ESARA: A Framework for Enterprise Smartphone Apps Risk Assessment

Majid Hatamian[1], Sebastian Pape[1,2][0000−0002−0893−7856], and Kai Rannenberg[1]

[1] Chair of Mobile Business & Multilateral Security
Goethe University Frankfurt, Germany
{majid.hatamian,sebastian.pape,kai.rannenberg}@m-chair.de
[2] Chair of Information Systems
University of Regensburg, Germany

Abstract Protecting enterprise's confidential data and infrastructure against adversaries and unauthorized accesses has been always challenging. This gets even more critical when it comes to smartphones due to their mobile nature which enables them to have access to a wide range of sensitive information that can be misused. The crucial questions here are: How the employees can make sure the smartphone apps that they use are trustworthy? How can the enterprises check and validate the trustworthiness of apps being used within the enterprise network? What about the security and privacy aspects? Are the confidential information such as passwords, important documents, etc. are treated safely? Are the employees' installed apps monitoring/spying the enterprise environment? To answer these questions, we propose *Enterprise Smartphone Apps Risk Assessment (ESARA)* as a novel framework to support and enable enterprises to analyze and quantify the potential privacy and security risks associated with their employees' installed apps. Given an app, *ESARA* first conducts various analyses to characterize its vulnerabilities. Afterwards, it examines the app's behavior and overall privacy and security perceptions associated with it by applying natural language processing and machine learning techniques. The experimental results using app behavior and perception analyses indicate that: (1) *ESARA* is able to examine apps' behavior for potential invasive activities; and (2) the analyzed privacy and security perceptions by *ESARA* usually reveal interesting information corresponding to apps' behavior achieved with high accuracy.

Keywords: smartphone · app · security · privacy · risk · enterprise.

1 Introduction

The amount of available apps for smartphones seems to be almost endless. The developers range from spare time developers to large companies. However, none of the app stores offers a dedicated security or privacy score for those apps. This does not only challenge individuals but also companies. However, smartphones are often used for personal matters and official business. *Bring your own device (BYOD)* is an attractive employee IT ownership model that enables employees

to bring and use their personal devices in enterprises. Such a model provides more flexibility and productivity for the employees, but may impose some serious privacy and security risks. This way not an administrator decides about the installation of apps but the user. Similar problems arise if users are allowed to install apps on the devices provided by the company. The problem raises when enterprise's confidential data is endangered as smartphones now being used to access enterprise email, calendars, apps and data. As a result, enterprises are facing the tricky task of protecting valuable data from threats such as data leakage and malware. As a consequence, it is quite challenging for enterprises to balance both their employees' needs and their security concerns. But even if the employees are not allowed to decide by themselves, then the decision would have to be made by the IT department. As a consequence, enterprises would have to provide black lists that contain apps that are not allowed to be used, or white lists that contain apps allowed for use. Grey lists may be established to list apps, where no decision was made. In any case either the IT department needs to make decisions which app belongs to which list or the employees need to make their own decisions, whether a specific app is to be used. Decisions will be made as a trade off between the necessity of the app for business purposes and the risk with regard to enterprise assets.

Our Work: In this paper, we propose *Enterprise Smartphone Apps Risk Assessment (ESARA)* as a novel framework aimed at supporting enterprises to protect their data against adversaries and unauthorized accesses. Our framework eases the process of privacy and security risk assessment for the use of smartphone apps. To achieve this goal, we propose two concepts regarding the privacy and security assessment of smartphone apps namely app *Behavior Analyzer (BA)* and app *Perception Analyzer (PA)*. We develop these two concepts along with two essential requirements namely *vulnerability checker* and *malware checker* that are cooperated with each other aiming at supporting enterprises to discriminate privacy and security misbehaviors. To the best of our knowledge, we are the first proposing the combination of these concepts and requirements that are jointly working with each other. Through experiments and implementations, we investigate how efficient and reliable the newly proposed concepts are.

Outline: The rest of this paper is organized as follows. Section 2 reviews the existing works in the area of smartphone app privacy and security preservation for general and enterprise use cases. In Section 3 the respective components and architecture of *ESARA* framework are presented. Section 4 elaborates on the main results obtained from the evaluation of different components of *ESARA* and highlights the key insights. Finally, we present the main conclusions in Section 5.

2 Related Work

In this section, we provide an overview of the relevant related work in the area of privacy and security enhancement in smartphone ecosystems and enterprise environments.

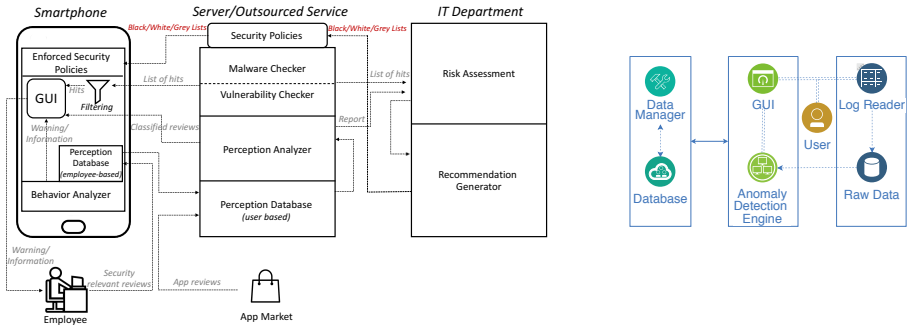
Agarwall and Hall [16] propose an approach called *ProtectMyPrivacy (PMP)* for iOS devices to detect access to private information at run-time and protect users by substituting anonymized data to be sent instead of sensitive information. Enck et al. [19] proposed *TaintDroid* for real-time tracking of information flows of smartphone apps. By focusing on personal resources, the system can reveal the manipulation or transfer of sensitive data and thus analyze the app's behavior. The monitoring procedure is based on identifying privacy-related information sources and labeling associated data. Moreover, other impacted data are tracked and identified before being transferred outside the system. The evaluation on 20 popular apps showed data leakage, e.g. phone identifier, location information and phone number being transferred to remote advertising servers. The *Apex* mechanism [25] is a name for an additional component for Android which enables users to selectively allow, deny or limit access to specific permissions requested by apps. Beresford et al. [17] propose an approach called *Mockdroid* to substitute private data with mock data when they are asked to be accessed by installed apps. *TISSA* [28] is another component for Android that enables user to choose a list of untrusted apps, and based on this list it provides mock data in place of private data at run-time. Appicaptor [14] is a framework that helps enterprises for app risk management. The goal of Appicaptor is to detect the potential privacy and security risks associated with mobile app by benefiting from static analysis of app binaries. Based on app's behavior, a ranking list is provided to classify apps into white and black lists. BizzTrust [9] is another framework that suggested the use of restricted and open areas on the employee's smartphone. Both approaches are mainly focused on security risks resulted from malicious apps.

We believe that one efficient solution should not only be focused on security behavior of mobile apps, but also privacy behavior. Importantly, consideration of users' perception about the behavior of apps plays an important role to have a more comprehensive solution. These are interesting works, but neither of them focuses on a comprehensive solution that fulfills the essential requirements of enterprise environment. In our work, we propose a solution that enhances the existing works and revamps the current enterprise app risk assessment models.

3 ESARA Framework

3.1 Goal and Requirements

Our framework makes use of different approaches from literature and combines them with our app behavior analyzer and our app perception analyzer to get a more realistic and holistic picture of installed apps. E.g., a malware checker does not detect data leakages or vulnerabilities in an app and a vulnerability scanner does not detect malicious behavior. Requirements for the development of *ESARA* were: (1) Reusing existing approaches; (2) Limiting the effort needed (since there is a large number of apps); (3) Scalability in the way that it should be easy to rely on external services and allowing several companies to share a same infrastructure; (4) Independence from app markets since even after several years



(a) ESARA (b) Behavior analyzer

Figure 1: High Level Overviews

none of them offers a decent security or privacy score; (5) Involving employees for feedback when using an app; (6) Involving employees for decisions.

3.2 Architecture Design

Fig. 1a shows an overview of the proposed architecture for *ESARA*. As can be seen, *ESARA* consists of three main modules: employee's smartphone, server and enterprise IT department. On the employee's device an app is running which analyzes the behavior of a certain installed app and ultimately communicates the results to the employee. It also stores the employee's security and privacy perception and receives results regarding the perception analysis and risk assessment from the other two modules. The server or an outsourced service is supposed to check apps for vulnerabilities and malicious activities by running a malware and vulnerability scanner, therefore, it does not collect any data from employees. This server/service is also responsible to analyze employees' and other users' perception about security and privacy behavior of apps. If security policies are put in place, black, white and gray lists can also be stored here. The enterprise IT department takes the final decisions about which app is to place on which list – either manually or automatically by defining certain rule sets.

3.3 Components

Malware Checker The impact of infection by a malware can be huge ranging from enterprise's infrastructure to the entire network. This component ensures the protection of enterprise's confidential data against malware. Therefore, we should not neglect the importance of this aspect of mobile apps while designing the *ESARA*'s architecture. As deploying a malware checker on resource constrained smartphones can be challenging [18], we propose the use of malware checker within the cloud as it has more computational resources. Therefore, this component is running on the server side. Checks need to be repeated with each update of the app or update of the malware checker's signature file.

Vulnerability Checker Vulnerabilities are exploited by hackers to gain access to the device's or enterprise's resources. Statistics and observations showed that mobile platforms are among the most vulnerable operating systems in 2017 [15]. An observation by NowSecure [10] demonstrated that 25% of mobile apps have at least one high risk security vulnerability. Also, the latest security report published by Arxan [11] showed that 59% of the analyzed Android finance apps contained three OWASP mobile top 10 risks [3]. Surprisingly, all the analyzed iOS apps had at least 3 top risks. Due to such shocking statistics, an in-depth vulnerability analysis is required to investigate the potential vulnerabilities imposed by the employees' installed apps. Therefore, we also consider the importance of vulnerability analysis in *ESARA*'s architecture. Similar to the malware checker, this component is also running on the server side. There is an availability of a diverse number of vulnerability checkers both for Android and iOS that can be exploited based on the requirements [2,4,5,6,7,22].

Behavior Analyzer *Behavior Analyzer (BA)* is an extension of our previous work [20] and a monitoring tool that analyzes the behavior of employee's installed apps. In contrast to run-time monitoring, where one could conclude what an employee was doing, we analyze the apps' behavior only by looking at the apps' permission requests. This way, the employees' privacy will be respected, while on the other hand security intrusive apps can be identified. Fig. 1b shows a high level architecture of the *BA* tool. In what follows, we elaborate on the core parts of *BA* and their respective role.

Log Reader The log reader collects the logs from `AppOpsCommand` and it sends a timer to the `PermissionUsageLogger` service periodically. When it is received, the logger queries the AppOps service that is already running on the phone for a list of apps that have used any of the operations we are interested in tracking. We then check through that list and for any app that has used an operation more recently than we have checked, we store the time at which that operation was used. These timestamps are then counted to get a usage count.

Anomaly Detection Engine This component is supposed to behaviorally analyze the installed apps by getting help from the results obtained from the log reader component. This is done according to a rule-based mechanism which is supposed to increase the functionality and flexibility of our approach. Consequently, we have defined a set of invasive behavior detection rules that are aimed to analyze the behavior of employees' installed apps. We initially defined a set of sensitive permissions (introduced by Android[3]) and we mainly analyze the accesses to these resources. The following pseudo-code shows a couple of rules (due to space limitations, we refrained from explaining all the defined rules).

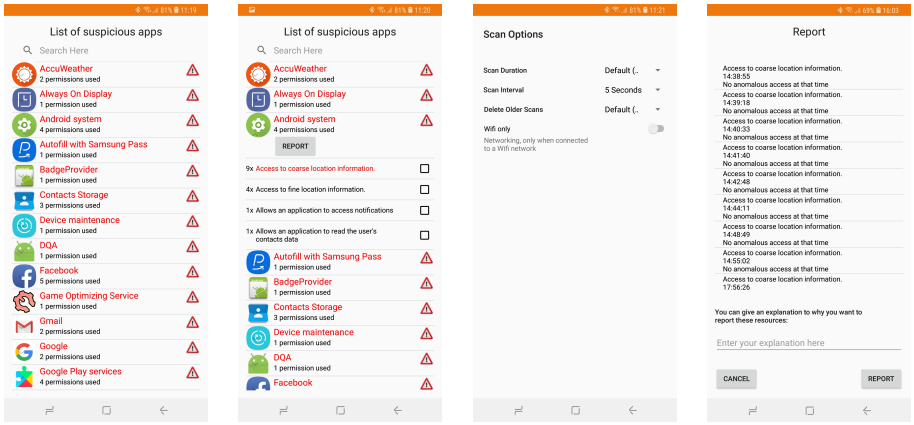
[3] https://developer.android.com/guide/topics/permissions/requesting.html

```
#app is in background and used a critical resource
if((inBackground == 0) && criticalRecources.contains(resource)){
    results.add("1");
    results.add("App was in background but accessed crit. resource");
    return results;}
#when the screen orientation < -3 (e.g. means laying on the desk) and display is off
if((screenOrientation < -3) && (screenState == 0) && (criticalRecources.contains(resource))) {
    results.add("1");
    results.add("Resource was accessed while device seemed to lie on desk");
    return results;}
#when the display is off and critical resource was used, but without the case of taking a phone call
if((criticalRecources.contains(resource)) && (screenState == 0) && !(closeToObject == 0) && !(resource.equals("RECORD_AUDIO"))){
    results.add("1");
    results.add("Screen was off and critical resource was used");
    return results;}
```

While implementing the *BA* tool, we paid special attention to the following elements to discern which resource access might be legitimate (needed by a certain app):

- Device's Orientation: This gives us information about the orientation of the device, e.g., if the screen is down or up. We register a `Listener` who is listening to changes in the accelerometer that ultimately gives us the values for x, y and z.
- Screen State: It describes whether the device's screen is on or off at a certain time. As long as a scan is running, we register a `Receiver` for the events `ACTION_SCREEN_ON` and `ACTION_SCREEN_OFF`.
- Proximity Sensor: The screen state alone, however, is not meaningful enough, as it may happen that the screen is indeed off but certain personal resources may still be accessed (e.g., when talking on the phone, the screen turns off when the phone is approaching the ear, but access to `RECORD_AUDIO` is justified at this time). Therefore, we read the proximity sensor to indicate whether an object is within a defined range of the mobile phone. We access it through `SensorManager` and the associated listener is called as soon as an object enters or leaves the defined range (the range varies from device to device, but on average, it is around 5cm).
- App State: We also consider the app state (at the time of access to a certain resource) as an important element while monitoring the apps' behavior. We distinguish the following app states: `SYSTEM_APP`, `PRE_INSTALLED_APP`, `INACTIVE`, `BACKGROUND` and `FOREGROUND`.

The *BA* tool operates like a watchdog and it only checks whether sensitive device resources are accessed. To protect the employees' privacy, the *BA* does not have the right/capability to access the sensitive data itself or track/monitor employee's activities. Furthermore, privacy controls are given to the employees to selectively choose the information that they want to share with the IT department. In particular, the IT department does not learn about all apps on the employees' device but only about those where the employees submit a report. For the sake



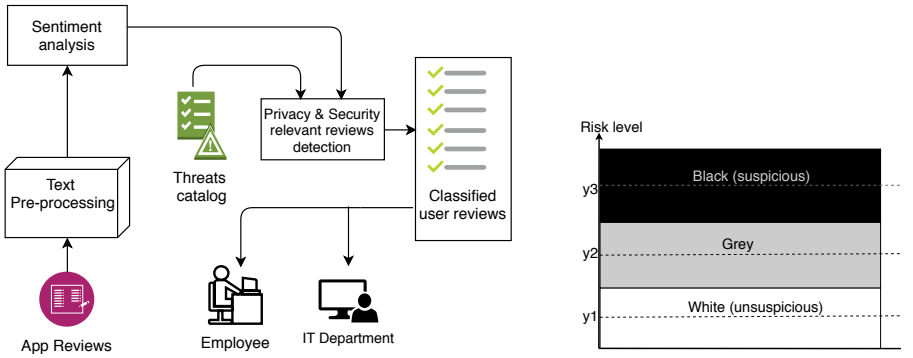
(a) List of Suspicious Apps (b) Details of Analysis (c) Analysis Settings (d) Report Option

Figure 2: The Proposed GUI for the BA Tool.

of user interface design and risk indicator communication to the employees, we designed user interfaces for *BA* as shown in Fig. 2.

Perception Analyzer The *BA* tool enables employees to write (optional) reviews regarding each privacy and security invasive activity that they observe. The main goal of *Perception Analyzer (PA)* is to mine these bunch of reviews to investigate how much privacy and security relevant claims/statements can be extracted that can be ultimately used for the risk assessment component. These self-written reports are sent to the IT security department of the enterprise as well. The main idea is to not only rely on individual's report, but also to consider a high level overview of apps' real behavior. This would enable the enterprise to improve the fairness of their decisions. Additionally, we enriched the reviews of the employees with reviews from app markets (e.g. Google Play). The reviews in app markets are in general more concerned about features and performance of the apps and only little of them contain comments about security or privacy. However, since for some of the apps there are tons of reviews, even a low percentage of reviews dealing with security and privacy can be helpful. Therefore, we used a machine learning approach to find the relevant reviews. Fig. 3a shows the proposed architecture for *PA*.

The *app reviews* are first pre-processed in *text pre-processing* component using typical natural language processing (NLP) techniques (e.g. tokenization, stemming and removing stop words). Further, we propose the use of *sentiment analysis* techniques to find both positive and negative reviews that talk about privacy and security aspects of apps. Afterwards, the machine learning model comes in and *threat catalog* helps to identify the associated threats with each



(a) Privacy and Security Reviews Analyzer (b) Risk Assessment in *ESARA*

Figure 3: Review Analyzer and Risk Assessment

Table 1: Identified Threats

#	Threat	Description
T1	Tracking & Spyware	Allows an attacker to access or infer personal data to use it for marketing purposes, such as profiling or targeted ads.
T2	Phishing	An attacker collects user credentials (e.g. passwords and credit card numbers) by means of fake apps or messages that seem genuine.
T3	Unauthorized Charges	The hidden and unauthorized charges through registration to a premium service AND/OR installation a certain app.
T4	Unintended Data Disclosure	Users are not always aware of all the functionality of smartphone apps. Even if they have given explicit consent, users may be unaware that an app collects and publishes personal data.
T5	Targeted Ads	Refers to unwanted ads and push notifications.
T6	Spam	Threat of receiving unsolicited, undesired or illegal messages. Spam is considered an invasion of privacy. The receipt of spam can also be considered a violation of our right to determine for ourselves when, how, and to what extent information about us is used.
T7	General	Comprises all the threats that are not categorized into other categories, e.g. permission hungry apps, general security concerns, etc.

user review by getting help from *privacy and security relevant reviews detection*. Finally the *classified reviews* are communicated to the *IT department* for risk assessment procedure. As it is obvious, *BA* is supposed to tell the IT security department how good/bad is a certain app in terms of privacy and security aspects based on its behavior in reality, and *PA* is aimed at providing a fair comparison by considering a consensus from employees and crowdsources. We detect not only a privacy and security relevant user review, but also determine the threat hidden in it. To this end, we performed a literature research [3,12,13,21,23] in order to identify the most relevant threats in the context of smartphone ecosystems. These threats are used as the input for the supervised classification algorithm as described in Table 1.

Risk Assessment Risk assessment examines the potential privacy and security risks associated with each employee's installed app. Therefore, it is highly dependent on the results obtained from *BA*, *PA*, *malware checker* and *vulnerability*

checker. In this paper, we assume three different risk levels, including black (seems suspicious), grey (requires more investigation) and white (seems unsuspicious) as shown by Fig. 3b. If a certain app does not successfully pass the investigations done by malware and vulnerability checkers, then it is automatically ranked as black and the outcome will be communicated to the employee. Otherwise, the risk assessment considers the real behavior and overall perception results in order to provide the recommendation generator with sufficient decision making information. It is worth mentioning that our main focus is on the grey risk level.

Recommendation Generator Recommendation generator gets the input from the risk assessment component. It helps the IT security departments to better classify apps as allowed or not allowed (e.g. blacklists and whitelists). Thus, it ranks similar functionality apps, i.e. those apps that have similar functionality (e.g. weather forecasting apps, navigation apps, etc.) are assigned ranks based on the analysis done by risk assessment. Moreover, it maintains a history of privacy and security behavior records (analyses) based on apps' versions, meaning that once a certain installed app is updated, a trend containing the behavior measurements related to the current and older versions will be issued. Therefore, the IT security department can follow and analyze the trend analyses done by *ESARA* from version to version. This would enable them to analyze the behavior of current versions and compare it with older versions. This is a substantial impact of *ESARA* that is based on the fact that there is no guarantee for privacy and security friendly apps to behave nicely in the future.

4 Evaluation

Since the efficiency of malware and vulnerability scanner is a topic of its own, we do not discuss it here. However, we discuss the results of the app behavior analyzer and the app perception analyzer. For the overall evaluation, we will discuss which component covers which kind of risk.

4.1 App Behavior Analysis Results

To evaluate the applicability of *BA* and its importance in the overall performance of *ESARA*, we demonstrate some initial results regarding the behavior analysis done by *BA*. To make our scope as narrow as possible and to have a fair analysis, we mainly focused on Android apps and chose one general purpose app category. To this end, we found *Health & Fitness* as the most interesting option that is widely used by people and has raised serious privacy and security concerns [24,27]. Therefore, we selected the top 20 apps in the *Health & Fitness* category and started the case study. We purchased six Android smartphones and installed all the 20 apps on each of them. We then used *BA* to analyze the behavior of the aforementioned apps. While we were implementing the case study, the *BA* tool was running in the background the whole time (i.e. it was monitoring apps' behavior). We ran the apps once and let them to be executed in the background.

Table 2: Resource Access Behavior Pattern Extracted by *BA*.

Resources	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
STORAGE	561	106	143	106	175	103	124	196	702	186	53	394	137	87	156	747	95	331	184	1376
CAMERA	0	0	0	0	0	0	0	25	53	86	0	0	0	0	0	0	0	15	0	14
READ_SMS	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0
READ_CONTACTS	0	62	0	0	0	0	0	53	31	653	0	0	34	0	0	0	0	0	0	142
LOCATION	0	32	0	0	0	0	985	183	650	403	217	0	116	96	412	3780	0	670	566	1526
PHONE_STATE	0	0	0	35	0	0	284	0	534	87	0	0	0	0	0	0	0	0	0	0
MICROPHONE	0	0	0	0	0	0	0	0	0	0	0	0	0	552	0	0	34	0	0	0
GET_ACCOUNTS	0	126	0	0	0	0	93	407	279	363	0	0	0	0	0	0	0	0	101	455
BODY_SENSOR	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Thus, we never interacted with the mobile devices during the experiment period (ranging from May 19, 2018 to May 28, 2018). This is mainly because mobile apps are task-specific and expected to only access resources when needed for their functionality. When the employee is using an app, it is harder to infer whether the app needs to have access to a certain sensitive resource as this requires to know what exactly the employee is doing which may violate his/her privacy. But when the app is not used, it is easy to detect non-security friendly sensitive resource accesses, e.g. access to enterprise's confidential data (e.g. employees' calendar, contacts, ...), since most of them will be unsolicited. Afterwards, we collected and analyzed the data generated by *BA*. In total, nine sensitive resources were accessed by the apps. The results of the analysis for each app and resources are shown in Table 2. The numbers in each cell show the number of times that each app accessed a certain resource.

In our case study we recorded every invasive behavior of the app if it *accessed resources besides the storage* or *accessed resources which are not required for the app functionality*. E.g., a health-based app traditionally needs access to `BODY_SENSOR` (e.g. App 2). The accesses to `READ_STORAGE` are not surprising because the smartphones were not completely turned off and all apps could read and write files placed on the external storage (e.g. cache files). However, five apps accessed `CAMERA` (apps 8, 9, 10, 18 and 20). These accesses are not privacy-friendly, since the user does not know that the app currently accesses the camera. Furthermore, `READ_CONTACTS` was accessed by six apps. In general, such accesses to the contacts should not be done by apps. In our case the apps are health-based, where it is not clear why they need access to the user's contacts. We assume accesses to `LOCATION` must not be a problem for the user's privacy in general since this permission is not extremely unrelated to the functionality of the apps. However, the majority of them accessed location. This is problematic from a privacy point of view since they do not need the location information to function while they are running in the background. `PHONE_STATE` is an interesting data resource since the respective information is highly sensitive. This permission enables an invasive party to gain access to sensitive resources such as phone number, cellular network information, outgoing call information, etc. The only relevant reason to access this permission is to stop the app when there is an ongoing call, however, we did not use SIM card on the devices, therefore, there is no reason of such resource access. This also happened to other sensitive resources such as `MICROPHONE`, `READ_SMS`, etc. We also observed that many of these resource

Table 3: Performance Measures of the Classification Algorithm

Classes	Recall	Precision	F-score
Tracking & Spyware	0.7549	0.8311	0.8214
Phishing	0.8588	0.8653	0.8601
Unauthorized Charges	0.7912	0.9583	0.8296
Unintended Data Disclosure	0.9010	0.9765	0.9218
Targeted Ads	0.9374	0.9971	0.9663
Spam	0.9374	0.9514	0.9388
General	0.7576	0.8639	0.8492
Overall	0.8895	0.9116	0.9059

accesses happened when: (1) the devices were in horizontal orientation, (2) the devices' screen was off and (3) the proximity sensor indicated that there is no nearby object.

4.2 App Perception Analysis Results

To validate the capability of our novel perception analyzer, we collected a dataset consisting of 75,601 user reviews corresponding to these 20 health-based apps using the scraper in [1]. Three experts went manually through the data and labeled them. We then used $CountVectorizer$ and $TfidfTransformer$ packages in scikit-learn [26] for the feature extraction phase. We then split the data set into training and testing data (70% for training and 30% for testing). Using scikit-learn we exploited several classification algorithms such as *Support Vector Machines (SVMs), Random Forest, Logistic Regression (LR),* etc. We observed *LR* outperforms others, therefore, we only show the results for *LR*. We used recall, precision and F-score metrics to evaluate the performance of the classifier. The values of these metrics show how good the classifier's results correspond to the annotated results. Table 3 shows the values for the aforementioned metrics corresponding to each identified threat. The observation is that the overall recall and precision values are of 88.95% and of 91.16%, respectively. Moreover, the values obtained for F-score show the good performance of our approach.

Tab. 4 shows some examples regarding the strength of perception analyzer in distinguishing different types of user reviews with different sentiments and relevant threat (shown by T). The obtained results clearly confirm the applicability and the positive influence of perception analyzer in the overall risk assessment done by *ESARA*.

4.3 Risk Coverage

As *ESARA* is a privacy and security risk assessment tool for mobile apps in enterprises, it is of particular importance to check its coverage of the most prevalent mobile app risks. We took Veracode [8] as one of the well-established references that categorizes the top 10 mobile app risks (considering the top 10 risks introduced by OWASP [3]) and we investigate the robustness of *ESARA* in assessment and detection of each individual risk. In Table 5 we clarify which

Table 4: An Example of Classified User Reviews

#	Sample user review	T
1	*You don't need to spy on my activities outside of this app. they don't care about their customers, they want to ruin the device with horrible bloatware spyware*	T1
2	*Im still getting warnings that my phone is infected with virus after i update and scan again. If its not going to work why download it. I have very limited memory to use. No need to download stupid apps that dont work*	T2
3	*Cheating Y the hell.. u cut my 50 rupees for nothing.. i just enter my card details and u cut my money without asking me.. i want it back*	T3
4	*SHit!Takes control of device.. why my photo is there??!!*	T4
5	*Ads are terrible Sorry but the ads are comparing to the website really irritating.*	T5
6	*had this problem about these Annoying full screen PoP-ups! instaled this... boom they're gone, Legit !*	T6
7	*Dangerous! requires unnecessary access to sensitive permissions! Uninstalled*	T7

Table 5: Coverage of Veracode Top 10 Mobile App [8] Risks by *ESARA*.

No.	Risk	Malware Checker	Vuln. Checker	Behavior Analyzer	Perception Analyzer
1	Activity monitoring and data retrieval	✓	–	(✓)	(✓)
2	Unauthorized dialing, SMS, and payments	✓	–	✓	(✓)
3	Unauthorized network connectivity	(✓)	–	–	(✓)
4	UI Impersonation	–	–	–	(✓)
5	System modification	✓	–	–	✓
6	Logic or Time bomb	✓	(✓)	–	–
7	Sensitive data leakage	(✓)	✓	✓	✓
8	Unsafe sensitive data storage	–	✓	–	(✓)
9	Unsafe sensitive data transmission	–	✓	–	✓
10	Hardcoded password/keys	✓	✓	–	–

component of *ESARA* may detect which identified risk. Thanks to the novel combination of *BA*, *PA*, malware and vulnerability checkers, the *ESARA*'s components totally (shown by ✓) or partially (shown by (✓)) cover all the risks. We observed that each risk is at least covered by two components (except UI impersonation which is one the most complex risk scenarios in terms of identification and mitigation).

4.4 Discussion and Limitations

We could address all the requirements we defined in Sect. 3.1 and cover the top 10 mobile app risks with at least one component. The results from the evaluation of the app behavior analyzer and the app perception analyzer are very promising. However, there is a limitation of our work. We have not tested our framework in a real company environment, yet. We only did user studies in a laboratory environment. Therefore, it remains to respectively show that employees would like the idea of getting support for the decisions about which apps they want to install and therefore actively make use of the potentials provided by *ESARA*.

5 Conclusion and Future Work

Smartphones have become ubiquitous within enterprise environments. At the same time, with the increased interest and not only the adoption of *BYOD*,

employees heavily rely on apps, sometimes also used for personal purposes, that have access to enterprise confidential data as well. As a result, security and privacy have become a big challenge in enterprises. In this paper, we proposed *ESARA* as a novel framework to analyze and quantify the potential privacy and security risks associated with employees' smartphone apps within an enterprise environment. After an in-depth analysis of the most relevant works in the literature, we proposed an approach that leverages a four-pillar mechanism, including malware checker, vulnerability checker, behavior analyzer and perception analyzer. The combination of these mechanisms that are jointly working together supports and enables enterprises to profoundly examine the privacy and security aspects of their employees' installed apps. Since malware and vulnerability checkers are well researched, we only evaluated the performance of the two newer components, the behavior analyzer (BA) and the perception analyzer (PA). We practically showed the applicability of using behavior and perception analyses to have a more fine-grained app risk assessment and our results confirmed that these two factors play a critical role in the overall quantification of app security and privacy risks. *ESARA* opens opportunities for further innovative solutions for risk assessment of mobile apps within enterprise environments, including easing the quantification of apps trustworthiness degree.

In our future work, we will further enhance the performance of the perception analysis component by providing more training and testing data. Additionally, user studies are planned to determine the employees' and IT departments' acceptance of our approach. Also, a comprehensive analysis in an enterprise environment to validate the whole framework in a real world scenario is planned in the future.

References

1. Google play scraper, https://github.com/facundoolano/google-play-scraper/
2. Mobile application security scanner, https://www.ostorlab.co/
3. Mobile top 10 2016-top 10, https://www.owasp.org/index.php/mobile_top_10_2016-top_10/
4. Nviso. apkscan., https://apkscan.nviso.be/
5. Quick android review kit, https://github.com/linkedin/qark
6. Quixxi integrated app management system, https://quixxisecurity.com/
7. Sanddroid – an automatic android application analysis system, http://sanddroid.xjtu.edu.cn
8. Veracode mobile app top 10, http://www.veracode.com/directory/mobileapp-top-10/
9. Protection of sensitive data and services, https://www.sit.fraunhofer.de/en/bizztrust/ (2012)
10. 2016 nowsecure mobile security report, `https://www.nowsecure.com/blog/2016/02/11/2016-nowsecure-mobile-security-report-now-available/` (2016)
11. Arxan's 5th annual state of application security report, https://www.arxan.com/press-releases/arxans-5th-annual-state-of-application-security-report-reveals-disparity-between-mobile-app-security-perception-and-reality (2016)

12. Eu general data protection regulation, https://eur-lex.europa.eu/legal-content/en/txt/html/?uri=celex:32016r0679 (2016)
13. Eu-u.s. privacy shield, https://iapp.org/resources/article/eu-u-s-privacy-shield-full-text/ (2016)
14. Framework for app security tests, https://www.sit.fraunhofer.de/en/appicaptor/ (2016)
15. 10 most vulnerable os of the year 2017, https://www.cybrnow.com/10-most-vulnerable-os-of-2017/ (2017)
16. Agarwal, Y., Hall, M.: Protectmyprivacy: detecting and mitigating privacy leaks on ios devices using crowdsourcing. In: Proc. of MobiSys. pp. 97–110 (2013)
17. Beresford, A., Rice, A., Sohan, N.: Mockdroid: trading privacy for application functionality on smartphones. In: the Proc. of the 12th Workshop on Mobile Computing Systems and Applications, Phoenix, Arizona, USA. pp. 49–54 (2011)
18. Chandramohan, M., Tan, H.B.K.: Detection of mobile malware in the wild. Computer **45**(9), 65–71 (Sept 2012). https://doi.org/10.1109/MC.2012.36
19. Enck, W., Gilbert, P., Chun, B., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: the Proc. of the the 9th ACM USENIX Conf. on Operating Systems Design and Implementation, Vancouver, BC, Canada. pp. 393–407 (2010)
20. Hatamian, M., Serna, J., Rannenberg, K., Igler, B.: Fair: Fuzzy alarming index rule for privacy analysis in smartphone apps. In: the Proc. of the 14th Int. Conf. on Trust and Privacy in Digital Business (TrustBus), Lyon, France. pp. 3–18 (2017)
21. Hogben, G., Dekker, M.: Smartphones: Information security risks, opportunities and recommendations for users. Tech. rep., ENISA report (2010)
22. Maggi, F., Valdi, A., Zanero, S.: Andrototal: A flexible, scalable toolbox and service for testing mobile malware detectors. In: Proc. of the 3rd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 49–54 (2013)
23. Marinos, L.: Enisa threat taxonomy: A tool for structuring threat information. Tech. rep., ENISA report (2016)
24. Martínez-Pérez, B., De La Torre-Díez, I., López-Coronado, M.: Privacy and security in mobile health apps: A review and recommendations. J. Med. Syst. **39**(1), 1–8 (Jan 2015)
25. Nauman, M., Khan, S., Zhang, X.: Apex: Extending android permission model and enforcement with user-defined runtime constraints. In: Proc. of the 5th ACM Symp. on Information, Computer and Communications Security. pp. 328–332 (2010)
26. Pedregosa, F., Varoquaux, G., Gramfort, A., V. Michel, B.T., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, J., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
27. Plachkinova, M., Andres, S., Chatterjee, S.: A taxonomy of mhealth apps – security and privacy concerns. In: 2015 48th HICSS. pp. 3187–3196 (Jan 2015)
28. Zhou, Y., Zhang, X., Jiang, X., Freech, V.W.: Taming information-stealing smartphone applications (on android). In: the Proc. of the 4th International Conf. on Trust and Trustworthy Computing, Pittsburgh, PA, USA. pp. 39–107 (2011)