Preserving User Privacy and Security in Context-Aware Mobile Platforms

Prajit Kumar Das, Dibyajyoti Ghosh, Pramod Jagtap, Anupam Joshi, Tim Finin
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
prajit1@umbc.edu, dibsghosh9@gmail.com, pamjagtap@gmail.com, {joshi, finin}@umbc.edu

***Abstract*—Contemporary smartphones are capable of generating and transmitting large amounts of data about their users. Recent advances in collaborative context modeling combined with a lack of adequate permission model for handling dynamic context sharing on mobile platforms have led to the emergence of a new class of mobile applications that can access and share embedded sensor and context data. Most of the time such data is used for providing tailored services to the user but it can lead to serious breaches of privacy. We use Semantic Web technologies to create a rich notion of context. We also discuss challenges for context aware mobile platforms and present approaches to manage data flow on these devices using semantically rich fine-grained context-based policies that allow users to define their privacy and security need using tools we provide.**

I. Introduction

Smartphones or mobile devices that run advanced mobile operating systems are transforming how we communicate with people and connect with the world. Modern mobile operating system platforms like Android and iOS provide applications or "apps" through their "marketplaces". Combining computing ability with apps allows a "smart" phone to accomplish tasks that would either require a personal computer or special hardware components. For example a user can take pictures, record a video, connect to the Internet, navigate using GPS, prepare a presentation and accomplish many other day-to-day tasks, on smartphones.

However, with great power that comes with substantial computing and special hardware based sensing ability of smartphones, comes with added risks to user data. Advanced sensing abilities on smartphones have given rise to a new generation of intelligent applications. Smart assistants like Siri, Google Now and Microsoft Cortana are just a few examples of intelligent applications that are context-aware. All such apps exploit a user's location context to deliver personalized services. They do this by leveraging the user's location at the level of position, i.e., geospatial (latitude-longitude) coordinates. Integrating this with readily available background knowledge allows such systems to identify the location with a known place (e.g., Baltimore), facility (e.g., the BWI airport) or an organization (e.g., UMBC). As a result, location becomes an important aspect of a user's context but there are additional contextual information that includes a user's activity, identity and temporal information [1]. Naturally, protecting the security and privacy of user data now includes the critical task of protecting contextual data. In this chapter, we will discuss access control issues that need to be focused on and discuss solutions that have been proposed by researchers in the domain.

II. Background

Access control generally refers to the process of determining what actions are allowed by a given subject upon objects and resources [2]. The security domain has seen the emergence of various access control models over the years. The most popular models include Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC). DAC refers to access control mechanisms where it is at the "discretion" of the owner of an object. On the other hand, MAC "mandates" control based on security labels assigned to an object. RBAC is a model that uses "roles" to determine access control and in this model permissions are associated with roles, and users are made members of appropriate roles. RBAC suffers from issues of setting up initial role structure and inflexibility in dynamic domains [3]. A pure RBAC solution will not consider dynamic attributes like time of day, which could be critical for determining user permissions. Essentially, it does not take into consideration the context aspect that we so often see, especially in the mobile domain. ABAC models are better equipped in handling access control for such dynamic systems. When it comes to using ABAC models one of the standard system implementations created by [4] is XACML. The XACML standard defines a declarative access control policy language implemented in XML and provides a processing model on how to evaluate access requests. The access control mechanisms that we will discuss are modeled on ABAC.

In this chapter, we focus on the work done in the Platys project and various solutions suggested in it. The Platys project has developed a high-level abstraction of context. Context in Platys is generated by leveraging capabilities of smartphones, discussed in the introduction. Today a significant portion of the human population owns a smartphone and such devices are always on their person. This allows an app on the phone to capture key elements of context: like the user's location and, through localization, characteristics of the user's environment, etc. This leads to a deeper contextual understanding that comes from semantics associated with the location coordinates that are captured. By semantics of a location we mean the notion of a Place, i.e., a location in conceptual terms. For example a place could be "at a study group meeting", "out for jogging" or "shopping for grocery at the local market" - descriptions that combine a set of positions with user's activity, properties of user's environment, and activities of people surrounding the user or interacting with the user. Context extraction using sensors on mobile devices has received great attention in the field. The techniques proposed can be broadly classified into machine learning based models [5] or context modeling based models. Context modeling is carried out using ontologies and rules, and reasoning to infer high level context information from low level sensor data [6]. The Platys project builds on previous work where strong support for context reasoning using ontologies for explicit semantic representation of context [7] has been developed. Platys uses Semantic Web technologies to specify high-level, declarative policies in the form of Jena rules, for defining information sharing constraints using semantic context model. Apache Jena or Jena in short is a free and open source Java framework for building Semantic Web and Linked Data applications.

In other related work, Rein, a decentralized framework for representing and reasoning over distributed policies [8], is an extension of the Rei policy specification language, developed as part of research done in the Platys core group [9]. The Rei language is based on OWL-Lite and allows policies to be specified as constraints over allowable and obligated actions on resources in the environment. Rein, on the other hand, permits policies to be represented in different policy ontologies and requires the use of Semantic Web rules encoded in N3.

KAoS [10] is an early research work in this domain that used the DARPA Agent Markup Language (DAML) language and Description Logic based ontology of the environment, app context and policy to determine access control at different levels of abstraction.

The final policy specification language we will look at is the Ponder policy specification language created by [11]. The Ponder language defines security policy specification for event triggered condition-action rules that define obligations and can be used for auditing access events to critical resources.

III. Main focus of the chapter

As we go forward, we need to ask an important question that motivates a need for strong access control: *What sort of data can be found on a user's mobile device?* These devices are capable of collecting and/or storing extremely private data. For example they can be used to generate a comprehensive digital profile of its users, through social media identities on the phone, photos, financial information, information about books users are reading or movie/TV shows they are watching, Web search history and users' contacts. At times mobile devices are used for authenticating logins into various services through secret shared key, thus acting as a substitute user identity. Mobile devices can also collect and store professional data due to their use in keeping track of work emails, messages, voicemails, calendar appointments and even accessing and storing digital files through cloud storage services. All of these features are available to users through apps that they may install from various app stores. Apps are capable of providing all these variety of services and at the same time they are capable of transmitting stored data on these devices. As a result, improper access controls may enable an app to steal a user's data.

Another reason why access control is becoming a critical requirement on mobile devices stems from the trend of corporations like IBM and many others (BYOD adoption rate is 74% among surveyed companies in January of 2015 as per the Tech Pro report [1]) including healthcare companies adopting Bring-Your-Own-Device (BYOD) as a corporate policy. Naturally, such a policy leads to a potential for data breach and thus to a need for stronger access control on mobile devices, one that is not necessarily available. A 2012 study of medical professionals [12], showed that 84% use the same device for personal and professional activities. Surprisingly 49% of the users' stated that their IT departments had not discussed mobile security issues with them. Such a glaring oversight in healthcare domain has happened due to a preference towards convenience over security, according to the study. Through the above-mentioned scenarios, it is evident that a mobile device has not only become an integral part of a user's life, but essentially it has also become a digital representation of that person. At the same time they are a critical part of a users' professional world.

What are the default security mechanisms that are in place on users' mobile devices? On one of the most popular mobile platforms in the world (i.e. Android), the security model deployed takes advantage of the features provided by the Linux kernel. In the Linux system one user cannot access files of another user. On mobile device users are traditionally implicit because they are an individual's device. As a result the UID associated with users in a Linux system are replaced by appID(s). Thus Android has the apps isolated at a process level and data level through the app's private directory. Access to components on the device, for example hardware like Camera or other OS services or to the Internet is controlled through permissions. Permissions are obtained at install time, in pre-Android Marshmallow era, or at run time starting from Android Marshmallow. Once obtained the OS enforces the permissions.

Why care then? We already have permissions! Unfortunately permissions defined in Android or iOS are too coarse-grained to adequately manage access control on mobile devices. Although the user may be able to use the device permission settings to manage access to say Internet for an app, they are not able to control what domains an app may connect to. They can control whether an app has access to the camera or not but they still do not have any way of stating that an app might only access the flashlight and not the camera. Users do not have the option to block an app from accessing ad api(s). On top of that none of the mobile platforms have provisions for fine-grained access control which are dependent on the context of the user.

[1]Tech Pro report http://www.zdnet.com/article/research-74-percent-using-or-adopting-byod/

For example, a user might prefer to disallow camera access to social media apps at a work location or they might want to disable camera completely during a certain time period of the day.

Therefore, there is a need for fine-grained context driven access control mechanism for apps. This requires a rich notion of context for usage in policy execution. Context may be represented using Semantic Web technologies which will allow handling of various data flow scenarios from and through users' mobile devices. Understanding the factors that impact users' privacy and security is an important part of this work. Only then it will be possible to design policies to mitigate such issues. The challenges in achieving the goal of strong access control on users' mobile devices can thus broadly be broken down into three parts listed below.

- Collaborative context modeling
- Access control policies
- Rule specification

In the following sections we delve into each of these sub problems.

A. *Collaborative context modeling*

In this section we are going to present techniques used by the Platys project for collaborative context modeling. We have seen applications for smartphones evolve to take advantage of features beyond localization like environmental data for example, ambiance, nearby people and resources, and the activities in which they are engaged. An ontology presented by [7] represents various categories of contextual information in pervasive computing environments, specifically, smart meeting rooms. Further generalization of the model to a lightweight, high-level context ontology, in form of the Place ontology, is carried out by [13]. This ontology is used to reason about a general notion of context, as well as to share contextual knowledge. The vision is to generate a collective context using various users' devices and integrating the shared context knowledge from them.

1) Semantic Context Model: In the Platys project, context is modeled to include a semantic notion of a Place. Such elements of context are used in mobile devices to serve intelligent functionality by personal agents. Personal agents are used to proactively control activities on the phone such as switching off during a scheduled meeting or enforcing relevant privacy policies. User's location captured at the level of position, i.e., geospatial (latitude-longitude) coordinates is mapped to a Place or geographic entity, such as a region, political division, populated place, locality, and physical feature. Although position and geographic place information are potentially valuable on their own, from the standpoint of context, Place is a more inclusive and a higher-level abstraction.

A *User* is associated with a *Device* whose *Position* maps to a geographic place (*GeoPlace*) such as *"ABC University"* and to a conceptual place (*Place*) such as *"At Work"*. Some *GeoPlaces* are part of others due to spatial containment and such relationship (*part_of*) is transitive. The mapping from *Positions* to *GeoPlaces* is many to one and the mapping from

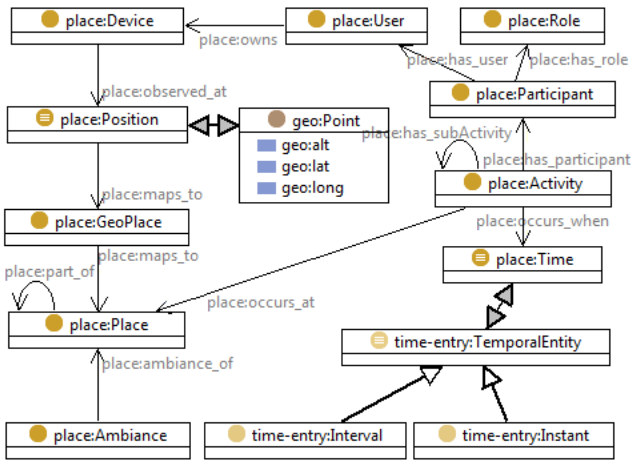


Fig. 1: The Place ontology models the concept of place in terms of activities that occur there

Positions to *Places* is many-to-many (the same *Position* may map to multiple *Places*, even for the same *User*; and, many *Positions* map to the same *Place*). Mapping from *Positions* to *Places* is done through *GeoPlaces* (*maps_to* is a transitive property). An *Activity* involves *Users* under certain *Roles*, and occurs at a given *Place* and *Time*. *Activities* have a compositional nature, i.e., fine-grained activities make up more general ones. This approach reflects the pragmatic philosophy that the meaning of a place depends mainly upon the activities that occur there, especially the patterns of lower-level activities. The idea applies at both the individual and collaborative level.

The Place ontology, mentioned earlier is a lightweight, high-level ontology that models the concept of place in terms of activities that occur at a geo-location. Description logics [14] is adopted, specifically the Web Ontology Language OWL [15], and associated inference mechanisms to develop the model. OWL supports the specification and use of ontologies that consist of terms representing individuals, classes of individuals, properties, and axioms that assert constraints over them. Figure 1 shows the core classes in the ontology and their relationships.

2) Information sharing policies: Users require appropriate levels of privacy control to protect the personal information their mobile devices are collecting including the inferences that can be drawn from the information. For example, in a healthcare scenario, if a user has an accident, it might be right to disclose relevant information (medical records, history, etc.) to the paramedics on the scene and only while they are providing their services. Semantic Web technologies are adopted due to two primary purposes:

1) Creation of models for representing and reasoning about a high-level notion of context
2) Specification of expressive policies to control the sharing of contextual information

Policies involve attributes of the subject (i.e., information recipient), target (i.e., the information) and their dynamic context (e.g., are the parties co-present). A prototype system in

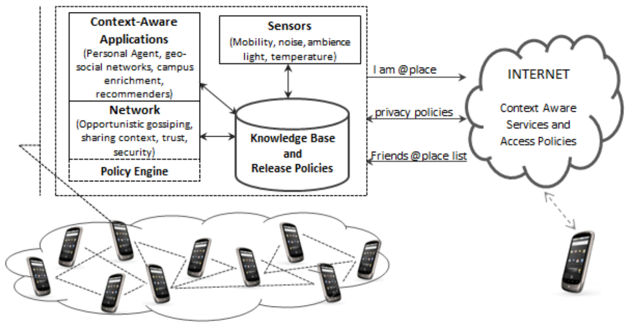
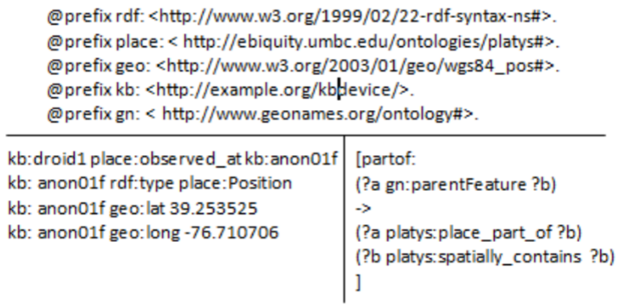

Context-Aware Applications
(Personal Agent, geo-social networks, campus enrichment, recommenders)
Sensors
(Mobility, noise, ambience light, temperature)
Network
(Opportunistic gossiping, sharing context, trust, security)
Policy Engine
Knowledge Base and Release Policies
I am @place
privacy policies
Friends @place list
INTERNET
Context Aware Services and Access Policies

Fig. 2: Interaction among entities in a collaborative information sharing, context-aware system



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix place: < http://ebiquity.umbc.edu/ontologies/platys#>.
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>.
@prefix kb: <http://example.org/kbdevice/>.
@prefix gn: < http://www.geonames.org/ontology#>.
```

```
kb:droid1 place:observed_at kb:anon01f
kb: anon01f rdf:type place:Position
kb: anon01f geo:lat 39.253525
kb: anon01f geo:long -76.710706
```

```
[partof:
(?a gn:parentFeature ?b)
->
(?a platys:place_part_of ?b)
(?b platys:spatially_contains ?b)
]
```

Fig. 3: An excerpt of the assertions made to the KB (left) in Turtle syntax and an example of a Jena rule used to integrate knowledge from GeoNames (right)

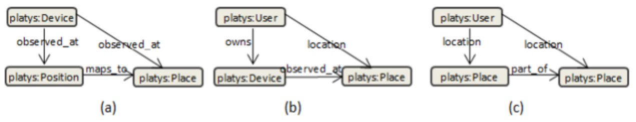


Fig. 4: Property chain axioms to assert knowledge about a user's location. a) Device is observed at the place whose location maps to b) User's location is the place where her associated device has been observed at c) Generalization of user location based on spatial containment (part_of)

a university environment is used to demonstrate the workings of the system. Information aggregated from sensors on an Android phone, online sources, as well as sources internal to the campus intranet are used to individually infer the dynamic user activity using existing machine learning algorithms. The system allows sharing of contextual information directly between devices or through a server. Each device in the system contains a knowledge base (KB) that aligns with the Place ontology. The system also implements a model for specifying and enforcing privacy through declarative policies. The policies allow users to specify situations under which they allow sharing of their context information as well as the level of accuracy at which such information should be shared.

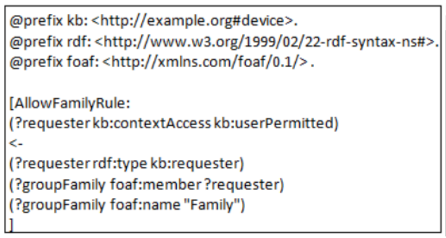
3) General Interaction Architecture: A general interaction architecture for mobile context-aware systems which share and integrate knowledge about their context is depicted in Figure 2. Sensors on devices sense the local context of the user, using mobility tracking and ambient sensing such as light, sound, and motion. The network component opportunistically gathers and disseminates local context information to neighboring fixed or mobile wireless devices. Its policy engine verifies the release policies to ensure context dependent release of information in accordance to the user preferences. Devices interact directly or through services on the Internet. Inferences such as current activity can be drawn from information collected by the sensors, the context information gathered, and additional resources (e.g., the user calendar and open geo-location KBs). The sensor's raw data as well as the inferred context knowledge is stored in a local knowledge base on the device. Context-aware applications and network components use this context knowledge to enhance their functionality. The locally inferred context knowledge may be sent to context-aware services located on the Internet. These services on the Internet, verify, if needed, the statements (proof) of the clients against the access policies. Depending on their functionality, these services provide context information of the user to other users.

Users' information sharing policies provide appropriate levels of privacy to protect the personal information their mobile devices collect, need to be expressive, flexible, and allow for context-dependent release of information. Semantic Web technologies represent a key building block for supporting expressive context policy modeling, reasoning and adaptation [16]. As a result, they are used to model a high-level notion of context and to specify high-level, declarative policies that describe users' information sharing preferences under given contextual situations.

Knowledge Base: The knowledge base (KB) on each device aligns with the Place ontology. Using the Place ontology, devices can share information about their context. The Android Location APIs are used to obtain the position of the device. Given the Position of the users' device, assertions are made and stored as triples into the KB (see Figure 3). Additional online resources are used, specifically GeoNames spatial KB (RDF version) and its associated services, to infer the user's GeoPlace using the following process:

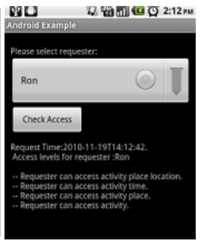
- Using reverse geocoding services to find the closest GeoNames entity to the current position
- Querying GeoNames through SPARQL to get further information about that entity
- Applying transformation rules to the data obtained from GeoNames (see Figure 3)
- Using OWL inference to obtain the triples corresponding to the spatial containment of entities (transitivity of the part of relationship)
- Using ad-hoc property chains (see Figure 4) to infer knowledge about a user's GeoPlace based on the places her associated device is observed.

Activity and Place Inference: The system uses machine learning algorithms to recognize activity (e.g., "sleeping", "walking", "sitting", "cooking"), coarse-grained geographic



```
@prefix kb: <http://example.org#device>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

[AllowFamilyRule:
(?requester kb:contextAccess kb:userPermitted)
<-
(?requester rdf:type kb:requester)
(?groupFamily foaf:member ?requester)
(?groupFamily foaf:name "Family")
]
```



```
Request Time:2010-11-19T14:12:42.
 Access levels for requester :Ron

-- Requester can access activity place location.
-- Requester can access activity time.
-- Requester can access activity place.
-- Requester can access activity.
```

Fig. 5: Left: Jena rule for expressing the policy "share detailed contextual information with family members all the time" Right: Android device screen with reasoning results. It shows access levels for requester "Ron" who is a member of the group family

place, and conceptual place (e.g., "at work", "at home") at different levels of granularity.

4) Privacy Reasoning and Enforcement: In the prototype system, the context is shared among devices by means of queries sent directly between them or through a server. The integration happens on individual device and is a simple operation where the results are added to the knowledge base. For privacy enforcement, users specify privacy policies that regulate the disclosure of Sensor and Inferred context information to server or Inferred context information to other users.

A user defines groups of contacts such as friends and family which are stored in the KB too. The user also specifies context dependent privacy policies and sharing preferences for each group. Privacy policies are expressed as Jena rules over the KB. The focus is not on the information exchange protocol, but on the privacy control mechanisms. Requests are simple messages with required information embedded in them. Whenever a request is received, either at the server or at a device, the privacy control module fetches static knowledge about a user (e.g. personal information and defined groups), the dynamic context knowledge and the user-specified privacy preferences. Access rights obtained by performing backward reasoning confirms conclusions by verifying conditions. Additionally, when access is allowed and according to the user defined sharing preferences, certain pieces of the information might be obfuscated in order to protect user privacy. The implementation used Jena Semantic Web framework [17]. Privacy rules are defined as Jena rules and the Jena reasoning engine is used to perform the reasoning. AndroJena, a porting for Jena to the Android platform [18] is used for drawing inferences on devices.

Policies for information sharing: These policies can describe which information a user is willing to share, with whom, and under what conditions. Conditions are defined based on attributes like a user's current location, current activity or any other dynamic attribute. Since users can have different networks of friends, a variety of group level privacy preferences are employed. For example: "share detailed contextual information with family members all the time" and "do not share my sleeping activity with Teachers on weekdays from 9am to 5pm". Figure 5 shows the representation of the first rule as a Jena rule (left) and the results on a test screen are provided using the results of the reasoning engine (right).

Policies for Obfuscating Shared Information: These policies can depict what information a user is willing to disclose with different accuracy levels. For instance, she may be willing to reveal to her close friends the exact room and building on which she is located, but only the vicinity or town to others. Furthermore, a user may decide not to disclose her location to advertisers. For these purposes generalization models maybe used, which are discussed in detail in the Rule Specification section. These models are simple subsumption hierarchies over location and activity entities (e.g., City is subclass of State which is a subclass of Country).

B. Access control policies

In this section, the two high level aspects of access control that we will learn about, involves the information sharing between a device and a server or another device, a device and the apps on the device. In the previous section we have discussed the generation of a collaborative context model and the use of information sharing policies that control the context data that will be shared to generate such a model. We further examine this scenario with respect to a BYOD policy set (as system level policy described in Rule Specifications section) and a user policy set in a social media application scenario.

1) Inter-device information sharing: In Figure 6 we present the major components of the system. The system consists of client devices, server side modules and the Internet services that provide social media requests. The client devices are context-aware smartphones. Client devices as well as the server side modules contains a user profile repository, a privacy control module and content preferences. The server side also contains content aggregator, learn & share module and privacy control module. The content aggregator combines social media data, photos, and videos from Internet services or other sources like university information portals. Learn & share module inferred a user's dynamic context using sensor data collected on phone, information from the content aggregator and online sources such as user's calendar. The inferred context is shared with corresponding client device so that the device along with server can handle further context sharing queries from other clients. The requester queries are passed through the privacy control module to constrain the information flow and hence to protect the user privacy. The privacy control module provides access control mechanisms and aids in controlling information flow within the system. On the client device, it enables privacy sensitive and resource sensitive reasoning over sensed data along with privacy enforcement between peer devices sharing contextual information. The interaction between various components of the system can be described as follows:

- The user has a client device to collect the sensor data periodically. This data is passed to the "learn & share" module on the server as allowed by the privacy control module on client device. The privacy control module

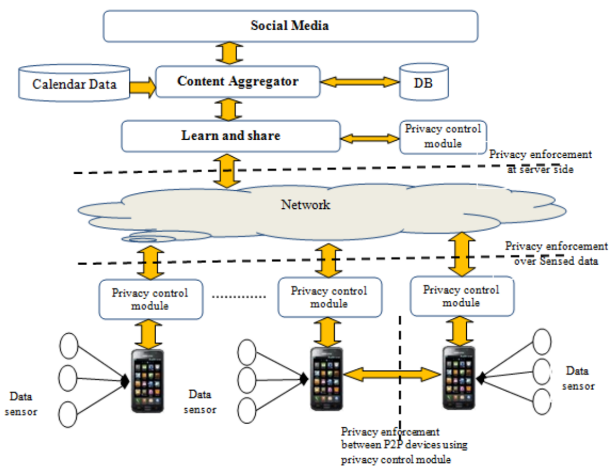


Fig. 6: Server âĂŞ Client data flow architecture

decides which specific sensor data can be shared with the server based on user-specified privacy policies.

- The “learn & share” module infers the user’s context using sensor data and information from content aggregator and other online sources. The context consists of current location, activity and additional surrounding information like nearby people. The inferred knowledge is passed to the corresponding client device so that it can handle context access queries from other clients.
- Access requests are passed through the privacy control module which in turn decides whether to allow or deny access. If the requester is granted the access then it determines a set of information to be shared by performing reasoning over the context information and user’s privacy preferences. These requests can be made by one client device to another or from a client device to the server.
- Figure 6 shows the three different ways in which information can be shared in the system, namely: context information sharing between the client devices, sensor data sharing between a client device and server, and context information sharing between a client device and the server.

Access Control: The user’s personal information can be shared between a client device and the server side application or between two client devices. To constrain the information flow, privacy enforcement can be done between peer client devices and at server side for contextual information.

Privacy enforcement between peer client devices: Learn & share module from server side shared the owner’s contextual information with corresponding client device. The client device further keeps track of the context and responds to queries made by other peer devices. Code 1 shows the sample contextual information for user “Alice”. The contextual information is to be protected and should be shared only with requesters having sufficient privileges. The user can provide detailed privacy policies specifying what context information can be shared with whom, when, and under what conditions. If users are reluctant to provide any specific policies then they can opt for either default models of the system viz. Optimistic Model - where the system can provide response to any query with all possible relevant information associated with a user’s activity such as associated place, location and the timing details, or (ii) Pessimistic Model - where the system can refrain from revealing activity associated information. Apart from these default system settings the user can define her privacy rules with various degrees of accuracy levels. She can also use the system to obfuscate certain pieces of information to protect the context information. This way the system can protect the users’ privacy by varying accuracy levels of activities, associated locations and timestamps.

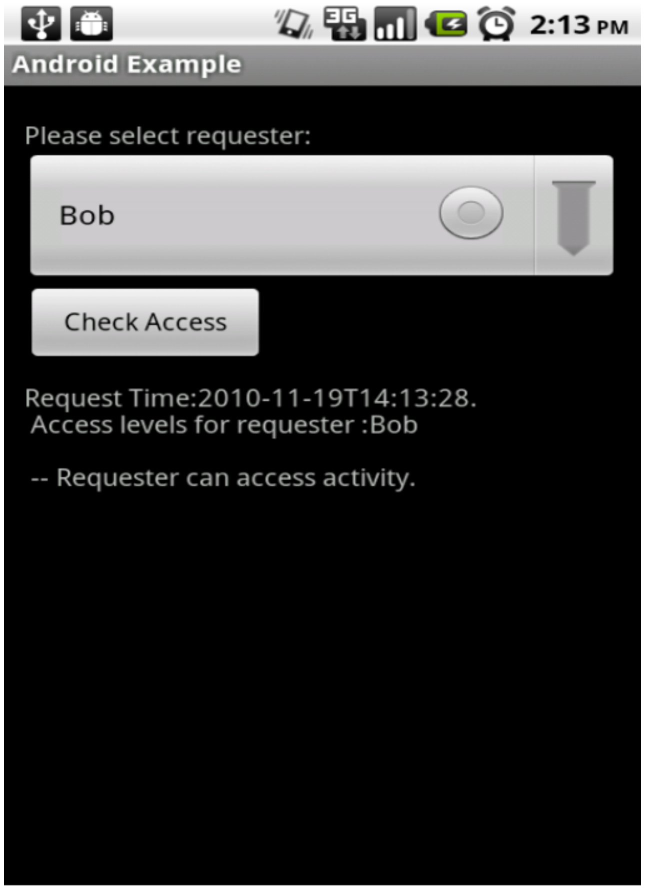
```
ex:Alice a foaf:Person ;
foaf:name "Alice" ;
platys:has role platys:Student .
platys:Sleeping a platys:Activity ;
platys:is performed by ex:Alice ;
platys:has participant ex:Alice, ex:John ;
platys:occurs at
platys:Class LH1 ;
platys:occurs when "2010-11-19T14:12:42".
platys:Class LH1 a platys:Place ;
platys:has location "39.253525, -76.710706".
```

Listing 1: Code 1: Contextual information represented in N3. It consists of activity, associated place, location, time and nearby users

Whenever any participant in the systems tries to access any protected resource (activity, place, location or any additional information) the query has to be sent to the privacy control module. This module fetches the user knowledge, dynamic knowledge and user-specified privacy preferences to evaluate the query. As a result it will decide whether participant is allowed to access the protected resource or not. In the former case, it might obfuscate certain pieces of information as per user-specified privacy policies to protect user’s privacy. These policies are represented as Jena rules in Code 2, Code 3 and Code 4 respectively. When a request would be made by “Ron” who is a family member of the user then he should be able to access the user’s detailed contextual information. If the request came from “Bob” who is a member of the friend group and the user’s current activity is “Sleeping” then the requester is allowed to access a user’s activity information excluding the associated place and location. Figure 8 shows the access level for requester “Ron” and Figure 7 shows allowed access for user “Bob” after performing reasoning on the device using user information, dynamic knowledge and privacy policies mentioned in Code 2, Code 3 and Code 4.

```
[AllowFamilyRule:
   (?requester ex:memberOf ?groupFamily)
   (?groupFamily foaf:name "Family")
->
   (?requester ex:canAccessActivity "True")
   (?requester ex:canAccessActivityPlace "True")
   (?requester ex:canAccessActivityTime "True")
   (?requester ex:canAccessPlaceLocation "True")]
```

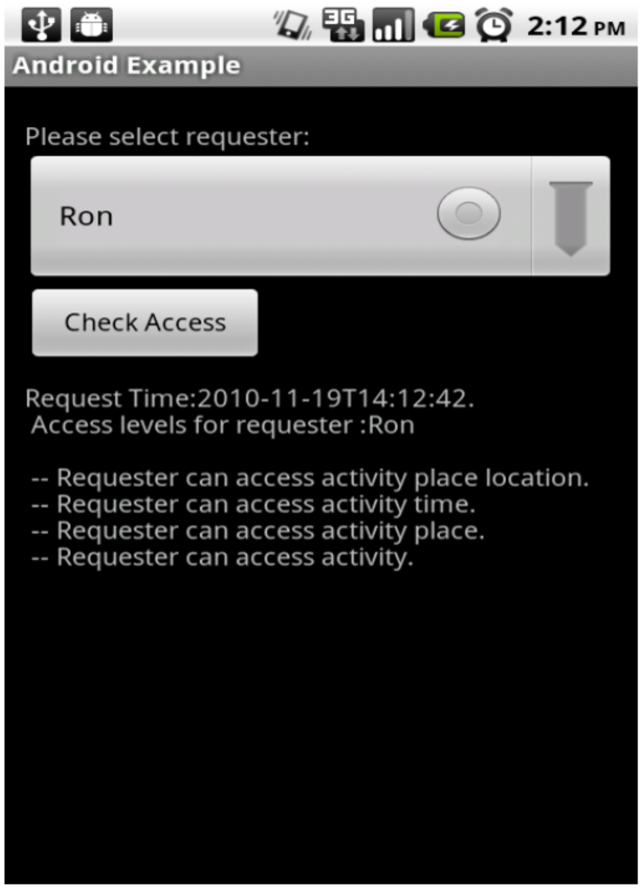
Listing 2: Code 2: Policy to share detailed contextual information with family members



Please select requester:
Bob
Check Access
Request Time:2010-11-19T14:13:28.
Access levels for requester :Bob

-- Requester can access activity.

Fig. 7: Android device screen with reasoning results. It has access levels for requester “Bob” who belongs to friend group



Please select requester:
Ron
Check Access
Request Time:2010-11-19T14:12:42.
Access levels for requester :Ron

-- Requester can access activity place location.
-- Requester can access activity time.
-- Requester can access activity place.
-- Requester can access activity.

Fig. 8: Android device screen with reasoning results. It has access levels for requester “Ron” who belongs to family member group

Privacy enforcement at the server side: At the server side learn and share module, infers the user’s dynamic context such as current activity, associated place and location, and nearby people. This contextual information needs to be protected and should only be shared with requesters with sufficient privileges. The server has information about all the system users whereas a client device has information about its owner. Due to this, the server can handle requests for all the users whereas the client device can handle requests about its owner only. The main distinction between the access requests made by a client device to a peer device and to a server is that the latter request contains a specific userId. This userId is used to retrieve specific users’ information. Consider a privacy policy as shown in Code 4, which states “allow location access to teachers on weekdays only between 9am and 6pm”.

```
[ShareActivityWithFriendsRule:
  (?requester ex:memberOf ?groupFriends)
  (?groupFriends foaf:name "Friends")
  (?someActivity platys:is performed by ex:Alice)
  notEqual(?someActivity, platys:Listening To
     Lecture)
->
  (?requester ex:canAccessActivity "True")]
```

Listing 3: Code 3: Policy to share activity information with friends all the time except when a user is attending lecture

The system uses the userId to retrieve the related information and then checks whether the requester is a member of the group by verifying the requesters’ userId. The example explained above involves representation of a user’s personal resources such as list of friends, group’s information, contextual attributes like current location and current activity.

```
[ShareActivityWithTeachersRule:
  (?requester ex:memberOf ?groupTeachers)
  (?groupTeachers foaf:name "Teachers")
  (?requester ex:requestTime ?localTime)
  (?localTime time:dayOfWeek ?day)
  ge(?day, 1) le(?day, 6)
  (?localTime time:hour ?hour)
  ge(?hour, 9) le(?hour, 21)
  (?someActivity platys:is performed by ex:someUser)
  equal(?someActivity, platys:Sleeping)
->
  (?requester ex:canAccessActivity "False")]
```

Listing 4: Code 4: Policy to not share sleeping activity with Teachers on weekdays from 9am - 9pm

Reasoning Engine: The reasoning engine handles the requester queries and performs reasoning for access control decisions. The system uses Jena Semantic Web framework [18] for performing the reasoning over context data. Jena inference

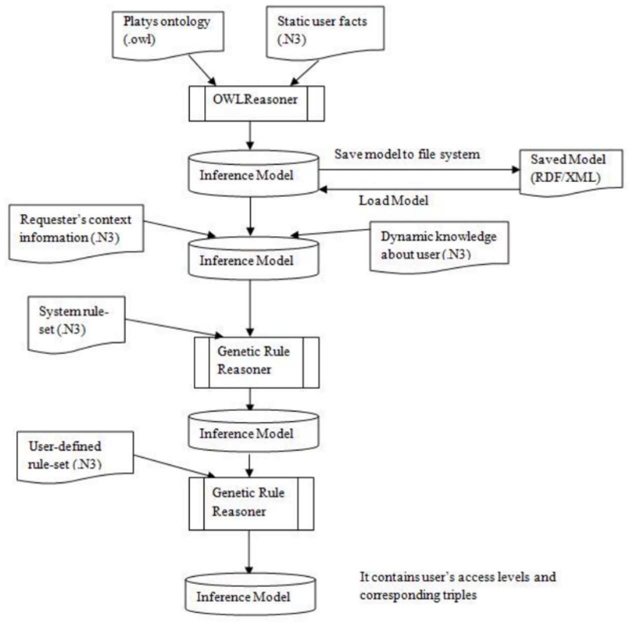


Fig. 9: Reasoning flow

system allows the support of various inference engines or reasoners. These reasoners are used to infer additional facts from the existing knowledge base coupled with ontology and rules. In particular, Jena uses the generic rule reasoner which is included in Jena2 as a general purpose rule-based reasoner. It is used to implement both the RDFS and OWL reasoners. It needs at least a rule set to define its behavior. In the system, the reasoner uses the context ontology, static user facts like identity and group information along with the user-specified privacy rules to generate an inference model. This inference model is used for responding to the requester queries. This process is shown in the Figure 9 and works as follows:

- Create the instance of OWL reasoner specialized for context ontology and then apply that to the users' static information to generate an inference model. This inference model consists of additional statements inferred from static knowledge and ontology. As the user information and ontology are not changed often, it is quite safe to save the model on external storage and reload it for subsequent queries rather than generating it each time.
- The requester's contextual information is extracted from requester query and along with user contextual information it is added to the inference model to generate a new model.
- The system-level polices are executed against the inference model using an instance of generic rule reasoner. It is an optional feature and it's used to enforce certain organization level policies. It will create a new model having SystemPermitted and SystemProhibited statements to enforce system policies over the users' contextual information. If the user is a sole owner of client device then this step can be skipped. The detailed description of this feature is provided in the next section.
- The user-specified privacy rules are executed against the inference model from previous step to generate a new inference model having requester access levels.

The system will use the new model to decide what can be shared with requester and respond accordingly.

System Level Policies: The context-aware systems are used by individuals to organization and from social-networking application to military domains. In case of military domains or organizations, the user may not be the sole owner of client device and there is a strong need of robust security mechanisms. It can be in the form of multi-level secure systems where the system-level policies must override user-level policies. This highlights the need of system-level policies along with user-specified policies. The system-level policies should be defined by the system-administrator to ensure that sensitive resources are always protected from illegitimate access. Consider a system-level policy as "Do not share the user's context if she is inside a military building BuildingXYZ" and a user-specified policy as "Share my context with family members all the time". The system- level policy states that the user context won't be shared with anyone if she is inside BuildingXYZ whereas in latter policy user specifies to share her context with family members all the time. In this case the system-level policy should override user- specified policy and hence, if the user is inside BuildingXYZ then her context will not be shared to anyone including her family members.

2) Intra-device information sharing: In this use-case scenario intra-device information sharing policies consider the data flow from the device's sensors to requester of said data. However, the requester in this case, is an entity which resides on the phone itself, i.e. an app. Data flow control, in this case, will have obvious user implications. When the data is leaving the device and is being shared with an external entity it makes sense that the user might want to control what information is shared. On the other hand when the data is being shared with apps on a user's device an implicit trust assumption should not be the norm. The reason being, most users install apps from a variety of developers and sometimes a variety of sources. It may be presumed that standard app market places are monitored by their respective owners but such a presumption may not necessarily be true [19]. There are examples of legitimate apps stealing user sensor data and sending them over the internet for ad purposes (Android Flashlight App Developer Settles FTC Charges It Deceived Consumers [2]). Under such circumstances it's important to control what data flows from the sensors on a mobile device to the apps.

As discussed in the introduction, Android's security model is based on the Linux kernel's security features and application sand-boxing. Android application packages execute in their individual sand box and are built from multiple components that provide various functionality. To understand the intra device access control let's take a look at a prototype system.

[2]FTC release https://www.ftc.gov/news-events/press-releases/2013/12/android-flashlight-app-developer-settles-ftc-charges-it-deceived

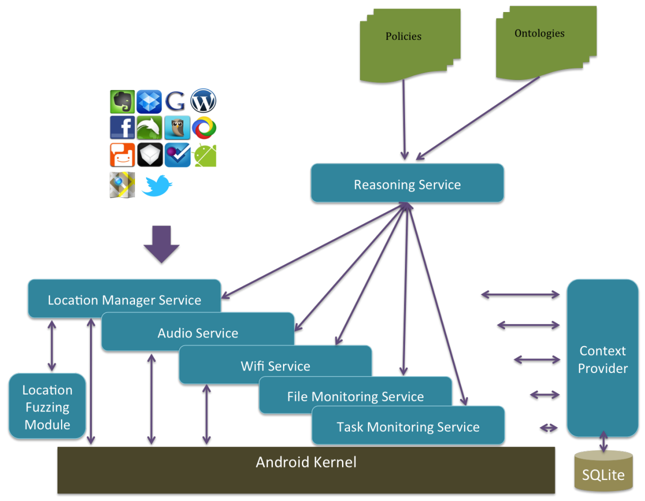


Fig. 10: Architecture of the intra-device access control prototype Android system

In this system few of the important device services on Android platform have been modified to serve as a way to control data flow. Specifically the LocationManagerService, AudioService and WifiService are modified. The altered system runs a reasoner on top of user-context and access control policies to determine response for an app's request to system resources. The system architecture of such a controlled system can be seen in Figure 10

Access Control: A significant point of failure for such a system, would be that apps do not expect to be blocked from accessing the data on a mobile device. As a result, apps would simply crash if such a block is put in place. Therefore, as an alternative solution, obfuscation is used. As part of the obfuscation solution a location randomization module is created. This module is used to generate fake coordinates for location of the device. The algorithm used for generation of a new set of coordinates from device's current location is similar to the ones deployed in apps reporting nearby places of interest.

Given a location L and a radius R location randomization module generates L' where $L' \epsilon \{l$: l is in the bounded circle with radius R and origin $L\}$. This technique is used to find points within a distance of a latitude/longitude using bounding coordinates. The shortest geodesic distance between two given points `P1=(lat1,lon1)` and `P2=(lat2,lon2)` on the surface of a sphere with radius R can be calculated using the formula:

```
d = arccos(
sin(lat1) * sin(lat2) +
            cos(lat1) * cos(lat2) *
            cos(lon1 - lon2)
            )
            * R
```

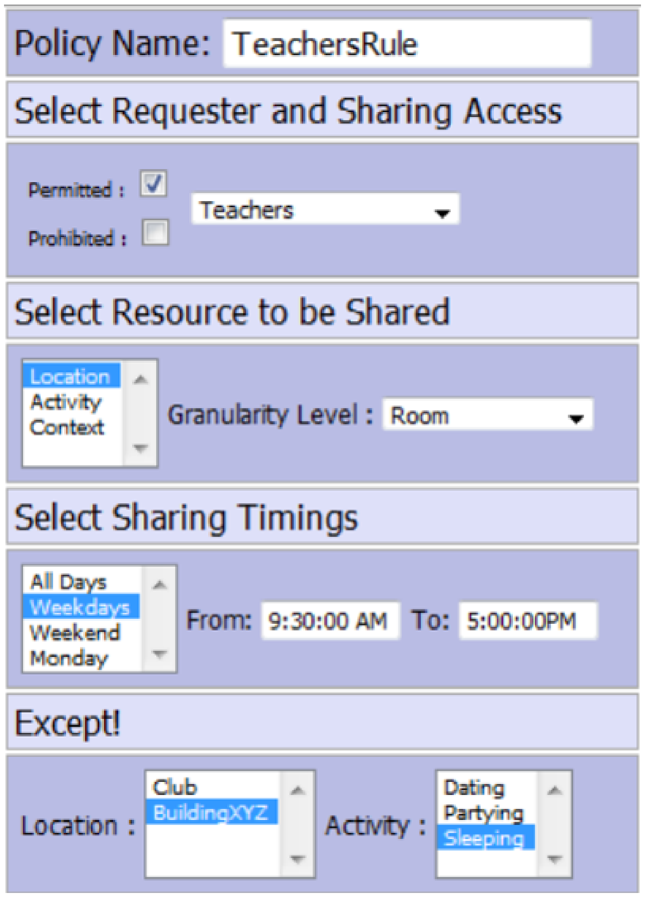
It computes the bounding coordinates of all points on the surface of a sphere that have a great circle distance to the point represented by this GeoLocation instance that is less or equal to the distance argument. Once these coordinates are obtained one maybe randomly selected and returned to the calling app instead of a failure response. This ensures that apps do not crash and user data is protected at the same time. Similar obfuscations can be generated for other services too. For example the camera component can be interrupted to provide fake images to a requesting app or fake contacts could be returned to an app which requests user's contact lists. The reasoning process remains the same as before. A set of static user data, an ontology defining user context and user's dynamic context information is combined with a list of system and user defined policies to reason over and decide whether the data should be shared or not.

Reasoning Engine: The heart of the access control system is the reasoning engine. Running as a system service it uses the policies that are stored on the device. The policies may be downloaded from a server or maybe selected by the user. Details of such a process will be discussed in the section Rule Specifications. As seen in the architecture diagram in Figure 10 upon a data/resource request made by an app, the various services on the device queries the reasoning service for an access control decision. The reasoner then uses policies on the device and based on the current context returns a decision asynchronously to the requesting service. The context string is forwarded to the reasoner by the requesting service.

C. Rule specifications

Till now we have discussed context generation and presented use-cases for access control implementations. The last and equally important aspect of managing data flow in a smartphone requires a process to specify rules to be implemented. For this purpose we discuss the use of system level access control rules that might be specified by administrators. We also discuss the use of user policies for added protection. The two scenarios effectively take care of BYOD use-case organizational policies and users' personal privacy and security policies. An initial set of default policies may be obtained through a trusted third party. In case of a rule conflict between system and user policies the following resolution process is used. Access predicate of the rules can take the following values: SystemPermitted, SystemProhibited, UserPermitted and UserProhibited. As default deny policy is followed to determine access control, if SystemProhibit is present in the set of conflicting rules, the access is denied. Additionally system policies trump User policies and therefore if SystemPermitted and UserProhibited are present in rule set, the access is granted. Finally, there are UserPermitted and UserProhibited values in the access predicate of the rules, data access is denied.

1) User policy editor: As described by [20], policies may be enforced indefinitely or for a certain time period based on a policy certificate validity period or a combination of timeout or loss of contact with an assigned network. However, the user has the option of modifying or adding rules to the policy through the interface shown in Figure 11. The Requester


Policy Name: TeachersRule

Select Requester and Sharing Access

Permitted : ✓

Prohibited :

Teachers

Select Resource to be Shared

Location

Activity

Context

Granularity Level : Room

Select Sharing Timings

All Days

Weekdays

Weekend

Monday

From: 9:30:00 AM To: 5:00:00PM

Except!

Location : Club, BuildingXYZ

Activity : Dating, Partying, Sleeping

Fig. 11: Policy editor settings app

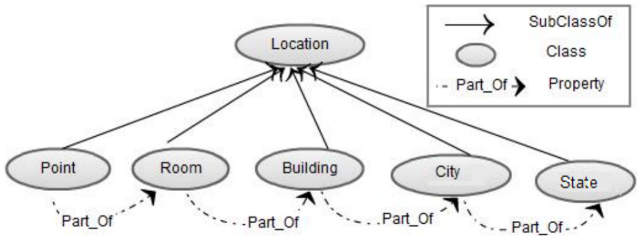


Fig. 12: Location hierarchical model

option allows choice of permitting or prohibiting access to specific entity. This entity may be outside the user's mobile device like a friend or family member's phone. It may also be an app on the user's own mobile device. The second clause defines what data/resource may be shared. The third clause defines a timing contextual constraint within which the rule applies and finally the system has exception clauses that may negate the rule if necessary. The generalization and specialization options for context constraints are defined using the Place ontology discussed in Section on Collaborative Context Modeling.

Generalization: Generalization involves replacing a value with a less specific but semantically consistent value in order to protect user data privacy [21]. The system uses context-data

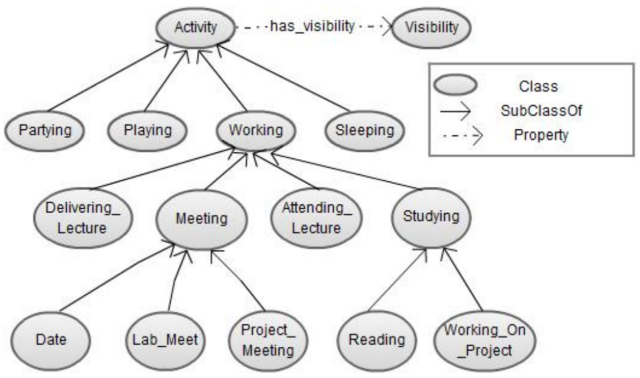


Fig. 13: Activity hierarchical model

generalization to allow information sharing on different levels of granularity.

Location Generalization: In order to support location generalization, the ontology uses hierarchical model for location. Location is a super class of Point, Room, Building, City and State classes. The Point class is used for denoting GPS coordinates whereas Room and other subclasses are used to denote different levels of abstractions for location. The transitive "Part Of" property creates a location hierarchy based on some simple axioms like "Room is a part of Building". The reasoning engine uses this ontology to infer different relations existing between instances of these subclasses (see Figure 12).

Activity Generalization: Along the lines of location generalization, let's look at activity generalization for allowing users to share different descriptions of their current activity to different set of requesters. In many cases, the user is willing to share more generalized activity rather than a precise one. For instance, if a user is attending a confidential "project meeting" then she might want to share it in a more generalized way as "working" or simply as a "meeting" (see Figure 13).

Policy determination: Selecting policy that needs to be implemented cannot always be driven by human users or administrators. A certain level of automation is desired in this process. Why? Studies have shown that people need to wade through a sea of information in order to determine the right privacy preference [22]. However, such information might not necessarily enable them to choose the best possible policies [23]. Allen Westin's perspective on privacy defines privacy as the ability for people to determine for themselves "when, how, and to what extent, information about them is communicated to others" [24]. However, these approaches on information access control have been found to be flawed [25]. According to [26] information accountability is the ability to determine whether usage of information is appropriate and the ability to identify the violator.

Considering data being accessed by apps on a user's mobile device, app provenance maybe used as a way to assign accountability. It should be noted that the notion of app provenance is not restricted merely to geographical origin; of the developer of the app. It can also refer to online repository from where the app is downloaded to the mobile device. Availability of such information opens up possibility of capturing a whole new set of access controls on device; apps can be restricted from being installed in first place based on geographical origin or online origin. Policies can be pre-written to restrict access to only a subset of device resources for apps, which takes into account app origin and other contextual information of the device user and the app itself.

Automatic policy generation/implementation can thus be done in three stages.

- Stage 1 (App data gathering phase): Apps are searched in android marketplace by name. Thereafter, package name, app version, app rating, app developer information, app permission data, app descriptions are collected. Once app developer information is available app's organization data from DBpedia is retrieved and location information of the organization is extracted. In [27] the author collected developer information from DBpedia.
- Stage 2 (Pre-decision making phase): Triples are generated which represent facts about the app and stored on the mobile device.
- Stage 3 (Decision phase): The reasoner upon a receipt of an access control request uses the triples stored about the app, the policy stored on the phone and the context information, generates a grant or deny response.

For this purpose, take a look at the PlatMob [28], ontology for mobile device applications to capture application provenance data from heterogeneous sources. PlatMob is an extension of the Platys ontology presented earlier. The Place ontology represents a high level context ontology by [13]. The concept of a requester of data is defined in the Platys ontology [29], [30]. The PlatMob ontology allows modelling of richer notion of an application's context and developing privacy preservation policies far more complex than that of Android's default all or nothing policy. PlatMob is an aggregation of four domains of knowledge as shown in Figure 14.

- PlatMobile, representative application data and context on the device the app is installed
- Platys, representative device users' context
- PlatMobileLOD, representative application context sourced from marketplace and linked open data cloud
- DBpedia: Country

PlatMobileLOD models data available for an app from external sources whereas PlatMobile models on-device runtime information about the application. PlatMobile represents on-device resources using the following entities. AppGroup defines an application group; can be apps accessing Internet or apps which deals with location data or apps which does media capture etc., Permissions defines classes of permissions a specific app has; and is represented as a collection of String literals which can be collected from apps AndroidManifest.xml file, File represents the file system resource of the device; it is further divided into ImageFile, AudioFile, VideoFile, BinaryFile, TextFile subclasses, Hardware abstracts devices hardware resources e.g. Camera, Keyboard, Microphone and Storage, Service is representative of system and third party services.

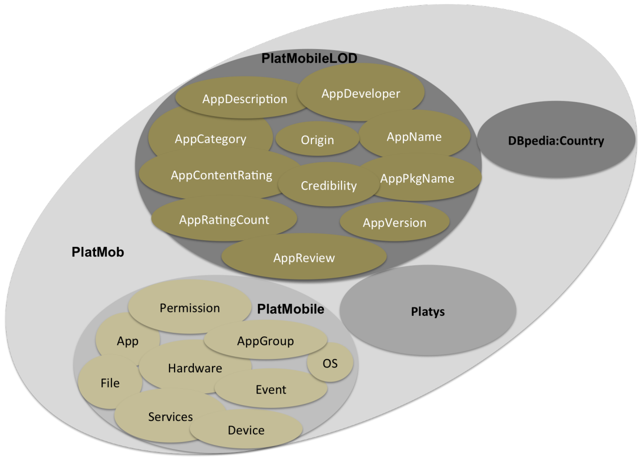


Fig. 14: PlatMob ontology

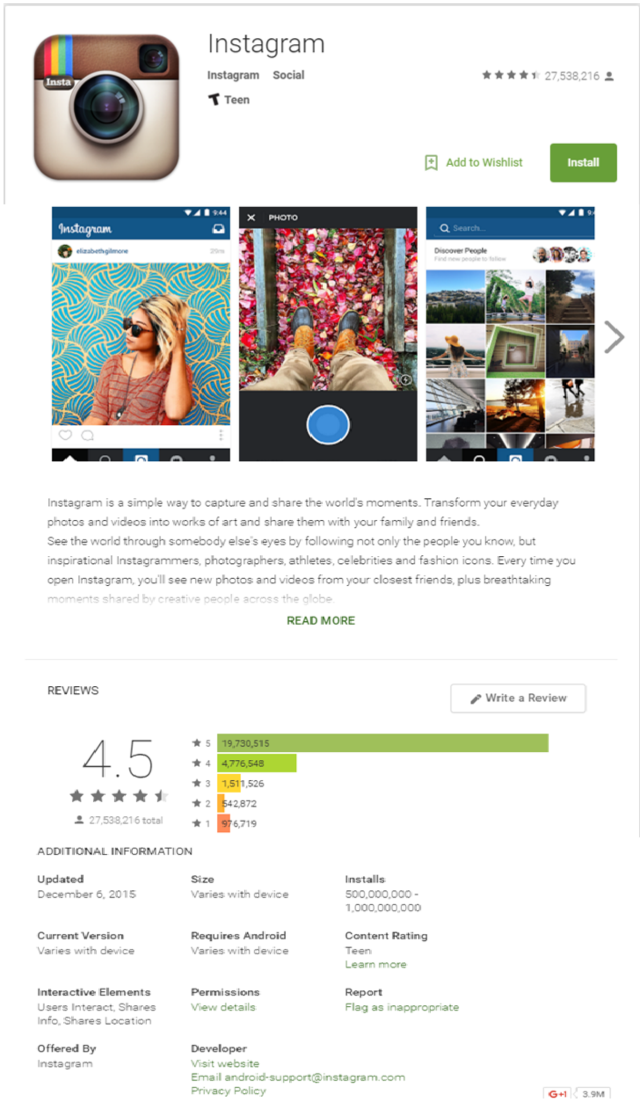
PlatMobileLOD represents different entities that captures app context from external data sources like app marketplace and linked open data sources like DBpedia. PlatMobileLOD comprises of AppCategory, AppContentRating, AppDescription, AppDeveloper, AppPkgName, AppName, AppRatingCount, AppReview, AppVersion, Origin, BlackList and Credibility entities. PlatMobileLOD uses DBpedia Country ontology to describe Origin and Blacklist entities, e.g.

- http://dbpedia.org/resource/Iran a platmob:BlackList .
- http://dbpedia.org/resource/Iran platmob:origin ex:com.farsitel.bazaar .

Using the dbpedia-owl:location property, origin country information can be extracted for a developer name. Figure 15displays Google Play's entry of a popular photography app, Instagram. Google Play displays a list of attributes, including but not limited to the ones listed below:

- App description
- User reviews
- User rating and rating count
- Developer information
  - Developer website
  - Developer email
- App version
- Download count
- User review count
- Category etc.

Complex policy generation: Using the app provenance information it is possible to represent highly rich context-based policies. An example scenario could be as follows. Imagine a person with a top security clearance who might be the target of a foreign entity for perpetrating acts of espionage. The targeted individual has access to sensitive information and is an avid smartphone user. The foreign entity has considerable reach to this person and has successfully installed a backdoor on this person's device. The targeted user visits a secure data facility on a regular basis due to his/her nature of job. The malware residing on this individuals


Instagram

Instagram Social

Teen

★★★★½ 27,538,216

Add to Wishlist

Install

Instagram is a simple way to capture and share the world's moments. Transform your everyday photos and videos into works of art and share them with your family and friends.
See the world through somebody else's eyes by following not only the people you know, but inspirational Instagrammers, photographers, athletes, celebrities and fashion icons. Every time you open Instagram, you'll see new photos and videos from your closest friends, plus breathtaking moments shared by creative people across the globe.

READ MORE

REVIEWS

Write a Review

4.5

★★★★½

27,538,216 total

★ 5	19,730,515
★ 4	4,776,548
★ 3	1,511,526
★ 2	542,872
★ 1	976,719

ADDITIONAL INFORMATION

Updated December 6, 2015	Size Varies with device	Installs 500,000,000 - 1,000,000,000
Current Version Varies with device	Requires Android Varies with device	Content Rating Teen Learn more
Interactive Elements Users Interact, Shares Info, Shares Location	Permissions View details	Report Flag as inappropriate
Offered By Instagram	Developer Visit website Email android-support@instagram.com Privacy Policy	

Fig. 15: Google Play entry for Instagram Android App

compromised device does a string of activities it starts audio/video capture at stipulated time of the day or day of the week based on this individuals calendar schedule and only if device screen is not active or the user is not in the middle of any calls at that point of time; these media files are then uploaded to remote servers when connected to Wi-Fi and not using the data plan to avoid raising suspicion; once uploaded the malware removes these files to cover its track. Accessing user calendar is possible for any user land app once it is given access at install time. And the use case delineated here is fairly generic since all the associated activities are straightforward. Existence of privacy policies capable of controlling application specific resource access based on dynamic requester app and device context can address breach of privacy under such circumstances. PlatMob may be used to generate privacy policies for situations outlined below:

Disable camera recording on weekdays between 9 AM - 5 PM if device user is in a meeting

OR

Disable audio / video capturing by some app A belonging to app group Internet and with origin country O, where O is Blacklisted between 9 AM - 6 PM if device user is at location L

OR

Do not allow some app A to run on device between time t on days d, when app A belongs to app group G, app A has origin country O, app rating > R, no of downloads > N, app category is P, app is developed by group X, device user is at location L engaged in an activity Q, and location L belongs to country C

For the first and second scenario above, all context information are available from the device. However, that changes in the third scenario. Applications rating, download count, category, developer and app origin information is harnessed from apps market place and DBpedia.

IV. Discussion

We have discussed two prototypes one for inter-device privacy and security implementation and the other for intra-device privacy and security control. The first prototype uses a client server model with the requester being able to send requests to another device directly or through a server. The access request is processed by the policy framework. For the intra-device privacy and security the prototype implementation has two major components: a privacy control module and a device operating system. The privacy control module aims to protect user privacy by performing reasoning over the context. It deals with the resource to be protected, the owner of a resource and the requester who wants to access it. More abstractly, it accepts an RDF triple (U, C, Q), where U is the identity of the requester, C is the requester's context (expressed as RDF triples in the Platys ontology), and Q is the query pertaining to context information. Both prototypes consider contextual information and sensor information as the resources that changes dynamically for the user, and provide mechanisms to specify more expressive policies to control its sharing. The users can create policies by using Policy Editor Interface.

Context-aware systems have been studied for a long time, though the focus has been mainly on the location and activity inference. The research project MyCampus [31] presented a mobile application that involved a collection of customizable agents capable of semi-automatically discovering and accessing Intranet and Internet services during the process of assisting their users in carrying out various tasks. During the past decade a body of work on rule-based policy frameworks and access control systems has emerged. Rei [9] is a policy language designed for pervasive computing applications. It has been used to build a security framework that addresses the issues of security for web resources, agents and services in Semantic Web. Rein (Rei and N3) [8] is a distributed framework for describing and reasoning over policies in Semantic Web. It supports N3 rules for representing interconnections between policies and resources. Taintdroid [32] uses a taint tracking mechanism to detect sensitive data flow inside the device. CRePE [33] is the first to introduce a policy based Android extension but the user context model and the user level CRePE assumes, is trivial when one considers the extent of granularity of user context and user role possible in real life circumstances.

V. Future Work and Conclusion

As mobile devices become the dominant communication and information access medium these devices will model our interests, activities and behavior. When appropriate, aspects of this learned information which includes context may be shared with other devices in order to collaborate and provide enhanced service. This development introduces a need for a stronger flow control. However, leaving this control solely in hand of users might not be safe. Eventually we will have to think of systems that are capable of making suggestions to users about how to protect their privacy and security. Such systems could provide understandable policies to users or even enforce generic corporate or certified policies from trusted authorities. One way of going forward is to work on logic based and learning based systems which are capable of determining security vulnerabilities on devices and make such suggestions.

Another problem lies on increasing the computing ability of mobile devices. Executing complex inference mechanisms or generating context models can be a compute intensive task. We need to make sure these tasks are efficient in order to ensure it does not consume all the resources on an already limited-resource device. Therefore, we need to work on algorithms to ensure efficient executions of privacy and security rules which consume less resources for generating context.

References

[1] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *First Int. symposium on Handheld and Ubiquitous Computing (HUC)*, 1999.

[2] R. Sandhu and P. Samarati, "Authentication, access control, and audit," *ACM Comput. Surv.*, vol. 28, no. 1, pp. 241–243, Mar. 1996. [Online]. Available: http://doi.acm.org/10.1145/234313.234412

[3] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding attributes to role-based access control," *Computer*, vol. 43, no. 6, pp. 79–81, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1109/MC.2010.155

[4] S. Godik, A. Anderson, B. Parducci, P. Humenn, and S. Vajjhala, "Oasis extensible access control 2 markup language (xacml) 3," Tech. rep., OASIS, Tech. Rep., 2002.

[5] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, Sept 2010.

[6] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," in *Proceedings of communication networks and distributed systems modeling and simulation conference*, vol. 2004, 2004, pp. 270–275.

[7] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *The knowledge engineering review*, vol. 18, no. 03, pp. 197–207, 2003.

[8] L. Kagal and T. Berners-Lee, "Rein: Where policies meet rules in the semantic web," *Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA*, vol. 2139, 2005.

[9] L. Kagal, T. Finin, and A. Joshi, "A policy based approach to security for the semantic web," in *International Semantic Web Conference*. Springer Berlin Heidelberg, 2003, pp. 402–418.

[10] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, "Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*. IEEE, 2003, pp. 93–96.

[11] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, ser. POLICY '01. London, UK, UK: Springer-Verlag, 2001, pp. 18–38. [Online]. Available: http://dl.acm.org/citation.cfm?id=646962.712108

[12] O. Fonseca, "Byod leads to data breaches in the workplace," Nov. 2012. [Online]. Available: https://github.com/lencinhaus/androjena

[13] L. Zavala, R. Dharurkar, P. Jagtap, T. Finin, and A. Joshi, "Mobile, collaborative, context-aware systems," in *Proc. AAAI Workshop on Activity Context Representation: Techniques and Languages, AAAI. AAAI Press*, 2011.

[14] F. Baader and U. Sattler, "Description logics with aggregates and concrete domains," *Information Systems*, vol. 28, no. 8, pp. 979–1004, 2003.

[15] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein, "Owl web ontology language reference. w3c," 2004.

[16] D. J. Weitzner, J. Hendler, T. Berners-Lee, and D. Connolly, "Creating a policy-aware web: Discretionary, rule-based access," *Web and Information Security*, vol. 1, 2006.

[17] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004, pp. 74–83.

[18] Lorecarra, "Androjena: Jena android porting," 2009. [Online]. Available: http://www.experian.com/blogs/data-breach/2012/05/02/medical-and-mobile-convenience-trumps-security/

[19] M. Lindorfer, S. Volanis, A. Sisto, M. Neugschwandtner, E. Athanasopoulos, F. Maggi, C. Platzer, S. Zanero, and S. Ioannidis, "Andradar: fast discovery of android applications in alternative markets," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014, pp. 51–71.

[20] A. Patwardhan, V. Korolev, L. Kagal, and A. Joshi, "Enforcing policies in pervasive environments," in *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*. IEEE, 2004, pp. 299–308.

[21] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, Oct. 2002. [Online]. Available: http://dx.doi.org/10.1142/S0218488502001648

[22] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, "Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings," in *Symposium On Usable Privacy and Security (SOUPS 2014)*, 2014, pp. 199–212.

[23] J. Lin, "Understanding and capturing people's mobile app privacy preferences," Ph.D. dissertation, Pittsburgh, PA, USA, 2013, aAI3577905.

[24] A. Westin, "Privacy and freedom. 1967," *Atheneum, New York*, 1970.

[25] L. Kagal and H. Abelson, "Access control is an inadequate framework for privacy protection," in *W3C Privacy Workshop*, 2010, pp. 1–6.

[26] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman, "Information accountability," *Communications of the ACM*, vol. 51, no. 6, pp. 82–87, 2008.

[27] D. Ghosh, "Context based privacy and security in smartphones," *Master's thesis, University of Maryland, Baltimore County*, 2012.

[28] D. Ghosh, A. Joshi, T. Finin, and P. Jagtap, "Privacy control in smart phones using semantically rich reasoning and context modeling," in *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*. IEEE, 2012, pp. 82–85.

[29] P. Jagtap, A. Joshi, T. Finin, and L. Zavala, "Preserving privacy in context-aware systems," in *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*. IEEE, 2011, pp. 149–153.

[30] ——, "Privacy preservation in context aware geosocial networking applications," Department of Computer Science and Electrical Engineering, Tech. Rep., 2011.

[31] N. M. Sadeh, T.-C. Chan, L. Van, O. Kwon, and K. Takizawa, "A semantic web environment for context-aware m-commerce," in *Proceedings of the 4th ACM conference on Electronic commerce*. ACM, 2003, pp. 268–269.

[32] W. Enck, P. Gilbert, S. Han, V. Tendulkar, 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,” *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, p. 5, 2014.

[33] M. Conti, V. T. N. Nguyen, and B. Crispo, “Crepe: Context-related policy enforcement for android,” in *International Conference on Information Security*. Springer, 2010, pp. 331–345.