Obligations in Risk-Aware Access Control

Liang Chen*, Jason Crampton†, Martin J. Kollingbaum* and Timothy J. Norman*
*dot.rural Digital Economy Hub, University of Aberdeen
†Information Security Group, Royal Holloway, University of London

***Abstract*—The increasing need to share information in dynamic environments has created a requirement for risk-aware access control systems. In this paper, we present a metamodel for risk-aware authorization that captures the key aspects of a system in relation to risk mitigation. In particular, we develop various risk-aware models as instances of the metamodel that broadly differ in the form of risk mitigation that is used (system obligations and user obligations respectively), and study how those obligations are applied to reduce and account for the risk incurred by granting access. Unlike system obligations, an access control system cannot guarantee that user obligations are fulfilled. We propose two approaches to defining risk-aware authorization semantics that takes unfulfilled obligations into account: one is to restrict users' future access because of prior failure to fulfill obligations, and the other is to "reward" users who have been diligent in fulfilling their obligations by permitting risky access requests.**

I. Introduction

One of the fundamental security services in computer systems is *access control*, a mechanism for constraining the interaction between (authenticated) users and protected resources. Generally, access control is implemented by an authorization service that includes an *authorization decision function* which determines whether a user request to access a resource (an *access request*) should be permitted or not. In its simplest form an authorization decision function either returns an allow or a deny decision.

Risk-aware access control (RAAC) [1]–[4] is a novel access control paradigm that was proposed to meet the increasing need to share information in the context of coalitions and collaborations. The core idea of RAAC is to provide an authorization decision function that is able to make decisions based on risk: how much risk is incurred by allowing access (which runs the risk of information disclosure or modification) or denying access (which runs the risk of preventing business objectives from being realized). In other words, access requests are subject to a risk analysis before an authorization decision is made.

Risk-aware access control is considered to be more permissive than traditional access control mechanisms (static binary decisions), in the sense that some risky or exceptional access is allowed, provided that the risk of allowing such access can be accounted for and is not unacceptably high. It is usually implemented by applying *risk mitigation methods* with risky access in order to make the requested user less likely to misuse her granted permission. Clearly, this approach provides many advantages over the previous "ad hoc" solutions for dealing with exceptional access where the associated risk is hard to account for.

More specifically, we believe that most existing work in RAAC attempts to answer two fundamental questions: (i) how to define the risk of granting access; and (ii) how to use risk mitigation methods to support rich types of access control decisions.

In general, the risk of granting a request to perform some permission can be determined by the cost that will be incurred if the permission is authorized and subsequently misused, and the likelihood of the permission being misused. There has been significant research on how to define the cost and the likelihood of permissions being misused in the context of multi-level security [2], [5], [6] and role-based access control [3]. This suggests that the way of defining risk depends on a specific access control model to be studied.

Complex authorization decisions may be used to implement risk mitigation [2], [3]. An upper risk threshold is defined, above which all accesses are denied, together with a lower boundary, below which all accesses are allowed. Between these boundaries, an access request is allowed only if a risk mitigation method can be applied. These risk mitigation methods adopted in prior work use system actions whose execution can be guaranteed, and thus can effectively account for and reduce the risk associated with granting access.

Alternatively, we may wish to impose risk mitigation actions (or obligations) on users, meaning that the user must perform certain actions in the future in order to allow a risky action to be performed now. In general, obligations assigned to individual users are not enforceable: the system cannot force a user to fulfill her obligations. As such, the imposition of user obligations may give rise to a system being unable to account for the risk incurred by granting access. The question of what actions a risk-aware system should conduct to compensate for failures of users to discharge their obligations, as far as we are aware, has not been adequately studied and will be the focus of this paper. More specifically, the contributions of this paper are as follows.

- We abstract common system components that are relevant to risk assessment and risk mitigation, and propose a formal metamodel to accommodate rich, risk-aware authorization semantics. In particular, we can define the authorization semantics with respect to the types of mitigation methods used and the way in which risk is represented and accounted for.
- We instantiate the above metamodel with system mitigation methods and an explicit authorization decision function. Compared with existing work [3], our resulting model is more flexible in terms of supporting richer types

of access control decisions.

- We propose an "optimistic" approach to defining risk-aware models with consideration of unfulfilled obligations. We assume that users are always diligent in fulfilling obligations to mitigate the risk incurred by their granted access. However, once a user has failed to fulfill an obligation, the system would be able to identify the user, and restrict her future access. We suggest two mechanisms that provide different degrees of restriction on users' future access. The first mechanism would eventually prevent a misbehaving user from accessing any resources. The second one is less restrictive, only preventing a user from re-invoking a permission for which previously incurred obligations remain unfulfilled.
- We further propose a "conservative" approach to defining risk-aware authorization semantics that is also concerned with the obligations that users are required to fulfill. The basic idea of this approach is that a user will be rewarded for fulfilling obligations by having subsequent risky access requests granted. As for the pessimistic approach, we study two methods that reduce the risk associated with user requests when existing obligations are fulfilled.

The rest of the paper is organized as follows. In the next section we present some essential features of obligations that would be included in our risk-aware models. Section III presents a formal metamodel for RAAC systems from the perspective of risk mitigation and risk-aware authorization. In Section IV we define a concrete risk-aware model as the first instantiation of the metamodel that considers the use of system mitigation methods. In Section V we define the construction of user obligations and obligation constraints, and propose two methods to define risk-aware authorization semantics that take unfulfilled obligations into account. In Section VI we define risk-aware models that take a conservative approach to dealing with user obligations. We compare our work with closely related work in Section VII, and conclude the paper with some suggestions for future work in Section VIII.

II. Properties of Obligations

In recent years, security policies have been proposed to support obligation requirements [7]–[9], which specify certain actions have to be taken by a subject as a condition of acquiring certain permissions. These obligation requirements are often classified as *pre-obligations* (also known as *provisions*) and *post-obligations*, where the former refer to some actions that must be executed prior to making an authorization decision, and the latter express some future actions that must be fulfilled after an authorization decision is made. In this paper, we focus on post-obligations, and hereafter we refer to post-obligations simply as obligations.

A characteristic property of an obligation is a *time window* within which the obligation must be fulfilled. It is also important to distinguish *user obligations* from *system obligations*, as they have different features in terms of enforcement. Generally, the access control system cannot guarantee the fulfillment of a user obligation directly, in contrast to a system obligation.[1] For example, if a user is allowed to use a certain service, then she is obliged to pay the service fee by the end of each month, but the system cannot force her to pay the fee on time. Certainly, what the system can do is to stop her using the service if she failed to pay the fee before the deadline. This example suggests that, from the system's perspective, the system should at least ensure that users have necessary privileges to perform their obliged actions (in this case that users are allowed to pay their service charges).

From the example above, although a system cannot force a user obligation to be fulfilled, it should have some mechanism in place to *punish* a user who has failed to fulfill an obligation with which she has been charged. Alternatively, the system could *reward* users for being diligent in fulfilling their obligations on time. We believe that it is important to introduce these incentives for users to fulfill their obligations in a system where the violation of obligations may have profound consequences.

Obligations arise in a system when an authorization decision is made in response to a user's access request. The subject who is charged with an obligation does not necessarily have to be the user who makes the request. This means that a user's action can cause the system or someone else receiving some obligation(s). For example, a policy may specify that when a user is allowed to create a purchase order, this gives someone else an obligation to review the order and create a payment for it. This requirement is usually called a *separation of duty* constraint. In this paper, we intend to consider user obligations as risk mitigation methods, and therefore we focus on the situation where a user who takes a risky access causes himself to be assigned an obligation.

III. A Metamodel for RAAC

In this section we present a formal metamodel defining the basic structure of a system that supports risk assessment and risk mitigation. In particular, this model captures the relationships between risk mitigation methods and authorization decision-making. We then later develop some concrete models that instantiate one or more features of the model.

Formally, an RAAC system consists of the following components.

- U: a set of users.
- P: a set of permissions. These permissions represent the set of action-object pairs for which a user may be authorized. The term permission is often used interchangeably with *access right* and *privilege* in the literature.[2]
- Q: a set of access requests. We model an access request as a pair (u, p), where $u \in U$ and $p \in P$.

[1]In XACML [10], for example, (system) obligations are discharged by the policy enforcement point.

[2]It should be noted that, however, in general the concepts of permission and right are not synonymous; the existence of a right implies that the system or other users are prohibited from preventing the user exercising this right, whereas permission does not [11], [12].

- $\mathcal{M}$: a set of risk mitigation methods. A *risk mitigation method* $m \in \mathcal{M}$ could be some action that is required to be performed by either a system or a user.[3]
- $\mathcal{L} = \{l \in \mathbb{R} : 0 \leqslant l \leqslant 1\}$: a risk domain. We write $[l, l')$ to denote the risk interval $\{x \in \mathcal{L} : l \leqslant x < l'\}$. We define a *risk mitigation strategy* to be a list $[(l_0, M_0), (l_1, M_1), \ldots, (l_{n-1}, M_{n-1}), (l_n, M_n)]$, where $0 = l_0 < l_1 < \cdots < l_n \leqslant 1$, and $M_i \subseteq \mathcal{M}$.
- π: a function that associates each permission p with a risk mitigation strategy. This approach provides a fine-grained way of managing risk thresholds and mitigation methods for use in individual permissions.
- Σ: a set of states. An RAAC model typically defines a collection of sets, functions and relations that are usually referred as *components* of the model. The *state* of a model can be thought of as a tuple defined by those components. We define a state $\sigma \in \Sigma$ as (U, P, π, τ), where τ is a fully abstract representation of all other features of the state. In the case of the competence- and role-based access control (RBAC$_C$) model [3], for example, τ comprises the set of roles R, the user-role assignment relation $UA \subseteq U \times R$, the permission-role assignment relation $PA \subseteq P \times R$, the role hierarchy $RH \subseteq R \times R$, and the user-role competence function $\beta : U \times R \to (0, 1]$.
- $risk : Q \times \Sigma \to [0, 1]$: a risk function that takes as input an access request $q \in Q$ and the current system state $\sigma \in \Sigma$, and returns the risk $l \in [0, 1]$ associated with the request. As we mentioned above, an explicit specification of the $risk$ function depends on a specific access control model at hand (and is not the concern of this paper).
- $Auth : Q \times \Sigma \to D \times 2^{\mathcal{M}}$: an authorization decision function that takes the access request $q \in Q$, and the current system state $\sigma \in \Sigma$ as input and outputs an authorization decision $d \in D$ with a set of mitigation methods $M \subseteq \mathcal{M}$. We assume that the decisions that arise from the authorization decision function are given by the set $D = \{\mathsf{allow}, \mathsf{deny}\}$. More specifically, given an access request $q = (u, p)$, a risk mitigation strategy $\pi(p) = [(l_0, M_0), \ldots, (l_n, M_n)]$ for p, and the current state σ, we define

$$Auth(q, \sigma) = \begin{cases} (d_i, M_i) & \text{if } risk(q, \sigma) \in [l_i, l_{i+1}),\ i < n, \\ (d_n, M_n) & \text{otherwise,} \end{cases}$$

where $d_i \in D$. In other words, if the risk associated with access request (u, p) is l, then $Auth$ returns an authorization decision, and a set of mitigations corresponding to the interval containing l. Indeed, the above authorization decision function abstracts the authorization policy of a system. The abstraction of the policy means $Auth$ can represent various authorization semantics for making risk-aware authorization decisions. In the following sections, we will explore different ways of instantiating the $Auth$ function, particularly with the consideration of two types of mitigation methods: system obligations and user obligations.

[3]In the following sections we will elaborate on possible representations of $\mathcal{M}$ and study different authorization semantics associated with the use of these mitigation methods. For the purposes of the discussion in this section, it is sufficient to assume the existence of some abstract risk mitigation methods.

IV. System Obligations

As we mentioned in Section II, a *system obligation* is an action that could be caused to happen by a system. In the context of access control, as in XACML [10], we assume that a system obligation is some action that must be taken by the policy enforcement point (PEP) when enforcing an access control decision. The use of system obligations as a risk mitigation method has received attention in the literature [2], [3]. These obligations, including auditing, obfuscating or redacting the accessed data, are employed to account for and reduce the risk when allowing some risky access.

Formally, let $\mathcal{B}^s$ denote a set of system obligations. We define a *system mitigation strategy* to be a list $[(l_0, B^s_0), (l_1, B^s_1), \ldots, (l_{n-1}, B^s_{n-1}), (l_n, B^s_n)]$, where $0 = l_0 < l_1 < \cdots < l_n \leqslant 1$, and $B^s_i \subseteq \mathcal{B}^s$. We associate each permission p with a system mitigation strategy, denoted by $\pi(p)$. Informally, a system mitigation strategy $\pi(p)$ for $p \in P$ specifies that a set of system obligations B^s_i will be executed if the risk of granting p is within the interval $[l_i, l_{i+1})$. Note that the PEP is not required to do anything if $B_i = \emptyset$.

We then give an explicit authorization semantics for $Auth_s$ function with the use of system mitigation strategies. Given an access request $q = (u, p)$, the current system state σ, and the system mitigation strategy $\pi(p)$ for p, we define $Auth_s$ as:

$$Auth_s(q, \sigma) = \begin{cases} (\mathsf{allow}, B^s_i) & \text{if } risk(q, \sigma) \in [l_i, l_{i+1}),\ i < n, \\ (\mathsf{deny}, B^s_n) & \text{otherwise.} \end{cases}$$

In other words, the request (u, p) is permitted but the PEP must enforce system obligations B^s_i if the risk of allowing (u, p) belongs to $[l_i, l_{i+1})$; the request (u, p) is denied but the PEP must enforce system obligations B^s_n if the risk of allowing (u, p) is greater than or equal to l_n. Note that system obligations may be imposed even when there is no risk associated with the request. For some objects, for example, we may wish to record every access for audit purposes. Conversely, we may want to audit all denied accesses.

V. User Obligations: An Optimistic Approach

In this section, we consider the use of *user obligations* as the risk mitigation method; that is, the act of granting access generates obligations on the requesting user in order to manage the risk of allowing access to resources. We note that user obligations are often imposed as remedies for overriding access in many existing systems. For example, when a doctor is not available, a nurse is allowed to access some parts of a patient record, but she maybe obliged to explain the purpose of and justify the access request in writing within 24 hours. This kind of obligation is used to ensure that risky accesses are granted for the right reasons. However, unlike system obligations, a user obligation may go unfulfilled, because the user has either deliberately failed or forgotten to take the obliged actions. Hence, there is no guarantee that all risky accesses can be appropriately controlled and accounted for when using user obligations as the risk mitigation method. In this section we will examine this issue more closely, and suggest two different ways of (re)computing the risk of permitting requests when user obligations go unfulfilled.

A. User Obligations

Let us first discuss how we model user obligations themselves. As mentioned in Section II, the system cannot force a user to perform the obliged actions, but it should be able to monitor whether these actions have been performed. Therefore, this requires that a user obligation must have some sort of deadline before which it must be fulfilled. Let S denote a symbolic temporal domain, in which intervals are associated with labels. We could, for example, define the symbolic intervals $\mathsf{Instantly}$, $\mathsf{10Hours}$, $\mathsf{2Days}$, etc. We model a user obligation b^u as a tuple (a, s, v), where a is an action, $s \in S$ is a temporal interval during which a must be performed, and $v \in [0, 1]$ is the amount of loss if the obligation has not been fulfilled within the time window defined by s. We write $\mathcal{B}^u$ to denote the set of user obligations.

Following the idea of defining risk mitigation strategies, we associate each permission p with a list of interval-obligation pairs. Formally, we define a *user mitigation strategy* to be a list $[(l_0, B^u_0), (l_1, B^u_1), \ldots, (l_{n-1}, B^u_{n-1}), (l_n, B^u_n)]$, where $0 = l_0 < l_1 < \cdots < l_n \leqslant 1$, and $B^u_i \subseteq \mathcal{B}^u$.[4] Informally, if the risk associated with access request (u, p) is l, then the obligations corresponding to the interval containing l, $\{b^u\}$ for example, is assigned to u who is required to fulfill the obligation before its deadline. More precisely, we assume the existence of a system function f_{tran} that takes the current system time, the access request (u, p) and the corresponding obligation $\{b^u\}$ as its input, and outputs an assigned *obligation constraint* whose execution the system can monitor.

So let us look at what form the obligation constraint should take. We assume the existence of a clock, whose ticks are indexed by the natural numbers $\mathbb{N}$. A *time interval* $i = [t_1, t_2]$, where $t_1, t_2 \in \mathbb{N}$ and $t_1 < t_2$, is the set $\{t_1 \leqslant t \leqslant t_2\}$. We define an obligation constraint c as a tuple (u, a, i, v, p), where u is a user, a is an action, i is a time interval during which u is obliged to take action a, $v \in [0, 1]$ that denotes the (normalized) loss if u did not perform action a during the timeframe i, and $p \in P$ denotes a permission which u has requested, resulting in the obligation constraint to be assigned.

We assume that the PEP would be able to enforce an authorization decision and enable the corresponding obligation constraints to be assigned at the same time. Given an access request (u, p), when an authorization decision regarding (u, p) along with the corresponding user obligations arrive at the PEP, we use f_{tran} to transform those user obligations into a set of obligation constraints, and simultaneously enforce the decision.[5] Specifically, suppose that $Auth((u, p), \sigma) = (d, B^u)$, and d is enforced by the PEP at time t. Then, for each $(a, s, v) \in B^u$, the PEP generates an obligation constraint $(u, a, [t_s, t_e], v, p)$, where $t_s = t$, $t_e = t + t'$ and t' is the duration of the symbolic interval s. In other words, the elements of an obligation constraint c are derived from an access request, and a user obligation incurred by that request.

An obligation constraint $c = (u, a, [t_s, t_e], v, p)$ may be in one of three states: active, satisfied, or violated. We say c is *satisfied* if u has fulfilled the obligation (performed action a) at some time t, where $t \in [t_s, t_e]$. We say c is *violated* if the obligation has not been fulfilled, but t_e has passed. We say c is *active* if it is neither satisfied nor violated.

B. Strong Restriction

Following the authorization semantics defined in Section IV, we take a completely optimistic approach that allows some risky or exceptional access, but imposes some obligatory actions to be performed by the user to whom access has been granted. This approach is optimistic in the sense that it assumes that users are always diligent in performing their obligations to mitigate the risk incurred by their granted access. Note that the system should also make sure that users have sufficient permissions and resources to carry out their obligations, so the only reason for an obligation being unfulfilled is that a user fails to perform obliged actions before the deadline. In order to provide an incentive for users to fulfill their obligations on time, we define a mechanism to punish those users who fail to fulfill their obligations by restricting future access to system resources.

The basic idea is to introduce a *diligence score* for each user in the system. Given $u \in U$, we write $d(u) \in [0, 1]$ to denote the diligence score of u; the initial value of $d(u)$ is defined to be 1 for all u. If u fails to fulfill a user obligation, then the loss value v of this obligation (recorded in the obligation constraint tuple) will be deducted from u's diligence score. More precisely, if u fails to fulfill a user obligation with obligation constraint $(u, a, [t_s, t_e], v, p)$, then at time $t_e + 1$, $d(u) \leftarrow d(u) - v$. In other words, once the time window for fulfillment of the obligation has expired, we update the diligence value.

We use $d(u)$ as part of the input to the function that computes the risk associated with an access request (u, p). Specifically, we construct an optimistic risk-aware access control (or RAAC^d_o) model that instantiates the RAAC metamodel with user mitigation strategies and diligence scores associated with users. Recall that, given an access request $q = (u, p)$, and a system state σ, we have $risk(q, \sigma)$ that is computed based on the components of a specific access control model. We now define a new risk function $risk^d_o$ that takes the diligence score of users into account along with the $risk$ function. More

[4]In practice, it is likely that B^u_0 and B^u_n will be empty.

[5]In addition, the PEP is usually configured to ensure that this permission p is only granted to u with the condition that u accepts the incurred user obligations. It also implies that the corresponding obligation constraints only arise in the system when u accepts the obligations.

specifically,

$$risk_o^d(q,\sigma) = \min\{1, risk(q,\sigma) + (1 - d(u))\}.$$

Note that a special case of the RAAC$_o^d$ model is that $d(u) = 1$ as a constant for all $u \in U$, which represents a situation where all users are always diligent in fulfilling their obligations. In this case, the above risk function $risk_o^d$ is reduced to $risk$ but, in general, for some $u \in U$, $d(u)$ may change based on the fulfillment of obligations with which u is charged. As we saw from the $risk_o^d$ function, if a user u had a low diligence score due to her unfulfilled obligations, then the risk associated with u's requested access is relatively high, which, in turn, means that requests from that user are most likely being denied.

Given a RAAC$_o^d$ state σ, an access request $q = (u,p)$ and a user mitigation strategy $\pi(p)$ for p, we define the authorization decision function $Auth_o^d$ as,

$$Auth_o^d(q,\sigma) = \begin{cases} (\mathsf{allow}, B_i^u) & \text{if } risk_o^d(q,\sigma) \in [l_i, l_{i+1}),\ i < n, \\ (\mathsf{deny}, B_n^u) & \text{otherwise.} \end{cases}$$

In the context of user obligations, we might expect that $B_0^u = B_n^u = \emptyset$; that is, the least and most risky intervals are associated with unconditional allow and deny responses, respectively. Otherwise, request (u,p) is permitted but u is obliged to perform actions B_i^u during a specified timeframe when the risk of allowing (u,p) belongs to $[l_i, l_{i+1})$, where $1 \leqslant i < n$.

C. Weak Restriction

The integration of a user's diligence score in the risk computation provides a dynamic and effective way to control risky access as far as unenforceable user obligations are concerned. More specifically, since the diligence score of a user changes based on the non-fulfillment of her assigned obligations (at time $t_e + 1$, say) she might not be able to perform some risky access that would have been allowed at earlier times ($t \leqslant t_e$). On the other hand, some requests (u,p) that may have been granted unconditionally at $t \leqslant t_e$, might now be granted with obligations (at $t > t_e$).

In some cases, where a user may have good reasons for not being able to fulfill her assigned obligations, the approach of limiting her access to all resources might be too restrictive for her to perform job duties. In addition, this might generate significant work for a line manager in reviewing her employees' diligence scores and re-assigning their scores when adequate explanations for failing to fulfill obligations are given. As such, we believe that it might be more appropriate to only restrict user's requests regarding the same permission that incurred obligations that remain unfulfilled; this, we call *weak restriction*.

We now explore a method for enforcing weak restriction. We employ a similar approach to the enforcement of historical constraints by creating a blacklist [13] – a dynamic access control structure that contains risky requests. In other words, in order to determine the risk of permitting a request from a user to invoke a permission, the system will also consult the blacklist to see whether the user has been diligent in fulfilling assigned obligations.

We now consider the implementation of blacklists in more detail. Let $c = (u, a, [t_s, t_e], v, p)$ be an obligation constraint. At time $t_e + 1$, c becomes violated, and we would include (u, p, v) in a blacklist. Hence the blacklist is updated over time to reflect the occurrence of violated obligation constraints.

We model a blacklist at time t as a weighted, bi-partite graph $G_t = (V_t, E_t, \lambda_t)$, where $V_t \subseteq U \cup P$, $E_t \subseteq U \times P$, and $\lambda_t : E_t \to \mathbb{R}$. For $(u,p) \in E_t$, $\lambda_t(u,p)$ denotes the cumulative loss due to unfulfilled obligations at time t incurred by u's previous invocations of p.

For ease of exposition, we assume that the detection of a violated obligation constraint and the action of updating the blacklist can be performed in a single clock tick, and its effect is reflected in the graph of the current clock tick. Suppose a blacklist graph at time t is G_t, and an obligation constraint c is violated at $t+1$. The graph G_{t+1} at time $t+1$ will be immediately affected by c's violation, rather than waiting until $t+2$. When multiple obligation constraints are violated at the same time t, we assume that the system is able to deal with those violated constraints in an unspecified, fixed order, and all of those operational effects are reflected in a single clock tick t. Clearly, we can see that the evolution of blacklists is a sequence of blacklist graphs $G_0 \xrightarrow{C_1} G_1 \ldots G_{n-1} \xrightarrow{C_n} G_n$, where C_i is a set of violated obligation constraints at time i.

We now present a sketch of an algorithm that is used to transform a blacklist graph G to G' when an obligation constraint $c = (u, a, [t_s, t_e], v, p)$ becomes violated. Specifically, we explain how the graph G_{t_e+1} is derived from G_{t_e} and c when the corresponding obligation remains unfulfilled at time $t_e + 1$. We first check to see if $(u,p) \in E_{t_e}$. (The existence of such an edge indicates that u has already failed to fulfill obligation(s) incurred by the granting of a previous request (u,p).) If such an edge exists in G_{t_e}, then

$$G_{t_e+1} = (V_{t_e}, E_{t_e}, \lambda')$$

where

$$\lambda'(x,y) = \begin{cases} \min\{1, \lambda_{t_e}(u,p) + v\} & \text{if } x = u \text{ and } y = p, \\ \lambda_{t_e}(x,y) & \text{otherwise.} \end{cases}$$

On the other hand, if $(u,p) \notin E$, then

$$G_{t_e+1} = \begin{cases} (V_{t_e} \cup \{u\}, E', \lambda') & \text{if } u \notin V_{t_e} \text{ and } p \in V_{t_e}, \\ (V_{t_e} \cup \{p\}, E', \lambda') & \text{if } u \in V_{t_e} \text{ and } p \notin V_{t_e}, \\ (V_{t_e}, E', \lambda') & \text{if } u \in V_{t_e} \text{ and } p \in V_{t_e}, \end{cases}$$

where $E' = E_{t_e} \cup \{(u,p)\}$ and

$$\lambda'(x,y) = \begin{cases} v & \text{if } x = u \text{ and } y = p, \\ \lambda_{t_e}(x,y) & \text{otherwise.} \end{cases}$$

The three cases above have the following interpretations.

(1) $u \notin V_{t_e}$ and $p \in V_{t_e}$: u has been always diligent in performing her assigned obligations, and this was the first time she failed to fulfill the obligation (let c become violated).

(2) $u \in V_{t_e}$ and $p \notin V_{t_e}$: u has not been diligent in performing her assigned obligations, as she has already appeared in the blacklist. Moreover, either no one has invoked p or all obligations incurred by previous invocations of p have been discharged.

(3) $u \in V_{t_e}$, $p \in P$ and $(u,p) \notin E_{t_e}$: u has not been diligent in fulfilling her obligations, someone other than u has failed to fulfill obligations incurred by some previous invocation of p, but this was the first time u failed to fulfill the obligations associated with her invocation of p.

Examining those different possibilities reveals that the constructed blacklist actually contains valuable information that can be analyzed for different purposes. For example, we might study the blacklist to evaluate the diligence of users in fulfilling their assigned obligations. Furthermore, we may like to identify those permissions whose invocation incurred obligations that were frequently unfulfilled. In the following, we suggest referring to the blacklist when determining the risk of permitting a request.

We now integrate the idea of blacklist into an optimistic RAAC model. We represent the optimistic risk-aware access control (or RAAC$_o^b$) state at time t as a tuple $\sigma_t = (U, P, \pi, G_t, \tau)$, where $G_t = (V_t, E_t, \lambda_t)$ is a blacklist graph. Given an access request at time t, denoted by $q_t = (u,p)$, we formally define a new risk function as $risk_o^b(q_t, \sigma_t)$

$$risk_o^b(q_t,\sigma_t) = \begin{cases} risk(q_t,\sigma_t) & \text{if } (u,p) \notin E_t, \\ \min\{1, risk(q_t,\sigma_t) + \lambda_t(u,p)\} & \text{otherwise.} \end{cases}$$

In other words, given an access request (u,p), if there does not exist an edge between u and p in the blacklist, the risk for (u,p) is determined solely by computing the *risk* function. In contrast, if that edge exists in the blacklist, the risk computation should also consider the loss value $\lambda(u,p)$ from the blacklist graph.

Given an RAAC$_o^b$ state σ, an access request (u,p), and a user mitigation strategy $\pi(p)$ for p, we can define an authorization decision function $Auth_o^b$ as we did for $Auth_o^d$.

VI. User Obligations: A Conservative Approach

Since user obligations are not enforceable, a system can never guarantee that an obligation will be fulfilled to mitigate the risk of allowing access. We believe that restricting future access is an effective way to control risky access in the context of organizations where the number of users is more stable. However, it cannot prevent a malicious user from obtaining sensitive information by somehow passing system-defined risk thresholds without performing an assigned obligation.

In this section, we take a conservative approach that only allows a request if there is no risk associated with the request. For any risky access request (u,p), we initially deny such access, but return obligations for u to perform. If u fulfills those obligations in order to get access, we may subsequently permit the request by reducing the associated risk.

Firstly, we refine the structure of a user obligation as a tuple $(a, s, (w, s'))$, where $w \in [0,1]$ denotes the reward credits if the action a is performed during the time interval s, and $s' \in S$ denotes the symbolic interval during which the reward credits w are valid. Note that w is usually defined by referring to the risk interval with which the obligation is associated. Let $(l_i, \{b^u\})$ be a risk-obligation pair, and $b^u = (a, s, (w, s'))$. We require that $w > l_i$ in order to ensure that any user who fulfills the obligation would be rewarded with enough credits to eliminate the risk associated with the corresponding permission. In other words, if there are multiple user obligations associated with the risk interval $[l_i, l_{i+1})$, we need to carefully define w in each obligation so that the user is required to fulfill all the obligations for being able to perform the permission.

When a user u attempts to invoke a permission p, suppose that the risk of allowing (u,p) results in the generation of the user obligation $b^u = (a, s, (w, s'))$. Then an obligation constraint $c = (u, a, [t_s, t_e], (w, s'), p)$ becomes active in a system when the requesting user u accepts the user obligation b^u. Note that the symbolic interval s' only has a concrete interpretation when the obligation b^u is fulfilled. Specifically, s' will be replaced by a concrete time interval $[t'_s, t'_e]$ when c is satisfied at t'_s, where $t'_s \in [t_s, t_e]$ and t'_e is computed by referring to s'. We make the assumption that this obligation constraint c will be purged from the system when c becomes violated. If u performs the obliged action a at any point during the timeframe $[t_s, t_e]$, we would grant a subsequent request (u,p) in the interval $[t'_s, t'_e]$ to "reward" u for fulfilling b^u. We propose two methods for implementing this reward mechanism below.

Again, we use a dynamic diligence score of users to adjust the risk associated with users' requests. In this conservative setting, however, we initially set $d(u) = 0$ for all users; users can increase their diligence scores by fulfilling obligations. We write $d_t(u)$ to denote the diligence score of u at time t. Suppose that an obligation constraint $(u, a, [t_s, t_e], (w, s'), p)$ is satisfied at $t'_s \in [t_s, t_e]$. Then

$$d_{t'_s}(u) \leftarrow d_{t'_s - 1}(u) + w \quad \text{and} \quad d_{t'_e + 1}(u) \leftarrow d_{t'_e}(u) - w,$$

where t'_e is computed on the basis of s'. In other words, we increase the diligence score of a user when she fulfills an obligation and reduce it when the reward expires.

Given an access request at time t, denoted by $q_t(u,p)$, and a conservative risk-aware access control (or RAAC$_c^d$) state σ at time t that includes user mitigation strategies and the diligence score of users, we define a new risk function $risk_c^d$ as:

$$risk_c^d(q_t,\sigma_t) = \max\{0, risk(q_t,\sigma_t) - d_t(u)\}.$$

We also define an authorization decision function $Auth_c^d$, where

$$Auth_c^d(q_t,\sigma_t) = \begin{cases} (\mathsf{allow}, B_0^u) & \text{if } risk_c^d(q_t,\sigma_t) < l_1, \\ (\mathsf{deny}, B_i^u) & \text{if } risk_c^d(q_t,\sigma_t) \in [l_i, l_{i+1}),\ 1 \leqslant i < n, \\ (\mathsf{deny}, B_n^u) & \text{otherwise.} \end{cases}$$

In other words, a request by u to perform p is allowed (usually unconditionally) if the risk of granting (u,p) is less than a specified risk threshold l_1 (and denied otherwise). Otherwise, some user obligations B_i^u are required to be carried out with the deny access if the risk is perceived as being relatively high (within some interval $[l_i, l_{i+1})$, where $1 \leqslant i < n$). By fulfilling those obligations, u will be rewarded with an increase in her diligence score, and subsequently the request might be granted. It can been seen that this approach ensures that risky accesses are only granted to diligent users whose degree of diligence is exhibited by their high diligence scores at some point in time. Furthermore, since the diligence score of a user is dynamic, the user must continue to fulfill obligations to gain access.

We now take a slightly more restrictive method to implement the reward mechanism using the idea of a *whitelist*. We model a whitelist at time t as a weighted bi-partite graph $G_t = (V_t, E_t, \theta_t, \eta_t)$, where $V_t \subseteq U \cup P$, $E_t \subseteq U \times P$, $\theta_t : E_t \to I$ and $\eta_t : E_t \to \mathbb{R}$. For $(u,p) \in E_t$, $\theta_t(u,p)$ denotes the set of points in time in which the association between u and p is enabled, and $\eta_t(u,p)$ denotes the reward credits u possesses for invoking p at time t. We require that if $(u,p) \in E_t$ at time t, then $t \in \theta(u,p)$. In other words, an edge (u,p) exists only if it is enabled. When an obligation constraint $(u, a, [t_s, t_e], (w, s'), p)$ is satisfied, we employ an algorithm similar to the one used to create a blacklist to add an entry into the whitelist. The only difference is that we need a function f'_{tran} that is able to transform the symbolic temporal constraint s' into the enabling times for the edge (u,p) in the whitelist. More specifically, if c is satisfied at time t'_s, f'_{tran} takes t'_s and s' as inputs and returns the enabling interval $[t'_s, t'_e]$ for reward w.

We represent a conservative risk-aware access control (or RAAC$_c^w$) state at time t as a tuple $\sigma_t = (U, P, \pi, G_t, \tau)$, where $G_t = (V_t, E_t, \theta_t, \eta_t)$ is a whitelist graph. Given an access request at time t, that is $q_t = (u,p)$, we define a new risk function $risk_c^w$ as

$$risk_c^w(q_t, \sigma_t) = \begin{cases} risk(q_t, \sigma_t) & \text{if } (u,p) \notin E_t, \\ \max\{0, risk(q_t, \sigma_t) - \eta_t(u,p)\} & \text{otherwise.} \end{cases}$$

In other words, if there exists an entry in the whitelist for a request (u,p), then the reward credits associated with the request are used to reduce the original risk, and subsequently the request will be allowed. Clearly, we can define $Auth_c^w$ in exactly the same way as we did for $Auth_c^d$.

VII. Related Work

There has been considerable research on obligation policies [7]–[9], [14], [15], "break-glass" policies [16]–[18] and risk-aware access control [1]–[6]. However, to the best of our knowledge, no previous work has systematically studied obligations in the context of RAAC. In particular, we are not aware of any work in the literature proposing different risk-aware authorization semantics that take unfulfilled obligations into account. In this section, we will review the research most closely related to ours, and highlight the novelty of our contributions.

We first examine work on the specification and monitoring of obligations in the context of access control [7], [8], [14]. Bettini *et al.* [7] formalize a rule-based policy framework in which an authorization decision is associated with a set of obligations and provisions. They focus on studying the problem of choosing appropriate policy rules to minimize the provisions and obligations that a user is required to fulfill in order to get access. Although their work, like ours, is concerned with the association between obligations and an authorization decision, it is under the assumption that actions in provisions and obligations can always be fulfilled. Bettini *et al.* [14] further extend their policy framework to investigate mechanisms for monitoring the fulfilment of obligations. Similarly, Gama and Ferreira [8] implement a prototype obligation-monitoring platform that supports the specification and enforcement of obligation policies based on the xSPL language [19]. This work influenced our decision to use monitors for checking the fulfillment of obligation constraints in the context of risk-aware models.

On the other hand, break-glass access control [16]–[18] and risk-aware access control [1]–[6] have been recently proposed. In our view, these two access control paradigms attempt, in some sense, to achieve the same goal from different perspectives: that is, to address the "semantic gap" between what can be encoded in authorization policies and what the access control requirements of an organization are [20]. Break-glass policies were introduced to supplement access control policies in order to deal with exceptional situations not anticipated by the access control policies (such as a lack of suitably authorized personnel in an emergency situation). A break-glass policy essentially encodes the conditions under which overrides are permitted (that is, the circumstances under which a previously unauthorized action becomes authorized). Marinovic *et al.* [17] define break-glass rules that can encode whether a user is permitted to override a denied decision based on the knowledge of obligations that he has accepted, fulfilled or violated. However, we explore this question in more detail, suggesting different approaches to encoding break-glass rules according to the degree of diligence exhibited by the user in fulfilling obligations.

Recent work on RAAC has extended access control policies to incorporate notions of risk, in order to make an authorization decision based on how much risk is incurred by allowing or denying access. As we mentioned above, most existing work in RAAC is concerned with risk estimation [2], [5], [6] and economic mechanisms for risk allocation [4]. However, there are only a few papers that study system mitigation strategies in RAAC [2], [3]. Unlike our models, none of these have focused on risk-aware authorization policies in conjunction with the consideration of user mitigation strategies.

In summary, we believe that our research is complementary to the above mentioned work on break-glass policies and RAAC: it provides a richer range of policies than work on break-glass policies; and it provides a model for defining user obligations together with a method of accounting for failure to fulfill those obligations, and features that are absent in other RAAC models.

VIII. Concluding Remarks

We have introduced a metamodel for capturing RAAC systems from which existing (risk-aware) access control models may be derived as particular cases, and from which several approaches can be developed for domain-specific applications.

We have also developed a number of risk-aware models as instances of the metamodel from the perspective of different types of risk mitigation methods: system obligations and user obligations. In particular, when using user obligations as mitigation methods, we studied risk-aware authorization semantics that account for failure to fulfill those obligations. We proposed punishment mechanisms (sanctions) as incentives for users to fulfill obligations to control and account for the risk incurred. We also discussed reward mechanisms that enable users to be permitted risky access through the fulfillment of assigned obligations. To our knowledge, this is the first attempt in the literature that has extensively studied obligations in the context of RAAC.

One obvious aspect of future work will be to develop other interesting risk-aware models as special cases of our metamodel. For example, on the basis of RAAC_o^d, an alternative approach of using diligence score would be to deduct v from the diligence score when the state of the obligation constraint becomes active and to add v to the diligence score when the state changes to satisfied. In this way, we provide an immediate incentive to a user to fulfill obligations rather than the delayed punishment that is encoded in the optimistic approach described earlier in the paper.

We also intend to enhance our punishment mechanisms by examining why users fail to fulfill obligations. We could assign weights to different reasons for the violation of obligation constraints, and take those weights into account as part of the input to the risk computation. A further interesting possibility for future work is to extend our risk-aware models to include spatio-temporal domains that may also be part of the input to the risk computation. In other words, the risk of permitting an access request may depend on contextual information, such as the location of the user and the time at which the access request is made. Finally, we would like to investigate the application of our risk-aware models in workflow systems, where it would be interesting to consider how much aggregated risk is appropriate by authorizing users to perform different tasks, in order to ensure that every instance of a workflow can be completed and associated organizational goals achieved.

Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments. This research is supported by the award made by the RCUK Digital Economy programme to the dot.rural Digital Economy Hub; award reference: EP/G066051/1.

References

[1] JASON Program Office, "Horizontal integration: Broader access models for realizing information dominance," MITRE Corporation, Technical Report JSR-04-132, 2004.

[2] P.-C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger, "Fuzzy multi-level security: An experiment on quantified risk-adaptive access control," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, 2007, pp. 222–230.

[3] L. Chen and J. Crampton, "Risk-aware role-based access control," in *Proceedings of the 7th International Workshop on Security and Trust Management*, 2011, pp. 140–156.

[4] I. Molloy, P.-C. Cheng, and P. Rohatgi, "Trading in risk: Using markets to improve access control," in *Proceedings of the 2008 Workshop on New Security Paradigms*, 2008, pp. 107–125.

[5] Q. Ni, E. Bertino, and J. Lobo, "Risk-based access control systems built on fuzzy inferences," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, 2010, pp. 250–260.

[6] J. A. Clark, J. E. Tapiador, J. A. McDermid, P.-C. Cheng, D. Agrawal, N. Ivanic, and D. Slogget, "Risk based access control with uncertain and time-dependent sensitivity," in *Proceedings of the International Conference on Security and Cryptography*, 2010, pp. 5–13.

[7] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera, "Provisions and obligations in policy management and security applications," in *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002, pp. 502–513.

[8] P. Gama and P. Ferreira, "Obligation policies: An enforcement platform," in *Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks*, 2005, pp. 203–212.

[9] K. Irwin, T. Yu, and W. H. Winsborough, "On the modeling and analysis of obligations," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 134–143.

[10] *eXtensible Access Control Markup Language (XACML) Version 2.0*, OASIS, 1 February 2005, OASIS Standard (T. Moses, editor).

[11] H. L. A. Hart, "Bentham on legal rights," in *Oxford Essays in Jurisprudence*, ser. Second Series, A. W. B. Simpson, Ed. Oxford University Press, 1973, pp. 171–201.

[12] T. J. Norman, C. Sierra, and N. R. Jennings, "Rights and commitment in multi-agent agreements," in *Proceedings of the Third International Conference on Multiagent Systems*, 1998, pp. 222–229.

[13] J. Crampton, "Specifying and enforcing constraints in role-based access control," in *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, 2003, pp. 43–50.

[14] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera, "Obligation monitoring in policy management," in *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*, 2002, pp. 2–12.

[15] M. Pontual, O. Chowdhury, W. H. Winsborough, T. Yu, and K. Irwin, "On the management of user obligations," in *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, 2011, pp. 175–184.

[16] C. A. Ardagna, S. De Capitani di Vimercati, T. Grandison, S. Jajodia, and P. Samarati, "Regulating exceptions in healthcare using policy spaces," in *Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2008, pp. 254–267.

[17] S. Marinovic, R. Craven, J. Ma, and N. Dulay, "Rumpole: A flexible break-glass access control model," in *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, 2011, pp. 73–82.

[18] A. D. Brucker and H. Petritsch, "Extending access control models with break-glass," in *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, 2009, pp. 197–206.

[19] C. Ribeiro, A. Zuquete, P. Ferreira, and P. Guedes, "SPL: An access control language for security policies and complex constraints," in *Proceedings of the Network and Distributed System Security Symposium*, 2001.

[20] D. Povey, "Optimistic security: A new access control paradigm," in *Proceedings of the 1999 Workshop on New Security Paradigms*, 2000, pp. 40–45.