Using Reputation to Augment Explicit Authorization

Phillip J. Windley Ph.D., Devlin Daley, Bryant Cutler, and Kevin Tew
Dept. of Computer Science, Brigham Young University
Provo, UT, USA
http://xri.net/=windley, devlin.daley@gmail.com, bryant.cutler@gmail.com,
kevin@tewk.com

ABSTRACT

Online social networks are formed when web applications allow users to contribute to an online community. The explosive growth of these social networks taxes the management capacity of human administrators. The continued health of an online social network depends upon the identification and utilization of users who make positive contributions to the community, but finding these individuals can be difficult. In addition, these contributing users must be *explicitly* granted authority to help maintain and grow these networks.

Automated reputation calculations based on user contributions and behavior can be used as an effective substitute for explicit authorization, giving online social networks greater flexibility and scalability. In this paper, we examine the underlying principles of online reputation, introduce *Pythia*, a flexible reputation system framework, and demonstrate the use of reputation calculations to augment explicit authorization in a web application.

Categories and Subject Descriptors

H.4.3 [**Information System Applications**]: Communications Applications; K.4.2 [**Computers and Society**]: Social Issues; H.4.m [**Information System Applications**]: Types of Systems; Decision support

General Terms

Algorithms, Human Factors

Keywords

reputation, identity, framework, blogging

1. INTRODUCTION

Most online interactions are devoid of many of the cues that people use in the physical world to make judgments about the character, stability, reliability, etc. of people, systems, and other entities. Yet these cues are critical to sophisticated interactions and transactions. Online reputation systems attempt to remedy this situation.

Reputation has been studied in the context of philosophy [12], psychology [3], economics [16], biology [26], and sociology [27]. Much of that work is informative of how reputation works in the physical world and can guide our thinking about reputation online.

Online reputation can be accessed informally or formally. Informal assessments are made by people from their experience on the Internet. For example, I may not trust email I get from the domain `hotmail.com` because I've received substantial Spam from addresses in the domain in the past. Formal systems collect and process online information about specific identifiers in an attempt to calculate reputation scores in a more systematic manner.

Computational models of reputation are quite recent[23], yet diverse strategies have been employed to meet specific needs. In *Reputations Systems: Facilitating Trust in Internet Interactions*,[21], Resnick et al. define a reputation system as a system that "collects, distributes, and aggregates feedback about participants' past behavior."

The primary domains where computational reputation has been explored are in artificial intelligence, peer-to-peer networking, and electronic commerce. A number of systems have been devised to provide reputation information for commercial Web sites. [8] Most of these systems have limited scope and are built for a single context. [19] They are effective in their specific domain, but lack generality.

Recent advances in user-centric identity systems such as CardSpace and OpenID provide a firm foundation for the establishment of general-purpose, online reputation systems [5, 20], which depend upon generalized identity systems for correlation of user actions. In many situations where explicit authorization is either impossible or infeasible, or even in situations where explicit authorization is currently in use, replacing or augmenting human-mediated authorization with calculated reputations scores can make an application more flexible and scalable.

This paper first presents a set of core principles that we believe should govern the operation of reputation systems. We detail the design of *Pythia* a generalized framework for creating reputation systems. We use *Pythia* to demonstrate the replacement of explicit authorization for blog comments with a customized reputation calculation.

2. REPUTATION PRINCIPLES

We believe there are important principles that apply to reputation systems and should govern their design and op-

DIM'07, November 2, 2007, Fairfax, Virginia, USA.

eration. Understanding these principles is vital to the success of any computational reputation system. We base these principles on our review of the literature, our own research, and a Berkman Identity Mashup group discussion. [29]

Reputation is one of the factors upon which trust is based. [21, 15] People often confuse trust and reputation in casual conversation. Trust is "the expectation of one person about the actions of others that affects the first person's choice when an action must be taken before the actions of others are known."[6] Reputation is a basis for trust.

Mui et al. presents a computational model of reputation that explicitly accounts for trust. [18] Mui's model uses reputation and the history of encounters between agents to calculate a value representing the expectation that one agent will reciprocate another agents actions.

Trust is important in many online systems, implicitly or explicitly[28], but is beyond the scope of this paper. Consequently, we are not concerned with what factors beside reputation go into trust, how trust is built, or how trust is exchanged–we are focused solely on reputation.

The expectation of future reciprocity or retaliation creates an incentive for good behavior in the present. [21] Obviously the party making the reputation request is doing so in an attempt to gain information that will guide some action. However, measuring reputation also affects the behavior of the targets of reputation requests. Axelrod calls this the "shadow of the future." [2]

When present actions will affect future judgments that others make, present behavior will change. For a reputation system to be effective, recorded feedback on past interactions must influence the actions of entities in the future. Put another way, entities must pay some attention to reputations in the system during their decision making process or it will cease to induce desired behaviors.

Reputation is personal. Fundamental to the operation of a reputation system is the notion that agents compute the reputation for other agents based on certain factors. [18] While one can control some of the factors upon which reputation is based, one cannot control how those factors are used by another agent in assessing reputation.

This principle leads to the conclusion that reputation computations should be personalized, so that each agent can compute a different reputation for another based on the information that it has available. In Sabater and Sierra's words, "we cannot talk about the trust/reputation of an individual x, we have to talk about the trust/reputation of an individual x from the point of view of individual y. "[23]

Reputation is a currency. While you can't change reputation directly, reputation can be used as a resource. For example, Paul Resnick *et. al.* has shown that a positive eBay reputation has real economic value. [22]

Reputation is narrative. Put another way, reputation varies with time. Reputation is dynamic because the factors that affect it are always changing. One of the problems with many existing reputation models is that reputation is treated as uniform across time. [18]

Reputation is based on identity. Reputation isn't part of an identity, but is *about* an identity. Resnick et al. states that entities in the system *must* be long lived in order to ensure a expectation of future interactions. [21] We distinguish between *identity* and *identifiers.* Any given entity may be known by any number of handles or identifiers.

Linking these identifiers brings together all of the information associated with them for subsequent reputation calculations. This linkage reduces or even prevents identity switching [11] where subjects of bad reputation judgments simply create a new pseudonym to avoid any negative consequences. By using broad-based identity, with linked identifiers, as a foundation for the reputation calculation, the cost of switching is increased.

Reputation is based on verified claims and transactions. Resnick et al. discuss how trust grows in social situations where by interacting with a person over time, a history of past interactions is built that characterizes an individual's character, abilities, and disposition. [21] Reputation systems collect and record feedback concerning interactions. This information is distributed to entities in the future.

We used these terms in the following ways. *Claims* are facts about an identity that can be verified. A claim may be something as simple as an email address or a URL. A claim could also be a fact from some online resource. The number of years that a domain name has been registered, the ratio of inbound to outbound email from an account, or the classification of a URL by a content analysis engine are examples.

Transactions refer to records of interactions between any two entities. Transactions facilitate both the direct and observed interactions that Sabater and Sierra discuss. [23] Direct interactions are those where you are one of the participants. Observed interactions are those that are visible to non-participants.

Most computational reputation systems take direct interactions into account, only a few allow for observed interactions. Note that observed interactions are not the same as opinions. Observed interactions are more reliably objective since they record facts about an interaction, not judgments about it.

Since reputation is personal, how these factors are used in determining reputation is individually determined. Individuals may use various evidence in making claims or proposing a certain rating or endorsement. The penalty for making false claims or giving false endorsements varies from context to context.

Lin et al. assert that trust should be based on verifiable identification of the subject entity, qualification of the entity to perform the requested service, and the subject entity's consistency of performance. [15] Their system, which is built for peer-to-peer reputation management, bases reputation on performance metrics from reported by a service's peers.

Reputation is based on opinion (indirect information from other witnesses). [23] A reputation isn't just based on facts, but is also based on other's beliefs about the subject of the reputation calculation. These beliefs are signaled to others in various ways depending on the context. Studies of reputation in social networks show the value of including the opinions of other entities. [27]

As we use the term, *opinions* are the indirect information that other entities may supply about the reputation subject. These may be ratings, endorsements, or other subjective judgments about an entity or a past transactions involving an entity. Naturally, a certain amount of uncertainty surrounds this information. Using it involves making more complicated judgments about its veracity. This makes reputation computations using opinions more costly.

Resnick et al. explain some of the challenges reputation systems face–many of those challenges are directly related to the use of opinion. [21] There is a human reluctance to give negative feedback. In order to mitigate losses and failures, individuals turn to negotiation and arbitration in place of giving negative feedback. In some systems, fear of retaliation leads to an acceptance of mediocrity. Finally, disgust and desires to avoid further confrontation may cause individuals to leave bad service in the past and move on without leaving negative feedback.

Reputation exists in the context of community. Any given context will have specific factors for what is important in determining reputation. Community provides a context for evaluating opinions and indirect information.

Sabater et al. discuss this issue under the heading of "visibility types" and classify the reputation of an individual as "global" or "subjective."[23] Global reputation is a single, universal value calculated from all the transactional data available to the system. Subjective reputation is calculated for pairs of individuals based on their interactions and the *opinions* of others.

Global reputation is often insufficient because of a lack of personalization. But Sabater et al. make the point that subjective systems require strong links between individuals so that information is shared frequently, providing a large body of information to serve as the basis reputation decisions.

But the need to collect large amounts of direct interactions between pairs of individuals can be mitigated through the increased use of observed interactions. That way the global set of transactions involving the reputation subject is available to any one individual, greatly increasing the probability that a sufficiently large set of data will be available for a reputation calculation.

Reputation exists in a particular context. When we speak of reputation we say things like "she has a reputation for ..." The fact that someone is a good plumber doesn't mean they will be a reputable babysitter. One of the failings of many current computational models for reputation is that they fail to take context into account or are only useful in a single domain and cannot be repurposed. [18]

There is a natural tradeoff between reputation and privacy. As noted above, reputation is calculated, in part, from a record of past interactions. These interactions record information that individually or in aggregate could threaten the privacy of the subject. People make the trade-off between reputation and privacy in the physical world every day–giving up some privacy to get a credit report, for example. Reputation systems should provide users with information and choice about this tradeoff.

The quality of a reputation algorithm should be regularly assessed. As stated, reputation algorithms take into account a number of factors. The goal of a reputation algorithm is to process identity transactions into a reputation that is easy to work with. However, simply returning a score can lead to false confidence. For example, suppose that reputation is based on a history of past interactions between only two parties. If meta data describing transactions is not available, scores based on this data might inappropriately carry as much weight as other calculations that utilize more comprehensive transaction data.

In *A Reputation and Trust Management Broker Framework for Web Applications*, Lin et al. describe a multi-level hierarchy for calculating reputation in a distributed way in peer-to-peer networks. [15] Examining Lin's work leads us to two conclusions:

1. Reputation response should have an accompanying audit trail of external brokers and services used to calculate the reputation.
2. Confidence measures such as the quantity of feedback transactions used during a reputation calculation can be used to evaluate the quality of a reputation framework response.

3. AUGMENTING AUTHORIZATION

Authorization systems are designed to meet conflicting requirements, including ease-of-use, security, and scalability. A broad spectrum of authorization systems has resulted; each system makes trade-offs to achieve specific goals. Some typical styles of authentication are as follows:

- *No authentication* Most common in public wikis, these systems grant privileges even to anonymous, non-authenticated users. This approach scales well, because no additional work is require besides providing application functionality; however, it is suitable only for the most open environments.
- *Authentication as authorization* Every authenticated user is granted full privileges. Many of these systems make user accounts freely available, but authentication is retained to allow auditing and sanctions against specific users. These systems are similarly easy to scale, because the creation of new accounts is easily automated.
- *Explicit authorization* Authenticated users are explicitly granted specific privileges, which are typically encoded as access control lists (ACL), capabilities, or roles (RBAC). These systems require human administrators to manually grant privileges to users, and are thus not suitable for self-service internet-scale applications.

We have attempted to bridge the gap between *authentication as authorization* and *explicit authorization*, in a way that maintains the security properties of explicit authorization without suffering from decreased scalability. In many cases, a reputation-based authorization scheme is a relatively straight-forward automation of a human reputation calculation. As a result, reputation-based authorization can retain the security characteristics of human-mediated systems.

3.1 Benefits

Web applications that replace explicit authorization reap several important benefits. The first is simplicity. By using a reputation service to calculate reputations on demand, application modularity increases in a way similar to aspect-oriented programming. Authorization is a cross-cutting concern, so it is well-suited to abstraction as a globally-available service. Calculating reputation on-the-fly also reduces the data storage overhead of the application, since explicit authorizations are not stored in user records.

Another significant benefit of using generalized reputation systems is that users can carry their reputations between contexts. While cross-context reputation applicability is a difficult problem we leave for future work, in many systems external reputation is enough at least to bootstrap new users into approximately the correct level of authorization. Game-theoretic models show [11] that the most efficient strategy for isolated communities is to give newcomers the minimum possible level of trust. By communicating with other contexts (by allowing some external reputation data) services avoid forcing new users to rebuild their reputation from scratch; this can increase the service's appeal, especially to users with high reputation (the users most likely to make sizable positive contributions to the social network).

Using calculated reputation scores rather then explicit authorization better matches the organic way that large-scale social web applications change over time. Because the authorization system is automated, it can handle start-and-stop growth spurts better than a system that relies on teams of human administrators. Additionally, the importance of reputation data can be made to gradually decay with time, so that long-forgotten back-doors or previously authorized users don't pose the security risks: only active, contributing users increase maintain their privileges over time.

3.2 Detrimental aspects

While reputation-based authorization retains many of the security benefits of explicit authorization, and is far more scalable, it has some significant problems. The biggest problem with trusting reputation systems in this way is that they are inherently inexact, based as they are upon inferring trends from user actions and feedback. This drawback manifests as false positives (authorization given to undeserving users) and false negatives (authorization denied to deserving users) when the defects in the reputation calculations are unintentional; when users intentionally act in a way that increases their reputation out of proportion to their contribution to the community, they are "gaming the system." These improper authorization decisions, intended and unintended, weaken security, introduce hard-to-quantify risk, and lead to complex liability and governance issues.

Even when reputation systems are not themselves the source of exploits, basing authorization on reputation calculations can be problematic in some domains. Reputation is simply not applicable in many static environments (i.e. the relationship between two major corporations) or in regulated environments (i.e. reputation scores are largely irrelevant when determining who should have access to medical records). In other domains, reputation may not be well-defined; there may be no consensus about which user actions are positive and which are negative.

4. PYTHIA

After surveying many of the existing reputation frameworks employed on the web, we discovered several common properties. Most reputation systems can be decomposed into storage of user transactions and algorithms operating on these transactions to produce a score. Action is then taken depending on the value of the score.

By building a flexible system that employs these common traits, these reputation systems and others can more easily be developed. The goal of *Pythia* is not to present a novel computational model for reputation. Rather, our goal is to provide a framework upon which such computational models can be built and evaluated.

4.1 Design Philosophy

Beyond the reputation principles discussed previously, we also created a design philosophy that guided the system architecture. The design philosophy is intended to be consistent with the principles espoused in the previous section and to put stakes in the ground on specific design issues.

***Pythia* supports multiple computational models for reputation.** There has been much research in the area of computational reputation models over the last decade. Sabater et al. gives a review of a dozen such systems. [23] Our goal in building *Pythia* was to create a system that supported many of these models and thus to provide a platform for future research in computational reputation.

***Pythia* pragmatically supports computing reputation from information already available online or information that is easily gathered**. Many of the available reputation systems are designed to work in a closed or specialized environment and thus exploit methods of interaction and observation that are not common interaction idioms on the Internet. [1, 9, 24, 10, 25, 4] In keeping with our goal to build a broad-based, general-purpose framework where reputation calculations can be performed on the emergent behavior of a large number of participants, *Pythia* employs common online identifiers such email addresses and URLs, uses online resources to verify those claims and generate additional facts about them, and captures transactional data automatically from systems through a Web-based API.

***Pythia* supports reputation computations for people**. As mentioned above, our focus has been leveraging emerging wide-area identity systems as a foundation for reputation. This goal influenced *Pythia* design choices in a way that favors reputation calculations for human subjects. The choice of which identifiers to support and how they're verified is an example of such a decision.

In the discussions that follow, we will refer to two groups of entities with respect to *Pythia*. The first is the group we call *users*. Users are the subjects of a reputation request. As we'll see, a reputation request instigates a context-sensitive reputation calculation. *Pythia* aggregates information about users and uses that information in processing reputation requests.

The second group is a subgroup of the users that we call *relying parties*. A relying party is a user making a reputation request about another user. Any user can be a relying party without any additional provisioning in the system.

***Pythia* uses aggregation rather than game-theoretic or stochastic methods to calculate reputation**. Reputation systems typically use either what Sabater and Sierra call "cognitive" or "game-theoretical" conceptual models. [23] Cognitive methods take into account the "mental states that lead [one agent] to trust another agent or assign a reputation." At present, *Pythia* does not use stochastic computation to any significant degree. Rather reputation scores are based upon the "weight of evidence" represented by the aggregated information in the database.

Pythia's use of aggregate data is similar to that of Sierra, the reference implementation of OpenPrivacy.org's Reputation Management System. [13] Sierra uses opinion objects, which are roughly comparable to the transactions in *Pythia* to calculate reputation. "OpenPrivacy's reputation management system can assemble a set of related opinions into a bias... often, a bias may consist of opinions from multiple nyms."[14] Biases are similar to *Pythia* rule sets (see below).

Users are identified by multiple identifiers. Multiple identifiers (email addresses, URLs, phone numbers, etc.) can be associated with a single user. The identifiers are validated through mechanisms like email challenges. As discussed above, linking these identifiers is an important method for increasing the body of information available about any given user for a reputation calculation. In *Pythia* users choose whether to maintain their privacy by keeping identifiers separate or to increase their reputation by linking them together.

Reputation is a personalized function of claims, transactions, and opinions. For our purposes, reputation can be represented as pseudo-mathematical expression:

$$Rep_u = F_{rp}(I_u, Tx_{u,u'}, O_{u,u'})$$

where

F_{rp} is the relying party's reputation algorithm
u is the user id
u' is all users
I_u is a vector of verified identifiers and other claims for u
$Tx_{u,u'}$ is a vector of transactions between u and every other user in the system
$O_{u,u'}$ is a vector of opinions about u

F_{rp} is personalized to each relying party. One of our design goals it to create a system that supports each relying party creating their own F_{rp}. In practice, each relying party can define multiple F_{rp} functions and chose which is used at the time a reputation query is made.

The choice of vectors in the foregoing expression over sets is meant to indicate that these values are ordered according to the time of their collection. F_{rp} can use all of the information in any given vector or can exclude values from vectors by employing filters to select appropriate values for the calculation.

Pythia does not require that reputation be a single number. It's possible that the function could return a vector of values if that was needed. This allows a single reputation query to return reputation "fitness for purpose" parameters for multiple contexts.

The vector I_u corresponds to the notion of *claims* from the last section. Any number and type of identifiers can be used in a reputation calculation. In addition, properties of identifiers, where available, can also be used. For example, a domain name that has been in use for a longer period of time may indicate a more trustworthy Web site since the owner has a large stake in its future viability. An email address with a ratio of inbound to outbound email messages significantly less than 1 might indicate an address that is used to send Spam.

Transactions and opinions we discussed in the last section. Since opinions are subject to manipulation "it is far more complex for trust and reputation models to use [them]. ...[W]itnesses manipulate or hide pieces of information for their own benefit". [23] Because of this complexity the current implementation of *Pythia* does not use opinions in a significant way. Future work will incorporate opinions into the reputation computation.

Transactions are jointly owned by the parties to the transaction. The exact nature of the transactions in the system depends on the particular use to which it's being put. Transactions always contain an identifier for each party to the transactions and a timestamp in addition to the facts pertinent to the transaction as defined by the domain.

Ownership implies that all parties to a transaction can *see* it, but not change it. Other users, not party to the transaction, can use it in a computation, but cannot inspect the details of the transaction.

Transactions are persistent and immutable. Once the system has recorded a transaction, it cannot be deleted by any of the parties to the transaction. A later transaction may record that it was nullified, but cannot remove it.

Pythia is designed to be user-centric, meaning that users have control over the reputation data they report and the algorithms they use for reputation calculation. However, one of our principles of reputation is that it is "your story about me." Unlike many facets of identity, users should *not* be able to directly modify transaction records; the reputation data is jointly owned by all the parties involved in the interaction, and must be preserved for reputation calculations to be meaningful.

Transparency. *Pythia* is designed to favor transparency wherever possible. Consequently,

- Users can see any information the system knows about them (e.g. transactions they are party to)
- Users can see any reputation queries about them, what the results were, and how the result was arrived at.
- As noted, users who are *not* party to a transaction cannot see it, but can use it to calculate reputation about any of the parties to the transaction.

This opens the possibility of information leakage and a subsequent, unintended loss of privacy. Transaction details could be "tapped out" through repeated interactions with the system. We believe that this danger is mitigated by the benefits that transparency offers.

Online resources should be consulted whenever possible to garner reputation information. As we've noted, claims about identifiers are validated in various ways. For example, an email address can be validated by sending a unique URL to the address that the user clicks on to validate receipt. URLs can be validated by having the user embed a unique code in the HTML for the page. [17]

In addition, data stores such as the `whois` database and Netcraft.com can be used to gather information about domains and Web sites respectively. These online sources of information provide facts that can be used as evidence in a reputation computation. Facts provide a means of assessing reputation that is based on information that is vouched for by a third party.

***Pythia* is centralized.** We made this decision primarily for the sake of simplicity. In Lin's distributed framework [15], for example, a reputation broker first attempts to answer a user's reputation requests based on the broker's local reputation database. If a broker lacks sufficient local feedback to make a recommendation it contacts peer brokers for reputation information. If the peer brokers lack sufficient feedback to return a reputation, a broker can contact institutional reputation authorities before resorting to untrusted sources. Consequently, Lin's hierarchy of diminishing trust is complex:

```
user ->
  broker ->
    peer brokers ->
      reputation authority
```

In contrast, *Pythia*'s model is much simpler, having only users and relying parties both interacting with a centralized reputation authority. The *Pythia* server answers reputation queries based only on the feedback that it has collected and publicly available online information. *Pythia*'s collected feedback may be thousands of transactions, ten or twenty transactions, or none at all. Additionally, centralization eases management and significantly improves performance in our experimental implementation due to the lack of multiple network requests cascading from an single query.

As a result of its centralized architecture, *Pythia* faces all the same costs as any centralized system including reliability concerns associated with a single point of failure, security and privacy concerns for having a large collection of information all in one place, and the scaling issues surrounding any large system dependent on a database. Similarly, these issues in *Pythia* may be addressed by the same techniques applied to any other large centralized system. We leave for future work the development of *Pythia* into a distributed online reputation system.

4.2 Architecture

The architecture of *Pythia* is based on the foregoing principles and design philosophy. Consequently,

- *Pythia* is identity system neutral and stores multiple identifiers for any user; correlation of actions by a user utilizing multiple identifiers enables more meaningful reputation calculations.
- *Pythia* uses a rules engine to allow each relying party to make customized queries against the transaction store. Relying parties thus use reputation algorithms tailored to their own needs.
- *Pythia* uses context-specific plug-ins to acquire reputation data from other online applications.

Figure 1 shows the high-level block diagram architecture for *Pythia*.

4.2.1 Identity Subsystem

Pythia is not an identity system, but is meant to rely on one or more existing identity systems for authentication and user IDs. As implemented, *Pythia* uses OpenID as an authentication mechanism, but other authentication systems could also be used.

The framework allows users to claim and verify identifiers from identity systems of all kinds. At present, these identifiers are limited to email addresses and URLs, but other types of identifiers could be supported without changing the underlying system significantly. Email addresses are verified by sending the email address a challenge message asking the user to click on a URL to claim the email address in *Pythia*.

URLs are verified using MicroIDs [Mic06], a standard for claiming online resources. A MicroID is a secure hash of an email address and a URI. A MicroID is not used to prove someone owns a resource (since anyone with control of the resource could insert the MicroID), but rather to verify claims to a resource.

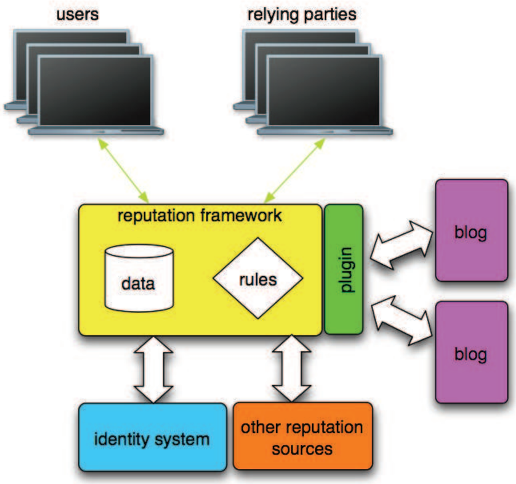


Figure 1: Reputation Framework Architecture

Pythia generates a MicroID for any claimed URL based on a verified email already in the system for that user. The system instructs the user to embed the MicroID in the page returned by the resource. In this way, the user proves that they have the ability to control the content of the resource identified by the URI.

4.2.2 Reputation Computation Engine

We support each relying party having the ability to calculate the reputation of users in a personalized manner. That means that reputation calculations are based on information (claims, transactions, and opinions) specific to the user, using an computation that is customizable by the relying party.

These computations are specified in a rule language designed to facilitate reputation calculations in *Pythia*.

Reputation computations are performed using rule-sets, groupings of rules that are executed to carry out the calculation. Relying parties can store as many rule sets in the system as they desire. Each rule in the rule-set comprises an optional filter, a condition, and an action.

Filters operate on transactions and opinions. Some example filters would be *transactions where the customer satisfaction score is greater than* 5 or *transactions where a blog comment was rejected*.

Conditions operate on information in the system as well as other reputation information from the Internet. For example, we have added an interface to the Akismet API,[1] a system for tracking URLs associated with Spam and a content classification engine.

Conditions are simple boolean expressions on information in the system. Permitted boolean expressions are <, >, ==, <=, >=. *Pythia* includes a set of aggregate functions that can be applied to the filtered data. Implemented aggregate functions are `count`, `max`, `min`, `sum`, `average`, and `sd` (standard deviation). An example condition would be, *if the number (count) of transactions is less than* 10... or *if the maximum customer satisfaction value is less than* 3...

Actions modify the reputation score when their associated conditions are satisfied on the filtered data. An action may add, subtract or multiply any fixed amount or an aggre-

[1] `http://akismet.com/development/api/`

gated amount (per above) to the current reputation score. If multiple rule conditions in a rule-set are satisfied, all of the corresponding rule actions will be executed.

Examples of complete rules (combining filters, conditions, and actions) that could be expressed using the rule language include the following:

- *if the average customer satisfaction value* > 2.2 *in transactions where the customer satisfaction value is not null, then increase the reputation score by* 10%*.*
- *if the user has more than* 2 *verified email addresses in the system, increase the reputation score by* 1.2*.*
- *if any of the user's claimed URLs can be categorized as "commercial" then decrease the reputation score by* 5*.*
- *if the number of interactions where this users comment was rejected is* > 3 *then decrease the reputation score by* 2*.*

One of the dangers of our system is that users might concoct rule sets that simply don't work well or that developing "good" rule-sets might be more difficult than can be managed by casual users of the system. This can be mitigated in two ways.

First, *Pythia* provides a set of system-defined rule sets that can be used as-is or customized by relying parties for their particular purpose.

Second, future systems will include methods for conducting A/B testing of rule sets so that the effect of a rule change can be clearly determined. The use of A/B testing presupposes that the number of reputation queries by the relying party is sufficient to provide statistically meaningful results in the testing.

Users create rule sets using the relying party Web interface, which is available to any user. Since we anticipate that relying parties will not necessarily be programmers, we chose to develop a Web front end for managing rule sets. Building rules using a Web form limits the expressive power of the rule language, but we deem this an acceptable trade-off for ease of use.

4.2.3 Plug-In Architecture

Our goal was to build a general-purpose reputation framework, but the base system necessarily needs to be specialized to a specific domain for most uses. We determined to develop a plug-in architecture so that this specialization could be accomplished without modifying the code for the base system.

The base system supports user provisioning, the underlying identity system, identifier claims and verifications, building rule-sets for that information, and reputation queries.

To specialize the framework for a particular use, a plug-in specifies what communication and transaction messages are necessary for each integrating system. For example, an online forum might want to categorize transactions reported for posted comments using an enumeration of the status of members (e.g. "veteran") and use this information in calculating reputation. This same transaction attribute would not necessarily be useful in a blog system.

A plug-in is defined in an XML-formatted file that specifies a transaction format. Since plug-ins are stored as files that are loaded by the system at startup, installing, managing and transferring plug-ins to other installations is easy.

4.2.4 Using Pythia

Relying parties interact with *Pythia* in three distinct ways: they define reputation algorithms by creating rulesets, they make reputation queries about users, and they submit transactions to record their interactions with users.

Reputation queries can be made about unauthenticated identifiers. A reputation query includes the identifier of the relying party, an identifier for the subject, and a rule-set identifier. Any identifier stored in the system, email address or URLs, can be the subject of a reputation request. The system uses the rule-set identifier to select a rule set from those previously defined by the relying party and carries out the calculation, using the subject identifier to gather the right data for the rule set to operate on.

Transactions can only be reported about authenticated subjects to avoid users purposely contaminating the transaction store in order to bias future reputation calculations. Any transaction feedback is done in an authenticated manner that records identifiers for the relying party and the user involved in the transaction.

4.3 Implementation

The *Pythia* framework is implemented in Ruby on Rails. The system implements Web interfaces for users, relying parties, and administrators. The user interface allows users to claim and validate identifiers, view transactions to which they were a party, and view any reputation requests that were made about them.

Authentication for *Pythia* is provided using OpenID. The Ruby libraries for OpenID are from OpenID Enabled[2].

The underlying rules engine used by *Pythia* is Rools. [3] Rule sets can be defined by plug-ins or through the Web interface by relying parties.

5. AN EXAMPLE APPLICATION

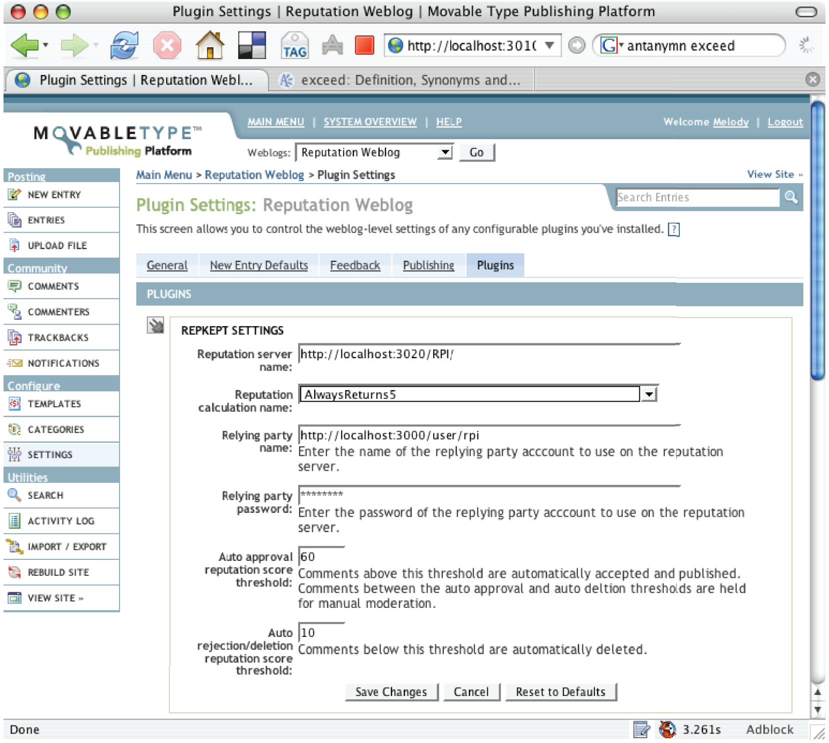
To demonstrate the validity of Pythia reputation calculations as a substitute for explicit authorization, we created a plug-in that allows *Pythia* to be used in blog comment moderation activities. Using this system, a relying party's blog system can query the reputation of people leaving comments. Users are not explicitly authorized to leave comments on the blog; instead, users with sufficiently high reputations have their comments automatically approved, while the comments of users with low reputations are automatically discarded. Users with intermediate reputations have their comments held for moderation. Comment approval is an identity transaction that raises user reputation, while comment deletion during moderation is a transaction that greatly decreases user reputation. New users are given a baseline reputation, to discourage users from abandoning their reputation data via pseudonym use. For purposes of the demonstration, we chose the MovableType blogging platform, but any blogging system could be used.

Customizing the general purpose framework for comment authorization required that we create a plug-in for the reputation framework. In addition, we built a module for MovableType that automated the *Pythia* transaction submission and reputation queries.

The plug-in defines its own schema for legal transaction types and transaction values specific to blog comments. The

[2] `http://openidenabled.com`

[3] `http://rools.rubyforge.com`



Plugin Settings: Reputation Weblog

This screen allows you to control the weblog-level settings of any configurable plugins you've installed.

REPKEPT SETTINGS

Reputation server name: http://localhost:3020/RPI/

Reputation calculation name: AlwaysReturns5

Relying party name: http://localhost:3000/user/rpi
Enter the name of the replying party acccount to use on the reputation server.

Relying party password: ********
Enter the password of the replying party acccount to use on the reputation server.

Auto approval reputation score threshold: 60
Comments above this threshold are automatically accepted and published. Comments between the auto approval and auto deltion thresholds are held for manual moderation.

Auto rejection/deletion reputation score threshold: 10
Comments below this threshold are automatically deleted.

Figure 2: RepKept Configuration Screen

framework uses the plug-in's schema definition to receive and store blog comment transactions.

Figure 2 shows the configuration screen in MovableType for the reputation-based comment moderation system. The MovableType plug-in we built to integrate with *Pythia* is called RepKept.

The blog owner, acting as the relying party, specifies the URL of the reputation server and gives authentication information for the reputation server. The relying party also selects which rule set they want to use to calculate the reputation score and gives thresholds for various actions. The rule-sets in the drop-down box are automatically populated from *Pythia* over the relying party API and represent rule sets the relying party has created.

In our test, the blogging software asks users to authenticate before they comment. Once the commenter has authenticated, the blog uses that identifier for requesting a reputation score and later submitting feedback. In our test, blogs use OpenID as the authentication mechanism, but theoretically, any identifier that was associated with the user in *Pythia* could be used.

Once a user submits a comment, a reputation request is made and a reputation value is computed according to the rule set the relying party has selected in the blog configuration. Depending on the value returned and the threshold values set by the relying party, the blog either automatically publishes the comment, holds it for moderation, or deletes it. Each of these generates feedback events. Each of these feedback events may alter the commenter's reputation score.

Feedback events contain the digital identifier of the relying party sending the feedback event and the authenticated digital identifier of the subject of the feedback event. The payload of a feedback event is simply a list of transaction records each represented by a key-value pair. The key is the transaction type and the value is a valid transaction value as defined by the plug-in schema.

In the blog comment case a transaction type could be the `Commenter's IP Address` and the corresponding transaction value could be `192.168.0.1`. A transaction can also be represented as an enumerated value. In this case the transaction type contains the enumerated value and the transaction value is empty.

The transactions in the blog comment example are predominately enumerated values. These values represent actions taken by the relying party given the reputation of the subject. The blog comment example has four types of feedback. In this particular example the feedback transaction types are mutually exclusive. However this need not be the case in general. The transaction types are generated according to the following scenarios:

- When the reputation for a subject exceeds the auto-publish threshold of the relying party the comment immediately published and a `AutoPublished` feedback event is sent to the reputation server.
- In the case where a subject's reputation falls below the relying party's auto-rejection threshold the comment is immediately deleted and a `AutoDeleted` feedback event is sent.
- If a subject's reputation lands between the auto-reject and auto-publish thresholds the comment is placed in a queue for manual moderation. When a human takes action on a comment in the manual moderation queue either a `ManualPublish` or a `ManualDelete` feedback event is sent to the reputation server.

6. FUTURE WORK

The framework we have described is still under development. While some key ideas are represented in the system, there is much to be done to complete our vision for them.

One of the most important additions to *Pythia* would be to expand the set of actions and conditional operators available in the rule language. Having more expressive operations would allow various computational models to be more easily implemented. Many of the extant models use sophisticated statistical functions well beyond the `average` and `sd` (standard deviation) operators now available.

The addition of built-in support for A/B testing would allow rules to be more easily evaluated for efficacy. Such A/B testing should be fine-grained so that effects of individual rules can be analyzed.

We're confident that the current framework can be used in any single domain with the addition of the appropriate plug-in to customize the type of transactions stored in the system. However, the use of information from multiple contexts (e.g. blog comments and online purchases) or of different types (e.g. transaction data vs. opinions) in a single system has proven too complex. In particular, the current 'hard-wired' semantics of the reputation data reported by plug-ins makes building cross-context reputation difficult; future versions of *Pythia* will allow more flexibility in defining and classifying transactions.

Currently, *Pythia* makes use of online data sources such as the `whois` database or Akismet by hard-coding support for the respective API into the framework. It should be possible to create plug-in style architecture for adding these online resources so that many more of then can be made available to *Pythia* applications.

The system could calculate audit data about the transactions and other information in the system and make that available to the rule language. In so doing, relying parties would be able to create rule sets that better take into account the quality of the data before proceeding with the calculation.

Dellarocas discusses the kind of gaming and unfair bias that can be present in online reputation systems. [7] While the issues surrounding security and data contamination were something we discussed consistently throughout the design of *Pythia*, we have not conducted a formal security audit of the system in order to elicit as many of these scenarios as possible and respond to them.

One part of our design that has yet to be implemented is allowing users to express opinions, ratings, and endorsements about other users. For this to be useful some way of recording interuser trust is necessary to measure the confidence that should be placed in a particular opinion. Otherwise, there's no way for the system to know whose ratings to use and whose to ignore.

Interuser trust can be defined explicitly by users, but might also be calculated or otherwise inferred. One way to infer interuser trust is by tapping into social cues that are already on the Web. For example, allowing users to claim OPML files from a site like Bloglines would give an indication of the URLs of other users whose blogs they read. FOAF (Friend of a Friend) data or other data from social networking sites could also be used.

Another avenue for future research is using the system as an identifier exchange service. The system already validates claims to identifiers and associates those identifiers together. *Pythia* could authoritatively state that a particular URL is controlled by the same person who controls a particular email address, for example. Identifier exchange services are useful in aggregating Web services where a client supplies one kind of identifier, but the server requires a different kind.

7. CONCLUSIONS

Reputation calculations can bridge the gap between the authentication-as-authorization approach used in open systems and explicit authorization, giving operators of large-scale web applications and social networks an easier way to leverage their communities without negatively impacting scalability. The emergence of generalized identity systems such as CardSpace and OpenID has provided the authentication foundation upon which scalable reputation systems may be constructed. This paper has described such a scalable and flexible reputation system framework, *Pythia*, which has been used to replace explicit authorization in web applications.

8. ACKNOWLEDGMENTS

This work grew out of a class project at Brigham Young University in the Winter of 2006 and 2007. The authors wish to acknowledge the significant contributions of Ryan Phelps, who was instrumental in *Pythia*'s implementation.

9. REFERENCES

[1] Amazon auctions. `http://auctions.amazon.com`, 2006.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[3] D. B. Bromley. *Reputation, Image and Impression Management*. John Wiley & Sons, 1993.

[4] J. Carter, E. Bitting, and A. A. Ghorbani. Reputation formalization within information sharing multiagent architectures. *Computational Intelligence*, 2(5):45–64, 2002.

[5] D. Chappel. Understanding Windows CardSpace. `http://msdn2.microsoft.com/en-gb/library/aa480189.aspx`, April 2006.

[6] P. Dasgupta. *Trust: Making and Breaking Cooperative Relations*, chapter Trust As a Commodity, pages 49–72. Department of Sociology, University of Oxford, 2000.

[7] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the* 2^{nd} *ACM Conference on Electronic Commerce*. ACM, 2000.

[8] C. N. Dellarocas. The digitization of word-of-mouth: Promise and challenges of online feedback mechanisms. *SSRN eLibrary*, 2003.

[9] eBay. `http://www.ebay.com`, 2006.

[10] B. Esfandiari and S. Chandrasekharan. On how agents make friends: mechanisms for trust acquisition. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies 2001*, pages 27–34, 2001.

[11] E. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[12] D. Hume. *A Treatise on Human Nature*. Penguine Classics (1975), 1739-1740.

[13] F. Labalme and K. Burton. OpenPrivacy.org. `http://www.openprivacy.org/`.

[14] F. Labalme and K. Burton. Enhancing the internet with reputations. Technical Report 0.7, OpenPrivacy.org, March 2001.

[15] K.-J. Lin, H. Lu, T. Yu, and C.-e. Tai. A reputation and trust management broker framework for web applications. In *International Conference on e-Technology, e-Commerce, and e-Services*, pages 262–269. IEEE, April 2005.

[16] P. R. Milgrom and J. Roberts. Predation, reputation, and entry deterrence. *Journal of Economic Theory*, 27:280–312, 1982.

[17] J. Miller. MicroID - small, decentralized, and verifiable identity. `http://microid.org/`.

[18] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th Hawaii International Conference on System Sciences*. IEEE, IEEE, 2002.

[19] L. Mui, M. Mohtashemi, and A. Halberstadt. Notions of reputation in multi-agent systems: A review. In *Proceedings of the First International Conference on Autonomous Agents and MAS*, pages 280–287, Bologna, Italy, July 2002. ACM.

[20] OpenID specification. `http://openid.net/specs.bml`, 2006.

[21] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, December 2000.

[22] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The value of reputation on eBay: A controlled experiment. *Experimental Economics*, 9(2):79–101, June 2006.

[23] J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, September 2005.

[24] M. Schillo, P. Funk, and M. Rovatsos. Using trust for detecting deceitful agents in artificial societies. *In Applied Artificial Intelligence, Special Issue on Trust, Deception and Fraud in Agent Societies*, 14(8):825–848, September 2000.

[25] S. Sen and N. Sajja. Robustness of reputation-based trust: boolean case. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 288–293, New York, NY, USA, 2002. ACM Press.

[26] R. L. Trivers. The evolution of reciprocal altruism. *Quarterly Review of Biology*, 46:35, 1971.

[27] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[28] P. J. Windley. *Digital Identity*. O'Reilly Media, 2004.

[29] P. J. Windley. Principles of reputation. `http://www.windley.com/archives/2006/06/principles_of_r`, June 2006.