Technische Universität Darmstadt

Center for Advanced Security Research Darmstadt

Technical Report TUD-CS-2013-0218

ConXsense – Context Sensing for Adaptive Usable Access Control

Markus Miettinen, Stephan Heuser, Wiebke Kronz
Ahmad-Reza Sadeghi, N. Asokan

System Security Lab
Technische Universität Darmstadt, Germany

CASED
Technische Universität Darmstadt
D-64293 Darmstadt, Germany

TUD-CS-2013-0218
First Revision: August 13, 2013
Last Update: August 13, 2013

ConXsense – Context Sensing for Adaptive Usable Access Control

Markus Miettinen*, Stephan Heuser*, Wiebke Kronz†, Ahmad-Reza Sadeghi‡, N. Asokan§
*Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany,
†Intel Collaborative Research Institute for Secure Computing at TU Darmstadt,
‡TU Darmstadt / Center for Advanced Security Research Darmstadt (CASED),
§University of Helsinki, Department of Computer Science

***Abstract*—In this paper, we present the design and implementation of ConXsense, a framework utilizing context sensing for easy-to-use and adaptive context-aware access control for mobile devices. Previous work often require either users to laboriously specify detailed policies or they rely on pre-specified, non-personalized and error-prone policies for generic context classes. Recent approaches attempt to address these deficiencies by learning from context data. Our approach improves on this by using context data to automatically estimate the sensitivity and safety of the user's context and using the estimates for dynamically enforcing access control rules in a highly personalized, non-intrusive and usable manner. Our initial implementation of the framework addresses two smartphone-related problem scenarios for context-aware access control: 1) how to prevent unauthorized apps (like sensory malware) from gathering information about the context of a mobile device (contextual privacy) and 2) how to protect the data and applications on the device from physical threats in the context (like thieves or device misuse by others). We start with a sociological user study, and use its results to inform the design and implementation of ConXsense. We carry out a data collection and analysis study based on which we evaluate the effectiveness and accuracy of ConXsense. Moreover, we integrate ConXsense with a fine-grained access control architecture and show how it can effectively protect against sensory malware as well as device theft and misuse.**

I. Introduction

Mobile devices today are equipped with a wide variety of sensors. Applications that make use of the context are becoming increasingly popular. Examples include location-based applications like *Foursquare* and *Google Latitude*, augmented reality applications like *Layar*, *Wikitude*, *Google Goggles HERE City Lens* and many more. Even mainstream applications like social network apps support context-based enhancements.

However, context information and sensing capabilities of smart devices also provide an attractive attack surface for misuse, as the recent development of sensory malware shows: malicious code, typically a Trojan appearing to be a legitimate app, uses the sensors to extract sensitive information from the surroundings. Prominent examples are *Stealthy Video Capturer* [1] (video via camera), *(sp)iPhone* [2] (keystrokes via accelerometer) *Soundcomber* [3] (spoken secrets via microphone), or the recent *PlaceRaider* [4] Trojan presented at NDSS 2013 (3D models via camera). Users may also have granted sensor access privileges to benign apps which use them too intrusively: for instance the user grants camera permissions to an augmented reality app without realizing that the app may take pictures of surroundings even when the user does not want to use augmented reality, as a means of enriching the app vendor's data collection. At the same time, the task of configuring sensible security and privacy policies is becoming increasingly difficult for ordinary smartphone users due to the ever-growing amount of applications and services used on smartphones. A quick remedy for this could be the increased use of default policies, but these can not take the personal security and privacy needs of individual smartphone users adequately into account. Several surveys [5], [6] point out that many mobile users do not use idle screen locks to protect their phones. One reason is that screen locks and other similar access control techniques available on mobile devices today are both too inflexible and hard to use. Exploiting context information to configure access control may address both problems [7].

Hence, there is a need to 1) control the use and release of contextual information in order to encounter threats arising from, e.g., *sensory malware*, and, 2) to utilize available contextual information to enable *context-aware access control*.

While context-aware access control is not new, most previous works require pre-specified policies [8], [9], [10]. Context features and context detection are mainly used as triggers for activating or deactivating access control rules as specified by these policies. Only recently have researchers started to explore how to improve the usability of context-aware access control by using context information more intensively [7], [11]. Our work constitutes an advancement in this line of research, but going even further. By utilizing a sophisticated context model and data analysis methods grounded in real-world ground truth data, we aim at a system that is not only capable of triggering access control rules based on context, but that actually provides the capability to assess the security and privacy of the context itself dynamically. We envisage that such functionality would be applicable also to many other security applications on smartphones, besides access control.

Contributions and Roadmap. Our contributions are as follows:

- **A sociological survey of user perceptions** as to how the location and surroundings of a smartphone affect

the security and privacy of users (Section III). We use survey techniques widely accepted in the social sciences community.

- The design and implementation of *ConXsense*, the *first* **context-aware access control framework** for smartphones that provides automatic, adaptive and *personalized* policy decisions for context dependent access control (Sections IV and VI). Our *interdisciplinary* approach uses the results of the *sociological* study to design and implement a sophisticated *context model* as well as *data analysis and machine learning* techniques for estimating the sensitivity and safety of contexts dynamically (Section V). It also integrates them with *operating system security* components and uses the sensitivity and safety assessments to select appropriate access control rules thus constructing an end-to-end context-aware access control system (Section IV).
- The application of ConXsense to two concrete use cases (Section II): **protection of smartphone data** by using ConXsense to dynamically configure idle screen locking, and **defending against sensory malware**. ConXsense, however, is applicable to a wide range of other security and privacy-related use cases.

After summarizing related work (Section VII), we conclude (Section VIII) with some outlines of future work.

II. Problem Description

Problem Setting. We focus on context-based access control in two different flavors: First, we aim at protecting the information in the ambient "context" of the device (i.e., the environment the device finds itself in) from malicious or untrusted applications on the device. In particular, we want to prevent or limit the ability of untrusted applications to gather information from contexts that are *sensitive*. We can further subdivide sensitive contexts into *private* contexts, where the sensitive information in the context is private to the user of the device, and *confidential* contexts where the sensitive information belongs to someone else but the user still wants to protect this information from untrusted apps. The user's home is an example of a private sensitive context, whereas the user's workplace is an example of a confidential sensitive context. Second, we want to protect the applications and data on the device from potential threats in the context: we want to ensure *safety* for the applications and user data by limiting the potential damage arising from someone physically accessing the device without the user's approval. In other words, we want to assure privacy (case 1) as well as safety (case 2).

Adversary Model. For the privacy case, the adversary is an app already installed on the device. We assume that the application has already been granted the necessary privileges during installation and has therefore access to the contextual sensors on the mobile device. The application may be a Trojan Horse (e.g., sensory malware) or a benign but somewhat intrusive application. For the safety case, the adversary is a person in the context with physical access to the device. The person may be malicious (a thief) or honest-but-curious (a colleague or sibling) or "clueless" (a small child).

Assumptions. We assume that the user's device is equipped with a variety of sensors capable of recording information about its ambient context. This information typically includes things like GPS location, information about WiFi access points and Bluetooth devices in range, accelerometer readings, camera snapshots, audio samples, magnetometer readings, luminosity and proximity sensor readings, etc. We also assume that the device has some form of a fine-grained access control system available (such as ours), and that malware on the device cannot circumvent the access control system. [1]

Objective and validation. Our objective is to design a context profiling framework that can use available contextual information to automatically and dynamically configure the fine-grained access control system in order to provide protection against the type of adversaries we discussed above.

With the help of our framework, the device would be able to autonomously identify relevant contexts and suitable policies for these contexts based on limited user feedback, without the need for the user to explicitly define these contexts and assign policies to them beforehand.

In order to be successful, our framework must be able to identify relevant contexts of the user, determine the sensitivity and safety of the context from the user's point of view and select appropriate access control rules for the fine-grained access control system.

In the case of privacy, we want to thwart sensory malware applications on the affected device from collecting sensitive information by using the mobile device sensors while the user is in a sensitive context. On the other hand, access to context sensors for benign third-party apps should be restricted as little as possible when the user is not in a sensitive context. We do not want to limit the functionality and thus the user-experience of legitimate apps that need relevant context information to function properly. Similarly, in the case of safety, we want to minimize the chances that an unauthorized person in the context has access to a user's device. We will do this by configuring the idle screen lock dynamically based on the safety level of the context, while trying to strike a balance between maximal safety and the user annoyance of having to unlock the device in safe contexts.

In both cases, we will evaluate the effectiveness of the framework by comparing how it performs against ground truth data that we collect from a user data collection trial.

III. User Survey

User perceptions of safety and sensitivity of contexts are important aspects for the design of our framework. To study the factors that influence the user's views of safety and sensitivity, we conducted a survey that was answered by 122 participants aged 18-56, including people from different household types

[1] Possible malware that would be able to use operating system (root) exploits to circumvent the enforcement of the context-aware access control system is outside the scope of this paper.

and representing different organizational positions, technical and non-technical, researchers, secretaries and managers.

Following a *Mixed Methods* [12] approach, we designed a statistical-quantitative survey [13] using quantitative questions to identify facts by statistical analysis [14] combined with open-ended, qualitative questions for investigating the underlying reasons for the perception of safety and sensitivity. The text of the survey questionnaire is available for reference in appendix A.

Our understanding of "safety" is adopted from social sciences: A private place like home is perceived as safe due to the predictability of and control over the environment and the familiarity of people therein. Public places on the other hand, lack these aspects and are typically not perceived as safe [15]. We explicitly exclude the notion of safety in criminal and other social contexts, and focus specifically on the notion of safety with regard to the usage of mobile devices. Therefore, we follow the assumption in [7] that familiar contexts with familiar people are "safe", and there is no need to protect the smartphone in such contexts (e.g., by activating the idle screen lock). While it does not seem surprising that a highly familiar place like "home" would be perceived as safe, some studies report perceptions of another very familiar context, namely "work" to be more diverse [16]. This motivated us to specifically investigate the differences in perceptions related to these contexts, "home" and "work".

Another dimension in addition to the perception of a context being "safe" or "unsafe" is the distinction between "sensitive" and "public" contexts. We consider a context "sensitive" if its context information is private to the user or contains confidential information so that applications on the user's smartphone should not have access to it without the user's explicit approval. For example, a person's workplace, the doctor's office, and a person's home are likely to be perceived as sensitive environments since these contexts typically contain private and confidential information. In contrast, a "public" context is of such generic or public nature, that it does not reveal significant private or confidential information.

With our survey, we sought to identify, which are the primary factors affecting the perception of contexts as safe or sensitive w.r.t. to the usage of smartphones. As suggested in earlier work [7], we aimed to particularly investigate the role of familiar contexts and familiar people as factors.

Survey Results. As shown in table I, the majority of people (94% and 55%, respectively) perceive familiar contexts (Home, Work) as safe. It seems clear that the familiarity of a place affects the perceived safety of the context positively. However, a significant fraction of people (40%) perceive "work" as "unsafe", suggesting that the familiarity of the context alone is not a sufficient indicator for safety, as we shall see below.

The survey data in table I also suggest that a significant fraction of respondents (46% and 42%, respectively) feel that a familiar context (Home, Work) is also sensitive. This dependency of the sensitivity of a context on the context's familiarity is, however, clearly not as strong as it is for safety. Nevertheless we consider the familiarity of a context to be a factor that affects the sensitivity of a context positively.

In table II we can see that the majority of respondents (66%) found safety to depend on people present in the context. From the responses to the open-ended questions, it is clear that the feeling of safety is particularly caused by people that are *familiar* to the user. Similarly, 68% of respondents say the presence of people can also cause a context to be perceived as unsafe. Answers to open-ended questions showed that the presence of unfamiliar people implies unpredictability and represents risk, leading to a feeling of unsafety. We can therefore conclude, that the feeling of unsafe is caused by the presence of people that are *unfamiliar* to the user.

Therefore, it seems that the following factors affect the perception of safety of a context: On the one hand, familiar people in the context cause a context to be considered safe. However, the presence of unfamiliar people in the context is a reason for that context to be considered unsafe. Keeping this in mind, we can understand, why according to table I 40% of respondents specified "work" as an *unsafe* place. While the place "work" itself is safe as long as there are only familiar people around, the appearance of unfamiliar people cause the perception of a formerly safe context to change to unsafe.

From table II we see that the perception of sensitivity does not appear to be affected by the presence of people. More respondents (43% and 42% vs. 39% and 36%, respectively) believe that the people present in the context do not influence whether the context is *sensitive* or *public*. It would therefore seem that it is the private or confidential information often present in specific contexts that plays a more significant role in the perception of sensivity of contexts. And in fact, confidential information is one of the main reasons given by respondents in open-ended question answers for why a context is considered sensitive.

TABLE I: Perceptions of Home vs. Work

Context	Sensitive	Public	Safe	Unsafe
Home	46%	17%	94%	4%
Work	42%	21%	55%	40%

TABLE II: Influence of people on the perceived safety and sensitivity of the context

Question	Yes	No
Safe depends on people	66%	14%
Unsafe depends on people	68%	11%
Sensitive depends on people	39%	43%
Public depends on people	36%	42%

Based on the analysis of the survey results above, we can identify following main factors affecting users' perceptions of sensitivity and safety:

- Familiar contexts tend to be regarded as safe.
- Familiar people in context tend to make contexts to be regarded as safe.
- Familiar contexts tend to be regarded as sensitive.
- Unfamiliar people in context tend to make contexts to be regarded as unsafe.

Since the familiarity of contexts and people seem to be central factors affecting the perceptions of mobile device users, we designed our context model in a way that we can build context profiles that can 1) identify relevant contexts and model their familiarity, and, 2) track encounters with other persons and identify familiar people by observing their mobile devices. As described more closely in sections IV and V, we designed heuristic evaluation functions that attempt to capture the logic behind these factors. Keeping these factors in mind, we also selected context features as input data for inductive evaluation functions that would allow machine learning models to learn about user's perceptions of sensitivity and safety and make predictions about them based purely on the context profiles and incoming context information.

Discussion. The analysis presented above supports our *common* understanding of the perception of safety and sensitivity and even shows which factors influence the perception of contexts. However, exceptions exist, like, e.g., those 4%, who consider home (familiar context with familiar people) as "unsafe". Reasons for this can be various. Answers to open-ended questions suggest that even in familiar contexts the perception of safety changes when untrusted people appear or it is caused by what we call the "toddler scenario": the influence of familiar people (e.g., a young child, or spouse) considered as clueless or honest but curious, causing a person to prefer her device to remain protected also in familiar contexts. To investigate these special cases, more detailed questions on contexts that are perceived as sensitive would need to be included. However, sociological literature suggests that statements about such contexts are perceived as intrusive and will therefore not be answered [15], [16]. This concern was also reflected in some responses we received about the online questionnaire. Therefore, and because exceptional cases seem to be a marginal phenomenon, we decided to focus our investigation at this point on the common cases and address more exceptional cases in subsequent studies.

IV. System Design

Figure 1 shows the high-level design of the ConXsense system. The Context Sensing Layer is responsible for acquiring relevant context data from the device's sensors at regular intervals (e.g., once a minute). These data are processed by the ContextProfiler to identify relevant context objects and accumulate context profiles for them. The context profiles are used by the Context Evaluator to assess the sensitivity/safety level of the context. The Context Evaluator regularly evaluates two functions: the *context sensitivity evaluation function* $\hat{\lambda}$ providing an estimate of the sensitivity of the context and the *context safety evaluation function* $\hat{\phi}$ providing an estimate of the safety of the context. We will show in section V, how these evaluation functions are computed utilizing contextual data from the environment.

The Context Evaluator furthermore continuously informs the Access Control Layer of changes in the device's context's sensitivity/safety level as shown in Figure 2. Context-dependent access control rules for sensitive functions are

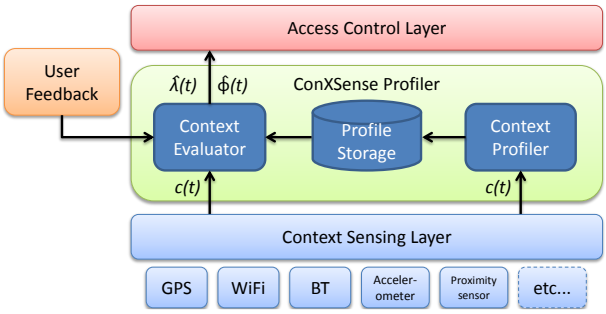


Fig. 1: ConXsense System Design

stored in the Access Control Policy. These rules describe in which contexts, more precisely in which sensitivity/safety level, apps of different trust levels may access sensitive functions. Policy Enforcement Points (PEPs) in relevant operating system functions query the PolicyServer for access control decisions at runtime. The PolicyServer thereby acts as a Policy Decision Point (PDP) in our access control architecture. Furthermore, the PolicyServer proactively forwards changes in the current context's sensitivity/safety level to selected components, such as the Lockscreen, to enforce changes in their runtime behavior.

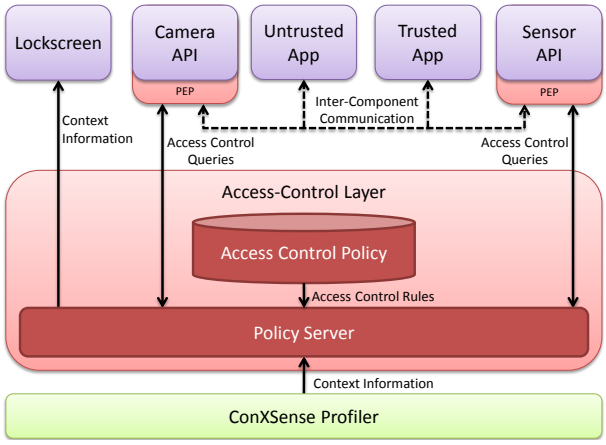


Fig. 2: Enforcement of Context-based Policies

In Section VI we apply the ConXsense architecture in two different use-cases: We first show how the ConXsense architecture can be used to protect the user from sensory malware. We then present a context-aware device Lockscreen which mitigates the effects of device loss and theft.

For the correctness and usability of the framework, the performance of the Context Evaluator is of vital importance, since the correctness of the estimates directly impacts the degree of protection the framework provides on one side, and, the impact on the usability on the other hand. We model therefore the goodness of the system using following figures of merit to estimate the performance of the framework.

Given a context $c(t)$ at timepoint t the Context Evaluator evaluates regularly two functions, the *context sensitivity evaluation function* $\hat{\lambda} : \{c(t) \,|\, t \in \mathcal{T}\} \rightarrow \{sensitive, public\}$

and the *context safety evaluation function* $\hat{\phi} : \{c(t) \mid t \in \mathcal{T}\} \rightarrow \{safe, unsafe\}$. In addition, we denote the 'true' sensitivity and safety levels of a context $c(t)$ at time point t with $\lambda(c(t))$ and $\phi(c(t))$, respectively.

Protection level. The *protection level* that our framework provides is defined by the fraction of times the system correctly identifies contexts as *sensitive* or *unsafe* and thus causes the system to enforce appropriate access control rules to protect against sensory malware or device theft/misuse.

The protection level $\pi_{\hat{\lambda}} \in [0, 1]$ of the context sensitivity evaluation function $\hat{\lambda}$ is defined as the fraction of times during which Context Evaluator correctly identifies sensitive contexts as *sensitive*:

$$\pi_{\hat{\lambda}} = \frac{\sum_{t \in \mathcal{T}} 1_{\{sensitive\}}(\hat{\lambda}(c(t))) \times 1_{\{sensitive\}}(\lambda(c(t)))}{\sum_{t \in \mathcal{T}} 1_{\{sensitive\}}(\lambda(c(t)))}$$

where $1_A(x)$ is an indicator function, s.t. $1_A(x) = 1$, if $x \in A$, and 0 otherwise.

Similarly, for the context safety evaluation function $\hat{\phi}$, we define the protection level $\pi_{\hat{\phi}} \in [0, 1]$ as the fraction of times the Context Evaluator correctly identifies unsafe contexts as *unsafe*:

$$\pi_{\hat{\phi}} = \frac{\sum_{t \in \mathcal{T}} 1_{\{unsafe\}}(\hat{\phi}(c(t))) \times 1_{\{unsafe\}}(\phi(c(t)))}{\sum_{t \in \mathcal{T}} 1_{\{unsafe\}}(\phi(c(t)))}$$

The higher the protection level that the evaluation functions provide, the better the framework is able to protect the user against aforementioned threats.

Usability deterioration. Each time the ConXsense framework causes the Access Control Layer to enforce access control limitations because the context is estimated to be *sensitive* (by limiting access to sensors) or *unsafe* (by activating the Lockscreen), the usability of the device inevitably deteriorates. In truly sensitive/unsafe contexts this is inevitable, but we want to limit the *unnecessary* enforcement of limiting access control rules as much as possible in order to provide better user experience for the user in such contexts in which strict protection measures are not needed. To be able to measure the unnecessary deterioration of user experience, we define the following figures of merit. For the context sensitivity evaluation function $\hat{\lambda}$, we define the usability deterioration $\xi_{\hat{\lambda}} \in [0, 1]$ to be the fraction of times the Context Evaluator incorrectly identifies a public context as *sensitive*:

$$\xi_{\hat{\lambda}} = \frac{\sum_{t \in \mathcal{T}} 1_{\{sensitive\}}(\hat{\lambda}(c(t))) \times 1_{\{public\}}(\lambda(c(t)))}{\sum_{t \in \mathcal{T}} 1_{\{public\}}(\lambda(c(t)))}$$

Similarly, for the context safety evaluation function $\hat{\phi}$, we define the usability deterioration $\xi_{\hat{\phi}} \in [0, 1]$ as the fraction of time the Context Evaluator incorrectly identifies a safe context as *unsafe*:

$$\xi_{\hat{\phi}} = \frac{\sum_{t \in \mathcal{T}} 1_{\{unsafe\}}(\hat{\phi}(c(t))) \times 1_{\{safe\}}(\phi(c(t)))}{\sum_{t \in \mathcal{T}} 1_{\{safe\}}(\phi(c(t)))}$$

The lower the usability deterioration scores $\xi_{\hat{\lambda}}$ and $\xi_{\hat{\phi}}$ are, the better usability for users our framework is capable to provide.

In section VI, we will use the above figures of merit to evaluate the implementation of our system based on real data collected from a user study. In the following, we present the context model based on which the input data for the sensitivity and safety evaluation functions $\hat{\lambda}$ and $\hat{\phi}$ are generated.

V. Context Model

In this section, we present our context model that we apply in our current implementation of the ConXsense framework. Based on the learnings of the user survey (section III), we base our context model on two important aspects: i) *Location Context* and *Contexts of Interest (CoI)* for modeling contexts and their familiarity and ii) the *Social Context* for modeling familiar and unfamiliar people in context.

Contexts-of-Interest (CoI). For our purpose, *Contexts-of-Interest* (CoIs) correspond to locations that a user visits often and/or spends a significant amount of time in, e.g., home, workplace, grocery store, etc. Some places might be best identified by profiling the GPS observations and identifying geographical areas that accumulate particularly many GPS observations. For other CoIs, however, especially for CoIs located indoors, using GPS alone might not be feasible, since the structure of buildings often obstructs reception of required GPS signals. Therefore, such CoIs are better identified by using the set of WiFi access points observed at these locations as 'signatures' for the CoI. Hence, we adopt both GPS and WiFi as means for identifying CoI and use them in parallel to capture important places of the user.

Social context. In order to capture different social contexts of users, we model *people* in the user's surroundings through their mobile devices that can be sensed through proximity sensing technologies like Bluetooth (BT).

To capture only devices that are typically carried by persons, we filter the BT observations by their device class so that we consider only mobile devices like cell phones, headsets, PDAs and other portable devices.

In the following we will define the metrics and algorithms for measuring the above parameters and how to validate them to estimate the context the user is in. For the reader's convenience, the relevant parameters used in the following definitions can be found in concise format also in the Appendix, Table IV.

A. Detection of Contexts of Interest (CoIs)

We consider two kinds of CoIs: GPS-based CoIs which are geographical areas on the surface of the earth, and WiFi-based CoIs that are characteristic sets of WiFi access points usually observed in a specific place and thus identifying the RF environment there. GPS CoIs capture significant places of the user in outdoor areas, and WiFi CoIs cover also indoor locations in urban areas, where typically coverage of WiFi access points is available. By combining both types of CoIs, we are able to identify and detect most significant places that users typically visit.

1) GPS-based CoIs: The identification of GPS-based CoIs is based on position observations $pos_i = (lat_i, lon_i)$ obtained via GPS.

We denote with $t(pos_i) \in \mathbb{N}$ the timestamp associated with the observation, and with $dist(pos_i, pos_j)$ the geographical distance of position observations pos_i and pos_j.

The sequence of GPS observations is divided into *GPS stay points*, which represent visits of the user to different places, during which the user stays within a radius of r_{sp} from the first GPS observation. In order for a visit to be considered a stay point, the visit is also required to last longer than t_min_{sp} and not to contain observation gaps longer than t_gap_{sp}.

Definition 1 (GPS stay point): A *GPS stay point* $gps\_sp = (pos_1, pos_2, \ldots, pos_n)$ is a sequence of position observations pos_i, such that $\forall i, 1 < i \leq n : \mathsf{dist}(pos_1, pos_i) \leq r_{sp}$ and $t(pos_i) - t(pos_{i-1}) \leq t\_gap_{sp}$ and $\mathsf{dur}(sp) = t(pos_n) - t(pos_1) \geq t\_min_{sp}$.

We calculate for each stay point an average position $pos_{\bar{sp}}$ as the average of all position observations belonging to the stay point, i.e., $pos_{\bar{sp}} = (lat_{\bar{sp}}, lon_{\bar{sp}})$, s.t. $lat_{\bar{sp}} = \frac{\sum_{k=1}^{n} lat_k}{n}$, and $lon_{\bar{sp}} = \frac{\sum_{k=1}^{n} lon_k}{n}$. The average position of a stay point represents the predominant location where the user has been located during her visit to the stay point.

The average positions $pos_{\bar{sp}}$ of individual stay points are aggregated to form rectangular geographical areas of at most gps_{max} width and length. An area is a *GPS-based Context-of-Interest*, if (i) the user has visited the area more than f_min_{coi} times and (ii) has spent in the area longer than t_min_{coi} in total. More formally,

Definition 2 (GPS-based CoI): Let us denote a rectangular area with $C_i = (lat_{min}, lon_{min}, lat_{max}, lon_{max})$ and let $\mathsf{gps\_sp}(C_i)$ denote the set of stay points sp whose average position $pos_{\bar{sp}}$ is contained in C_i. The area C_i is a *GPS-based CoI*, if following conditions hold:
(1) $|\mathsf{gps\_sp}(C_i)| \geq f\_min_{coi}$,
(2) $\sum_{sp_j \in \mathsf{gps\_sp}(C_i)} \mathsf{dur}(sp_j) \geq t\_min_{coi}$,
(3) $lat_{max} - lat_{min} \leq gps_{max} \wedge lon_{max} - lon_{min} \leq gps_{max}$.
An observation pos falls in CoI C_i, if $lat_{min} \leq lat_{pos} \wedge lon_{min} \leq lon_{pos} \wedge lat_{pos} \leq lat_{max} \wedge lon_{pos} \leq lon_{max}$.

Example. As an illustrative running example, let us consider a user who is a full-time working professional, regularly commuting between her place of work and home, predominantly using public transport. Other places she regularly visits are a grocery store close to her home, a public sports facility and the home of a close relative. She usually carries her smartphone with her, which continuously senses her context and context data of her GPS location and WiFi access points.

Assume now, our example user goes to the grocery store and stays there for 32 minutes, i.e., longer than $t\_min_{sp} = 10$ minutes and moves only within a radius of $r_{sp} = 100$ meters, a stay point sp (Def. 1) of duration $\mathsf{dur}(sp) = 32$ minutes will be generated. The average of all position observations pos_i during the stay point visit will be the stay point average position $pos_{\bar{sp}}$, most likely located in or near the grocery store. Waypoints along her daily commuting routes, however, would not generate any stay points, since on her way she does not spend sufficiently long time in the same limited area.

Now, if our user visits the grocery store 10 times and stays each time for 32 minutes, ten stay points will be generated with average positions $pos_{\bar{sp}}$ located in or near the grocery store. These average positions will be aggregated into a GPS-based CoI C (Def. 2), because their total stay duration of 5 hours and 20 minutes is longer than the required $t\_min_{coi} = 30$ minutes and there are more than the required $f\_min_{coi} = 5$ stay points falling inside the CoI. The area of the CoI will be the smallest rectangle containing all the stay point average positions $pos_{\bar{sp}}$ associated with the stay point visits. The same holds for GPS-based CoIs covering, e.g., her home or the sports facility.

Note that our notion of stay points and CoIs is not bound to a stationary life. Also frequent travelers would obtain GPS-based CoIs over time, given that they visit the same places (e.g., airports or hotels) often enough and spend sufficiently long time there.

2) WiFi-based CoIs: For identifying WiFi-based CoIs, WiFi access point observations rf_i are used. Each observation consists of the MAC address of a detected WiFi access point and the timestamp of the observation, which we denote with $t(rf_i)$. The sequence of individual WiFi observations is divided into *WiFi snapshots*, which are subsequences corresponding to observations obtained during a single WiFi scan of duration t_max_{wifi}.

Definition 3 (WiFi Snapshot): A sequence of WiFi access point observations rf_i that occur within t_max_{wifi} constitute a *WiFi snapshot*, denoted as $wifi$. That is, $wifi = (rf_1, rf_2, \ldots, rf_n)$, such that $t(rf_n) - t(rf_1) < t\_max_{wifi}$. We denote with $t(wifi)$ the first timestamp belonging to the snapshot, i.e., $t(wifi) = t(rf_1)$.

Following the notion of stay points for GPS observations, we extend this concept to WiFi and divide the sequence of WiFi snapshots into so-called *WiFi stay points*. The similarity between snapshots is determined by calculating the *Jaccard distance*[2] between the first snapshot and subsequent snapshots one-by-one. As long as the Jaccard distance between the snapshots is less than or equal to 0.5, which means that the intersection of the snapshots is at least as large as half of their union, the subsequent snapshots are assigned to the stay point. The staypoint is considered complete, if the Jaccard distance to new WiFi snapshots grows beyond 0.5 or there is a gap between consecutive WiFi snapshots that is longer than t_gap_{sp}.

Definition 4 (WiFi Stay Point): A *WiFi stay point* $wifi\_sp$ is a sequence of WiFi snapshots $wifi$, in which each snapshot has a Jaccard distance of less than or equal to 0.5 to the first snapshot of the sequence, no consecutive snapshots have a time difference larger than t_gap_{sp}, and the duration of the snapshot is greater or equal to t_min_{sp}:
$wifi\_sp = \{wifi_1, wifi_2, \ldots, wifi_n\}$, such that
(1) $\forall i, 1 < i \leq n : J_\delta(wifi_1, wifi_i) \leq 0.5$,

[2]The Jaccard distance measures the dissimilarity between sets. It is calculated for two sets A and B as $J_\delta(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$

(2) $t(wifi_i) - t(wifi_{i-1}) \leq t\_gap_{sp}$, and
(3) $\mathsf{dur}(wifi\_sp) = t(wifi_n) - t(wifi_1) \geq t\_min_{sp}$.
These criteria for WiFi stay points were selected, because it is not uncommon that WiFi access points are missed by the context scan [17]. This is apparently not dependent on the signal strength of the missed access point, so one needs to take into account that even very strong access point beacons will be missed from time to time.

Each WiFi stay point has a characteristic set of access points $\mathsf{char}(wifi\_sp)$ which includes those access points that occur at least in half of all WiFi snapshots belonging to the stay point.

Definition 5 (Characteristic Set): A *characteristic set* $\mathsf{char}(wifi\_sp)$ of WiFi access points rf for WiFi stay point $wifi\_sp$ is defined as
$\mathsf{char}(wifi\_sp) = \{rf_i | \ \mathsf{count}(rf_i, wifi\_sp) \geq \frac{|wifi\_sp|}{2}\}$, where $\mathsf{count}(rf_i, wifi\_sp) = |\{wifi_k \in wifi\_sp | rf_i \in wifi_k\}|$.
A set of access points is a *WiFi-based CoI*, if there are at least f_min_{coi} WiFi stay points having this set of access points as their characteristic set of access points, and the stay points have a duration of at least t_min_{coi} in total.

Definition 6 (WiFi-based CoIs): Let C be a set of WiFi access points and $\mathsf{wsp}(C)$ be the set of WiFi stay points that have C as their characteristic set, i.e.,
$\mathsf{wsp}(C) = \{wifi\_sp \,|\, \mathsf{char}(wifi\_sp) = C\}$.
The set C is a *WiFi-based CoI*, if following conditions hold:
(1) $|\mathsf{wsp}(C)| \geq f\_min_{coi}$, and
(2) $\sum_{wifi\_sp \in \mathsf{wsp}(C)} \mathsf{dur}(wifi\_sp) \geq t\_min_{coi}$.
A WiFi snapshot *wifi falls within CoI* C, if $J_\delta(C, wifi) \leq 0.5$.

When our example user arrives at her workplace, a WiFi snapshot $wifi$ is recorded. This snapshot and following snapshots having a Jaccard distance of less than or equal to 0.5 to the first one form a WiFi stay point $wifi\_sp$, given that the time difference of the first and last snapshot is greater than t_min_{sp} and there are no gaps in the WiFi snapshot observations longer than t_gap_{sp}. The characteristic set $\mathsf{char}(wifi\_sp)$ of access points of this stay point consists of access points mostly observed at the workplace. During subsequent visits to the workplace, more WiFi stay points with the same characteristic set will be generated. If at least f_min_{coi} such stay points have been observed and the total visit duration $\mathsf{dur}(wifi\_sp)$ of these stay points reaches t_min_{coi}, the characteristic set constitutes a WiFi-based CoI for the user's workplace.

B. Context Detection

Once the GPS- and WiFi-based CoIs have been identified, new incoming GPS, WiFi and Bluetooth observations can be used to identify the location context and social context of the user at any point in time. These are required in order to be able to model, which places a user has visited and which other people the user has encountered.

1) Location context: We first define the notions of *visits* and *visit time*, which are required to determine, when and for how long a user has been visiting a particular CoI.

Definition 7 (Visits): A user's *visit* V_C to a GPS-based CoI $C = (lat_{min}, lon_{min}, lat_{max}, lon_{max})$ is a sequence of position observations $pos_i = (lat_i, lon_i)$ falling within the CoI and having timestamps at most ϵ_V apart from each other: $V_C = (pos_1, pos_2, \ldots, pos_n)$, where $\forall pos_i \in V_C : lat_{min} < lat_i \wedge lon_{min} < lon_i \wedge lat_i < lat_{max} \wedge lon_i < lon_{max}$, and $\forall i, 1 < i \leq n : t(pos_i) - t(pos_{i-1}) < \epsilon_V$. Similarly, a visit V_C to a WiFi-based CoI C is a sequence of WiFi snapshots $wifi$ falling within the CoI and having timestamps at most ϵ_V apart from each other. That is, $V_C = (wifi_1, wifi_2, \ldots, wifi_n)$, where $J_\delta(C, wifi_i) \leq 0.5$ and $\forall i, 1 < i \leq n : t(wifi_i) - t(wifi_{i-1}) < \epsilon_V$. We denote the set of all visits V_C of the user to CoI C with $\mathcal{V}_C$.

In our running example, whenever our user arrives at her workplace, a WiFi snapshot $wifi$ (or a position observation pos) will fall into the workplace CoI. Beginning from this snapshot (or position observation) all subsequent observations falling within the CoI will be part of a visit V to the CoI, as long as the time distance between consecutive snapshots (or position observations) is less than $\epsilon_V = 5$ minutes.

Definition 8 (Visit Time): We define the *visit time* T_V of a visit V to consist of the sequence of timestamps falling between the first and last position observation or wifi snapshot belonging to the visit. That is, $T_V = (t_1, t_2, \ldots, t_n)$, such that $\forall t_i : t(pos_1) \leq t_i \leq t(pos_n)$ for GPS-CoI visits and $t(wifi_1) \leq t_i \leq t(wifi_n)$ for WiFi-CoI visits. If a visit V consists of only a single position observation or WiFi snapshot, i.e., $V = (pos_i)$ or $V = (wifi_i)$, we *assume* that the visit covers a timespan of half of the duration of the scanning interval t_{scan}: $T_V = (t_{i-k}, t_{i-k+1}, \ldots, t_i, \ldots, t_{i+k-1}, t_{i+k})$, where $t_{i+k} - t_{i-k} = \frac{t_{scan}}{2}$ and $t_i = t(pos_i)$ or $t_i = t(wifi_i)$, respectively.

Following this definition, the visit time T_V of a visit V of our example user to the grocery store CoI C covers all timestamps t_i starting from her arrival, i.e., the timestamp of the first WiFi snapshot $t(wifi_1)$ (or GPS observation $t(pos_1)$) falling into the CoI up until to when she leaves the CoI, i.e., until the timestamp $t(wifi_n)$ (or $t(pos_n)$) of the last WiFi snapshot $wifi_n$ (or GPS observation pos_n) that falls into the CoI.

Definition 9 (Location Context): A *location context* L_t at timestamp t is the set of CoIs C that the user is visiting during that point of time, i.e., for which there exists a visit V, whose visit time T_V contains t, i.e., $L_t = \{C \,|\, \exists V \in \mathcal{V}_C : t \in T_V\}$
Note, that CoIs can be overlapping, which means that a user can be visiting several CoIs simultaneously. If the user is not visiting any of the CoIs at a specific point in time, the corresponding location context will be empty.

Definition 10 (Familiar CoIs): The set of *Familiar CoIs* C_{fam} is the set of all such CoIs that are particularly important for the user, that is, CoIs where he has spent longer than t_min_{famcoi} of time during at least f_min_{famcoi} visits. That is, $C_{fam} = \{C_1, C_2, \ldots, C_n\}$, such that $\forall\, C_i \in C_{fam} : \sum_{V \in \mathcal{V}_{C_i}} T_V \geq t\_min_{famcoi} \wedge |\mathcal{V}_{C_i}| \geq f\_min_{famcoi}$.
For our example user, familiar CoIs would be identified at places like her home or her workplace, since she has visited

these places more often than $f\_min_{famcoi} = 5$ times and has spent altogether longer than $t\_min_{famcoi} = 60$ minutes during these visits.

2) Social context: The notions of *encounter* and *encounter time* are required in order to be able to identify, which devices (representing other people) and for how long the user's mobile device has encountered, so that a distinction of familiar and unfamiliar people in the social context can be made.

Definition 11 (Encounters): An *encounter* E_d of a user with a device d is a sequence of Bluetooth observations bt_i of device d with timestamps that are at most ϵ_E apart from each other: $E = (bt_1, bt_2, \ldots, bt_n)$, where $\forall i, 1 < i \leq n : bt_i = d \wedge t(bt_i) - t(bt_{i-1}) < \epsilon_E$. We denote the set of all encounters of the user with a device d with $\mathcal{E}_d$.

When our example user arrives at her workplace, her device obtains a Bluetooth observation $bt_1 = d$ of her colleague's device d. This observation and any subsequent device observations $bt_i = d$ of the colleague's device form an encounter E_d with the colleague's device, as long as the time distance between consecutive device observations is less than $\epsilon_E = 5$ minutes.

The purpose of allowing gaps of at most $\epsilon_E = 5$ minutes is to be able to handle missed device observations (which are not uncommon with Bluetooth sensing) without breaking a contiguous sequence of device observations into two separate encounters too easily.

Definition 12 (Encounter Time): We define the *encounter time* T_E of an encounter E to consist of the sequence of timestamps falling between the first and last device observation of the encounter. That is $T_E = (t_1, t_2, \ldots, t_n)$, such that $\forall t_i : t(bt_1) \leq t_i \leq t(bt_n)$. If an Encounter E consists of a single device observation, i.e., $E = (bt_i)$, we *assume* that the encounter covers a timespan of half of the duration of the scanning interval t_{scan} and set: $T_E = (t_{i-k}, t_{i-k+1}, \ldots, t_i, \ldots, t_{i+k-1}, t_{i+k})$, where $t_{i+k} - t_{i-k} = \frac{t_{scan}}{2}$ and $t_i = t(bt_i)$.

The encounter time T_E of an encounter of our example user with the device of her colleague would cover all timestamps beginning from timestamp $t(bt_1)$ of the first observation bt_1 of the colleague's device, up until timestamp $t(bt_n)$ of the last observation bt_n of the colleague's device, made immediately before she leaves work and the colleague's device gets out of range.

Definition 13 (Device Context): A *device context* D_t at timestamp t is the set of devices d that are encountered during that point of time, i.e., for which there exists an encounter E, whose encounter time T_E contains t, i.e., $D_t = \{d \mid \exists E \in \mathcal{E}_d : t \in T_E\}$.

Definition 14 (Familiar Devices): The set of *familiar devices* $\mathcal{D}_{fam}$ is the set of all such devices that the user has encountered at least f_min_{famdev} times and for which the total duration of the encounters is at least t_min_{famdev}. That is, $\mathcal{D}_{fam} = \{d_1, d_2, \ldots, d_n\}$, such that $\forall d_i \in \mathcal{D}_{fam} : \sum_{E \in \mathcal{E}_d} T_E \geq t\_min_{famdev} \wedge |\mathcal{E}_d| \geq f\_min_{famdev}$.

Familiar devices d for our example user would be the mobile devices of familiar people like her spouse or her colleagues at work which she has encountered more often than $f\_min_{famdev} = 5$ times and the total duration of these encounters is longer than $t\_min_{famdev} = 30$ minutes.

However, a mobile device of a person like the member of the house cleaning service at her office would not become a familiar device, since even though that person's device might be encountered nearly daily, the encounters would typically be of such short duration that they would not aggregate significant amounts of total encounter time.

C. Context Profiles

Based on the above context model, the following context profiles are aggregated for the user: a *CoI profile* $CoIs$ and a *device profile* $Devs$. The CoI profile $CoIs = \{\mathcal{C}, C_{fam}, P\}$ consists of the set of all identified CoIs $\mathcal{C}$, the set of familiar CoIs C_{fam} and mapping $P : \mathcal{C} \rightarrow \mathbb{N} \times \mathbb{R}, C \mapsto (visits_C, dur_C)$ providing the total amount of visits $visits_C$ and total duration of visits dur_C to each CoI $C \in \mathcal{C}$.

Similarly, the device profile $Devs = \{\mathcal{D}, \mathcal{D}_{fam}, O\}$ consists of the set of all encountered devices $\mathcal{D}$, the set of familiar devices $\mathcal{D}_{fam}$ and a mapping $O : \mathcal{D} \rightarrow \mathbb{N} \times \mathbb{R}, d \mapsto (enc_d, dur_d)$ providing the total amount enc_d and total duration dur_d of encounters with each device $d \in \mathcal{D}$.

D. Context Evaluation Functions

Using the above context profiles, we instantiate two alternative versions of the context evaluation functions $\hat{\lambda}$ and $\hat{\phi}$ introduced in section IV: heuristic and inductive.

1) Heuristic evaluation functions: In section III, we identified the major factors affecting the user's perceptions of context sensitivity and safety. Following these factors, we formulate following heuristics for evaluating the sensitivity and safety of contexts:

(a) If one is in a familiar CoI, the context is sensitive.
(b) If one is in a familiar CoI, the context is also safe, unless there are unfamiliar devices in the context.

Definition 15 (Heuristic Sensitivity Evaluation): Let $c(t) = (D_t, L_t)$ denote the *context* of the user's device at timestamp t. The *heuristic evaluation function for sensitivity* $\hat{\lambda}_{heur}(c(t))$ evaluates context $c(t)$ as *sensitive* if any CoI C in the current location context L_t is a familiar CoI. That is

$$\hat{\lambda}_{heur}(c(t)) = \begin{cases} \textit{sensitive} & L_t \cap C_{fam} \neq \emptyset \\ \textit{public} & L_t \cap C_{fam} = \emptyset \end{cases}$$

Definition 16 (Heuristic Safety Evaluation): The *heuristic evaluation function for safety* $\hat{\phi}_{heur}(c(t))$ evaluates context $c(t)$ as *safe* if the user is in a familiar CoI and at most d_max_{unfam} devices d in the device context D_t are not familiar. That is

$$\hat{\phi}_{heur}(c(t)) = \begin{cases} \textit{safe} & |D_t - \mathcal{D}_{fam}| \leq d\_max_{unfam} \\ \textit{unsafe} & \text{otherwise.} \end{cases}$$

2) Inductive evaluation functions: The inductive evaluation functions $\hat{\lambda}_{ind}$ and $\hat{\phi}_{ind}$ utilize statistical machine learning models that are based on supervised learning. This means that the models are trained with labeled training vectors $(label, \vec{tr})$, where $\vec{tr} = (f_1, f_2, \ldots, f_n)$. The labels for the training vectors

are obtained from *user feedback*, i.e., from events where the user of the device has provided explicit feedback about the perceived sensitivity and safety of the context:

Definition 17 (User Feedback): A *user feedback* $fb(t)$ at timepoint t is a tuple (fb_{sens}, fb_{safe}), where $fb_{sens} \in \{sensitive, public\}$ and $fb_{safe} \in \{safe, unsafe\}$.

For each user feedback event $fb(t)$, a training vector $\vec{tr}(t)$ is generated based on the context $c(t)$. The components f_i (also called *features*) of $\vec{tr}(t)$ are calculated based on the current context at time of the feedback event, $c(t) = (L_t, D_t)$ and the aggregated context profiles $CoIs$ and $Devs$. The values of the individual components of the training vectors $\vec{tr}$ are detailed in table III.

The training vectors are used to train two classifier models M, one for predicting the sensitivity of a context, M_{sens}, and one for predicting the context's safety, M_{safe}.

$$M^a_{sens} = train(a, F_{sens}),\ M^a_{safe} = train(a, F_{safe})$$

where a denotes the used machine learning algorithm and $F_{sens} = ((fb_{sens}(t_1), \vec{tr}(t_1)), \ldots, (fb_{sens}(t_n), \vec{tr}(t_n)))$ and $F_{safe} = ((fb_{safe}(t_1), \vec{tr}(t_1)), \ldots, (fb_{safe}(t_n), \vec{tr}(t_n)))$ denote the sequences of labeled training data vectors. Using the trained classification models, we can now define the inductive context evaluation functions $\hat{\lambda}$ and $\hat{\phi}$ as follows:

$$\hat{\lambda}(c(t)) = classify(a, M^a_{sens}, \vec{e}(t))$$

$$\hat{\phi}(c(t)) = classify(a, M^a_{safe}, \vec{e}(t))$$

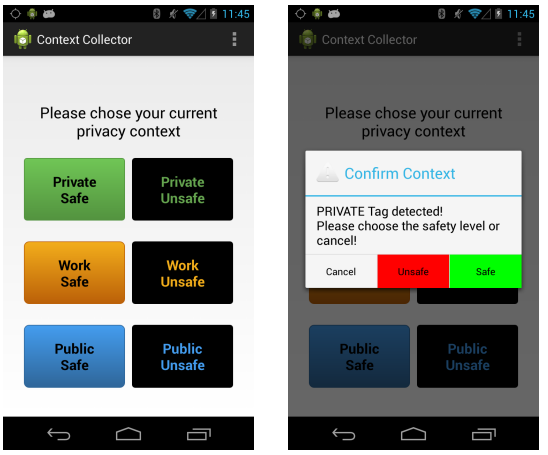
where $\vec{e}(t) = (f_1, f_2, \ldots, f_n)$ denotes an evaluation vector derived from the context $c(t)$ and the context profiles $CoIs$ and $Devs$. The components of the evaluation vector are the same as for the training vectors $\vec{tr}$ and are detailed in table III.

VI. Implementation and Evaluation

Our implementation of ConXsense is based on the Android OS and comprises three main components: 1) a Context Data Collector app for mobile devices which collects contextual data and ground truth feedback; 2) A ContextProfiler for preprocessing and evaluating the collected data and generating predictions about the sensitivity/safety of contexts; 3) An Access Control Layer based on the *FlaskDroid* architecture [18] which enforces context-dependent access control policies using the context predictions of the ContextProfiler.

A. Context Data Collector

1) Implementation: Our Context Data Collector app for the Android operating system uses a background Service to collect context data in intervals of 60 seconds. This is a required tradeoff between the battery lifetime and the quantity of the collected data, since for the data collection we aim at a battery lifetime of at least 12 hours. The collected context data comprises location information (GPS, WiFi and cellular network triangulation), nearby Bluetooth devices and WiFi access points, acceleration sensor information as well as information about user presence and her interaction with apps (Activities).[3] In order to evaluate the context evaluation algorithms, the Context Data Collector app also collects ground truth data from the participating users. The users regularly report their perceived sensitivity and safety of the current context in one of the following ways (cf. Figure 3): 1) The participants use context feedback buttons on the device's UI; 2) NFC tags provided to the participants are used to trigger context reporting; 3) If no ground truth data has been provided by the user in the last two hours, the app reminds the user to do so via sound, vibration and flashing LED notifications. Context and ground truth data are stored in a SQLite database and periodically uploaded to a server via HTTPS.



(a) Feedback using Context Feedback Buttons

(b) Feedback using Context NFC Tags.

Fig. 3: Android Data Collector App

2) Evaluation: We installed the Context Data Collector app on the Android smartphones of 15 test users. We then collected data over a period of 18-68 days from different users, 56 days per user on the average. The total dataset contained data from 844 distinct user days. On the average, users provided ground truth feedback on 46 days of the data collection period, resulting in a ground truth dataset containing 3757 labeled data points about the *sensitivity* of the context. Based on early results of the user survey we conducted in parallel, we decided to also incorporate feedback about the perceived safety of the context to the Context Data Collector. Thereby, the ground truth dataset could be enriched with 1861 labeled data points providing also feedback about the users' perception of the *safety* of the context.

Since our test participants tended to spend most of their time in contexts that are sensitive and/or safe, the distributions of ground truth are also skewed towards *sensitive* and *safe* feedbacks. 80.44% of the ground truth feedback specified contexts as being *sensitive* and 19.56% as *public*. For 'safety' the distribution was 81.84% for *safe* and 18.16% for *unsafe*.

Some users had provided only very little feedback for

[3] Context Data Collector is a generic solution collecting more data than required for the current ConXsense algorithms.

TABLE III: Features for training the inductive evaluation classifiers

	Feature name	Description
f_1	max-gps-coi-visit-time	Maximum visit time of any GPS-based CoI in L_t
f_2	nbr-gps-coi-visits	Number of visits to GPS-based CoI with maximum visit time
f_3	max-wifi-coi-visit-time	Maximum visit time of any WiFi-based CoI in L_t
f_4	nbr-wifi-coi-visits	Number of visits to WiFi-based CoI with maximum visit time
f_5	nbr-btdev	Number of Bluetooth devices in D_t, i.e. $\|D_t\|$
f_6	nbr-fam-btdv	Number of familiar Bluetooth devices in D_t, i.e. $\|D_t \cap \mathcal{D}_{fam}\|$
f_7	avg-encounter-time	Average encounter time of familiar devices in D_t
f_9	avg-nbr-encounters	Average number of encounters of familiar devices in D_t

some ground truth classes (especially for *public* and *unsafe*). Training a classifier with too few data points for a class would not provide meaningful results, and therefore we decided to reject such users' ground truth datasets from our evaluation which did not contain at least 10 data points in a ground truth class. Hence, we had to remove the data of four participants from the sensitivity dataset, so that our evaluation dataset contained a total of 3512 labeled data points (319.2 per user on average). For the ground truth dataset for safety, eight participants had provided sufficient amounts of feedback, so that the evaluation dataset for safety contained finally a total of 1551 labeled data points (193.9 points per user on average).

3) Battery lifetime: During the data collection a battery lifetime of at least a working day (12h) was reported by the users on Samsung Galaxy Nexus and Nexus S devices, which we consider reasonable since no optimization regarding power consumption has been performed yet.

B. Context Profiler

1) Implementation: We implemented the functionality of the ContextProfiler as off-line data processing scripts utilizing `bash` shell scripting, `awk` and `Python`. The scripts were used to identify individual GPS and WiFi CoIs for each user, and to calculate the familiarity of Bluetooth devices that the users had encountered during the data collection period. The heuristic context evaluation functions were also implemented as data processing scripts. The scripts were also used for extracting the feature vectors for training and evaluating the inductive evaluation functions which were realized and evaluated by using the Weka data mining suite [19] and its provided algorithm implementations for k-NN, Random Forest and Naïve Bayes classfiers.

2) Evaluation: The heuristic evaluation functions $\hat{\lambda}_{heur}(c(t))$ and $\hat{\phi}_{heur}(c(t))$ were run on all context observations $c(t)$ for which also ground truth feedback $fb(t)$ was available. The outputs of the heuristic evaluation functions were then compared with the ground truth $fb(t)$ and the protection levels $\pi_{\hat{\lambda}_{heur}}$ and $\pi_{\hat{\phi}_{heur}}$ and the usability deterioration measures $\xi_{\hat{\lambda}}$ and $\xi_{\hat{\phi}}$ (cf. section IV) were calculated.

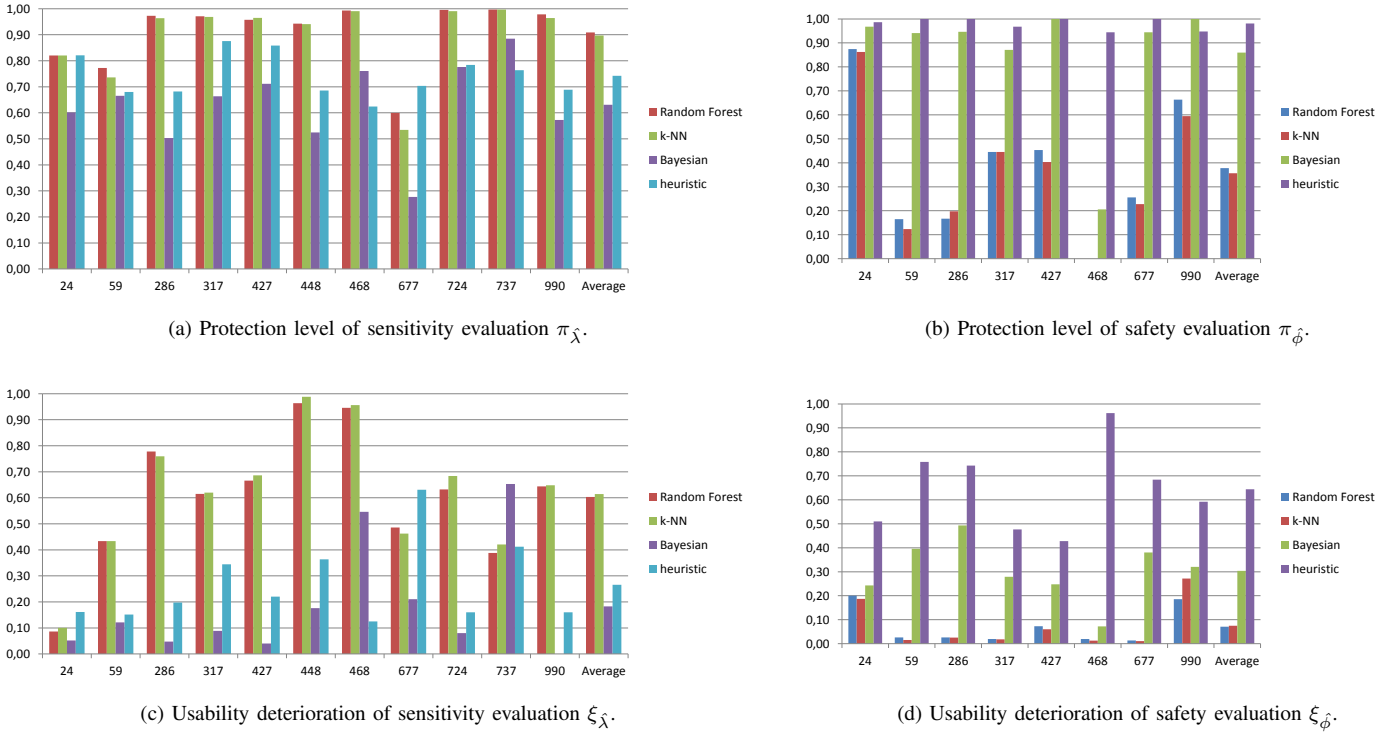
The same input data were used also to investigate three alternative inductive context evaluation functions based on popular classifier algorithms: $\hat{\lambda}_{ind}^{kNN}$ and $\hat{\phi}_{ind}^{kNN}$ utilizing a k-nearest neighbour (kNN) classifier, $\hat{\lambda}_{ind}^{NB}$ and $\hat{\phi}_{ind}^{NB}$, based on a Naïve Bayesian classifier and $\hat{\lambda}_{ind}^{RF}$ and $\hat{\phi}_{ind}^{RF}$ using a Random Forest classifier.

The k-nearest neighbours (kNN) classifier bases its prediction on comparing a testing datapoint to the n closest observations to it in the training dataset. The prediction is the most frequent class label in this set of observations. The Naïve Bayes classifier is a simple probabilistic classifier which has been successfully used, e.g., in spam e-mail detection [20]. Random Forest is an ensemble method, that is commonly used for classification tasks. It randomly picks subsets of input attributes and trains decision trees for them. It uses the most frequently predicted label provided by this set of tree classifiers as the final prediction.

The inductive context evaluation functions $\hat{\lambda}_{ind}^{kNN}, \hat{\lambda}_{ind}^{NB}, \hat{\lambda}_{ind}^{RF}$ and $\hat{\phi}_{ind}^{kNN}, \hat{\phi}_{ind}^{NB}, \hat{\phi}_{ind}^{RF}$ were tested using 10-fold cross-validation and the above-mentioned figures of merit were calculated for each algorithm and each user. The results of the evaluation are shown in figure 4.

As can be seen from figure 4a, all tested context evaluation functions provide reasonable results for context sensitivity protection levels, the best results being provided by the Random Forest and k-NN classifier-based functions (average of both being > 0.9). However, looking at the usability deterioration metrics in figure 4c, one sees that these evaluation functions pay a high price for their high protection level: the level of usability deterioration is for many users very significant (average being > 0.6). The Bayesian classifier-based evaluation functions shows significantly better performance in this regard (average being 0.18), but also has to acknowledge a lower protection level (average being 0.63). Somewhat surprising is, how well the very simple heuristic evaluation functions perform in sensitivity evaluation: even though the heuristic context sensitivity function provides relatively high protection levels (average being 0.74), the usability deterioration measures remain acceptably moderate (average being 0.27).

For safety evaluation, the situation is very different. As seen in figure 4b, both Random Forest and k-NN classifier-based evaluation functions fail to provide sufficient protection levels (average being < 0.40). This has probably to do with the fact that contrary to the sensitivity case, here the evaluation functions are required to predict the more infrequent (*unsafe*) context labels correctly. The only evaluation functions that perform adequately with regard to safety protection level are the Bayesian classifier-based and heuristic-based evaluation functions (showing average protection levels of 0.86 and 0.98, respectively). Of these, however, only the Bayesian



(a) Protection level of sensitivity evaluation $\pi_{\hat{\lambda}}$.

(b) Protection level of safety evaluation $\pi_{\hat{\phi}}$.

(c) Usability deterioration of sensitivity evaluation $\xi_{\hat{\lambda}}$.

(d) Usability deterioration of safety evaluation $\xi_{\hat{\phi}}$.

Fig. 4: Protection level and usability deterioration for sensitivity and safety evaluation functions $\hat{\lambda}$ and $\hat{\phi}$

classifier-based evaluation function shows also an acceptably low usability deterioration measure, as can be seen from figure 4d.

C. Access Control Layer

1) Implementation: For our Access Control Layer we adopt and adapt the *FlaskDroid* [18] architecture, a fine-grained mandatory access control framework for Android 4.0.4 (cf. Figure 2). *FlaskDroid* extends *Security Enhanced Android (SEAndroid)* [21] with fine-grained context-aware type enforcement on Android's middleware layer. In *FlaskDroid*, Android components which provide access to sensitive resources, such as the SensorService which provides access to sensor information, are modified to act as UserSpace Object Managers (USOMs) which control access to the resources they manage. More specifically, USOMs control access from $subjects$ (i.e., apps) to $objects$ (e.g., data) they manage using $types$ assigned to $subjects$ and $objects$. Thereby USOMs act as Policy Enforcement Points (PEPs) in our architecture.

FlaskDroid uses an Access Control Policy (cf. Figure 2) which describes types and access control rules. An access control rule defines whether a certain $subject\ type$ may perform an $operation$ on an $object\ type$ of a certain $object\ class$. In addition, *FlaskDroid* supports context-dependent access control rules by means of ContextProviders – software components which evaluate the current context of the device and activate/deactivate rules at runtime.

At boot time, the PolicyServer (cf. Figure 2) parses the Access Control Policy and translates it into an in-memory representation. It proceeds to assign app types (e.g., *trusted* or *untrusted*) to all installed apps based on application meta-data, such as the package name or the developer signature. Applications installed by the user are assigned corresponding types during the installation process. Whenever apps access an USOM at runtime, for example the SensorService to query the device's sensors or the CameraService to take pictures, the USOM queries the PolicyServer, which is part of Android's SystemService, for access control decisions.

To meet our goals we extended *FlaskDroid* with additional USOMs. We furthermore implemented a ContextProvider which feeds context tags from the ConXsense ContextProfiler into *Flaskdroid's* PolicyServer (cf. Figure 2), which in turn activates/deactivates access control rules at runtime.

To mitigate, respectively reduce the effects of sensory malware, such as *Placeraider* [4] or SoundComber [3], access control on the sensors of a device is required. For example, *Placeraider* uses the device's camera and the acceleration sensor to covertly construct 3D images of the surroundings of the user. To mitigate the effect of Placeraider, we extended Flaskdroid to perform access control in Android's CameraService using a corresponding USOM which filters queries to the *takePicture* and *startPreviewMode* methods based on the $type$ of the calling app. Furthermore, we used *Flaskdroid's* access control on sensors to filter events delivered to SensorEventListeners registered by apps. It should be noted that in *Flaskdroid's* original implementation the enforce-

ment points in the SensorManager Java class are insufficient to block sophisticated attacks, since the SensorManager class is executed in the context of (potentially malicious) apps. Thus, we replaced *Flaskdroid's* SensorManager USOM with a corresponding USOM in Android's native SensorService. Similarily, the combination of ConXsense and *FlaskDroid* can be used to address other variants of sensory malware, such as *Soundcomber*, by identifying the relevant Android APIs, instrumenting them as USOMs and extending the *FlaskDroid* policy with corresponding context-dependent access control rules.

To allow for changes in the Android Lockscreen policy based on the current privacy context, we modified the PolicyServer to proactively propagate context tags to the Android Lockscreen component. We modified Android's Lockscreen component to automatically dismiss the Lockscreen when the device is used in a safe environment, while still showing the Lockscreen in unsafe environments.

2) Evaluation: We evaluate the effectiveness and performance of our Access Control Layer in two use cases: 1) preventing sensory malware from gathering sensitive sensor information in sensitive contexts, and 2) enhancing the usability of Android's Lockscreen security feature using context information. For this evaluation we used a Samsung Galaxy Nexus smartphone equipped with *FlaskDroid* for Android 4.0.4 and the ConXsense extensions.

2a) Mitigation of Sensory Malware: PlaceRaider [4] is a sensory malware which generates 3D models of the user's surroundings by combining acceleration sensor information with camera pictures. *PlaceRaider* registers listeners in Android's SensorManager to get notified about changes in the orientation of the device. When a significant orientation change is discovered, PlaceRaider takes a picture using the Camera API without user notification. It then uploads the picture and the corresponding sensor information to an external server. When enough pictures are collected, the server constructs a 3D view of the surroundings of the user.

To demonstrate the feasibility of ConXsense we designed a *FlaskDroid* policy which assigns the type $trusted$ to all pre-installed system apps (e.g., the camera app), and the type $untrusted$ to all third-party apps installed by the user. In a real-world scenario this trust level could be derived from the reputation of the app in an app market. Our context-dependent access control rules state that access to the CameraService and SensorService USOMs is generally allowed for all apps in $public$ contexts but is prohibited in $sensitive$ contexts for all $untrusted$ apps.

We tested our implementation using a slightly modified version of the *PlaceRaider* malware generously provided to us by its authors.[4] By installing the malware on our device and logging the context information and access control decisions we verified that *FlaskDroid* successfully filtered all data delivered from Android's SensorService and CameraService components to the *untrusted PlaceRaider* app when the device was used in a sensitive context, thus rendering the attack futile. We further verified that *trusted* apps could still use the sensors and the camera. No false positives or false negatives emerged during the evaluation of the Access Control Layer, which is not surprising since it merely enforces the context-dependent access control rules.

To evaluate the performance impact we implemented an app which automatically triggers 10000 access control queries by reading sensor data and taking pictures. On average, our *FlaskDroid* based Access Control Layer caused an overhead μ of 4.886 ms (standard deviation σ 17.593 ms) for the SensorService and CameraService USOMs. The high standard deviation σ is caused by Dalvik's garbage collector: By analyzing Android's system logs we verified that during the irregularly slow access control queries responsible for the high standard deviation the garbage collector caused a stall. Overall, 95% of all access control decisions are handled in less than 4.2 ms (cf. Figure 5), which we consider reasonable.

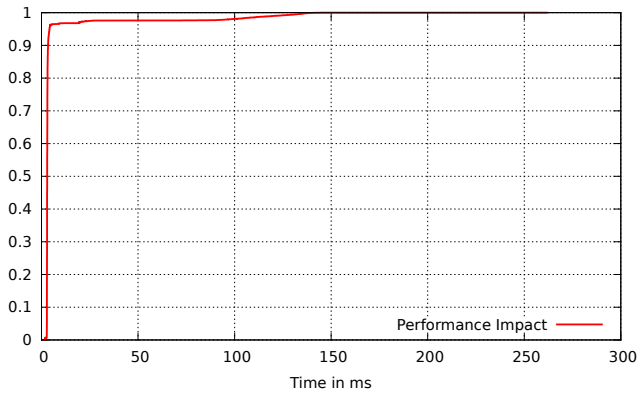


Fig. 5: Cumulative distribution of the Access Control Layer performance

2b) Context-Aware Device Lockscreen: Android provides different methods for locking the device whenever the power button is pressed or the device is not used for a pre-defined amount of time. However, the locking mechanism is static and does not respect the current usage context: When used in a *safe* context, it might be desirable to enforce relaxed security policies which do not require the user to manually unlock the device using a PIN or password. However, when a context is *unsafe*, the Lockscreen must always be displayed. On stock Android, the user might be tempted to disable the Lockscreen regardless of the current context, which is problematic in case the device is stolen or lost.

We instrumented Android's Settings component to be notified by *Flaskdroid's* PolicyServer about context changes by means of a Broadcast Intent. To enforce context-based access control rules we modified the LockPatternKeyguardView class to query the Settings component for the current value of the safety property. If the context is considered *safe* and hence the Lockscreen should not be active, the LockPatternKeyguardView class automatically dismisses the Lockscreen. We furthermore added a low-watermark mechanism to ensure that

[4]The original *PlaceRaider* malware is incompatible with Android 4.0.4.

whenever an intermediate *unsafe* context has been detected or the device has been rebooted, the Lockscreen is always displayed regardless of the current context's safety property. The low-watermark mechanism is required to prevent an attacker from unlocking a locked device stolen in an *unsafe* context by moving it to a context the owner considers *safe*.

We instrumented the LockPatternKeyguardView class to log when the Lockscreen was displayed and when it was dismissed. We further modified the system to periodically wake the device from sleep and switch on the screen. In an automated testbed we verified that the Lockscreen was automatically dismissed whenever the context was considered *safe* unless an intermediate or current *unsafe* context has been detected or the device has been rebooted.

VII. Related Work

Covington et al. [8] introduce a policy framework incorporating context attributes in a smart home access control setting. They utilise a *Generalised Role Based Access Control* (GRBAC) model and Environment Roles activated by context observations to adapt access control decisions. They demonstrate an XML-based definition language for the access control policies, but require those to be set up and maintained manually through a GUI tool.

Hull et al. [22] present the *Houdini* framework for specifying and enforcing context-dependent privacy policies. They introduce rule-based policy management to mitigate the complexity that value-based customization of policies would imply. User-provided preferences are used to generate rules for privacy enforcement. They mention also support for automatically-learned preferences that would be transformed into rules, but do not provide support for such automation at the time of writing.

Damiani et al. [23] introduce a spatially-aware RBAC model using location as a component for access control decisions. They focus on the theoretical aspects of the model and do not address, how the required access control rules would be created and maintained in real-world implementations.

A recent patent application by Bell et al. [24] discloses a system which enables to specify context-triggered policies enforced on mobile devices controlling the access of applications to sensors and other resources on the smartphone. Also in their approach the access control policies are either pre-specified or are uploaded to the devices by external entities.

Conti et al. [9] describe the *CRePe* framework for Android for enforcement of context-dependent access control policies. The policies are expressed in the form of rules, which allow or deny access to specific resources depending on the currently detected active context. Also they do not address, how the rules should be created or updated. In the *MOSES* framework [10] Rusello et al. propose a combination of dynamic taint tracking using the *TaintDroid* architecture [25] and policy enforcement on Android's middleware layer to enable context based access control on resources and apps with the goal of providing isolated environments called security profiles. Similarly, the *TrustDroid* [26] architecture proposed by Bugiel et al. provides lightweight security domain isolation on Android with basic support for context-based network access control policies. In Saint [27], the authors describe a context-aware fine-grained access control framework for Android, which focuses on enabling app developers to define context-dependent runtime constraints on Inter-Component Communication between apps. Nauman et al. present *Apex* [28], a modification of the Android operating system which extends the standard permission system with conditional permissions. It provides to some extent support for context-based access control by allowing the user to define context-dependent resource restrictions (e.g., based on the time of day).

In contrast to *MOSES*, *TrustDroid* and *Apex*, our access control architecture is based on the more generic and flexible *FlaskDroid* platform [18], which is also able to cover (most of) the use cases described in *Saint*, and, more importantly, our work focuses on the automatic prediction of the active security context of a device using a probabilistic approach based on heuristics and machine learning algorithms.

Riva et al. [11] use various contextual cues to estimate the likelihood that the legitimate user of a device is in proximity and use this to configure the idle screen lock. Although we have the same use case, our approach is based on estimating the safety of the context, rather than inferring the presence of the user.

Hayashi et al. [29] introduce *Context-Aware Scalable Authentication*, an approach which uses the location of the device as a passive authentication factor in a probabilistic framework to determine the active authentication factors to be used for user authentication (e.g., PIN or password) on smartphones. While the context-aware Lockscreen implemented in ConXsense resembles their approach, our approach is different in that it builds on relatively complex modeling of dynamic contexts aiming at determining the dynamic safety level of the context based on several context features. In contrast, their approach utilizes a comparatively simple context detection method that requires users to explicitly name and classify contexts as "home", "work" or "other". Their safety classificaton of distinct contexts is also static and depends purely on locations and safety labels users have provided for these locations.

Danezis [30] proposes an approach for automatically defining privacy settings in social networking applications. Therein, *security contexts* are determined by analyzing the social graphs of users and identifying highly connected subgraphs. The identified subgraphs are used as a basis for social network security settings. Similarly to us also, Danezis justifies the need for inferring privacy policies automatically by the overwhelming burden that managing security settings imposes on the user.

Sadeh et al. [31] investigate a policy definition and management system for the *PeopleFinder* application. They argue, that users are not very successful in defining privacy policies on their own, reaching only accuracy levels of around 60-70% for the policy sets that they initially define and refine. They also use case-based reasoning (CBR) and Random Forest classifiers in making policy decisions, reporting significant improvement

over user-defined policy sets. In a follow-up work, Kelley et al. [32] introduced a user-controllable policy learning system that builds on incremental policy improvements proposed to the users based on recorded history events.

Bai et al. propose a solution for fine-grained usage control on Android [33]. Their work extends the UCON access control model [34] by using context information (e.g., location and time) as an additional input for policy decisions. While they focus on the implementation of a generic context-aware usage control framework, we focus on the automatic prediction of the active security context based on the surroundings of a device.

Kang et al. [35]. introduced the idea of time-based clustering of position observations underlying our method of identifying stay points. The concept of stay points and stay regions were introduced by Zheng et al. [36] and developed further by Montoliu et al. [37]. We have adopted a slightly modified form of the notion of stay areas in our concept of GPS-based CoIs. We also extend the notion of a stay point to non-locational data in the form of WiFi stay points. The use of WiFi fingerprints for identification and detection of places has been reported by Dousse et al. [17]. We use key learnings from them and adopt a simplified version of their scheme considering only intersections of WiFi snapshots for our WiFi-based CoI detection.

Gupta et al. [7] were the first to use context profiling. Our work extends their work in several ways: First, they did not have reliable ground truth data whereas our data collection produced reliable ground truth data; second, they only attempted heuristic techniques to predict safety but we use both heuristic and inductive techniques and we predict both safety and sensitivity. Finally, we integrated ConXsense into a fine-grained access control enforcement architecture and demonstrated its effectiveness.

VIII. Conclusions and Future Work

In this paper, we described how we designed and implemented ConXsense, a framework for adaptive and easy-to-use access control. We demonstrated its effectiveness in two ways: by collecting ground truth data and using it to evaluate our ConXsense context evaluation algorithms and by applying ConXsense, integrated with a fine-grained access control architecture, to defend against sensory malware and potential device misuse. ConXsense can be extended in a number of directions which we are currently working on: One aspect is extending the context model and context evaluation to enable personalization of individual contexts. This will make it possible to address also the "toddler-scenario" that we alluded to earlier. Moreover, having validated the effectiveness of ConXsense context evaluation, the next step is to evaluate its usability. We plan to implement on-device versions of our context evaluation algorithms and create a downloadable mobile app that we can use for user studies focusing on the usability aspects related to our framework.

References

[1] N. Xu, F. Zhang, Y. Luo, W. Jia, D. Xuan, and J. Teng, "Stealthy video capturer: a new video-based spyware in 3g smartphones," in *Proceedings of the second ACM WiSec*, ser. WiSec '09. New York, NY, USA: ACM, 2009, pp. 69–78. [Online]. Available: http://doi.acm.org/10.1145/1514274.1514285

[2] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers," in *ACM CCS*. ACM, 2011, pp. 551–562. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046771

[3] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones," in *NDSS*, 2011, pp. 17–33.

[4] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia, "PlaceRaider: Virtual theft in physical spaces with smartphones," in *NDSS*, Feb. 2013.

[5] C. Camp, "The BYOD security challenge: How scary is the iPad, tablet, smartphone surge?" WeLiveSecurity Blog post, February 2012.

[6] R. Siciliano, "More than 30% of people don't password protect their mobile devices," McAfee Blog Central, February 2013.

[7] A. Gupta, M. Miettinen, N. Asokan, and M. Nagy, "Intuitive security policy configuration in mobile devices using context profiling," in *SocialCom/PASSAT*. IEEE, 2012, pp. 471–480.

[8] M. Covington, P. Fogla, Z. Zhan, and M. Ahamad, "A context-aware security architecture for emerging applications," in *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, 2002, pp. 249 – 258.

[9] M. Conti, V. Nguyen, and B. Crispo, "CRePE: Context-Related Policy Enforcement for Android," in *ISC 2011*, ser. LNCS, vol. 6531. Springer, 2011, pp. 331–345. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-18178-8_29

[10] G. Russello, M. Conti, B. Crispo, and E. Fernandes, "Moses: supporting operation modes on smartphones," in *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, ser. SACMAT '12. New York, NY, USA: ACM, 2012, pp. 3–12. [Online]. Available: http://doi.acm.org/10.1145/2295136.2295140

[11] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos, "Progressive authentication: deciding when to authenticate on mobile phones," in *Proceedings of the 21st USENIX Security Symposium*, 2012.

[12] A. Tashakkori and C. Teddlie, *Handbook of Mixed Methods in Social & Behavioral Research*. SAGE Publications, 2003.

[13] C. Teddlie and A. Tashakkori, *The Quantitative Tradition: Basic Terminology and two Prototypes*. SAGE Publications, 2009, ch. 1, pp. 5–6.

[14] B. Alan and E. Bell, *Business Research Methods*. Oxford University Press, Incorporated, 2007.

[15] U. Beck, *Risk Society: Towards a New Modernity*, ser. In: association with Theory, Culture & Society. SAGE Publications, 1992. [Online]. Available: http://books.google.de/books?id=QUDMaGlCuEQC

[16] S. Cohen and L. Taylor, *Escape Attempts: The Theory and Practice of Resistance in Everyday Life*. Taylor & Francis, 1992. [Online]. Available: http://books.google.de/books?id=-ijAXz4QCxwC

[17] O. Dousse, J. Eberle, and M. Mertens, "Place Learning via Direct WiFi Fingerprint Clustering," in *Mobile Data Management (MDM), IEEE 13th International Conference on*, 2012, pp. 282–287.

[18] S. Bugiel, S. Heuser, and A.-R. Sadeghi, "Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies," in *22nd USENIX Security Symposium (USENIX Security '13)*. USENIX, 2013.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[20] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk e-mail," in *Learning for Text Categorization: Papers from the 1998 workshop*, vol. 62, 1998, pp. 98–105.

[21] S. Smalley and R. Craig, "Security Enhanced (SE) Android: Bringing Flexible MAC to Android," in *NDSS*. The Internet Society, 2013.

[22] R. Hull, B. Kumar, D. Lieuwen, P. Patel-Schneider, A. Sahuguet, S. Varadarajan, and A. Vyas, "Enabling context-aware and privacy-conscious user data sharing," in *Mobile Data Management, 2004. Proceedings. IEEE International Conference on*, 2004, pp. 187 – 198.

[23] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca, "GEO-RBAC: A spatially aware RBAC," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 1, Feb. 2007. [Online]. Available: http://doi.acm.org/10.1145/1210263.1210265

[24] M. Bell and V. Lovich, "Apparatus and methods for enforcement of policies upon a wireless device," US. Patent 8254902, Aug. 2012.

[25] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924971

[26] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry, "Practical and lightweight domain isolation on android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, ser. SPSM '11. New York, NY, USA: ACM, 2011, pp. 51–62. [Online]. Available: http://doi.acm.org/10.1145/2046614.2046624

[27] M. Ongtang, S. Mclaughlin, W. Enck, and P. Mcdaniel, "Semantically rich application-centric security in android," in *In ACSAC 09: Annual Computer Security Applications Conference*, 2009.

[28] M. Nauman, S. Khan, and X. Zhang, "Apex: extending android permission model and enforcement with user-defined runtime constraints," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '10. New York, NY, USA: ACM, 2010, pp. 328–332. [Online]. Available: http://doi.acm.org/10.1145/1755688.1755732

[29] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley, "Casa: context-aware scalable authentication," in *Proceedings of the Ninth Symposium on Usable Privacy and Security*, ser. SOUPS '13. New York, NY, USA: ACM, 2013, pp. 3:1–3:10. [Online]. Available: http://doi.acm.org/10.1145/2501604.2501607

[30] G. Danezis, "Inferring privacy policies for social networking services," in *AISEC Workshop at ACM CCS*, 2009, pp. 5–10.

[31] N. Sadeh, J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker, and J. Rao, "Understanding and capturing people's privacy policies in a mobile social networking application," *Personal and Ubiquitous Computing*, vol. 13, pp. 401–412, 2009. [Online]. Available: http://dx.doi.org/10.1007/s00779-008-0214-3

[32] P. G. Kelley, P. H. Drielsma, N. M. Sadeh, and L. F. Cranor, "User-controllable learning of security and privacy policies," in *ACM CCS*, 2008, pp. 11–18.

[33] G. Bai, L. Gu, T. Feng, Y. Guo, and X. Chen, "Context-aware usage control for android," in *Security and Privacy in Communication Networks*. Springer, 2010, pp. 326–343.

[34] R. Sandhu and J. Park, "Usage control: A vision for next generation access control," in *Computer Network Security*. Springer, 2003, pp. 17–31.

[35] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello, "Extracting places from traces of locations," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 3, pp. 58–68, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1094549.1094558

[36] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with gps history data," in *WWW*, M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, Eds. ACM, 2010, pp. 1029–1038.

[37] R. Montoliu, J. Blom, and D. Gatica-Perez, "Discovering places of interest in everyday life from smartphone data," *Multimedia Tools Appl.*, vol. 62, no. 1, pp. 179–207, 2013.

APPENDIX A
SURVEY QUESTIONNAIRE

The purpose of this questionnaire is to study the perceptions of smartphone users about risks of misuse related to their smartphones. We are especially interested in these risks from the perspective of surroundings factors of the smartphone.
We consider the following two main aspects:
1. *"place"* is a physical location where the smartphone is at a given instance.
2. *"situation"*: is a particular configuration of a place (in terms of the people and objects present in that place at a given instance)
We would like to ask you to consider two kinds of risks:
1. Physical misuse: Somebody steals your smartphone or somebody uses it without your permission.
2. Improper data capture by an application: An application that is installed on your smartphone collects information about your place or situation and sends it to an external party without your permission or knowledge. This external party can thereby gain potentially sensitive information about you and the places you visit.
When answering the questionnaire, please consider following definitions:
- Context information: is any information that your smartphone can collect using its sensors. These include, e.g., your location, the Bluetooth devices around you, the WiFi access points around you, camera image snapshots, sound samples from your smartphones microphone, readings of the movement sensor (the accelerometer) of your smartphone, etc.
We will provide further explanations during the questionnaire.

(Explicit instructions were included for most questions - questions about basic demographic features of the respondents are excluded here for brevity.)

1) Which places you visit through your daily routine do you expect to be "safe"?
2) Which places you visit through your daily routine do you expect to be "unsafe"?
3) Do you experience your workplace as a "safe" place?
4) Please explain why your workplace is "safe" or "unsafe".
5) Does your experience of places or situations as "safe" depend on people surrounding you?
6) Please give examples of when your experience of places or situations as "safe" depends on peole surrounding you or tell us why people have no influence on your perception of "safe".
7) Please give examples of when your experience of places or situations as "unsafe" depends on peole surrounding you or tell us why people have no influence on your perception of "unsafe".
8) Do you experience a place as "safe" when there are friends or people you know well surrounding you?
9) Do you experience a place as "unsafe" when there are strangers surrounding you?
10) Does your feeling of a place or situation being "safe" or "unsafe" change during different times of day?
11) Please give examples of when you think the time of day affects your feeling of "safe" and "unsafe" or why time of day has no effect on your perception.
12) How well do you think you can asses the risk of your mobile device being stolen or misused?
13) Do you think that the risk of your mobile device being stolen or misused might change depending on where you are?
14) Is there any tool or app installed on your phone that protects your smartphone and data in the case your mobile device gets stolen or misused?
15) If you have a tool or app installed, please let us know

TABLE IV: Overview of Parameters used in the Context Model

Name	Description	Example value
r_{sp}	max radius for GPS observations within a Stay Point	100 meters
t_min_{sp}	min duration of visit to an area for it to be considered a Stay Point	10 minutes
t_gap_{sp}	max length of gap in GPS observations allowed in a Stay Point	5 minutes
gps_{max}	max width and length of a GPS-based CoI	100 meters
f_min_{coi}	min number of visits to a place for it to be considered a CoI	5
t_min_{coi}	min total time spent in a place for it to be considered a CoI	30 minutes
t_max_{wifi}	WiFi observations within this time window belong to a single WiFi snapshot	10 seconds
ϵ_V	max length of gap between CoI observations allowed within a visit V	30 minutes
f_min_{famcoi}	min number of visits to CoI for it to be considered familiar	5
t_min_{famcoi}	min total time of visits to CoI for it to be considered familiar	60 minutes
f_min_{famdev}	min number of encounters for a device for it to be considered a familiar device	5
t_min_{famdev}	min total time of encounters for a device to be considered a familiar device	30 minutes
ϵ_E	max length of gap between device observations allowed within an encounter E	5 minutes
t_{scan}	Scanning interval of context scans	60 seconds
d_max_{unfam}	max amount of unfamiliar devices in familiar context still considered safe	0

which one(s). If you have no app installed, please explain why you havent installed one.

16) Which places you visit through your daily routine do you expect to be “sensitive”?
17) Which places you visit through your daily routine do you expect to be “public”?
18) Do you experience your workplace as a “sensitive” place?
19) Please explain why your workplace is “sensitive” or not.
20) Does your experience of places or situations as “sensitive” depend on people surrounding you?
21) Please give examples of when you think “sensitivity” depends on peole or why it does not depend on people.
22) Does your experience of places or situations as “public” depend on people surrounding you?
23) Please give examples of when you think a place is “public” because of the presence of people surrounding you or tell us why people have no influence on your perception of “public”.
24) Do you experience a place as “sensitive” when there are friends or people you know well surrounding you?
25) Do you experience a place as “public” when there are strangers surrounding you?
26) Does the time of day affect your feeling of “sensitive” and “public”?
27) How well do you think you can asses the “sensitivity” of a place or situation?
28) Do you believe that “sensitivity” might change depending on where you are?