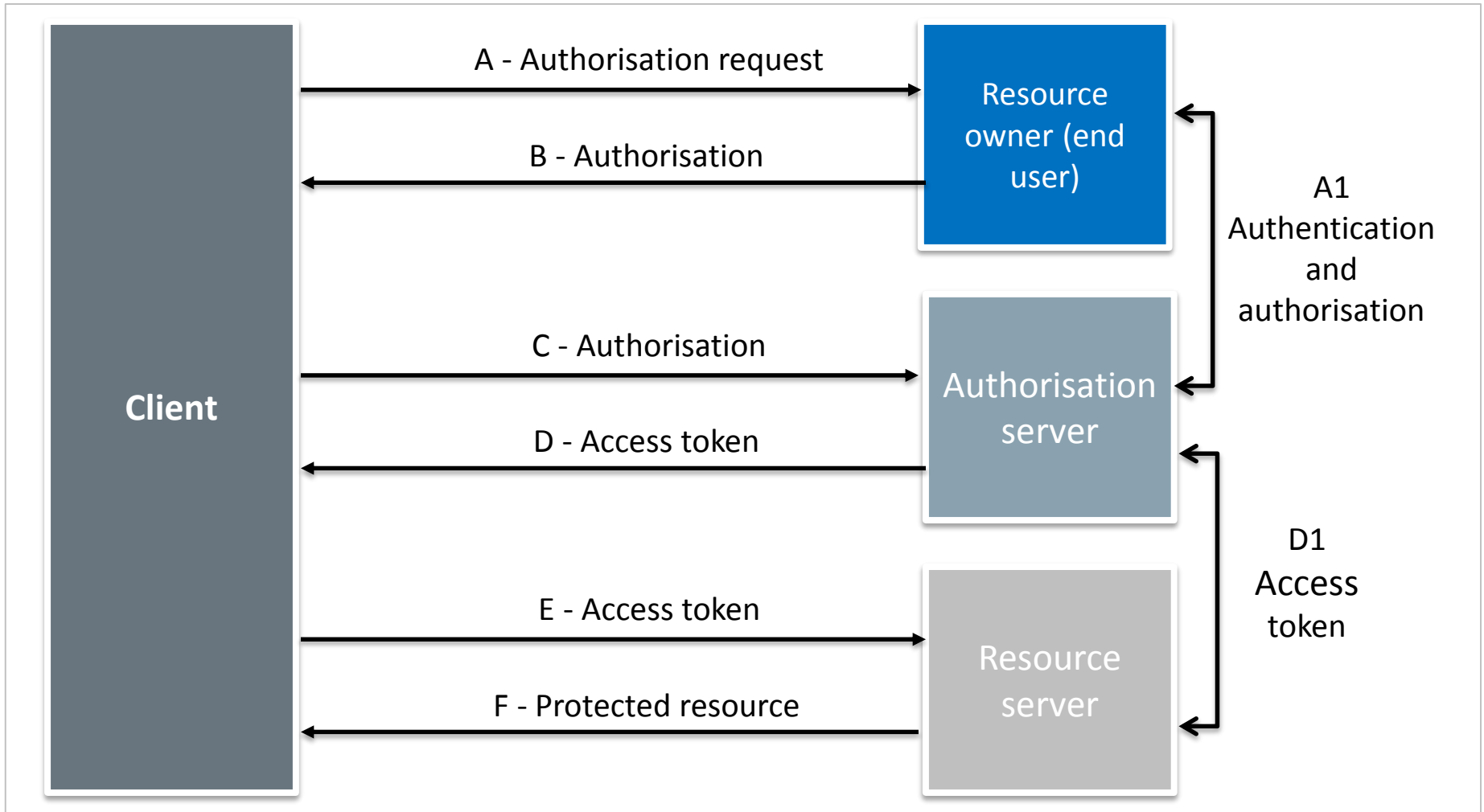
## Roles

- Four different roles participate during the execution of the OAuth protocol:
  - **Resource owner:** an entity capable of granting access to a specific resource. In most cases of the mobile environment, it will be the end user.
  - **Resource server:** the server hosting the client's resource. It is able to provide access to it if the proper credentials are used (access tokens that will be described later).
  - **Client:** the application that requests access to the resource on behalf of the resource owner. In the case of mobile applications, it may be the application itself or its back-end.
  - **Authorisation server:** server that creates access tokens for the client when the resource owner has been authenticated in the resource server. The resource server and the authorisation server roles are often executed by the same entity.



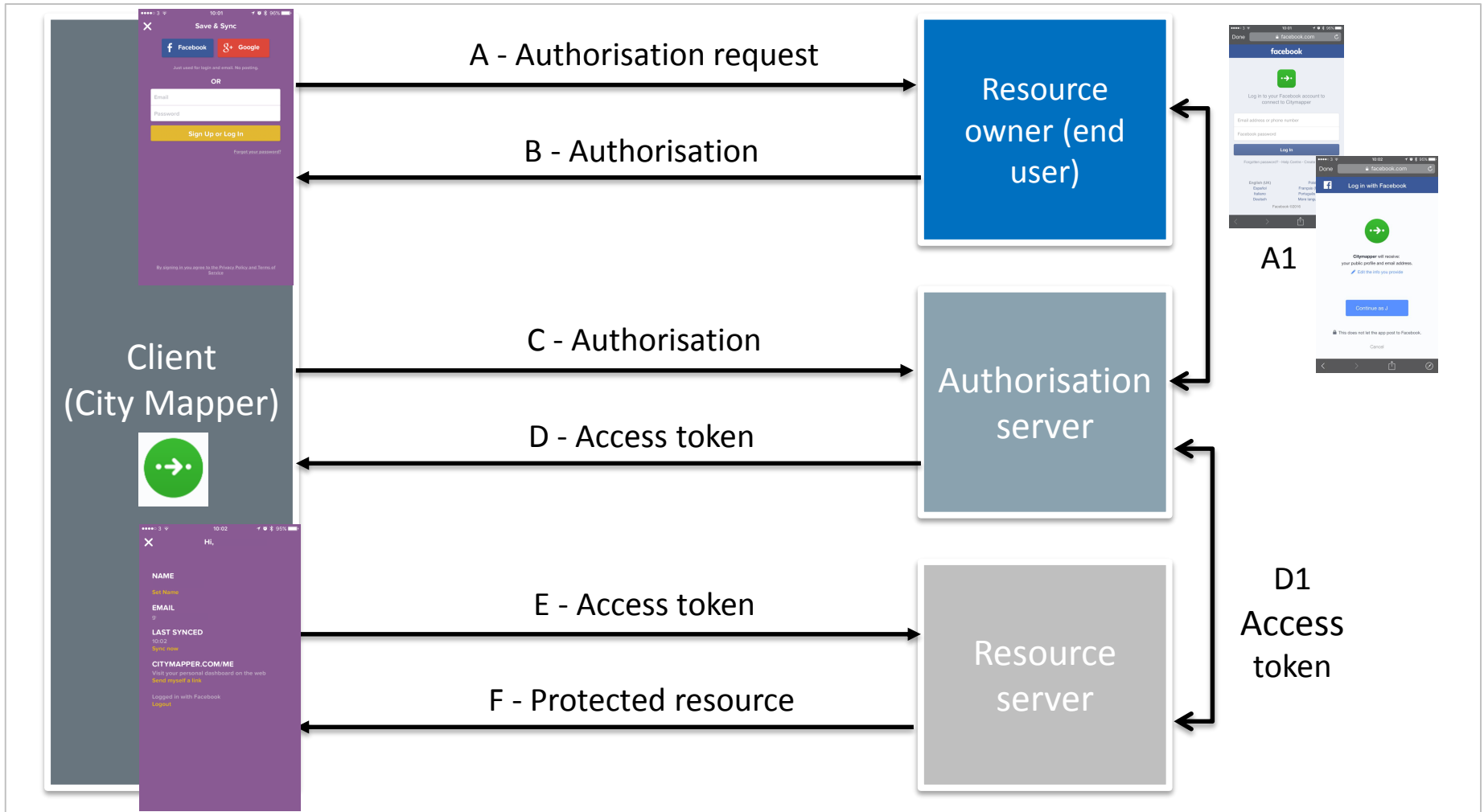
# ◆◆◆ Authentication in Mobile Applications

## Scheme



# ◆◆◆ Authentication in Mobile Applications

## Scheme on the Example of City Mapper



# ◆◆◆ Authentication in Mobile Applications

## Description of the Scheme

- A) The client requests the authorisation from the owner of the resource. The authorisation may be solved locally (in the mobile device) if the resource (account) is configured within the device itself (account management frameworks). If the account is not configured, it can be obtained with the authorisation server acting as intermediary (A1).
- B) The client receives the authorisation (a credential that means that the client has access to that specific resource). OAuth 2.0 defines four different types of authorisation:
  - Authorisation code: obtained when stage A1 is executed.
  - Implicit code: the client is issued the authentication token directly after A1. When possible, it should be avoided due to security implications.
  - Password: user and password credentials are used. It eliminates all the scheme's security; therefore, it should always be avoided.
  - Client's credentials: if the resource belongs to the client or access has been allowed, it is only necessary to use the client's credentials.

# ◆◆◆ Authentication in Mobile Applications

## Description of the Scheme

- C) The client is authenticated in the authorisation server by using credentials (it is necessary to be registered) and the authorisation received during the previous stage.
- D) The server validates the client's credentials and authorisation. If they are correct, it creates an access token:
  - The authorisation and resource servers share the access token (in case they are not the same entity) through an out-of-band channel.
  - In addition to the resources that the access token enables access to, it may also include the time during which access is allowed.
- E) The client requests the resource server access to the resource. To this end, the client presents the access token obtained in the previous stage.
- F) The resource server validates the access token and, if the validation is correct, it sends the requested resource.

## Authorisation in Mobile Operative Systems

- In the mobile environment, the authorisation and resource servers may be accessed via three different mechanisms:
  - **APIs of the operative system accounts:** the application that provides the resources should have registered an account in the operative system itself. The client requests the system accounts, and the system redirects the request to the specific authentication/authorisation API of the account selected by the user.
  - **Applications:** it is similar to the previous case, but the client requests the authorisation directly to the resource/authorisation server through a specific application already installed in the client. To this end, communication libraries between applications are used.
  - **Web views:** if the resource/authorisation server application is not installed on the device, the web view can be used to carry out the process. Once the user has been authorised, the client may extract the access token from the corresponding web view.



# ◆◆◆ Authentication in Mobile Applications

## Authorisation on Android – AccountManager I

- Android's `AccountManager` can be use to create authorisation requests to the Google account installed in the device or to the accounts of other providers that have implemented the required methods within their application.
- Permissions required to access the devices' accounts.

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
```

- During the creation of the activity it is verified whether the user has already provide authorisation.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    AccountManager accountManager = AccountManager.get(this);
    if (tokenGuardado() != null) {
        accionesAutenticado(tokenGuardado());
    } else {
        escogerCuenta();
    }
}
```

# ◆◆◆ Authentication in Mobile Applications

## Authorisation on Android – AccountManager II

- If the client has not been authorised, the account that is going to be used to access the resource should be requested (com.packet.account).

```
private void escogerCuenta() {  
    int ACCOUNT_REQUEST_CODE = 1601;  
    Intent intent = AccountManager.newChooseAccountIntent(null, null,  
        new String[] { "com.paquete.cuenta" }, false, null, null, null, null);  
    startActivityForResult(intent, ACCOUNT_REQUEST_CODE);  
}
```

- The response will include the name of the account, therefore, we should obtain the account and make the authentication token request for this resource.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    if (resultCode == RESULT_OK) {  
        if (requestCode == ACCOUNT_REQUEST_CODE) {  
            String accountName = data.getStringExtra(AccountManager.KEY_ACCOUNT_NAME);  
            for (Account account : accountManager.getAccountsByType("com.paquete.cuenta")) {  
                if (account.name.equals(accountName)) {  
                    userAccount = account;  
                    break;  
                }  
            }  
            //RESOURCE WILL DEPEND ON THE RESOURCE THAT IS GOING TO BE ACCESSED  
            //IT IS REQUIRED FO HANGOUT "https://www.googleapis.com/auth/googletalk";  
            accountManager getAuthToken(userAccount, "oauth2:" + RECURSO null, this,  
                new OnTokenAcquired(), null);  
        }  
    }  
}
```

# ◆◆◆ Authentication in Mobile Applications

## Authorisation on Android – AccountManager III

- Once the authorisation token has been obtained, a net object such as `OnTokenAcquired` is called, which acts as callback for the reception of the authentication token. The token should be stored as a sensitive data of the application.

```
private class OnTokenAcquired implements AccountManagerCallback<Bundle> {  
    @Override  
    public void run(AccountManagerFuture<Bundle> result) {  
        try {  
            Bundle bundle = result.getResult();  
            Intent launch = (Intent) bundle.get(AccountManager.KEY_INTENT);  
            if (launch != null) {  
                //FAILED AUTHORISATION, YOU SHOULD RETRY IT  
                startActivityForResult(launch, AUTHORIZATION_CODE);  
            } else {  
                String token = bundle.getString(AccountManager.KEY_AUTHTOKEN);  
                guardarToken(token)  
                accionesAutenticado(token);  
            }  
        } catch (Exception e) {  
            throw new RuntimeException(e);  
        }  
    }  
}
```

# ◆◆◇ Authentication in Mobile Applications

## Authorisation on Android – Applications' SDKs

- It is possible that the resource server accepts resources requests through their application, but without being integrated with Android's account system.
- In such cases, the service itself often provides exhaustive documentation on how to use the application for the request of authentication tokens.
- The documentation generally includes the registration process of the client application and how to make calls between both applications in order to request and receive the authorisation token.
- Find some examples of such kind of authorisation:
  - Facebook: <https://developers.facebook.com/docs/facebook-login/android>
  - Twitter: through the following APIs:
    - Twitter Kit for Android <https://github.com/twitter/twitter-kit-android>
    - Twitter Core if Fabric is used <https://docs.fabric.io/android/twitter/twitter-core.html>

# ◆◆◆ Authentication in Mobile Applications

## Authorisation on Android - Browser

- If the application is not installed, a WebView (provided by the system) can be used to access the information. The functioning of OAuth may differ from one service to another.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    String url = URL_DE_OAUTH + "?client_id=" + ID_NUESTRA_APP_EN_SERVICIO_OAUTH;
    WebView webview = (WebView) findViewById(R.id.webview);
    webview.getSettings().setJavaScriptEnabled(true);
    webview.setWebViewClient(new WebViewClient() {
        public void onPageStarted(WebView view, String url, Bitmap favicon) {
            String accessTokenFragment = "access_token=";
            String accessCodeFragment = "code=";

            if (url.contains(accessTokenFragment)) {
                // Capture the request to look for authorisation codes and the token
                String accessToken = url.substring(url.indexOf(accessTokenFragment));
                guardarToken(accessToken);
            } else if (url.contains(accessCodeFragment)) {
                // If it is an access code, make another request to obtain the token
                String accessCode = url.substring(url.indexOf(accessCodeFragment));
                guardarCodigoAutorizacion(accessCode);
                String query = "client_id=" + ID_NUESTRA_APP_EN_SERVICIO_OAUTH + "&client_secret=" +
                    SECRETO_OBTENIDO_DURANTE_EL_REGISTRO_DE_NUESTRA_APP + "&code=" + accessCode;
                view.postUrl(OAUTH_ACCESS_TOKEN_URL, query.getBytes());
            }
        }
    });
    webview.loadUrl(url);
}
```

# ◆◆◆ Authentication in Mobile Applications

## Authorisation on iOS - Accounts Framework

- iOS enables the configuration of Facebook and Twitter accounts in the device's settings, so that other applications may request access to them.
- In this case, iOS privacy preferences allow the user to revoke access tokens directly.
- Find an example below to access the email account stored in the user's Facebook account.

```
ACAccountStore *accountStore = [[ACAccountStore alloc] init];
ACAccountType *accountType = [accountStore
accountTypeWithAccountTypeIdentifier:ACAccountTypeIdentifierFacebook];
NSDictionary *options = @{ ACFacebookAppIdKey: APPID_DEL_CLIENTE_EN_FACEBOOK,
                           ACFacebookPermissionsKey: @[@"email"]
                           };
[accountStore requestAccessToAccountsWithType:accountType
options:options
completion:^(BOOL granted, NSError *error){
    if (granted) {
        _currentUser.facebook = [accounts firstObject];
    } else {
        //ERROR MESSAGE
    }
}];
```



# ◆◆◆ Authentication in Mobile Applications

## Authorisation on iOS – Applications' SDKs

- The same as on Android, it is possible that the resource server accepts resources request through its application installed on iOS.
- In such cases, the service itself often provides exhaustive documentation on how to use the application for the request of authentication tokens.
- Find some examples of such kind of authorisation:
  - Facebook:  
<https://developers.facebook.com/docs/facebook-login/ios>
  - Foursquare:  
<https://developer.foursquare.com/resources/libraries>
  - Twitter: through Fabric's API:  
<https://docs.fabric.io/ios/twitter/authentication.html>



# ◆◆◇ Authentication in Mobile Applications

## Authorisation on iOS – Web Views

- On iOS, it is possible to use any of the three available web views to implement OAuth.
- Both UIWebView and WKWebView allow users to access credentials that the user writes on the view; therefore, it is recommended to use SafariViewController.
- SafariViewController runs in a separate process and, in addition, it will not request the user's credentials if they have previously been used with Safari.
- There is an OAuthSwift library with support to make OAuth requests via SafariViewControllers written in Swift.
  - <https://github.com/mwermuth/OAuthSwift/tree/swift2.0>
- Among the services supported, we can find Twitter, Flickr, Github, Instagram, Foursquare, Fitbit, LinkedIn, Dropbox and Salesforce, among others.

# Laboratory



## Introduction

- In this section of the unit, two laboratories will be carried out in order to display practically guidelines and good practices described throughout this unit.
- The laboratories will imply solving vulnerabilities identified in both applications (Android and iOS) analysed during unit 3.
- The laboratories will be focused on tasks related to the application's code. An S-SDLC should expand its activities to all the tasks performed when developing an application.



## Tasks

- Modifications made to the source code of the applications have been organised into tasks.
- Tasks are part of a work that students should carry out on their own.
- In order to motivate learning, tasks are divided into two essential parts:
  - Motivation and description of the task to perform, including the type of results expected.
  - Procedure to carry out the task and expected results.
- Both parts are described in different slides.
- This is intended for students to try to perform the task with no access to the procedure.
- Students will be able to use the previously described procedure in order to check the solution and to solve possible doubts regarding the topic.



A group of green Android robots standing in a row, with the text "Android Laboratory" overlaid in the center.

Android Laboratory

## Procedure

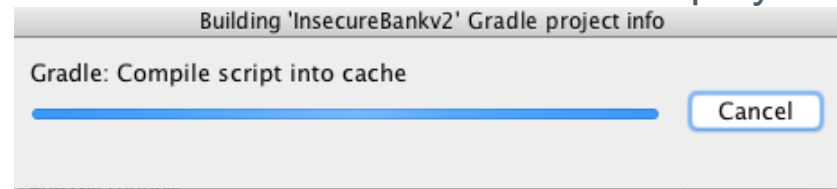
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____

- The following tasks will be performed during the laboratory:
  - Preparation of the work environment.
  - Correct configuration of components of the application.
  - Credentials storage.
  - Elimination of the administrator functionality.
  - Elimination of information leakages through logs.
  - Permissions.



## Preparation of the Environment

- A folder is created in **Santoku** or in the computer in which the environment for the development of Android is installed.
  - > `cd Documents`
  - > `mkdir laboratorio_android`
  - > `cd laboratorio_android`
- The repository is cloned:
  - > `git clone https://github.com/dineshshetty/Android-InsecureBankv2.git`
  - > `git checkout 6267a02c80a6a5bfff7c26b71d9125c0c7039fe79` .
  - > `cd Android-InsecureBankv2`
- On Android, navigate to the *InsecureBankv2* folder in the mentioned directory and select “Choose”.
- After a load period, Android Studio’s window will be displayed with the project.



## Correct Configuration of Components of the Application

- In this task, the student will solve all the problems detected in the configuration of components during the analysis of the application.

### **Task**

Modify the configuration of all the components of the application that have been declared in the manifest so that they cannot be used in a malicious way by other applications.

### **Expected result**

A manifest file including the correct security configuration for each of the components of the application.

## Correct Configuration of Components of the Application

### Solution

- First, elements that include *exported=true* are verified.
- Activities:

```
<activity
    android:name=".ChangePassword"
    android:exported="true"
    android:label="@string/title_activity_change_password" >
</activity>
<activity
    android:name=".DoTransfer"
    android:exported="true"
    android:label="@string/title_activity_do_transfer" >
</activity>
...
```

```
<activity
    android:name=".ViewStatement"
    android:exported="true"
    android:label="@string/title_activity_view_statement" >
</activity>
<activity
    android:name=".PostLogin"
    android:exported="true"
    android:label="@string/title_activity_post_login" >
</activity>
```

- Receivers:

```
<receiver
    android:name=".MyBroadCastReceiver"
    android:exported="true" >
    <intent-filter>
        <action android:name="theBroadcast" >
        </action>
    </intent-filter>
</receiver>
```

- Provider:

```
<provider
    android:name=".TrackUserContentProvider"
    android:authorities="com.android.insecurebankv2.TrackUserContentProvider"
    android:exported="true" >
</provider>
```

### Correct Configuration of Components of the Application

#### **Solution**

- After analysing the activities' functionality, none of them needs to be accessed by other applications; therefore, the attribute is eliminated.
- When the functionality of the receiver is analysed, it is observed that, according to its comments and code, its objective is to send the user an SMS message to confirm the change of password.
- It should not be executed from the phone due to different reasons:
  - It implies an economic cost for it.
  - It is highly likely that the user executes the application in a device in which the bank account is configured.
  - The operation is not performed from a reliable element of the system (the phone).
- Therefore, for the time being, the functionality of the mobile application is eliminated, and thus, the mention to the manifest receiver is also eliminated (apart from the code).
- The functionality of the receiver should be implemented in the server via an SMS gateway (outside the scope of the laboratory).

## Correct Configuration of Components of the Application

### Solution

- The code responsible for executing the BroadcastIntent, once the change of password has been performed, should also be eliminated.
- Such code is located in the ChangePassword activity.

```
/*  
The function that handles the SMS activity  
phoneNumber: Phone number to which the confirmation SMS is to be sent  
*/  
  
broadcastChangepasswordSMS(phoneNumber, changePassword_text.getText().toString());
```

```
private void broadcastChangepasswordSMS(String phoneNumber, String pass) {  
  
    if(TextUtils.isEmpty(phoneNumber.toString().trim())) {  
  
        System.out.println("Phone number Invalid.");  
    }  
    else  
    {  
        Intent smsIntent = new Intent();  
        smsIntent.setAction("theBroadcast");  
        // String actdns= smsIntent.getAction().toString();  
        // Toast.makeText(getApplicationContext(),actdns , Toast.LENGTH_LONG).show();  
        smsIntent.putExtra("phonenumber", phoneNumber);  
        smsIntent.putExtra("newpass", pass);  
        sendBroadcast(smsIntent);  
    }  
}
```

## Correct Configuration of Components of the Application

### Solution

- The `TrackUserContentProvider` provider, as it is described, is in charge of maintain a list of users registered in the application.
- Without going into detail regarding such kind of provider in a real environment and whether the implementation is correct (it will be verified in another task), we will protect it by defining new permissions to display the protection process of a provider via permissions.
- This way, applications that want to access such information should declare some of the r

```
<permission android:name="com.android.insecurebankv2.READ_TRACKING" />
<permission android:name="com.android.insecurebankv2.WRITE_TRACKING" />

<provider
    android:name=".TrackUserContentProvider"
    android:authorities="com.android.insecurebankv2.TrackUserContentProvider"
    android:exported="true"
    android:readPermission="com.android.insecurebankv2.READ_TRACKING"
    android:writePermission="com.android.insecurebankv2.WRITE_TRACKING" >
</provider>
```

- First, permissions are defined and, then, they are assigned in the provider.

## Credentials Storage

- In this task, students will review the code related to credentials storage that is performed in the `DoLogin` activity in order to adapt its security.

### Task

Review and take the proper actions regarding the storage of credentials performed in the `DoLogin` activity by default.

### Expected result

A `DoLogin` activity that does not store the user's credentials insecurely.



## Credentials Storage

### Solution

- First, we will study the code of the method mentioned in the statement.

```
/*
The function that saves the credentials locally for future reference
username: username entered by the user
password: password entered by the user
*/
private void saveCreds(String username, String password) throws UnsupportedOperationException, InvalidKeyException {
    // TODO Auto-generated method stub
    SharedPreferences mySharedPreferences;
    mySharedPreferences = getSharedPreferences(MYPREFS, Activity.MODE_PRIVATE);
    SharedPreferences.Editor editor = mySharedPreferences.edit();
    rememberme_username = username;
    rememberme_password = password;
    String base64Username = new String(Base64.encodeToString(rememberme_username.getBytes(), 4));
    CryptoClass crypt = new CryptoClass();
    superSecurePassword = crypt.aesEncryptedString(rememberme_password);
    editor.putString("EncryptedUsername", base64Username);
    editor.putString("superSecurePassword", superSecurePassword);
    editor.commit();
}
```

- We can observe that a specific class is being used to encrypt credentials and it is then stored in a Shared Preferences file.
- The analysis of the cryptographic library shows that cryptography is not being used in a secure way.

```
// The super secret key used by the encryption function
String key = "This is the super secret key 123";


// The initialization vector used by the encryption function
byte[] ivBytes = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

## Credentials Storage

### Solution

- Since credentials stored are related to a bank account, they should not be stored in the device.
- Therefore, the first option should be the elimination of functionalities of the application related to the storage of such credentials.
- During the elimination, it is verified whether the login is correct. Details of such login are provided in the device's log. Then, such functionality is also eliminated.

```
InputStream in = responseBody.getEntity().getContent();
result = convertStreamToString( in );
result = result.replace("\n", "");
if (result != null) {
    if (result.indexOf("Correct Credentials") != -1) {
        Log.d("Successful Login:", " , account=" + username + ":" + password);
        trackUserLogins();
        Intent pL = new Intent(getApplicationContext(), PostLogin.class);
        pL.putExtra("uname", username);
        startActivity(pL);
    } else {
        Intent xi = new Intent(getApplicationContext(), WrongLogin.class);
        startActivity(xi);
    }
}
```



## Credentials Storage

### Solution

- In addition, the interface elements that allow users to reset credentials stored and methods that recover information from the LoginActivity and its corresponding interface should be eliminated.
- Button for data fill-in:

```
Button fillData_button;
```

- Initialisation from onCreate():

```
fillData_button = (Button) findViewById(R.id.fill_data);
fillData_button.setOnClickListener((v) -> {
    // TODO Auto-generated method stub
    try {
        fillData();
    } catch (InvalidKeyException | UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
});
```

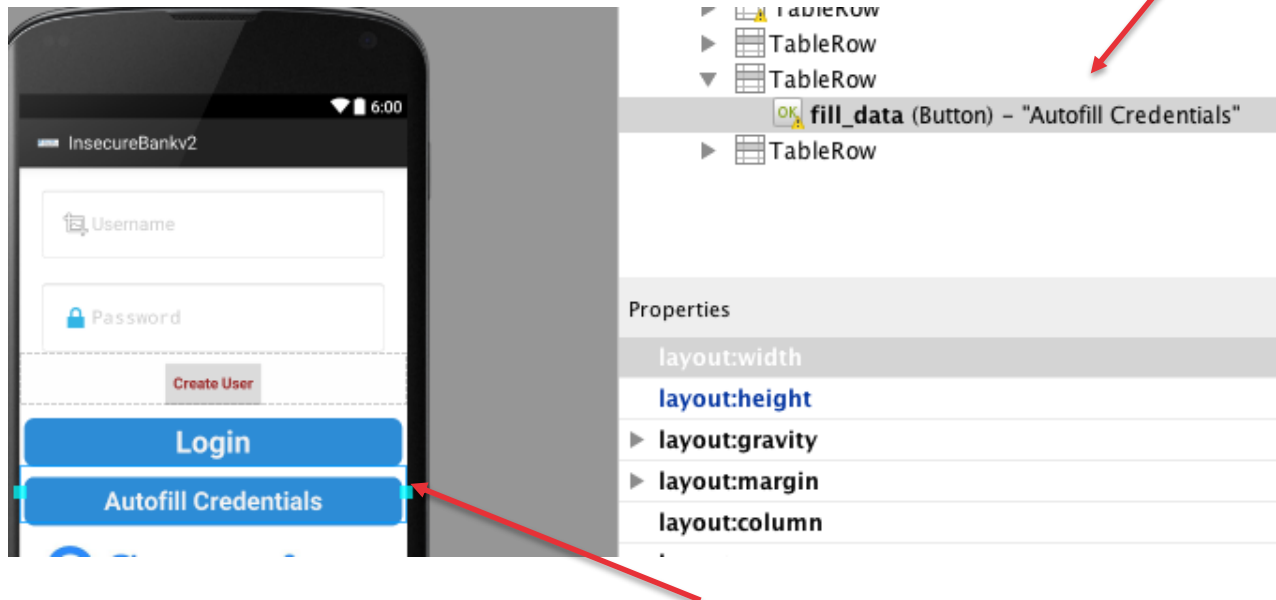
- fillData method:

```
/*
 * The function that allows the user to autofill the credentials
 * if the user has logged in successfully at least one earlier using
 * that device
 */
protected void fillData() throws UnsupportedEncodingException, InvalidKeyException {
    // TODO Auto-generated method stub
    // ... (code omitted) ...
}
```

## Credentials Storage

### Solution

- The corresponding element of the *res/activity\_log\_main.xml* layout is eliminated:



## Credentials Storage

### Solution

- From Android 6.0 it is possible to use the system's KeyStore to create symmetric keys that allow users to protect certain secrets of the device:
  - Once the device has been authenticated, it receives an authentication token from the server.
  - A key used to encrypt the token is created by using the KeyStore.
  - The encrypted token is stored in the device's internal memory.
  - When the key is required, the user will be asked for the lock code.
- Find below the code for the creation of a key that requires the user to enter the lock code every time that it is going to be used:

```
KeyGenParameterSpec.Builder builder = new KeyGenParameterSpec.Builder("claveapp",
    KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT);
KeyGenParameterSpec keySpec = builder
    .setKeySize(256)
    .setBlockModes("CBC")
    .setEncryptionPadding("PKCS7Padding")
    .setRandomizedEncryptionRequired(true)
    .setUserAuthenticationRequired(true)
    .setUserAuthenticationValidityDurationSeconds(5 * 60)
    .build();
KeyGenerator kg = KeyGenerator.getInstance("AES", "AndroidKeyStore");
kg.init(keySpec);
SecretKey key = kg.generateKey();
```

## Credentials Storage

### Solution

The application also saves a file in the SD card for each account movement.

- In order to increase the security of such data, the directory in which files are stored is moved to the application's sandbox.
- In the DoTransfer activity:

```
JSONObject = new JSONObject(result);
acc1 = jsonObject.getString("from");
acc2 = jsonObject.getString("to");
System.out.println("Message:" + jsonObject.getString("message") + " From:" + from.getText().toString() + " To:" + to.getText().toString());
final String status = new String("\nMessage:" + "Success" + " From:" + from.getText().toString() + " To:" + to.getText().toString());
try {
    // Captures the successful transaction status for transaction history tracking
    String MYFILE = Environment.getExternalStorageDirectory() + "/Statements_" + usernameBase64ByteString + ".html";
    BufferedWriter out2 = new BufferedWriter(new FileWriter(MYFILE, true));
    out2.write(status);
    out2.write("<hr>");
}
```

- In the ViewStatement activity:

```
Intent intent = getIntent();
uname = intent.getStringExtra("uname");
//String statementLocation=Environment.getExternalStorageDirectory()+ "/Statements_" + uname + ".html";
String FILENAME="Statements_" + uname + ".html";
File fileToCheck = new File(Environment.getExternalStorageDirectory(), FILENAME);
System.out.println(fileToCheck.toString());
if (fileToCheck.exists()) {
    //Toast.makeText(this, "Statement Exists!!",Toast.LENGTH_LONG).show();

    WebView mWebView = (WebView) findViewById(R.id.webView1);
    // Location where the statements are stored locally on the device sdcard
    mWebView.loadUrl("file://" + Environment.getExternalStorageDirectory() + "/Statements_" + uname + ".html");
    mWebView.getSettings().setJavaScriptEnabled(true);
    mWebView.getSettings().setSaveFormData(true);
    mWebView.getSettings().setBuiltInZoomControls(true);
}
```

### Elimination of the Administrator Functionality

- In the previous task, it was observed that there is a button that is not showed in the application by default and that enables the creation of users in the application.

#### **Task**

Eliminate all the hidden functionalities of the login activity in the mobile application that may allow attackers to abuse the service via reverse engineering processes.

#### **Expected result**

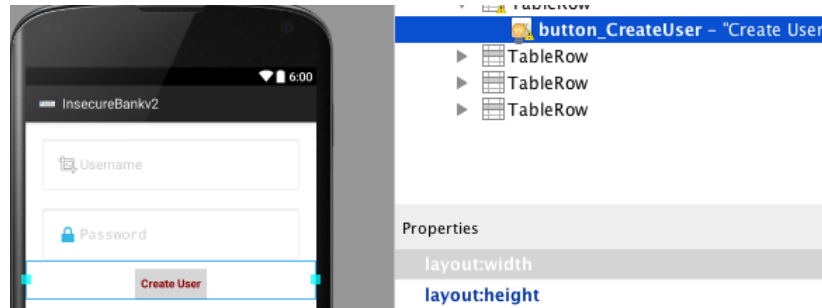
The code of the mobile application without hidden functionalities in the Login activity.



## Elimination of the Administrator Functionality

### Solution

- The added button should be eliminated in the layout file:



- Even though when checking the code, it is observed that there is no special functionality added, the code related to the button is eliminated:

```
// The Button that calls the create user function
Button createuser_buttons;

createuser_buttons = (Button) findViewById(R.id.button_CreateUser);
createuser_buttons.setOnClickListener((v) -> {
    // TODO Auto-generated method stub
    createUser();
});
```

## Elimination of Information Leakages Through Logs

- In this task, possible information leakages created by the device's console during the execution of the application will be eliminated.

### **Task**

Eliminate all sensitive information leakages that may be created during the execution of the application.

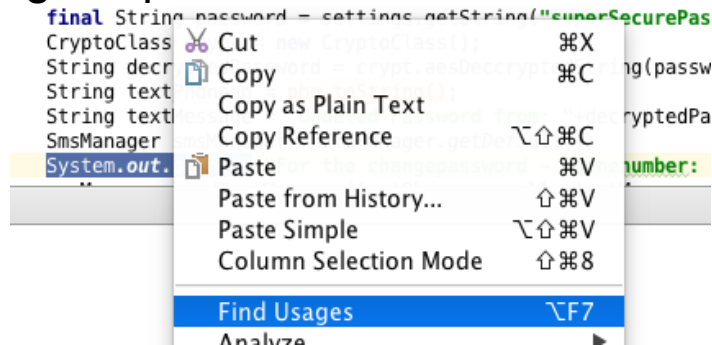
### **Expected result**

The code of the application without calls to logs or the console, including information that may be considered as sensitive.

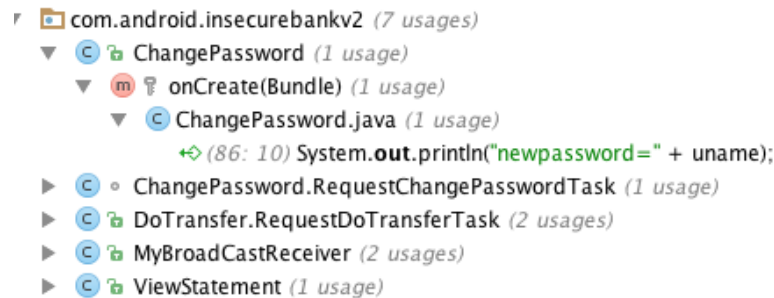
## Elimination of Information Leakages Through Logs

### Solution

- First, all the usages of System.out in the application should be searched through the findUsages option.



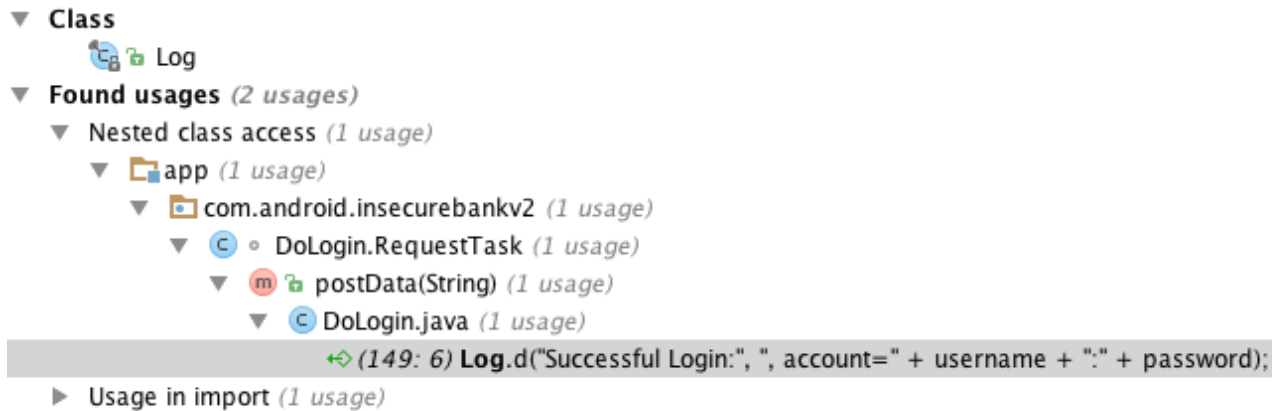
- The following results are obtained and eliminated after being inspected.



## Elimination of Information Leakages Through Logs

### Solution

- A search of uses of the Log class within the application is conducted. The following result should be obtained (if the Login entry has not been previously eliminated).



## Permissions

- Following the least privilege principle, in this task, students will eliminate permissions that the application does not require to be executed.

### **Task**

Analyse the use of each of the permissions of the application and eliminate them if they are not required for its proper functioning.

### **Expected result**

An updated manifest including the list of permissions strictly necessary for the execution of the application.

## Permissions

### Solution

- The application uses the following permissions:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.SEND_SMS" />

<!--
  To retrieve OAuth 2.0 tokens or invalidate tokens to disconnect a user. This disconnect
  option is required to comply with the Google+ Sign-In developer policies
-->
<uses-permission android:name="android.permission.USE_CREDENTIALS" /> <!-- To retrieve the acc
<uses-permission android:name="android.permission.GET_ACCOUNTS" /> <!-- To auto-complete the e
<uses-permission android:name="android.permission.READ_PROFILE" />
<uses-permission android:name="android.permission.READ_CONTACTS" />

<android:uses-permission android:name="android.permission.READ_PHONE_STATE" />
<android:uses-permission
  android:name="android.permission.READ_EXTERNAL_STORAGE"
  android:maxSdkVersion="18" />
<android:uses-permission android:name="android.permission.READ_CALL_LOG" />

<application
  android:allowBackup="true"
  android:icon="@mipmap/ic_launcher"
  android:label="@string/app_name"
  android:theme="@android:style/Theme.Holo.Light.DarkActionBar">
```

## Permissions

### **Solution**

- The Internet permission is necessary for the application to connect to the back-end of the bank that maintains it.
- The external storage was used to store movements. Since the storage location has been modified, we can eliminate read and writing permissions of the SD card.
- The permission to send SMS is not required anymore, since such operations should be performed from a gateway in the back-end.
- The application does not need to access the device's credentials. According to its current configuration, credentials are not stored and, in case they were, they would be credentials of the application and would not need any additional permission to access them. GET\_ACCOUNTS and USE\_CREDENTIALS permissions will be eliminated.



## Permissions

### Solution

- Permissions related to the phone and the history of calls are required for the application to know the number to which an SMS should be send when the password is modified. In the ChangePassword activity, we can observe the

```
TelephonyManager phoneManager = (TelephonyManager)getApplicationContext().getSystemService(Context.TELEPHONY_SERVICE);  
String phoneNumber = phoneManager.getLine1Number();  
System.out.println("phonno:"+phoneNumber);
```

- Since `<uses-permission android:name="android.permission.READ_PROFILE" />` and `<uses-permission android:name="android.permission.READ_CONTACTS" />` permissions.  
`<android:uses-permission android:name="android.permission.READ_PHONE_STATE" />`
- Students can also check that there are wrong defined permissions in the application (first on Android). They are also eliminated.
- The permission to read contacts is not required, since the application does not need to access them.
- The only permission that is still in the manifest is the INTERNET one.

## Additional Task

- Apart from the previous tasks, students may also perform additional tasks that would improve the security on the studied application.
- As an additional task, it is suggested to implement certificate pinning in the client's application.
- To this end, it is recommended to follow the steps below:
  - The server application code should be modified to provide SSL connections (<http://flask.pocoo.org/snippets/111/>).
  - It would be necessary to create a public and a private key with its corresponding certificate.
  - Finally, the pinning would be added to the application, as it was previously explained in the unit.
    - It is important to take into consideration that when creating the pinning, it is not necessary that the certificate is signed by a CA already registered in the device.
- If the student decides to complete this task, it is advisable to share the solution in the corresponding forum.



# iOS Laboratories

## Procedure

<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____
<input checked="" type="checkbox"/>	_____

- The following tasks will be performed during the laboratory:
  - Preparation of the work environment.
  - Credentials storage.
  - Avoid runtime handling.
  - Elimination of the administrator functionality.
  - Discover information leakages by using the logs.
  - Permissions.

## Preparation of the Environment

- This task should be performed at a computer configured with Mac OS X, as it was described in unit 0.
- In order not to interfere with the results of the analysis performed in unit 3, create a folder named “ios\_laboratory” in the documents directory.
  - > `cd Documents`
  - > `mkdir ios_laboratory`
  - > `Cd ios_laboratory`
- The repository is cloned:
  - > `git clone https://github.com/prateek147/DVIA.git`
  - > `cd DVIA`
- *DVIA/DamnVulnerableIOSApp/DamnVulnerableIOSApp.xcodeproj* is opened with Xcode.

## Data Storage

- In this task, the `saveInUserDefaultsTapped` code of the method will be reviewed in order to avoid information to be stored non encrypted in a file, by using `NSUserDefaults`.

### Task

In the `InsecureDataStorageVulnVC` file, modify the `saveInUserDefaultsTapped` method for it to store information securely in the device.

### Expected result

The information introduced in the field should be securely stored in the device.

## Data Storage

### Solution

- First, we will study the code of the method mentioned in the statement:

```
- (IBAction)saveInUserDefaultsTapped:(id)sender {
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    [defaults setObject:self.userDefaultsTextField.text forKey:@"DemoValue"];
    [defaults synchronize];
    [DamnVulnerableAppUtilities showAlertWithMessage:@"Data saved in NSUserDefaults"];
}
```

- The best storage method to avoid information to be stored without being encrypted, is to store data in the device's KeyChain.
- To this end, it is possible to use the API of Apple's KeyChain directly.
- There is a widely tested library that acts as a wrapper for KeyChain's API, called PDKeychainBindingsController.
- The library allows the user to store objects with the same methodology as NSUserDefaults.

```
- (IBAction)saveInUserDefaultsTapped:(id)sender {
    PDKeychainBindings *bindings = [PDKeychainBindings sharedKeychainBindings];
    [bindings setObject:self.keychainTextField.text forKey:@"DemoValue"];
    [DamnVulnerableAppUtilities showAlertWithMessage:@"Data saved in KeyChain"];
}
```



## Avoid Runtime Handling

- This task is aimed at preventing attackers from add a debugger to the application to modify its execution flow.

### **Task**

Add the required code to the application to avoid the use of debuggers during the execution of the application.

### **Expected result**

It will not be possible to load the application when the simulator or a device connected for its debugging are being used.

## Avoid Runtime Handling

### Solution

- During the unit, we have verified that it is possible to avoid the use of a debugger by using `PT_DENY_ATTACH`.
- If the user wants to prevent the debugger even from inspecting the beginning of the application, the call to *ptrace* should be added with the previous parameter when initiating the application, in the *main* method:

```
int main(int argc, char * argv[])
{
    ptrace(PT_DENY_ATTACH, 0, 0, 0);
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

- If the user wants to restrict the usage to some scenarios, it is possible to add guidelines to the compiler.

```
int main(int argc, char * argv[])
{
    #ifdef DEBUG
        ptrace(PT_DENY_ATTACH, 0, 0, 0);
    #endif
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

## Avoid the Use of the Clipboard

- In this task, we will prevent a user from copying information from a text field to the clipboard. Specifically, this task is focused on the view of leakages in the `SideChannelDataLeakageDetailsVC` clipboard.

### Task

Modify the configuration of the `SideChannelDataLeakageDetailsVC` view for the information of fields not to be copied.

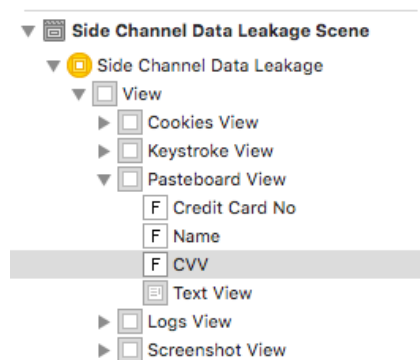
### Expected result

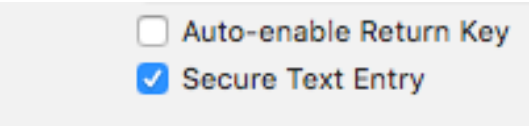
The application will not display the copy option in any of the fields that allow data entry.

## Avoid Runtime Handling

### Solution

- Properties of text fields are defined in the storyboard field of the application. Specifically, they are defined in the *Main.storyboard* file.
- To modify the properties of the field, it is necessary to access the corresponding view.



- The CVV field has  ☐ Auto-enable Return Key and ☒ Secure Text Entry. This configuration is used to disable the copy of data entered.
- When properties of other fields are modified the same way, the copy of data entered in the field is avoided.

## Certificate Pinning

- Nowadays, the connection via certificate pinning fails because the certificate stored in the application does not coincide with the one obtained when making the request to google.co.uk.

### **Task**

Modify the application for the connections performed via certificate pinning to be properly carried out.

### **Expected result**

The application will not display the copy option in any of the fields that allow data entry.

## Certificate Pinning

### Solution

- The first task to perform is to download the correct certificate in order to validate it from the application. In this solution, we will modify the approach and the pinning will be made via the SHA-256 summary of the certificate.
- The summary may be calculated from the app itself by modifying the code of the `willSendRequestForAuthenticationChallenge` delegate in the `TransportLayerProtectionVC` controller.

- If \ 

```
NSData *remoteCertificateData = CFBridgingRelease(SecCertificateCopyData(certificate));
NSMutableData *macOut = [NSMutableData dataWithLength:CC_SHA256_DIGEST_LENGTH];

CC_SHA256(remoteCertificateData.bytes, remoteCertificateData.length, macOut.mutableBytes);

NSLog(@"macOut: %@", macOut);
```

- The summary of the certificate is obtained through the log.

```
2016-02-22 23:59:45.926 DamnVulnerableIOSApp[3100:666535]
macOut: <71923f11 3890a377 4ac55b67 33a41d10 bc4014bc 74c7229a
e4e78507 ef0efa98>
```

## Certificate Pinning

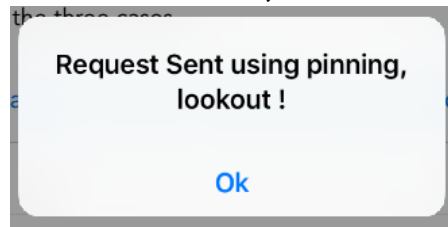
### Solution

- Then, it is only necessary to modify the code in order to change the verification made during the connection to verify the certificate received.

```
NSData *remoteCertificateData = CFBriddingRelease(SecCertificateCopyData(certificate));
NSMutableData *macOut = [NSMutableData dataWithLength:CC_SHA256_DIGEST_LENGTH];
CC_SHA256(remoteCertificateData.bytes, remoteCertificateData.length, macOut.mutableBytes);
NSString *pin= @"71923f113890a3774ac55b6733a41d10bc4014bc74c7229ae4e78507ef0efa98";
NSUInteger dataLength = [macOut length];
NSMutableString *remoteHashString = [NSMutableString stringWithCapacity:dataLength*2];
const unsigned char *dataBytes = [macOut bytes];
for (NSUInteger idx = 0; idx < dataLength; ++idx) {
    [remoteHashString appendFormat:@"%02x", dataBytes[idx]];
}

if ([remoteHashString isEqualToString:pin]) {
    [DamnVulnerableAppUtilities showAlertWithMessage:@"Request Sent using pinning, lookout !"];
    NSURLCredential *credential = [NSURLCredential credentialForTrust:serverTrust];
    [[challenge sender] useCredential:credential forAuthenticationChallenge:challenge];
}
```

- When the application is executed, it is verified that the validation works properly.



- This task may also be performed by using [TrustKit](#).





# Research Exercise

## ◆◆◆ Research Exercise

### Statement

- In this exercise, the student should conduct a research and write the results obtained on the subject's forum, in order to discuss them with the rest of students.
  - As it was verified during the unit and the different laboratories, data storage in the device is one of the greatest challenges for data security, to the point that many applications include encryption functions to protect, not only credentials, but all their data.
  - In this research exercise, the student is required to:
    - Identify, as we studied on Android and iOS, mechanisms included in Windows Phone and BlackBerry 10 operative systems for protection of data at rest.
    - Identify the weakest element of data protection and the circumstances required for an attacker to be able to recover them. Justify the response. This task should be performed for the four main operative systems: Windows Phone, BlackBerry 10, Android, and iOS.

A hand holding a pen is writing on a notebook. In the foreground, a black calculator and a textbook are visible. The textbook contains math problems, including arithmetic series and sigma notation. A blue semi-transparent banner is overlaid on the image, containing the text "Assessment Test".

# Assessment Test

Thank you for your  
attention

