Mobile Application Software Security

Jordan Gribben

CMP417: Engineering Resilient Systems Part 1

BSc Ethical Hacking Year 4

2021/22

Contents

1 Context

1.1 Introduction

Smartphones are an essential part of today's society, in the year 2020 1.38 billion devices were sold. Due to the valuable and extremely personal data they contain, it is important both the device along with any applications are as secure as possible. This report will focus on ScottishGlen's mobile application, this application allows staff to change their account details. With this information in mind the application is most likely communicating with a backend database to update the staff's information. Additionally, the app could potentially store data on the phone, this could be done in case the user does not currently have an internet connection and the data stored on the device will be uploaded as soon as an internet connection is established.

Like any piece of software mobile applications could potentially contain vulnerabilities ready to be exploited. These vulnerabilities could be present for various reasons such as something as simple as an oversight or even negligence. When it comes to mobile applications, positive technologies in 2018 found that insecure data storage was the most common issue between both iOS and Android devices, with this critical issue affecting around 76% of applications. This issue puts sensitive private information at risk, such as passwords, usernames, health data, and even financial information. (Positive Technologies, 2019)

Additionally, OWASP have produced a "mobile top 10", this list details the top 10 risks to mobile devices. In the 2016 list, insecure data storage is the second biggest risk to mobile devices (OWASP, 2016). This list also details the various impacts of an attack due to insecure data storage, such as looking at both the technical and business impact of a successful attack. Finally, the list also provides tips on how to prevent insecure data storage by recommending threat modelling the application, and understanding how the API's handle the applications processes. Internal and external links are also provided that contain guides on securing applications. (OWASP, 2016)

When it comes to mobile devices there are two main choices an Apple device or an Android device. In February 2022 devices using Android's Operating System (OS) had 70.97% of the global market share with iOS devices having 28.27% (Statcounter, 2022). With these statistics it is likely that ScottishGlen's employees use both Android and Apple devices. With this knowledge, the company's mobile application must be built to work for both platforms, or two separate applications specifically made for each operating system. When it comes to threats on mobile applications positive technologies discovered that high-risk vulnerabilities are present in around 38% of Apple's applications, and around 43% of Android applications (Positive Technologies, 2019). Even if this shows that Apple applications tend to have better cyber security the 5% difference is too small for Apple devices to ignore cyber security. ScottishGlen needs to keep their mobile application on both platforms secure, as if an insecure data storage vulnerability is found in one version, it could potentially be present in the other. This offers a unique challenge for mobile developers as instead of focusing on one application they may have to focus on multiple.

1.2 CVE

Previous Common Vulnerabilities and Exposures (CVE) have revealed various insecure data storage issues on both Android and Apple devices. The following is a list of CVEs relating to insecure data storage on mobile devices.

1.2.1 CVE-2014-0647

In version 2.6.1 of the Starbucks application for iOS devices the Crashlytics log filestores sensitive information in plaintext. This file can be located on the mobile device at the following file location "/Library/Caches/com.crashlytics.data/com.starbucks.mystarbucks/session.clslog". Reading the session.clslog file reveals information such as usernames, passwords, and email addresses (Mitre, 2014).

1.2.2 CVE-2021-43388

This vulnerability was discovered in the Unisys Cargo Mobile Application, any version of this application before version 1.2.29 has sensitive information stored in cleartext. If this were to be exploited the sensitive data could be leaked (Mitre, 2021).

1.2.3 CVE-2019-5633

On iOS mobile devices the Hickory Smart application stores sensitive information insecurely. The Hickory smart application is a simple device locking application that allows users to pair and control their Hickory Smart Bluetooth keypad deadbolt. The application allows users to control their lock via the app and give up to 10 people access to the lock. Any version of the application on 01.01.07 or prior has information within the application's database that could be used to control the locks remotely (Mitre, 2019).

1.2.4 CVE-2019-5632

Similarly to CVE-2019-5633 the Hickory Smart application was once again found to have insecure data storage. However instead of this vulnerability being found within iOS devices, CVE-2019-5632 is present within Android devices due to this, this vulnerability is found within version 01.01.43 and prior (Mitre, 2019).

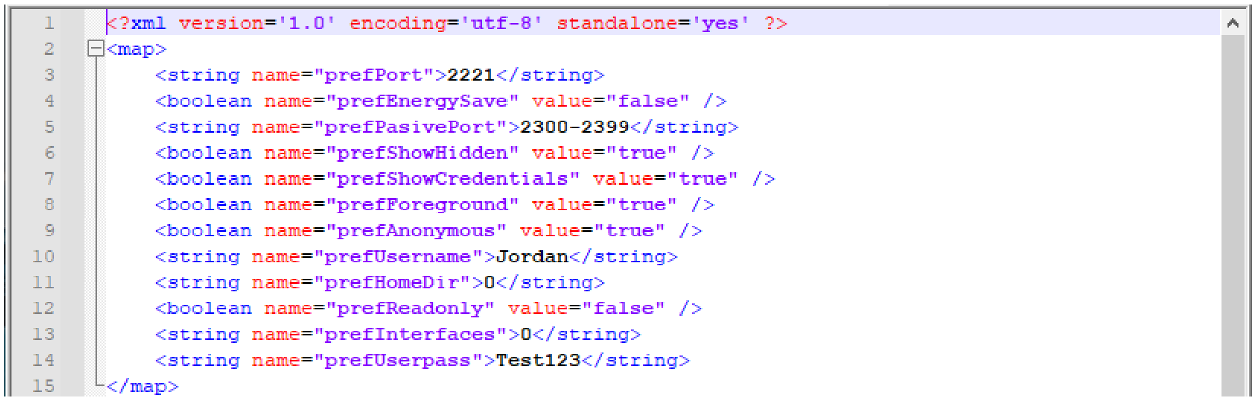
1.2.5 CVE-2018-11544

On Android devices, there is an FTP server application created by The Olive Tree. For this application a username and password can be set. However, these credentials are stored in plain text within an XML file. This issue persists in version 1.32 of the application (Mitre, 2018).

2 Implementation

By using OWASP's mobile top 10 it is possible that the insecure data storage on these CVEs could have been avoided. CVE-2018-11544 fall under OWASPs definition of insecure data storage. While the username and password should not be stored in plaintext as these credentials should be hashed and salted, more could be done to secure the file they are stored on.

This data is easily discoverable on the mobile device with the XML file being located at "data/data/com.theolivetree.ftpserver/shared_prefs/com.theolivetree.ftpserver_preferences.xml". Using Android studio the application was downloaded onto an emulated android device, within the application the username and password were changed from the default credentials to "Jordan" and "Test123". The XML file was then extracted, the contents of the file can be seen in figure A.



```xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="prefPort">2221</string>
    <boolean name="prefEnergySave" value="false" />
    <string name="prefPasivePort">2300-2399</string>
    <boolean name="prefShowHidden" value="true" />
    <boolean name="prefShowCredentials" value="true" />
    <boolean name="prefForeground" value="true" />
    <boolean name="prefAnonymous" value="true" />
    <string name="prefUsername">Jordan</string>
    <string name="prefHomeDir">0</string>
    <boolean name="prefReadonly" value="false" />
    <string name="prefInterfaces">0</string>
    <string name="prefUserpass">Test123</string>
</map>
```

Figure A - com.theolivetree.ftpserver_preferences.xml

The CVE related to the FTP server application is from 2018, while this CVE is a major security flaw due to it storing sensitive credentials insecurely, the developers of the application have chosen to neglect this security flaw rather than fix it. This CVE affects version 1.32 of the application which is the most recent version on the google play store, showing that this issue has been neglected for around 4 years. Additionally, this version of the app was published in 2017 meaning the vulnerability has been present for around 5 years. With the app having over a million installs, and reviews of the application as recent as February 2022 (The Olive Tree, 2017), this vulnerability being neglected is unacceptable and puts users at risk.

2.1 Mitigation

The CVEs surrounding insecure data storage could have potentially been prevented if a secure code review was completed. This review does not need to be carried out by the app's development team, the review could also be completed by an external party such as a cyber security specialist. Additionally, this review should take place before it is uploaded to any store that would allow it to be downloaded, in doing this it would make sure the application is safe for users before download. Consumers have a right to be safe online. It is the duty of companies that create apps and their developers to ensure the product they are providing is as safe as possible. By completing a secure code review, and following the recommendations in the OWASP mobile top 10, consumers will be able to feel safe knowing developers are doing all they can to keep their user's data safe. The following will focus on other ways to help solve the problem of insecure data on both iOS and Android devices.

2.1.1 iOS

When developing apps for iOS devices it is important developers use all the tools available to them, to ensure any data they are storing is kept secure. To ensure developers are storing their data securely Apple have created a framework that secures data, authenticates users, authorizes users, and help establish the validity of code. This framework is known as “Security Framework” (Apple, 2022). One way user data can be stored securely using Security Framework, is with the Keychain Services API, this should be used to store various amounts of user information such as usernames, passwords, and bank details. These details are then stored on Apple's Keychain, figure B shows secrets being stored within the Keychain (Apple, 2022).

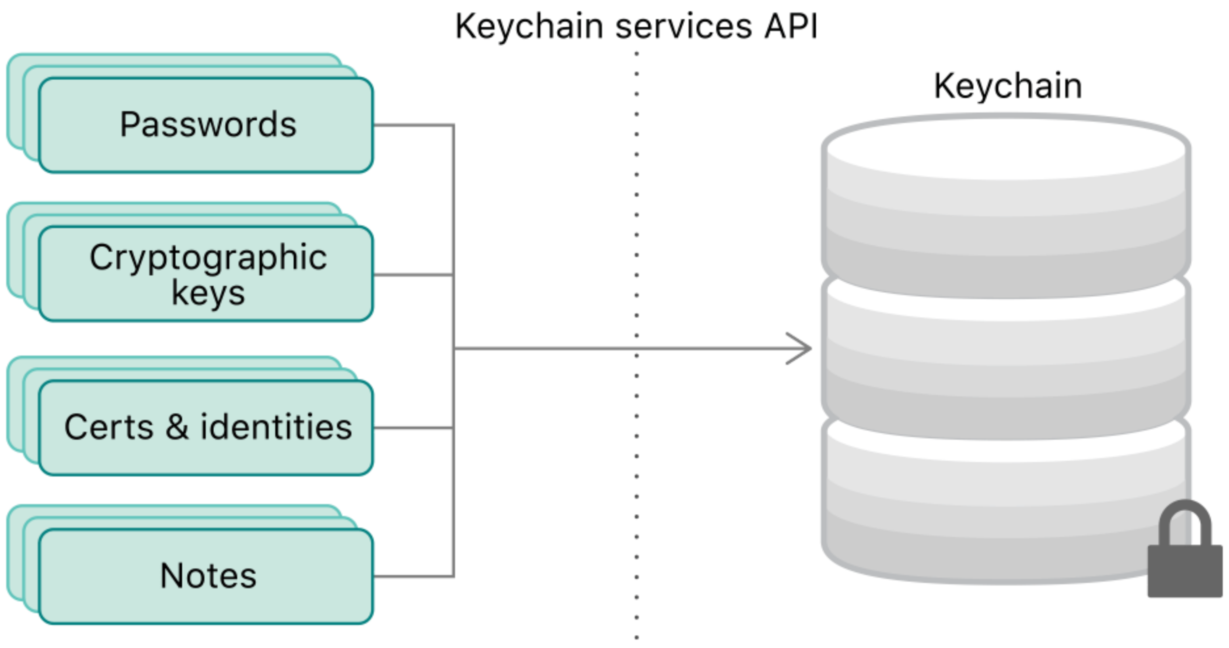


Figure B - Securing the user's secrets in a keychain

Additionally, the National Cyber Security Centre (NCSC) have produced guidance on application development, this guide includes information on Secure iOS application development (National Cyber Security Centre, 2018). While this guide looks at various topics, it takes an in depth look into Secure data storage. The guide recommends that when securing data, credentials should be stored using the Keychain APIs, when storing sensitive file system data the iOS Data protection API should be used. This guide should be essential for any app developers, and due to the NCSC being an official government body it can be seen as a legitimate and safe source when it comes to development.

2.1.2 Android

When developing for Android devices, Google have published a series of tutorials explaining various concepts and coding practices under the title Android for Developers. Android for Developers also includes tutorials on how to use Android Jetpack (Google, 2022). Android Jetpack is a set of libraries that helps developers follow best practices, it also includes a crypto library that helps developers store their data securely (Google, 2022).

Using Android Jetpack, developers can protect files that contain sensitive data, ensuring they are stored securely. This is done by generating various keys, with a master key being used to encrypt all the subkeys. Android Jetpack's, Jetpack Security (JetSec) provides a default master key, this key is obtained using the MasterKeys class and is then stored in the AndroidKeyStore. The AndroidKeyStore stores keys within a Trusted Execution Environment (TEE) or a StrongBox, this means that they are stored on a virtual environment on the operating system or a separate chip entirely. This makes the keys extremely difficult to extract making anything stored using them secure. Appendix A contains example code that uses JetSec to encrypt code (Markoff, 2020).

3 CONCLUSION

When developing applications for any device, developer guides and security information from trusted sources should be followed to ensure the app is secure as possible during development. However, no matter how careful developers are during the development process threats and vulnerabilities are bound to appear. Due to this, a security focused code review must be performed on apps before publication as these would most likely detect any issues before it is made available to the public. If this review took place on the FTP server application, these vulnerabilities would likely have been detected.

Having a specialist perform a secure code review before publication would provide developers with enough time and information to mitigate these issues before it puts any user's sensitive data at risk. These reviews are an extremely effective solution for discovering vulnerabilities, as they help discover a wide range of potential vulnerabilities within applications.

Conducting these reviews may also prevent companies from paying significant fines. There are various laws in place that hold companies accountable when it comes to protecting user information. For example, in the UK, GDPR alongside the data protection act 2018 have a maximum fine of £17.5 million or 4% of annual global turnover. Whereas the EU GDPR laws set a maximum fine of €20 million or 4% of annual global turnover (itgovernance, 2022). Ensuring the data is secured safely would help protect against a data breach so companies would not have to worry as much about facing these fines.

4 REFERENCES

Apple, 2022. *Keychain Services.* [Online]
Available at: https://developer.apple.com/documentation/security/keychain_services
[Accessed 4 March 2022].

Apple, 2022. *Security.* [Online]
Available at: https://developer.apple.com/documentation/security
[Accessed 4 March 2022].

Google, 2022. *Android Jetpack.* [Online]
Available at: https://developer.android.com/jetpack
[Accessed 3 March 2022].

Google, 2022. *Work with data more securely.* [Online]
Available at: https://developer.android.com/topic/security/data#kotlin
[Accessed 3 March 2022].

itgovernance, 2022. *GDPR penalties and fines.* [Online]
Available at: https://www.itgovernance.co.uk/dpa-and-gdpr-penalties
[Accessed 9 March 2022].

Markoff, J., 2020. *Data encryption on Android with Jetpack Security.* [Online]
Available at: https://medium.com/androiddevelopers/data-encryption-on-android-with-jetpack-security-e4cb0b2d2a9
[Accessed 8 March 2022].

Mitre, 2014. *CVE-2014-0647.* [Online]
Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0647
[Accessed 1 March 2022].

Mitre, 2018. *CVE-2018-11544.* [Online]
Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=2018-11544
[Accessed 1 March 2022].

Mitre, 2019. *CVE-2019-5632.* [Online]
Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=2019-5632
[Accessed 1 March 2022].

Mitre, 2019. *CVE-2019-5633.* [Online]
Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=2019-5633
[Accessed 1 March 2022].

Mitre, 2021. *CVE-2021-43388.* [Online]
Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-43388
[Accessed 1 March 2022].

National Cyber Security Centre, 2018. *Secure iOS application development.* [Online]
Available at: https://www.ncsc.gov.uk/collection/application-development/apple-ios-application-development/secure-ios-application-development
[Accessed 8 March 2022].

OWASP, 2016. *M2: Insecure Data Storage.* [Online]
Available at: https://owasp.org/www-project-mobile-top-10/2016-risks/m2-insecure-data-storage
[Accessed 28 February 2022].

OWASP, 2016. *OWASP Mobile Top 10.* [Online]
Available at: https://owasp.org/www-project-mobile-top-10/
[Accessed 28 February 2022].

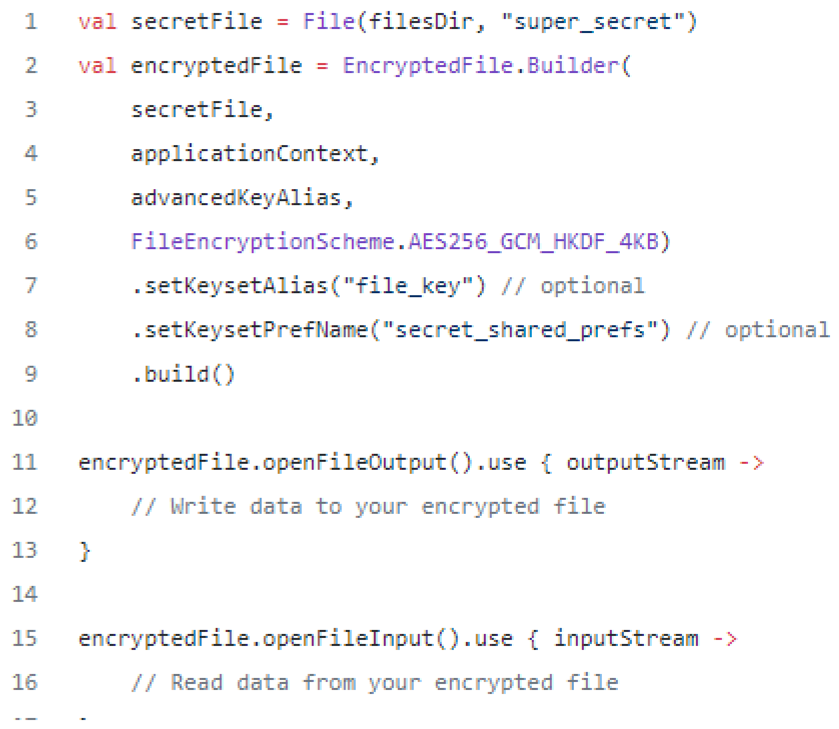
Positive Technologies, 2019. *Vulnerabilities and threats in mobile applications, 2019.* [Online]
Available at: https://www.ptsecurity.com/ww-en/analytics/mobile-application-security-threats-and-vulnerabilities-2019/
[Accessed 1 March 2022].

Statcounter, 2022. *Mobile Operating System Market Share Worldwide.* [Online]
Available at: https://gs.statcounter.com/os-market-share/mobile/worldwide
[Accessed 1 March 2022].

The Olive Tree, 2017. *Ftp Server1.* [Online]
Available at:
https://play.google.com/store/apps/details?id=com.theolivetree.ftpserver&hl=en_GB&gl=US
[Accessed 1 March 2022].

APPENDICES

APPENDIX A – ENCRYPTING FILE ON ANDROID DEVICES



```
val secretFile = File(filesDir, "super_secret")
val encryptedFile = EncryptedFile.Builder(
    secretFile,
    applicationContext,
    advancedKeyAlias,
    FileEncryptionScheme.AES256_GCM_HKDF_4KB)
    .setKeysetAlias("file_key") // optional
    .setKeysetPrefName("secret_shared_prefs") // optional
    .build()

encryptedFile.openFileOutput().use { outputStream ->
    // Write data to your encrypted file
}

encryptedFile.openFileInput().use { inputStream ->
    // Read data from your encrypted file
```