

# **CMP417: Engineering Resilient Systems 1**

## **Software Security**

*Thomas MacKinnon*  
*School of Design and Informatics*  
*Abertay University*  
*DUNDEE, DD1 1HG, UK*

## **Abstract**

This reports takes a look at the dangers that Insecure Data Storage in Mobile Applications could cause if left untreated, and documents how a simple Secure Engineering Practice can mitigate this vulnerability type. This is expressed through a hypothetically scenario were a small business is at risk of an attack and needs to upgrade the security of their software systems.

The mobile app, used to access a back end database, is deemed priority with the focus on securely storing data on the device. Code Reviewing meetings are presented as the best solution to avoid potential attacks for the business and is solving the issue of Insecure Data Storage. To show this a vulnerable mobile application is used to show how easily an Insecure file can be stolen from a device, and how easily it can be fixed with proper programming and Code Reviewing.

## Contents

1	Context	4
2	Recommendation	6
3	Implementation	7
4	References	9
5	Appendix	11

# 1 Context

Recent threats from a Hactivist group has put “Company Redacted” on edge, leading to the CEO investing more resources into Cyber Security. The company does not have a dedicated security specialist, so each employee is going to receive training about secure coding practices in order to make the business more resilient to malicious hackers.

Out of the available software systems the Mobile application was deemed to be first priority, as it can access staff information, which could be sensitive in nature, and has the potential to change the information stored. It is assumed this Mobile app is used to update and access information stored on a database webserver after authenticating the user. It is also probable that this Mobile app stores information on the users phone, such as remembered sign-in information or a token to allow access.

Mobile devices are increasingly becoming the target of attacks as they become a more and more prevalent in our daily life and even an essential tool for the majority of the population. With each passing year the number of CVEs (Common Vulnerabilities and Exposures) affecting Mobile devices and their Applications increase greatly (Mobliciti, 2020), meaning the need for secure software engineering is also increasing. Mid 2020 saw the rise of “Strandhogg 2.0” (CVE-2020-0096), an Android security vulnerability that gave hackers access to all of the victims apps, login details, GPS locations, texts and much more through undetectable phishing screens (Kumar M, 2020). This flaw in code has left over a billion devices vulnerable, and that’s just one vulnerability. A 2019 report by “Positive Technologies” (Positive Technologies, 2019) found that a staggering 76% of Mobile Application contained Insecure Data storage, leaving many users sensitive information like passwords and data at risk. Figure 1 shows that this security flaw occurs twice as much as the next result, making it likely that this vulnerability also effects the staffs app and should be priority when searching for weak spots.

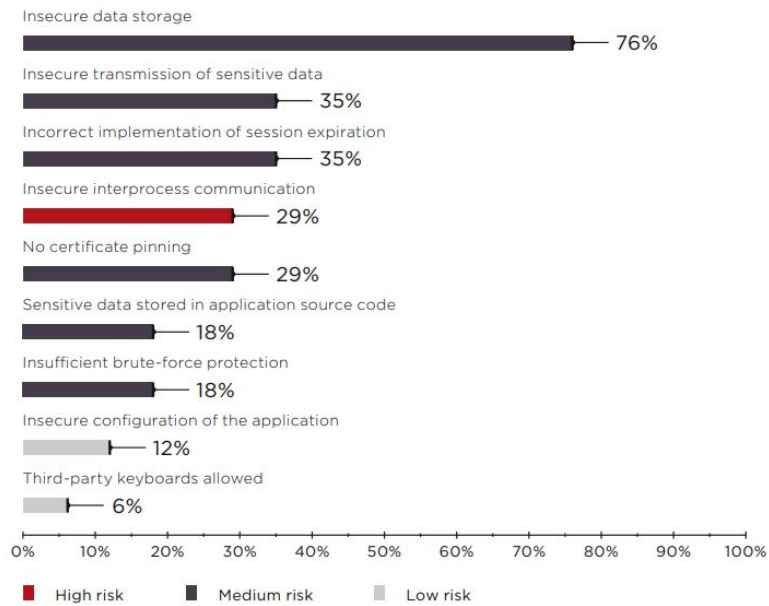


Figure 1: Positive Technologies bar chart of Mobile Application vulnerabilities by type

OWASP (Open Web Application Security Project), a foundation that works on improving security of software, backs this up further by placing Insecure Data Storage vulnerabilities as their second highest risk for mobile applications (OWASP, 2016). They highlight how this security can easily be exploited through computer based tools and can have severe impact on a business if not addressed. “Company Redacted” could face cases of Identity theft, data leakage, fraud and a huge hit to their reputation if this vulnerability is exploited. Insecure Data Storage has been documented as “CWE-922: Insecure Storage of Sensitive Information” in the “Common Weakness Enumeration (CWE)” list of software/hardware weakness types, with many associated listings of more specific weaknesses (The MITRE Corporation, 2020). A collection of CVEs related to Insecure Data Storage have been collected in order to give more context to the issue:

**CVE-2013-6986** - The ZippyYum Subway Kiosk app (version 3.4) for iOS stores a SQLite database on the users phone. However, all of the data is insecurely stored in plaintext, even sensitive information such as passwords (The MITRE Corporation, 2013).

**CVE-2014-0647** - The Starbucks app (version 2.6.1) for iOS stores all the users information in a plaintext Crashlytic log file named “session.clslog” which can be easily accessed in the device file system (The MITRE Corporation, 2014).

**CVE-2019-13096** - TronLink Wallet (version 2.2.0) allows users to store money remotely and safely. However, the users key to their wallet is stored in a plaintext XML file, if found by a malicious actor it would give them complete access to the wallet and all the money inside (The MITRE Corporation, 2019).

**CVE-2010-2913** - Citibank Citi Mobile app (version 2.0.3) stores all of the users information in an insecure file which can easily be accessed (The MITRE Corporation, 2010).

## 2 Recommendation

Code Review is a method of secure engineering that provides a very thorough examination of code to eliminate any potential faults. OWASP describe code review as the art of “understanding the code of the application, external components, and configurations to have a better chance of finding the flaws” in their 2017 Code Review guide (OWASP, 2017). This is done through scheduled meetings (bi-monthly, weekly or even more often) where the developers gather to intensely review code in order to find vulnerabilities and errors with their work. Often the group size is kept small (under ten participants) in order to keep the meeting efficient and avoid too many interfering hands. Another step to keeping meetings efficient is the presence of a Code Review Checklist which covers topics like Security, Performance, Scalability and much more (Gutha S.R, 2015). Issues like these are often missed or skipped over during development as they are seen as not being the priority, Code Review Checklists helps push these important areas back in the developers faces so that the proper attention is given to them.

This practise has the added benefit of speeding up patching time, as the developers know the code inside out, so can fix issues without the hassle of finding the problem code. Code reviewing is a timely exercises that cannot be automated, so often is pushed aside in favour for new developments. Figure 2 shows the decline in defects found as the workload increases for the hour time slot of Code Review, clearly showing that 100-200 lines of code is the sweet spot for efficiency and effectiveness (SmartBear, 2021). Although slow this practice is always worth the time invested, as it provides excellent vulnerability catching and unifies developers in the security of their codebase (Google Github, 2020).

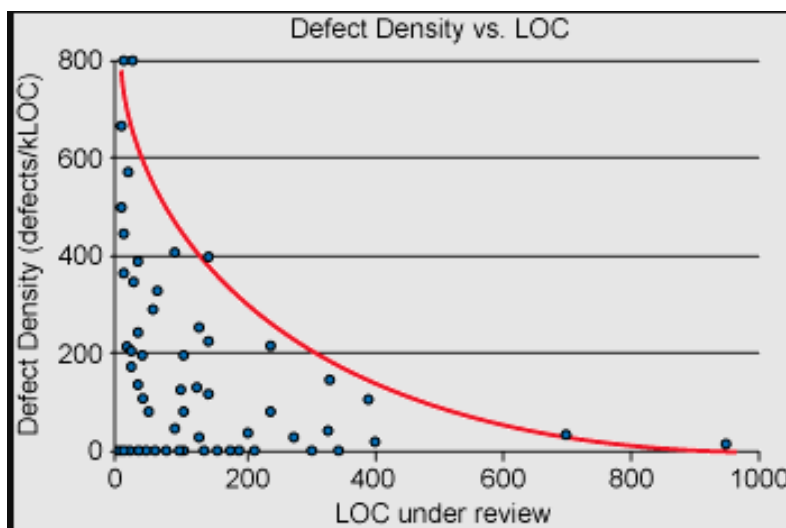
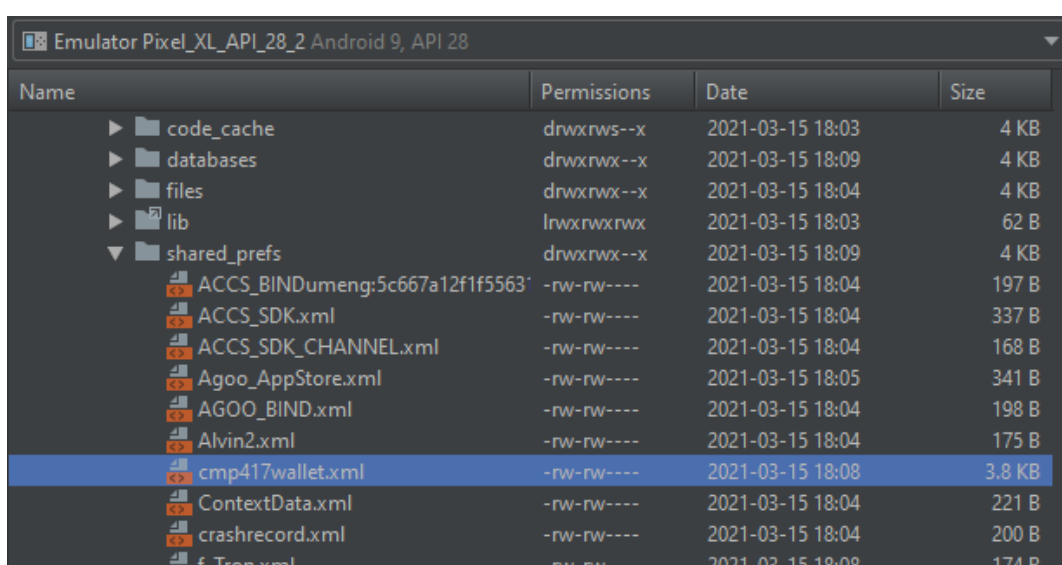


Figure 2: Graph showing the decline of effective Code Reviewing as Lines of Code increase.

For “Company Redacted” Code Reviewing seems like the perfect fit for their business type, having only six developers works very nicely with the meeting capacity, and this secure engineering practice is very cost effective at solving these potential vulnerabilities. With weekly meetings dedicated to systematically reviewing code and removing bugs, the threat from the Hackivist group is greatly mitigated, not just for the Mobile Application but for all aspects of their system.

### 3 Implementation

The Mobile Application was presumed to be used to authenticate users before allowing access to a remote database of staff information. In many apps like this a “Remember me” feature is included to allow users to quickly access the contents without signing in, this can take form as saved login credentials or a security token, both of which stored on the device memory. If these files are not properly encrypted (or even left in plaintext) than any malicious actor who finds the login details has full access to the database and its contents, with the power to change any information they want. If the team was to adopt Code Reviewing as a Secure Engineering Practice then they would definitely spot the Insecure data storage, as all Code Review Checklists contain a Security section, where sensitive information is checked to see if it is properly encrypted. With this simple practice the risk of identity fraud, data leakage and data loss are mitigated through proper management of sensitive information, leaving a much safer Mobile App.



Name	Permissions	Date	Size
code_cache	drwxrws--x	2021-03-15 18:03	4 KB
databases	drwxrwx--x	2021-03-15 18:09	4 KB
files	drwxrwx--x	2021-03-15 18:04	4 KB
lib	lrwxrwxrwx	2021-03-15 18:03	62 B
shared_prefs	drwxrwx--x	2021-03-15 18:09	4 KB
ACCS_BINDumeng:5c667a12f1f55631	-rw-rw----	2021-03-15 18:04	197 B
ACCS_SDK.xml	-rw-rw----	2021-03-15 18:04	337 B
ACCS_SDK_CHANNEL.xml	-rw-rw----	2021-03-15 18:04	168 B
Agoo_AppStore.xml	-rw-rw----	2021-03-15 18:05	341 B
AGOO_BIND.xml	-rw-rw----	2021-03-15 18:04	198 B
Alvin2.xml	-rw-rw----	2021-03-15 18:04	175 B
cmp417wallet.xml	-rw-rw----	2021-03-15 18:08	3.8 KB
ContextData.xml	-rw-rw----	2021-03-15 18:04	221 B
crashrecord.xml	-rw-rw----	2021-03-15 18:04	200 B
f_Tron.xml	-rw-rw----	2021-03-15 18:08	174 B

Figure 3: Finding the wallets XML file with the keystore inside.

Finding these files is rather easy when looking through the file system, to demonstrate this a vulnerability addressed in “Context” section was loaded onto Android Studios emulator (Android Developers, 2021). The vulnerability in question was **CVE-2019-13096** which related to the TronLink Wallet app version 2.2.0, where the key to the wallet was stored in a plaintext XML file. The TronLink app was installed on the emulator using an APK of the vulnerable version (APKPure, 2019) and a Wallet was created with the name “cmp417wallet” and a password of “WalletPassword123”. The device’s filesystem was then navigated through Android studio to `/data/data/com.tronlink.wallet/shared_prefs/` which contained the wallets XML file, as seen in Figure 3. The file was downloaded and opened with a browser, revealing the plaintext keystore, which can be used to gain unauthorized access to the Wallet, the full XML file can be found in the Appendix. The location of the XML file seemed to be a common place to store sensitive information when researching other CVEs related to this one, for a knowledgeable attacker this would be the first place they look. Code reviewing would likely find this fault very quickly, as the entire app is based around this file so would be the first to be put under inspection.

It is shocking that Insecure Data Storage in Mobile Applications is such a big issue in the first place, both iOS and Android offer security features that prevents this from happening. Apple offers a Keychain service for developers, where sensitive information can be stored securely and centrally, totally mitigating the risk from an attacker (Apple Developers, 2021). Android offers an excellent library allowing developers to easily encrypt any output files, with a two part system of encryption for added security (Android Developers, 2021), where each keyset for encrypted files are also encrypted under a primary key.

With the simple adoption of Code Reviewing and utilising the provided developer security features it is certain that the risk of Insecure Data Storage, and potentially many more vulnerability types, are completely mitigated, leaving a safe and secure Mobile app with all of the staffs sensitive information kept secret.



## 4 References

- Android Developers. 2021. Android Studio Download. [online] Available at: <https://developer.android.com/studio> [Accessed 14 March 2021]
- Android Developers. 2021. Work with data more securely. [online] Available at: <https://developer.android.com/topic/security/data> [Accessed 16 March 2021]
- APKPure. 2019. TronLink Download Versions. [online] Available at: <https://apkpure.com/tronlink-wallet-tron-blockchain-wallet/com.tronlink.wallet/versions> [Accessed 14 March 2021]
- Apple Developers. 2021. Keychain Services Overview. [online] Available at: [https://developer.apple.com/documentation/security/keychain\\_services](https://developer.apple.com/documentation/security/keychain_services) [Accessed 16 March 2021]
- Google Github. 2020. Google's Engineering Practices documentation: The Standard of Code Review. [online] Available at: <https://google.github.io/eng-practices/review/reviewer/standard.html> [Accessed 12 March 2021]
- Gutha S.R., 2015. Code Review Checklist – To Perform Effective Code Reviews. [online] Evoke Technologies. Available at: <https://www.evoketechnologies.com/blog/code-review-checklist-perform-effective-code-reviews/> [Accessed 12 March 2021]
- Kumar, M. 2020. New Android Flaw Affecting Over 1 billion Phones Let Attackers Hijack Apps. [online] The Hacker News. Available at: <https://thehackernews.com/2020/05/stranhogg-android-vulnerability.html> [Accessed 9 March 2021]
- The MITRE Corporation, 2020. CWE-922: Insecure Storage of Sensitive Information. [online] CWE. Available at: <https://cwe.mitre.org/data/definitions/922.html> [Accessed 9 March 2021]
- Mobliciti. 2020. MOBILE OS VULNERABILITIES: THE LURKING CULPRITS IN YOUR MOBILE FLEET. [online] Mobliciti. Available at: <https://mobliciti.com/mobile-os-vulnerabilities-mobile-fleet/> [Accessed 9 March 2021]
- OWASP. 2017. OWASP CODE REVIEW GUIDE 2.0. [online] OWASP. Available at: [https://owasp.org/www-pdf-archive/OWASP\\_Code\\_Review\\_Guide\\_v2.pdf](https://owasp.org/www-pdf-archive/OWASP_Code_Review_Guide_v2.pdf) [Accessed 12 March 2021]
- OWASP. 2016. OWASP Mobile Top 10. [online] OWASP. Available at: <https://owasp.org/www-project-mobile-top-10/> [Accessed 9 March 2021]
- Positive Technologies. 2020. Vulnerabilities and threats in mobile applications, 2019. [online] Positive Technologies. Available at:

<https://www.ptsecurity.com/ww-en/analytics/mobile-application-security-threats-and-vulnerabilities-2019/> [Accessed 9 March 2021]

Smart Bear. 2021. Best Practices for Code Review. [online] Smart Bear. Available at: <https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/> [Accessed 12 March 2021]

**CVE References:**

The MITRE Corporation, 2013. CVE-2013-6986. [online] CVE. Available at: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-6986> [Accessed 9 March 2021]

The MITRE Corporation, 2014. CVE-2014-0647. [online] CVE. Available at: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0647> [Accessed 9 March 2021]

The MITRE Corporation, 2019. CVE-2019-13096. [online] CVE. Available at: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-13096> [Accessed 9 March 2021]

The MITRE Corporation, 2010. CVE-2010-2913. [online] CVE. Available at: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2913> [Accessed 9 March 2021]

## 5 Appendix

```
<map>
<long name="create_time_key" value="0" />
<int name="mnemonic_length" value="0" />
<string name="wallet_address_key">TSrk6g6QHHnaneZfLUBfP7bSNPB26DZ8e2</string>
<long name="bandwidth_key" value="0" />
<string name="wallet_name_key">cmp417wallet</string>
<boolean name="is_cold_wallet_key" value="false" />
<string name="name_key" />
<string name="assets_v2_key">{}</string>
<long name="energy_used_key" value="0" />
<long name="net_free_limit_key" value="0" />
<long name="balance_key" value="0" />
<long name="wallet_createtime_key" value="1615831728174" />
<long name="freeze_bandwidth_key" value="0" />
<long name="energy_limit_key" value="0" />
<long name="total_energy_limit_key" value="0" />
<long name="latest_operation_time_key" value="0" />
<long name="freeze_energy_key" value="0" />
<string name="assets_key">{}</string>
<long name="total_energy_weight_key" value="0" />
<string name="wallet_keystore_key">
{"address": "41b9412f56821d56cd819725b2df6c5e3dec64013f",
"crypto": {"cipher": "aes-128-ctr",
"cipherparams": {"iv": "f91470e255c9905136b38b07dbe980cf"}},
"ciphertext": "d814d0d405925b0ff6eab317cc5d067294d2ae8e6d5d1453b17707197a88cab7",
"key": "kdf":
"script", "kdfparams": {"dklen": 32, "n": 65536, "p": 1, "r": 8,
"salt": "b7257ea19f5a9bc2c41a18548e83a3d3802215cd3031fe9c9acb6d784e1923de"},
"mac": "8e3a6058e1623ab4455826f860b2d47dd57f7ea99978c1c539c9d7f2f68c55df"},
"id": "7c81cbc0-cc54-46f9-a7dc-7556f64e4664", "version": 3}
</string>
<int name="wallet_color_key" value="-1" />
<string name="pwd_key">0205149ec6358ee0ece80a4293a274ea</string>
<long name="net_free_used_key" value="0" />
<int name="wallet_createtype_key" value="0" />
<long name="energytime_key" value="0" />
<long name="net_used_key" value="0" />
<long name="total_net_limit_key" value="0" />
<string name="frozen_key">{}</string>
<string name="wallet_icon_key">two</string>
<string name="pub_key">
04dd3abb4a01a9af7b03992904252c485a6b1bc3133b19df30
d4cb5248cacfd5d55819e8f8fcec59b075ff90f2add61ed902efb804c49399c683112f828ea303dd
</string>
<string name="votes_key">{}</string>
<long name="total_net_weight_key" value="0" />
```

```

<boolean name="set_hasaccount_key" value="true"/>
<boolean name="is_watch_only_setup_key" value="false"/>
<string name="address_key">3QJmh</string>
<long name="delegated_frozen_balance_for_bandwidth_key" value="0"/>
<long name="delegated_frozen_balance_for_energy_key" value="0"/>
<boolean name="backup_key" value="false"/>
<long name="net_limit_key" value="0"/>
<string name="wallet_newmnemonic_key">
{"address": "41b9412f56821d56cd819725b2df6c5e3dec64013f",
"crypto": {"cipher": "aes-128-ctr", "cipherparams":
{"iv": "f4356a4ac07080d697e09255d02df346"},
"cipertext": "06e2a0440dff936f0c0cf20fc31a7d97b2b8bc
6517fc44b18b73ae454a4c4c250d3c0b409fb19bff52b5af5e81
931fa562f9c23521f026a46fdc00cf417d87b2fa83429f2e170
f34ba2a6b52d2e77171cd", "kdf": "scrypt", "kdfparams"
: {"dklen": 81, "n": 65536, "p": 1, "r": 8, "salt": "0b7ec31db
312486656abbaf308ad335651d7007d5646ab450d410f6aa0396
811136faaf27792b3f3da5076819bcd212533ed5db720d6902f
437c45903e0ed00b18eba28b10fdd1301e7cc299359d45d679"}
, "mac": "ee3111f6b5b666e6ad4c8562923765e7f60f2fbefe89
1f1ef0321e5f113432b9"}, "id": "aff3cde0-6278-4764-a7f
5-43b712a77872", "version": 3}
</string>
<long name="energy_key" value="0"/>
</map>

```