TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Kirke Pralla 178134IVCM

CREATION OF FREELY ACCESSIBLE INTERACTIVE TRAINING MATERIALS FOR SECURE ANDROID DEVELOPMENT

Master’s thesis

Supervisor: Margus Ernits
MSc

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kirke Pralla 178134IVCM

VABALT KÄTTESAADAVA TURVALISELE ANDROIDI ARENDUSELE SUUNATUD INTERAKTIIVSE TREENINGMATERJALI LOOMINE

Magistritöö

Juhendaja: Margus Ernits
MSc

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kirke Pralla

22.04.2019

Abstract

Mobile market is constantly growing [1] and leaving mobile devices a reasonable target for malicious actors who aim to target vast amount of people. Growing market also means growing need for new products and applications. Unfortunately, mobile developers cannot keep up with the need causing shortcomings in quality of the products and applications [2] [3] [4].

This thesis aims to raise security awareness amongst software developers by giving them an opportunity to participate in interactive trainings. A system, containing database, web service and Android applications, is built as a result of this research. The system is needed so that the workshop can be built on top of a real system allowing creation of real-life-like scenarios.

In order to verify if the workshop described before is useful, an interactive workshop was held on 17th of April 2019. The workshop was rated to be useful by the participants, giving overall score 8.23 out of 10. 84.3% of the participants also stated that they would attend more similar workshops if given the chance.

This thesis is written in English and is 73 pages long, including 7 chapters, 19 figures and 1 table.

Annotatsioon

Vabalt kätte saadava turvalisele Androidi arendusele suunatud interaktiivse treeningmaterjali loomine

Pidevalt kasvav mobiiliturg [1] muudab mobiilsed seadmed sihtmärgiks kuritahtlike kavatsustega inimestele, kelle eesmärgiks on mõjutada võimalikult suurt hulka inimesi. Kasvav turg suurendab ka vajadust uute toodete ja rakenduste järgi. Kahjuks ei jõua toodete ja rakenduste arendajad seda vajadust täita, mis põhjustab aina halveneva kvaliteedi [2] [3] [4].

Antud lõputöö eesmärk on suurendada tarkvara arendajate teadlikkust turvaküsimustes. Teadlikkust proovitakse suurendada läbi interaktiivsete koolituste ja töötubade. Nende tarbeks on ehitatud terve süsteem – andmebaas, veebiteenus ning Androidi rakendused. Tervet süsteemi on vaja selleks, et oleks võimalik luua võimalikult tõsiseltvõetavaid stsenaariumeid. Eesmärgiks on treeningud teha võimalikult päris elu sarnaseks, et osalejad ei näeks treeningut kõigest õpetussõnadena vaid seoksid seda päriseluga.

Valideerimiseks viidi 17. aprillil läbi interaktiivne töötuba, kus osalesid tarkvaraarendajad, turvaspetsialist ning paar tarkvara kvaliteediga tegelevat inimest. Töötuba oli võrdlemisi edukas. Osalejad hindasid töötoa kasulikkust 8.23 punktiga kümnest. 84.3% osalejatest väitsid ka, et sooviksid osaleda veel sarnastes töötubades kui neil oleks selleks võimalus. Lisaks eelnevale ütlesid osalejad ka, et koolitus pani neid turvaprobleeme teisiti nägema ning neile rohkem mõtlema ja tähelepanu pöörama.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 73 leheküljel, 7 peatükki, 19 joonist, 1 tabelit.

List of abbreviations and terms

ADB	Android debug bridge. ADB gives an opportunity to access Android device's data from another device (such as computer) and give orders to a mobile device from another device.
apk	File extension. Android compiled applications that can be run directly on an Android device.
CRUD	Create-read-update-delete
HTTP	Hypertext transfer protocol
HTTPS	Hypertext transfer protocol secure
Google Play	Official application store for Android. Provides possibility of uploading and downloading applications that are approved by Google.
API	Application programming interface
URL	Uniform resource locator
SQL	Structured query language
CEO	Chief executive officer
IT	Information technology
CPU	Central processing unit

Table of contents

1 Introduction 12
2 Related research 14
2.1 Hardware security 14
2.2 Operating system security 14
2.3 User-related security issues 16
2.4 Backend security 16
2.5 Environment security 17
2.6 Application security 17
2.7 Related solutions 18
2.7.1 Vulnerable applications 18
2.7.2 Blog posts and tutorials 19
2.7.3 Conferences and courses 20
2.7.4 Code analysis tools 20
2.8 Related research summary 21
3 System analysis 22
3.1 Vulnerabilities to be used 22
3.1.1 Stealing files through enabled backup 22
3.1.2 Getting information from broadcasted user's data 23
3.1.3 Hardcoded API key in Android application 25
3.1.4 Bypassing authentication by calling exported activity 26
3.1.5 Allow user to perform actions that (s)he does not have rights to do 27
3.1.6 Attack through unnecessary permissions granted by the user 28
3.1.7 Stealing sensitive information from screenshots 28
3.1.8 Allowing too weak passwords 29
3.1.9 Poor encryption in password storage 29
3.1.10 Storing sensitive data in phone's external storage 30
3.2 System requirements 31
3.3 Main application design 34
3.3.1 Log in screen and authorization 34
3.3.2 Notifications screen 35
3.3.3 Project accounting screen 35

3.3.4 Review page 37
3.3.5 Employees' page 38
3.4 Malicious mobile application design...39
3.5 Database design..40
3.6 Web service design...43
4 Technical description 47
4.1 Main application architecture...47
4.2 Main application components 47
4.2.1 View 47
4.2.2 Presenter..48
4.2.3 Model 48
4.2.4 Communication with web service...49
4.2.5 Session management 49
4.3 Malicious application 49
4.4 Backend...50
4.4.1 Communication with application 50
4.4.2 GET requests...50
4.4.3 POST requests...51
4.4.4 Communication with database 51
4.5 Database 51
4.6 Setting up the environment 51
4.7 Adding vulnerabilities...52
5 Validation..53
5.1 Scenarios used...53
5.2 Results...58
6 Future 60
7 Summary 61
References...63
Appendix 1 – Experts opinion on the topic...74
Appendix 2 – Sentences used in MySQL database creation and updates.......................77
Appendix 3 – Example web service requests and responses 82
Appendix 4. Feedback questionnaire 89
Appendix 5. Local backend setup 90

Appendix 6. Training feedback from Head of Cyber Security of a software development company 91

List of figures

Figure 1. Different Android versions used [11] 15
Figure 2. Screenshot of SESSION_INFO.xml file retrieved from a backup 23
Figure 3. Decompiled application logs showing information about broadcast 24
Figure 4. Aapt output for exportable activity 26
Figure 5. Log in screen 34
Figure 6. Send new message screen 35
Figure 7. Notifications screen 35
Figure 8. Project accounting screen 36
Figure 9. Report working hours screen 36
Figure 10. Previously reported hours screen 37
Figure 11. Request vacation screen 37
Figure 12. New tool request review field 38
Figure 13. Working hours review field 38
Figure 14. Vacation request review field 38
Figure 15. Review page 38
Figure 16. Employee details page 39
Figure 17. All employees page 39
Figure 18. Database model 43
Figure 19. Usefulness ratings given by participants of the workshop 59

List of tables

Table 1. Hardware manufacturers Android system customizations................................15

1 Introduction

Growing mobile market causes an extensive lack of mobile developers [1]. Therefore, leading to mobile application being created by amateur software developers who make mistakes due to their inexperience in the field [2] [3] [4].
Second concern related to lack of developers is a lack of time. Professional developers do not have time to focus on quality and security as they are forced on providing as much functionality as fast as possible [4].
Third problem arises with documentation and articles the developers use while writing applications. Multiple studies have shown the impact of using unreviewed sources of information such as Stack Overflow [3] [5].
Finally, according to author's knowledge, there is deficiency in freely accessible hands-on materials. Therefore, developers have very limited options for train themselves in Android security. Multiple purposely vulnerable application can be found but most of them are not up to date. Applications that are up to date are not developed for training software developers. The materials are targeted to penetration testers. This matter is discussed in related research chapter.
The previous four sections were used in homework of Seminar course (ITC8100). The sections are reworded.
The goal of this research is to create a publicly accessible training material that would address security-related development flaws created by app developers. The project can be downloaded and customized by any interested parties. Also, there shall be a possibility to update the current solution by adding more vulnerabilities. The solution contains of backend and application part. The application part of the project contains two application – one that will have secure implementation of topics covered and another one demonstrating the flawed way. Therefore, the vulnerable version can be used to conduct challenges, hands-on training programs or security bug demonstrations in order to raise awareness.
Result of this thesis will be tested during a few-hour long workshop. The author of the thesis will firstly introduce the application used in the workshop. The participants are acting as a security team solving incidents by analyzing the code and discussing possible

causes of the incident. The author of the thesis will ask questions and give hints if needed in order to help solving the incident successfully. After the training session, the participants are expected to fill in the feedback form saying if they learned something and if they got more interested in learning more of security problems.
The author of the thesis has worked as an Android developer for a bit more than 2 years. During this time, she has seen many security problems caused by small programming errors and lack of security knowledge. She even saw one security attack against a product she was developing. As she has seen the field and market, it is a personal interest to raise awareness and reduce the number of security-related bugs in Android applications.
The thesis contains of three five chapters: related research, system analysis, technical description, validation and future work.
Related research chapter introduces the field – not only application-related security but also other Android security problems such as hardware, operation system and backend security. The introduction is needed in order to define the scope.
System analysis chapter introduces the vulnerabilities developed used in first iteration, system requirements in order to develop previously mentioned vulnerabilities and system design. System, containing database, web service and applications, is needed in order to provide hands-on experience to the developers.
Technical description chapter introduces the technical solution of the system. It will cover application structure, components and architecture, web service endpoints, technical details of the database. Also, it explains how to add vulnerabilities and customize the system to whomever needs.
Validation chapter contains information about testing the result of this research in real life amongst real developers. It introduces the way the course was developed; what kind of scenarios were used and what was the feedback from the developers participating the course.

2 Related research

The scope of this research is android application security as the problem to solve is related to the mistakes Android developers make while developing the application. In order to define Android security, it is grouped into four categories in this research: hardware security, operation system security, backend security and application security. Defining Android security is crucial in order to clarify the scope.

All aspects of Android security are of great importance to an Android developer and should be considered during development process. For example, if there is a known vulnerability in hardware security, a developer should think carefully about it in order to minimize that vulnerability's effect on the application.

2.1 Hardware security

Hardware security issues are issues related to the device itself – the physical pieces of the device. For example, microchips and processors. There are two bigger groups of hardware security issues in Android.

Firstly, as a lot of companies develop hardware for Android devices, the development is chaotic and not properly standardized. Therefore, the market has devices with very low hardware quality and devices with very high quality [6].

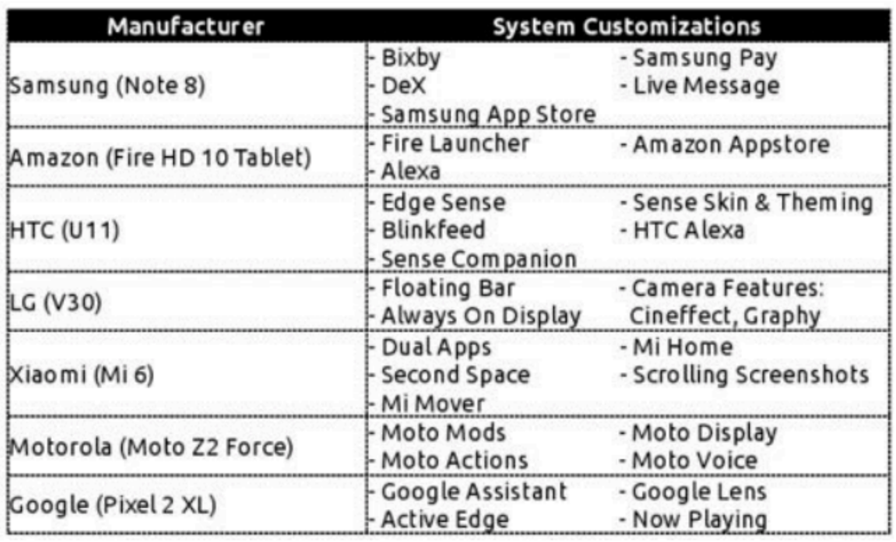
Secondly, there are a lot of different Android devices in the market in different price ranges. Some of those devices are not built for upgrades [7] [8]. Meaning that it is impossible to upgrade to operating system on that device, leaving the device vulnerable without any solutions for all old OS-related security issues.

2.2 Operating system security

Operating system security issue group contains issues related to Android operating system that runs Android applications. Such as Android updates, security patches, illegal communication between applications installed on the device.

Android has 87,9% of the mobile market share [9]. Therefore, it is a great target for attackers aiming to affect as many people as possible. As Android is an open source platform for mobile devices, each manufacturer can develop their own variation of the operating system. Therefore, a lot of different variations of Google-released Android exist in the market. Examples of bigger hardware manufacturers' customizations of operating systems are displayed in the Table 1 [10].

Table 1. Hardware manufacturers Android system customizations [10]



Manufacturer	System Customizations	
Samsung (Note 8)	- Bixby - DeX - Samsung App Store	- Samsung Pay - Live Message
Amazon (Fire HD 10 Tablet)	- Fire Launcher - Alexa	- Amazon Appstore
HTC (U11)	- Edge Sense - Blinkfeed - Sense Companion	- Sense Skin & Theming - HTC Alexa
LG (V30)	- Floating Bar - Always On Display	- Camera Features: Cineffect, Graphy
Xiaomi (Mi 6)	- Dual Apps - Second Space - Mi Mover	- Mi Home - Scrolling Screenshots
Motorola (Moto Z2 Force)	- Moto Mods - Moto Actions	- Moto Display - Moto Voice
Google (Pixel 2 XL)	- Google Assistant - Active Edge	- Google Lens - Now Playing

In the end of 2018 Google analyzed active Android versions being used [11]. The results are displayed on Figure 1 [11] illustrating the wide usage of many different Google-released platform versions.

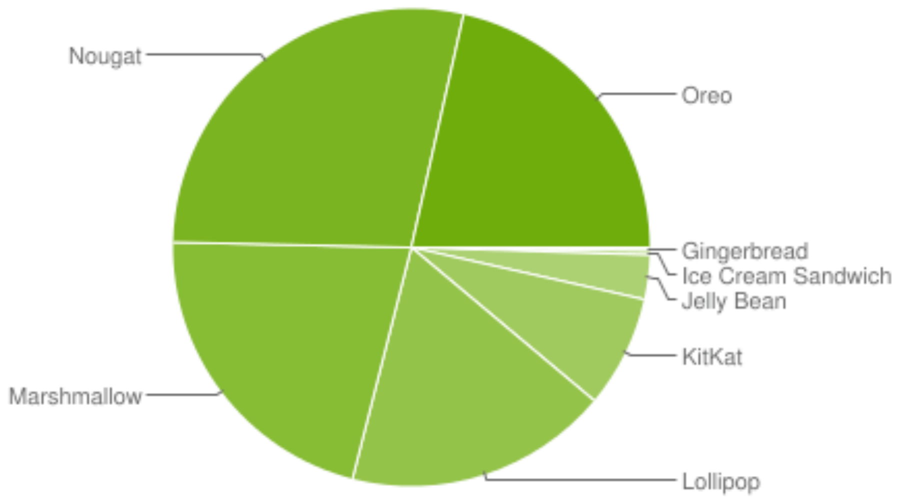


Figure 1. Different Android versions used [11]

Secondly, even though Android operating system has a built-in security in architecture, it is still containing bugs that are exploited by hackers. Those bugs enable attacks such as privilege escalation, memory corruption, spyware and denial of service [12] [13]
Thirdly, Android vendors are not making updates available to users on time [14]. A German security firm discovered that some Android vendors delay security-related updates for months and meanwhile claim the phone's firmware is up to date leaving users vulnerable to already patched security flaws. [15] [16]

2.3 User-related security issues

Most of user-related security issues are caused by lack of knowledge and being in a comfort zone. Updates may take some time and a user cannot use the device meanwhile. Information security company Duo Labs [17] investigated the issue and found out that 25% of the people who receive regular security patches update their device [18]. Analysis done by Google showed that less than 0,1% of Android users have upgraded their devices to use the new Android within 3-month period. [11]
Most of Android users are no security experts. Furthermore, they do not know much about Android permissions. When installing an application, an alert box is shown to users to request permissions the application claims to need in order to perform all needed actions. According to research made in Prince Sultan University 80,3% of the most rated Android applications in Google Play request unnecessary permissions [19]. Mostly users allow all applications to use whatever permissions needed so that the app could be used without knowing what they agreed upon [2].

2.4 Backend security

Backend security group often contains problems related to databases and web services the app communicates with. A research showed that 40% of 10 000 of the most popular applications have problematic logic that might lead to security issues [20]. The same research showed that ~17% of those apps were using HTTP protocol and ~10% were vulnerable to hijacking vulnerabilities.
Researchers studied public APIs and discovered that the most popular vulnerabilities were Cross Site Scripting, Injection Flaws, information leakage and improper error handling, broken authentication and session management, insecure cryptographic storage,

insecure communication and failure to restrict URL access. [21] Some of the causes for those vulnerabilities might be improper input validation, logging too much information or showing too much information in error messages, trusting the user is who he is claiming to be, not storing keys and secrets properly, using HTTP instead of HTTPS and leaving logic implementation to client side.

2.5 Environment security

Environment security group contains security issues related to other applications such as what happens if malware attacks the device.

The major cause of this problem is again Android popularity. In December 2018 Google Play had 2,6 million apps available for download [22]. In order to upload an app to Google Play, a user must have developers account that anyone can create. After creation of an account, user should upload the app for review. If the review response is positive, the app is uploaded to Google Play. Due to Android popularity, there is a very limited amount of time the reviewers can spend on one application. Therefore, it is quite easy to get the approval even if the application is malicious [23] [24] [25] [26]. Furthermore, Android is a great and interesting target to malware developers being the operating system with biggest market share [9].

In addition to malicious actors, flawed external libraries used by developers may lead to serious security issues. In 2016, researchers in Saarland University discovered that a flaw in cryptographic APIs would affect 296 most popular applications and 3,7 billion devices [27].

2.6 Application security

Application security group contains the mistakes made by the developer of the application. Such as cache problems, too many permissions, too much logging, code quality, input validation. This group of security issues have many reasons.

Firstly, a big amount of Android applications are developed by amateur developers who make security mistakes due to lack of knowledge and experience [2] [3] [4].

Secondly, as the mobile market is growing so rapidly, leaving experienced developers focused only implementing functionality instead of thinking about security [2] [3]. Thirdly, the developers tend to use unofficial materials while solving issues during development or learning about new technologies [5]. Those materials are often insecure or have a big potential to be insecure when copy-pasting it directly to the application in development. Researchers from University of Maryland investigated the impact of information sources on code security [3]. The research showed that only 25% of accessed materials from Stack Overflow [28] were useful and only 17% of them contained secure code snippets. Another research done in Saarland University analyzing the impact copy-pasting snippets from Stack Overflow [5]. The research analyzed 1,3 million apps and found out that 15,4% of the apps used security-related code snippets from Stack Overflow. 97,9% of the 15,4% of the applications used insecure snippets. The results indicate that a huge amount of security-related code-snippets found from internet are insecure as Stack Overflow is one of the most important sources used by developers [5] [3].

Thirdly, according to author's knowledge, currently there are no freely accessible applications, frameworks nor labs to learn about secure Android application development by the November of 2018. Freely accessible means that the learning opportunity is accessible to anyone who wants to access it and does not cost anything. Existing solutions will be addressed in next paragraph.

2.7 Related solutions

2.7.1 Vulnerable applications

Many vulnerable Android applications can be found in the internet. "Vulnerable" in this context means purposely vulnerable applications build for exploiting the vulnerabilities. The search was done in November 2018 by using keywords "Android Vulnerable Application" in Google. Most of them are not up to date and seem to be abandoned. For example, Damn Insecure and Vulnerable App for Android [29] was updated 3 years ago, Android Digital Bank [30] was last updated 4 years ago, OWASP GoatDroid [31] was updated 6 years ago and Android Security Labs [32] 8 years ago.

There are also few that are more or less up to date. The main difference with author's solution is that none of the solutions that were found, did not focus on secure development. All of them focused on penetration testing.

Android Insecure bank [33] has a lot of different vulnerabilities covered. It also has guides for successfully exploiting vulnerabilities. This project is great to study and practice penetration testing, but it is not meant for improving knowledge on how to develop applications securely. The walkthrough shows how to exploit the app itself not why and how the code makes the application vulnerable. Also, no examples are provided.

Security Shepherd [34] is a project developed by OWASP. It is designed for penetration testing training. It has many small pieces that can be used in separate demos or exercises. It does not have a real-application feel and it does not have any tutorials about secure development as its target is not secure development.

UnCrackable Mobile Apps [35] is another tool developed by OWASP. It is targeted to reverse engineering training. It contains compiled Android application and no source code. Therefore, it is not suitable for secure development training.

Damn Vulnerable Hybrid Mobile App (DVHMA) [36] is a tool developed for security training for hybrid mobile app developers. Hybrid mobile applications use different technologies than native mobile applications. Therefore, it is also not usable for secure native development training.

Purposefully Insecure and Vulnerable Android Application (PIVAA) [37] is a tool developed for demonstrating vulnerabilities in native applications. The project contains precompiled application and source code. The project does not contain any tutorials or tips for secure development. It only provides short few-line introduction to vulnerabilities included in the project.

Vulnerable Android App Oracle [38] is a tool that is seemingly created for demonstration purposes in some presentations. It does provide instructions on how to find vulnerabilities and fix them. According to author's knowledge, it does not have hands-on training built around it and it consists of small parts of applications like Security Shepherd.

2.7.2 Blog posts and tutorials

In addition to vulnerable application, there are also numerous blog posts and tutorials on how to test Android application security. For example, there are posts showing how to exploit vulnerable applications [39]. This lab contains two exercises. Both are relatively short and not very informative. There are also some tutorials that are quite informative, like Android Hacking Event 2017 tutorial [40]. This one has a lot of content, but it is not practical, and its scope is more low level – crypto and tokens.

Also, articles can be found on how to improve Android security. A lot of articles are solving one specific problem. For example, "Hands on mobile API security: Get rid of

client secrets" [41]. The articles are very useful if a developer is facing a specific problem that he needs to solve.
The best tutorials are Android's best practices blog post [42], "Application security checklist" [43] and OWASP Android security testing guide [44]. All those materials contain coding examples and theory on Android security. Unfortunately, they do not provide hands-on experience.

2.7.3 Conferences and courses

Different courses are provided by multiple firms and organizations. Unfortunately, those courses do not solve the problem under advisement in this research.
Firstly, none of those courses are dedicated to secure development. The scope is more on how to hack and test application security.
Secondly, none of those courses are free of charge. For example, Advanced Mobile Exploitation course by Attify Store costs 699$ [45], Mobile application Security and Penetration Testing Course costs 899-1299 [46] per person. Some companies market their courses with no price tags on them. For example, Android Mobile Application Hacking course by Appsec Labs [47].
In addition to courses, many different conferences are available for learning about Android security. Troopers 2019 conference in Germany that has a training about mobile penetration testing class [48]. NullCon 2019 in India also has mobile penetration testing class [49]. AppSec California 2019 conference has also mobile security course available [50]. Previously mentioned courses cost 545-2080€ per person [51] [52].
Unfortunately, those conferences also do not solve the problem addressed by this research. The conferences are not easy to access – they are costly and in different parts of the world.

2.7.4 Code analysis tools

There are various code analysis tools that are built to help developer avoid bugs while developing the application. They are quite convenient – allowing developer to see security issues as regular compiler errors and therefore, recognizing security problems early on. For example, for Android – Android Lint [53], CodeDX [54], Findbugs [55], Veracode [56]. Unfortunately, they are not efficient enough and cannot detect even all known vulnerabilities. A study shows that it recognizes up to 80% of input validation vulnerabilities [57]. Another study tested out free code analysis tools. The results showed that currently available free analysis tools are not able to detect many known vulnerabilities. All the tools tested together could only detect approximately 71,4% of the known vulnerabilities [58].

2.8 Related research summary

Numerous Android security practical trainings exist. Unfortunately, they are costly or not targeted to software developers. Also, other solutions can be found in the internet: code analysis tools, blog posts and tutorials. Code analysis tools are not 100% effective and is not able to detect all possible vulnerabilities. Furthermore, code analysis tools cannot see the whole system for example how the user might use the application and what kind of systems the application is connected to. Blog posts and tutorials does not provide creating interactive challenges nor hands-on training that would help involve more participants and make the training more interesting and fun for trainees. It also does not support creating scenarios and therefore, creating real-life-like situations for the trainees would have to solve.

Paragraph 2.6 discusses severity of the problem in Android application security showing that a solution is needed. Paragraph 2.7 shows different solution the author thinks may try to solve this problem but none of them have solved it.

As per author knowledge, there are no freely accessible training material that is targeted to software developers and could be used in order to involve trainees in the training by requesting them to solve a security incident by themselves or in a group. Therefore, a solution provided in this thesis is novel.

3 System analysis

The author will create freely accessible training material that can be used to develop hands-on training materials, challenges or security bug demonstrations. The training material is created in attempt to solve the deficit of freely accessible Android secure development training material.
In order to find out which mistakes to cover, different sources are used. The main sources will be OWASP materials: “OWASP Mobile Security Testing Guide” [44] and “OWASP Mobile Security issues top 10” [59]. According to research done by mobile app testing company NowSecure [60], 85% of App Store apps fail in at least one point of OWASP top 10 [61].
In addition, real vulnerabilities found in live applications are used. The real use cases were taken from vulnerability reports from HackerOne [62] and some scenarios are created by the author of this paper.

3.1 Vulnerabilities to be used

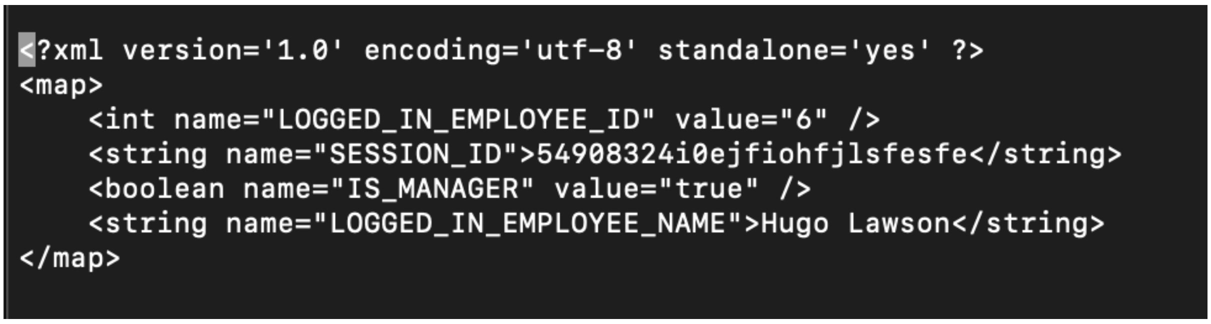
3.1.1 Stealing files through enabled backup

ADB (Android Debug Bridge) is a tool that allows user to back up the data and application state. It contains keys, files, other data and application itself [63]. The backup can be saved in device’s local storage. And therefore, it can be stolen resulting to possible sensitive information leakage. Allow backup flag is true by default so in order to disable it, a developer must set it false. It is still true if the flag is missing from the manifest [64]. Backing up sensitive data is also considered as a threat by OWASP [65]. Furthermore, in 2017 the same vulnerability was found in Boozt Fashinon AB’s mobile application and was reported on HackerOne [66].
Vulnerable application should have ADB backup enabled. The steps how to get sensitive information from ADB backup are taken from InfoSec institute blog [67].
Firstly, the device must allow USB debugging in order to create a backup via Android debug bridge. The backup can be encrypted with a password, but it can be left decrypted. The backup may contain sensitive data. For example, if an application uses shared preferences for keeping session information, it can be retrieved from that backup.

The backup file's header must be removed in order to unzip the rest of the file. Following commands can be used for removing the header and unzipping the file:

```
dd if=<backup_name>.ab bs=1 skip=24 | python -c "import
zlib,sys;sys.stdout.write(zlib.decompress(sys.stdin.read()))" >
<backup_container_name>.tar
tar -xf <backup_container_name>.tar
```

The extracted container contains "apps" folder. A SESSION_INFO.xml file can be found inside the container showing session data of user who was logged in while the backup was created. Figure 2 shows the information retrieved from session info file.



```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="LOGGED_IN_EMPLOYEE_ID" value="6" />
    <string name="SESSION_ID">54908324i0ejfiohfjlsfesfe</string>
    <boolean name="IS_MANAGER" value="true" />
    <string name="LOGGED_IN_EMPLOYEE_NAME">Hugo Lawson</string>
</map>
```

Figure 2. Screenshot of SESSION_INFO.xml file retrieved from a backup

Secure application should have ADB backup disabled.

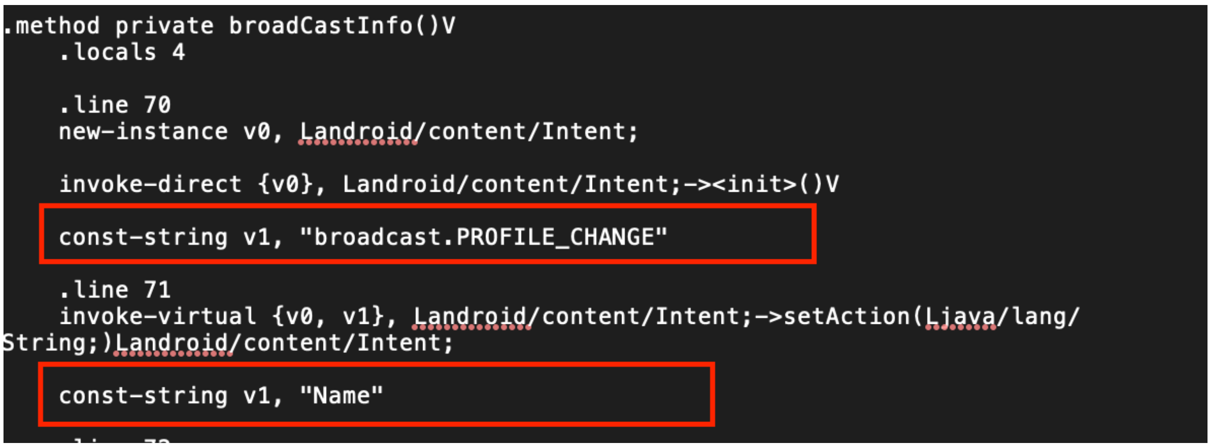
3.1.2 Getting information from broadcasted user's data

Broadcast is a feature that allows sending data from one application to another [68]. In case of not being careful with what kind of data to whom to give, third party apps can listen to broadcasted conversation. In 2017, Twitter broadcasted user's location to all the applications installed in the device, causing a potentially serious sensitive information leak [69].

Vulnerable application should use global broadcast to notify other application when user data changes. Sending global broadcast from main application can be done as follows:

```
Intent intent = new Intent ("
      kirke.app_sec_main_application.broadcast.PROFILE_CHANGE");
intent.putExtra("Name", employeeName.getText());
getActivity().sendBroadcast(intent);
```

Figure 3 shows how all needed information is displayed in decompiled application.



```
.method private broadCastInfo()V
    .locals 4

    .line 70
    new-instance v0, Landroid/content/Intent;

    invoke-direct {v0}, Landroid/content/Intent;-><init>()V

    const-string v1, "broadcast.PROFILE_CHANGE"

    .line 71
    invoke-virtual {v0, v1}, Landroid/content/Intent;->setAction(Ljava/lang/
String;)Landroid/content/Intent;

    const-string v1, "Name"
```

Figure 3. Decompiled application logs showing information about broadcast

Attacker can easily get all the information needed in order to start listening to broadcast messages the application sends out – broadcast intent name and what kind of data is inside the broadcast. The attacker has to define BroadcastReceiver and register when the malicious app starts as shown below.

```
public class MyBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String name = intent.getExtras().get(“Name”);
// do something with the data
    }
}
IntentFilter filter = new IntentFilter(
"kirke.app_sec_main_application.broadcast.PROFILE_CHANGE");
        BroadcastReceiver receiver = new MyBroadcastReceiver();
        registerReceiver(receiver, filter);
```

First of all, the secure application should define to whom the broadcast is meant for. It can be done by adding component to an intent:

```
Intent intent = new Intent();
intent.setAction("broadcast.PROFILE_CHANGE");
intent.putExtra("Name", employeeName.getText());
intent.setComponent(new ComponentName(
      "<package_of_the_app_to_whom_the_message_is_meant_for>",
            "<receiver_class_that_should_receive_the_message"));
getActivity().sendBroadcast(intent);
```

This approach is more secure than without defining the receiver, but it is still not secure enough because receiver data can be accessed from decompiled apk file and therefore, a malicious application can claim to be the legitimate receiver. Therefore, there is a need for setting custom permissions to the broadcast message so that only applications signed with developer's certificate can listen to broadcast messages. In order to define permission in application, a developer has to add it to manifest file and set application to use this permission:

```
<permission android:name="<permission_name>"
android:protectionLevel="signature"/>
```

```
<uses-permission android:name="<permission_name>" />
```

In order to set broadcast to use this permission, it has to be added as a parameter when sending broadcast:

```
getActivity().sendBroadcast(intent, "<permission_name>");
```

Using this approach ensures that malicious applications cannot listen on broadcast messages as they are not signed with the developer's key.

3.1.3 Hardcoded API key in Android application

An Indian security company Fallible [70] studied 16 000 applications from Google Play and found out that 2 804 applications have third-party credentials exposed [71].

Plain text API keys are easily accessible after decompiling apk file. Decompiling can be done by numerous freely accessible tools for example apkTool [72]. Decompiled application does not look exactly like it did before compiling but it is still possible to access strings.

For vulnerable application, the API key is stored in ApiClient.java class as private final static string.

```
private static final API_KEY = "jfgyu5g354hui3bfu322ih3h"
```

After compiling the application in release mode and decompiling it with apktool, the key is available in ApiClient.smail file.

```
.field private static final API_KEY:Ljava/lang/String; =
"jfgyu5g354hui3bfu322ih3h"
```

In case of public repository, it is even easier to get the API key stored in the way described before – it can be accessed directly from the repository.

For secure version, the API key is stored in a separate file in <key_name>=<key> format. The file can be uploaded to the repository but only with fake contents. It might be useful in case of continuous integration. The real content of the file can never be uploaded to repository. If it does get uploaded, any trace of the keys should be deleted as soon as possible.

Keyfile should be defined in app's gradle file by adding `def <key_file_variable_name> = <file_location>.file("<file_name>")` in the beginning of gradle file. A developer can define a resource name for the key inside of buildTypes tags in gradle file. Different values can be assigned to release and debug builds allowing to use different API keys for both. Defining the resource should be done by adding as follows: `resValue 'string', '<key_name_for_project>, <key_file_variable_name>[<key_name]`

The key is accessible from calling string value `R.string.<key_name_for_project>`. Therefore, no string variables are needed inside the project.

3.1.4 Bypassing authentication by calling exported activity

Exported activities are listed as a common weakness in Common Weakness Enumeration page [73]. Exported activities are also considered serious problem in a book called "Hacking Android" [74].

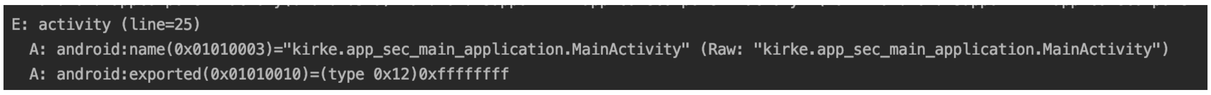
Authentication can be bypassed by start MainActivity from application that is installed to the same device as the main application. MainActivity is the activity started after login process is complete.

It is possible if activity is exportable from main application. Making activity exportable means adding one line of code to android manifest file – `android:exported=true`. Another way of making activity exportable is by adding <intent-filter> tags inside of activity tags. It is quite dangerous because there is no suggestion to making application exportable.

Exported activities are easily detectable from Android application package (apk) file. Apk file can be analysed with tool called aapt that is included in Android SDK. Determining exportable activity can be done with one command that will show the contents of Android manifest file:

```
/<path_to_android_sdk_build_tools>/<build_tools_version>/aapt dump
xmltree /<path_to_sdk> AndroidManifest.xml
```

Figure 4 shows how exported activity is shown in aapt output.



```
E: activity (line=25)
  A: android:name(0x01010003)="kirke.app_sec_main_application.MainActivity" (Raw: "kirke.app_sec_main_application.MainActivity")
  A: android:exported(0x01010010)=(type 0x12)0xffffffff
```

Figure 4. Aapt output for exportable activity

From the output displayed earlier, an attacker can get all the information needed: which activities are exportable and how to call them from any other application.

In order to open the activity, another app is needed. The new application has to have only one activity as it needs to be executable. Only few lines of code need to be added to the activity to be called when the activity is created:

```
Intent intent = new Intent();
Intent.setComponent(new ComponentName(“kirke.app_sec_main_application”
,
      ”kirke.app_sec_main_application.MainActivity”));
startActivity(intent);
```

Those lines tell the activity to start an activity from another app when the app is executed. Therefore, MainActivity of the main application is opened without having to log in.

3.1.5 Allow user to perform actions that (s)he does not have rights to do

An application must save user log in information so that user should not have to log in every time (s)he opens the application after the application is closed. As user experience and comfort are very important nowadays, single sign-on is used in many applications.
Vulnerable application should cache user information (e.g. cookies) and not properly clean it when logging out so that the next user who logs in, can see the previous user's information and has its rights.
Vulnerable application should not properly clean caches when user logs out. For example, cleans only logged in user id but does not reset isManager flag. When the next person who logs in is not a manager. The vulnerable application is setting isManager flag only if a manager logs in because it assumes isManager flag to be false by default. Therefore, if a regular user logs in after a manager, (s)he is logged in as a manager.
Worst case would be not clearing any logged in user information from caches because it is assumed to be overwritten when a new user logs in. Furthermore, device's back button allows users to go back to the previous page. If a person logs in and out on a device, the session can be restored by just clicking back button on the device. Resulting random user being logged in as a previous user. Even if previous user was not a manager, a random user can access sensitive data (s)he should not have access to.
Just clearing the caches does not solve the problem. If an application is properly clearing all the caches it might still have a vulnerability when a user clicks back button after logging out. In order to avoid that bug, intent has to be defined with special flags FLAG_ACTIVITY_CLEAR_TASK and FLAG_ACTIVITY_NEW_TASK. FLAG_ACTIVITY_CLEAR_TASK will clear all tasks associated with the activity to be

cleared and any old activities are cleared [75]. FLAG_ACTIVITY_NEW_TASK will set the activity defined in the intent to be the start of a history stack [76].

3.1.6 Attack through unnecessary permissions granted by the user

Each interaction with device needs permission [77]. Which means the application must ask the user to grant the permissions. When a third-party library is used in application, it might also need some permissions. In that case, a developer should look them through and make sure that there is nothing suspicious and unnecessary there. For example, a parser application that should be used for parsing strings to date and vice versa, should not request READ_EXTERNAL_STORGAGE permission. The impact of unnecessary permissions is illustrated in the section "stealing sensitive information from screenshots". Unnecessary permissions are also considered to be an issue and brought out in OWASP Mobile Security Testing Guide [78].

3.1.7 Stealing sensitive information from screenshots

Screenshots are taken in order to share a screen image with whomever – just one person, friends list in social media or every person who comes across it. Furthermore, malicious applications can take screenshots without user knowing [79]. Every screen in mobile application is not for sharing as it might contain sensitive information. Such as log in screens containing filled in username and password fields. Even if password characters are replaced with any other icons, the length of a password can still be read from it. Furthermore, screenshots are accessible by other possibly malicious applications installed in the same device by just asking READ_EXTERNAL_STORAGE permission from user.

This vulnerability is making it easier to malicious actors to brute-force the application if a filled log in screen can be captured and the length of the password is shown in the image. For example, using a combinations formula it can be calculated how many combinations it takes to guess a password without knowing the length and compare it with the number of combinations needed if the length is known.

$$C = \sum_{i=1}^{r} \frac{x!}{r!\,(x-r)!}$$

In the formula, C represents the number of combinations needed in order to guess a password, r is the length of a password and x is number of letters in the Latin alphabet – 26 in the example. In the example, it is assumed that passwords contain only letters and the password length is 12 characters. In order to try every possible 1-12-character password, one must try 28 354 131 passwords. If the attacker knows the password length is 12 characters, (s)he must try 9 657 700 characters. That is ~65,9% less meaning faster computation.

As the level of security awareness is relatively low [80], it might be a serious threat. This matter is also considered a possible security risk in mobile security testing guide by OWASP [79].

Vulnerable application should not filter what kind of data can be shown in screenshots and which data not.

Secure application should set up FLAG_SECURE in the activity that contains sensitive data by adding following line to the method of creation of the activity [79]:

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
WindowManager.LayoutParams. FLAG_SECURE);
```

Setting up the flag means that no information is displayed on a screenshot. Meaning, that user cannot take screenshots of the sensitive data as it simply will not be shown in the screenshot.

3.1.8 Allowing too weak passwords

According to Mobile Applications Security testing guide bugs in password policies are one of the biggest concerns related to the 4th vulnerability in OWASP top 10 mobile vulnerability list, insecure authentication [81] [59]. Weak passwords are easily crackable by automated cracking tools. According to a research, an average CPU is able to crack 95% of the passwords in few days [82]. Therefore, weak passwords leave application vulnerable by allowing unauthenticated users to log in.

Vulnerable application should only set minimum character limit to the length of the password. No limitation on password complexity should be set.

Secure application should allow only passwords that are allowed by OWASP [83].

The password should be at least 10 characters long. Otherwise they are considered as weak passwords [84]. They should not use any sequences and many identical characters in a row. Finally, passwords should contain uppercase and lowercase characters, a digit and a special character [85]. [83]

3.1.9 Poor encryption in password storage

According to study made in University of Piraeus "Evaluation of Cryptography usage in Android Applications" 87,8% of applications they tested had some mistakes

cryptography [86]. Cryptography mistakes are also mentioned in OWASP mobile security top 10. Poor cryptography implementation and weak algorithms might give developers false sense of security.
Using weak and fast cryptographic algorithms leave the passwords crackable using brute force attack, dictionary attack or rainbow-table attack.
Vulnerable application should use SHA-256 algorithm for storing password hashes. When only hashing algorithm is used without salts, same password hashes look the same. If an attacker knows one user's password, (s)he knows all other passwords that have the same hash.
Secure application should first demonstrate using SHA-256 with a salt. If salts are used, in addition to saving just password hash, also salt needs to be saved. The salt is added to the password and the result is hashed. Using salt makes sure that all hashes are different. Therefore, an attacker does not know which users use the same password. Also, due to the randomness added to each password, the cracking is more complex than without salts. As SHA-256 is very fast algorithm, a lot of different combinations can be computed in a short amount of time. Therefore, adding salts is not secure enough. In order to slow hash computation down, secure application should use key derivation functions in order to defend against brute force attacks. Those functions are built to be slow. OWASP password storage cheat sheet [87] recommends using Argon2 [88]. Therefore, the secure version should demonstrate usage of Argon2 algorithm.

3.1.10 Storing sensitive data in phone's external storage

Many options of data storage are provided by Android – shared preferences, internal storage, external storage [89].
Shared preferences is a good way of storing session data – it is written in private folder that is not accessible to other application. Unfortunately, shared preferences are quite limited. It can hold only key-value pairs. The value can be either boolean, float, int, long or sting. Custom data objects nor pictures are not supported. Internal storage is a place to store sensitive data not suitable for shared preferences. It is not accessible to other applications and therefore as safe as shared preferences. External storage is accessible to other applications, but it does have more space.
Developers store sensitive data to external storage, even though it is not a reasonable place for storing sensitive data as every application installed in the device can access it [90].

Vulnerable application should write login data to external storage.

First of all, the application needs read and write permissions to external storage in order to store data that way. Therefore, making malicious applications life easier. Read and write permissions are considered dangerous by Android [91]. Reading and writing to external storage can be done with BufferedReader and FileWriter. The files are written to external storage folder “DCIM”.

A malicious application can access the file from external storage exactly the same way with BufferedReader.

Secure application should write login data to shared preferences and data objects to internal storage.

3.2 System requirements

This section introduces requirements that needs to be implemented in order to demonstrate the previously mentioned vulnerabilities.

In order to implement the vulnerabilities, a whole system is needed – database, web service and two applications – main application to demonstrate small development mistakes that have a big impact and malicious application to demonstrate how easily third-party applications can interact with other applications if given a chance.

Vulnerability 1. Stealing files through enabled ADB backup

- ADB [92] installed in the computer in order to access the device from terminal.
- The application should store data in a private storage area in order to demonstrate accessing private files through enabled ADB backup.
- The application should have a possibility to enable and disable ADB backup.
- The device should have enabled USB debugging bridge in order to access the phone through command line from a computer.
- The computer should have python in order to remove header from backup file and save it as a compressed container.
- The computer should have access to dd tool.

Vulnerability 2. Getting information from broadcasted user’s data

- The application should broadcast data in order to demonstrate listening on unprotected broadcast messages.

- The application should define custom permissions in order to demonstrate defending broadcast messages with custom permissions.
- The application should define a broadcast receiver in order to demonstrate the difference between directed broadcast message and undirected global broadcast message.
- A malicious application that is listening on unprotected broadcast messages in order to demonstrate receiving broadcast messages.
- The computer should have apktool [72] installed in order to decompile Android application.

Vulnerability 3. Hardcoded API key in Android application

- Application should hold an API key in order to demonstrate the importance of protecting API keys.
- Application should be connected to git in order to demonstrate how to ignore uploading changes files to repository.
- The computer should have apktool [72] installed in order to decompile Android application.

Vulnerability 4. Bypassing authentication with malicious applications

- The main application should have a logged in activity exported and therefore accessible to other applications installed to the same device.
- The application should have different activities for logged in state and not logged in state in order to demonstrate bypassing authentication by executing application from logged in activity.
- A malicious application is needed that executes the main application in logged in state through exported logged in activity.
- The computer should have aapt tool [93] installed in order to identify exported activities in main application.
- The malicious application should execute activities in order to execute main application.

Vulnerability 5. Allow user to perform actions that (s)he does not have rights to do

- The application should have different activities for logged in state and not logged in state in order to demonstrate faulty Android task stack usage that allows restoring logged in state by clicking back button after destroying user session.

- The application should store user's data in order to provide single sign-on possibility and demonstrate what happens when the data is not cleared properly when user logs out.
- The application should have different user roles in order to demonstrate gaining more rights than should.

Vulnerability 6. Attack through unnecessary permissions granted by the user

- The application should request unnecessary permission in order to demonstrate the impact of allowing dangerous permissions.

Vulnerability 7. Stealing sensitive information from screenshots

- The Android device should be able to take screenshots and store it in external storage in order to demonstrate accessing screenshots from third-party applications installed on the same device.
- The application should be able to set FLAG_SECURE on an activity in order to demonstrate not allowing screenshots to be taken from sensitive information.
- The application should be able to have the previously mentioned flag removed easily in order to demonstrate screenshots without the flag.

Vulnerability 8. Allowing too weak passwords

- The application should have a possibility to change password requirements – from not limiting anything to limiting password strength according to OWASP standards.

Vulnerability 9. Poor encryption

- The applications should use three different ways of hashing passwords – plain SHA-256, SHA-256 with salt and SHA-256 with Argon2 in order to demonstrate the difference.
- A computer is needed in order to use hash crackers and rainbow-tables in order to demonstrate retrieving password from the hash.

Vulnerability 10. Storing sensitive data in phone's external storage

- One version of the main application should write login details to external storage and another one to shared preferences private folders in order to demonstrate the difference.

- A malicious application should read device’s external storage in order to access login details written to public folders.

3.3 Main application design

The application is designed to be a management tool for medium-sized to big companies. The main goal of the application is to simplify bureaucratic processes in companies. It allows to easily send notifications, request vacations and new tools, report working hours and have an overview of the employees of the company from one place. Therefore, it combines a lot of sensitive and confidential information in one place making the application an interesting target to a malicious actor. Few of the mentioned functionalities under this paragraph are not implemented during the first iteration – during this thesis.

3.3.1 Log in screen and authorization

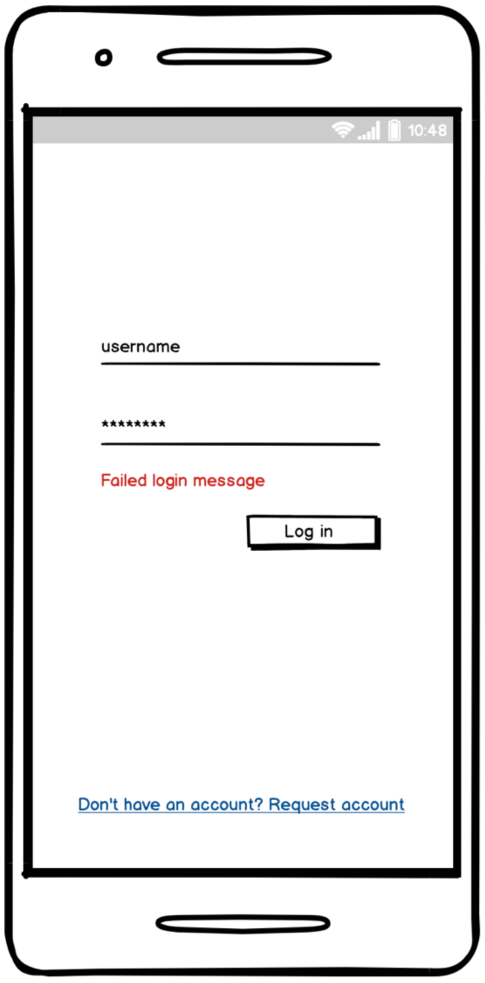


Figure 5. Log in screen

The app is accessible only to authenticated and authorized users. There are two roles available – manager and regular employee. Manager has rights to review and accept or decline regular employees’ requests and reports. Two roles are needed in order to demonstrate authorization problems caused by poor implementation.

Figure 5 shows the log in screen. It contains of two input fields – one for username and one for password, error message field, log in button and link to web view from where a user can request an account for the application.

Error message field is useful in order to demonstrate problems caused by showing too much information to a user. Link to web view is needed in order to demonstrate intent injection attack.

3.3.2 Notifications screen

Figure 7 shows the design of notifications screen. It contains of a greeting text, list of messages sent to the user and a button that allows user to send messages to other users. Current screen should serve as to do list. A user can delete a message or can mark it as done.

Sending new message will put an entry to other employees to do list. Send new message screen is shown on Figure 6.

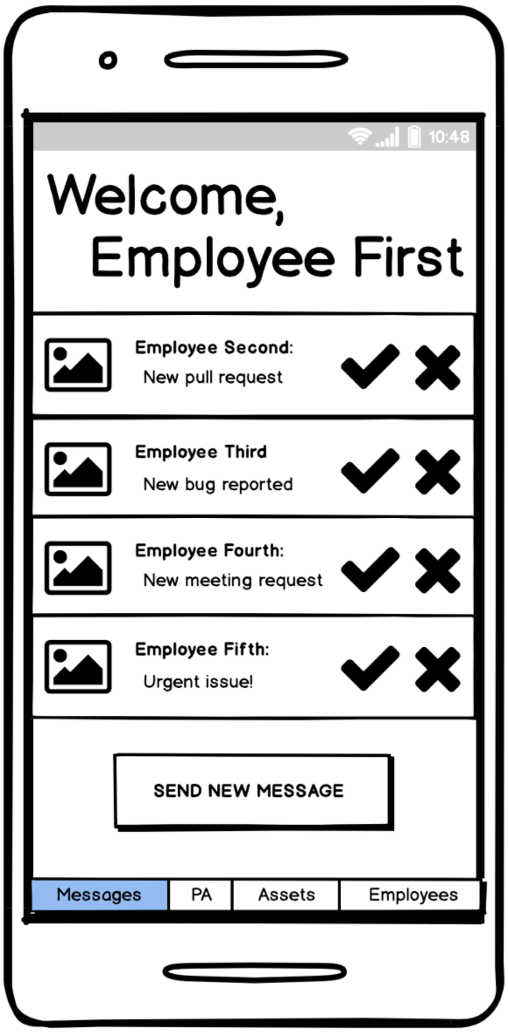


Figure 7. Notifications screen

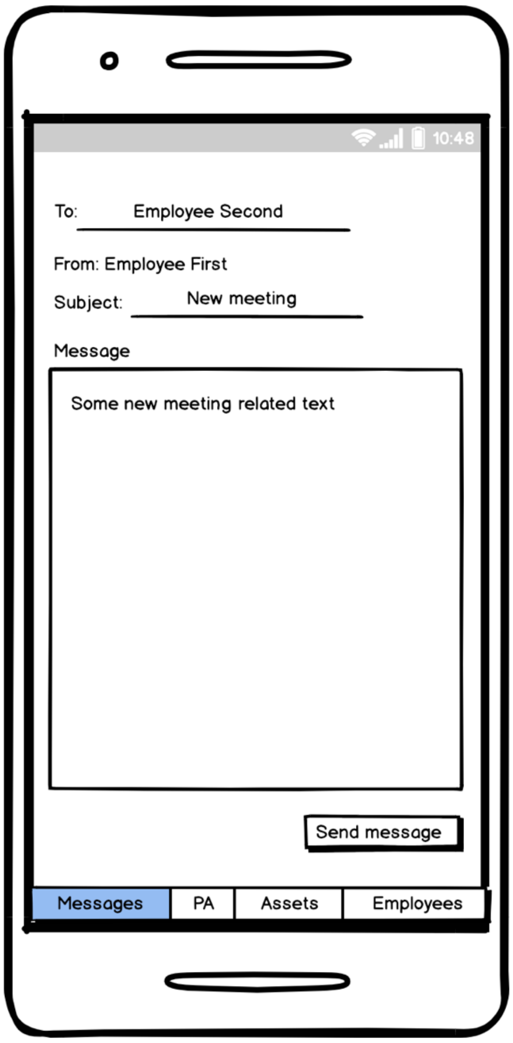


Figure 6. Send new message screen

3.3.3 Project accounting screen

The aim of project accounting screen (shown on Figure 8) is to provide possibility to report working hours, see reported working hours and request a vacation. The review working hours and vacation requests button is visible only for managers and therefore is marked green. The button opens a review page that is described below.

Report working hours button opens up a page that has date picker, three input fields and one dropdown menu. The design is shown on figure 9. The purpose of this view is to allow employees log their work. This view contains somewhat sensitive data about ongoing projects.

Request vacation page consists of two date picker fields, one dropdown menu and one input field. The view is needed so that an employee could create a vacation request that could be accepted or declined by the manager. The design is shown on figure 11.

Previous working hour page (shown on figure 10) is a convenient way of keeping track on previously completed tasks. The view shows information about how much the employee has spent time on a specific task.

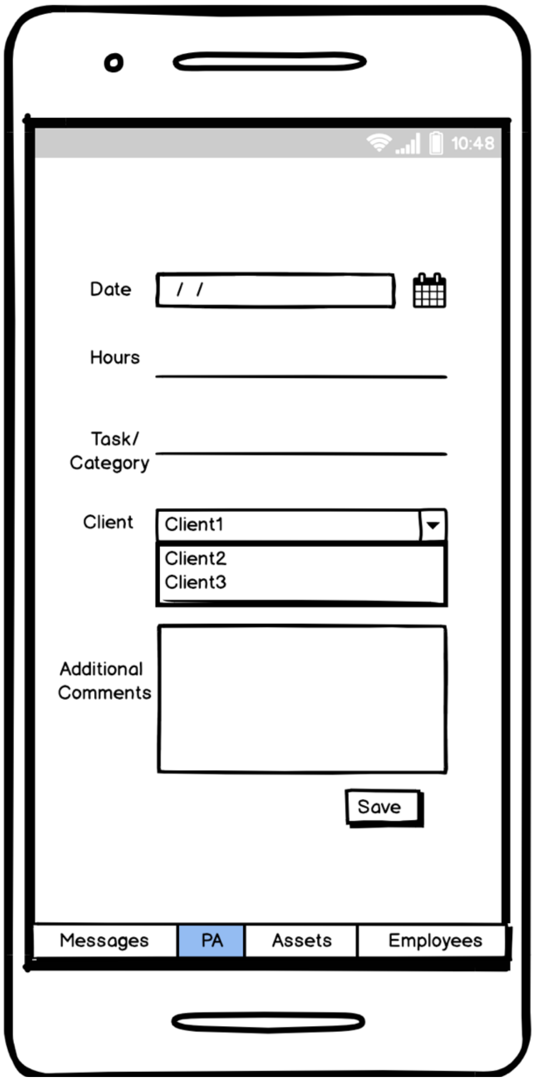


Figure 9. Report working hours screen

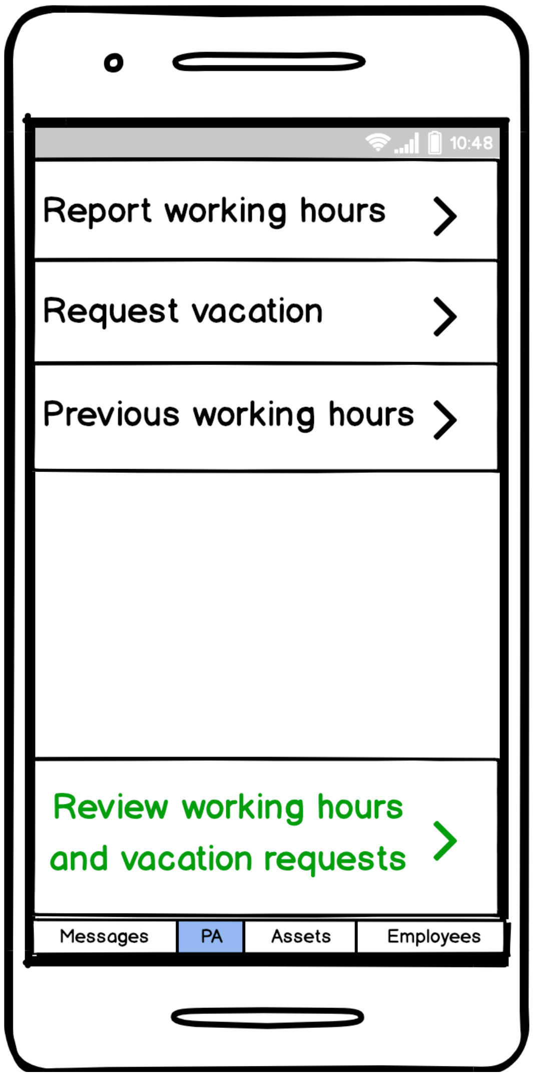
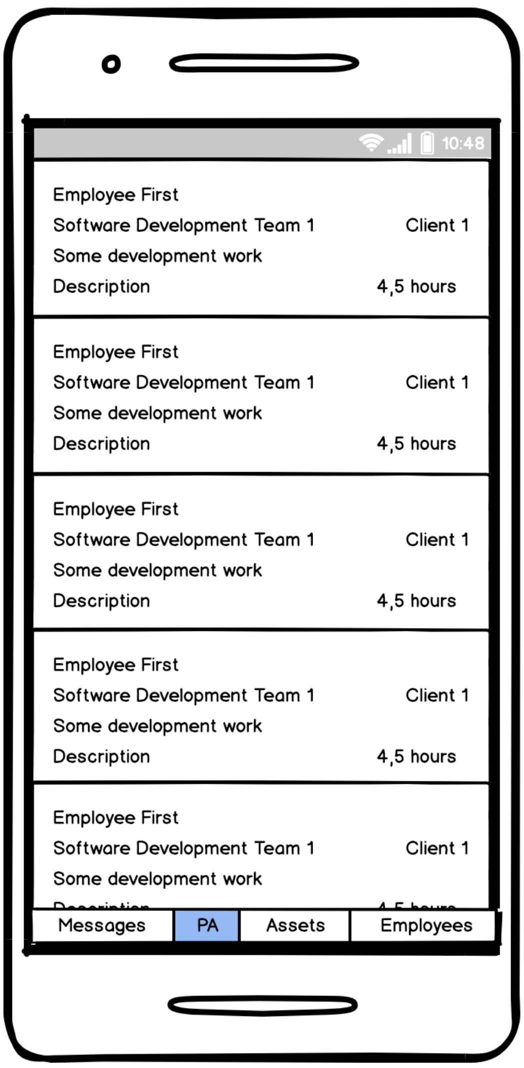


Figure 8. Project accounting screen


10:48

Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

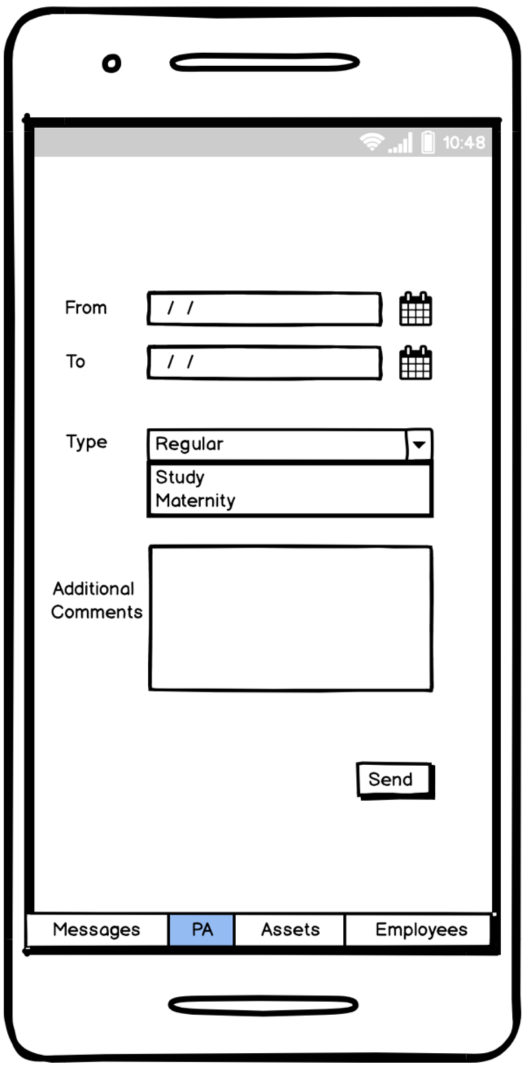
Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

Employee First
Software Development Team 1 Client 1
Some development work

Figure 10. Previously reported hours screen


10:48

From / /
To / /
Type Regular
Study
Maternity
Additional Comments

Figure 11. Request vacation screen

3.3.4 Review page

A review page (shown on Figure 15) is visible only for managers. It consists of a list view that shows checkbox and more detailed information about request or report. The view also has two buttons – approve or decline. If a user clicks on one of those buttons, a notification is sent to the person who made the request.

There are three types of items that can be shown in the list. All types show employee’s name and team in the field. In addition to name and team, all fields have custom fields. Firstly, working hours review field (shown on Figure 13) has task, work description, client’s name to whom was the work done for and amount of time spent on a task. Secondly, vacation request review field (shown on Figure 14) shows type (regular, study or maternity) and additional comments fields. Thirdly, new working tools review field (shown on Figure 12) consists of required new tool's type, reason for the requirement and a link to the requested tool.

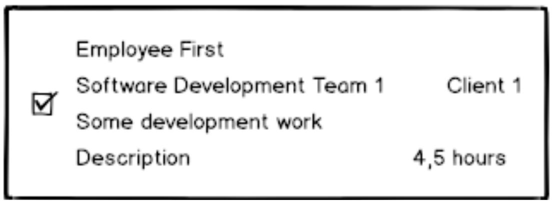

☑
Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

Figure 13. Working hours review field

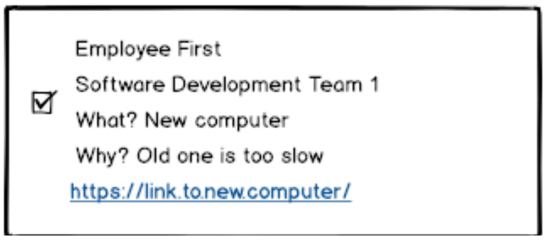

☑
Employee First
Software Development Team 1
What? New computer
Why? Old one is too slow
https://link.to.new.computer/

Figure 12. New tool request review field

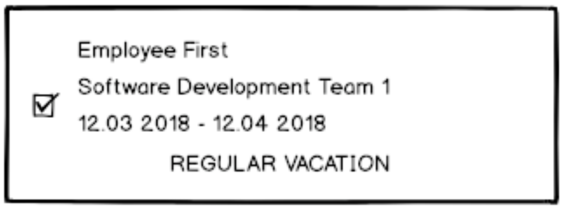
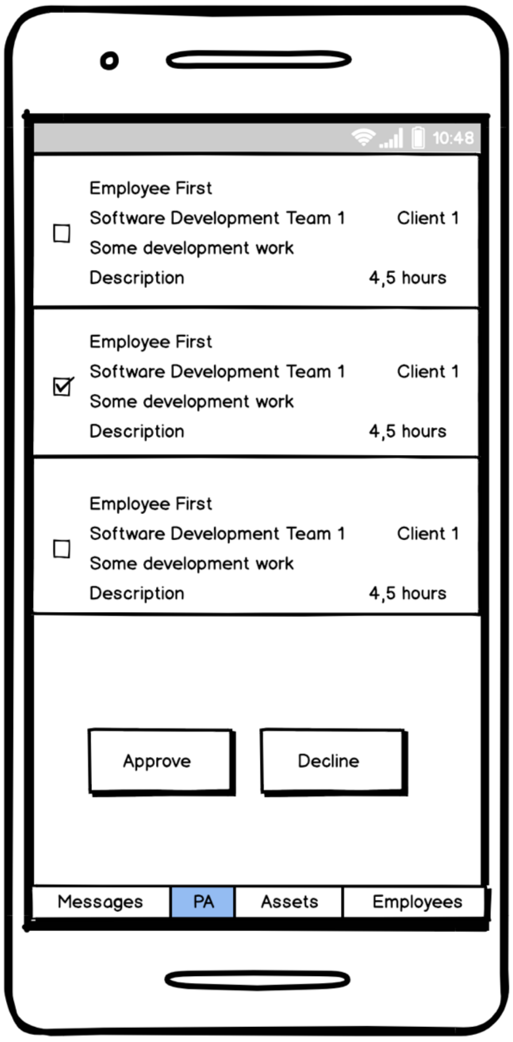

☑
Employee First
Software Development Team 1
12.03 2018 - 12.04 2018
REGULAR VACATION

Figure 14. Vacation request review field


10:48

☐
Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

☑
Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

☐
Employee First
Software Development Team 1 Client 1
Some development work
Description 4,5 hours

Figure 15. Review page

3.3.5 Employees' page

Employees page (Figure 17) shows the list of all employees working in the company. It also has search possibility in order to provide better user experience as there might be a lot of employees in one company. In the list, a user can see employee's name, position and picture. When user clicks on one of the employee's list item, a detailed view will be opened.

Detailed view looks different for employees and managers. Detailed view's design is shown on Figure 16. The fields painted green (social security number, IBAN, salary and edit info button) are displayed only for the users logged in as manager. Picture, employee score, name, job title, team, send messages and praise buttons are visible for all. Praise button can be used to raise employee's score.

Edit info turns text views into input fields that allows manager to edit employee's name, picture, job title, social security number, IBAN or salary per hour.
This view holds very sensitive information that should be accessible only by managers. Therefore, it gives a great opportunity to show the impact of authorization bugs in an application.

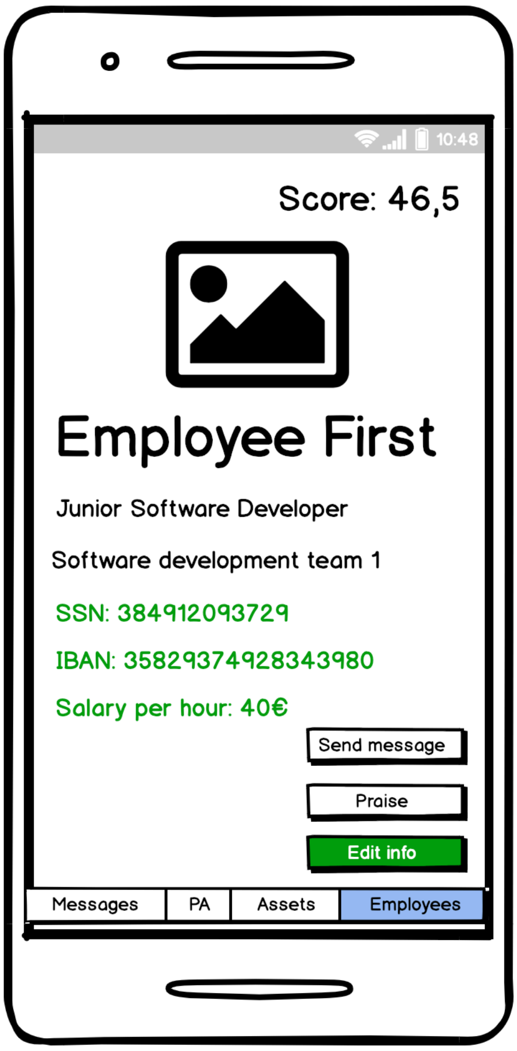


Figure 17. All employees page

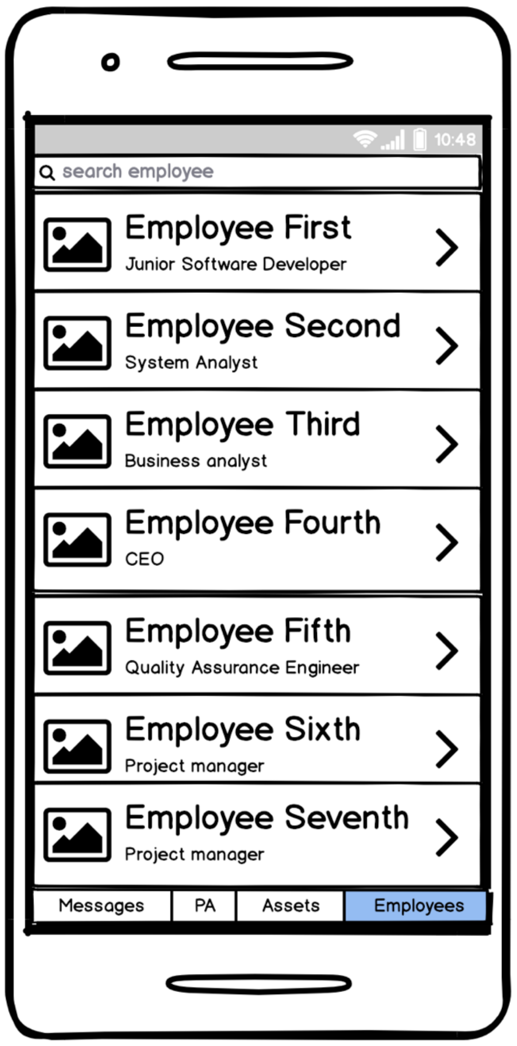


Figure 16. Employee details page

3.4 Malicious mobile application design

Malicious mobile application will not have any user interface. It will listen on broadcast messages, start public activities of other applications and read public storage of a device it is installed to.
Listening to broadcasts is needed to demonstrate how to listen to unprotected and undirected broadcasts sent by any other application installed on the device.

Starting public activities must be possible in order to demonstrate interacting with any application installed on the same device that exposes any components.
Reading public storage is important to show how to access data from other application if it is not protected and stored properly.

3.5 Database design

The main goal of the database is to serve the mobile application. It is designed to hold all the information needed for the mobile application. The database model is shown on Figure 18. The database contains of 11 tables: Work_type, Work_unit, Position, In_position, Actor, Actor_type, Message, Vacation, Vacation_type, Team and In_team.
Table “Work_type” holds different types of work: testing, analysis, development, administrative work. The table has two fields: automatically generated primary key work_type_id that is an integer and maximum 50 characters long varchar type that has to be unique.
Table “Actor_type” holds different types of actors: employee, manager, client, client_representative. The table has two fields: automatically generated primary key actor_type_id that is an integer and maximally 50 characters long varchar type that has to be unique.
Table “Actor” holds information about all employees and clients related to the company. The table consists of seven fields: automatically generated primary key “actor_id”, foreign key “actor_type_id” that represents “actor_type_id” from table “Actor_type”, nullable foreign key “reports_to_actor_id” that represents “actor_id” from table “Actor”, maximum 50 characters long varchar “name”, maximum 25 characters long unique varchar “identificator”, nullable maximum 50 characters long varchar “avatar_uri”, nullable double “score”. “Reports_to_actor_id” field shows who is the direct manager of the specific actor. If a user does not have a manager, the field is set to NULL. “Name” field shows first and last name of the actor. “Identificator” field shows social security number if actor is an employee and business registration number if actor is a client. “Avatar_uri” holds a location to a user-selected avatar. The field can be empty. “Score” field shows the score of a specific actor. The field is set to NULL until someone rates the actor.
Table “Position” holds different types of positions: business analyst, system analyst, quality engineer, software engineer, product owner, assistant, recruiter, accountant, COO,

CEO, CTO, team manager. All non-manager positions have senior, junior and middle-level positions. The seniority is also defined in position title field. Manager positions – COO, CEO, CTO, team manager does not have different levels. Table “Positions” also holds position-specific price per hour and salary coefficient. Price per hour shows how much client has to pay for that position’s work. Salary coefficient shows how much of the money will, the person who did the work, earn from hourly work. For example, if price per hour is 100€ and salary coefficient is 0.2 then the person who did the job will get 20€ per hour.

Table “In_position” shows which actor is in which position at what time. The table consists of five fields: automatically generated primary key “in_position_id”, foreign key “position_id” that represents “position_id” from table “Positions”, foreign key “actor_id” that represents “actor_id” from table “Actor”, date “from_date” and nullable date “to_date”. “From_date” shows the time when an actor started being in the specific position. “To_date” field is needed if the end of an actor being in a specific position is known. For example, if a person leaves the company or changes positions. If the end date is not known, “to_date” field is set to NULL.

Table “Team” holds all active team names in the company. The table consists of two fields: automatically generated public key “team_id” and unique maximum 50 characters long varchar “name”.

Table “In_team” is needed to keep track on which actor is part of which team at what time. The table consists of five fields: automatically generated primary key “in_team_id”, foreign key “team_id” that represents “team_id” from table “Team”, foreign key “actor_id” that represents “actor_id” from table “Actor”, date “from_date” and nullable date “to_date”. “from_date” field represents the time when an actor started working in a team. “to_date” can be either a date when a person stops working for the team or if it is unknowhń then it can be set to NULL.

Table “Work_unit” is needed in order to keep track on the work that has been done. The table consists of eight fields: automatically generated primary key “work_id”, foreign key “work_type_id” that represents “work_type_id” from table “Work_type”, foreign key “in_position_id” that represents “in_position_id” from table “In_position”, foreign key “in_team_id” that represents “in_team_id” from table “In_team”, date “date_done”, double “time_spent”, maximum 10 characters long varchar “task_number” and nullable text “description. “date_done” field shows on which day the work was done. “time_spent” shows how many hours was spent on completing the task or part of the task on that specific day. “task_number” shows what task of part of a task was done. “description” is optional field that can be filled by an actor if something related to a specific work unit needs to be specified.

Table ”Vacation_type” holds different types of vacations: regular vacation, study vacation, maternity vacation, other. The table consists of two fields: automatically generated primary key “vacation_type_id” and unique maximum 25 characters long varchar “type”.

Table “Vacation” is needed in order to keep track on which actors are taking a vacation, when they are taking a vacation and is it approved or not. The table consists of eight fields: automatically generated primary key “vacation_id”, foreign key “requested by actor_id” that represents “actor_id” from table “Actor”, nullable foreign key “reviewer_actor_id” that represents “actor_id” from table “Actor”, foreign key “vacation_type_id” that represents “vacation_type_id” from table “Vacation_type”, nullable Boolean “approved”, date “from_date”, date “to_date”, nullable text “description”. “Requested_by_actor_id” field shows who is requesting to take a vacation. “Reviewer_actor_id” field shows who reviewed the vacation request. If no one has reviewed it yet, the field is set to NULL. “Approved” field shows if the application is approved, rejected or has not been reviewed yet. If it is approved, the field is set to TRUE, if it is rejected, the field is set to FALSE, and if it has not been reviewed yet, the field value is NULL. “from_date” and “to_date” show the duaration of the requested vacations. “Description” field is present in order to provide possibility to add any comments or agreements to the vacation request.

Table “Message” holds messages that are sent from one actor to another one. The table consists of four fields: automatically generated primary key “message_id”, foreign keys “from_actor” and “to_actor” that represent “actor_id” from table “Actor”, text “message”. “From_Actor” shows who sent to message. “To_actor” shows who the message was sent. “Message” field holds the message content.

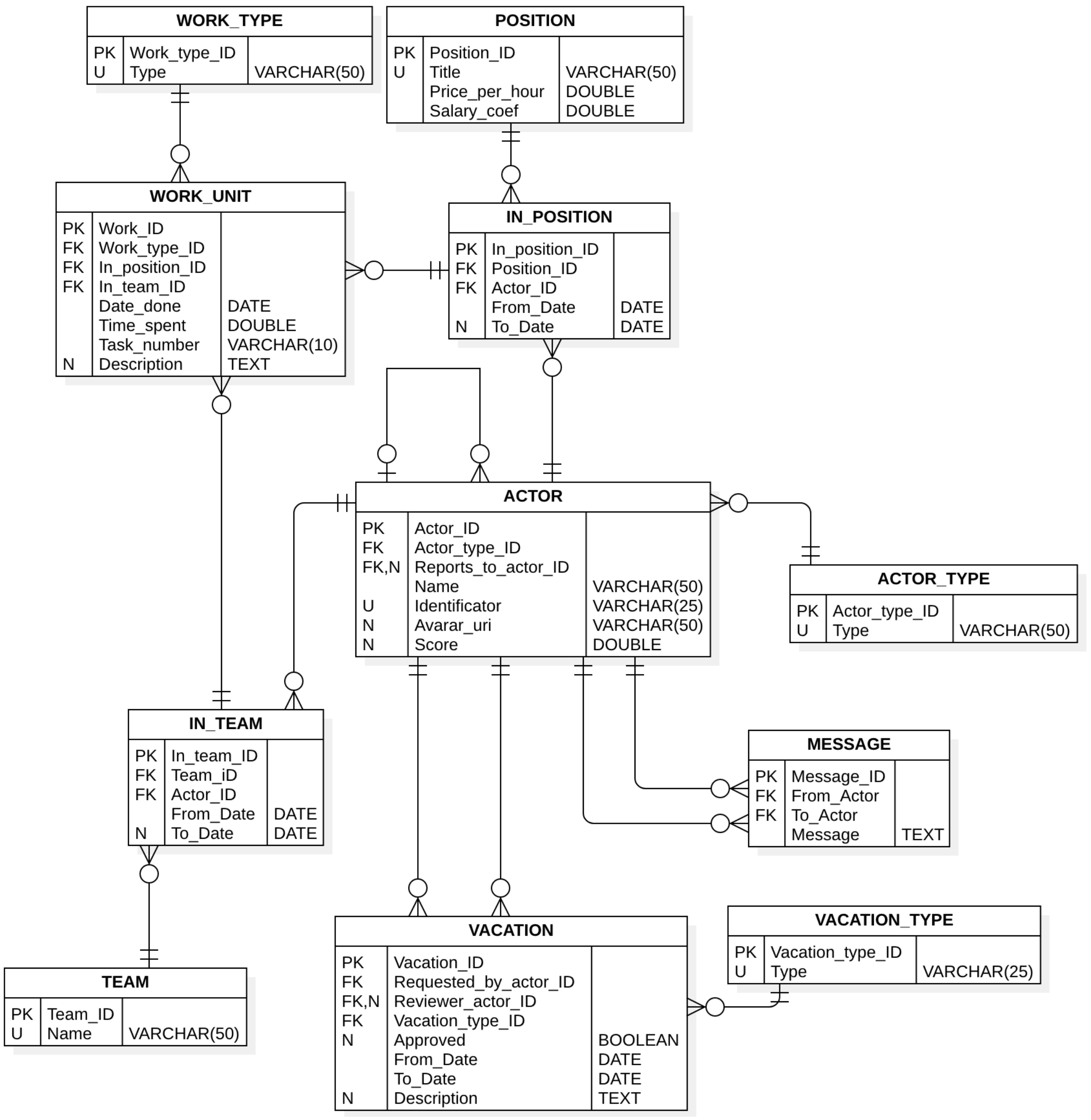


Figure 18. Database model

3.6 Web service design

Web service is needed in order to connect database and mobile application. The web service returns objects designed for the application.

Web service requirements:

Requirement 1. The service should authenticate the user in order to implement log in in the mobile application. Log in functionality is needed in order to demonstrate privilege escalation and bypassing authentication.

Requirement 2. The service should return login response with session ID or error code in order to identify the result of log in.

Requirement 3. The service should return messages list of a logged in employee in order to fill the messages list view. Having messaging functionality in the application allows demonstrating a social engineering aspect while having a vulnerable application.

Requirement 4. The service should remove message from a database when application requests removing messages by message id. This requirement is needed in order to make sense of messaging functionality in the application. If messages can be sent, it is logical to have a remove or hiding possibility. Otherwise, important messages might get lost in between other messages and users would not have a possibility to fix the problem.

Requirement 5. The service should add message to a database when user sends a message by sending message content, message receiver, message sender and a timestamp to the service. This requirement is also needed to make sense of the messaging functionality in the application.

Requirement 6. The service should show list of reported working hours of a logged in employee by sending following data to the application: team name, client name, task id, description and amount of time spent on the task. Working hours are included because on one hand, being sensitive information, it might be an interesting target for attacker. On the other hand, it is a detail that has to be approved by a manager. Meaning it is also needed to demonstrate privilege escalation vulnerability.

Requirement 7. The service should let the application know about possible work categories the employee can choose from by sending a list of strings to the application. Work categories can change in time – new ones can be added, and old ones can be deleted. Therefore, it is useful to have them in the back end. Entire application should be re-released if they were hard coded in the application.

Requirement 8. The service should add working hours to the database for a logged in employee by application sending the following data to the service: date of doing the work, hours spent on a task, team name, client name, work category, task id, description and employee id who did the work. This requirement is needed in order to complete working hours reporting functionality. The reported hours should be saved somewhere so that they could be accessible by the user who reported them and the manager who has to add an action to it.

Requirement 9. The service should show list of previously requested vacations by sending the following data to the application: vacation type, is vacation approved, vacation start date, vacation end type and description. Vacation functionality is needed for the same reasons as working hours. It also gives the application fuller feel. Meaning, the application feels more like a real application as you can do more relevant actions in the application. Furthermore, it allows hiding more vulnerabilities in the code and create more interesting scenarios.

Requirement 10. The service should let the application know about possible vacation types by sending the information to an application in a list of strings. The vacation types should also be defined in the back end. The reasons are listed under requirement 7.

Requirement 11. The service should add new vacation request to the database by application sending the following data to the service: vacation start data, vacation end date, vacation type and additional comments. The reasons behind this requirement are listed under requirement 8.

Requirement 12. The service should send list of manager's subordinates reported working hours containing following data: working hours object id, date of the work done, hours spent on the task, team name, client name, employee name, task id, task category and description. This requirement is needed so that manager could add actions to subordinate's requests. The subordinates cannot make requests to this endpoint. It is needed so that the user roles had different rights.

Requirement 13. The service should send list of manager's subordinates requested vacations containing following data: vacation object id, vacation start date, vacation end date, vacation requester name, vacation requester team and client name and description. The reasons behind this requirement are listed under requirement 12.

Requirement 14. The service should allow approving and rejecting reported working hours by application sending the following data to the service: work object id, is approved or not, manager's id who adds the action to the object. This requirement is needed in order to complete the functionality mentioned in requirement 12.

Requirement 15. The service should allow approving and rejecting requested vacations by application sending the following data to the service: vacation object id, is approved or not, manager's id who adds the action to the object. This requirement is needed in order to save actions made in the functionality mentioned in requirement 13.

Requirement 16. The service should return list of all employees in order to fill all employees view. The list should contain employee's name and job title. Employee list is needed in order to demonstrate stealing large amount of sensitive data with little effort as the whole set of data is in one place.

Requirement 17. The service should return employee details in order to show employee’s details view. The response should contain following information: employee score, employee name, job title, team name, social security number, iban and salary per hour. Employee details view is needed in order to have highly sensitive information in the application because it allows generating scenarios with severe consequences.

Requirement 18. The service should update employee details information when application requests the update and sends the same data as mentioned in previous requirement. This requirement allows to generate scenarios where employee details are changed by either a malicious user or any other user. For example, change someone’s salary to gain profit or remove all the data to cause chaos.

4 Technical description

4.1 Main application architecture

The main application uses Model-View-Presenter (MVP) pattern because the first iteration validation target group knows this pattern the best. The goal is to emphasize security problems. Therefore, to put as less effort into understanding the architecture as possible.

Model-View-Presenter pattern splits application into three components – model component, view component and logic (presenter) component. Model component holds data models and handles receiving the data. Model component communicates with logic component. Logic component is responsible for all the business logic in the application. Also, keeping model and view up to date – display any changes in models to the view and user interactions in view to the model. View component is responsible only for displaying the view and responding to user interactions with it.

4.2 Main application components

4.2.1 View

The application view contains of layout files, drawable files, adapters, fragments and activities. Layout files are written in xml, drawable files are in xml and in PNG. Other files are written in java. Layout files are used to define the user interface. Drawable files are used to hold background designs and icons.

Adapters are used to fill a view with same layout multiple times – for lists. The application uses six – MessagesAdapter for information view messages list, EmployeesAdapter for employee list, PreviousWorkHourAdapter for displaying reported working hours, RequestedVacationsAdapter for displaying requested vacations and their states, ReviewHoursAdapter for displaying reported working hours waiting for manager's action and ReviewVacationsAdapter for displaying requested vacations waiting for approval.

The application has two activities – LoginActivity and MainActivity. "An activity is a single, focused thing that the user can do" [94]. Activity is usually a full-screen layout container for pieces of views. The reason for having only two is that almost the whole application needs a bottom navigation bar. Only log in screen does not need it. Therefore, it is useful to have an activity for when user has logged in and another one when user sees the log in screen. LoginActivity's layout contains all the views visible in log in screen – username input field, password input field, log in button, error message layout and request user link. The reason why LoginActivity does not need a separate fragment is that it holds exactly one type of view at all times. Meaning, that the log in user interface looks the same for every user.

MainActivity holds a RelativeLayout with FrameLayout and BottomNavigationView inside it. FrameLayout is the part of the layout that starts to change when user interacts with the view.

Fragments are used in order to change contents of FrameLayout. "Fragment represents a behavior or a portion of user interface" [95]. The main application has ten fragments: ReviewFragment, RequestedVacationsFragment, RequestVacationFragment, PreviousWorkFragment, ReportWorkingHoursFragment, AccountingFragment, EmployeeDetailsFragment, EmployeesFragment, MessagesFragment and NewMessageFragment. Every view a user sees in the screen consists of either LoginActivity or MainActivity and one of previously mentioned fragments inside it.

4.2.2 Presenter

The application has 10 presenters – LoginPresenter, NewMessagePresenter, MessagesPresenter, EmployeesPresenter, EmployeeDetailsPresenter, ReportWorkingHourPresenter, PreviousWorkingHoursPresenter, RequestVacations-Presenter, RequestedVacationsPresenter and ReviewPresenter. The general goal of presenters in current application is to fetch the data from repositories, process the response and show something to a view – either successful response or an error.

4.2.3 Model

The data layer (model) of the application consists of data classes and repositories. Data classes define the structure of the data object. The application has 9 – Client, Credentials, Employee, LoginResponse, Message, Position, Team, Vacation and Work. The structure of a data object is defined by a web service and is followed by data objects in the application.

Repository responsibility is to request the data from web service, process it, save to cache and notify presenters. The application has 5 repositories – VacationsRepository, WorkingHoursRepository, EmployeesRepository and MessagesRepository.

4.2.4 Communication with web service

The application uses OkHttp [96] and Retrofit [97] libraries to communicate with web service.
Retrofit is an abstraction of the API the application wants to communicate with [98]. In the main application the abstraction is done in ApiService interface. It defines all the endpoints the app can communicate with.
OkHttp library allows to easily create a HTTP client connecting the application to the actual API defined in the interface mentioned earlier. The OkHttp client creation is done in ApiClient class in the main application.

4.2.5 Session management

Sessions are handled in SharedPreferencesManager and SessionManager classes. Secure version uses SharedPreferencesManager class and unsecure version uses SessionManager class.
SharedPreferencesManager saves session-related data to SharedPreferences. SharedPreferences is a singleton built in to Android framework, allowing to store session-based data. It means that when the app is closed, the data still exists until session is closed. Session can be closed by either a timeout or when a user logs out from the application. SharedPreferencesManager holds logged in employee's ID, name, information about employee being a manager or not and session ID. The information is saved to SharedPreferencesManager when user logs in.
SessionManager writes login data to a file in local storage when user logs in. The file is deleted when user logs out.

4.3 Malicious application

Malicious application contains of two Java classes: MainActivity and BroadcastReceiver. MainActivity sets up broadcast receiver, request storage read and write permissions, tries to read external storage and tries to open activity from another application. BroadcastReceiver handles what to do with the data once messages have been captured.

4.4 Backend

4.4.1 Communication with application

The main application gets all its data from web service. The service is needed as a connection between database and the application. The service has 15 endpoints. Example requests and responses can be found in Appendix 3.

4.4.2 GET requests

"/allEmployees" request returns list of all employees as a response. Request does not require any parameters. This request is meant for receiving the most basic information about employees. As teams' information and prices are not needed, they are set to either null or 0.0.

"/reportedHours" request needs logged in employee id as a request parameter and will return list of previously reported working hours by logged in employee.

"/requestedVacations" request needs logged in employee id as a request parameter and will return list of previously requested vacations by logged in employee.

"/messages" request needs logged in employee id as a request parameter and will return list of messages sent to logged in employee.

"/employeeDetails" request needs id of an employee whose details are requested as a request parameter and will return requested employee's details.

"/reportedHoursReview" request needs logged in employee id as a request parameter and will return list of hours reported by logged in employee's subordinates where approved field is null.

"/requestedVacationsReview" request needs logged in employee id as a request parameter and will return list of vacations requested by logged in employee's subordinates where approved field is null.

"/vacationTypes" request does not need any request parameters. The request will respond with list of vacation types as strings.

"/workCategories" request does not require any request parameters. It will respond with list of possible work categories as strings.

"/removeMessage" request requires integer message id as request body and will not return any data. The return message contains only status code.

4.4.3 POST requests

“/login” request requires Credentials data object as request body and will return LoginResponse data object.
“/reportHours” request requires Work data object as request body and will not return any data. The return message contains only status code.
“/requestVacation” request requires Vacation data object as request body and will not return any data. The return message contains only status code. “/sendMessage” request requires Message data object as request body and will not return any data. The return message contains only status code.
“/editEmployeeDetails” request requires changed employee data object as request body and will not return any data. The return message contains only status code.

4.4.4 Communication with database

Web service links its CRUD operations to the database.

4.5 Database

Database is running on MySQL engine. It was chosen due to wide usage in the industry as a SQL database. It is currently the most-used freeware SQL database [99]. Oracle is ranked the most-used overall but it is not freeware solution.

Database can be set up locally using Docker [100] engine. Further information provided in paragraph 4.6.

SQL sentences used in database creation can be found in Appendix 2.

4.6 Setting up the environment

Local setup requires four components:

- Main application (paragraphs 4.1 and 4.2)
- Malicious application (paragraph 4.3)
- Backend (paragraph 4.4)
- Compose repository (links the database and backend together and exposes the stack for the main application).

In addition to previously mentioned repositories, the local setup requires Android Studio [101], Docker [100] and Docker account. Creation of Docker account is free of charge. Android Studio is needed so that the main application and the malicious application could be built and run on Android emulator. It also allows a developer to browse the source code in a professional manner. Docker is needed to containerise the database and backend. Docker solution is not developed by the author of this thesis. It was developed by infrastructure specialist Matten Mätlik [102]. Further information about local backend setup can be found in Appendix 5.

In order to set up the application part, two repositories should be downloaded: Main application (https://gitlab.cs.ttu.ee/Kirke.Pralla/buggy-main-app) and Malicious application (https://gitlab.cs.ttu.ee/Kirke.Pralla/malicioso). The application should be opened in Android Studio and run in the same emulator so that they would end up in the same device. There is also third repository that can either looked at online or downloaded to see the behavior of the secure version of implementation of previously mentioned vulnerabilities. The repository can be downloaded from https://gitlab.cs.ttu.ee/Kirke.Pralla/app-sec-main-application.

4.7 Adding vulnerabilities

Vulnerabilities can be added by anyone interested. The tool can be locally customized. There is also possibility to add vulnerabilities to the online version. In order to do so, the developer should create a pull request to both secure and unsecure versions of the applications. Also, a documentation is needed in order to help people using the tool to test and teach the vulnerabilities. The documentation should contain an introduction to the vulnerability – why is it important. Also, a tutorial how to exploit it and a scenario with symptoms and reasons are necessary.

As any interested party can access and modify the whole system – database, web service, applications, and add parts to the system, such as new applications, the options are almost limitless.

5 Validation

In order to validate the results of current research, the author organized a interacting training session instructed by herself. The training is targeted to software developers who do not have participated on Android security training before. The training will start with introducing the importance of the topic. Next, the system will be introduced, and the application will be showed to the participants – a quick introduction is given about the main application in order to simplify finding the security problems.
The training itself will be a role play. The developers will act as security officers of a company whose system is flawed – they have to solve incidents. First, the author will talk about the symptoms of the incident. The participants have to find out a reason why this incident occurred. Brainstorming and cooperation is encouraged as it helps generating new ideas. When the participants go too far from the real solution, the author will start pointing out important details and guide them back to the scenario in order to help finding out the real issue. After detecting the flaw, the author will demonstrate and talk about the impact of the mistake to the system in order to bring out the seriousness of small mistakes.
In the end of the training, all participants will receive a feedback questionnaire. The questions can be found in Appendix 4. The goal of this questionnaire is to see how the people rate the training based on how useful it was for them and would they attend any similar training. The author also asks feedback on what was great about the training and what should be changed in order to make it better. Finally, the experience of developers is asked in order to see how experienced people participated in the training and if people with more experience in IT find the training more useful than less experienced people or vice versa.
The validation is considered successful if participants rate the training to be useful – the usefulness rating is above 50% and more than 50% of participants would like to participate part 2 of the training.
Moreover, a security specialist will attend the training in order to give review the training. He will not participate in discussions.

5.1 Scenarios used

A company ordered development of project accounting application called SYC. The application is meant for employees to communicate with each other, to request vacations, report working hours, approve subordinates' requests and see information about other employees. Soon after releasing the application, there have been many complaints about weird activities.

Scenario 1.

Symptoms: The employee's list have been private in the company. Now a partial employee list is floating around the internet.

Additional questions and hints:

- The list leaked after another new company's application got released.
- How the applications communicate with each other? How does SYC get its information?
- Business requirement: changes have to be broadcasted because making backend requests every time the app is opened is expensive.
- The new app was sent to employees by CEO as APK file in an email.
- Why would CEO send APK file in an email?
- The email was fake and not sent by CEO. Therefore, multiple employees installed a suspicious application to their devices that listened to broadcasted information.

Reason: The company ordered one more application – in-house car-pooling application. An email seemingly from CEO was sent to all the employees with good news of new application release. Many employees installed the application to their devices. It later came out that CEO was on a vacation and the application was not ready yet. The APK contained a malicious application that listened on broadcasts from main application.

Illustrates vulnerabilities: **Getting information from broadcasted user's data**

Scenario 2.

Symptoms: A backend developer notices something weird in the logs – thousands of requests from application trying to log in. Another weird aspect is that the password (password itself, not the hash) has the same length in every request.

Additional questions and hints:

- Why was the password length the same in every request?
- How did the attacker know the length of the password?
- There is a malicious application installed on the device that is able to read external storage of the device.
- Log in information is stored securely on private folders so that no third-party apps can access it.

Reason: An employee of the company started using the company's new mobile application. She enters her credentials and tries to log in. She has been having difficulties using the phone – she keeps taking screenshots by accident.
During a busiest season of a year, she discovered a new icon in her phone she had never seen before. She wanted to know what it was, so she opened it. A popup showed asking for some permissions that she accepted as she did not know the purpose of it and wanted to see what the app does. It turned out the application did not show any user interface. It was a malicious application reading the device's external storage for any sensitive and useful information.
Illustrates vulnerabilities: **Stealing sensitive information from screenshots, attack through unnecessary permissions granted**

Scenario 3.
Symptoms: A management got a report of ten times bigger bill from API services than usually.
Reason: The application has hardcoded API keys stored inside the project
Illustrates vulnerabilities: **Hardcoded API keys**
Scenario 4.
Symptoms: Weird messages from few employees
Additional questions and hints:

- As the bug cannot be in the back end, it has to be in the application.
- Where should the security team start looking from?
- How is the data stored?

Reason: The session information was written on device's external storage that is accessible to all applications installed in the same device. A young security enthusiast wanted to try penetration testing on applications for his research. He found the company's application's apk and saw sensitive information stored to external storage. He shared the information publicly thinking he is helping the company. Instead, a hacker found it and smuggled a storage reader application to few employee's devices and started reading sensitive data. As it got session ids, he was able to send fake messages from employee's accounts.
Illustrates vulnerabilities: **Storing sensitive data in phone's external storage, Attack through unnecessary permissions granted**

Scenario 5.

Symptoms: A manager realizes that entire team's salary has been risen with his approval, but he had not received the requests.

Additional questions and hints:

- How did the requests end up with the manager's approval?
- Someone must have gotten his privileges – session ID or log in credentials.
- What should security team do? Ask the manager questions? What kind of questions?
- Has anyone had had access to the manager's device, or has he used someone else's device?
- Has anyone contacted him with weird personal or company-related questions?
- The manager used subordinate's device but insists he logged out after using it.
- How is the log out implemented?

Reason: It was a busy day and a manager lost his mobile device. He needed to send few messages as soon as possible so he borrowed a device from a subordinate. When he was done, he logged out and gave the device back to the subordinate. He accidentally clicked back button when locking the device and did not see he was logged in again. As he usually did not log out, he did not find it weird that he was logged in the next time he picked up his device. But it did look a bit different this time. He looked through the things he thought was some new features and started approving vacations without realizing the impact of his deeds.

Illustrates vulnerabilities: **user to perform actions that (s)he does not have rights to do**

Scenario 6.

Symptoms: The security officer is scanning the dark web again and finds a complete list of usernames in sale.

Additional questions and hints:

- Usernames are first name + first letter of last name. Therefore, the list of usernames can be guessed from employee's list.
- Scanning the logs, no signs of brute force or any other suspicious activity is found.
- Employees of the company are thoroughly questioned, and no one has given out any information about employees. No case of a social engineering is found.

- Logs are looked through once more. It shows that someone without user ID has browsed the application. Meaning, that unauthenticated user accessed protected parts of the application.

Reason: A malicious actor got the application’s apk and installed it on his device. He started analysing the apk and found out that the MainActivity is exportable. As it also had LoginActivity, he assumed that MainActivity should be in logged in state, meaning that the application can be executed from log in state. He wrote a few lines of code that should start the MainActivity and installed it to the same device. He succeeded executing the activity and found himself logged in to the company’s application. As he looked around in the application, he noticed a complete list of employee names. He assumed the usernames are in the format of first name + first letter of last name, he made a list and put it on sale in the dark web.

Illustrates vulnerabilities: **Bypassing authentication with malicious applications**

Scenario 7.

Symptoms: Once again, a list of company’s app credentials are in sale in the dark web.

Additional questions and hints:

- Several employees got hacked. How?
- A malicious application was installed on the same device as company’s application. The malicious application was able to read external storage.
- Storing credentials is done properly to private folders.
- While looking for any vulnerabilities in the application, developers found allowBackup=true in manifest file.
- A malicious application was able to retrieve a password hash from backup file.
- Password was hashed with SHA-256 and no salts were used.

Reason: There was a massive attack launched against the company. All the employees got an email with apk file and request to test the first version of the new car-pooling application. The email also said the employees should enable USB debugging in order for it to work. Most of the employees installed the application immediately as they were looking forward to it. When employees opened the application, it requested the READ_EXTERNAL_STORAGE application. As this permission was also mentioned in the email, most employees were okay with it and accepted whatever the application requested. When the setup was done, the app looked like a real car-pooling app.

While the application was installed, it pulled ADB backup files from the application and read password hashes from the shared preferences file. The hashes were cracked by automatic cracking tools and put on sale in the dark web.
Illustrates vulnerabilities: **Allowing too weak passwords, Poor encryption, Stealing files through ADB backup**

5.2 Results

Training took place on 17th of April 2019 with 15 trainees participating. Most of the participants were quite experienced software developers. Only 3 of 15 participants had less than 2 years of software development experience. In addition, chief of cyber security in a software development company participated the training in order to give feedback. The feedback can be found in Appendix 6.

The workshop lasted for 1.5 hours. Participants were actively asking questions and trying to solve the security problems causing the incidents described in paragraph 5.1. Trainees found it quite difficult to think about small bugs that could have caused that big problems. Therefore, mostly first guesses were huge problems in multiple components of the system, including application and web service. At the end of the workshop, many participants said that the workshop was helpful because they did not know how small issues can have big impact.

In addition, many people said they liked that the workshop was interactive and game-like. They had a chance to participate as security team and try problems. People liked the scenarios being based on a real application and the security incidents being based on a real system. Participants also stated that the workshop got them thinking more about security.

On the negative side, people stated that the workshop was too short. They did not have enough time to think by themselves and they would've wanted to spend a bit more time looking through the code. Some people asked for slides and advised me to write an article about my workshop to bring awareness to more software developers.

11 people out of 13 said they would like to attend security workshop part 2. One person did not give a straight answer to the question and one said to not have enough experience so (s)he would feel that it would be too technical for him/her.

Feedback does not have all the participants opinion because two participants did not answer feedback questionnaire. The feedback contains 13 trainees' opinion. Overall rating to the usefulness of the workshop was 8.23 out of 10. The ratings can be seen in Figure 19.

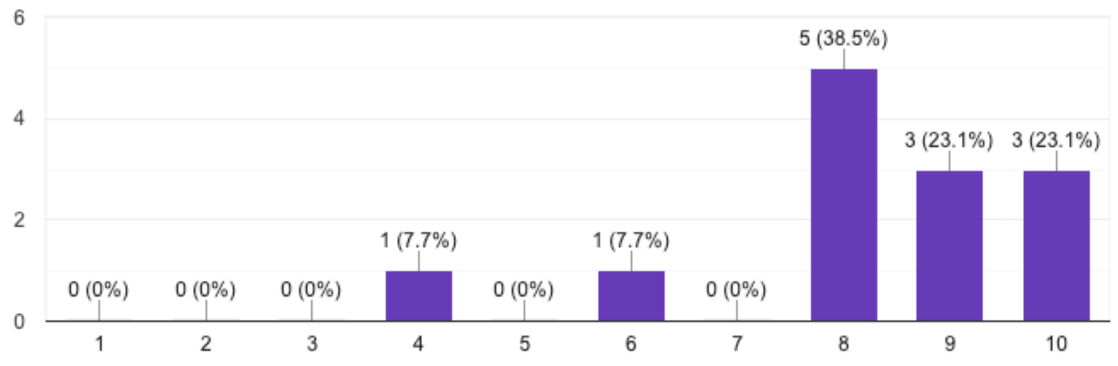


Figure 19. Usefulness ratings given by participants of the workshop

In conclusion, it can be said the workshop was useful as the overall usefulness rating was 8.23 and 84,6% of people were interested in workshop part 2 if it would be provided.

6 Future

Firstly, the author wishes to develop the system so that the vulnerable application would not have any known vulnerabilities in it. For example, currently, the secure version communicates to the web service over unsecure link – HTTP. The first goal is to change the communication to HTTPS. It is not yet done because network communication is a big topic that did not fit the current scope.

Secondly, the author wishes to add more functionality to the system and develop the system as planned in analysis chapter. It would be needed in order to provide more functionality and therefore, more options for security incident scenarios. On the one hand, it would make creation of incidents easier. On the other hand, it is believed by the author that it would also make the training sessions more interesting to the trainees. It would also provide possibility to make multiple different trainings with different content.

Thirdly, the goal would be to involve more people in the training sessions in order to raise awareness amongst developers. It could be done by communicating with different companies and providing short training sessions to the employees. The sessions would also be useful for IT students in order to emphasize the seriousness of security problems.

Fourthly, the author wishes to automate the training. For example, by setting up the environment in web application that could be accessible via URL. Currently the system has to be set up in local machine. Having the training automated and accessible as a web application would make the spreading of it simpler.

7 Summary

This thesis aimed to find a way to improve Android application security. Mobile market is constantly growing increasing the need for new developers [1]. The growth decreases the quality of Android applications as experienced developers are expected to work faster and market is open to a greater amount of inexperienced mobile developers [2] [3] [4]. Moreover, the growth makes Android devices more desirable target for malicious actors whose goal is to affect as many people as possible.

While looking for a solution to previously mentioned problem, author found multiple courses, blog posts, articles, code analysis tools and purposely vulnerable applications. Existing courses were costly and therefore not accessible to everyone. Blog posts and articles describe vulnerabilities and development mistakes but do not allow creation of interactive training or workshop. There is no possibility for user to try solving the problem by himself/herself. Code analysis tools are not 100% effective. Automated tools may miss big development flaws. Therefore, using only code analysis tools without any knowledge of security is not sufficient. All purposely vulnerable applications found were focused on training penetration testers not software developers. The projects consisted of small pieces of application not a whole application itself. Therefore, no scenarios can be created for trainees. Those kinds of projects only allow testing parts of an application. Due to the lack of materials, Android developers have limited possibilities to train themselves.

Author of this thesis wanted to find out if software developers and people involved in software development would benefit from interactive security workshop. Furthermore, if the workshop makes them think more about security and thereby improve security.

The solution provided by the author was creation of a system that could be used in interactive workshops and hands-on trainings. The system must be big enough to allow creation of scenarios. In this thesis, a project accounting system was. It allowed participants to log in as different user types, submit vacation requests, report working hours, send messages to each other and more. Participants of the workshop must have an opportunity to solve a problem by themselves not only get to know the solution. For example, participants receive information about leaked log in credentials. A discussion should start amongst the participants on what might have caused this incident. They also have the possibility to see the code and look at the logs. The scenarios are designed to be real-life like so that participants would feel like they are the security team of a company whose responsibility is to solve security incidents. All the scenarios are based on very small development flaws in order to demonstrate how small mistakes can cause big incidents.

The outcome of this thesis is a system including database, web service and Android applications that can be used to conduct security workshops, challenges and hands-on trainings. The system was put in test with 12 software developers, a chief of cyber security and 2 software quality engineers. The workshop was said to be very useful having 8.23 as usefulness rating out of 10. People also stated the workshop made them think more about security.

In the future, the system is developed even more, and more workshops will be held. Additional goal for the future is to develop the system to be more automated and used in online hands-on trainings.

References

[1] I. Gonacharov, "Agileengine.com," 2018. [Online]. Available: https://agileengine.com/mobile-developer-hiring-trends/. [Accessed 25 October 2018].

[2] Y. Acar, M. Backes, S. Bugiel, P. McDaniel and M. Smith, "SoK: Lessons Learned from Android Security Research for Appified Software Platforms," May 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7546516/. [Accessed July 2018].

[3] Y. Acar, M. Backes, S. Fahl , K. Doowon, M. Mazurek and C. Stransky, "You Get Where You're Looking for: The Impact of Information Sources on Code Security," May 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7546508. [Accessed July 2018].

[4] K. Underwood and E. Locasto , "In Search of Shotgun Parsers in Android Applications," May 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7527764/. [Accessed July 2018].

[5] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes and S. Fahl, "Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security," May 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7958574/. [Accessed July 2018].

[6] R. Westervelt, "Report warns of Android security issues, increased malware, Web attacks," TechTarget, August 2011. [Online]. Available: https://searchsecurity.techtarget.com/news/2240039134/Report-warns-of-Android-security-issues-increased-malware-Web-attacks. [Accessed October 2018].

[7] I. Majocha, "Report: Top Android Security Problems in 2017," December 2017. [Online]. Available: https://dzone.com/articles/report-top-android-security-problems-in-2017. [Accessed October 2018].

[8] D. Maier, T. Müller and M. Protsenko, "Divide-and-Conquer: Why Android Malware Cannot Be Stopped," September 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6980261. [Accessed March 2019].

[9] Statista, "Android version market share distribution among smartphone owners as of September 2018," 2018. [Online]. Available: https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/. [Accessed October 2018].

[10] A. Peters, "How Android differs depending on the hardware manufacturer," 2017. [Online]. Available: http://cdn.makeuseof.com/wp-content/uploads/2017/11/How-Android-Differs-Depending-on-the-Hardware-Manufacturer2.pdf. [Accessed October 2018].

[11] "Distribution dashboard," Google, [Online]. Available: https://developer.android.com/about/dashboards/. [Accessed August 2018].

[12] J. Joshi and C. Parekh, "Android smartphone vulnerabilities: A survey," April 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7578857. [Accessed October 2018].

[13] S. Iqbal, A. Yasin and T. Naqash, "Android (Nougats) security issues and solutions," April 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8394488. [Accessed November 2018].

[14] "How Android phones hide missed security updates from you," 12 April 2018. [Online]. Available: https://www.wired.com/story/android-phones-hide-missed-security-updates-from-you/. [Accessed October 2018].

[15] B. Heater, "Are hardware makers doing enough to keep Android phones secure?," 2018. [Online]. Available: https://techcrunch.com/2018/04/12/are-hardware-makers-doing-enough-to-keep-android-phones-secure/?guccounter=2. [Accessed June 2018].

[16] K. Nohl and J. Lell, "The Android ecosystem contains a hidden patch gap," April 2018. [Online]. Available: https://srlabs.de/bites/android_patch_gap/. [Accessed October 2018].

[17] "Duo Labs," [Online]. Available: https://duo.com/labs . [Accessed October 2018].

[18] O. Anise, "Thirty percent of Android devices susceptible to 24 critical vulnerabilities," Duo Labs, 28 June 2016. [Online]. Available: https://duo.com/decipher/thirty-percent-of-android-devices-susceptible-to-24-critical-vulnerabilities. [Accessed October 2018].

[19] M. Alenezi and I. Almomani, "Abusing Android permissions: A security perspective," October 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8257772. [Accessed November 2018].

[20] A. Mendoza and G. Gu, "Mobile Application Web API Reconnaissance: Web-to-Mobile Inconsistencies & Vulnerabilities," May 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8418636. [Accessed August 2018].

[21] A. Masood, "Cyber security for service oriented architectures in a Web 2.0 world: An overview of SOA vulnerabilities in financial services," November 2013. [Online]. Available: https://ieeexplore.ieee.org/document/6698966 . [Accessed July 2018].

[22] Statista, "Number of available applications in the Google Play Store from December 2009 to December 2018," December 2018. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ . [Accessed January 2019].

[23] P. K. Granthi and B. S. M, "Android Security: A Survey of Security Issues And Defenses," July 2017. [Online]. Available: https://www.irjet.net/archives/V4/i7/IRJET-V4I794.pdf . [Accessed July 2018].

[24] J. Vijayan, "Google's Industry Rivals Report Security Issues on Play Store, Chrome," October 2017. [Online]. Available: https://www.eweek.com/security/google-s-industry-rivals-report-security-issues-on-play-store-chrome. [Accessed October 2018].

[25] O. Storey, "More malware found on Google Play Store," 6 September 2017. [Online]. Available: https://www.eset.com/uk/about/newsroom/blog/more-malware-found-on-google-play-store/?ref=AFC-CJ&attr=8179212&pub=10539651&shop=100099X1555751X30c92c82cc6cb926ecd7891555d0b200&utm_source=8179212&utm_medium=affiliate&utm_content=10539651&cjevent=bf1c1fc34b1. [Accessed November 2018].

[26] "ExpensiveWall: A dangerous 'packed' malware on Google Play that will hit your wallet," Check Point Software Technologies Ltd, 14 September 2017. [Online]. Available: https://blog.checkpoint.com/2017/09/14/expensivewall-dangerous-packed-malware-google-play-will-hit-wallet/. [Accessed October 2018].

[27] M. Backes, S. Bugiel and E. Derr, "Reliable Third-Party Library Detection in Android and its Security Applications," October 2016. [Online]. Available: https://www.researchgate.net/publication/310824047_Reliable_Third-Party_Library_Detection_in_Android_and_its_Security_Applications. [Accessed September 2018].

[28] "Stack Overflow," [Online]. Available: https://stackoverflow.com/. [Accessed March 2019].

[29] A. Jakhar, "DIVA Android," January 2016. [Online]. Available: https://github.com/payatu/diva-android. [Accessed October 2018].

[30] A. Sejpal and K. Sawheny, "Digitalbank," 16 August 2015. [Online]. Available: https://github.com/CyberScions/Digitalbank. [Accessed October 2018].

[31] J. Mannino, "OWASP GoatDroid Project," 16 September 2012. [Online]. Available: https://github.com/jackMannino/OWASP-GoatDroid-Project. [Accessed October 2018].

[32] M. Veytsman, "Android Labs," 6 July 2011. [Online]. Available: https://github.com/SecurityCompass/AndroidLabs. [Accessed October 2018].

[33] D. Shetty, "Android Insecure Bank v2," 2018. [Online]. Available: https://github.com/dineshshetty/Android-InsecureBankv2 . [Accessed October 2018].

[34] S. Stubbs, "OWASP Security Shepherd," 2018. [Online]. Available: https://github.com/OWASP/SecurityShepherd. [Accessed November 2018].

[35] B. Mueller, "UnCrackable Mobile Apps," November 2018. [Online]. Available: https://github.com/OWASP/owasp-mstg/tree/master/Crackmes. [Accessed November 2018].

[36] A. Brucker, "Damn Vulnerable Hybrid Mobile App," August 2018. [Online]. Available: https://github.com/logicalhacking/DVHMA. [Accessed November 2018].

[37] HTBridge, "Purposefully Insecure and Vulnerable Android Application," February 2018. [Online]. Available: https://github.com/htbridge/pivaa. [Accessed November 2018].

[38] A. Peruma, H. Surve, P. Sane and S. Mohapatra, "VulnerableAndroidAppOracle," July 2018. [Online]. Available: https://github.com/dan7800/VulnerableAndroidAppOracle. [Accessed November 2018].

[39] "Lab Activities," [Online]. Available: https://sites.google.com/site/mobilesecuritylabware/6-mobile-coding-vulnerability/lab-activities. [Accessed July 2018].

[40] S. Vetrini and D. Linguaglossa, "AHE17 : Android Hacking Events 2017," 27 June 2017. [Online]. Available: https://github.com/mseclab/AHE17. [Accessed July 2018].

[41] S. Hovsmith, "Hands On Mobile API Security: Get Rid of Client Secrets," 11 May 2017. [Online]. Available: https://hackernoon.com/hands-on-mobile-api-security-get-rid-of-client-secrets-a79f111b6844 . [Accessed July 2018].

[42] "App security best practices," Google, [Online]. Available: https://developer.android.com/topic/security/best-practices. [Accessed July 2018].

[43] B. Mueller, "Android App Security Checklist," 17 January 2018. [Online]. Available: https://github.com/b-mueller/android_app_security_checklist. [Accessed July 2018].

[44] B. Mueller, S. Schleier and J. Willemsen, "OWASP Mobile Security Testing Guide," 2017. [Online]. Available: https://github.com/OWASP/owasp-mstg#android-testing-guide. [Accessed October 2018].

[45] "Mobile Security And Exploitation Training," Attify Store, 2018. [Online]. Available: https://www.attify-store.com/collections/real-world-training/products/mobile-security-and-exploitation-training. [Accessed March 2019].

[46] "Mobile Application Security and Penetration Testing," eLearn Security, [Online]. Available:

https://www.elearnsecurity.com/course/mobile_application_security_and_penetration_testing/. [Accessed October 2018].

[47] "Android Mobile Application Hacking 3-days," AppSec Labs, [Online]. Available: https://appsec-labs.com/portfolio-item/android_hacking_3d/#tab-id-3. [Accessed July 2018].

[48] A. Abolhadid, "Hacking mobile applications," [Online]. Available: https://www.troopers.de/troopers19/trainings/jlhrvg/. [Accessed July 2018].

[49] S. Rajguru, "Mobile application hacking - master class," February 2019. [Online]. Available: https://nullcon.net/website/goa-2019/training/mobile-application-hacking.php. [Accessed March 2019].

[50] S. Hovsmith, "Preventing Mobile App API Abuse," January 2019. [Online]. Available: https://2019.appseccalifornia.org/index.php/program/sessions/. [Accessed March 2019].

[51] "NullConn registration," [Online]. Available: https://nullcon.net/website/register.php. [Accessed March 2019].

[52] "Troopers trainings," [Online]. Available: https://www.troopers.de/troopers19/trainings/. [Accessed March 2019].

[53] "Improve your code with lint checks," Google, [Online]. Available: https://developer.android.com/studio/write/lint. [Accessed March 2019].

[54] "CodeDx," [Online]. Available: https://codedx.com/. [Accessed March 2019].

[55] B. Pugh and A. Loskutov, 3 June 2015. [Online]. Available: http://findbugs.sourceforge.net/. [Accessed October 2019].

[56] "Veracode," [Online]. Available: https://www.veracode.com/. [Accessed March 2019].

[57] A. Z. Baset and T. Denning, "IDE Plugins for Detecting Input-Validation Vulnerabilities," May 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8227300. [Accessed July 2018].

[58] V. P. Ranganath and J. Mitra, "Are Free Android App Security Analysis Tools Effective in Detecting Known Vulnerabilities?," June 2017. [Online]. Available: https://www.researchgate.net/publication/325986364_Are_Free_Android_App_

Security_Analysis_Tools_Effective_in_Detecting_Known_Vulnerabilities. [Accessed July 2018].

[59] " Mobile Top 10 2016-Top 10," OWASP, 2016. [Online]. Available: https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10. [Accessed November 2018].

[60] "Now Secure," [Online]. Available: https://www.nowsecure.com. [Accessed July 2018].

[61] A. Hoog and B. Reed, "85% of App Store Apps Fail OWASP Mobile Top 10: Are you exposed?," 23 January 2018. [Online]. Available: https://www.nowsecure.com/webinars/85-appstore-apps-fail-owasp-mobile-top-10-exposed/. [Accessed October 2018].

[62] "HackerOne," 2018. [Online]. Available: https://www.hackerone.com/. [Accessed November 2018].

[63] R. Christian, "Exploiting Android backup," 28 April 2018. [Online]. Available: https://securitygrind.com/exploiting-android-backup/ . [Accessed November 2018].

[64] "allowBackup Documentation," Google, [Online]. Available: https://developer.android.com/reference/android/R.attr.html#allowBackup. [Accessed February 2019].

[65] B. Mueller, S. Schleier and J. Willemsen, "Testing Backups for Sensitive Data," OWASP, 2018. [Online]. Available: https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05d-testing-data-storage#testing-backups-for-sensitive-data. [Accessed March 2019].

[66] "ADB Backup is enabled within AndroidManifest," 27 July 2017. [Online]. Available: https://hackerone.com/reports/170398. [Accessed October 2018].

[67] "Hacking Android Apps Using Backup Techniques," Infosec Institute, 3 February 2014. [Online]. Available: https://resources.infosecinstitute.com/android-hacking-security-part-15-hacking-android-apps-using-backup-techniques/ . [Accessed March 2019].

[68] "Broadcasts overview," Google, [Online]. Available: https://developer.android.com/guide/components/broadcasts. [Accessed October 2018].

[69] M. Reizelman, "Twitter for android is exposing user's location to any installed android app," 13 January 2017. [Online]. Available: https://hackerone.com/reports/185862. [Accessed October 2018].

[70] "Fallible," [Online]. Available: https://fallible.co/. [Accessed March 2019].

[71] "We reverse engineered 16k apps, here's what we found," Fallible, 15 January 2017. [Online]. Available: https://hackernoon.com/we-reverse-engineered-16k-apps-heres-what-we-found-51bdf3b456bb. [Accessed March 2019].

[72] C. Tumbleson and R. Wiśniewski, "Apktool," 2019. [Online]. Available: https://ibotpeaches.github.io/Apktool/. [Accessed March 2019].

[73] "Improper Export of Android Application Components," 27 December 2018. [Online]. Available: https://cwe.mitre.org/data/definitions/926.html. [Accessed March 2019].

[74] S. R. Kotipalli and M. A. Imran, in *Hacking Android*, Birmingham, Packt Publishing, 2016, p. 198.

[75] "Intent FLAG_ACTIVITY_CLEAR_TASK," Google, [Online]. Available: https://developer.android.com/reference/android/content/Intent#FLAG_ACTIVITY_CLEAR_TASK. [Accessed March 2019].

[76] "Intent FLAG_ACTIVITY_NEW_TASK," Google, [Online]. Available: https://developer.android.com/reference/android/content/Intent#FLAG_ACTIVITY_NEW_TASK. [Accessed March 2019].

[77] "Permissions Overview," Google, [Online]. Available: https://developer.android.com/guide/topics/permissions/overview. [Accessed March 2019].

[78] B. Mueller, S. Schleier and J. Williamsen, "Testing App Permissions," in *Mobile Security Testing Guide*, OWASP, 2018, pp. 158-163.

[79] B. Mueller, S. Schleier and J. Willemsen, "Finding Sensitive Information in Auto-Generated Screenshots," in *OWASP Mobile Security Testing Guide*, OWASP, 2018, pp. 121-122.

[80] J. Palmerino, "Improving Android Permissions Models for Increased User Awareness and Security," May 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8543431. [Accessed March 2019].

[81] B. Mueller, S. Schleier and J. Willemsen, "Best Practices for Passwords," in *OWASP Mobile Security Testing Guide*, OWASP, 2018, pp. 32-35.

[82] L. Bošnjak, J. Sreš and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," May 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8400211. [Accessed 9 May 2019].

[83] D. Righetto, "Authentication Cheat Sheet," OWASP, 28 December 2018. [Online]. Available: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Authentication_Cheat_Sheet.md. [Accessed March 2019].

[84] M. Turan, E. Barker, W. Burr and L. Chen, "Recommendation for Password-Based Key Derivation Part 1: Storage Applications," December 2010. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf. [Accessed 9 May 2019].

[85] OWASP, "Password special characters," 27 January 2018. [Online]. Available: https://www.owasp.org/index.php/Password_special_characters. [Accessed 9 May 2019].

[86] A. Chatzikonstantinou, M. Group, C. Ntantogian, C. Xenakis and G. Karopoulos, "Evaluation of Cryptography Usage in Android Applications," December 2015. [Online]. Available: https://www.researchgate.net/publication/290181523_Evaluation_of_Cryptography_Usage_in_Android_Applications. [Accessed August 2018].

[87] J. Steven, J. Manico and D. Righetto, "Password Cheat Sheet," 2019. [Online]. Available: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md#ref7. [Accessed March 2019].

[88] A. Biryukov, D. Dinu and D. Khovratovich, "Argon2 specification," 26 December 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf. [Accessed March 2019].

[89] "Data Storage," Google, [Online]. Available: https://developer.android.com/guide/topics/data/data-storage. [Accessed November 2018].

[90] S. Makkaveev, "Man-in-the-Disk:Android Apps Exposed via External Storage," 12 August 2018. [Online]. Available: https://research.checkpoint.com/androids-man-in-the-disk/. [Accessed March 2019].

[91] "Dangerous Permissions," Google, [Online]. Available: https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions. [Accessed March 2019].

[92] "Android Debug Bridge (adb)," Google, [Online]. Available: https://developer.android.com/studio/command-line/adb. [Accessed March 2019].

[93] "AAPT2," Google, [Online]. Available: https://developer.android.com/studio/command-line/aapt2. [Accessed March 2019].

[94] "Activity documentation," Google, [Online]. Available: https://developer.android.com/reference/android/app/Activity. [Accessed March 2019].

[95] "Fragments documentation," Google, [Online]. Available: https://developer.android.com/guide/components/fragments. [Accessed March 2019].

[96] "OkHttp," Square, [Online]. Available: https://square.github.io/okhttp/. [Accessed March 2019].

[97] "Retrofit," Square, [Online]. Available: https://github.com/square/retrofit. [Accessed March 2019].

[98] "Retrofit Documentation," Square, [Online]. Available: https://square.github.io/retrofit/. [Accessed March 2019].

[99] "Ranking of the most popular database management systems worldwide, as of March 2019," Statista, [Online]. Available: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/. [Accessed 14 April 2019].

[100] "Docker," [Online]. Available: https://www.docker.com/. [Accessed 14 April 2019].

[101] "Android Studio," Google, [Online]. Available: https://developer.android.com/studio/?gclid=EAIaIQobChMIuqz6qtjP4QIVR6sYCh2P5w9vEAAYASAAEgJxLPD_BwE. [Accessed 14 April 2019].

[102] "Matten Mätlik," Linkedin, [Online]. Available: https://www.linkedin.com/in/mmatlik/. [Accessed 14 April 2019].

Appendix 1 – Experts opinion on the topic

Ekspert 1.

Kas Teile teadaolevalt on sellise sisuga tööd varem tehtud?

On olemas blogisid koos koodunäidetega. Palju teksti blogi + pdf, aga handson/konkreetne hunnik valmis tükke, mida saaks ise example rakenduses läbi proovida, ei tea, et oleks. Ise ei ole kohanud, aga pole otsinud.

Kas Teie arvates oleks sellisest tööst kasu?

Kindlasti. Ise android teinud alates android beetast (10+ a). Selle aja jooksul palju sama teemaga tegelenud. Uued arendajad ei tea turvalisusest ning ei oska sellele tähelepanu pöörata. Läheb palju aega enne kui turvalisusest aimu hakkab saama. Androidispetsiifilised turvateemasid tuntakse väga vähe.

Kas Teie arvates on adresseeritavad vead asjakohased või peaks midagi lisama/eemaldama?

Need on väga üldiselt kirjas. Raske öelda. Kõik on väga valiidsed. Oleneb kui palju ja mida valdkonns katta. Midagi Android-spetsiifilist panna juurde. Näiteks kuidas rakendus enda komponente teiste rakenduste eest kaitseb.

Kas Te arvate, et programmeerijad kasutaksid sellist raamistikku, kui oleks aeg, koht ning võimalus?

Mõned kindlasti iseseisvalt. Enamike peaks suunama. Paljud ei taha süvitsi minna, tundub jube keeruline asi. Töökeskkonnas tiimijuht peaks suunama.

Ekspert 2.

Kas Teile teadaolevalt on sellise sisuga tööd varem tehtud?

Idee on hea, aga praegu on olemas mitu challenge äppi (tehtud tahtilikult valesti), mis üritavad saavutada sarnast asja, nagu antud töös (nt DIVA). Nendes rakendustes on üritatud katta OWASPi välja toodud punkte. Probleem nende äppidega on see, et need ei ole asjakohased (viimane commit on 2.5 aastat tagasi). Kõige paremini õpibki läbi praktilise kogemuse - päriselt proovida kuidas ründed käivad, et saada aru riskidest ja valesti tehtud lahendustest.

Kas Teie arvates oleks sellisest tööst kasu?

Kindlasti. Kogemus näitab, et praegu ei olda Androidi maailmas turvalisuse poolest väga teadlikud.

Kas Teie arvates on adresseeritavad vead asjakohased või peaks midagi lisama/eemaldama?

Kindlasti on asjakohased. Tasub ka mõelda sellele, et Androidis ei ole ühtset viisi, kuidas asju valesti tehakse. Sama viga võidakse teha paljudel erinevatel viisidel.

Kas Te arvate, et programmeerijad kasutaksid sellist raamistikku, kui oleks aeg, koht ning võimalus?

Kogemus näitab, et arendajad tihti ise sellist asja kätte ei võta. Taolisi veebirakendusi ja veebirakenduste turvaga seotud maailma materjale on palju, kuid arendajad ise neid läbi töötama ei lähe. Eksperdid (mitte hobiprogejad) ei mõtle tihti turvaaspektidele vaid pigem implementatsioonile endale. Oleks vaja hooandjat - näiteks tööandja pushimist.

Ekspert 3.

I think it is a very good topic and it is definitely worth looking into it. There are quite a few vulnerable web applications out there, but so far, I haven´t seen any good for Mobile Apps.

Also, most of the applications focus on being vulnerable but not on how to fix the issues.

To your knowledge, there anything similar available at the moment?

So yes, I think this topic is needed and I personally have not seen such work on Mobile Applications before, directed to developers. (There are some, with courses, directed to Security people and Penetration Testing practices)

Do you think this kind of tool would be useful?

I think it would be useful for developers to see the errors and fixes as well. And if they are able to play it through, maybe something will stick as well.

Do you think the vulnerabilities mentioned are relevant?

I think it is a good start and you will find more and relevant topics on Mobile Applications when you dive into existing work. Errors in principal are quite similar to web application ones, but you have differences in how Android operating system works, what are the practices of storing keys and other secret data etc. I think these specifics could be included in the work as well.

Do you think the developers would use this tool if they had time and opportunity?
Some will, some wont. People who generally care about what they are doing and want to learn would use it. Specially, when it more interactive than just list of requirements or some other form of checklists. But you will always have some developers who don´t want to learn and there is nothing you can do about that.

Appendix 2 – Sentences used in MySQL database creation and updates

```
CREATE TABLE Team(team_id int NOT NULL AUTO_INCREMENT, name NOT NULL
VARCHAR(50) UNIQUE, PRIMARY KEY (team_id));
CREATE TABLE Actor_type(actor_type_id int NOT NULL AUTO_INCREMENT, `Type`
VARCHAR(50) NOT NULL UNIQUE, PRIMARY KEY (actor_type_id));
CREATE TABLE actor(actor_id int NOT NULL AUTO_INCREMENT, actor_type_id int NOT
NULL,reports_to_actor_id int, actor_name VARCHAR(50) NOT NULL, identificator
VARCHAR(25) NOT NULL UNIQUE, avatar_uri VARCHAR(50), score double(5,2), PRIMARY
KEY (actor_id), FOREIGN KEY (actor_type_id) REFERENCES
ActorType(actor_type_id), FOREIGN KEY (reports_to_actor_id) REFERENCES actor
(actor_id));
CREATE TABLE message(message_id int NOT NULL AUTO_INCREMENT, from_actor int
NOT NULL, to_actor int NOT NULL, message TEXT NOT NULL, time_stamp TIMESTAMP
DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP, PRIMARY KEY
(message_id), FOREIGN KEY(from_actor) REFERENCES actor (actor_id), FOREIGN
KEY(to_actor) REFERENCES actor(actor_id));
CREATE TABLE Vacation_type(vacation_type_id int NOT NULL AUTO_INCREMENT, `Type`
VARCHAR(25) NOT NULL UNIQUE, PRIMARY KEY (vacation_type_id));
CREATE TABLE Vacation(vacation_id int NOT NULL AUTO_INCREMENT,
requested_by_actor_id INT NOT NULL, reviewer_actor_id INT, vacation_type_id
int NOT NULL, approved Boolean, from_date date NOT NULL, to_date date NOT NULL,
description TEXT, PRIMARY KEY (vacation_id), FOREIGN KEY
(requested_by_actor_id) REFERENCES actor (actor_id), FOREIGN KEY
(reviewer_actor_id) REFERENCES actor (actor_id), FOREIGN KEY
(vacation_type_id) REFERENCES Vacation_type(Vacation_type_id));
CREATE TABLE credentials(credentials_id int AUTO_INCREMENT, username
VARCHAR(25) UNIQUE, passwordhash VARCHAR(256), actor_id int, PRIMARY KEY
(credentials_id),FOREIGN KEY (actor_id) REFERENCES actor (actor_id));CREATE
TABLE In_team(in_team_id int NOT NULL AUTO_INCREMENT, team_id INT NOT NULL,
actor_id INT NOT NULL, from_date date NOT NULL, to_date date, PRIMARY KEY
(in_team_id), FOREIGN KEY (team_id) REFERENCES Team (team_id), FOREIGN KEY
(actor_id) REFERENCES Actor(actor_id));
CREATE TABLE Work_type(work_type_id int NOT NULL AUTO_INCREMENT, `Type`
VARCHAR(50) NOT NULL UNIQUE, PRIMARY KEY (work_type_id));
CREATE TABLE Position (position_id int NOT NULL AUTO_INCREMENT, position_title
VARCHAR(50) NOT NULL UNIQUE, price_per_hour DOUBLE(5,2) NOT NULL, salary_coef
DOUBLE(2,2) NOT NULL, PRIMARY KEY (position_id));
CREATE TABLE In_position(in_position_id int NOT NULL AUTO_INCREMENT,
position_id INT NOT NULL, actor_id INT NOT NULL, from_date DATE NOT NULL,
to_date DATE, PRIMARY KEY (in_position_id), FOREIGN KEY (position_id)
REFERENCES Position (position_id), FOREIGN KEY (actor_id) REFERENCES Actor
(actor_id));
CREATE TABLE work_unit(work_id INT NOT NULL AUTO_INCREMENT, work_type_id INT
NOT NULL, in_position_id INT NOT NULL, in_team_id INT NOT NULL, done_date DATE
NOT NULL, time_spent DOUBLE(3,1) NOT NULL, task_number VARCHAR(10) NOT NULL,
description TEXT, reviewer_actor_id INT, approved BOOLEAN, PRIMARY KEY
(work_id), FOREIGN KEY (work_type_id) REFERENCES Work_type(work_type_id),
FOREIGN KEY (in_position_id) REFERENCES In_position(in_position_id), FOREIGN
KEY (in_team_id) REFERENCES In_team (in_team_id), FOREIGN
KEY(reviewer_actor_id) REFERENCES Actor(actor_id));
INSERT INTO Actor_Type(`Type`) VALUES ('Employee');
INSERT INTO Actor_Type(`Type`) VALUES ('Manager');
INSERT INTO Actor_Type(`Type`) VALUES ('Client');
INSERT INTO Actor_Type(`Type`) VALUES ('Client_representative');
INSERT INTO Work_type(`Type`) VALUES ('Testing');
INSERT INTO Work_type(`Type`) VALUES ('Analysis');
INSERT INTO Work_type(`Type`) VALUES ('Development');
INSERT INTO Work_type(`Type`) VALUES ('Administrational work');
INSERT INTO Vacation_type(`Type`) VALUES ('Regular');
INSERT INTO Vacation_type(`Type`) VALUES ('Study');
INSERT INTO Vacation_type(`Type`) VALUES ('Maternity');
INSERT INTO Vacation_type(`Type`) VALUES ('Other');
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES( 2, NULL, "Andrew Williams", "93840283940", null, 9.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 1, "Jonathan Wilson", "93840283941", null, 90.9);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri`, `score`)
VALUES(1, 1, "Zac Robertson", "93840283942", null, 90.9);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES( 1, 6, " Evangeline Ponce", "93840283944", null, 8.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES( 2, NULL, "Hugo Lawson", "93840283943", null, 87.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(3, NULL, "Thornton Corp", "BN1129479322034", null, 0.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(3, NULL, "Russell Industries", "BN112947932438902", null, 80.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(3, NULL, "Read Read Read", "BN11294793221232233", null, 86.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(4, NULL, "Esme Daniels", "43892749028", null, 86.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(4, NULL, "Jenna Fields", "43892749078", null, 46.99);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(4, NULL, "Luke Mccann", "33892748778", null, 96.00);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(4, NULL, "Hasan Carver", "33892748789", null, 36.00);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, NULL, "Elizabeth Brown ", "33892748769", null, 56.00);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 1, "Samantha Davis", "33892748766", null, 96.00);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 6, "Stephanie Blaese", "33892483927", null, 16.00);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 23, "Jessica Wilson", "33829473846", null, 43.09);
INSERT  INTO  actor(`actor_type_id`,  `reports_to_actor_id`,  `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 6, "David Moore", "32829473846", null, 73.92);
```

```
INSERT   INTO   actor(`actor_type_id`,   `reports_to_actor_id`,   `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 1, "William Smith", "32829473096", null, 86.5);
INSERT   INTO   actor(`actor_type_id`,   `reports_to_actor_id`,   `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 23, "Michael Blaese", "328294728391", null, 54.3);
INSERT   INTO   actor(`actor_type_id`,   `reports_to_actor_id`,   `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(1, 1, "Joshua Jones", "3282943048391", null, 93.3);
INSERT   INTO   actor(`actor_type_id`,   `reports_to_actor_id`,   `actor_name`,
`identificator`, `avatar_uri` ,`score`)
VALUES(2, NULL, "Bailey Mann", "328292937482", null, 84.22);
INSERT INTO Team(`name`) VALUES("Thornton Corp");
INSERT INTO Team(`name`) VALUES("Russell Industries");
INSERT INTO Team(`name`) VALUES("Read Read Read");
INSERT INTO Team(`name`) VALUES("("Russell Industries 2");
INSERT INTO Team(`name`) VALUES("Admin team");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Junior Business Analyst", "60.0", "0.10");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Business Analyst", "70.0", "0.20");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Senior Business Analyst", "100.0", "0.25");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Senior System Analyst", "120.0", "0.25");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("System Analyst", "90.0", "0.20");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Junior System Analyst", "70.0", "0.10");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Junior Quality Engineer", "50.0", "0.15");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Quality Engineer", "60.0", "0.15");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Senior Quality Engineer", "90.0", "0.3");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Junior Software Developer", "60.0", "0.15");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Software Developer", "75.0", "0.25");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Senior Software Developer", "100.0", "0.35");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Product Owner", "130.0", "0.35");
INSERT INTO Position (position_title, price_per_hour, salary_coef) VALUES("IT
support", "60.0", "0.28");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Office Assistant", "30.0", "0.2");
INSERT   INTO   Position   (position_title,   price_per_hour,   salary_coef)
VALUES("Recruiter", "50.0", "0.2");

INSERT   INTO   In_team   (team_id,   actor_id,   from_date)   VALUES("2",   "1",
"15.05.20");
INSERT   INTO   In_team   (team_id,   actor_id,   from_date)   VALUES("5",   "1",
"15.03.15");
INSERT   INTO   In_team   (team_id,   actor_id,   from_date)   VALUES("2",   "3",
"13.04.01");
INSERT   INTO   In_team   (team_id,   actor_id,   from_date)   VALUES("2",   "5",
"13.04.01");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("1", "6", "15.1.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("3", "6", "12.12.12");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("1", "7", "13.2.24");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("5", "15", "15.2.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("3", "16", "13.1.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("1", "17", "10.10.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("4", "18", "10.5.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("1", "19", "10.10.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("2", "20", "17.11.11");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("4", "21", "13.1.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("5", "22", "16.10.11");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("4", "23", "09.12.11");

INSERT INTO In_team (team_id, actor_id, from_date) VALUES("1", "11", "10.10.1");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("2", "12", "15.05.10");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("3", "14", "12.10.11");
INSERT INTO In_team (team_id, actor_id, from_date) VALUES("5", "13", "15.2.1");
INSERT INTO in_team (team_id, actor_id, from_date) VALUES("1", "8", "10.10.1");
INSERT INTO in_team (team_id, actor_id, from_date) VALUES("2", "9", "15.05.10");
INSERT INTO in_team (team_id, actor_id, from_date) VALUES("3", "10", "12.10.11");
INSERT INTO in_team (team_id, actor_id, from_date) VALUES("5", "9", "12.10.11");



INSERT INTO In_position (position_id, actor_id, from_date) VALUES("13", "1", "15.05.20");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("11", "3", "15.03.15");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("8", "5", "13.04.01");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("13", "6", "15.1.1");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("4", "7", "13.2.24");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("11", "15", "15.2.1");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("11", "16", "13.1.1");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("12", "17", "10.10.1");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("15", "18", "10.5.1");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("12", "19", "10.10.1");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("9", "20", "17.11.11");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("14", "21", "13.1.1");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("10", "22", "16.10.11");
INSERT INTO In_position (position_id, actor_id, from_date) VALUES("16", "23", "09.12.11");

INSERT INTO credentials(username, passwordhash, actor_id) VALUES("andreww", "admin", 1);
INSERT INTO credentials(username, passwordhash, actor_id) VALUES("jonathanw", "root", 3);
INSERT INTO credentials(username, passwordhash, actor_id) VALUES("zacr", "password123", 5);
INSERT INTO credentials(username, passwordhash, actor_id) VALUES("hugol", "password", 6);
INSERT INTO credentials(username, passwordhash, actor_id) VALUES("evangelinep", "password567", 7);
```

Appendix 3 – Example web service requests and responses

GET REQUESTS:

“/allEmployees”

Example request: request does not require any parameters.

Example response:

```
[
  {
    "id": 1,
    "name": "Andrew Williams",
    "position": {
      "jobTitle": "Product Owner",
      "standardPricePerHour": 0.0,
      "salaryCoefficient": 0.0
    },
    "teams": null,
    "score": 9.99,
    "ssn": "93840283940",
    "iban": "NL13ABNA9071327418"
  },
  {
    "id": 3,
    "name": "Jonathan Wilson",
    "position": {
      "jobTitle": "Software Developer",
      "standardPricePerHour": 0.0,
      "salaryCoefficient": 0.0
    },
    "teams": null,
    "score": 90.9,
    "ssn": "93840283941",
    "iban": "NL62ABNA5528005485"
  }
]
```

“/reportedHours”

Example request: `/reportedHours?id=6`

Example response:

```
[
  {
    "id": 2,
    "date": null,
    "hoursSpent": 5.0,
    "team": {
      "name": "Russell Industries",
      "client": {
        "name": "Russell Industries"
      }
    },
    "category": "Administrational work",
    "taskID": "463643646",
    "description": "",
    "employee": {
      "id": 6,
      "name": "Hugo Lawson",
      "position": null,
      "teams": null,
      "score": 0.0,
      "ssn": null,
      "iban": null
    },
    "worker": {
      "id": 6,
      "name": "Hugo Lawson",
      "position": null,
      "teams": null,
      "score": 0.0,
      "ssn": null,
      "iban": null
    }
  }
]
```

"/requestedVacations"

Example request: `/requestedVacations?id=6`

Example response:

```
[
  {
    "id": 1,
    "reviewer": null,
    "requester": {
      "id": 6,
      "name": "Hugo Lawson",
      "position": null,
      "teams": null,
      "score": 0.0,
      "ssn": null,
      "iban": null
    },
    "type": "Regular",
    "approved": null,
    "fromDate": "12.03.2019",
    "toDate": "21.03.2019",
    "description": "any description"
  }
]
```

"/messages"

Example request: `/messages?id=6`

Example response:

```
[
  {
    "messageId": 3,
    "to": 6,
    "from": {
      "id": 1,
      "name": "Andrew Williams",
      "position": null,
      "teams": null,
      "score": 0.0,
      "ssn": null,
      "iban": null
    },
    "timestamp": "2019-03-07 15:24:47.0",
    "message": "This is something urgent!",
    "employeeTo": 6
  },
  {
    "messageId": 4,
    "to": 6,
    "from": {
      "id": 1,
      "name": "Andrew Williams",
      "position": null,
      "teams": null,
      "score": 0.0,
      "ssn": null,
      "iban": null
    },
    "timestamp": "2019-03-08 11:00:48.0",
    "message": "Hi Hugo!",
    "employeeTo": 6
  }
]
```

"/employeeDetails"

Example request: `/employeeDetails?id=6`

Example response:

```
{
  "id": 6,
  "name": "Hugo Lawson",
  "position": {
    "jobTitle": "Product Owner",
    "standardPricePerHour": 130.0,
    "salaryCoefficient": 0.35
  },
  "teams": [
    {
      "name": "Read Read Read",
      "client": {
        "name": "Read Read Read"
      }
    },
    {
      "name": "Thornton Corp",
      "client": {
        "name": "Thornton Corp"
      }
    }
  ],
  "score": 87.99,
  "ssn": "93840283943",
  "iban": NL17ABNA5597655787
}
```
"/reportedHoursReview"

Example request: `/reportedHourReview?managerId=6`

Example response:

```
[
  {
    "id": 1,
    "date": null,
    "hoursSpent": 4.0,
    "team": {
      "name": "Read Read Read",
      "client": {
        "name": "Read Read Read"
      }
    },
    "category": "Testing",
    "taskID": "5",
    "description": null,
    "employee": {
      "id": 7,
      "name": "Evangeline Ponce",
      "position": null,
      "teams": null,
      "score": 0.0,
      "ssn": null,
      "iban": null
    },
    "worker": {
      "id": 7,
      "name": "Evangeline Ponce",
      "position": null,
      "teams": null,
      "score": 0.0,
      "ssn": null,
      "iban": null
    }
  }
]
```
"/requestedVacationsReview"

Example request: `/requestedVacationsReview?managerId=6`

Example response:

```
[
  {
    "id": 1,
    "reviewer": null,
    "requester": {
      "id": 7,
      "name": "Evangeline Ponce",
      "position": null,
      "teams": [
        {
          "name": "Thornton Corp",
          "client": {
            "name": "Thornton Corp"
          }
        }
      ],
      "score": 8.99,
      "ssn": "93840283944",
      "iban": "EE471261725194496268"
    },
    "type": "Regular",
    "approved": null,
    "fromDate": "21.03.2019",
    "toDate": "12.03.2019",
    "description": ""
  }
]
```

"/vacationTypes"

Example request: request does not require any parameters.

Example response:

```
[
  "Maternity",
  "Other",
  "Regular",
  "Study"
]
```

"/workCategories"

Example request: request does not require any parameters.

Example response:

```
[
  "Administrational work",
  "Analysis",
  "Development",
  "Testing"
]
```

"/removeMessage"

Example request: `removeMessage?id=4`

Example response: Response contains only information about request success. No additional data is added.

POST REQUESTS:

"/login"

Example request:

```
{
      "password":"password",
      "username":"username"
}
```

Example response:

```
{
       "sessionId":"mangerid123",
       "employeeId":225,
       "isManager":true,
       “name”:”Employee Name”
}
```

“/reportHours”

Example request:

```
       {
  "category": "Development",
  "description": "any comment about the task done",
  "hoursSpent": 5,
  "taskID": "38472",
  "team": {
    "client": {
      "name": "Read Read Read",
      "score": 0.0
    },
    "name": "Read Read Read"
  },
  "date": "8.3.2019",
  "worker": {
    "id": 6,
    "name": "Hugo Lawson",
    "position": {
      "jobTitle": "Product Owner",
      "salaryCoefficient": 0.35,
      "standardPricePerHour": 130.0
    },
    "score": 87.99,
    "ssn": "93840283943",
    "teams": [
      {
        "client": {
          "name": "Read Read Read",
          "score": 0.0
        },
        "name": "Read Read Read"
      },
      {
        "client": {
          "name": "Thornton Corp",
          "score": 0.0
        },
        "name": "Thornton Corp"
      }
    ]
  }
}
```

Example response: Response contains only information about request success. No additional data is added.

“/requestVacation”

Example request:

```
{
  "description": "",
  "fromDate": "21.3.2019",
  "requester": {
    "id": 6,
    "name": "Hugo Lawson",
    "position": {
      "jobTitle": "Product Owner",
      "salaryCoefficient": 0.35,
      "standardPricePerHour": 130.0
    },
    "score": 87.99,
    "ssn": "93840283943",
    "teams": [
      {
        "client": {
          "name": "Read Read Read",
          "score": 0.0
        },
        "name": "Read Read Read"
      },
      {
        "client": {
          "name": "Thornton Corp",
          "score": 0.0
        },
        "name": "Thornton Corp"
      }
    ]
  },
  "toDate": "25.3.2019",
  "type": "Study"
}
```

Example response: Response contains only information about request success. No additional data is added.

"/sendMessage"

Example request:

```
{
  "from": {
    "iban": "NL17ABNA5597655787",
    "id": 6,
    "name": "Hugo Lawson",
    "position": {
      "jobTitle": "Product Owner",
      "salaryCoefficient": 0.0,
      "standardPricePerHour": 0.0
    },
    "score": 87.99,
    "ssn": "93840283943"
  },
  "message": "Hi Zac!",
  "to": 5
}
```

Example response: Response contains only information about request success. No additional data is added.

Appendix 4. Feedback questionnaire

How much experience do you have in Android development (in scale 1-10) + years of experience?
Overall IT experience (in years)?
How much experience do you have in security (in scale 1-10) + years of experience?
How useful the **workshop** was for you (in scale 1-10)?
Which part was the most useful/you liked the best?
What should be changed? What would you do differently?
Would you attend Android Security Workshop part 2? Why?
Any other comments/suggestions?
The feedback questionnaire can be accessed in https://forms.gle/qzet75yL2qJrSPhL9

Appendix 5. Local backend setup

Step 1. A user has to download following compose repository - https://github.com/mmatlik/sec-compose

Step 2. Docker can be downloaded from the following url: https://www.docker.com/get-started. Run docker and log in.

Step 3. Open terminal and navigate to sec-compose repository that was downloaded in step 1.

Step 4. Run the following command: `docker-compose up -d sec-service`

Appendix 6. Training feedback from Head of Cyber Security of a software development company

Security officer in a software development company. Previously CERT member in big telco and software security in big international corporation:
“I participated in the training to learn myself and see how developers react to this as well. It seemed, that training is well put together and presenter was prepared. Developers did get into discussion and started thinking about what can go wrong. And that is most important. Also, they were engaged in finding errors and trying to propose ideas how to mitigate it. It was mostly attended by seniors / persons who have few years of development experience as well.

As for the content, I think it was relevant and new. I myself didn't knew most of the things mentioned since I am not working with anything Android specific. It was well made and showed developers, how one missing line or default value of a parameter can create a big mess.”