R2Q: A Risk Quantification Framework to Authorize Requests in Web-based Collaborations

Nirnay Ghosh
nirnay@cs.iiests.ac.in
IIEST Shibpur
Howrah 711103, India

Rishabh Singhal
rishabh.x2.singhal@jpmchase.com
JP Morgan Chase & Co.
Mumbai, Maharastra, India

Sajal K Das
sdas@mst.edu
Missuori University of S&T
Rolla, MO 65409, USA

ABSTRACT

Web-based collaboration provides a platform which allows users from different domains to share and access information. In such an environment, mitigating threats from insider attacks is challenging, particularly if state-of-the-art token-based access control is used to authorize (permit or deny) requests. This entails the need for an additional layer of authorization based on soft-security factors such as the reputation of the requesters, risks involved in requests, and so on to make the final decision. In this paper, we propose a novel risk quantification framework, called *R2Q*, which exploits a weighted regression approach to compute the expected threat related to a collaboration request. Our model combines the shared object's sensitivity, access mode of the request, requester's security level and reputation, and maps the expected threat to a risk score using the *prospect theory (PT)* inspired value functions to actualize decision making under uncertainty of economic outcomes (loss or gain). Simulation-based performance evaluation validates the efficacy of our framework and demonstrates that it can classify requesters based on their past behaviours, and also enables the collaboration platform to achieve higher rates of successful authorization.

KEYWORDS

Access request; Authorization; Collaboration; Prospect theory; Risk

ACM Reference Format:
Nirnay Ghosh, Rishabh Singhal, and Sajal K Das. 2019. R2Q: A Risk Quantification Framework to Authorize Requests in Web-based Collaborations. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3321705.3329852

1 INTRODUCTION

Recently web-based collaborations among multiple autonomous organizations (public-public, public-private, and private-private) have become essential as they allow easy access of shared resources from remote locations. Examples include document-centric collaboration, project management, wikipages, feeds from social networks, file sharing and synchronization, and so on.

Authentication of a requester and subsequent authorization of a request in collaboration platforms, viz., Google Cloud Storage, Microsoft Azure, etc., are done through issuance and verification of access tokens[1], a concept similar to the OAuth 2.0 protocol[2] protocol. Resources such as files, images, databases, etc., are stored in the form of objects and are accessed by universal resource identifiers (URIs). The object owner determines the privileges (e.g., view, edit, execute, etc.) on the shared objects for every user with which s/he intends to collaborate and provides this information in form of access control list (ACL) to the authorization module of the security server. The authentication module issues access token containing identity information to potential requesters on behalf of the owner. At the time of the request, the querying application (requester) attaches this token with the hashed code of the object. On receiving the request, the authentication module checks the identity of the requester, and if it finds the requester to be legitimate, forwards to the request to the authorization module. Next, the authorization module reads the policies given in the ACL, and after comparing with the token, determines whether to allow or reject the request. If the user is authenticated and the ACL grants permission for the requested operation, the access is granted until the token is timed out. Otherwise, a 403 Forbidden Error (Access Denied) is returned.

As token-based authentication and authorization are not based on the properties of the requesting entity, it does not bind a user to its purported behaviour or actions. It does not convey any information about the behaviour of the requester between the time the token was issued and its actual use. Authorization is done strictly on the basis of the identity of the requester and the validity of the token issued for a particular session. Either the requester's token is accepted and the required privileges are allowed, or the token is rejected and the access is denied. However, there is no provision to determine if any user with valid token and legitimate identity will cause any security breach with the shared information. Thus, it is imperative to look beyond static policy-driven token-based access control mechanisms to mitigate threats from malicious insiders.

Preventing insider attack is a non-trivial challenge as it needs to track the behaviours of the users and eventually forecast future accesses based on a behaviour-based metric. Thus, the scope of this work is to propose a novel risk quantification framework, called *R2Q*, which models uncertainty due to sharing and incorporates soft security mechanism (e.g., reputation, risk) to thwart insider attacks. Our framework is generic and involves human interventions only in setting the model parameters to adapt the collaboration platform's risk tolerance attitude.

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329852

[1] https://cloud.google.com/storage/docs/authentication
[2] https://oauth.net/2/

The major contributions of the paper are highlighted below: (1) We model the *expected threat* of an access request as a response variable in a weighted regression function consisting of two explanatory variables: *uncertainty due to requester's shared resource usage history* (weighed by his/her security level) and the *utility expected to be gained by abusing/misusing the given access mode* (weighed by object's sensitivity); (2) Next, we transform the expected threat to a measure that captures the risk of the request. The transformation is achieved using *prospect theory (PT)* inspired link functions that model varying risk tolerance attitudes (e.g., risk-seeking, risk-averse, risk neutral) and facilitates decision making under uncertainty; (3) An extensive simulation-based evaluation validates our framework and demonstrates that the framework segregates requesters based on their behaviours and also achieves higher rates of successful authorization.

The rest of the paper is organized as follows. Section 2 reviews the existing works on the risk-based authorization. Section 3 introduces the system and threat models. In Section 4, we present the functional descriptions of different modules of the *R2Q* framework for quantification of risk in web-based collaborations. Section 5 presents the simulation environment and the results obtained from the experiments. Finally, conclusions and future research directions are drawn in Section 6.

2 RELATED WORKS

The primary objective of risk-based access control (RAC)[17][15] is to maximize the sharing of information, provided that the access risk is kept under a permissible level. A key step in the RAC model is to quantify the risk associated with each access request, and depending on whether the value exceeds a threshold or not, the access decision is made. In this section, we review some of the works which quantify risk in a multi-user environment.

In [6], fuzzy multi-level security (MLS) access control quantifies the risk as a function of the value of information and the probability of unauthorized disclosure. The probability is modelled using the requester's *temptation index*, which is a function of the object and subject's security clearance levels. Similarly, in [17], a fuzzy inference system is proposed to evaluate access risk based on predefined rules satisfying the simple security property of the *Bell-LaPadula model*. These rules have the object and subject's security levels as the antecedents, and risk as to the consequent. In [1], authors introduce a trust-and-obligation based framework to detect and mitigate violation of *posterior* obligations associated with resource access. The *risk-adaptive access control (RAdAC)* model [12] proposes the use of RAC for military applications, where the access decision is made based on the computation of security and operational needs. It makes access control flexible by allowing risky but operationally relevant requests. In [15], the risk associated with the read access is quantified and based on the level of risk, various mitigation actions are proposed. In [18], the trust and risk values related to each subject-object pair is evaluated. It is based on the past behaviour of the subject while handling a particular object. If the trust exceeds the risk, the access is permitted, otherwise, it is denied. A similar approach is attempted in [5], where trust is incorporated into role-based access control, called *TrustBAC*. Here, users are assigned to trust levels instead of roles based on user credentials, user behaviour history, user recommendation etc. Trust levels are assigned to roles which are assigned to permissions as in role-based access control.

Several works which considered cloud environment as one of the major providers of online collaboration service have proposed risk-based access controls. In [7], a method to quantify risk from the infrastructure provider's perspective is presented. It uses system-level functional information and high-level non-functional contextual information of the service provider. In [8] [9], the risk in cloud-hosted collaborations is evaluated as a weighted sum of three categories of metrics: (i) 27 contextual metrics (as given in the RAdAC model), (ii) 3 security metrics (confidentiality, integrity, and availability tenets), and (iii) subject historical metric. If the contextual information is missing, it is dynamically inferred by an ontological model that is composed offline. The fuzzy inference-based approach is seen in [10], which aggregates different antecedents (viz., object sensitivity, subject security level, technical, operational, legal risks, etc.) and quantify the interoperability risk as to the consequent.

Limitations: Synthesis of the literature enables us to summarize its limitations which forms the motivation of this work. Following are some of their limitations: (1) The proposed models lack generality and are highly sensitive to the weights assigned to different risk factors by the domain experts; (2) They are computationally expensive as they need to perform additional non-trivial tasks, such as parsing of ontology structure, verifying service level agreement (SLA) negotiation clauses, etc. in order to infer missing information or check the conformance, which incurs additional overheads; (3) In [8] [9], the application area is specific viz., military application, health care, and so on; (4) These works did not model the *uncertainty* of misusing the shared resources before giving access to any remote user.

3 SYSTEM AND THREAT MODELS

In this section, we introduce the system and threat models of the proposed *R2Q* framework.

3.1 System Model

Consider a web-based collaboration platform which allows users to host their resources or objects and share them with others (refer to Fig. 3 in Appendix-A). As Software-as-a-Service (SaaS) clouds are one of the major proponents of online collaboration [14], the collaboration platform may be offered as service by a SaaS cloud. The platform perceives a participating domain as either local or remote, depending on whether it is the provider or the consumer of the service, respectively. Following are the important components of the proposed *R2Q* framework:

Object Owner: An object owner u_i is a user in the local domain who owns one or more objects. This is similar to the hospital in the above motivating example. The owner shares the objects and sets the level of accesses (view, edit, execute) on them (steps 1 and 2 in Fig. 3). S/he is also liable to send feedbacks to the *R2Q* framework after the collaboration (step 3).

Shared Object: An object o_j is an entity (viz., document, image, program, etc.) which is fully managed, configured, and shared by its owner on the collaboration platform (step 1). It is similar to the patient's electronic health record (EHR) in our motivating example. Each object belongs to one of the multiple linearly ordered classes,

reflecting its sensitivity to the owner [4]: *Top Secret* ≻ *Secret* ≻ *Confidential* ≻ *Unclassified*. The higher the object class, the more sensitive is the object, and the greater is the need to protect it.

Requester: A requester r_k is a user either from the remote domain or from the Internet. In our motivating example, the requester is either the medical practitioner or staffs. The requester needs to obtain an access token from the collaboration platform to initiate collaboration (step 4). Furthermore, r_k needs to attach this token with the request to access any object o_j shared by u_i (step 6). The requester r_k can access objects if the server grants the access token, validates the request with respect to the policies designed by the owner, and also finds that the access risk of the request is under the permissible limit (step 10). A requesting user is liable to have a reputation score and a security level. Depending on the reputation score, we can classify the users as: *honest, selfish, and malicious* (refer Section 5.2).

Authentication and Authorization server: The authentication and authorization server is managed by the collaboration platform (it is the health care cloud in the above example scenario). It enables a user u_i to set predefined policies on object o_j (step 2). Moreover, the server is responsible to grant or deny both access token and collaboration request to the requester (steps 5 and 9).

R2Q Framework: This is the proposed framework which can be either a part of the collaboration platform or implemented as a separate Web service to reduce the overhead on the collaboration platform. It is invoked as a service by the authentication and authorization server at the time of ascertaining risk score(s) of collaboration requests(s) to make the authorization decision(s) (steps 7 and 8). The framework also receives feedback from the resource owning domain (step 3) following a collaboration.

It is clear that access request is the central aspect of our system model and it essentially constitutes the operation to be performed against a particular object. We have considered three typical access modes viz., *VIEW*, *EDIT*, and *EXECUTE*. We define an access request (or simply request) as follows.

Definition 1. *(Request). A request q_j is defined as a tuple $\langle o_j, a_j \rangle$, where o_j is an object and a_j is the access mode requested on o_j.*

In most organizations, the employees are usually hierarchically arranged into different *security levels* based on their designations and scopes of access privileges. This is similar to a widespread concept of 'roles', also used in collaboration platform like the *Google Storage Cloud*. A role defines the access scope and consists of varying permissions. A requester has to be mapped to one or more role(s) before he can access the resource(s). We denote the roles as 'security levels' in our context and assume that the collaboration platform provides interfaces using which the participating domains can assign security levels to their users.

One popular approach to establishing multi-level security is the *Bell-LaPadula (BLP) model* [2]. Traditionally, this model was used in military organizations to prevent an unauthorized read of the shared documents through the linear ordering of the security levels. The requester has to specify his/her security level while making a collaboration request. For anonymous users from the Internet, the administrator can use a default security level. We formally define a collaboration request as follows.

Definition 2. *(Collaboration request). A collaboration request, c_j, is defined as a four-tuple $\langle u_{ID}^{d_r}, u_{SL}^{d_r}, d_l, q_j \rangle$, where $u_{ID}^{d_r}$ is the identity of a requesting user belonging to a remote domain d_r and $u_{SL}^{d_r}$ is the requester's security level (assigned by remote domain administrator) whereas d_l is the local domain which owns the resources, and q_j is the actual resource request.*

3.2 Threat Model

We assume that the web-based collaboration service is publicly available and multiple domains use this service to share their resources. Following are the two possible threats posed to the objects (shared via collaboration service) by legitimate but malicious requesters: (i) *Threat on confidentiality*: disclosure of the contents to the unauthorized users, and (ii) *Threats on integrity and/or availability*: misappropriation or deletion of contents from the object.

Threat (i) is similar to the credential theft which can be exercised through disclosure of personal data to the unauthorized agents. Such a threat can be organized even if the perpetrator has only the VIEW permission. On contrary, threat (ii) is materialized through misuses of EDIT and EXECUTE privileges, which enable the malicious user either to tamper the data or compromise the remote server through arbitrary code(s), send as input.

Thus, there is a need for a soft security mechanism, like risk, reputation, etc. as an additional security layer on top of the existing token-based authorization module. We term any attempt of sending arbitrary requests or tampering of data as *bad interaction*. Furthermore, if any resource owner reports a possible breach of confidentiality (such as information leakage) following a transaction, we also consider it to be bad. Our framework keeps track of bad interactions of a requester based on the feedback received from the collaborating domains. It aggregates such information, feeds the same to a custom reputation model, updates the requester's reputation score, and eventually uses it to compute the risk.

4 PROPOSED R2Q FRAMEWORK

For estimating the risk associated with a collaboration request, we propose the *R2Q* framework, which can be envisioned as an additional security layer above the token-based authorization method. Fig. 4 in Appendix-B presents a schematic representation and control flow of different modules of the framework. The functional descriptions of the major modules are as follows.

4.1 Modeling Requester's Reputation

We use inverse Gompertz function [11] to model a requester's reputation. In order to avoid the cold-start problem, for any new user, we tend to trust him/her by believing that he will not misuse the shared objects. Consequently, we assign him/her with maximum reputation score which enables him/her to readily access objects with different sensitivity levels. However, if s/he is found to misbehave during a particular interaction, then it should negatively affect his/her overall reputation; if s/he persists with malicious activities in subsequent collaborations, then his/her reputation should ideally drop to zero over time. This will result in revocation of the access token.

The output of the inverse Gompertz function non-linearly decreases from its upper asymptote to reach the lowest value and is reflective of decay in the trust relationship in any multi-user environment [16]. Thus, the higher the number of misbehaviours over a given period of time, the faster is the rate in the drop of

reputation. The reputation score of any requester r_k is given as:

$$\rho_{r_k} = 1 - A \cdot e^{-B \cdot e^{-C \cdot \tau_{r_k}}} \tag{1}$$

where the input τ_{r_k} gives the number of bad interactions or misbehaviours for the requester r_k. This information comes as feedback from different resource providers with which r_k has interacted till the present time. Here, A is the upper asymptote to decide on the maximum value the reputation can attend (for our case $A = 1$); B is a positive constant which controls the displacement of the reputation score, and C adjusts the decay rate. Refer to Appendix-C for parameter study.

4.2 Expected Threat in Collaboration Requests

As given by Definition 2, a collaboration request contains the object requested, the required access mode, the identity and security level of the requester. Granting access to a shared object involves a certain degree of threat of misuse. We intuit that the magnitude of threat for any collaboration request c_j potentially depends on two factors (explanatory variables): (i) the uncertainty due to the requester's shared resource usage history (w_l), and (ii) the utility s/he is expected to gain (from resource owner's perspective) if the given access mode is misused on the requested object (w_s). We model the response variable, expected threat (θ_{c_j}), as a weighted linear regression of the explanatory variables given by:

$$\theta_{c_j} = l \cdot (w_l) + s \cdot (w_s) \tag{2}$$

where, l and s are the coefficients to the explanatory variables and termed as the requester's security level and object's sensitivity, respectively. If any requester with higher security level is a potential perpetrator, and the object s/he has requested is very sensitive, our model magnifies the threat perception. This is comprehensible, as, in any organization, privileged users (belonging to higher security level) enjoy complete access on most of the resources, and are capable of causing greater damage, than normal ones.

4.2.1 Modeling Uncertainty Function (w_l). In Eqn. (2), w_l is the uncertainty whether the requester will misuse the access, given his/her usage history. Generally, if the security level is high, the user has the privilege to access sensitive information in an organization. Thus, if a requester, with a known security level, has shown pieces of evidence of misbehaviour in the past, the uncertainty about genuine usage of future accesses looms large. To measure the degree of uncertainty, our framework initially assigns the highest reputation score to all users. This allows easy access to all classes of shared resources. Furthermore, the framework keeps track of their subsequent misbehaviours based on the feedback received from the collaborating domains. The resultant reputation score, generated after a certain number of bad interactions, is used to estimate the uncertainty w_l related to the current request. Essentially, uncertainty prevails even if the requester belongs to a higher user security level with a moderate reputation score. For low reputation, the uncertainty is maximum and it amortizes non-linearly for requesters with a good reputation. We capture this nature by the following equations:

$$w_l = \begin{cases} 1 - e^{-\frac{\lambda}{\rho_{r_k}}}, & \text{if } \rho_{r_k} > 0 \\ 1, & \text{if } \rho_{r_k} = 0 \end{cases} \tag{3}$$

Table 1: Impact of Access Modes on Security Requirements

Access Mode	Object Type	$\mathcal{C}$	$\mathcal{I}$	$\mathcal{A}$
VIEW	Sensitive	1	0	0
EDIT	Sensitive	0	1	1
EDIT	Non-sensitive	0	1	1
EXECUTE	Sensitive	0	1	1
EXECUTE	Non-Sensitive	0	1	1

where, ρ_{r_k} is the reputation score for requester r_k, and $0 < \lambda < 1$ is the decay rate to be determined by the collaboration platform (refer to Appendix-D).

4.2.2 Modeling Expected Utility Function (w_s). Apart from the object sensitivity, the expected threat need to consider the impact of a requested action on the *confidentiality, integrity, and availability (CIA)* requirements of the information (object) hosted in the collaborating platform. To quantify the damage that is expected to be caused by misuse of the access modes, we customize the model proposed in [19]. We consider the objects belonging to *Top secret*, *Secret*, and *Confidential* classes as *Sensitive*, while those included in the *Unclassified* category are considered as *Non-sensitive*. Table 1 presents the impact of access modes in terms of three security tenets (confidentiality ($\mathcal{C}$), integrity ($\mathcal{I}$), and availability ($\mathcal{A}$)) when subjected to different object types. Here, 1 means the access mode has an impact on the considered attribute, and 0 otherwise.

For computing the risk of a request, the *R2Q* framework needs to quantify the damage it (request) is expected to cause (for the resource provider) in the context of misuse of the allowed access modes. The expected damage Δ must combine the probability of occurrence of the corresponding access mode p_{a_j} from the historical data and its impact on the security requirements of the provider domain (as depicted in Table 1) [19]. Thus, it is given as:

$$\Delta_{c_j} = (\mathcal{C} \cdot p_{a_j}) + (\mathcal{I} \cdot p_{a_j}) + (\mathcal{A} \cdot p_{a_j}) \tag{4}$$

Intuitively, the lesser the expected damage Δ_{c_j}, the lower will the malicious requester's utility be, thereby contributing lesser expected threat. Additionally, the utility likely to be gained will also depend on the risk perception of the collaborating domain. To model this nature, we use the following function:

$$w_s = 1 - e^{-\frac{\Delta_{c_j}}{\vartheta}} \tag{5}$$

where w_s is the expected utility the requester will gain by misusing the access mode, $0 < \vartheta < 1$ is called the *risk tolerance* parameter (refer to Appendix-D). This parameter models the platform's perception on the utility to be achieved by the requester. It also forms a basis of avoiding risk (termed as *risk aversion*).

4.3 Risk Related to Collaboration Request

In Eqn. (2), θ_{c_j} is the expected threat for the collaboration request c_j. Now, the collaboration platform needs to determine if the request is risky or not if the threat is actually exercised. This consideration is important as the collaboration yields utility in different manifestations viz., monetary gain, business goodwill, accomplishing a mission-critical task, establishment of reliability, satisfaction, etc. Hence, there exists a notion of loss and gain with collaborations, and the collaboration platform is not certain about the outcome.

In general, if the response/predictor variables are categorical (true/false, yes/no), the error distribution is non-normal. Thus, we need a link function to establish a relationship between the predictor variable (expected threat) and the mean of the distribution defining the risk. Normally, under risk-neutral case, a *logit* link function can be a possible option [3]. However, when it comes to trust relationships under risk and uncertainty with economic objectives, another factor to be considered is the risk tolerance attitude. Logit link function is inappropriate as it is symmetric around the origin and does not embed such an attitude while making decisions.

Therefore, we propose the use of *prospect theory (PT)* inspired link function to embed such risk tolerance. PT [13] models a decision maker's interpretation of risky prospects, the outcome of which may lead to losses and gains. We use the following properties of PT in this work:

Reference Point: A decision maker judges a prospect based on the potential gains or losses with respect to a *reference point*. It acts as a neutral boundary about which gains or losses of collaboration are visualized. In our case, $\theta = 0.5$ is the reference point.

Asymmetric S-shaped Value Functions: A decision maker is by default risk or loss averse, and he strongly prefers avoiding losses than achieving gains. As a result, the value function is *S-shaped* and *asymmetrical*. Mathematically, it is concave for gains, convex for losses, and steeper for losses than for gains.

In our framework, we establish a link between θ_{c_j} (normalized in the interval [0, 1]) and the risk related to the request c_j by customizing the value functions from PT [13]:

$$\mathcal{R}_{c_j} = \begin{cases} \theta_{c_j}^{\phi}, & \text{if } \theta_{c_j} \geq 0.5 \\ \theta_{c_j}^{\varphi}, & \text{if } \theta_{c_j} < 0.5 \end{cases} \tag{6}$$

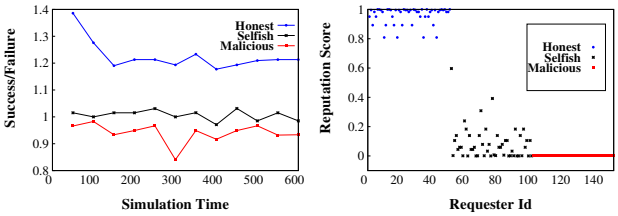
where, $\mathcal{R}_{c_j}$ is the risk score relevant to the request c_j and has value in the interval [0, 1], the value function exponents are $0 < \phi < 1$, and $\varphi > 1$. Expected threat values $\theta_{c_j} > 0.5$ and $\theta_{c_j} < 0.5$ instill the notions of losses and gains, respectively and are perceived as *more risky* or *less risky*. The parameter study has been done in Appendix-E.

5 PERFORMANCE EVALUATION

In this section, we evaluate the performance of the *R2Q* framework in terms of requester classification and successful authorization. We developed a prototype of *R2Q* framework using *Java* and simulated a realistic environment for collaboration platform with five participating domains, each consists of 100 users. Each user can share and give remote accesses to at most 10 objects. The simulator invokes the the *R2Q* framework to ascertain risk scores of requests. Table 2 presents the values of different system parameters used in performance evaluation.

Table 2: Simulation Parameters

Parameters	Values
$A/B/C$ (Gompertz's parameters)	1/10/0.3
λ (Uncertainty parameter)	0.45
ϑ (Risk tolerance parameter)	0.3
ϕ/φ (PT-value function exponents)	0.4/1.35
$p_V/p_E/p_X$	0.346/0.313/0.341



(a) Success/Failure in Auth. **(b) Requester Classification**

Figure 1: *R2Q* Performance

5.1 Success/Failure of Authorization

As mentioned in Section 4.1, the requesters will start with maximum reputation, but it is expected to gradually drop as more bad interactions are exercised. Thus, the initial risk score related to the requests may either increase or decrease depending on the reputation profile of the requester. We attribute such transitions of requests as success or failure of authorization in the following way:

Successful authorization: If the authorization is done for the following transitions - *high to low, high to medium, medium to low.*

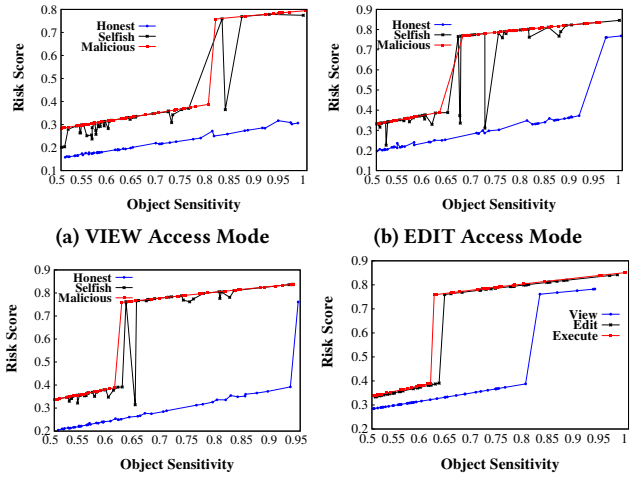
Failed authorization: If authorization is done for these transitions - *low to high, low to medium, medium to high.*

Fig. 1a shows the ratio of authorization success to failure for different requester types obtained over increasing simulation time. As evident, the success-failure ratio for honest requesters is significantly higher compared to that of the other requester types. Moreover, for honest requesters, the ratio is always greater than 1, implying a higher proportion of successful transition. Selfish and malicious requesters end up with a higher percentage of failed authorizations, depicting the fact that risk scores generated by the *R2Q* framework enable the platform to authorize the majority of non-risky requests. It is to be noted that there is a drop in the successful authorization for honest requesters at the later stages of the simulation. This is owing to the fact that bad interactions for this user class also increases over time, leading to a diminished reputation score.

5.2 Reputation-based Requester Segregation

One of the major challenges for the collaboration platform is to classify honest, selfish, and malicious requesters to ensure fair authorization decision. Such classification requires observing the requesters' behaviours over time and then segregating them based on the reputation scores.

Fig. 1b shows the average reputation scores for a sum of 150 honest, selfish, and malicious requesters, evaluated after running the simulation ten times for a fixed time (200-time units) over a predefined request set. It is noteworthy that the reputation of honest users are significantly higher than the selfish ones, while the scores for malicious users drop to zero due to a higher frequency of bad interactions. One or more selfish requesters end up with relatively higher scores due to their sporadic misbehaving nature. Low reputation will result in a higher risk score for the collaboration request initiated at the current time instant. Nevertheless, such reputation-based user classification facilitates in making authorization decisions.



(a) VIEW Access Mode (b) EDIT Access Mode

(c) EXECUTE Access Mode (d) Overall Access Mode Risk

Figure 2: Effect of Access Modes on Risk Score

5.3 Effect of Access Modes on Risk Scores

As stated earlier we modelled the expected damage to be caused if VIEW, EDIT, and EXECUTE access modes are misused on objects with different sensitivities. However, the risk posed by such misuses will also vary with the reputation profile of the requester.

Figs. 2a, 2b, and 2c show the variations of risk scores for different object sensitivities under three reputation profiles (honest, selfish, malicious). In all the access modes, the degree of risk posed by the honest requesters is substantially low compared to that of other requester types. Also, the risk posed by VIEW operation is comparatively less than the other two. This is because the effect of abusing VIEW operation (unauthorized disclosure) on sensitive objects is not instantaneous, has long-term implications, and difficult to quantify. It can be observed that the risk values for the requests from malicious requesters peaks at a lower object sensitivity values than that for selfish users. Thus, *R2Q* framework ensures that the collaboration platform prevents malicious requesters from giving access to sensitive objects.

In Fig. 2d, we computed the average risk yielded by aggregating the three access modes across all requester types. As expected, the risk scores for EDIT and EXECUTE peak at lower object sensitivity values compared to VIEW. The extent of instantaneous damage caused by VIEW is on the lower side compared to the other two access modes. Thus, for requesters with average reputation, the collaboration platform can prefer authorization of VIEW access to highly sensitive objects than either of EDIT or EXECUTE accesses.

6 CONCLUSION

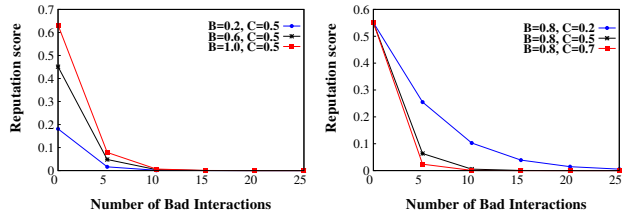
In web-based multi-domain collaborations, the service provider authenticates a remote requester by verifying the latter's access token information but is uncertain of the actual usage of the shared objects. Thus, there is a need for an additional authorization layer based on soft-security factors such as risk, reputation, etc. In this paper, we propose a novel risk quantification framework, *R2Q*, which incorporates multiple attributes specific to both the shared object and requester and exploits *Prospect Theory*-value functions to compute risk score. Such score facilitates access decisions under conditions of risk and uncertainty. A prototype implementation of the framework is done and simulation-driven experiments are conducted for performance evaluation. For future work, we will model the misbehaviour of requesters by studying variances in their historical request patterns, rather than depending on the feedback obtained from the resource owner. Further, in the event of bursty traffic (multiple requests received at the same time), we extend the capability of our framework in the selection of an optimal set of requests, having lower risk and higher service benefit.

ACKNOWLEDGMENTS

The authors acknowledge the support of IIEST Shibpur, JP Morgan Chase & Co., and Missouri S&T to carry out this research. The work of S.K. Das is partially supported by NSF grants under award numbers CNS-1818942, CNS-1545050, and CCF-1725755.

REFERENCES

[1] N. Baracaldo and J. Joshi. 2013. Beyond Accountability: Using Obligations to Reduce Risk Exposure and Deter Insider Attacks. In *Proceedings of ACM SACMAT*. 213–224.
[2] D. E. Bell and L. J. LaPadula. 1973. *Secure Computer Systems: Mathematical Foundations.* Technical Report MTR-2547, Vol-1. MITRE CORP, BEDFORD MA.
[3] S. Bhattacharjee, N. Ghosh, V. K. Shah, and S. K. Das. 2017. QnQ: A Reputation Model to Secure Mobile Crowdsourcing Applications from Incentive Losses. In *Proceedings of IEEE CNS*. 1–9.
[4] M. Bishop. 2002. The Art and Science of Computer Security. (2002).
[5] S. Chakraborty and I. Ray. 2006. TrustBAC: Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems. In *Proceedings of ACM SACMAT*. 49–58.
[6] P. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger. 2007. Fuzzy Multi-level Security: An Experiment on Quantified Risk-adaptive Access Control. In *Proceedings of IEEE SP*. IEEE, 222–230.
[7] K. Djemame, D. Armstrong, J. Guitart, and M. Macias. 2016. A Risk Assessment Framework for Cloud Computing. *IEEE Transactions on Cloud Computing* 4, 3 (2016), 265–278.
[8] D. R. dos Santos, R. Marinho, G. R. Schmitt, C. M. Westphall, and C. B. Westphall. 2016. A Framework and Risk assessment Approaches for Risk-based Access Control in the Cloud. *Journal of Network and Computer Applications* 74 (2016), 86–97.
[9] D. R. dos Santos, C. M. Westphall, and C. B. Westphall. 2014. A Dynamic Risk-based Access Control Architecture for Cloud Computing. In *Proceedings of IEEE NOMS*. 1–9.
[10] D. Fall, T. Okuda, Y. Kadobayashi, and S. Yamaguchi. 2016. Risk Adaptive Authorization Mechanism (RAdAM) for Cloud Computing. *Journal of Information Processing* 24, 2 (2016), 371–380.
[11] B. Gompertz. 1825. On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies. *Philosophical Transactions of the Royal Society of London* 115 (1825), 513–583.
[12] C. Jason. 2004. *HORIZONTAL INTEGRATION: Broader Access Models for Realizing Information Dominance.* Technical Report 500Technical Report JSR-04-132. MITRE Corporation.
[13] D. Kahneman and A. Tversky. 1979. Prospect Theory: An Analysis of Decision under Risk. *Econometrica* 47, 2 (1979), 263–291.
[14] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf. 2011. *NIST Cloud Computing Reference Architecture.* Technical Report 500. NIST, US Department of Commerce.
[15] I. Molloy, L. Dickens, C. Morisset, P.-C. Cheng, J. Lobo, and A. Russo. 2012. Risk-based Security Decisions Under Uncertainty. In *Proceedings of ACM CODASPY*. 157–168.
[16] H. Mousa, S. B. Mokhtar, O. Hasan, O. Younes, M. Hadhoud, and L. Brunie. 2015. Trust Management and Reputation Systems in Mobile Participatory Sensing Applications: A Survey. *Computer Networks* 90, 2 (2015), 49–73.
[17] Q. Ni, E. Bertino, and J. Lobo. 2010. Risk-based Access Control Systems Built on Fuzzy Inferences. In *Proceedings of ACM ASIACCS*. 250–260.
[18] R. A. Shaikh, K. Adi, and L. Logrippo. 2012. Dynamic Risk-based Decision Methods for Access Control Systems. *Computers Security* 31, 4 (2012), 447–464.
[19] M. Sharma, Y. Bai, S. Chung, and L. Dai. 2012. Using Risk in Access Control for Cloud-Assisted eHealth. In *Proceedings of IEEE HPCC-ICESS*. 1047–1052.



(a) Displacement Parameter: B **(b) Growth Parameter: C**

Figure 5: Parameters: Inverse Gompertz Function

A SYSTEM MODEL

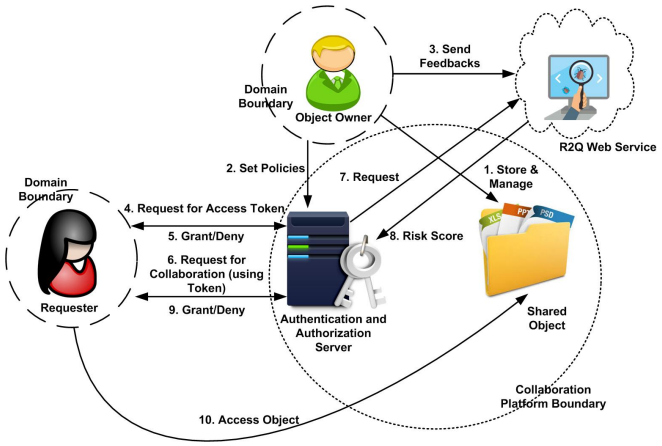


Figure 3: System Model

B MODULES OF R2Q FRAMEWORK

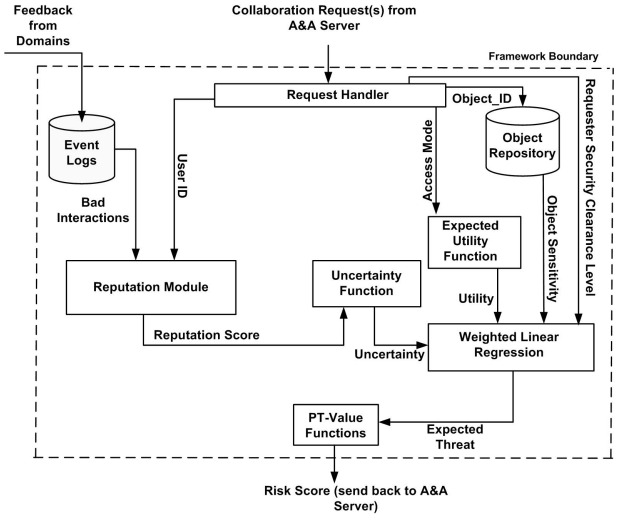


Figure 4: *R2Q* Framework: Modules & Control Flows

Request Handler: This module accepts collaboration requests from the authentication and authorization server (A & A) for estimating the risk of a request. It extracts the identities of the object and the requester, its security clearance level, and access mode.

Object Repository: It contains the sensitivity of the objects provided by the owners across all participating domains at the time of sharing. Object sensitivity depends on the linearly (total) ordered security classes [3]: *Top Secret* ≻ *Secret* ≻ *Confidential* ≻ *Unclassified*. Thus, the higher the security class, the greater is the sensitivity, and the larger is the need to prevent misuse.

Event Logs: It is also a repository from which the *R2Q* framework can retrieve the activity logs specific to a requester, such as the number of times he has interacted with different resource owners, the types of requests made, the number of times the requester has been found to misbehave, etc. The information in this repository is partially obtained from the feedbacks submitted by the resource providers after completion of collaborations.

Reputation Module: We use inverse Gompertz function [12] to model the requester's reputation. Traditionally, Gompertz function models time-varying entities, such as the population growth in a confined space or market impact on finance, where growth is slowest at the start and end of a time period. Moreover, it has been used in generating cumulative reputation in other paradigms, such as participatory sensing, online social networks, and so on [18]. To address the *cold start* problem, we adopt a liberal approach by assigning a maximum reputation to all new requesters and penalize them only if they have been reported to misuse the shared resources. Otherwise, the requester continues with maximum reputation and all legitimate accesses on any object are permitted. The reputation function accepts the number of misbehaving interactions related to a particular requester as the input, and then outputs his reputation score.

Expected Utility Function: It estimates the expected utility which a potential malicious requester will gain by misusing the allowed access modes on the requested object. The utility is weighed by the object sensitivity in the linear regression model to compute the expected threat related to the request.

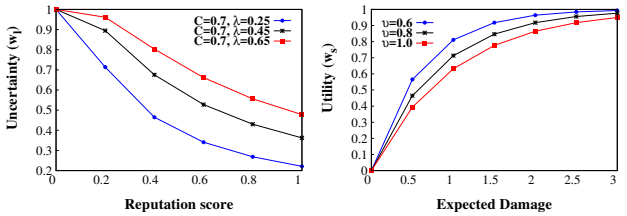
Uncertainty Function: It is used to model the security uncertainty from a resource owner's perspective with respect to the requester's current reputation. It is weighed by requester's security level to magnify the threat posed by privileged users.

Weighted Linear Regression: It computes a weighted sum of the degree of uncertainty due to requester's usage history, and his/her expected utility if the given access mode is misused. The resultant sum generates the expected threat related to a collaboration request.

Prospect Theory (PT)-Value Functions: Prospect Theory (PT)-value functions [15] will be used as a link function to transform the expected threat to the corresponding risk values. Such mapping enables us to replicate practical decision making in the presence of economic losses and gains.

C INVERSE GOMPERTZ FUNCTION PARAMETERS

Fig. 5a illustrates the effect of parameter B on the initial value of the reputation, before the latter enters into the exponential decay phase. For a fixed decay rate, this parameter acts as a discounting factor to the maximum reputation score. The choice of its value depends upon if the collaboration platform is risk seeking (corresponds to high B) or risk-averse (corresponds to low B).



(a) Uncertainty Parameter: λ **(b) Risk Tolerance Parameter: ϑ**

Figure 6: Parameters for w_l and w_s

Parameter C controls the rate of decay of the reputation scores, lower bounded by 0, as shown in Fig. 5b. For higher values of C, the reputation score drops rapidly with relatively fewer bad interactions. For an "open" collaborating platform, it can choose a smaller C. Such system tolerates a larger number of bad interactions before their reputations drop significantly.

D PARAMETERS FOR w_l AND w_s

Fig. 6a illustrates the decays of uncertainty for different values of λ. Higher values of λ reduces the uncertainty at a rapid rate. In situations where the domains need to interact for emergency requirements, the collaboration platform can maintain a high λ so that the requester, with average reputation, is allowed to access the shared resources. However, for domains which share sensitive information, λ can be set to low values to ensure that the uncertainty amortizes only if the reputation remains persistently high.

Fig. 6b shows the utility which a potential malicious requester can gain by misusing shared objects, having different sensitivities. The risk tolerance parameter ϑ controls the rate of growth of the utility. Lower ϑ (i.e., low-risk tolerance level) is reflective of a conservative collaboration platform, which perceives the higher utility of the requester from less sensitive assets. This may result in non-sharing of sensitive information. However, for mission-critical applications, the collaboration platform can increase the risk tolerance level (by setting a higher value of ϑ), thus enabling accesses to different objects (low or high sensitivity).

E PROSPECT THEORY PARAMETERS

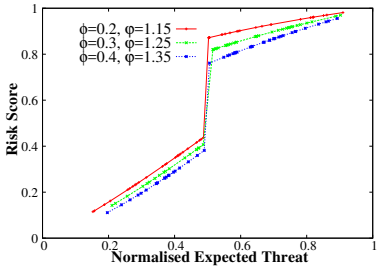


Figure 7: Prospect Theory Value Function Exponents

Fig. 7 shows the functions to map the expected threat to the corresponding risk score. The exponent ϕ enables the collaboration platform to control the rate of growth of the risk above the reference point and it also allows the adaptation of different decision making attitudes. For instance, if the platform wants to prevent the misuse of resources, it prefers rapid increment of risk w.r.t. the expected threat (red and green curves: *concave* nature) which results in denying the requested access. This entails a *risk averse* attitude and it has an implicit effect on collaboration. However, to boost up collaboration, the platform can always tune the exponent such that the threshold risk (for designing access/deny policies) is attained at higher threat levels (blue curve).

In contrast, the nature of risk below the reference point is always *convex* and its steepness depends on the exponent φ. This is because, the platform is expected to exhibit *risk seeking* attitude to the requests with low expected threat values, which results in relatively low risk, thus enticing the platform to allow collaboration.