Combining Smart Phone and Infrastructure Sensors To Improve Security in Enterprise Settings

Palanivel Kodeswaran
IBM Research India.
Bangalore
India
palani.kodeswaran@in.ibm.com

Dipanjan Chakraborty
IBM Research India.
Delhi.
India
cdipanjan@in.ibm.com

Parikshit Sharma
IIT Delhi
New Delhi
India
ps040791@gmail.com

Sougata Mukherjea
IBM Research India
Delhi
India
smukherj@in.ibm.com

Anupam Joshi
UMBC
Baltimore
Maryland 21250, USA
joshi@cs.umbc.edu

UbiComp'13 Adjunct, September 8–12, 2013, Zurich, Switzerland.

http://dx.doi.org/10.1145/2494091.2499773

Abstract

There is an increasing trend among employees to bring in their own personal device to work , thereby making the enterprise more vulnerable to security attacks such as data leakage from phones. Additionally, users are increasingly running phone apps in a mixed-mode i.e. both for enterprise and personal commitments. For example, phone cameras and microphones are used to record business meetings, often resulting in the case that both employers and employees become unaware of the existence of business data on the phone at a later point in time. The lack of employer control over personal devices raises enterprise data leakage threats, when an employee's phone is lost or stolen. In this paper we describe a system that leverages sensors available on the phone as well as on the enterprise infrastructure to identify business data resident on the phone for further secure handling. Office spaces have traditionally been instrumented with badge swipe readers, cameras, wifi access points etc. that can be used to provide passive sensory data about employees. For example, badge swipes can be used provide approximate location information of an employee where as calendar entries provide information about their schedule and activities. We propose a distributed architecture that leverages the context of the user for speculatively identifying enterprise data from personal data. The basic idea is to understand whether a user is engaged in

enterprise or personal work by inferring her context from a combination of phone and infrastructure sensors. The contextual attributes in our system, such as location, can be sourced from a plurality of sensors on the phone as well as on the infrastructure. We exploit this diversity and propose a cost optimized distributed rule execution framework that chooses the optimal set of predicates to sense on the phone as well as on the infrastructure to reduce sensing cost. Furthermore, the framework also chooses the appropriate site for rule evaluation, either on the infrastructure or phone, to optimize for network transfer cost incurred due to shipping of sensed predicates between the two sites . Combined together,the above two optimizations reduce the battery drain caused due to context inferencing on the phone.

Author Keywords
Security, Context

ACM Classification Keywords
D.4.6 [Security and Protection]: Context.

Introduction
Our world is getting increasingly instrumented with large numbers of sensors that enable us to measure and optimize a number of processes. In addition to these planned sensor deployments, crowd sourced opportunistic sensing is emerging as a pervasive sensing platform, particularly in regions where infrastructure sensors are hard or costly to deploy. This phenomenon can be attributed to the unprecedented rise in the growth of smart phone usage in the recent past. Smart phones come equipped with a myriad of sensors such as GPS,camera, accelerometer, gyro meter etc. An open research problem remains how to collaboratively sense using both infrastructure and smart phone sensors, and combine this data to extract meaningful observations about either an individual or a community [5].

In this paper, we address a manifestation of this problem in traditional office spaces, and describe how we can combine phone and enterprise sensors to infer user context and apply that to secure sensitive enterprise data on smart phones. Smartphones of late have pervaded traditional office spaces as well. There is an increasing trend with employees bringing their own devices (BYOD) within enterprises to access and produce enterprise information. The presence of employee owned, as opposed to, employer issued devices within the enterprise raises serious security issues. Firstly, enterprises find it hard to impose blanket security policies on personally-owned devices as they are restricted from installing custom security software on these phones. Secondly, employees are increasingly using personal applications to generate and consume enterprise information. For example, journalists use *Evernote* during interviews while employees are increasingly starting to use cameras and microphones to capture meeting notes. In other words, there is a convergence between the ways in which personal and enterprise applications are being used on the phone. As a consequence of the above trends, neither the employer nor the employee are able to completely identify enterprise data resident on the phone, a critical first step for protecting enterprise data. To address this, we propose a rule based system that improves enterprise data visibility on the phone by using a combination of phone and enterprise infrastructure sensors. The sensory data from these sources is combined to determine the user's context as enterprise or personal. The inferred context is then applied to speculatively tag files created on the phone as enterprise files, over which additional security policies such as encryption and access control can be applied. We reduce smart phone battery

drainage caused due to sensing by building a distributed rule execution framework that runs on both the phone and the infrastructure. The rule framework minimizes rule evaluation cost by intelligently partitioning sensing between the phone and infrastructure sensors. The main contributions of our work can be summarized as follows:

- We introduce the usage of semantic context towards managing security policies on BYOD smart phones
- We leverage infrastructure context for user activity modelling in enterprise settings to label enterprise data and improve the expressivity possibilities of rules
- We describe an initial cost-optimized distributed rule execution framework for minimizing phone battery usage while inferring user context and enforcing security policies.

Security in Bring Your Own Device (BYOD) Settings

A widely adopted practice for protecting enterprise data in BYOD settings is to create multiple logical containers on the phone [2] - one each for enterprise and personal apps. However this requires the user to manually switch containers when moving between enterprise and personal apps, incurring cognitive overload. Human users prefer to stick to their normal usage styles and any changes to that raises adoptability questions. Moreover, these approaches don't work well with most users that exhibit mixed-use models while at work. A key research question that we address in our work is *How do we speculatively tag enterprise data from personal data on the device while introducing minimal cognitive overhead on the user?*

We believe, in a mixed-use application model, understanding the current context of the user is critical for securing enterprise data that is generated on the phones. Consider the following scenario where in Alice is surrounded by her colleagues at her work place. Alice's context is determined by a number of factors including her current activity, location, as well as availability information. By processing her phone accelerometer readings we may infer that she is sitting while her phone bluetooth neighbors may suggest that she is surrounded by her colleagues. By accessing the calendar information of Alice and her neighbors, we may be able to additionally infer that Alice and her neighbours are scheduled to be in a meeting for that period. The above evidence enables us to predict that Alice is engaged in enterprise work as opposed to having a coffee break with her colleagues. Any data that is created by Alice on her smart phone while she is engaged in enterprise work would typically be enterprise sensitive and hence should be labelled as enterprise data for further secure handling.

Threat Model

The threat model in this paper assumes that the users are benign. Further, users run third party apps in mixed-mode i.e. for both enterprise and personal purposes. The third party apps themselves are assumed to be non-malicious.We assume phones can be easily lost/stolen resulting in enterprise data leakage. We also assume once a data item is identified as enterprise, additional security mechanisms such as encryption and access control can be selectively applied on these data artifacts.

Inferring Context in Enterprise Settings

In this section we describe how we combine the phone and infrastructure sensors to infer user context in enterprise settings. There is significant recent research in

understanding user context by mining smartphone sensor data (e.g. [4, 6]. We exploit this body of work and build our solution using semantic predicates that can be extracted reliably from device sensors such as GPS, accelerometer, microphone etc.
Phone Sensors: We run activity detection on the commonly available phone sensors such as bluetooth, accelerometer etc. to infer low level predicates such as *isSitting*, *isInOffice*, *isSleeping*, *isInWorkGroup*, *isInMeeting* etc. Other researchers [3] have worked on energy efficient approaches for activity mining, and we use them here to infer the above low level predicates.
Infrastructure Sensors: Traditional enterprises are equipped with a number of passive sensors that can be used for free such as card swipes, file server activity logs, calendar information, online and enterprise social network status etc. For the rest of the paper, we use the infrastructure sensors to infer the following predicates viz. *isInMeeting*,*avblOnlineOSN*, *avblEntChat* where *avblOnlineOSN* returns true if the user's status is "available" on online social networks such as facebook, while *avblEntChat* returns true if the user's status is available on the enterprise chat system.

Rules for inferring Context
Now that we can sense using both the phone and infrastructure, the next step is to devise a set of meaningful rules that can be used to infer user context from the sensed data. Consider the following rule

(R1) $isInOffice(user) \wedge isInMeeting(user) \wedge isInWorkGroup(user) \wedge isAvblEntChat \rightarrow enterpriseContext(user)$

The above rule states that the user is engaged in enterprise work if she is in a meeting, and surrounded by her colleagues at office, and is available on the enterprise chat system. This ability to combine infrastructure and phone predicates provides us the flexibility to fine tune the rules to accurately characterize a user's smart phone behaviour within the enterprise. Note that the predicates *isInMeeting* and *isInOffice* can be evaluated both at the phone as well as the infrastructure, incurring different costs.

System Architecture and Implementation

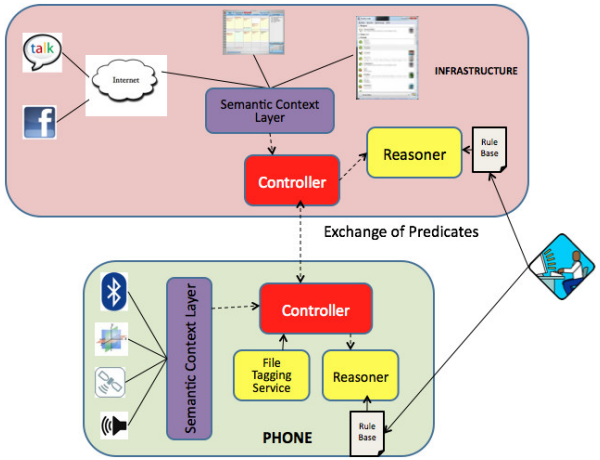

Figure 1: System Architecture

We begin by describing various components of the system.

Context Sensors: Our system consists of context sensors both on the infrastructure and phone side. For example, on the phone, the context sensors include bluetooth, GPS, accelerometer while the infrastructure side context sensors include calendar information, online enterprise and social network availability, file access logs etc. The context sensors are actuated by the controller and help in determining the truth value of the predicates used in our

rules. For example, the truth value of *isInWorkGroup* is determined by scanning the bluetooth MAC ids of nearby devices, and checking if the corresponding owners belong to the same team as the user. Similarly, GPS is used to determine if the user *isInOffice* while we use the accelerometer to determine if the user is travelling.

Reasoner: We run a reasoner both on the infrastructure and the phone that are always in sync i.e. they are loaded with the same rule base provided by the enterprise. We use a forward reasoning engine that implements the RETE algorithm. Given a set of phone and infrastructure predicates, the reasoner evaluates the rules in the rule base to determine user context, which is returned to the controller. In our current implementation, we use the open source AndroJena [1], which is an Android port of the semantic web rule engine JENA.

Controller: The controller module is responsible for actuating the sensors on the phone and infrastructure, when required, to evaluate the truth values of the predicates. The controller is also responsible for shipping predicates between the phone and the infrastructure during distributed rule execution. Internally the controller implements a predicate cache that caches the values of predicates till a defined expiry period. The sensing and network optimizations (described in later sections) are implemented within the controller module.

Phone Module
File Tagging Service The file tagging service is a background service that runs on the phone, monitoring the file system for the creation of new files. When a new file is created by an application, the tagging service queries the phone controller for user context. The phone controller in turn calls an appropriate set of context sensors to determine the values of the predicates on the phone and infrastructure. After assembling all the predicate values, the controller feeds them to the reasoner which evaluates the rules to determine the context of the user which is returned back to the controller. The phone controller ultimately relays the inferred context to the File Tagging Service which labels the file as enterprise/personal so that appropriate security measures such as encryption/password protection can be applied on the file.

Infrastructure Module
The infrastructure module is responsible for querying context sensors on the enterprise such as the calendar information of users as well as their availability on enterprise and online social networks.
Querying Calendar Information: We wrote a simple agent that parses a Lotus notes user's calendar information and exports it into an ics file. When the controller queries for calendar information, the ics file is parsed to determine if the user is in a meeting or not, and the result returned back to the controller, along with the meeting room information if applicable.

Tracking Online Social Network status: Facebook and Gtalk implement XMPP protocol for Instant Messaging whereas IBM uses sametime protocol for it's internal chat system. We use PIDGIN which implements a variety of different protocols to log status changes of any user who is a friend with the system on facebook or is in the system's roster on Gtalk and IBM Sametime. We process the logs periodically to update the current status of the user in our database which is returned when queried by the controller.

Distributed Rule Evaluation

As the number of context sensors on the phone and infrastructure increases, the sensing and data transfer cost increases. Particularly, we must optimize for the battery

drainage on the phone caused by *Sensing on the phone* and *Data transfer between the infrastructure and the phone*

Cost Based Rule Evaluation
With each predicate, we associate a cost of sensing and a cost of network transfer. The cost associated with network transfer is common for all predicates since each predicate is atmost a network packet size. The cost of sensing on the other hand depends on the energy drainage caused by activating a sensor and determining the value of the predicate. We assume the cost of sensing on the infrastructure is a constant minimal value $C_{infraSensing}$. Previous works have measured the cost of using each type of sensor on the phone, and we use that data for determining the sensing cost for each type of predicate. For the rest of the paper, we assume the following order by cost for the predicates on the phone

$$C_{isInOffice} > C_{isInWorkGroup} > C_{isInMeeting}$$

Rule Cost: We associate a cost with each rule based on the predicates in the rule. For example, the cost of the previous rule R1 is obtained by summing the cost of each predicate
$C(R1) = C_{isInOffice} + C_{isInWorkGroup} + C_{isInMeeting} + C_{isAvblEntChat}$

We sum the cost of each individual predicate since the rule is in conjuctive form. However, if the rule were to involve disjunctions, then the cost of the rule would be the minimum of predicate costs. Consider the following rule that allows us to infer *isInMeeting*.
(R2) $isInMeeting(user)_{infrastructure} \vee isInMeeting(user)_{phone} \rightarrow isInMeeting(user)$
Since sensing either at the infrastructure, or the phone is sufficient to infer *isInMeeting*, we argue

C(R2)=$min(C_{isInMeeting(user)_{infrastructure}}, C_{isInMeeting(user)_{phone}})$
Sensing Optimization: Mobile sensors are a great source of energy drain on phones. Whenever we run rules on the phone to infer user context, we utilize phone sensors to evaluate the phone predicates. There have been recent studies that investigate battery drainage for individual sensors on the phone. Based on these studies, we can sort the sensors on the phone by the cost incurred during each use. Cost here is defined as the power consumption caused by sensing as well as network transfer. Thus, given a set of predicates, we define a minimal cost set as the set of predicates that are sufficient for inferring user context such that the cumulative cost of evaluating them is $\leq$ C. We restructure our model as follows

Least cost rule first(LCRF)
The RETE algorithm is a pattern matching based algorithm that builds a dependency tree based on the truth values of the predicates in the rule base. When new facts are instantiated into the model, instead of evaluating all the rules, RETE evaluates only rules that match the truth values of the predicates recently instantiated. In our system, when new predicates are instantiated, we query the RETE engine to determine

1. Rules that will never fire in the current model
2. Predicates that will never need to be evaluated

Thus by sensing low cost sensors first, and asserting them into the model, we can avoid sensing high cost sensors that will never get used in any rule in the rule base. Consider the following rule base with two rules
(R3) $isInOffice(user) \wedge isAvblEntChat(user) \rightarrow EnterpriseWork(user)$

(R4) $isInWorkGroup(user) \wedge !isAvblEntChat(user) \rightarrow EnterpriseWork(user)$
We can evaluate $isAvblEntChat(user)$ at the infrastructure for minimal cost. If $isAvblEntChat(user)$ is false, RETE will return that R3 will never fire, thus avoiding sensing of the high cost location predicate. On the other hand, we will have to sense $isInWorkGroup(user)$ to evaluate R4, which is a lower cost predicate compated to *isInOffice*.

We follow an iterative process for rule evaluation. We initially feed the reasoner with a minimal set of predicates. If this initial set of predicates allows the reasoner to evaluate a rule and determine the context, we relay the inferred context to the File Tagging Service. If the initial set is not sufficient to evaluate any rule, the reasoner responds with the set of void rules which can no longer be fired. Based on the returned set of void rules, we determine the remaining set of active rules that can fire, and their corresponding predicates. We next determine the minimal set of predicates that need to be sensed and iterate the above process till atleast one rule fires, and context can be determined. Our iterative model saves sensing cost for expensive predicates, thus minimizing energy drain on the phone.

Network Optimization
The basic goal of network optimization in our system is to reduce the number of data transfers between the phone and the infrastructure to reduce network bandwidth and power cost.

Controller Modification
Our distributed rule evaluation architecture enables us to optimize on the number of data transfers required to evaluate the rule base. As shown in 1, we run a reasoner both on the phone as well as on the infrastructure. Both the reasoners are in sync with each other i.e. that they are loaded with the same rule base, and therefore the rules can be evaluated either at the phone or the infrastructure. The network optimization decision is to choose the site for rules evaluation–phone or infrastructure. This decision process is performed as follows. Given a set of predicates (as determined at each iteration of LCRF), the controller chooses the rule site based on the number of predicates to be sensed at each site. For example, if we need three predicates from the infrastructure and two from the phone for evaluating a rule, the cost of evaluating the rule on the phone would be three data transfers and on the infrastructure would be two. In this case, our system would choose the infrastructure as the site of rule evaluation. Note that we assume that each predicate is shipped in a packet, and therefore the number of packets and not the size of packets determines the cost of the rule.

Evaluation

Distributed Rule Evaluation

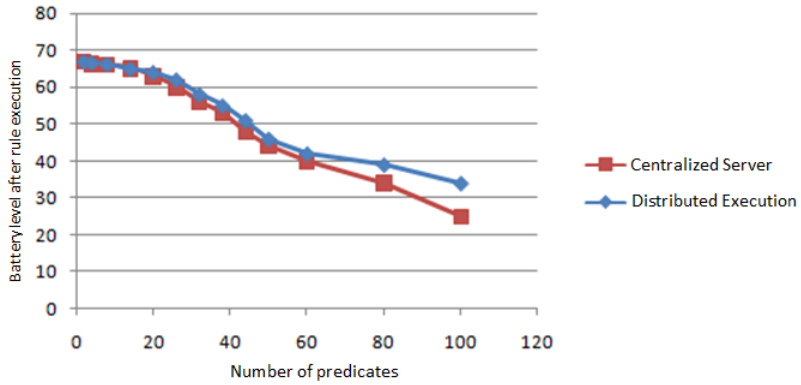

Figure 2: Battery drain between centralized and distributed rule execution

In this section, we present a preliminary evaluation of our system. The experimental set up consists of sensing accelerometer, bluetooth and GPS on the phone, while we emulate server predicates by contacting a local server and waiting for a fixed period of time for a response from the server. We sample predicates from the predicate set such that the lower cost accelerometer and bluetooth predicates are sampled more often than the higher cost GPS, reflecting our sensing optimization on the phone. From the sampled predicates, we run the RETE optimization to determine the set of rules that need to be evaluated and the site of evaluation. Figure 2 compares the battery drain caused by evaluating all rules on the server (red line) with distributed evaluation on the phone and infrastructure (blue line). The y-axis depicts the remaining phone battery level while the x-axis shows the number of predicates. As the number of predicates increases, the battery drain in the centralized execution model becomes more pronounced due to the large number of network transfers involved. On the other hand , the distributed model reduces the number of required network transfers as only predicates that appear in rules that will fire are transferred.

Future Work and Conclusion

The accuracy of the system depends on the quality of the rules specified. One of the key research challenges is to design a set of high quality rules. One possible approach is to speculatively label the files based on existing rules, and periodically, prompt the user to manually label files as enterprise vs. personal. By comparing the manual labels vs. the system generated ones, we can determine the quality of the rules as well as use the feedback to improve the rule set. Another research challenge deals with evolving a set of generic enterprise specified rules in order to personalize them for end users. In future work, we are planning to conduct a large scale pilot to investigate the above research questions.

References

[1] Androjena: http://code.google.com/p/androjena/.

[2] Andrus, J., Dall, C., Hof, A. V., Laadan, O., and Nieh, J. Cells: a virtual mobile smartphone architecture. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, ACM (New York, NY, USA, 2011), 173–187.

[3] Kang, S., Lee, J., Jang, H., Lee, H., Lee, Y., Park, S., Park, T., and Song, J. Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, ACM (New York, NY, USA, 2008), 267–280.

[4] Nath, S. Ace: Exploiting correlation for energy-efficient and continuous context sensing. In *Mobisys* (2012).

[5] Roy, N., Misra, A., and Cook, D. Infrastructure-assisted smartphone-based adl recognition in multi-inhabitant smart environments. In *PerCom* (2013), 38–46.

[6] Wang, Y., Lin, J., Annavaram, M., Jacobson, Q. A., Hong, J., Krishnamachari, B., and Sadeh, N. A framework of energy efficient mobile sensing for automatic user state recognition. In *Mobisys*, MobiSys '09, ACM (New York, NY, USA, 2009), 179–192.