ConXsense – Automated Context Classification for Context-Aware Access Control

Markus Miettinen
Technische Universität Darmstadt
markus.miettinen@trust.cased.de

Stephan Heuser
Technische Universität Darmstadt
stephan.heuser@trust.cased.de

Wiebke Kronz
Technische Universität Darmstadt
wiebke.kronz@cased.de

Ahmad-Reza Sadeghi
Technische Universität Darmstadt
ahmad.sadeghi@trust.cased.de

N. Asokan
Aalto University and University of Helsinki
asokan@acm.org

ABSTRACT

We present *ConXsense*, the first framework for context-aware access control on mobile devices based on context classification. Previous context-aware access control systems often require users to laboriously specify detailed policies or they rely on pre-defined policies not adequately reflecting the true preferences of users. We present the design and implementation of a context-aware framework that uses a probabilistic approach to overcome these deficiencies. The framework utilizes context sensing and machine learning to automatically classify contexts according to their security and privacy-related properties. We apply the framework to two important smartphone-related use cases: protection against device misuse using a dynamic device lock and protection against sensory malware. We ground our analysis on a sociological survey examining the perceptions and concerns of users related to contextual smartphone security and analyze the effectiveness of our approach with real-world context data. We also demonstrate the integration of our framework with the *FlaskDroid* [7] architecture for fine-grained access control enforcement on the Android platform.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, Invasive software*

Keywords

Mobile security; Context sensing; Privacy policies; Context-awareness

1. INTRODUCTION

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

The improved sensing capabilities of modern smart devices also provide an attractive attack surface against the contextual privacy of users, as the recent development of sensory malware shows: malicious code, typically a Trojan Horse appearing to be a legitimate app, uses the sensors of the device to extract sensitive information from the surroundings of the user. Context-aware access control can be used to encounter this threat by limiting the access of untrusted 3$^{\text{rd}}$-party applications to context information.

Various context-aware access control mechanisms and systems have been proposed. Some of these works are based on modifications of the RBAC model [12, 13] in which context-awareness is realized through roles that are triggered based on context parameters. Other approaches use explicit policies conditioned on contextual parameters [33, 2, 11, 5] or rules for expressing higher-level contextual preferences [21].

All of these works are based on *user-defined* or *pre-defined* policies. User-defined policies are very laborious to set up and maintain. This can be encountered with pre-defined policies provided by system administrators or app vendors, but these are inaccurate and inflexible: they cannot adequately capture and adapt to the highly personal and dynamic nature of individual users' contexts and privacy preferences. In our work, we overcome these deficiencies by using automatic context classification as a basis for access control decisions instead of static access control policies.

Furthermore, improved sensing capabilities also provide better possibilities to encounter threats arising from the context, like the threat of physical device misuse. Current static device locking methods severely deteriorate device usability with unnecessary password prompts in low-risk contexts, causing many users to leave their device unprotected [38]. Some earlier works attempted to improve device locking by using context information to probabilistically determine a level of confidence in the user's authenticity [31, 19]. We take a different approach: we do not try to authenticate users, but adjust device locking criteria according to the perceived security risk level of a context.

ASIA CCS'14, June 3–6, 2014, Kyoto, Japan.

http://dx.doi.org/10.1145/2590296.2590337.

To ground our work we use an interdisciplinary approach in which a sociological study is used to identify the concerns and perceptions of smartphone users in different contexts.

Contributions. Our contributions are as follows:

- We introduce *ConXsense*, the first context-aware access control framework for smartphones that uses **context profiling** and **automatic, adaptive and personalized context classification** for making context-aware access control decisions.
- We apply our framework to two important smartphone use cases: **protecting against device misuse**, and **defending against sensory malware**. ConXsense, however, is also applicable to a wide range of other security and privacy-related use cases.
- We apply **context profiling** and **machine learning** techniques on real-world data collected in a user study and evaluate the efficiency of automatic context classification in addressing the aforementioned threats.
- We integrate ConXsense with the *FlaskDroid* [7] architecture for fine-grained access control enforcement on the Android platform in order to realize the first adaptive, personalized and context-aware access control system of its kind for mobile devices.

The rest of this paper is structured as follows. A description of the problem and the ConXsense framework is given in Sections 2 and 3. In Section 4 we introduce the results of a sociological study on users' concerns and perceptions related to smartphones and present use cases addressing these concerns in Section 5. The design and implementation of our context model are described in Sections 6 and 7. The results of a user study evaluating the performance of context classification are presented in Section 8, and Section 9 shows its integration with the *FlaskDroid* architecture. After summarizing related work in Section 10, we conclude with outlines of future work in Section 11.

2. PROBLEM DESCRIPTION

While the idea of context-aware access control is not new [12, 21, 13, 11], currently proposed solutions mostly rely on policies specifying access control rules conditioned on values of contextual parameters.

User-specified policies have the potential to correctly reflect the user's true security and privacy preferences, but the amount of work required to set up and maintain a comprehensive set of context-dependent policies is high. Average users of mobile devices are hardly willing to spend significant amounts of time maintaining their policy set. In addition, it is questionable, whether regular users are capable to fully understand the implications of the policy settings they define. A study concerning location sharing policies [33] showed that the initial accuracy of location disclosure rules specified by users was only 59% and improved to 65% after users modified the rules based on a review of concrete enforcement decisions resulting from these rules. A recent study on the users' willingness to share their data with prominent single-sign on services [3] showed that the *majority* of users did not correctly understand the sharing implications of a sign-on dialog directly presented to them. These results suggest that the users' practical ability to control their security and privacy settings is limited.

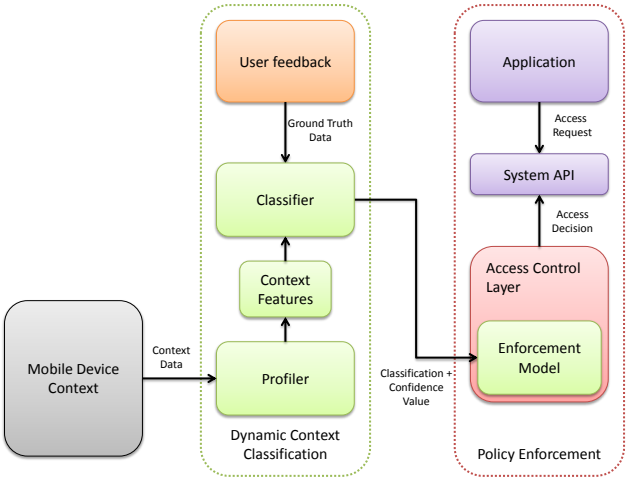


Figure 1: Context-based access control enforcement in ConXsense

An approach to tackle this problem of complexity and required user effort is to simplify things for users by predefined policies provided by administrators or app vendors. While this effectively reduces the user's burden, it fails to address the individual needs and privacy preferences of users. Pre-defined policies are by necessity only generalizations and not capable of accurately capturing concrete contexts and situations in individual users' lives. Also the fact that the privacy preferences of users may vary significantly from person to person can be very difficult to address using predefined policies.

It seems also unlikely that the privacy and security implications of pre-defined policies would be fully understood by regular users. This may lead to undesired situations: users under- or overestimating the level of privacy and security protection that the pre-defined policy provides to them.

User- or pre-defined policies alone are not sufficient to capture the highly dynamic and highly personal nature of a user's context (i.e., the ambient environment the device finds itself in) due to the problems outlined above. We aim at encountering these deficiencies by designing a context-aware access control framework that captures and adapts to the user's perception of the context and performs automatic context classification for making fine-grained context-aware access control decisions without the need for users to explicitly define contextual constraints. In the following section we describe the ConXsense framework in more detail.

3. FRAMEWORK DESCRIPTION

The ConXsense framework provides context-aware access control decisions by performing automatic classification of the context with regard to its security-relevant properties. The classification is based on machine learning models and user feedback providing ground truth information for training these models. Figure 1 shows a high-level overview of the framework.

The framework architecture is driven by context data observed with the sensors of the mobile device. The data are fed to a Profiler to calculate features describing the context. The Profiler implements a context model and aggregates profiles for relevant objects (e.g., significant places of the user

or devices the user encounters) in the model. The Profiler evaluates incoming observations based on the profiles and the context model, and periodically calculates feature vectors characterizing the current context of the user.

Apart from sensed context data, the framework obtains input through user feedback. This feedback can be derived from explicit user interaction (e.g., feedback given through the device's UI) or by monitoring the user's actions.

The Classifier uses the context feature vectors and user feedback events to train and update the context classification models. Once the models have been trained, they are used to classify new observations with regard to the context's security and privacy-relevant properties. The classification estimate of the Classifier and its associated confidence value are forwarded to the Access Control Layer, which takes them into account when making access control decisions. The framework can be used to address any use case for which contextual factors play a role, and it can accommodate any sensors, resources, functionalities, communication links or other data objects for which the Access Control Layer provides enforcement support.

In the following, we instantiate the ConXsense framework for protecting the privacy of the user. We consider two central use cases we identified and demonstrate the applicability and effectiveness of our approach by evaluating it with real-world contextual data obtained from a user study.

4. USER SURVEY

To investigate smartphone users' perceptions and concerns with regard to their smartphones in different contexts, we conducted a user survey following a *Mixed Methods* [40] approach commonly applied in sociological studies. We designed a statistical-quantitative survey [41] using quantitative questions to identify facts by statistical analysis [1] combined with open-ended, qualitative questions for investigating the underlying reasons for the users' perceptions.

The survey was answered by 122 participants aged 18-56 including people from different household types and representing different organizational positions. The participants were recruited using word-of-mouth, electronic communications like e-mail and social networks, targeting particularly smartphone users, to obtain a cross-sectional sample of different age groups and backgrounds. Thus, the set of respondents in our sample is representative of the target group of our framework, namely active smartphone users.

The survey contained questions on which contexts and contextual factors users deem relevant for their perceptions of contextual privacy and security. The answers to the open-ended questions confirmed our initial assumptions that two major context-related concerns dominate the users' minds. Firstly, people are concerned about *device misuse*, i.e., that their device is stolen and/or misused without their knowledge. The second concern relates to *privacy exposure*: users fear that private or confidential context information related to their life is revealed to unauthorized parties.

In some earlier studies on contextual behavior patterns of mobile device users, two central contexts have been used: "home" and "work" [43]. Also lifetime studies in social sciences suggest that these contexts are the most important contexts in average users' lives [37]. Therefore, we included dedicated questions about these context types as prototypes of very familiar contexts. Tables 1 and 2 show the results.

Table 1: Perceptions of Home vs. Work

	Privacy exposure		Risk of misuse	
Context	high	low	low	high
Home	46%	17%	94%	4%
Work	42%	21%	55%	40%

Table 2: Influence of people on the perceived privacy exposure and risk of misuse in the context

Question	Yes	No
Low risk of misuse depends on people	66%	14%
High risk of misuse depends on people	68%	11%
High privacy exposure depends on people	39%	43%
Low privacy exposure depends on people	36%	42%

4.1 Survey Results

Risk of device misuse. As can be seen in Table 1, the majority of people (94% and 55%, respectively) perceive "home" and "work" as having a low risk of misuse. This is explained by answers to open-ended questions, like "(...) places like home or office tend to remain secure regardless of time of day", supporting the intuition that familiar places are commonly perceived as having a low risk of misuse. While it is not surprising that "home" is perceived as safe, perceptions of "work" are more diverse [10], as reflected in our survey results. A significant fraction of people perceive "work" as a context with high risk of misuse. The reasons for this are indicated in several answers to the open-ended questions, e.g., "At work there are people around that I don't know and I don't trust them." This suggests that the familiarity of the context alone is not a sufficient indicator for estimating the risk level, also the people in context play an important role.

From Table 2 we can see that the people present in the context clearly affect the perceived risk of device misuse in that context. A clear majority of respondents stated that a low or high risk of device misuse is dependent on the persons present. From the responses to open-ended questions, we identified that the feeling of low risk is particularly caused by people that are *familiar* to the user (e.g., "I trust my friends and colleagues"). Similarly, the presence of *unfamiliar* persons in the context was indicated as a reason for perceiving the context as having a high risk of device misuse (e.g., "Where there are people around me that I don't know, I don't feel secure", or "Unknown people represent threat").

This underlines the fact that location information alone is not a sufficient basis for access control decisions. Also other context information affecting the perceptions of security need to be taken into account.

Privacy exposure. The data in Table 1 suggest that a significant fraction of respondents feel that a familiar context also has high privacy exposure, i.e., contains private or confidential context information related to the user.

Table 2 shows, however, that the perception of privacy exposure does not appear to be affected by the presence of people. More respondents believe that the people present in the context do not contribute to the level of privacy exposure of the context.

Conclusions and Discussion. Based on the analysis of the survey results above, we see that two main factors affect users' perceptions on privacy exposure and risk of misuse in contexts: the familiarity of the context itself and the familiarity of persons present in the context. Therefore, we need

to design our context model in a way that we can (1) identify relevant contexts and model their familiarity, and, (2) track encounters with other persons and measure their familiarity by observing their mobile devices. In Section 6 we construct such a context model.

The analysis presented above supports our *common* understanding and shows which factors typically influence the perception of contexts. However, exceptions exist, like, e.g., those 4% who consider home a high-risk environment. Reasons for this can be various. Answers to open-ended questions suggest that even in familiar contexts the perception of risk of misuse changes when untrusted people appear, or, it is caused by what we call the "Toddler Scenario", i.e., the influence of familiar people (e.g., a young child, or spouse) considered as "clueless" or "honest but curious", causing a person to consider the risk of device misuse significant also in familiar surroundings. To investigate these special cases, more detailed questions on familiar contexts would need to be included. However, sociological literature suggests that questions about such contexts are perceived as intrusive and will therefore not be answered [4, 10]. This concern was also reflected in some of the feedback we received about the online questionnaire. Hence, and because exceptional cases seem to be a marginal phenomenon, we focus on the common cases and address more exceptional cases in subsequent studies.

5. USE CASES

Based on the user survey, we focus on the most prominent concerns expressed by the users: fear of device misuse and disclosure of private or confidential context information. To mitigate these concerns, we identified following use cases.

5.1 Misuse Protection: Usable Device Lock

Several surveys [9, 38] point out that many mobile users do not use device locks (also known as idle screen locks) to protect their phones even though that would effectively protect against device misuse. One reason for this is that screen locks and other similar access control techniques on mobile devices today are both too inflexible and hard to use. A solution could be to make the locking mechanism more usable, so that users would be more willing to use device locking. The approach taken by Gupta et al. [17] was to use context data to adapt the locking time out of the device lock in different contexts. We adopt this approach and want to use the estimated *risk of device misuse* in a context as a means to decide, whether and how fast to lock the device in case it is not used.

Adversary model. For this use 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).

Goal. We want to protect the applications and data on the device from potential threats in the context by limiting the potential damage arising from someone physically accessing the device without the user's approval. Therefore, we want to minimize the chances that an unauthorized person in the context has access to the user's data. We do this by configuring the device lock dynamically based on the risk of device misuse in the context, while trying to strike a balance between maximizing protection on one hand and minimizing user annoyance of having to unlock the device in low-risk contexts on the other hand.

5.2 Resisting Sensory Malware

Sensory malware is an emerging class of malicious applications (typically Trojans) that use the context sensors of a mobile device to collect potentially sensitive information from the user's context. Prominent examples of sensory malware are *Stealthy Video Capturer* [44] (video via camera), *(sp)iPhone* [25] (keystrokes via accelerometer) *Soundcomber* [36] (spoken secrets via microphone), or the recent *PlaceRaider* [42] Trojan (3D models via camera). Users may also have granted sensor access privileges to benign apps which use them too intrusively: for instance an augmented reality app may take pictures of surroundings even when the user is not actively using augmented reality, as a means of enriching the app vendor's data collection.

Adversary Model. For this use case, the adversary is an app already installed on the device. We assume that the application has been granted the necessary privileges during installation and has therefore access to the contextual sensors on the mobile device, but cannot circumvent the access control system[1]. The application may be a Trojan Horse (e.g., sensory malware) or a benign but somewhat intrusive application.

Goal. We aim at protecting sensitive information in the context of the device from the adversary. We do this by preventing or limiting the ability of the adversary to gather information from contexts with high *privacy exposure*, i.e., contexts that contain information that the user would want to protect from the adversary. Such information can be either *private*, i.e., information about the user herself, or, *confidential*, i.e., other sensitive information not directly related to the user. The user's home (private) and workplace (confidential) are examples of typical contexts with high privacy exposure.

6. CONTEXT MODEL

In this section, we present a context model used to extract context features reflecting the familiarity of contexts and the persons in the context. The context features are input for the Classifier and used for classifying contexts as having high or low privacy exposure and/or risk of misuse. The context model is based on two main concepts: *Contexts of Interest (CoI)* for modelling the familiarity of contexts and *Bluetooth devices* for modelling familiar and unfamiliar people in context.

6.1 Detection of Contexts of Interest (CoIs)

For our purpose, *Contexts-of-Interest* (CoIs) correspond to locations that a user often visits and/or spends a significant amount of time in, e.g., home, workplace, grocery store, etc. 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 GPS can't be used but coverage of WiFi access points typically is available. By combining both types of CoIs, we can identify and detect most significant places that users typically visit.

[1]Malware that uses operating system (root) exploits to circumvent the enforcement of the context-aware access control system is outside the scope of this paper.

6.1.1 GPS-based CoIs

To identify GPS-based CoIs, we adopt the notions of *stay points* and *stay regions* as introduced by Zheng et al. [46] and developed further by Montoliu et al. [26]. The identification of GPS-based CoIs is based on position observations obtained via GPS.

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 $r_{sp} =$ 100 m 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 $t\_min_{sp} = 10$ min and not to contain observation gaps longer than $t\_gap_{sp} = 5$ min.

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 $gps_{max} = 100$ m width and length. An area is a *GPS-based Context-of-Interest*, if (i) the user has visited the area more than $f\_min_{coi} = 5$ times and (ii) has spent longer than $t\_min_{coi} = 30$ min in total in the area.

Example. As an illustrative running example, let us consider a user who regularly commutes between her workplace and home. Other places she regularly visits are a grocery store and a public sports facility. She usually carries her smartphone with her, which continuously senses her GPS location and other context data.

When the user goes to the grocery store and stays there for 15 minutes, i.e., longer than $t\_min_{sp} = 10$ min and moves only within a radius of $r_{sp} = 100$ m, a stay point sp of duration $\mathsf{dur}(sp) = 15$ min 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.

If our user visits the grocery store 10 times and stays each time for 15 minutes, ten stay points will be generated. These average positions will be aggregated into a GPS-based CoI C, because their total stay duration of 2 hours and 30 minutes is longer than the required $t\_min_{coi} = 30$ min 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 $pos_{\bar{sp}}$.

6.1.2 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. 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 $t\_max_{wifi} = 10$ sec.

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}.

These criteria for WiFi stay points were selected, because it is not uncommon that WiFi access points are missed in scans [14]. 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.

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

Example. 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 C for the user's workplace.

6.2 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.

6.2.1 Location context

The location context of the user is defined in terms of the CoIs that the user visits at a specific point in time.

DEFINITION 1 (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*

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

$\forall i, 1 < i \leq n : t(wifi_i) - t(wifi_{i-1}) < \epsilon_V$. *We denote the set of all visits V_C of the user to CoI C with $\mathcal{V}_C$.*

DEFINITION 2 (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.*

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.

6.2.2 Social context

In order to capture *people* in the user's surroundings, we observe their mobile devices that can be sensed through proximity sensing technologies like Bluetooth (BT). Bluetooth has a range of approximately 30 meters given a direct line of sight, so its physical properties quite well reflect our notion of a context comprising the space immediately observable by the user (e.g., a room). Bluetooth has been commonly used in ubiquitous computing literature to model the presence of persons in a perimeter (cf., e.g., [27]). 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.

Some users may not keep Bluetooth enabled and discoverable on their devices or always carry their devices with them. Therefore, we will not always be able to reliably detect the presence of all persons in the context using Bluetooth alone. However, this is not necessary, since our probabilistic framework utilizes Bluetooth as one factor for identifying the type of context the user is in and not as a 'tripwire' for detecting potentially malicious users. Especially in public contexts where many persons are present and the likelihood that at least some Bluetooth devices can be detected is high, Bluetooth works well as a context classification factor. In addition, familiar, known devices (e.g. devices of family and friends) can be polled even if they are in hidden mode, if the BDADDR of the target device is known. For example, two devices that are paired can detect each other this way even if they remain invisible to other devices.

The social context is defined in terms of the devices that are detected in the user's context at a specific point in time.

DEFINITION 3 (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 this size is to be able to handle missed device observations not uncommon with Bluetooth sensing.

DEFINITION 4 (DEVICE CONTEXT). *The* device context D_t *at timestamp t is the set of devices d that are encountered during that point of time.*

DEFINITION 5 (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}.*

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.

6.3 Context Profiles

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

6.4 Context Features

Based on the context model, we define following features:

Context familiarity features.

Let $\mathcal{C}^{\text{GPS}}$ denote the subset of all GPS-based CoIs and $\mathcal{C}^{\text{WiFi}}$ the subset of all Wifi-based CoIs in $\mathcal{C}$. Then we have:

$f^{\text{GPS}}_{\max_{\text{dur}}}$: maximum total visit time of any GPS-based CoI in current location context

$$f^{\text{GPS}}_{\max_{\text{dur}}}(t) = \begin{cases} \max_{C \in \{\mathcal{C}^{\text{GPS}} \cap L_t\}} dur_C, & L_t \cap \mathcal{C}^{\text{GPS}} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

$f^{\text{GPS}}_{\max_{\text{dur}}}$: number of visits to the GPS-based CoI with the maximum total visit time

$$f^{\text{GPS}}_{\max_{\text{dur}}}(t) = \begin{cases} visits_{C_i}, & i = \arg\max_{C \in \{\mathcal{C}^{\text{GPS}} \cap L_t\}} dur_C \wedge \\ & L_t \cap \mathcal{C}^{\text{GPS}} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

$f^{\text{WiFi}}_{\max_{\text{dur}}}$: maximum visit time of any WiFi-based CoI in the location context

$$f^{\text{WiFi}}_{\max_{\text{dur}}}(t) = \begin{cases} \max_{C \in \{\mathcal{C}^{\text{WiFi}} \cap L_t\}} dur_C, & L_t \cap \mathcal{C}^{\text{WiFi}} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

$f^{\text{WiFi}}_{\max_{\text{freq}}}$: number of visits to the WiFi-based CoI with the maximum total visit time

$$f^{\text{WiFi}}_{\max_{\text{freq}}}(t) = \begin{cases} visits_{C_i}, & i = \arg\max_{C \in \{\mathcal{C}^{\text{WiFi}} \cap L_t\}} dur_C \wedge \\ & L_t \cap \mathcal{C}^{\text{WiFi}} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

Device familiarity features.

$f^{\text{BT}}_{\text{num}}$: Number of Bluetooth devices and familiar Bluetooth devices in device context D_t

$$f^{\text{BT}}_{\text{num}}(t) = |D_t|, \;\; f^{\text{BT}}_{\text{fam}}(t) = |D_t \cap \mathcal{D}_{fam}|$$

$f^{\text{BT}}_{\text{fam}_{\text{avg-time}}}$: Average encounter time of familiar devices in D_t

$$f^{\text{BT}}_{\text{fam}_{\text{avg-time}}}(t) = \begin{cases} \frac{\sum_{d \in \{D_t \cap \mathcal{D}_{fam}\}} dur_d}{|D_t \cap \mathcal{D}_{fam}|}, & D_t \cap \mathcal{D}_{fam} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

$f^{\text{BT}}_{\text{fam}_{\text{avg-freq}}}$: Average number of encounters of familiar devices in D_t

$$f^{\text{BT}}_{\text{fam}_{\text{avg-freq}}}(t) = \begin{cases} \frac{\sum_{d \in \{D_t \cap \mathcal{D}_{fam}\}} enc_d}{|D_t \cap \mathcal{D}_{fam}|}, & D_t \cap \mathcal{D}_{fam}| \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

The Profiler calculates context feature values based on a history of observation data and labels them based on user feedback. The feature values are used by the Classifier to train machine learning-based classifiers for classifying new observations. In the following sections, we show how we applied this context model on real-world context data to evaluate the effectiveness of the model and the Classifier.

7. IMPLEMENTATION

To evaluate the ConXsense framework, we created a prototype implementation consisting of a Data Collector app, a Profiler and Classifier. The output of the Classifier was integrated with the Access Control Layer (cf. section 9) to provide enforcement.

7.1 Data Collector

For collecting context data, we implemented a Data Collector app for Android. It 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 collected data for reaching a battery lifetime of at least a working day (12h) on, e.g., the Samsung Galaxy Nexus and Nexus S devices. The collected data comprise location information, nearby Bluetooth devices and WiFi access points, acceleration sensor information as well as information about user presence and her interaction with apps (Activities).[3]

The Data Collector app also collects *ground truth data*. The user regularly reports the perceived risk of device misuse in the current context by specifying the context to be "safe" (low risk of misuse) or "unsafe" (high risk of misuse). In addition, users are asked to classify the current context as "home" or "work", if the context has high privacy exposure due to context information being either *private* or *confidential*, respectively, or, "public" if the context has low privacy exposure. By using concise words that are easy to follow and relate to helped us in keeping the user interaction as simple as possible. To avoid misunderstandings, an introductory explanation was given to study participants beforehand.

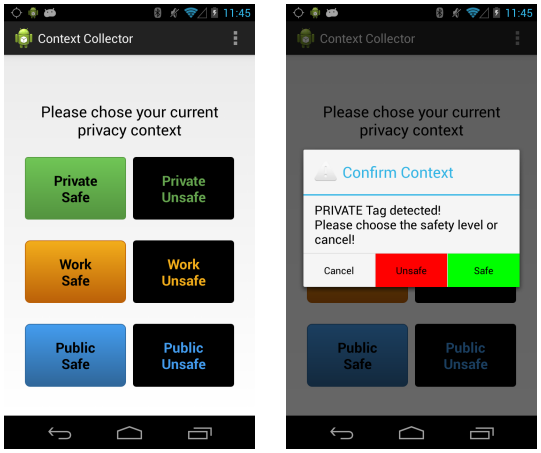
The users provided the above feedback for the current context either by using context feedback buttons on the device's UI or by using dedicated NFC tags provided to the participants for triggering context reporting (cf. Figure 2). The feedback UI was either spontaneously invoked by the user, or, if no ground truth had been provided during the last two hours, the app reminded the user to do so via sound, vibration and flashing LED notifications. Context and ground truth data were stored in a SQLite database and periodically uploaded to a server via HTTPS.

7.2 Profiler and Classifier

We implemented the functionality of the Profiler 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. Scripts were also used for extracting the context feature vectors.

The functionality of the Classifier was realized and evaluated using the Weka data mining suite [18] and its provided algorithm implementations for k-NN, Random Forest and Naïve Bayes classifiers.

[3]Data Collector is a generic solution collecting more data than required by the current Profiler.



(a) Feedback using Context Feedback Buttons

(b) Feedback using Context NFC Tags.

Figure 2: Android Data Collector App

8. EVALUATION

To evaluate the context classification, the Data Collector app was installed on the Android smartphones of 15 test users having technical and non-technical backgrounds. A test user group of this size is large enough for verifying the validity of the concept and is in line with previous works evaluating context-aware access control by Riva et al. [31] ($n = 9$) and Sadeh et al. [33] ($n = 12$ and $n = 19$). Users provided context and feedback data over a period of 68 days, 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.

From the collected data, the Profiler calculated personal context profiles and context features. The features were used by the Classifier to train classification models for predicting the privacy exposure and misuse risk levels of contexts. The context labels obtained through user feedback were used to attach class labels to the context feature vectors.

Each test user provided at least 50 or more feedback labels. This would roughly correspond to the user providing 2-3 feedbacks per day over a period of three weeks, which seems like a manageable burden on the user. After this initial training period of the Classifier, the need for explicit user feedback would significantly diminish. The user would need to provide only occasional corrective feedback in cases of incorrect predictions of the Classifier.

In constructing the Classifier, we experimented with three different machine learning algorithms: 1) A k-nearest neighbors (kNN) classifier, which 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. 2) A Naïve Bayes (NB) classifier is a simple probabilistic classifier which has been successfully used, e.g., in spam e-mail detection [34]. 3) Random Forest (RF) 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. For each participant, we trained the Classifier using the labeled context feature vectors and evaluated the performance of the classifiers using 10-fold cross-validation.

We assume that by default restrictive protection measures are in place (access to sensors disabled, device lock active). The Classifier's task is therefore to predict situations, in which the protections could be relaxed, i.e., if the context has low privacy exposure or a low risk of device misuse.

Even though most accurate results would be obtained by direct measurement of on-line enforcement on users' mobile devices, we had to rely on an offline evaluation of the Classifiers performance, since we wanted to be able to experiment with several different machine learning algorithms. Implementing or porting several different algorithms on the mobile device and conducting a separate user study for each of them was not feasible given the resource limitations. Therefore we intend to evaluate the performance of on-line enforcement in a subsequent user study involving devices with enhanced device locking functionality.

Protecting against device misuse. Figure 3 shows the average receiver operating characteristic (ROC) curves of the classifiers for users who provided at least five feedback datapoints for each context class.

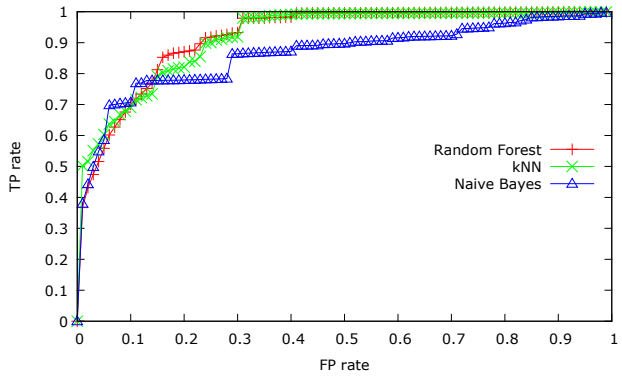


Figure 3: Average receiver operating characteristic (ROC) curves for classifying contexts with low risk of device misuse.

All classifiers perform reasonably well on the testing data, providing usable results for practical use. The classifiers reach a true positive rate of approximately 70% with a fairly moderate false positive rate of 10%. This would mean that by applying a relaxed device locking scheme in low-risk contexts, we can potentially reduce the amount of unnecessary authentication prompts shown to the user by 70%. Only one time in ten would a relaxed locking mechanism be enforced while the user is in a context with higher risk of misuse. This means that a thief or other unauthorized user would likely have a less than 10% chance of finding the device in an unlocked state, when obtaining physical access to it. These results clearly outperform the progressive authentication scheme presented by Riva et al. [31], who report a reduction of 42% in unnecessary authentication prompts presented to the user.

Protecting against sensory malware. Figure 4 shows the average performance of the classifiers in identifying contexts with low privacy exposure.

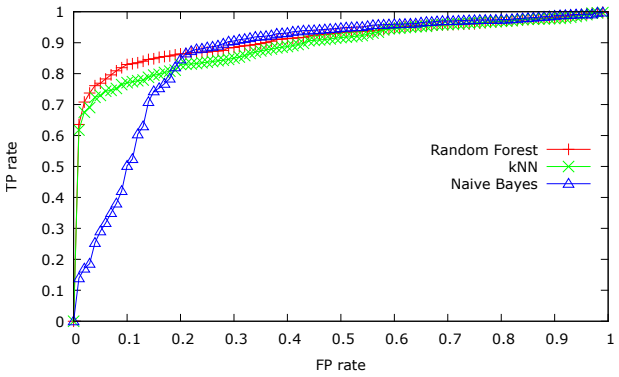


Figure 4: Average ROC curves for classifying contexts with low privacy exposure.

For this use case, the Random Forest and kNN classifiers provide best performance. They reach a true positive rate of 70% at a very low false positive rate of 2-3.5%. This would mean that if a sensory malware protection scheme with a 'default deny' policy is enforced, only in less than 3.5% of the cases would access control be relaxed in contexts with high privacy exposure. In practice, this would severely limit a sensory malware application's ability to extract useful sensitive information about the user.

Through the use of a default deny policy, our framework errs by default on the safe side, i.e., sensory malware is by default denied access to sensor information. The true positive rate of 70% means that our scheme is able to relax the access restrictions to sensors in public or low-privacy exposure contexts in 70% of the cases. The remaining 30% can be handled through manual overriding of the default policy by the user. Fortunately, the use of context information by many apps is often user-driven, i.e., sensor data are utilized, when the user is actively using the app (e.g., using a navigation app to locate a nearby restaurant). Adding an override confirmation dialog to the user interaction in such situations should therefore be easy, since the device already is in the focus of the user's attention. This approach also has the benefit that the overriding action can be used as additional ground truth data for updating the classification model and thus improving subsequent classification accuracy.

9. ENFORCEMENT

To verify the applicability of our framework to practical access control enforcement, we integrated it with an Access Control Layer for which we adopted and adapted the *FlaskDroid* [7] architecture, a fine-grained mandatory access control framework for Android 4.0.4 (cf. Figure 5). We now show how the combination of ConXsense and *FlaskDroid* can address the previously defined use-cases, namely *Resisting Sensory Malware* and *Usable Device Lock* (cf. Section 5). For our implementation we use a Samsung Galaxy Nexus smartphone.

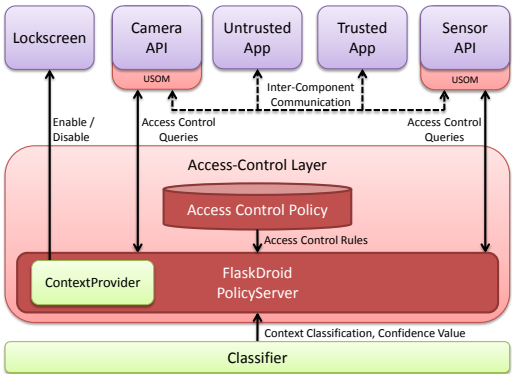


Figure 5: Enforcement of Context-based policies

9.1 Implementation

FlaskDroid extends *Security Enhanced Android (SEAndroid)* [39] with fine-grained type enforcement on Android's middleware layer. In *FlaskDroid*, Android components that provide access to sensitive resources, such as the SensorService which provides access to sensor information, act as User-Space Object Managers (USOMs) which control access to resources they manage. More specifically, USOMs control *operations* from *subjects* (i.e., apps) to *objects* (e.g., data) using *types* assigned to *subjects* and *objects*.

At boot time, *FlaskDroid's* PolicyServer (cf. Figure 5) parses an Access Control Policy and proceeds to assign app types (e.g., *trusted* or *untrusted*) to all installed apps based on application metadata (e.g., package name or developer signature). Apps installed by the user are assigned types during their installation. Whenever apps request access to a USOM, 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 SystemServer, for access control decisions. *FlaskDroid* supports conditional access control rules by means of ContextProviders that evaluate the current context and enable or disable rules at runtime.

To meet our goals we extended *FlaskDroid* with additional USOMs and implemented a ConXsense ContextProvider. It uses the context classification information and confidence values provided by the Classifier to activate or deactivate conditional rules at runtime (cf. Figure 5) and to influence the Lockscreen behaviour. The ContextProvider can be tuned with individual user-, use-case and sensor-specific thresholds for the expected confidence values. These thresholds could be set, e.g., by specifying a desired maximal false positive rate and adjusting the confidence threshold accordingly based on the observed historical performance of the context classifier. Access to more sensitive context sensors like GPS could require a higher prediction confidence than less sensitive sensors like the magnetometer.

Mitigation of Sensory Malware. To mitigate, respectively reduce the effects of sensory malware (e.g., *Placeraider* [42] or SoundComber [36]), 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. We transformed Android's CameraService into an USOM which filters queries to the *takePicture* and *startPreviewMode* methods. Furthermore, we used *FlaskDroid* to filter acceleration sensor events delivered to SensorEventListeners registered by apps. It should be noted that *FlaskDroid's* original implementation of the SensorManager USOM is insufficient to block sophisticated attacks, since the SensorManager is loaded into the memory space of (potentially malicious) apps. Thus, we replaced *FlaskDroid's* SensorManager USOM with a corresponding USOM in Android's SensorService, which is not under the control of apps.

Similarily, the combination of ConXsense and *FlaskDroid* can address also other variants of sensory malware, such as *Soundcomber* [36], by identifying the relevant Android APIs, instrumenting them as USOMs and extending the *FlaskDroid* policy with corresponding conditional rules.

Usable Device Lock. To allow for changes in the Android Lockscreen policy based on the current risk for device misuse, we use the ConXsense ContextProvider to configure Android's Lockscreen dynamically at runtime. We modified Android's Settings component to be notified by our ContextProvider about changes in the current risk for device misuse by means of a Broadcast Intent. We further modified Android's LockPatternKeyguardView which is used to display the Lockscreen to query the Settings component for context information. While the device is used in a context with *low* risk for device misuse, the LockPatternKeyguardView class automatically dismisses the Lockscreen. Whenever the device is rebooted or the risk for device misuse changes to *high*, a low-watermark mechanism ensures that the Lockscreen is always displayed regardless of the current risk for device misuse. This mechanism is required to prevent an attacker from bypassing the Lockscreen by changing the context, emulating a context the user considers to have low risk for device misuse or rebooting the device. In addition, to mitigate the effect of sensory malware which uses the acceleration sensor as a side channel to derive user credentials (e.g., Lockscreen PIN or password) [45, 30, 8], we use the SensorService USOM, our ContextProvider and corresponding conditional access control rules to block access to the acceleration sensor by 3rd-party apps while the Lockscreen is displayed.

9.2 Evaluation

Mitigation of Sensory Malware. To mitigate the effects of the *PlaceRaider* [42] sensory malware we designed a *FlaskDroid* policy to assign the type *trusted* to all preinstalled system apps (e.g., the camera app), and the type *untrusted* to all 3rd-party apps. In a real-world scenario this trust level could be derived from the app's reputation in an app market. We use conditional access control rules for the CameraService and SensorService USOMs to prevent all *untrusted* apps from accessing the acceleration sensor and the camera when the risk for privacy exposure is *high*.

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 risk for privacy exposure was *high*, 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 Con-

[4]The sample we received is incompatible with Android 4.0.4.

trol Layer, which is not surprising since it merely enforces context-dependent access control rules.

To evaluate the performance impact of the Access Control Layer we implemented an app which automatically triggers 10,000 access control queries by reading sensor data and taking pictures. On average, the Access Control Layer caused an overhead μ of 4.9 ms (standard deviation σ 17.6 ms) for the SensorService and CameraService USOMs on a Samsung Galaxy Nexus smartphone. The high standard deviation σ is caused by the garbage collector used in Android's Dalvik Virtual Machine: While studying Android's system logs we noticed that during the irregularly slow access control queries, which are responsible for the high standard deviation, the garbage collector started and caused a stall. Overall, 95% of all access control decisions are handled in less than 4.2 ms, which we consider reasonable.

Usable Device Lock. To test our implementation of the context-aware device lockscreen we modified the Android operating system to periodically wake the device from sleep and switch on the screen. We furthermore installed a synthetic malware, which registers SensorEventListeners in Android's SensorService to be notified of acceleration sensor readings. By logging and analyzing the Lockscreen behavior, context information and sensor readings we verified that the Lockscreen was only automatically dismissed in valid situations and that our synthetic malware did not receive any sensor readings while the Lockscreen was active.

10. RELATED WORK

In the digital society, context data have been extensively used to analyze numerous aspects of human everyday life. Examples range from the prediction of health status by interpreting context data [24] to analyzing ethnographics [20] or person matching based on similar interests [15]. Our framework brings this idea of contextual analysis to the area of security and privacy protection for the most important tool of modern life - the smartphone.

A number of works have approached the problem of context-aware access control. Contrary to our work, all of them rely on user-defined or pre-defined policies in the form of role definitions, conditions on context parameters, or context-dependent rules. For example, Covington et al. [12] use a *Generalised Role Based Access Control* (GRBAC) model utilizing Environment Roles that are activated and deactivated based on context observations, and Damiani et al. [13] utilize roles in their spatially-aware RBAC model using location as a component for access control decisions.

Others have used user- or pre-defined policies conditioned on context parameters. Examples include Sadeh et al. [33] who investigate a policy definition and management system for the *PeopleFinder* application and Kelley et al. [23], who introduced a user-controllable policy learning system that builds on incremental policy improvements proposed to the users based on recorded history events. For mobile devices, Bai et al. propose a solution for fine-grained usage control on Android [2]. Their work extends the UCON access control model [35] by using context information (e.g., location and time) as an additional input for policy decisions.

Hull et al. [21] present the *Houdini* framework for mitigating the complexity that value-based customization of policies implies by using user-provided higher-level preferences to generate rules for privacy enforcement. They mention the possibility for automatically-learned preferences, but do not provide support for such automation at the time of writing.

Many recent papers have addressed context-aware access control enforcement on mobile devices. For example, Conti et al. [11] describe the *CRePe* framework for Android for enforcement of context-dependent access control policies allowing or denying access to specific resources depending on the currently detected active context. In the *MOSES* framework [32] Rusello et al. propose a combination of dynamic taint tracking using the *TaintDroid* architecture [16] 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* [6] architecture provides lightweight security domain isolation on Android with basic support for context-based network access control policies. Saint [29] features a context-aware fine-grained access control framework for Android, which focuses on enabling app developers to define context-dependent runtime constraints on inter-app communication. Nauman et al. present *Apex* [28], which extends the Android operating 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). All of these works heavily rely on user- or pre-defined rules, whereas our work relies on dynamic context classification utilizing machine learning as a source for access control enforcement. Also, in contrast to *MOSES*, *TrustDroid* and *Apex*, our access control architecture is based on the more generic and flexible *FlaskDroid* platform [7], which is also able to cover (most of) the use cases described in *Saint*.

A recent patent application by Bell et al. [5] discloses a system using context-triggered policies controlling the access of applications to sensors and other resources on a smartphone. Also their approach relies on either pre-defined policies or policies uploaded to the devices by external entities.

Addressing the problem of more usable user authentication on mobile devices, Riva et al. [31] use various contextual clues to (partially) authenticate the user by estimating the likelihood that the user is in proximity and use this information to configure the device lock. Similarly, Hayashi et al. [19] introduce *Context-Aware Scalable Authentication*, an approach which uses the location of the device in a probabilistic framework to determine the active authentication factors to be used for user authentication (e.g., PIN or password) on smartphones. Although we cover a similar use case as these papers, our approach is very different. We do not authenticate the user, but rather adjust device locking behavior based on automatic classification of the context according to its perceived risk level.

Kang et al. [22] introduced the idea of time-based clustering of position observations, which Zheng et al. [46] used to introduce the concepts of stay points and stay regions, further developed by Montoliu et al. [26]. We adopt a slightly modified form of the notion of stay regions to define our GPS-based CoIs. In addition, we also extend the notion of a stay points to non-locational data in the form of WiFi stay points. Dousse et al. [14] have successfully demonstrated the use of WiFi fingerprints for identifying and detecting places based on WiFi. We adopt a simplified version of their place identification scheme considering only intersections of WiFi snapshots for our WiFi-based CoI detection.

Gupta et al. [17] were the first to use context profiling and the notion of CoI and device familiarity for estimating the 'safety' level of a context. Their system relied on a simple heuristic model based on time-discounted familiarity measures and suffered from having to specify a fixed threshold for distinguishing between context classes, which fails to take into account context- and user specific differences. Since we apply a sophisticated context model and more powerful machine learning models for context classification, our approach is capable to take better into account also context- and user-specific differences in perceptions of risk level and privacy exposure.

11. CONCLUSIONS AND FUTURE WORK

In this paper, we described ConXsense, a context-aware access control framework for mobile devices utilizing automated classification of contexts based on sensed context data. We applied it to two concrete smartphone-related use cases: defending against sensory malware and device misuse. We showed that context classification can be used for context-aware access control enforcement, effectively addressing true security concerns that smartphone users have. In this, however, we do not see the task merely as a prediction problem, but rather we consider true contextuality as a continuous process of learning from and adapting to the individual needs and preferences of smartphone users.

Having validated the effectiveness of ConXsense, the next step is to evaluate its usability. We plan to implement on-device versions of the Profiler and Classifier and create a mobile app for user studies focusing on the usability aspects related to our framework. We intend also to develop further richer context models incorporating more context sensors, and addressing other context-aware access control use cases.

12. REFERENCES

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

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

[3] L. Bauer, C. Bravo-Lillo, E. Fragkaki, and W. Melicher. A comparison of users' perceptions of and willingness to use google, facebook, and google+ single sign-on functionality. In *Workshop on digital identity management (DIM) in conjunction with the 20th AMC Conference on Computer and Communications Security (ACM CCS 2013)*, Berlin, Germany, Nov. 2013.

[4] U. Beck. *Risk Society: Towards a New Modernity*. Association with Theory, Culture & Society. SAGE Publications, 1992.

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

[6] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry. Practical and lightweight domain isolation on android. In *1st ACM workshop on Security and privacy in smartphones and mobile devices*, SPSM '11, pages 51–62, New York, NY, USA, 2011. ACM.

[7] 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.

[8] L. Cai and H. Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *6th USENIX Conference on Hot Topics in Security*, HotSec'11, pages 9–9, Berkeley, CA, USA, 2011. USENIX Association.

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

[10] S. Cohen and L. Taylor. *Escape Attempts: The Theory and Practice of Resistance in Everyday Life*. Taylor & Francis, 1992.

[11] M. Conti, B. Crispo, E. Fernandes, and Y. Zhauniarovich. CRêPE: A system for enforcing fine-grained context-related policies on android. *Information Forensics and Security, IEEE Transactions on*, 7(5):1426–1438, 2012.

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

[13] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A spatially aware RBAC. *ACM Trans. Inf. Syst. Secur.*, 10(1), Feb. 2007.

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

[15] N. Eagle and A. Pentland. Social serendipity: mobilizing social software. *Pervasive Computing, IEEE*, 4(2):28–34, 2005.

[16] 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 *9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[17] A. Gupta, M. Miettinen, N. Asokan, and M. Nagy. Intuitive security policy configuration in mobile devices using context profiling. In *International Conference on Privacy, Security, Risk and Trust (PASSAT), and 2012 International Confernece on Social Computing (SocialCom)*, pages 471–480. IEEE, Sept. 2012.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[19] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley. CASA: context-aware scalable authentication. In *Ninth Symposium on Usable Privacy and Security*, SOUPS '13, pages 3:1–3:10, New York, NY, USA, 2013. ACM.

[20] J. Höflich and M. Hartmann. *Mobile Communication in Everyday Life: Ethnographic Views, Observations and Reflections*. Frank & Timme, 2006.

[21] 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 *2004 IEEE International Conference on Mobile Data Management*, pages 187 – 198, 2004.

[22] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello. Extracting places from traces of locations. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(3):58–68, July 2005.

[23] P. G. Kelley, P. H. Drielsma, N. M. Sadeh, and L. F. Cranor. User-controllable learning of security and privacy policies. In *1st ACM Workshop on Workshop on AISec*, AISec '08, pages 11–18, New York, NY, USA, 2008. ACM.

[24] A. Madan, M. Cebrian, D. Lazer, and A. Pentland. Social sensing for epidemiological behavior change. In *12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10, pages 291–300, New York, NY, USA, 2010. ACM.

[25] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *18th ACM Conference on Computer and Communications Security*, CCS '11, pages 551–562, New York, NY, USA, 2011. ACM.

[26] R. Montoliu, J. Blom, and D. Gatica-Perez. Discovering places of interest in everyday life from smartphone data. *Multimedia Tools Appl.*, 62(1):179–207, 2013.

[27] F. Naini, O. Dousse, P. Thiran, and M. Vetterli. Population size estimation using a few individuals as agents. In *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 2499 – 2503, July 2011.

[28] M. Nauman, S. Khan, and X. Zhang. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 328–332, New York, NY, USA, 2010. ACM.

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

[30] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: Password inference using accelerometers on smartphones. In *Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 9:1–9:6, New York, NY, USA, 2012. ACM.

[31] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos. Progressive authentication: deciding when to authenticate on mobile phones. In *21st USENIX Security Symposium*, 2012.

[32] G. Russello, M. Conti, B. Crispo, and E. Fernandes. Moses: supporting operation modes on smartphones. In *17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 3–12, New York, NY, USA, 2012. ACM.

[33] 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*, 13:401–412, 2009.

[34] 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*, volume 62, pages 98–105, 1998.

[35] R. Sandhu and J. Park. Usage control: A vision for next generation access control. In V. Gorodetsky, L. Popyack, and V. Skormin, editors, *Computer Network Security*, volume 2776 of *Lecture Notes in Computer Science*, pages 17–31. Springer Berlin Heidelberg, 2003.

[36] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *Network & Distributed System Security Symposium (NDSS'11)*, pages 17–33, 2011.

[37] R. Sennett. *The Uses of Disorder: Personal Identity and City Life*. Sociology: History. W. W. Norton, Incorporated, 1992.

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

[39] S. Smalley and R. Craig. Security Enhanced (SE) Android: Bringing Flexible MAC to Android. In *Network & Distributed System Security Symposium (NDSS'13)*. The Internet Society, 2013.

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

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

[42] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia. PlaceRaider: Virtual theft in physical spaces with smartphones. In *Network & Distributed System Security Symposium (NDSS'13)*, Feb. 2013.

[43] H. Verkasalo. Contextual patterns in mobile service usage. *Personal and Ubiquitous Computing*, 13(5):331–342, Mar. 2008.

[44] 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 *Second ACM Conference on Wireless Network Security*, WiSec '09, pages 69–78, New York, NY, USA, 2009. ACM.

[45] Z. Xu, K. Bai, and S. Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 113–124, New York, NY, USA, 2012. ACM.

[46] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with GPS history data. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *19th International Conference on World Wide Web*, pages 1029–1038, New York, NY, USA, 2010. ACM.