Security Policy Model for Ubiquitous Social Systems

Vladimir Jovanovikj[1,2](✉), Dušan Gabrijelčič[1], and Tomaž Klobučar[1]

[1] Laboratory for Open Systems and Networks, Jožef Stefan Institute, Ljubljana, Slovenia
vladimir@e5.ijs.si
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

Abstract. Ubiquitous social systems encompass ubiquitous computing, enterprise mobility and consumerization of IT, amplifying the threats associated to these fields. Context-aware security systems have been proposed as solutions for many of these threats. We argue that policy models used by these systems are not suitable for ubiquitous social systems. They lack of sufficient abstractions for specification and analysis of security policies and unnecessarily burden them with context reasoning rules. This can compromise the correctness of security policies and the performance of security systems. To address these issues, we propose a security policy model for ubiquitous social systems. The model defines all possible contextual information as policy abstractions, enabling clear and precise analysis of how they influence access control. Moreover, it takes into account the social related aspect and introduces an object life cycle. As a result, our model provides more intuitive abstractions and facilitates policy specification and context-aware security provisioning.

Keywords: Security · Context · Security policy · Ubiquitous computing · Consumerization of IT · Mobile devices

1 Introduction

Ubiquitous social systems are environments which encompass ubiquitous computing and the trends of enterprise mobility and consumerization of IT [8]. In such environments, a user uses her own mobile device for performing activities within both domains of professional and personal life, usually simultaneously, with the same consumer-oriented applications and in surroundings different than those in which they are natively performed. This blurs the boundaries between these domains, meaning the activities cannot be easily separated and secured. As a result, the threats associated to the encompassed fields are amplified, which makes ubiquitous social systems attractive targets for attacks.

Many researchers have proposed context-aware security systems as possible solutions for ubiquitous social systems (e.g., [1,3–5,9,10,12]). However, they provide limited solutions, as they deal only with targeted use cases (e.g., smart home [4], smart hospital [3,5,9], or mobile workers [1,12]). In particular, they

DOI: 10.1007/978-3-319-57837-8_24

lack of appropriate security policy models, which are able to abstract the characteristics details of security policies including security context. The policy models used by these systems are not applicable mainly due to two reasons. First, they do not provide sufficient abstractions and lack of comprehensive analysis about how security context influences access control. Thus, it is unclear how existing security systems adapt their behavior upon context changes. Second, they use an arbitrary level of abstraction of contextual information in security policies. Researchers have pointed out that it is more sophisticated for context-aware systems to adapt to high-level contextual information [2,6], since they meaningfully interpret raw sensor data through the process of context reasoning. Several policy models (e.g., [5,12]) support such contextual information, but they unnecessarily burden policies with reasoning rules. As a result, this can compromise the correctness of security policies and the performance of security systems.

In this paper, we propose a security policy model for ubiquitous social systems. Our model represents a range of applicable security policies and enables their analysis. It gives security context a primary role in decision making and defines all possible contextual information as policy abstractions, based on our conceptual model of security context presented in [8]. This enables more clear and precise analysis of how important contextual information influence access control, as the underlying security service. Our policy model provides intuitive abstractions for ubiquitous social systems, as it takes into account the social related aspect, and introduces a novel state diagram to represent the life cycle of objects. The latter makes provision of context-aware security behaviour easier, in particular detection of context changes and specification of reactions upon these changes. As a whole, our policy model facilitates specification and analysis of security policies applicable for ubiquitous social systems.

The paper is structured as follows. First, we describe the policy models used by existing context-aware security systems in Sect. 2. Next, we shortly present our conceptual model of security context as background in Sect. 3. Afterwards, we describe a motivation scenario to illustrate ubiquitous social systems in Sect. 4. Then, we present our policy model for ubiquitous social systems in Sect. 5. This is followed by an evaluation of the applicability of the model for describing our motivation scenario in Sect. 6. Finally, we conclude the paper in Sect. 7.

2 Related Work

Several works have tried to extend traditional policy models to include security context. Most of them are based on Role-based access control (RBAC) model [7], extending it mainly in two directions. First, RBAC's policy abstractions and assignments are constrained with contextual conditions (e.g., roles in [4], objects in [9], both rights and roles in [3], and subject-role and role-right assignment in [9]), meaning they are active only when these conditions hold, w.r.t. the current security context. Second, RBAC is extended with additional abstractions defined through contextual conditions. For example, [5] introduces *context* to express various security contexts over subjects, objects and actions, which complement RBAC rules, essentially specifying that a subject can perform an action over an object only if the conditions hold w.r.t. the current security context. Usage control (UCON) model [11] has also been extended in [1], with an additional abstraction *context*, similar to the UCON's attribute. Context is used for specification of conditions that are evaluated prior granting rights or during their usage. These constraints denote that a subject can be granted a right, or it can use this right, as long as the specified conditions hold, w.r.t. the current security context. Otherwise, the right is not given, or needs to be revoked.

Few works defined policy models particularly for context-aware security. [12] defines context for an operation performed over an object, also through contextual conditions. If the conditions hold, a context is active and the subject is allowed to perform the operation over the object. As context changes, so does active contexts and allowed operations over objects. Furthermore, [10] represents security policies graphically, as so called contextual graphs. A node in such graph denotes a contextual condition, which needs to be checked w.r.t. the current security context, or a security action, which needs to be performed when reached. Policy evaluation, in this case, is a path through the graph that determines what security actions need to be performed w.r.t. the current security context.

Existing policy models for context-aware security mostly consider security context as an add-on and rarely model reactions upon context change. Moreover, they mainly take low-level contextual information into account, which change frequently, leading to poor performance of security systems and even towards denial of service attacks. Some models (e.g., [5,12]) support high-level contextual information, but place the reasoning rules in security policies. This requires policy makers to be extensively familiar with context reasoning, which can influence the correctness of the policies. It can even compromise the performance of security systems as context reasoning will be performed during each decision.

3 Background

We shortly describe security context according to our conceptual model. For detailed description, we refer the reader to [8].

Security context is any information that can be used to characterize the situation of an entity (person or device, which can be also seen as a composition of its resources), considered relevant for security, regarding a particular task or activity. *Activity* is a process of an entity executing an operation over a resource, while trying to accomplish certain goals. Activities can be performed over three types of resources: data—a persistent storage of information, channel—a pathway for exchanging information, and application—a software component for achieving functionalities over information. More precisely, a security context of an activity a is a tuple, $ctx_a = (a, focus, assoc_a, sett_a, sprop_a)$, consisting of the activity itself, the focus of the user, $focus$, the social concepts that characterize a—association, $assoc_a$, and setting, $sett_a$—and the values of the security properties[1] of resources currently involved in a. We describe focus, association and setting through *social groups*, which are communities where people participate in order to achieve their goals more easily. Within them, people perform different activities, according to their roles. Social groups can be categorized into social domains depending on the goals and roles they support[2]. Focus is a social group, which describes the user's motivation while performing an activity, and whose goals are considered primary for the user at a given moment. Association and setting are social groups common for all participants and observers[3] of an activity, respectively, which describe them appropriately.

4 Motivation

We present a typical ubiquitous social system scenario. It includes use cases encountered in common practice, which represent the characteristics of IT consumerization, and take into account standardized security guidelines. Our use case centers around Alice—a finance manager at ACME1, which is a traditional company, trying to follow modern technological trends, by allowing its employees to use their own mobile devices (e.g., phone or tablet) for work. It is still conservative in this regard, allowing them to use their devices exclusively for work purposes while at work. Thus, employees can install various apps on their devices, but they can use only apps approved by the company policy for work. For example, intelligent assistant apps are forbidden, as they send audio recordings to remote servers and can potentially leak confidential company conversations. But, Alice's favorite word processing app is allowed, as ACME1 approves it.

Scene 1. Alice is in her office, preparing a report on her phone with the *Office* app. The app collects usage statistics and tries to send them to a remote server, but this should not be allowed, as it might also leak the confidential report. Also, Alice should not be able to check a personal document, since ACME1 allows her to open only work documents while using the *Office* app for work purposes.

Scene 2. Alice wants to call home, but ACME1 does not allow her to use her phone for other than work purposes while at the office. So, she decides to leave company premises. As she logs out on the access control device with her phone, all apps she has been using for work are automatically closed.

Scene 3. Alice enters her apartment, after unlocking the door with her phone. She wants to review the report she made earlier, but is not able to open it. According to the company policy, employees can access work documents only inside company premises.

[1] Security property is a quality that describes a resource or its usage, with respect to the objectives of security, i.e., confidentiality, integrity and availability.

[2] Examples of social domains are family or research department, whereas examples of social groups are their particular realization.

[3] Participant is an entity involved in an activity by consuming or generating resources, whereas observer is an entity that can monitor an activity which is being performed.

5 Context-Aware Activity Control Model

We propose a security policy model for ubiquitous social systems. Our model enables each social group to protect its own assets by controlling their interactions with other assets during activities[4]. It is based on security context and emphasizes activity instead of access as a single unit of control. (Access is more related with the operation inside an activity and does not leave an impression of a concept from which security context rises. Also, access suggests only a single interaction between the involved assets (i.e., subject vs. object), whereas we advocate several possible ones.) Thus, we refer to our model as context-aware activity control (CA-ACON) model. Below, first we define its policy abstractions in Sect. 5.1, and then we define its policy rules in Sect. 5.2.

5.1 Policy Abstractions

Policy abstractions are placeholders in security policies, which are replaced by particular elements from the system state during their evaluation. Our policy model consists of the following abstractions: activity, subject, object, origin, security context of activity, continuous activities and current security contexts. Figure 1 illustrates these policy abstractions and the relations between them[5].

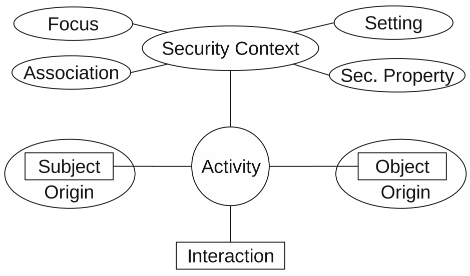

Fig. 1. Policy abstractions of the model

Activity. We define *activity* same as in [8] (see Sect. 3). But, we deal only with simple activities and treat complex activities as sequences of simple ones, which need to be controlled separately. We denote the set of activities as A.

Subject and Object. *Subject* is a resource which performs an activity by carrying out an operation, whereas *object* is a resource over which an activity is performed. A resource can reside in two states: active and passive. A resource in

[4] Asset is a resource assigned to a social group, or an entity defined as its member. Entities are assets as they are responsible for achieving the social group goals.

[5] Since continuous activities and current security contexts are activities and security contexts, respectively, they are represented through these abstractions on the figure.

passive state is used for storing information and functionalities in the device's memory for future use. It is kept like a closed container whose contents are not accessible directly. Contrary, a resource in active state is like an open container, whose information can be read and modified, in case of data and channel, or whose functionality can be executed, in case of application. For example, in Linux a passive application is an executable file, a passive data/channel is a regular file, an active application is a process, and an active data/channel is a file descriptor. If we take resource states into account, a subject is an application in active state, whereas an object is data, channel or application, either in passive or active state. It follows any resource is an object and that applications can be both subjects and objects, i.e., subjects are also objects. We denote the set of all subjects and objects as S and O, and we denote the sets of passive and active objects with O_P and $O_A, O = O_P + O_A$.

Object Life Cycle. Seven types of activities can be performed over objects, depending on the operation carried out during the activity. They are: *create*, *open*, *execute*, *read*, *write*, *close* and *destroy* activity. Each of them is performed over a particular object state and has a particular effect on the object, which results in state transitions. The possible activity types, together with the possible object states and state transitions, compose the life cycle of an object (Fig. 2).

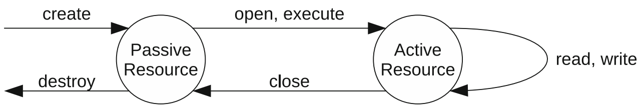

Fig. 2. Object life cycle

We describe object life cycle as follows. Each object o begins its life cycle when certain subject s performs a *create* activity, $(s, create, o)$, which adds a new element in $O_P, O_P = O_P + \{o\}$. In order to be directly used, o must be activated. Data and channels are activated with the *open* activity, whereas applications are activate with the *execute* activity[6]. Both $(s, open, o)$, and $(s, execute, o)$ move o from O_P to $O_A, O_P = O_P - \{o\}, O_A = O_A + \{o\}$. While in active state, a data or channel can be read or written numerous times, without changing its state. During a *read* activity, $(s, read, o)$, information from o is transferred into s, whereas during a *write* activity, $(s, write, o)$, information from s is transferred into o, modifying it. On the other hand, while in active state an application becomes capable of providing its functionality, i.e., it becomes a subject, $S = S + \{o\}$, and can perform activities over other objects. An object o stays in an active state as long as it is not deactivated with a *close* activity, $(s, close, o)$, which moves o back to $O_P, O_A = O_A - \{o\}, O_P = O_P + \{o\}$. It also removes o from $S, S = S - \{o\}$, in case o is an application. The transitions between the

[6] Since activating applications has different security implications than activating data and channels, we use two activity types in order to control them separately.

passive and active object states can be performed many times during its life cycle. Finally, a *destroy* activity, $(s, destroy, o)$ destroys the object and removes it from the $O_P, O_P = O_P - \{o\}$.

Security Context of Activity. We define *security context of activity* same as in [8] (see Sect. 3). Security context can change as other activities are performed.

Origin. Subjects and objects can be classified according to their origin. *Origin* is a single social group a resource can be related with, which captures the purpose of its creation. For an application, origin captures the social group whose goals it is used for achieving, whereas for a data or a channel, origin captures the social group whose information it is used to contain. Resources inherit their origin from the subject which created them. We assume there is an application for each social group from which resources related with that social group inherit their origin, either directly or indirectly. We define origin as a function, $org : Res \rightarrow SG$, which maps a resource to a social group. (We denote the set of all social groups as SG.) The origins of all subjects and objects compose the set of all origins, Org, defined as a set of tuples $(o, org(o))$, where $o \in O$[7].

Continuous Activities. Each activity is performed when the subject carries out the operation of the activity over the object, after which, we assume that the effects of that activity over the involved assets are manifested. *Continuous activity* is an activity whose effect over the involved assets continues to last after it is performed. We consider as continuous only *open* and *execute* activities[8], because they move an object into active state, where its functionality becomes available to the user, in case of applications, or its information becomes available to the subject, in case of data and channel. For example, an open activity, $(s, open, o)$, makes a data object o available to the subject s for reading and writing, during number of activities in the future. This is different than a single read activity, $(s, read, o)$, during which the information is exchanged between the subject and the object only when it is performed, and not afterwords.

Current Security Contexts. We consider that the security context of a continuous activity can change after it is performed. As a result, it needs to be controlled after the activity is allowed, at each context change. In order to be able to detect context changes, we keep track of all continuous activities which are currently being performed, and we constantly monitor their security context. We define a function named *current security context*, $cctx : A \rightarrow Ctx$, which maps an activity to its security context at the given moment. We denote the set of all security contexts as Ctx. We define *current continuous activities* $A_C \subset A$, as a set of all continuous activities whose effect still lasts over some object. We further define *current security contexts* $CCtx$ as a set of tuples $(a, cctx(a))$ of the current security context of each continuous activity $a \in A_C$. The elements in $CCtx$ are constantly updated in order to contain relevant values.

[7] Since subjects are objects, this implies that Org also includes origins of subjects.

[8] *Destroy* activity as not continuous, because its effect over the involved object does not exist, as the object is destroyed after its execution.

Interaction. We define *interaction* as an ability or a mean for communication between two assets involved in an activity, which can have a security implication. During an activity, parallel interactions occur between all the involved assets. An asset can be involved in an activity as follows: a resource can act as subject or object, whereas an entity can act as user, participant or observer of the activity[9]. All interactions can be easily translated into interactions between social groups, by mapping the involved assets to their relating social groups: (i) the subject and object are related with the social group in their origin, (ii) the user is related with the social group in his focus, i.e., the focus of the activity, (iii) the participants are related with the social group in the association, and (iv) the observers are related with the social group in the setting of the activity. We denote the set of all interactions as Int. We consider only interactions involving the subject as relevant for security, that is, subject-object, subject-user, subject-participant and subject-observer interactions (Fig. 3). The reason for this is twofold. First, the subject performs activities and hence it drives each interaction. Any interaction involving the object either: (i) goes through the subject, for example in case of data, since the user/participants/observers cannot access data directly, or (ii) is negligible, for example in case of channel and application, since channels only serve as pathways for communication between subjects and the user/participants/observers, and the purpose of applications is to be used as subjects. Second, the subject is usually the only target over which security policies can be enforced. Any discussion of direct interactions between the user, the participants and the observers is futile, as it cannot be enforced.

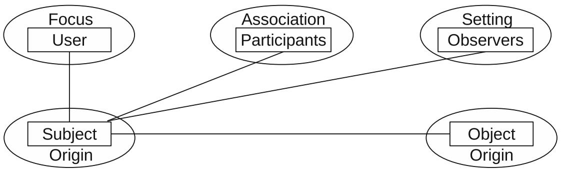

Fig. 3. Security relevant interactions between assets involved in an activity

5.2 Policy Rules

Policy rules are statements specified over policy abstractions, which must hold in order for the desired security objectives of the policy to be achieved. Our policy model recognizes two types of policy rules: group and default policy rules.

Group Policy Rules. These rules apply for assets related with a particular social group. They are specified in form of authorization and express the security requirements of a social group regarding particular interactions during an activity. We refer to a set of group policy rules, specific to a single social group, as

[9] One entity acts both as user and participant in an activity. In the former case, it initiates the activity, whereas in the latter it participates in it.

its group policy. A group policy is taken into account for each decision regarding an activity, which involves an asset related with that social group. Since they are specific for each social group, group policy rules need to be specified explicitly, with the help of a policy language.

Default Policy Rules. These rules apply for assets related with any social group. They are constant and can be implemented implicitly by the context-aware security system using our model. We define three default policy rules:

- *Basic rule:* An activity is allowed to be performed only if all interactions during it are allowed by the social groups whose assets are involved.
- *Contextual rule:* If a security context of a continuous activity changes, then the basic rule for this activity must be reevaluated. In case the activity is not allowed according to the basic rule, then the effect of this activity needs to be terminated, according to the termination rule. Otherwise, it remains in the set of continuous activities.
- *Termination rule:* The effects of a continuous activity are terminated by closing the object over which it is performed. In case there are other continuous activities involving the same object, they need to be terminated too.

6 Evaluation

We show the applicability of our policy model by using it to analyze two scenes from our motivation scenario from Sect. 4. We describe how a context-aware security system (CASS), running on Alice's mobile device, makes security decisions, based on the security policies specified with our model. The CASS monitors each resource and intercepts each activity request, upon which, examines the security context of the requested activity and decides whether to allow or deny it. For a continuous activity, the CASS continues to monitor its security contexts. If the context changes and becomes inappropriate, the CASS terminates the activity.

We make several assumptions prior the analysis. First, we assume Alice is a member of three social groups, referred[10] to as: work, home, and public[11]. We assume the CASS keeps information about the structure of these social groups (members and their roles) and has their security policies specified. For example, Alice's work social group, representing the ACME1 company, consists of all employees and equipment, including: Alice as finance manager, Adam as accountant, AP_1 as access point, R_1 as router, P_1 as printer, S_1 as server, etc.

Furthermore, we assume each social group has its own set of applications, which are used by the user while achieving its goals. If the user wants to use the functionality of an application in several social groups, it needs to be installed for each of them. We label applications according to the social group for which they are installed. For example, if Alice wants to use the *Launcher* app for starting

[10] Since people are often part of a single social group from a social domain, we refer to social groups by their domains, e.g., work is ACME1 and home is Alice's family.

[11] As explained in [8], the public social group is default and contains all entities.

apps in each work, home and public social group, it needs to be installed three times, as $Launcher_W$, $Launcher_H$, and $Launcher_P$, respectively.

Finally, we assume each social group has its own group policy, which states what contextual conditions need to be satisfied in order particular interactions during activities to be allowed for its assets. For example, one rule from Alice's work policy states that: "*Subject-user* interaction during *execute* activity is allowed, only if subject's origin and user's (activity's) focus are the *work* social group". From the aspect of work social group as subject's origin, this rule allows work subjects to execute objects only in work focus. On the other hand, from the aspect of work social group as user's focus, this rule allows only work subjects to execute objects while focus is work. Table 1 lists Alice's work, home and public policies, in particular, what kind of policy rules they consist of. We refer to a rule which allows only interactions between assets from the same social group, as closed rule. Consequently, we refer to a rule which allows interactions between asset from one social group and asset from any other social groups as open rule.

Table 1. Group policies applicable in our motivation scenario.

Policy	Interaction			
	Sub-obj	Sub-usr	Sub-par	Sub-obs
Work policy	Closed	Closed	Closed	Closed
Home policy	Closed	Closed	Closed	Open
Public policy	Open	Open	Open	Open

Scene 1. Alice is in her office and her focus is her work social group. This can be determined from her previous activity of logging on the electronic access control system installed at the entrance of ACME1 premises. She prepares the report with the $Office_W$ app as follows. First, $Office_W$ app opens the report doc_{W1}, as data object, $a_1 = (Office_W, open, doc_{W1})$. Then, it reads the report's content, $a_2 = (Office_W, read, doc_{W1})$ and shows it on the screen, through a screen channel, $scr_{W1}, a_3 = (Office_W, write, scr_{W1})$. $Office_W$ captures what Alice is typing through a channel to the touch sensor, $tch_{W1}, a_4 = (Office_W, read, tch_{W1})$, and writes that to the report, $a_5 = (Office_W, write, doc_{W1})$. $Office_W$ app submits a request for each of these activities, and the CASS makes a security decision for each request. We assume association and setting of a_1, a_2, and a_5 are empty as they happen inside the mobile device and have no participants and observers. On the other hand, association and setting of a_3 and a_4 are the work social group as there are only work group members around. We assume doc_{W1}, sch_{W1} and sch_{W1} has been created earlier by the $Office_W$, so their origin is the work social group. Since work policy is the only applicable for these decisions, all interactions during these activities are allowed according to it and the *basic rule*. After a_1 is performed, it is added in the set of continuous activities A_C and its security context is added in the set of current contexts, $CCtx$.

Furthermore, the decision making for the activity of opening a network channel, $ch_{W1}, a_6 = (\mathit{Office}_W, \mathit{open}, ch_{W1})$, goes as follows. The CASS obtains the following needed information for the decision. The focus of a_6 is the work social group same as above, whereas its association and setting are the public social group, as the remote server is unknown and by default member only of the public social group. We assume ch_{W1} has been created earlier by the Office_W, so its origin is the work social group. Thus, the CASS takes two *group policies* into account: work and public. The evaluation goes as follows: (i) subject-object interaction is allowed because the subject and the object originate from the same social group; (ii) subject-user interaction is allowed because the subject's origin is same as the (user's) focus; (iii) while subject-participants interaction is allowed from the aspect of public social group, because it allows any interaction, this interaction is not allowed from the aspect of work social group, as the association is different than the subject's origin. (iv) while subject-observer interaction is allowed from the aspect of public social group, because it allows any interaction, this interaction is not allowed from the aspect of work social group, as the setting is different than the subject's origin. As a result, the CASS makes a decision that a_6 is not allowed, according to the *basic rule*, i.e., Office_W app cannot open the ch_{W1} channel to communicate with the remote server.

Finally, Office_W app tries to open a home document $doc_{H1}, a_7 = (\mathit{Office}_W, \mathit{open}, doc_{H1})$. For this activity, the CASS takes into account the home policy, in addition to the work policy. According to these group policies, the subject-object interaction is not allowed during this activity, from aspect of both work and home social group, as the subject and object originate from different social groups. Thus, the CASS evaluates that a_7 is not allowed, according to the *basic rule* and prevents the Office_W from opening the data.

Scene 2. Alice is in her office and wants to call home, but is not able to use home applications at work. According to her work policy, subject-observer interactions are not allowed during such activities, as the subject's origin (home social group), is different than the setting of these activities (work social group, as only work group members around). Thus, she leaves the company premises.

On her way out, Alice logs out with her mobile device at the access control system, which changes her focus from work to public social group. This applies for all continuous activities in A_C, and for all future activities. The CASS detects these changes when $CCtx$ is updated with the new values[12], upon which it starts making new security decision for each activity in A_C. This is done according to the *contextual* and the *basic rules*. $CCtx$ consists of the execution activities of all subjects and the opening activities of the channel and data objects they use. One can imagine that, at this moment, this set contains the execution activities of the Office_W and the $\mathit{Launcher}_W$, and the opening activities of the screen and touch channels these apps use for communication with Alice. All these objects originate from the work social group. The CASS takes two *group policies* into account for

[12] The values of the association and setting will also change, since there are public group members around.

these decisions: public and work. Since these activities are not allowed according to the *basic rule*, they need to be closed according to the *termination rule*.

As soon as Alice is outside, she starts the $Launcher_H$ app, which changes her focus from public to home social group. Now, she is able to operate with home apps, because the setting of these activities is not her work social group anymore, but public. Both public and home policies allow the subject-observer interaction during these activities. Also, all other interactions are allowed according to the home policy, as it is the only one applicable for them.

7 Conclusion and Future Work

In this paper, we defined a security policy model for ubiquitous social systems. Our model centers around security context and defines all possible contextual information as policy abstractions. Since these information are few and of high level of abstraction, our model enables clear and precise analysis of how they influence access control. Furthermore, our policy abstractions are more intuitive for ubiquitous social systems, as they include social concepts and are specifically tailored for providing context-aware security behaviour. In addition, our model combines security policies specified by different policy makers, enabling each social group to specify its own security policy.

As future work, we plan to define a policy language for our model, based on Datalog, which will enable specification of group policies for ubiquitous social systems. We also plan to implement a prototype of context-aware security system, which will use our policy model and language for making security decisions.

References

1. Bai, G., Gu, L., Feng, T., Guo, Y., Chen, X.: Context-aware usage control for Android. In: Jajodia, S., Zhou, J. (eds.) SecureComm 2010. LNICSSITE, vol. 50, pp. 326–343. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16161-2_19
2. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
3. Bonatti, P., Galdi, C., Torres, D.: ERBAC: Event-driven RBAC. In: Proceedings of the ACM Symposium on Access Control Models and Technologies, pp. 125–136. ACM (2013)
4. Covington, M.J., Long, W., Srinivasan, S., Dev, A.K., Ahamad, M., Abowd, G.D.: Securing context-aware applications using environment roles. In: Proceedings of the ACM Symposium on Access Control Models and Technologies, SACMAT 2001, pp. 10–20. ACM (2001)
5. Cuppens, F., Cuppens-Boulahia, N.: Modeling contextual security policies. Int. J. Inf. Secur. **7**(4), 285–305 (2008)
6. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
7. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur. (TISSEC) **4**(3), 224–274 (2001)

8. Jovanovikj, V., Gabrijelčič, D., Klobučar, T.: A conceptual model of security context. Int. J. Inf. Secur. **13**(6), 571–581 (2014)
9. Kulkarni, D., Tripathi, A.: Context-aware role-based access control in pervasive computing systems. In: Proceedings of the ACM Symposium on Access Control Models and Technologies, SACMAT 2008, pp. 113–122. ACM (2008)
10. Mostefaoui, G.K.: Towards a conceptual and software framework for integrating context-based security in pervasive environments. Ph.D. thesis, University of Fribourg (2004)
11. Park, J., Sandhu, R.: The UCON ABC usage control model. ACM Trans. Inf. Syst. Secur. (TISSEC) **7**(1), 128–174 (2004)
12. Toninelli, A., Montanari, R., Kagal, L., Lassila, O.: Proteus: A semantic context-aware adaptive policy model. In: Proceedings of the IEEE International Symposium on Policies for Distributed Systems and Networks, POLICY 2007, pp. 129–140. IEEE Computer Society (2007)