Short Paper: A Framework for the Privacy Access Control Model

Sandugash Askarova[1(✉)], Darkhan Mukhatov[2], Altynbek Sharipbayev[1], and Dina Satybaldina[1]

[1] L.N. Gumilyov Eurasian National University, Astana 010000, Kazakhstan
sandugash.kz@gmail.com, sharalt@mail.ru, satybaldina_dzh@enu.kz
[2] "Zerde" Holding JSC, Astana 010000, Kazakhstan
dmukhatov@zerde.gov.kz

Abstract. Today privacy is a key issue when securing business processes. It has received increasing attention from consumers, companies, researchers and legislators. Organizations claim to have their own privacy policy as well as guarantee its proper enforcement. In this work we consider privacy features at the early stages of the systems development and specifically focus on modelling and analysis of the system requirements. A framework for modelling privacy access control policies was created through (*i*) defining access control policies that satisfy privacy requirements (*ii*) verification of designed privacy access control policy, and (*iii*) a set of heuristics for defining policy.

Keywords: Security management · Access control · Privacy policy modelling languages · Information systems

1 Introduction

Privacy is an increasingly important business concern in health care, financial services, and other organizations. Organizations that collect and use personal information face the growing challenge of conducting their business effectively while managing privacy risks and compliance requirements [1]. Organizations have adopted various strategies to protect personal data privacy. In particular, in the financial sector laws and regulations have been created to protect privacy data such as Basel II, Sarbanes-Oxley Act (SOX), Gramm-Leach-Bliley Act (GLBA) [2].

Access control policies are defined as rules, which regulate how users can access resources [3]. These access control policies are created based on models. The classical models are Mandatory Access Control, Discretionary Access Control and Role Based Access Control [4]. Access control models cannot enforce privacy policies and designed access control policies do not include privacy requirements such as purpose binding, conditions and obligations [5].

In order to enforce privacy policies in organizations, access control policies satisfying the privacy requirements in the requirements engineering should be formally identified [6]. Existing privacy policy languages such as P3P [7], APPEAL [9], E-P3P [8], EPAL [10] and XACML [4] do not completely solve privacy issues, and they are isolated from requirements analysis. As a result, defined privacy policies do not comply with system requirements. There are many frameworks for access control requirements modelling such as *i** framework [12], GBRAM and its extension [13], analytical role modelling framework (ARMF) [14], and Knowledge Acquisition in Automated Specification [15].

The focus of this research study is to model privacy requirements into access control policies. Our research is aimed at delivering a model of designing privacy aware systems by incorporating privacy requirements into access control policy. The framework for modelling privacy access control policies was created. This framework is developed by extending ARMF [14]. Sections 2 and 3 present our framework for modelling privacy access control policies and its heuristics for defining and verifying these policies. In Sect. 4 we summarize the findings by focusing on the aim and objectives of study and provide future work.

2 A Framework for Modelling Privacy Access Control Policies

The framework was developed based on ARMF [14], by adding the purpose meta-concept and corresponding relations. Purpose specification principle has been selected to investigate how access control policies can be defined and how it can be enforced during the requirements modelling. It also has been selected as the only principle stated in the Law of the Republic of Kazakhstan [6]. The framework uses notations of Z language [16].

Policies define restrictions to access valuable assets (privacy data). Such an access is required to carry out tasks. The tasks cannot be processed in a way that may be incompatible with the purposes for which the data have been collected. To include these notion into the framework the following meta-concepts are needed:

`[Asset]` – represents privacy data that we wish to protect;

`[Task]` – the activities that an organizational unit or individual carries out;

`[Purpose]` – personal data that shall be collected for specified, lawful and legitimate purpose or purposes and not processed in ways that are incompatible with the purposes for which data have been collected.

The meta-concepts of an agent and role are identified as follow:

`[Agent]` – represents a physical person;

`[Role]` – an assignment of an obligation, of performing some function, which is a composite element representing the organisational function, organisational domain, and authority. Three types of roles are defined according to organizational structures: roles based on seniority, roles based on function, and roles based on market. The meta-concepts are as following:

`[Authority]` – represents the seniority of a role;

`[Org_Function]` – a functional grouping within an organisation;

`[Org_Domain]` – represents a “market based” grouping i.e. a grouping that is delegated a market to serve such as a set of clients in a specific geographic location.

The meta-concept role is a composite of authority, organizational function, and organizational domain and is defined formally as follows:

Role $\triangleq$ [authority: Authority; org_function: Org_Function; org_domain: Org_Domain]

The inheritance between organizational functions is formally defined as follows: Inheritance_f $\triangleq$ {inhf: Org_Function $\nrightarrow$ Org_Function; org_function: $\mathbb{P}$ Org_Function | ($\forall$of: org_function $\bullet$ of $\notin$ inhf$^+$ (|{of}|))}.

The inheritance of roles is formally defined as follows: Inheritance_r $\triangleq$ {inhr: Role $\nrightarrow$ Role; role: $\mathbb{P}$ Role| ($\forall$r: role $\bullet$r $\notin$ inhr$^+$ (|{r}|)}.

The aggregation hierarchy for organizational domain is formally identified as follows:

Aggregation_d $\triangleq$ {aggd: Org_domain $\nrightarrow$ Org_domain; org_domain: $\mathbb{P}$ Org_domain| ($\forall$od: org_domain $\bullet$ od $\notin$ aggd$^+$ (|{od}|)}

Formally the task aggregation is defined as follows:

Aggregation_t $\triangleq$ {aggt: Task $\nrightarrow$ Task; task: $\mathbb{P}$ Task|($\forall$t: task $\bullet$ t $\notin$ aggt$^+$ (|{t}|))}.

Purpose aggregation is defined as follows:

Aggregation_p $\triangleq$ {aggp: Purpose $\nrightarrow$ Purpose; purpose: $\mathbb{P}$ Purpose| ($\forall$ p: purpose $\bullet$ p $\notin$ aggt$^+$ (|{p}|))}.

The relationship between tasks and purposes is represented by a task purpose dependency relation as follow:

Task_purpose_dependency $\triangleq$ {task_purpose_dependency: Task $\rightarrow$Purpose |($\forall$t: Task. $\exists_1$p: Purpose $\bullet$ (t, p) $\in$ task_purpose_dependency) $\wedge$ ($\forall$t: aggt$^+$(|{t}|). $\exists_1$p:aggp$^+$(|{p}|) $\bullet$ (t, p) $\in$ task_purpose_dependency)}.

The relation between purpose and asset or assets is represented by purpose asset dependency as follow:

Purpose_asset_dependency $\triangleq$ {purpose_asset_dependency: Purpose $\rightarrow\mathbb{P}$ Asset; purpose: $\mathbb{P}$ Purpose; asset: $\mathbb{P}$ Asset| ($\forall$p: purpose. $\exists$a: asset $\bullet$ (p, a) $\in$ purpose_asset_dependency)}.

The relationship between asset and organizational domain is formally identified as:

Asset_domain $\triangleq$ {asset_domain: Asset $\rightarrow$ Org_Domain| ($\forall$a: Asset. $\exists_1$od: Org_domain $\bullet$ (a, od) $\in$ asset_domain)}.

Policies will be defined using the following composite type:

Authorization_Policy $\triangleq$ {role: Role; task: Task}

There are implicit assumptions in this defined policy: firstly, the policy applies to any subtask if the task in the policy; secondly the organizational domain in the role of the policy applies to all assets associated with the task through the following relations:

Task_purpose_dependency $\triangleq$ {task_purpose_dependency: Task $\rightarrow$Purpose |($\forall$t: Task. $\exists_1$p: Purpose $\bullet$ (t, p) $\in$ task_purpose_dependency) $\wedge$ ($\forall$t: aggt$^+$(|{t}|). $\exists_1$p:aggp$^+$(|{p}|) $\bullet$ (t, p) $\in$ task_purpose_dependency)};

Purpose_asset_dependency $\triangleq$ {purpose_asset_dependency: Purpose →P Asset; purpose: P Purpose; asset: P Asset| (∀p: purpose. ∃a: asset • (p, a) ∈ purpose_asset_dependency)}.

3 Heuristics for Defining and Verifying Policies

Application of ARMF extensions is applied through six steps: (*i*) identifying organizational groups (*ii*) identifying level of authority (*iii*) defining roles (*iv*) identifying tasks, assets, purposes (*v*) defining policies, and (*vi*) verifying policies.

Once we have identified organizational functions, we need to show specialization hierarchy using the principle of inheritance as follow: ***Definition***: inhf: Org_Function ⇸ Org_Function; ***Constraint***: org_function: P Org_Function•(∀of: org_function • of ∉ $inhf^+$ (|{of}|)).

Similarly, once we have identified organizational domains, we need to show them in aggregation hierarchy as follow: ***Definition***: aggd: Org_domain⇸ Org_domain; ***Constraint***: org_domain: P Org_domain • (∀od: org_domain • od ∉ $aggd^+$ (|{od}|).

Levels of authority need to be assigned to groups. Once we have identified authority's levels we need to show their seniority as follow:

Definition: senior: Authority ⇸ Authority; ***Constraint***: authority: P Authority • (∀a: authority • a∉ $senior^+$(|{a}|)).

In next step we need to identify tasks and their associated purposes and then assets related to purposes in the organization: ***Definition:*** aggt: Task⇸ Task; ***Constraint:*** task: P Task • (∀t: task • t ∉ $aggt^+$ (|{t}|)).

Next, we need to identify purposes for defined tasks, which enable tasks to have access to asset or group of assets. Identified purposes are needed to be organized in hierarchical structure. It can be done by aggregation hierarchy as follow:

Definition: aggp: Purpose⇸ Purpose; ***Constraint:*** purpose: P Purpose • (∀ p: purpose • p ∉ $aggt^+$ (|{p}|)).

After that we need to show task purpose dependency as follow:

Definition: task_purpose_dependency: Task →Purpose: ***Constraint:*** (∀t: Task. $\exists_1$p: Purpose • (t, p) ∈ task_purpose_dependency) ∧ (∀t: $aggt^+$(|{t}|). $\exists_1$p:$aggp^+$(|{p}|) • (t, p) ∈ task_purpose_dependency).

The relation between purpose and asset or assets is represented by purpose asset dependency: ***Definition:*** purpose_asset_dependency: Purpose →P Asset; ***Constraint:*** no constraint.

The relationship between asset and organizational domain is formally identified as follow: ***Definition:*** asset_domain: Asset → Org_Domain; ***Constraint:*** no constraint.

Once we have identified organizational context, roles and tasks we now define policies as follow: ***Definition:*** Authorization_Policy $\triangleq$ {role: Role; task: Task}; ***Constraint:*** no constraint.

The final step is to verify policies through scenarios. In creation scenario the following domain concepts should be instantiated. Instantiation of domain:

Definition: insd: Org_Domain $\twoheadrightarrow$ Org_Domain; ***Constraint:*** org_domain: $\mathbb{P}$ Org_Domain $\bullet$ ($\forall$od$_1$; od$_2$: org_domain $\bullet$ od$_2 \in$ insd (| {od$_1$} |) $\Longrightarrow$ insd (| {od$_2$} |)=$\emptyset$); ***Constraint:*** $\forall$ od$_1$; od$_2$: Org_Domain $\bullet$ od$_1 \in$ aggd (| {od$_2$} |) $\Longrightarrow$ (insd (| {od$_1$} |)$\neq\emptyset$ $\wedge$ insd (| {od$_2$} |)$\neq\emptyset$) $\vee$ (insd (| {od$_1$} |)= $\emptyset$ $\wedge$ insd (| {od$_2$} |)= $\emptyset$).

Instantiation of role: ***Definition:*** insr: Role $\twoheadrightarrow$ Role; ***Constraint:*** role: $\mathbb{P}$ Role $\bullet$ ($\forall$r$_1$; r$_2$: role $\bullet$ r$_2 \in$ insr (| {r$_1$} |) $\Longrightarrow$ insr (| {r$_2$} |)=$\emptyset$); ***Constraint:*** $\nexists$ role: Role $\bullet$ (insr (| {role} |) = $\emptyset$ $\wedge$ insd (| {role.org_domain} |) $\neq\emptyset$) $\vee$ (insr (| {role} |) $\neq\emptyset$ $\wedge$ insd (| {role.org_domain} |) = $\emptyset$); ***Constraint:*** $\nexists$ policy: Authorization_Policy $\bullet$ insr(| {policy.role} |) $\neq\emptyset$.

Instantiation of task: ***Definition:*** inst: Task $\twoheadrightarrow$ Task; ***Constraint:*** task: $\mathbb{P}$ Task $\bullet$ ($\forall$t$_1$; t$_2$: task $\bullet$ t$_2 \in$ inst (| {t$_1$} |) $\Longrightarrow$ inst (| {t$_2$} |)=$\emptyset$)

Instantiation of purpose: ***Definition:*** insp: Purpose $\twoheadrightarrow$ Purpose; ***Constraint:*** purpose: $\mathbb{P}$ Purpose$\bullet$($\forall$p$_1$; p$_2$:purpose$\bullet$p$_2 \in$ insp(| {p$_1$} |)$\Longrightarrow$ insp (| {p$_2$} |)=$\emptyset$)

Instantiation of asset: ***Definition:*** insa: Asset $\twoheadrightarrow$ Asset; ***Constraint:*** asset: $\mathbb{P}$ Asset $\bullet$ ($\forall$a$_1$; a$_2$: asset $\bullet$ a$_2 \in$ insa (| {a$_1$} |) $\Longrightarrow$ insa (| {a$_2$} |)=$\emptyset$)

Instantiated roles are assigned to agents as follow: ***Definition:*** role_assignement: Agent $\leftrightarrow$ Role. ***Constraint:*** no constraint.

There is needed to model the carrying out of a task by an agent. This will be represented by relation performs, which defines an agent performing a task:

Definition: performs: Agent $\leftrightarrow$Task; ***Constraint:*** $\forall$ p: perform $\bullet$ $\forall$task: ran performs $\bullet$ inst(| {task} |) $\neq$ $\emptyset$. ***Constraint:*** $\forall$p: performs $\bullet$ $\forall$task : ran performs $\bullet\forall$ ins_purpose: task_purpose_dependency (| {task} |) $\bullet$ $\exists$ purpose: task_purpose_dependency (| inst(| {task} |) |) $\bullet$ purpose$\in$ insp(| {ins_purpose} |).

Constraint: $\forall$p: performs $\bullet$ $\forall$task : ran performs $\bullet\forall$ ins_asset: purpose_asset_dependency (| {purpose} |)$\bullet$ $\exists$ asset: purpose_asset_dependency (| insp(| {purpose} |) |) $\bullet$ asset$\in$ insa(| {ins_asset} |).

After identifying a scenario, we determine relation performs between specific agent and corresponding instantiated task. It can be done by using elimination rules and substituting instantiated elements, which was used in creating performs relation.

4 Conclusion

This paper has addressed the problem of modelling access policies in order to ensure that security goals can be achieved and that operational requirements are consistent with access policies. The framework includes a meta-model and a set of heuristics.

The meta-model represented a link between organizational context and privacy enforcement in order to capture the whole privacy domain. Heuristics were determined for defining policies and scenarios.

The limitation of this research is that proposed framework was created in Z language. This language requires a special knowledge in a set theory and mathematical logic. In addition, heuristics for defining and verifying policy were not illustrated by any example. The future research for this research study is implementation of the proposed framework in banks. In addition the future research can be done by considering other policies and validating them in the case studies.

References

1. Barth, A., Datta, A., Mitchell, J.C., Sundaram, S.: Privacy and utility in business processes. In: Proceedings of 20th IEEE Computer Security Foundations Symposium, pp. 279–294 (2007)
2. Anton, A.I., Earp, J.B., Potts, C., Alspaugh, T.A.: The role of policy and privacy values in requirements engineering. In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01), Toronto, Canada, pp.138–145 (2001)
3. Sandhu, R., Samarati, R.: Access control: principles and practice. IEEE Commun. Mag. **32**(9), 40–48 (1994)
4. Ni, Q., Trombetta, A., Bertino, E., Lobo, J.: Privacy-aware role based access control. In: Proceedings of SACMAT'07, Sophia Antipolis, France (2007)
5. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: Role-Based Access Control, 2nd edn. Artech House, London (2007)
6. Law of the Republic of Kazakhstan "On informatization", Astana (2007)
7. Lu, C.: Powerful Privacy Potential: P3P in the Context of Legislation and Education (2003)
8. Stufflebeam, W., Antón, A.I., He, Q., Jain, N.: Specifying privacy policies with P3P and EPAL: lessons learned. In: Proceedings of the Workshop on Privacy in the Electronic Society, Washington (2004)
9. Anton, A.I., Earp, B., Bolchini, D., He, Q., Jensen, C., Stufflebeam, W.: The lack of clarity in financial privacy policies and the need for standardization. IEEE Secur. Priv. **2**(2), 36–45 (2003)
10. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES'02), Washington (2002)
11. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practices: privacy-enabled management of customer data. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 69–84. Springer, Heidelberg (2003)
12. Liu, L., Yu, E.S.K., Mylopoulos, J.: Security and privacy requirements analysis within a social setting. In: Proceedings of 11th IEEE International Conference on Requirements Engineering (RE'03), Monterrey, USA, pp. 151–61 (2003)
13. He, Q., Anton, A.I.: A framework for modelling privacy requirements in role engineering. In: Proceedings of 9th International Workshop on Requirements Engineering – Foundation for Software Quality (REFSQ'03), pp. 137–146, Klagenfurt/Velden, Austria (2003)
14. Crook, R., Ince, D., Nuseibeh, B.: On modelling access policies: relating roles to their organisational context. In: Proceedings of 13th IEEE International Requirements Engineering Conference (RE'05), Paris, France (2005)

15. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, England (2009)
16. ISO/IEC 13568:2002. Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics (2002)