Managing Trust and Reputation in the XenoServer Open Platform

Boris Dragovic, Steven Hand, Tim Harris,
Evangelos Kotsovinos and Andrew Twigg

University of Cambridge Computer Laboratory, Cambridge, UK
{firstname.lastname}@cl.cam.ac.uk

January 13, 2003

Abstract

Participants in public distributed computing do not find it easy to trust each other. The massive number of parties involved, their heterogeneous backgrounds, disparate goals and independent nature are not a good basis for the development of relationships through purely social mechanisms.

This paper discusses the trust management issues that arise in the context of the XenoServer Open Platform: a public infrastructure for wide-area computing, capable of hosting tasks that span the full spectrum of distributed paradigms. We examine the meaning and necessity of trust in our platform, and present our trust management architecture, named XenoTrust.

XenoTrust follows the same design principles that we are using throughout the XenoServer project: it provides a flexible platform over which many of the interesting distributed trust-management algorithms presented in the literature can be evaluated in a large-scale wide-area setting.

1 Introduction

The XenoServer project [9] is developing a public infrastructure for wide-area distributed computing, creating a world in which XenoServer execution platforms are scattered across the globe and available to any member of the public. Users of our platform are able to run programs at points throughout the network in order to reduce communication latencies, avoid network bottlenecks and minimize long-haul network charges. They can also use it to deploy large-scale experimental services, and to provide a network presence for transiently-connected mobile devices. Resource accounting is an integral part of the XenoServer Open Platform, with clients paying for the resources used by their programs and server operators being paid for running the programs that they host.

The public and open nature of the platform imposes a need for a trust management system. In the real world, servers and clients operate autonomously. Servers may be unreliable; they may try to overcharge clients, may not run programs faithfully, or may even try to extract client secrets.

Clients may attempt to abuse the platform; they may try to avoid paying their bills, or to run programs with nefarious, anti-social or illegal goals.

This paper covers the trust and reputation management architecture that is used in the XenoServer Open Platform. In Section 2 we introduce the context in which we are doing this work and then in Section 3 we identify the threat model within which XenoTrust is designed to operate. Section 4 presents the XenoTrust architecture and the notions of trust and reputation it manages. In Section 5 we discuss the implementation choices and trade-offs that exist. Section 6 describes related work and Section 7 concludes.

1.1 Contributions

This paper makes two primary contributions:

- Firstly, as we shall see in Section 2, the architecture of the XenoServer platform sets it apart from those within which existing distributed trust-management systems operate. Unlike simple peer-to-peer recommendation services, we are concerned with running real tasks on real servers for real money within a federated system whose constituent parts may have different notions of "correct" behaviour.
- Secondly, within this setting, the XenoTrust architecture provides a trust-management system which accommodates flexible policies for allowing its participants to derive their own context-dependent notions of one anothers' reputations. This is again in contrast to existing systems which either assume either that reputation is a global property or that the way in which it is assessed is fixed.

2 XenoPlatform Overview

Figure 1 illustrates the high-level architecture of the XenoServer Open Platform, distinguishing the various rôles and interfaces involved. On the left hand side we see a XenoServer, on the right hand side a client, and at the top an entity called XenoCorp. XenoServers host tasks that are submitted by clients and XenoCorp acts as a trusted third party. For exposition, it is easiest to assume a single XenoCorp. However our architecture is designed to support multiple competing entities, providing that they follow the same basic interfaces. This is analogous to the way in which the commercial world supports many distinct and competing institutions and currencies.

To set the general scene, it is important to realize the separation between a XenoCorp and the organizations running XenoServers. The former provides authentication, auditing, charging and payment, and has contractual relationships with clients and with XenoServer operators – much as

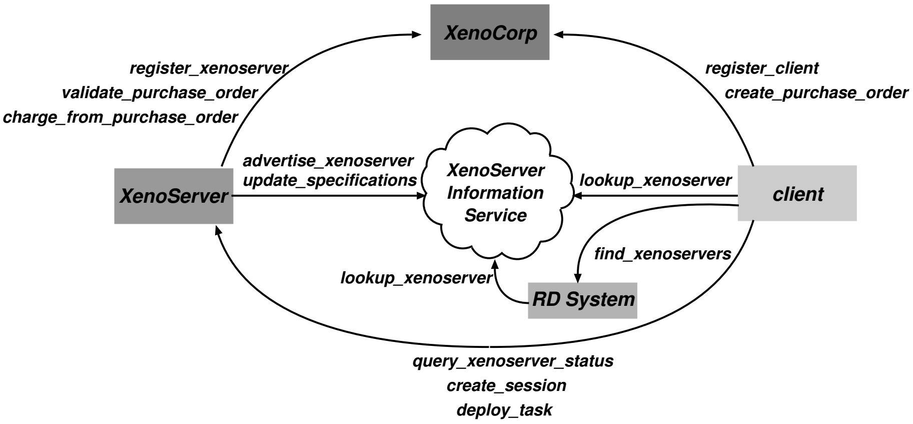

Figure 1: The XenoServer Open Platform

VISA or MasterCard act as intermediaries between credit card holders and the merchants from which they make purchases. It is the third party trusted by both clients and servers.

The XenoServers themselves may be run by disparate organizations, akin to the way in which server hosting facilities currently operate. XenoServers function on a commercial basis using well-maintained machines with long-term network presence – not in "spare cycles" on users' desktop systems. This, along with the existence of XenoCorps, is an important distinction between our system and "peer-to-peer" distributed computing projects.

In the center of Figure 1 we see the XenoServer Information Service. This acts as an intermediary between XenoServers, who advertise information within it, and clients, who perform queries on the service. There are no architectural conventions requiring that there be only one XIS although we envisage that this situation will develop naturally since it makes sense for XenoServers to advertise as widely as possible, and for clients to draw on the largest possible information base when performing lookups.

The information held in the XIS takes the form of low-level information about particular XenoServers – for instance their location and an indication of the facilities that they offer. Queries take simple forms such as "Which servers claim to be in the UK?" or "Which servers support IA-64 tasks?". The timescales over which XIS updates are made and the currency which queries should enjoy follows much the same model as the existing Domain Name Service (DNS). That is, we expect that XenoServer's advertisements within the XIS will be updated on the order of hours or days and that it is acceptable to use aggressive caching within the XIS. Of course, this model raises some of the same problems that occur with the DNS – for instance, who should operate and fund it. As we shall see, these same questions also arise for XenoTrust and we shall return to them in Section 5.

The initial XIS implementation is a distributed storage service optimized under the assumptions that (i) writes are always total rewrites, (ii) XenoServers arrange that there is only ever one writer for each piece of data, (iii) reads of stale data are always either safe or can be verified and (iv) information held in the XIS is for use by tools rather than humans. Self-certifying names [10] are used to ensure the authenticity of retrieved information and to allow clients to complain to the XenoServer's XenoCorp about inaccurate advertisements.

Although there is nothing to stop clients from using the XIS directly to select appropriate XenoServers, or indeed from contacting prospective servers directly without introduction, we anticipate that most will make use of one of a number of Resource Discovery (RD) systems. These perform a matchmaking process between clients and XenoServers, receiving specifications of clients' requirements and using a search algorithm to identify a number of suitable servers. As with the XIS, the information returned is based on the advertisements received; clients then query the suggested XenoServers directly to obtain up-to-date resource availability and a spot price. However, unlike the XIS, the queries are at a much higher level, for instance corresponding to "Find a XenoServer for a networked game of Quake that is suitable for clients A, B and C".

There may be multiple such RD Systems, either for simple competition (as exists between online search engines) or for specialisation to particular kinds of client, XenoServer or task. The algorithm with which the mapping is performed is entirely dependent on the RD mechanism.

In companion papers we introduce this high-level architecture in more detail [11] and describe our prototype XenoServer platform [3].

3 Threat model

We assume that some out-of-band mechanism is available to provide assurance of XenoCorp's identity; for instance, our first XenoCorp will interact over *https* connections with a suitably signed certificate. In turn, one of XenoCorp's rôles is to authenticate XenoServers and clients to its satisfaction; in our prototype deployment this will be evinced by signed credentials that can be tied to the issuing XenoCorp.

This means that both XenoServers and clients have a way of mapping one another back to a real-world identity, either directly or through a XenoCorp. This capability ultimately provides a way in which they can carry complaints through to non-technical solutions in the judicial system. What, then, is the rôle of XenoTrust? In answering this question, and identifying the requirements of XenoTrust, we shall consider the major kinds of threat which exist to the participants in platform.

Threats to XenoServers. XenoServers face the clearest threats. Disreputable clients may try to place tasks on them but then not pay. They may also try to run tasks that use XenoServers as the source of some nefarious activity on the network, perhaps to try to conceal the actual originator. Server operators must therefore take care over both the jobs that they accept, and their management of auditing information for investigating any problems reported. If security flaws are discovered in the XenoServer software then clients may try to exploit those.

Threats to clients. Clients may find that the jobs they submit to a particular XenoServer are not executed faithfully. A malicious server operator may overcharge for resource consumption. Others operators may simply have been unlucky and bought unreliable machines to use as XenoServers. In either case, technical solutions are not present; to date, software mechanisms such as proof carrying code have focussed on ensuring the safety properties of clients, rather than of servers. Although "trusted computing" schemes such as TCPA [15] provide mechanisms for ensuring tamper-proof execution of core system components, they cannot prevent software errors, or faults in the underlying hardware. In any case, such schemes do not fit with our goal of allowing the XenoServer platform to host code written in a broad range of existing distribution formats which may not be supported on the trusted platforms that arise.

Threats to XenoCorps. If there are many XenoCorps then disreputable clients or server operators may try to register with each in succession if they continue to be ejected from the platform.

3.1 Discussion

Where these threats involve an identifiable component misbehaving, either a XenoServer or a task running on a XenoServer, then it is of course possible for the other parties involved to complain to their common XenoCorp. Problems can arise, however, in a number of ways. Firstly, it is quite unrealistic to expect all of the participants to agree on common standards of behaviour – even if a single "acceptable use policy" could be enforced by XenoCorp then it would most likely be written in a natural language. Secondly, the volume of complaints may be large and the distinction between trivial and important ones unclear. Finally, issuing a complaint is difficult when the source of misbehavior is not straightforward to determine.

Consider, for example, a situation where a client is running a POSIX application on a XenoServer – perhaps within the XenoLinux environment that forms part of our initial deployment. Monitoring suggests that the XenoLinux environment in question starts requiring more network bandwidth than initially specified when the job was started. There are several possible explanations:

- The client's application is misbehaving as part of its normal operation, perhaps acting as a server subject to an unexpectedly high level of demand.
- Perhaps the client's estimate of the resources necessary is simply inaccurate. The XenoServer will be multiplexing its physical resources between a range of tasks and predicting context

switch overheads or the effects of contention in caches and shared buffers is difficult. Similarly, there will be subtle differences between the performance of processors or execution engines even if they accept the same execution formats (for instance between a Pentium-3 and Pentium-4 processor, or between a JVM from Sun and a JVM from IBM).

- Perhaps the client's application has been compromised and is thus being caused to misbehave.
- The XenoServer itself could have been compromised and be misbehaving.

One can also envisage more intricate causes of problems. For instance, it would be naïve to assume that the XenoServer software will not exhibit the kinds of fault that occur in mainstream operating systems. One task could exploit such a fault and plant code to cause latent problems for subsequent jobs, to extract data from them, to consume their resources, to masquerade as their owner or simply to cause them to fail. Such problems can often only be tracked down after the fact, once forensic evidence of such an exploit has been gathered.

In a setting like this it is very difficult to identify which of these explanations is the actual cause of the higher resource demands. Any naïve decision making policy at this stage would open numerous possibilities for misuse and deception by sophisticated attackers.

4 XenoTrust Design

Fundamental to all of these concerns is a notion of *trust.* Should a XenoServer trust a prospective client to submit reasonable code for execution? Should a client trust that a XenoServer it's considering will execute its code correctly? Should a XenoCorp trust that a prospective XenoServer operator or client is going to be a trouble-free participant? As is usually observed, these questions are all subjective and context dependent.

We take a two-level approach to managing trust in the XenoServer Open Platform by distinguishing *authoritative* and *reputation-based* trust:

- *Authoritative trust.* This boolean property is established between a XenoCorp and the clients and servers that register with it. It is evinced by the credentials issued by the XenoCorp. These can be validated by any of that XenoCorp's other clients and servers.
- *Reputation-based trust.* This is a continuous property which quantifies, in a particular setting, the trustworthiness that one component ascribes to another.

We discuss these aspects of our trust management architecture in Sections 4.1 and 4.2 respectively.

4.1 Authoritative Trust

As was illustrated in Figure 1, each XenoCorp acts as an authority which is responsible for registering the clients and servers that wish to participate. The XenoCorp has ultimate authority for registering and ejecting participants and for establishing authoritative trust between them and itself.

It issues credentials to correctly-validated participants and can rescind these credentials should the participant de-register from the system or be ejected from it. Note that the validation step indicated in Figure 1 and performed between a XenoServer and XenoCorp before starting a session provides an opportunity to detect rescinded credentials.

This “authoritative trust” underpins the XenoServer Open Platform because it confirms that a XenoCorp is aware of a “real-world” identity bound to the participant. This enables recourse through the courts should it be necessary and – so long as new real-world identities cannot be created trivially – makes it harder for a participant to re-register once ejected.

An analogy with the real world is that an individual may decide to be law-abiding and socially responsible, or they may decide to behave objectionably and thereby risk prosecution by a globally acceptable authority – the judicial system. In important transactions, interaction is often underpinned by the examination of identification documents or checks for criminal records and so on – that is, by evidence of authoritative trust between the state and the parties involved.

XenoCorps will differ in exactly what information is required at registration time – in a commercial setting this may be a credit card number registered to the participant, or physical evidence of a passport or picture-ID enabling charges to be traced.

4.2 Reputation-based Trust

Observing the same analogy as presented above, people in the real world, whether offenders or not, have a reputation which influences their relationships with other members of society. In an extreme case, someone might be believed to be an offender, but not yet be convicted. This notion of reputation is subjective, as listeners attach different significance to what they hear. Mrs Smith’s reputation for always being ten minutes late may be of little consequence to her friends and family (who, knowing this fact, may already incorporate it into their own timekeeping arrangements), but be of great importance to a potential employer. These observations carry over to how interactions are managed in the XenoServer Open Platform and the way in which clients and XenoServers form opinions about one another.

The second layer of the XenoTrust model is this form of “reputation-based trust”. Compared to the authoritative trust relationship, this second-level is established on a point-to-point and highly dynamic basis between individual XenoServers and clients. Unlike registration with XenoCorp, it is entirely up to participants to choose whether or not to use the system – as in the real world, some participants may choose to bear what others believe to be an unacceptable risk or to rely on other sources of information.

Conceptually, each component forms a *reputation vector*, which stores values about different aspects of the reputation of other components in the platform. Ordinarily, individual clients and servers will build up this information locally as they interact with others. Their views will be subjective, perhaps with some clients favouring one aspect of a server's behaviour (say, factually correct price advertisements) and others favouring other aspects (say, a reliable service with low jitter during network transmission). It is the participant's responsibility to decide how to interpret the scores in its reputation vector, e.g. to decide at what point a counterpart is deemed unworthy of further interaction.

Participants will also tell others about their observations. Again, it is up to participants to decide how to deal with the reports that they hear. However, the community as a whole may benefit from participants exchanging reputation information with one other, each using that information to influence their own reputation vector. For instance, when a new member arrives in the XenoServer Open Platform, and is looking to co-operate with other components, it is useful to give it access to existing participants' reputation information.

Of course, a direct implementation of such a scheme would not be practical an any large setting. Broadcasting information from every participant is unfeasible. In the XenoServer platform many participants will interact with only a small subset of others and will only need to form trust judgements about them. Instead, we will present the operation of XenoTrust first of all at the level of the operations that it exposes to participants and we will then discuss, in Section 5, some of the implementation and deployment options that exist.

Our model involves three steps; *statement advertisement* in which one participant provides XenoTrust with information about another, *rule-set deployment* in which a participant describes how to compute its reputation vector and then *reputation retrieval* in which entries from its reputation vector are actually evaluated and delivered to the participant.

Statement advertisement. We define a *statement* as the unit of reputation information that components advertise to others. A statement is a tuple, including the advertiser's identity, the subject component's identity, a token representing the aspect of the subject's reputation that is being considered and a series of values which indicate the extent of this reputation. The tuple is signed using the advertiser's authoritative credentials to prevent forgery.

The interpretation of these tokens and values is a matter for convention between the components that are using XenoTrust. For exposition, we may consider statements such as $(C, X, \text{payment}, 1, 50)$ giving component C's view on how X scores in terms of making payments. The scores 1, 50 may indicate a maximum value of 1 on a $[0 \dots 1]$ scale based on an experience of 50 interactions. In other statements, C may report $(C, X, \mathrm{belief}, 0.8)$ meaning that C attaches a weight of 0.8 to what they hear from X. Again, the interpretation of these is a matter of convention by C.

Although self-certifying names are used to prevent forged statements, XenoTrust does nothing beyond this to ensure that statements are in any sense valid: as with real-world advertisements, users are under no compulsion to believe what they see or to pay any attention to it.

Rule-set deployment. Components are welcome to search XenoTrust directly for statements that have been made. For instance, before hosting a job from a particular client, a XenoServer may query the XenoTrust system for any statements that have previously been made about the client. Again, these would be simple queries of the form "return all the statements naming participant A". However, this scheme is far from scalable and – unless very extensive queries are made – it will only take into account direct statements made about the component in question.

Instead, the approach that we take is to move the computation of reputation vectors from the participants involved into XenoTrust itself. This allows the aggregation of information to take place within XenoTrust and, crucially, allows it to be updated incrementally as new statements are received. It also allows the results of this computation to be shared between participants who are using the same rulesets to derive their reputation vectors. Some examples of the kind of rules we anticipate are given within the example scenario considered in Section 4.3.

Reputation retrieval. The final step in using XenoTrust is for one component to query the rule-sets that it has deployed in order to retrieve elements from its reputation vector.

4.3 Example

Figure 2 provides an overview of how these stages of XenoTrust may be used. This example shows statements issued by five components – four servers (A, B, C, D) which have been hosting jobs on behalf of a client X. All of these statements are held in the XenoTrust system, available for public view.

Statements labeled "payment" refer to the server's view of whether the client is reputable when it comes to settling their bills, indicating how likely they believe an interaction with X is to proceed without problems in this regard (a value between 0 and 1) and how many times they have interacted with X (an integer). For instance, A has interacted with X 123 times and assigns X the maximum score. In contrast, B has interacted with X just once and assigns the lowest possible score. Furthermore, D has made additional statements indicating the value it places on statements made by each of the other three servers, again as a value from 0 to 1. In this case the client X has also made a statement about itself, claiming to have interacted a great many times

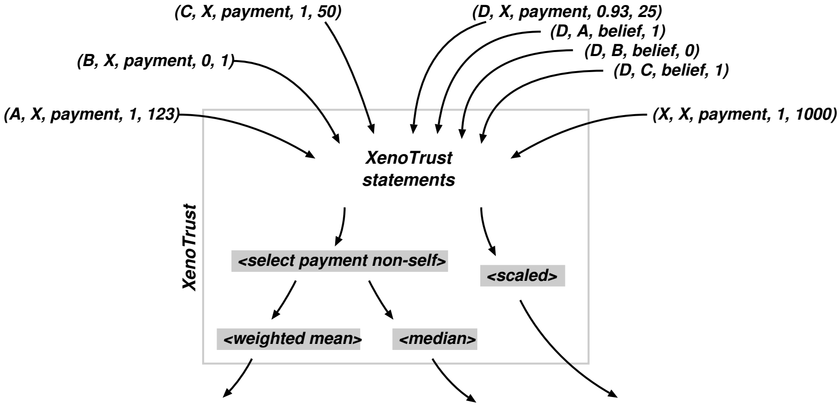

Figure 2: XenoTrust overview

and always successfully – other users of XenoTrust would be wise to attach little credence to this statement.

Various users of XenoTrust have installed different rule-sets for quantifying their trust in clients on the basis of the statements that have been made. In this case D is using a rule set "<scaled>" which combines the "payment" statements according to D's own "belief" statements. For instance B and X's statements would not influence D, but A and C's would. Other rule-sets select statements made by other parties (i.e. discarding X's self-referential statement) before combining them according to various averages.

4.4 Users of XenoTrust

The service provided by XenoTrust is valuable across the XenoServer platform in ways beyond its direct use by clients and by XenoServer operators:

Resource discovery. As we saw in Section 2 we anticipate that many clients will take advantage of search facilities provided by resource discovery systems. These RD services may use XenoTrust to determine the "trustworthiness" of a XenoServer as one of the axes along which a resource search may be performed. If an RD service specializes in a particular category of task – such as networked games – then this would influence the kinds of rule-set that it would deploy.

XenoCorps. To make valid decisions over when to consider ejecting a participant, a XenoCorp can use information gathered through XenoTrust. This creates a link between the authoritative trust level and the reputation-based trust level, whereby the former is influenced and ultimately modeled by the latter.

From the point of view of the XenoTrust service there is no distinction between XenoCorp and any other component in the system. Rule-sets used by XenoCorp and the algorithms employed for deciding whether a component should be ejected are implementation are policy specific and outside the scope of this paper. The level of sophistication at a XenoCorp can be expected to be proportional to the financial, commercial etc. sensitivity inherent in a particular setting; for example, the amount of anti-fraud insurance cover obtained should be proportional to the incentive to commit fraud.

The outlined inter-layer links ensure that misbehaving components, whether faulty or malicious, will eventually be penalized platform-wide by XenoCorp. An interesting aspect of the approach is that optional reputation-based trust is projected onto a mandatory authoritative trust level. The effect of this should be an incentive for components to participate fully within the XenoTrust model. Furthermore, if an inter-XenoCorp trust information sharing agreement is in place, offending components may suffer global consequences for misbehaving.

4.5 Discussion

Most widely spread attacks on reputation systems are from the family of so-called *Sybil attacks* [8]. Sybil attacks assume that a single hostile or faulty entity can present multiple indentities, and by using those, substantially influence the behavior of the overall system. As argued in [8], these attacks are prevalent in systems in which it is easy to come by fresh identities. However, we argue that Sybil attacks are not feasible in the XenoServer platform for two main reasons:

1. XenoCorp acts as a centralized, trusted, certification agency that validates all identities used in the platform. For an entity to fraudulently register multiple identities it would be neccessary to present XenoCorp with the same number of real world, unlinkable, legal identities. The nature of credentials required by XenoCorp depends on the level of security required against Sybil attacks; we can expect a balance to be drawn between the cost of validating credentials and the cost of a successful Sybil attack.

2. As participation in XenoTrust is optional and as the way in which each component uses reputation data obtained from XenoTrust is implementation-specific, it would be difficult for a hostile entity to accurately estimate the impact of an attack. In effect, an attacker would not be able to balance the costs of obtaining multiple identities against its financial gain. Furthermore, as the platform is expected to be dynamic in terms of participants, even analysis of historical behavior of the system would not gain an attacker deep insight into the future on which it could base its gain estimates.

A further possible kind of attack on XenoTrust is *shilling*; providing a financial incentive to real world entities for providing fake reputation data about a set of components to XenoTrust. The problem of uncertainty for the attacker, as described earlier, further influenced by the number of components in the system, would provide a strong disincentive for this type of attack.

5 Deploying XenoTrust

The previous section discussed the core architecture of XenoTrust. We will now turn to a number of more practical questions, implementation considerations and further issues.

5.1 XenoTrust implementation

There are several options that one could consider in terms of where XenoTrust will be hosted, how statements are advertised, and how rulesets are deployed and evaluated. XenoTrust statements could be stored centrally in each XenoCorp, or in a specialized component, or, indeed, it may be that XenoTrust would be implemented over the XIS (XenoServer Information Service). In this case, statements advertised are stored in XIS, and rule-set deployment and reputation retrieval are performed on XIS.

With XenoTrust one can imagine some people wanting to run storage nodes for themselves holding their statements, rather than casting them into the "best effort" XIS cloud. One possibility is for a XenoCorp to provide such a store and make it mandatory for its clients and servers to feed information back. Alternatively a "Which?"-style consumer magazine or a "Slashdot"-style online community could run one from which to promulgate its recommendations.

In the case of our prototype deployment, we anticipate that the same people who run the first XenoCorp would also run the first XIS and XenoTrust systems, providing the latter two as an incentive to encourage business on the former – much as existing DNS servers are run as a convenience to the Internet community without explicit transaction-based charging.

5.2 Funding XenoTrust

An interesting aspect of the operation of XenoTrust is how the system's operation will be funded in the XenoServer Open Platform. Resources in our platform need to be paid for in order to make the system economically practical.

One option would be to use a subscription-based system, where components would have to sign up for using the XenoTrust in advance and pay for it. A problem with this model is that all components are charged the same, which can be unfair for ones that use the system less.

Another possibility is to charge components whenever they advertise a statement. The disadvantage here is that participants that advertise a lot of – potentially valuable – information are discouraged from doing so, while components that overuse the query mechanisms for obtaining that information are not charged at all. Participants can also be charged in a per-query basis, but this model does not encourage components to advertise information. Another difficulty is that queries can vary from very simple to fairly complex, so figuring out the price of each query according to the deployed rule-sets may be difficult and/or costly.

Rewarding advertisement is in general rather difficult since there is no easy way to distinguish between credible statements and ones advertised simply for gaining credit. Using the "belief" component of the reputation of an advertiser may offer possibilities here.

5.3 Rule-set selection

The selection of a suitable set of rules or the language for defining them needs to take into account the computational cost of applying them, the possibility for sharing of this computation between users with similar rules and (perhaps) the possibility of XenoTrust forming a covert-channel communication mechanism.

In terms of concrete rule-sets that may be used, a promising direction is to use XenoTrust to support the evaluation of some of the policies being developed by the SECURE project [16]. This uses a lattice to represent trust values and allows principals to provide expressions that define their rules for delegation. For instance one principal might specify:

$$Pol(t_g, p) = (p = \text{Tim})?0.5 : (t_g(\text{Alice})(p) \sqcap t_g(\text{Bob})(p))$$

In this case t_g is the *global trust space*, conceptually the combination of all of the principals' current trust values and p is the principal about whom a trust decision is being evaluated. In this case if p is the principal "Tim" then a value of 0.5 is returned directly, otherwise the least-upper-bound of the trust values believed by Alice and Bob is returned. The use of a functional language should allow evaluation work to be shared between users whose policies co-incide.

Of course, we can do other things with this, but it provides an effective framework whilst still not requiring, on a global basis, a specific method of computation or set of trust values.

5.4 Load balancing

The XenoTrust system can also be used to facilitate load balancing within the network. Highly loaded servers will disobey QoS agreements more often than less loaded ones, as the resources on those servers will become more congested.

Thus, under the XenoTrust model, this will lead to more negative statements being issued about them, and, therefore, to a reduction of the tasks that will be submitted to them for deployment. This will help reduce their load, as the reputation of fresh and less loaded servers will increase for as long as they provide good service. This also provides an incentive for XenoServer owners not to over-estimate what work their machines can do – brief overloading could lead to long-term underloading, or to their having to lower their prices.

This is an intriguing use of a reputation system, although it throws up a large number of further issues. We hope to investigate these further in the future.

6 Related Work

Over the last ten years, several researchers have developed trust models and implementations based on them. Josang provides definitions and categorization of several kinds of trust [12]. One of the first trust models and formal definitions of trust in electronic communities was introduced by Marsh [13]. His model, based on social and psychological properties, is considered rather complex, essentially theoretical and possibly impractical. Beth et al. suggest a theoretical model and methodology to represent and derive trust relationships [4].

Real-world paradigms that incorporate trust management subsystems include on-line auctions and retailers, like eBay and the `amazon.co.uk` marketplace. In such systems, buyers and sellers can rate each other after each transaction. The reputation of each participant is computed as the average of the scores reported over a specific amount of time – e.g. the last six months – and stored in a centralized repository. Kasbah [7] introduces the use of designated “regulator agents” that roam the marketplace to ensure that participant agents do not misbehave.

Systems like Kasbah and the on-line auctions and retailers have very limited expressiveness, as they assume that all participants agree on the criteria on which such reputations are based, and that they attach equal credence to everyone’s statements. Moreover, reputation in those systems is usually single-dimensional, and their scalability is restricted, as the central repository or agent that stores the reputation information is a single point of failure, and has to grow in proportion to the number of participants.

Another research avenue relates to peer-to-peer systems. In the model suggested by Rahman and Hailes, each participant uses an agent to decide both which participants to trust, and which other agents’ opinions to trust [1]. Even though implementation issues are not discussed in detail in that paper, reputation statements advertised are somehow broadcast or multicast between the participants and are then stored in each peer independently. This is inefficient in terms of network traffic as well as information redundancy; in the XenoServer system, many participants will never interact with one another and need not know one another’s reputation, while statements are stored in XenoTrust rather than in the participants themselves. Yu and Singh [17] propose a social mechanism of reputation management relying on agents that exchange and assess reputation information on behalf of the users. The model proposed by Aberer and Despotovic is the one that is closest to ours [2]. Our approach is fundamentally different from all the above, as in the

XenoServer Open Platform all entities are associated with a real or legal identity, and operate in a pseudonymous rather than totally anonymous manner.

Approaches like the PGP system, the X.509 framework, PolicyMaker [6] and KeyNote [5] use a different notion of trust, as they focus on techniques to formulate security policies and credentials, determining whether particular sets of credentials satisfy the relevant policies and deferring trust to third parties. Maurer proposes a probabilistic alternative to the above models [14]; trust management in that context is tightly coupled with hard security and the mapping between keys and identities. In such systems, identity-based certificates create an artificial layer of indirection between the information that is certified (which answers the question "who is the holder of this public key") and the question that a secure application must answer ("can we trust this public key for this purpose?"). Hard security approaches help establish that the party one is dealing with is authenticated and authorized to take some particular actions, but do not ensure that that party is doing what is expected and delivering good service.

Our system differs fundamentally from these in that it combines hard security trust and soft, reputation-based trust, while maintaining a flexible and scalable architecture.

7 Conclusion

The XenoServer Open Platform allows users to deploy and run computations on a large number of globally dispersed XenoServers, while performing accurate accounting and charging for the resources consumed. In such an environment, it is crucial to have a model of trust which allows individual components to decide whether to interact with each other. We have presented XenoTrust, a two-layer model which combines authoritative and reputation-based trust: the former prevents the Sybil attacks that most peer-to-peer and other ad-hoc open platforms are defenseless against; the latter avoids the notion of a global interpretation of "trust" and "risk". This is crucial in a large-scale computing platform with users and suppliers from many cultures and jurisdictions. It gives individuals the flexibility to make whatever game-theoretic trade-offs they wish.

The design is simple and should lead to a straightforward implementation. However the ultimate test will take place when we make our initial public deployment of the XenoServer Open Platform. Please contact us if you would like to be involved.

References

[1] Abdul-Rahman, A., and Hailes, S. Supporting Trust in Virtual Communities. In *Proceedings of the Hawaii International Conference on System Sciences 33, Maui, Hawaii (HICSS)* (January 2000).

[2] Aberer, K., and Despotovic, Z. Managing Trust in a Peer-2-Peer Information System. In *CIKM* (2001), pp. 310–317.

[3] Barham, P. R., Dragovic, B., Fraser, K. A., Hand, S. M., Harris, T. L., Ho, A. C., Kotsovinos, E., Madhavapeddy, A. V., Neugebauer, R., Pratt, I. A., and Warfield, A. K. Xen 2002. Tech. Rep. UCAM-CL-TR-553, University of Cambridge, Computer Laboratory, Jan. 2003.

[4] Beth, T., Borcherding, M., and Klein, B. Valuation of Trust in Open Networks. In *Proceedings of the 3rd European Symposium on Research in Computer Security – ESORICS '94* (1994), pp. 3–18.

[5] Blaze, M., Feigenbaum, J., and Keromytis, A. D. KeyNote: Trust Management for Public-Key Infrastructures (Position Paper). *Lecture Notes in Computer Science 1550* (1999), 59–63.

[6] Blaze, M., Feigenbaum, J., and Lacy, J. Decentralized Trust Management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy* (May 1996), pp. 164–173.

[7] Chavez, A., and Maes, P. Kasbah: An agent marketplace for buying and selling goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96)* (London, UK, 1996), Practical Application Company, pp. 75–90.

[8] Douceur, J. R. The Sybil Attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems, Boston, MA* (March 2002).

[9] Fraser, K. A., Hand, S. M., Harris, T. L., Leslie, I. M., and Pratt, I. A. The Xenoserver computing infrastructure. Tech. Rep. UCAM-CL-TR-552, University of Cambridge, Computer Laboratory, Jan. 2003.

[10] Fu, K., Kaashoek, M. F., and Mazieres, D. Fast and secure distributed read-only file system. *Computer Systems 20*, 1 (2000), 1–24.

[11] Hand, S., Harris, T., Kotsovinos, E., and Pratt, I. Controlling the XenoServer Open Platform, November 2002. To appear in the Proceedings of the 6th International Conference on Open Architectures and Network Programming (OPENARCH), April, 2003.

[12] Josang, A. The right type of trust for distributed systems. In *Proceedings of the 1996 New Security Paradigms Workshop.* (1996), C. Meadows, Ed., ACM.

[13] Marsh, S. *Formalising Trust as a Computational Concept.* PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[14] Maurer, U. Modelling a Public-Key Infrastructure. In *ESORICS: European Symposium on Research in Computer Security* (1996), LNCS, Springer-Verlag.

[15] Trusted Computing Platform Alliance (TCPA): Main Specification, Version 1.1b, 2002. TCPA Specfciation, available at `http://www.trustedcomputing.org/docs/main\%20v1_1b.pdf`.

[16] Twigg, A. SECURE project overview. http://www.cl.cam.ac.uk/Research/SRG/opera/projects/index.html.

[17] Yu, B., and Singh, M. P. A Social Mechanism of Reputation Management in Electronic Communities. In *Cooperative Information Agents* (2000), pp. 154–165.