Resilient Trust Management for Web Service Integration

Sungkeun Park, Ling Liu, Calton Pu, Mudhakar Srivatsa, Jianjun Zhang
College of Computing, Georgia Institute of Technology
{mungooni, lingliu, calton, mudhakar, zhangjj}@cc.gatech.edu

Abstract

In a distributed web service integration environment, the selection of web services should be based on their reputation and quality-of-service (QoS). Various trust models for web services have been proposed to evaluate the reputation of web services/service providers. Current mechanisms are based on tracing the feedbacks to the past behaviors of web services. However, very few of them consider the robustness and attack-resiliency of the trust models. In this paper, we present an attack resilient distributed trust management system in a web service management environment. The proposed attack resilient trust model uses two vectors to capture the behavior and the trustworthiness of a web service/service provider based on our analysis on the possible attacks against the trust models. We also present a set of experiments that show the effectiveness of our trust model in detecting malicious behavior of service providers.

1 Introduction

Web services enable cross-platform and language-neutral accesses to data and applications over the Internet. By following standards such as SOAP [4], UDDI [19], and WSDL [20], web services can dynamically interact with one another and thus can execute complicated business processes. Such interoperability drives an increasing trend to *compose* complex and value-added web services by integrating existing ones.

Two issues have to be addressed to enable web service integration amongst several autonomous web service applications and service providers. First, a scalable *web service discovery mechanism* is essential for customers to select the web services that can satisfy their functionality and quality-of-service (QoS) requirements. Secondly, a *reputation management mechanism* is necessary to differentiate service providers based on their reputation in providing dependable web services.

To answer the first challenge, a number of research works [7, 18, 15, 14] use peer-to-peer techniques (as opposed to centralized techniques) for web service discovery and process management. The service definition of a web service usually includes two parts. The syntactic properties of the web services are defined using an interface definition language (WSDL). Service level agreement (SLA) specifies the QoS attributes like service availability, maximum response time and minimum throughput. The search mechanisms of peer-to-peer networks are employed for searching web services that match the functionality and QoS requirement of customers [16, 22]. A third party is usually involved in facilitating the enactment of the SLA agreements [13, 10].

Reputation management systems [17] have been widely used in business and electronic commerce systems as an effective mechanism to manage risks involved when two *unknown* peers transact with one another. Complementary to the service process management and discovery mechanisms, various reputation models have been proposed to address the second issue in web service integration [22, 8, 7]. Those reputation models evaluate the reputation of a web service by tracing the feedbacks on its past behavior. Nevertheless, they are not designed to handle malicious and deliberate attack on the trust models. In order to make accountable recommendations to the users, reputation management systems need to be able to detect possible attacks or threats that malicious users pose to the system. To reduce the amount of negative experience, it is necessary for the reputation management system to warn users of possible bad integration attempts and discourage them from performing such integration.

Bearing these issues in mind, we present a trust management framework for decentralized peer-to-peer web service discovery and process management systems, with emphasis on controlling possible attacks and threats to the systems. Our contribution can be summarized in the following three aspects:

- First, we describe a detailed threat model focusing on possible attacks to the reputation management system.
- Second, we propose a resilient trust model that defines the trust of a web service using trust vector and penalty vector. The trust vector captures direct trust, authenticity, and reputation trust from other users. The penalty vector incorporates the penalty measure for various malicious behaviors such inconsistency, and misusage of trust.
- Third, we present a set of experiments that shows the effectiveness of our trust model in detecting malicious behavior of service providers.

2 Design Considerations

The service model we use in this paper is a web service integration environment with an unstructured overlay network topology. Each service provider offers certain web services. The users are interested in dynamically composing and integrating web services to form more complex and value added services. A query feature, similar to that deployed in Gnutella-like peer-to-peer file sharing systems is provided. A user query locates services of interest to the user. This is usually accomplished by flooding the query on the overlay network. Whenever a user obtains a list of service providers that claim to meet its requirement, the user may inquire others to make suggestions on which providers are more trustworthy compared to others on the candidate list.

Given this service model, we focus on handling the following threats that can be imposed on the system by malicious service providers. Malicious service providers may offer malicious or inauthentic services that do not match what has been advertised in the WSDL document. Further, malicious service providers may distribute dishonest opinions on other service providers (inflating or downplaying other providers' reputation). We will discuss the details of the threat model in the next section.

We add two characteristics to our trust model to make it more reliable. First, it may take a long time for participants to establish their reputation, but a relatively a short time to loose their reputation. We consider this to be important, since malicious participants may take advantage of any trust model that lacks this characteristic and perform malicious actions frequently while retaining their trust level. We assume unintentional bad performances of non-malicious participants do not happen frequently and they will not be affected by this characteristic. Another characteristic of our trust model is maintaining a personal storage of trust information rather than using a system-level trust information repository. An advantage of this approach is that it enables participants to retrieve trust information selectively from other participants chosen based on their credibility. On the other hand, a central repository based approach suffers from being a single point of failure and provide less meaningful information resulting from summarizing the reports of all the participants, including the ones that are not credible, in the system.

For the remainder of this paper, we shall use the terms *provider*, *consumer* and *peer* in the following context: Any participant, regardless of the role it plays, will be referred to as *peer*. In a web service integration environment, any participant may play the role of requesting a web service or providing it at any given point in time. A participant playing the former role will be referred to as *consumer* and the latter as *provider*. We assume that this definition also applies when it comes to requesting/providing opinions about *providers*.

3 Threat Model

3.1 Malicious Behavior

Providing inauthentic web services. Among the threats described in the previous section, the first type of threat is the simplest form of threat that can be frequently observed within web service integration environments. Malicious service providers may advertise that they would provide services that match consumers' interest, and provide different services when actual requests are received. The disguised services may not be harmful, but in the worst case, they could inject viruses or Trojan horses capable of imposing critical damage to the consumer.

Distributing dishonest opinions. Since the service model allows customers to share their transaction experiences with a service provider and make suggestions, some malicious customers may take advantage of this feature to mislead less informed peers into believing false information about other providers. One of the two variations of this threat is downplaying trustworthy providers' reputation resulting in preventing consumers from paying attention to and/or issuing requests to trustworthy providers. This doesn't immediately seem to do any harm to non-malicious consumers. However, it may lead consumers to choose less trustworthy providers, possibly malicious ones, and receive undesirable services which may be harmful. The other variation, which can do more harm, is some peers boosting untrustworthy providers' reputation unreasonably, possibly as a part of collusion, to trick consumers into trusting untrustworthy providers. In this case, it is likely that consumers face the first type of threat described in this section, possibly receiving harmful services.

3.2 Collusion

Individual malicious service providers. This type of providers has intention of offering malicious services to consumers. They are not aware of other malicious providers, so their attacks are not specifically targeted to good consumers. Their behavior will be captured by consumers' feedbacks that indicate that they are providing bad services to consumers or dishonest opinions about other providers. However, it is necessary to make distinction between newly introduced providers and malicious providers because providers in both situations are subject to having low trust values. As for newly introduced providers, their number of transactions will be equal to or not much bigger than zero. On the other hand, providers consistently behaving in a malicious manner will have low trust value, but with a longer history of transactions in comparison with newly introduced providers.

Collusive malicious service providers. Unlike individual malicious service providers, collusive malicious providers are able to cooperate with each other. They strategically boost each other's trust value. For example, a set of colluding providers are split into two groups. Providers in group A will always give high ratings to providers in group B. As a consequence group B providers may behave maliciously with other peers in the system and yet not loose their reputation entirely. We believe, however, this strategy will not be quite effective when they are communicating with friendly peers for the following reason. Unless reports are gathered restrictedly from the members of the colluding clique, other reports from honest peers will be collected as well. As long as trustworthy reports can be secured from malicious tampering, it will not be difficult to filter out dishonest reports from colluding peers and find out whether or not the provider is to be trusted.

3.3 Complex strategy

Malicious service providers that are knowledgeable of the rules of the game will use a complex strategy of pretending to be good for a certain amount of time and garner reputation from consumers. Then the malicious service providers exploit this reputation to deceive consumers that request its services. The customers will be duped into thinking that they are being properly served when they are actually receiving inauthentic or even harmful services. Referred to as *contour discovery attack* [11], given that a system uses a single threshold value, artful malicious providers that have found out the threshold value of the system may maintain its trust value at a certain level in order to perform malicious actions whenever they intend to. If all the peers use different threshold values, malicious providers will have a hard time figuring out all the threshold values. For malicious providers to be recognized as friendly at all time, higher reputation needs to be achieved, requiring even more work. We believe this policy may discourage many malicious providers.

4 Trust Model

Most trust metrics [3, 9, 21] generally consider a small number of quantified values for evaluating the trustworthiness of other peers. We take more issues into consideration so as to obtain more accurate evaluation of the providers' reputation. More so, to protect consumers from receiving services from malicious providers, we need safeguard mechanisms that can effectively minimize the extent to which a malicious provider may take advantage of the trust model by behaving maliciously while retaining high trust value. Further, these mechanisms should be able to discourage any provider behaving non-maliciously over an extended period of time to attain good reputation. However, we should guarantee that the cost of increasing a provider's reputation depends on the extent to which it misbehaved in the past. For example, a provider that has a long history of malicious behavior in the past should find it very hard, if not impossible, to build reputation in a short span of time.

Our trust metrics is composed of two parts – *trust vector* and *penalty vector*. The value of each parameter in the vectors is measured based on its relevance to the description of the parameter on a continuous scale between 0 and 1. The overall trust value of a peer is evaluated by subtracting the penalty vector from the trust vector. Let TV_{ik} denote the overall trust value of provider k in the viewpoint of peer i, T_{ik} be the trust value computed using trust-related information about provider k available to peer i, and P_{ik} be the penalty value of the provider k computed in viewpoint of peer i. Peer i can evaluate the trust value of provider k using the following equation.

$$TV_{ik} = \alpha * T_{ik} - \beta * P_{ik} \;\; (0 \leq \alpha, \beta \leq 1)$$

The values for α, β should be chosen based on how optimistic a peer is. The more optimistic a peer becomes towards providers' behavior, it will choose the values for α, β such that $\frac{\alpha}{\beta}$ is large so that the overall trust value is less affected by *penalty vector* value. On the other hand, the more pessimistic a peer becomes towards other providers' behavior, it will choose the values for α, β such that $\frac{\alpha}{\beta}$ is small so that the overall trust value is sensitive to the value of the *penalty vector*.

4.1 Trust Vector

The trust vector of peer i for provider k is composed of three parameters that represent the trustworthiness of provider k as viewed by peer i. In a distributed, decentralized computing environment, peers experience sparsity of information which is used to evaluate others. Some of the peers may have *a priori* knowledge of some providers which will help them estimate their behavior. However, in many cases peers are not familiar with the providers, and a number of transactions need to be performed before a peer can make evaluations about providers. Parameters in the trust vector represent the trust value of a specific provider that has been computed by the holder of this vector based on the type of knowledge that was used for the computation. Each parameter is related to one another, but not dependent upon one another in its entirety. For each parameter in the vector, we will present its concept, its usage scenario, possible threats it can deal with, and how it is computed.

$$\vec{T_{ik}} = < dt_{ik}, au_{ik}, rt_{ik} >$$

Direct Trust (dt_{ik}) dt_{ik} denotes peer i's trust in provider k based on previous experience or direct acquaintance. This type of trust was built in different context which doesn't have direct relation with the system. However, we believe it still can provide peers with information about other peers where availability of such information is not high.

Suppose Alice newly joined the system to download files of her interest. It will not be so difficult for her to locate the services using the search feature the system provides. However, amongst a long list of possible providers of the service, it will not be easy for her to choose to which provider she should make the request if she doesn't have much information about which providers are more reliable compared to the others. Being a newcomer to the system, she lacks transaction history with providers that she can use to make a prediction based on providers' past behavior. In this situation, assume that she found Bob whom she has known and built trust with for a long time among the list of provider candidates. Even though she didn't have any direct experience with Bob within the system, she can expect a reliable transaction with Bob by taking a long standing acquaintance with him in other context into consideration. Thus, Bob will be her choice over other possible providers.

Hence, in direct trust, without requiring certain number of transactions to prove trustworthiness, provider k is trusted by peer i. This relationship can be represented by personal certificates. Each certificate should be renewed after a designated time interval. Trust value also should be reevaluated based on provider k during that time interval.

Authenticity (au_{ik}) au_{ik} denotes peer i's belief that the identity of provider k is authentic. While direct trust can be used as a valuable piece of information for decision making, the identity of a peer might itself be questionable. Therefore, it is imperative to verify the identity of the peer before performing a transaction with that peer based on direct trust value for that peer. It is possible that some malicious peers impersonate some reputable peers to trick other peers with less information into believing the impostors with fake identities. While direct trust value can be defined as the extent to which a peer can place trust on another peer based on previous acquaintance assuming that the identity is authentic, *authenticity* can be defined as the extent to which a peer believes that another peer's identity is authentic.

Extending the previous example, Alice's trust in Bob needs to be supported with the authenticity of Bob's identity in the system. If Alice cannot be sure of the identity of the peer that claims to be Bob, she cannot assign a high value to $authenticity$, which results in low overall trust value. As it becomes clear that the peer that claims to be Bob is in fact whom it claims to be, Alice can place a high value on $authenticity$, which leads to a high overall trust value.

Reputation Trust (rt_{ik}) rt_{ik} refers to the quantified reputation value based on peer i's feedback after its transaction where provider k was involved. Using direct trust and authenticity

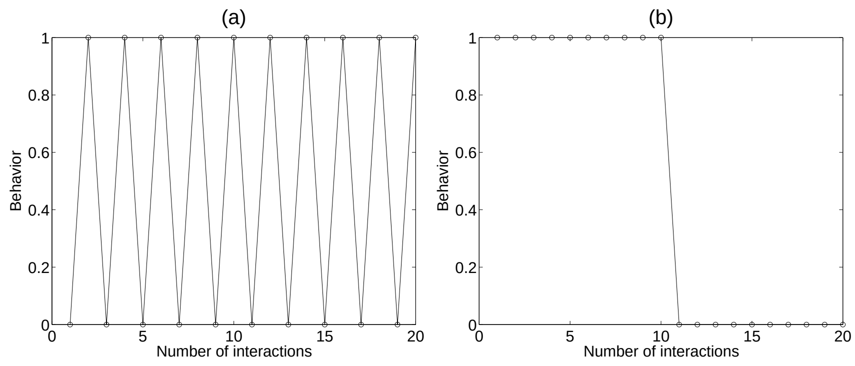

Figure 1: Malicious providers with different behavioral patterns

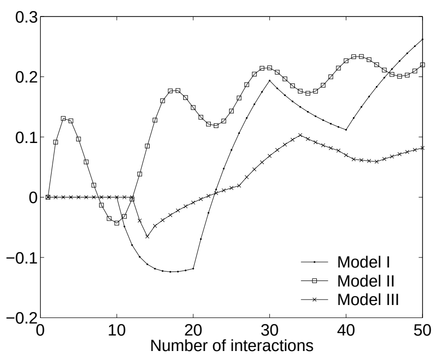

Figure 2: Cost for performing malicious actions

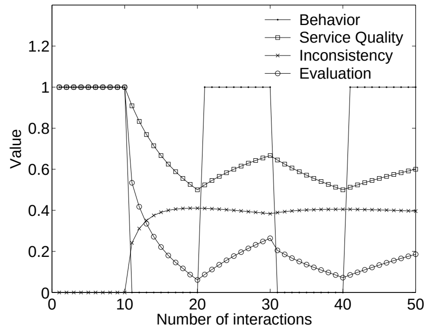

Figure 3: Model I

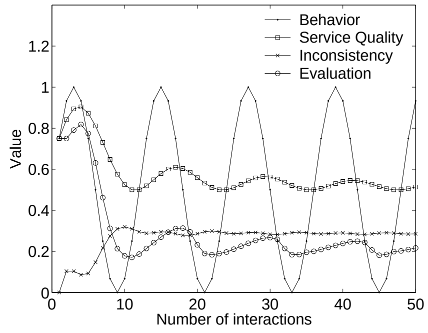

Figure 4: Model II

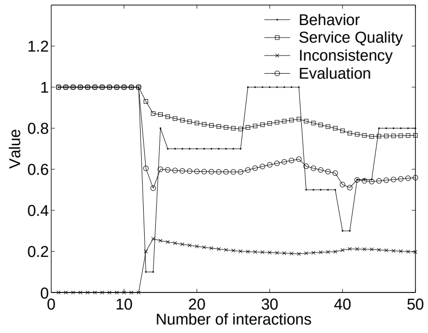

Figure 5: Model III

value is helpful in choosing which providers to transact with for consumers with short transaction history, but the context in which the trust was built may be different from the one that the system conveys. As a consumer gathers more experience with providers, it may quantify its experience by assigning a feedback value for each transaction it performed. *Reputation Trust* value is a summary of all the feedback values a consumer assigned for transactions with different providers.

Suppose Alice now has accumulated sufficient transaction history to account for behavior of other peers and compare their reliability with Bob whom she has known for a long time. If there are providers with high *reputation trust* value compared to that or direct trust value of Bob among the list of provider candidates, given Alice issued a request, Alice will prefer providers with higher *reputation trust* value to Bob. Alice wasn't familiar with most of them when first joining the system, but she now has found out that in this system, which offers different context in which to build trust compared to that in which she build trust with Bob, some providers offer more reliable service than Bob whom she has known and trusted for a long time.

If provider k responded to the request that had been issued by peer i, peer i evaluates how well provider k performed in the transaction after it has been executed. For now, we assume that all the feedbacks in the system are honest. In such a scenario, peer i keeps record of evaluations on transactions with provider k and computes the average value which is maintained as rt_{ik}.

4.2 Penalty Vector

While trust vector represents general trustworthiness of providers, it is likely to fail in capturing the behavior of malicious providers especially when they know the rule of the game and try to maintain certain amount of trust value to perform contour discovery attacks. In order to portray the unpredictable and fluctuating behavior of malicious providers, we introduce *penalty vector*, which is composed of two parameters that represent the maliciousness of provider k in previous transactions in the viewpoint of peer i. The description of each parameter is as follows.

$$\vec{P}_{ik} = < ic_{ik}, mi_{ik} >$$

Inconsistency (ic_{ik}) ic_{ik} denotes the spread of peer i's entire feedback values on transactions with provider k. Intuitively, low ic_{ik} value may enable peer i to place high confidence on the trust vector $\vec{T}_{ik}$. $Inconsistency$ value is computed by calculating the standard deviation of all the feedbacks of peer i based on its transactions with provider k where each transaction may have different weight in the computation. The following equation is used for computing $inconsistency$ value, where $x_{ik,j}$ is peer i's evaluation of j^{th} transaction with provider k and w_j is the weight of the j^{th} transaction:

$$ic_{ik} = \sqrt{\frac{\sum_{n=1}^{maxH}\{w_n \times (x_{ik,n} - rt_{ik})^2\}}{\sum_{n=1}^{maxH} w_n}}$$

Misusage (mi_{ik}) mi_{ik} denotes the amount of trust value that provider k is currently taking advantage of in performing transactions with peer i. The goal of incorporating this factor into *penalty vector* is to reflect the cost of behavioral fluctuation to be paid by provider k. Its general formula is as follows:

$$mi_{ik} = \frac{\sum_{n=1}^{maxH}\{w_n \times max(0, rt_{ik} - ic_{ik} - x_{ik,n})\}}{\sum_{n=1}^{maxH} w_n}$$

We can apply appropriate values to w_n and $maxH$ so as to serve peer that place different weight on feedbacks based on their recentness and/or have different storage policies. Peers that treat all feedbacks equally, regardless of the point of time at which they happened, may apply 1 to w_n for all n. On the other hand, peers that consider recent transactions more important may place more weight on recent transactions, and exponentially decrease the weight of transactions according to the length of time elapsed. For example, if the length of time window is chosen as t, $w_n = \frac{1}{2^{j-1}}$ where $(j-1)\times t+1 \leq n \leq j\times t$. Entities can also choose $maxH$ value of their preference depending on their storage policies.

Overall Trust Value. When peer i evaluates provider k, the overall trust value can be computed by using direct trust value alone. The decision can be made more easily if a personal certificate is present. However, if peer i needs to compute overall trust value of provider k, using reputation trust value, penalty vector should be applied to the computation. Weight of a, b is placed on parameters in penalty vector, *inconsistency* and *misusage* respectively. Amount of trust value being amortized will be $a \times ic_{ik} + b \times mi_{ik}$. Thus, the *overall trust value* of provider k in the viewpoint of peer i will be $\alpha \times rt_{ik} - \beta \times (a \times ic_{ik} + b \times mi_{ik})$.

At first glance, $inconsistency$ and $misusage$ value seem to capture similar characteristics of malicious providers' behaviors. However, the reason we differentiate the two factors is to observe malicious providers' behaviors over time frames of different length. Let us take a look at Figures 1(a) and 1(b). The behavior of a provider depicted in Figure 1(a) is constantly oscillating between two extremes, while a malicious provider whose behavior is illustrated in Figure 1(b) remains completely honest for 10 transactions, then suddenly changes its behavior to the opposite extreme, where it stays for an equal number of transactions. It is difficult to tell which provider is worse than the other. However, our penalty vector is capable of capturing the behavioral patterns of both the providers. The computation of $inconsistency$ value makes it easy to find out that both providers had the same number of transactions at either extremities. (Note that, computed values may not be exactly the same, if different weights are applied to each entry in the transaction history.) In addition, by computing $misusage$ value of both providers, one can see that the behavior of the provider depicted in Figure 1(a) is more fluctuating in a shorter period of time, while the provider's behavior in Figure 1(b) doesn't change frequently. Figures 2-5 show that malicious peers will be at disadvantage for displaying malicious or inconsistent behavior. *Service Quality* shows the simple average value of *Behavior* and *Evaluation* corresponds to the computed trust value.

5 Experiments

In this section, we will present results of our experiments that will show the effectiveness of our trust model. For our evaluation metrics, we will measure the *integration success rate* which represents the rate of consumers successfully locate reliable services and perform service integration with them.

Experiment Setup. We are considering a web service integration environment with unstructured topology. Peers are connected to a number of neighbors. The requests are forwarded to their neighbors excluding the originators of the requests. Providers that are able to offer the service specified in the request will send a response to the consumer. Upon collecting all the responses, the consumer will choose a provider that is most reliable in providing the requested service, based on the computed trust value. Other than requests, peers may also ask for reports of peers' experience with certain service providers in the same way that requests are processed.

There are 4000 peers in the system. We simulate two types of providers in our experiments, namely, good providers and malicious providers. We vary the population of malicious providers in the system and the frequency of malicious providers offering inauthentic services to the consumers. Details will be discussed in the next subsection. (For trust computation, the values for trust metric parameters in section 4.2 are $\alpha = 0.9, \beta = 0.1$.)

# of service provider	4000
% of providers that provide requested service	5%
% of services good peers provide bad service	5%
% of services malicious peers behave maliciously	$mrate$
% of malicious peers in the system	MP
% of peers utilizing reputation-based trust	R
% of peers utilizing direct trust	D

Table 1: Experiment Settings

5.1 Effectiveness of proposed trust framework

Individual malicious providers. In this experiment, we tested the performance of our trust model by performing 600 transactions per peer and then using feedbacks for those transactions to build trust. In addition, we assign some providers to utilize direct trust. Using direct trust does not require warm-up phase that was needed to build trust based on reputation. Figures 6 and 7 show comparison between systems, one of which used reputation to build trust and the other is not a trust based system.

As expected, Figure 6 shows the system utilizing reputation to build trust allows good peers to achieve higher integration success when compared to the system that doesn't utilize trust of any sort. We varied the number of malicious providers in the system, and assumed that the malicious providers always provide inauthentic services. The setup for experiment that yielded result shown in Figure 7 fixed the number of malicious providers in the system. We increased the fraction of transactions when malicious providers offer inauthentic services in steps of 10%. Figure 7 shows a result similar to that of Figure 6. As the number of providers that utilize reputation trust is high, the transaction success rate increases. As malicious providers offer bad services more frequently, their integration success rate decreases primarily because other peers choose not to utilize services from a low reputation service provider.

Next, we examine the effect of combining direct trust along

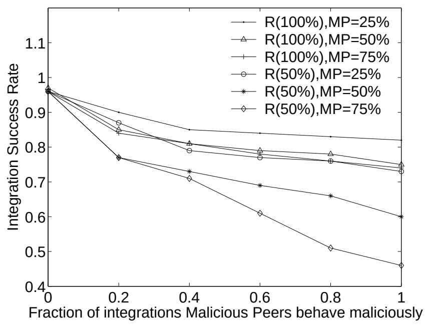

Figure 7: Comparison of experiment parameters' effect on reputation

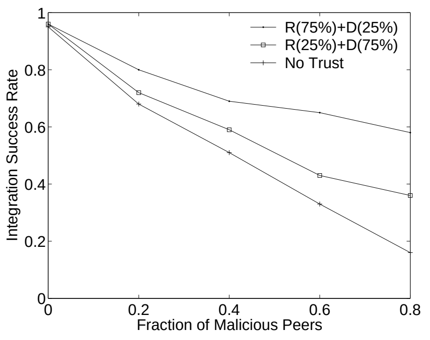

Figure 8: Performance comparison between reputation based trust and direct trust ($mrate$=1)

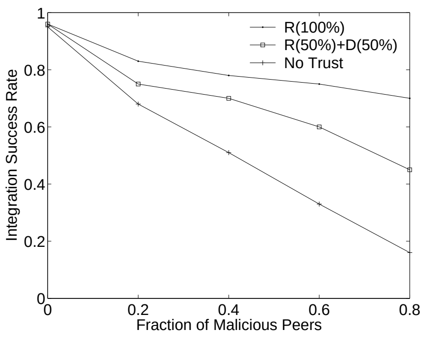

Figure 9: Collusion experiments I (Population of malicious providers varies, $mrate$=1)

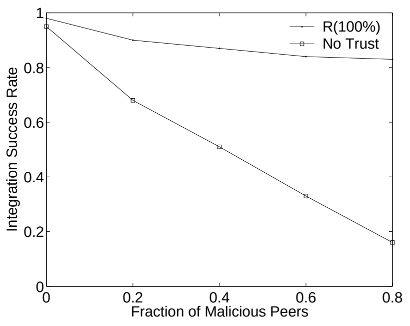

Figure 6: Performance comparison between reputation based system and no trust system ($mrate$=1)

with reputation as a means of building trust. Intuitively, performing transactions with providers that one has direct trust with is more reliable than with those that one has built trust based on reputation. However, in a distributed web service integration environment, it is difficult for a peer to have direct trust with a large number of providers. Due to this fact, while it is clearly a better choice to make transactions with providers that one has direct trust with, one's trust relationships with other providers in general should be complemented with trust built on reputation. In Figure 8, we assumed that peers have direct trust with 10% of providers in the system. In Figure 6, all the peers in the system build trust with others based on reputation or direct trust. In one of the experiments, 75% of the peers use only reputation to build trust while others use direct trust as a means of identifying trust relationships. The other experiment assumed 25% of the peers used reputation trust and the rest used direct trust. The two groups of peers, namely, the peers using reputation trust and those using direct trust, do not overlap with each other. Figure 6 shows that, under our model, peers using reputation trust are more likely to be successful in locating reliable services compared to peers using direct trust only. We can explain this observation with the previous argument. While peers using reputation trust can evaluate all the providers in the system, even though it may have errors, peers only using direct trust do not have any knowledge about providers that they don't have direct trust with. Using direct trust is most reliable in determining the provider with which one wish to have transactions, but it must be complemented with reputation trust.

We can conclude that reputation trust is the dominant factor in deciding the transaction success rate. However, there are differences in how one applies these two types of trust in evaluating providers. When reputation trust is utilized, a peer will honor information about providers that it stores the most. In case information about certain providers is missing, it can still ask its neighbors for reports on their experience with the providers of interest. Reports from reputable peers are highly valued, and they can be used to build some form of transitive trust when we utilize reputation trust. This is not the case with direct trust. We assumed that consumers only send requests to the providers that they have direct trust with. Properly deriving transitive trust is an important issue in a reputation management system, but we will leave this matter to be addressed in the future.

Colluding malicious peers. For the experiment which yielded the result shown in Figure 9, total number of malicious providers performing collusion was increased in steps of 10%. Each provider in the colluding clique always provided inauthentic services to the consumers. They also boosted the trust value of their accomplices regardless of their behavior, while downplaying the trust value of good providers. The performance of our trust model, compared to experiments for individual malicious providers, is visibly hit. We believe this to be a natural phenomenon, for the colluding clique's strategy creates pitfalls that trust systems can fall into from time to time. Still, with a large number of colluding malicious providers, we can say that integration success rate is fairly high. Figure 10 shows an interesting result. For this experiment, fraction of malicious providers in the system was fixed to 50%. The fraction of transactions in which malicious providers offer inauthentic services was varied, starting from 0%, increasing in steps of 10%. However, in a system where malicious providers collude, we believe their attack will be more effective when they provide inauthentic services intermittently. When the services are always inauthentic, reports on malicious providers' behavior between good providers and malicious providers will be entirely polarized. If malicious providers offer good services frequently, reports issued by good peers and malicious peers will have less difference compared to the previous case. We believe this may increase the effectiveness of their attack. The system that fully utilized reputation based trust suffered the most when malicious providers offered authentic services for half of the transactions, as we can see in Figure 10. As malicious providers offer inauthentic services more frequently, systems that utilized trust system tend to perform better. Another interesting to note is that, although good peers were able to achieve higher integration success rate when the fraction of authentic services being provided was higher, it may be the case that the effectiveness of malicious providers' attack was even higher. Since they provided authentic services more frequently, they could gather higher trust even from good peers. Thus for every attempt malicious providers make to provide an inauthentic service, good peers might have fallen for it. Malicious providers may take advantage of this scheme to launch attacks less frequently but make good peers sustain larger damage for each attack that they fall for. However, dealing with so-called one-time attacks requires designing metrics for measuring the magnitude of different attacks and is out of scope of this paper.

Malicious peers with complex strategy. For experiments to examine if our system can detect malicious providers that behave smarter, we simulated two types of strategic malicious service providers. First type of provider's behavior toggles between 1 and 0.4 (Type A). The behavior cycle of the other provider will be $\{1, 0.4, 0.4\}$ (Type B). Arithmetic mean of the behaviors of these two providers will be 0.7 and 0.6, respectively. The third type of malicious providers will be the one we have been using for the previous experiments, always displaying malicious behavior. We will call this type of malicious provider as type C. The ratio of three types of malicious service providers in the system is Type A:Type B:Type C=35%:35%:30%.

We tested three types trust system to see how they handle each type of malicious service providers. One type of trust system(system S_α) only utilized reputation based trust, and we expect it will not be able to cope with any kind of malicious service providers with complex strategy though it may still detect malicious service providers with naive scheme. Another type of trust system(system S_β) utilizes penalty vector, but with lower threshold(0.6). We expect it will be capable of detecting strategic malicious service providers. However, malicious service providers that are able to maintain their trust value at a higher level can still elude this trust system. Thus, we test a third trust system (system S_γ), similar to the second one but using a higher threshold value of 0.7. From Figures 11 and 12, we can see that system S_γ outperforms other two trust systems. Compared to system S_β, same trust measures were utilized. Using a higher threshold will allow the system to regard smarter malicious service providers as malicious, and this explains the difference of performance. System S_β was able to classify type B providers as malicious, but not type A providers for the most of the time. There is a trade-off to consider in deciding which threshold to use. If the threshold is set too high, it may rule out most of malicious providers from possibly becoming the provider, but it may also regard good providers that have made honest mistakes as malicious. On the other hand, using a threshold value that is too low will result in classifying many strategic malicious providers as good ones. For system S_α, it could only detect type C malicious providers. Both type A and type B providers were able to deceive system S_α and inject inauthentic files into the system while disguised as good providers.

6 Related Works

Abdul-Rahman et al. [1] proposed a model for supporting trust in virtual communities, based on direct experiences and reputation. They introduced the general definitions of trust and reputation, along with their characteristics. They used them to design their trust model which also had inference features. Beth et al. [3] employs a directed graph where nodes represent peers and edges represent trust relationships. The edges are associated with a value in the range [0, 1] which is quantified based on numbers of positive/negative experiences between peers that each edge connects. There are two types of edges based on the context of the trust. Direct trust is built on positive experiences a peer had with another peer. Recommendation trust shows the extent to which a peer is willing to accept reports from another peer about experiences with third parties. Their formula, with two types of trust values as parameters, evaluates the overall trustworthiness of a graph. Maurer's metric [12] takes similar approach to evaluating the trustworthiness of trust relationships. A probabilistic model is used to compute the trust value. PGP model [23] is a popular public-key management system. It also employs a graph, where nodes are public-keys and edges represent relationship between the nodes. Each edge is labeled with attributes and associated with certificates. The holder of a certificate can verify the attributes that are bound to the key on the other side of the edge. After each node is authenticated, each node is assigned a trust value with which the trust relationships among the nodes of the graph can be evaluated.

Several reputation/trust management systems specific to distributed P2P system context have been proposed, and Aberer et al. [2] was the seminal work. Their system was based on negative feedbacks, which works in very limited cases and is over-sensitive to a skewed distribution of the community and to several misbehaviors of the system. They believed there are incentives only for submitting feedbacks on unsatisfactory transactions. Cornelli et al. [5] proposed P2PRep which is a protocol, where all the peers in the network can keep track of good and bad experiences about other peers it has interacted with, identified by their pseudonyms, and share the information with other peers. One can poll about the reliability of prospective sources before downloading resources from the peers. Their protocol can be used to extend existing P2P protocols, as demonstrated on top of Gnutella. Damiani et al. [6] proposed a protocol that had peer selection scheme similar to that of P2PRep. EigenTrust [9] is another reputation management system for P2P networks. Their evaluation stems from the concept of transitive trust, while not discussing the different contexts of trust that may be applied to infer the transitiveness. They also assumed the existence of a small number of trustworthy peers whose opinions on other peers are known to be trusted. More recently, there were researches addressing the problem of utilizing trust and reputation systems in the context of web service environment [7, 8, 18].

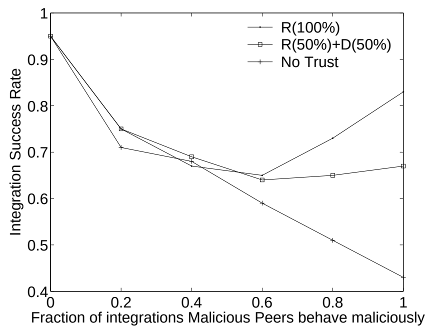

Figure 10: Collusion experiments II (Malicious peers strategically upload inauthentic files, MP=50%)

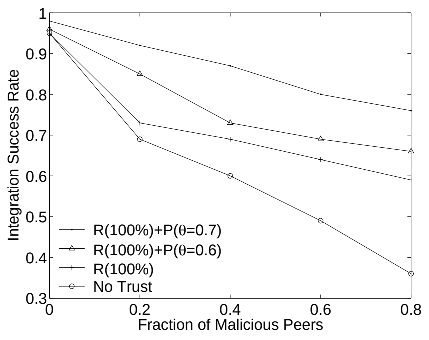

Figure 11: Malicious peers with complex strategy I (individual), MP=50%

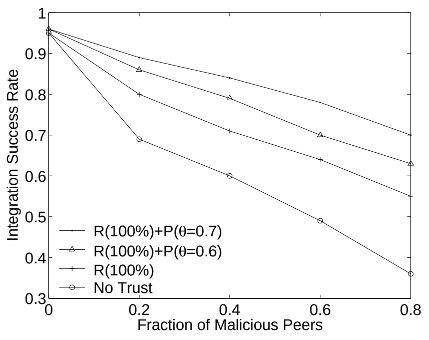

Figure 12: Malicious peers with complex strategy II (collusive), MP=50%

7 Conclusion

We have presented an overview of our effort to build a resilient trust management framework for a distributed web service integration environment. There are three main contributions in this paper. First we provided a detailed threat model focusing on representative attacks to a reputation management system. Second, we described a resilient trust model that promotes the use of trust vector and penalty vector to compute the trust of a service provider. The trust vector aggregates several trust measures such as authenticity, service quality, direct recommendation and indirect recommendation from other users. The penalty vector incorporates the penalty measures for various malicious behaviors, including inconsistency and misuse of trust. Our third contribution is presenting a set of experiment results that demonstrate the effectiveness of our trust model. In the future, we plan to put more effort on improving the trust usage model, especially developing trust inference schemes in complicated situations, and to deploy this model in a real web service integration environment to observe its practicality.

References

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of HICSS-33*, 2000.

[2] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of CIKM*, 2001.

[3] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of ESORICS'94*, 1994.

[4] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1. *Technical report, http://www.w3c.org/TR/SOAP*, 2000.

[5] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servents in a p2p network. In *Proceedings of WWW*, pages 376–386, 2002.

[6] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216, 2002.

[7] F. Emekci, O. D. Sahin, D. Agrawal, and A. E. Abbadi. A structured Peer-to-Peer framework for discovery and reputation model for web services. In *Proceedings of ICWS*, 2004.

[8] S. Kalepu, S. Krishnaswamy, and S. W. Loke. Reputation = f(user ranking, compliance, verity). In *Proceedings of ICWS*, 2004.

[9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of WWW*, 2003.

[10] A. Keller and H. Ludwid. Defining and monitoring service level agreements for dynamic e-Business. In *Proceedings of LISA'02*, 2002.

[11] D. W. Manchala. E-commerce trust metrics and models. *Internet Computing*, 4(2):36–44, 2000.

[12] U. Maurer. Modelling a public–key infrastructure. In *Proceedings of ESORICS'96*, 1996.

[13] J. M. Myerson. Guarantee your web service with an SLA: Introduction, architecture and testing mechanisms. http://www-106.ibm.com/developerworks/webservices/library/sw-sla.htm, April 2002.

[14] M. Paolucci, K. P. Sycara, T. Nishimura, and N. Srinivasan. Using daml-s for P2P discovery. In *Proceedings of ICWS*, 2003.

[15] M. P. Papazoglou, B. J. Kramer, and J. Yang. Leveraging web-services and peer-to-peer networks. In *Proceedings of CAiSE*, 2003.

[16] S. Ran. A model for web services discovery with QoS. *ACM SIGecom Exchange, Volume 4 Issue 1*, March 2003.

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

[18] C. Schuler, R. Weber, H. Schuldt, and H.-J. Schek. Scalable Peer-to-Peer process management - the OSIRIS approach. In *Proceedings of ICWS*, 2004.

[19] UDDI. UDDI technical white paper. *Technical report, http://www.uddi.org/*, 2000.

[20] W3C. Web Services Description Language (WSDL) version 1.2. *http://www.w3c.org/TR/wsdl12/*, 2003.

[21] Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *Proceedings of IEEE P2P'03)*, 2003.

[22] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Web engineering: Quality driven web service composition. In *Proceedings of WWW*, 2003.

[23] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, May 1995.