

Distributed Dynamic Protection of Services on Ad Hoc and Peer to Peer Networks

Jimmy McGibney and Dmitri Botvich

Telecommunications Software & Systems Group,
Waterford Institute of Technology, Waterford, Ireland
{jmcgibney, dbotvich}@tssg.org

Abstract. A collaborative system for dynamic refinement of security in peer-to-peer and mobile ad hoc networks is described in this paper. This is based on a closed loop system where live distributed trust measures are used to modify access control settings in a changing threat environment. A service oriented trust overlay architecture and model underlies this system. In this model, services have associated trust thresholds – the more sensitive the service, the higher the threshold. The results of simulations of the dynamics of this kind of system are presented and a variety of algorithmic approaches to managing trust are analysed and discussed. It is demonstrated that this dynamic system has the potential to enhance security and access control efficiency and that it displays properties of robustness when faced with malicious entities that attempt to corrupt the system.

Keywords: Intrusion detection, distributed trust management, network security.

1 Introduction

With the advent of mobile ad hoc and other unstructured peer-to-peer systems, existing approaches to security are no longer sufficient. It cannot be assumed, for example, that participants always have access to centralised resources such as a public key infrastructure to validate credentials. At the same time, traditional perimeter security is of little benefit. The risks are substantial with the use of shared radio spectrum, and there is a history of weak wireless security protocols.

In this paper, we present an approach to dynamic management of security that is based on distributed trust. Nodes offer services and peer nodes use them. Access to some such services require more trust than others. Having certain services enabled on a node could expose that node to various types of attack, even though it might have good protection mechanisms in place and be kept up to date with patches as new exploits are common and it is easy for users to misconfigure systems and services.

How trust information is shared and how it is handled by individual nodes is of course critical. We consider algorithms for sharing trust information according to individual nodes' preferences and also for nodes to effectively update trust. This needs to take into account to risk of some "bad" nodes deliberately attempting to corrupt the system, possibly in collusion with each other.

For the purposes of this work, we adopt a two layer model for communications between nodes. Nodes can either interact for service usage or to exchange trust information. For modelling purposes, each service usage interaction is a discrete event. As we will see in section 3, a logically separate trust management layer handles threat notifications and other pertinent information.

The remainder of this paper is organised as follows. The next section describes related work. Section 3 then outlines our trust overlay architecture and this is followed in section 4 with a detailed model of our system. Section 5 presents and discusses results of initial simulations to evaluate the system. Section 6 concludes the paper and discusses possibilities for future work.

2 Related Work

A significant body of literature has emerged in the recent years on trust and reputation systems. Much of this work attempts to model trust computationally so that it can be used in making decisions related to electronic transactions, in a sense mimicking people's well-evolved forms of social interaction. A recent paper by Jøsang et al. [1] surveys the state of the art.

Many open issues exist and challenges remain. The social concept of trust is very complex and sophisticated, perhaps deceptively so. Trust is closely related to many other social concepts such as belief, confidence, dependence, risk, motivation, intention, competence and reliability [2]. It is also interwoven with the areas of accountability [3] and identity [4].

There are currently several different definitions and interpretations of what trust means and how it can be used. In some cases, trust is modelled as a probabilistic measure indicating confidence in a certain type of behaviour [5], and this measure is used as a basis for deciding whether to rely on another entity. In other cases, "to trust" is taken to mean making the decision itself. In our work, we model trust as a vector of measures to allow for a variety of trust-related factors that might influence a decision, such as a score based on experience and reputation, service-specific scores, confidence in these scores and a measure of their recency (as trustworthiness might be expected to decay over time).

Several authors have proposed reputation systems specifically for peer-to-peer networks. Most of these systems either rely on a global view of reputation, via a centralised server, or attempt to estimate the global view by polling a certain number of other peers. For example, the *secure EigenTrust* algorithm [6] polls a set of M peers that are designated as score managers for each peer i . Our system differs from this in that we do not have any such designation of peers, and thus we avoid having to maintain complex management information. Likewise, Gupta et al. [7] propose a partially distributed system that depends on reputation computation agents. Damiani et al. [8] propose a more truly distributed system, though it depends on polling peers and computing reputation each time a peer tries to download a resource. Other systems, like Google's *PageRank* [9] are essentially centralised and peers do not directly participate at all. Repantis & Kalogeraki have recently proposed a decentralised reputation system [10].

Our system differs from these in that all collaborating peers can make referrals equally, and the trusting node makes a subjective determination of how to weight these – usually based on its level of trust in each recommending node, its recency, and a requirement for corroboration. In fact, different nodes may have their own various strategies for updating trust and making referrals. Crucially, there is no dependence on specific nodes (such as score managers) being always switched on or within range.

Several authors have also looked more specifically at trust in wireless ad hoc networks. The usefulness of distributing trust in this situation was identified by Zhou & Haas [11] and more generally for ubiquitous services by Stajano & Anderson [12]. Much of the focus to date on securing ad hoc networks has been on the protection of routing, and often on specific protocols. Yang et al. [13], for example, propose a collaborative solution that is applicable to a variety of routing and packet forwarding protocols. This system is reactive, dealing dynamically with malicious nodes. Reputation systems for mobile ad hoc networks have also been proposed (e.g. by Buchegger & Le Boudec [14]).

3 Trust Overlay

3.1 Representing Trust Information

In an ad hoc or truly peer to peer network, there is no way to store trust information centrally. In any case, trust by its nature is subjective and thus best managed by the entity doing the trusting. This decentralised trust management is the essence of distributed trust.

Rather than having binary off/on trust, as exists in typical public key infrastructures, we would prefer to more accurately mimic social trust by setting a more fuzzy trust level. Different services can then make appropriate decisions based on this trust level – e.g. certain actions may be allowed and others not. In our system, we model trust as a vector. In the simplest case, at least if there is just one service, this can be viewed as a simple number in the range (0,1). Each node may then maintain a local trust score relating to each other node of which it is aware. If node A's trust in node B is 1, then node A completely trusts node B. A score of 0 indicates no trust. Note that this is consistent with Gambetta's definition of trust as "a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action and in a context in which it affects his own action" [5]. If trust is to be a probability measure, then the (0,1) range is natural.

3.2 Architectural Overview

As in [15], we propose the overlay of a distributed trust management infrastructure on top of the service delivery infrastructure. With this architecture, a trust management overlay layer operates separately from all aspects of service delivery and usage. Two message passing interfaces are defined between the service usage layer and the trust management layer and another between the trust managers of individual nodes. The interfaces are as follows:

- (1) Experience reports: Service engine \rightarrow Trust manager
- (2) Trust referrals: Trust manager \leftrightarrow Trust manager
- (3) Policy updates: Trust manager \rightarrow Service engine

Service usage is as normal, with the trust information just influencing protection mechanisms. The trust manager gathers usage experience and uses this together with the experience of collaborators to inform its trust in other nodes.

We then need to have a system to build trust as nodes gain experience of each other or learn of each other's reputation. In our system, trust can be updated in two ways:

- 1) *Direct experience*: Following a service usage event between two nodes, each node updates its trust in the other based on a measure of satisfaction with the event.
- 2) *Reputation*: Node A notifies other nodes in its *neighbourhood* of the trust score that it has for node B. This will change significantly following a security event.

How this neighbourhood is defined is significant. The neighbourhood of a node is the set of nodes with which it can communicate or with which it is willing to interact. The choice of neighbourhood nodes is up to each individual node to decide, though these collaborations will normally be two-way and based on topology (which in a wireless environment usually relates to location). In our simulations, we distribute nodes on a plane and define neighbourhood for each node based on distance, so nodes share information with nearby nodes.

The main benefit of this system is in using these trust scores to tune security settings. Trusted nodes can be dynamically provided with more privileges than untrusted nodes. In our system, as mentioned, we model all interaction between nodes in terms of services. Each node then sets a threshold trust level for access to each service it offers. If the trust score of a node decreases, for example due to detected suspicious activity by that node, the fewer services are made available to that node.

4 Design Model

4.1 Assumptions

We present a model here for using trust information to enhance security through collaboration in a rich multi-service decentralised environment. One of the problems of modelling decentralised systems like ad hoc networks is that it is unrealistic to take a top-down "bird's eye" view. Thus we model the set of nodes as those nodes of which a specific node *is aware*.

As mentioned earlier, our approach is service centric. All relevant activity between network nodes is modelled as service usage [16]. Nodes offer services and peer nodes use them. We assume that neighbour discovery and routing services are in place. We also assume that some kind of service registration and discovery is available to allow nodes to reach an understanding of the set of services available. Handorean & Roman [17] have proposed a secure service discovery technique for ad hoc networks.

It should be noted that the assumption of a common trust threshold across all nodes that provide the same service is something of a simplification. Even having all nodes share an understanding of the relative meaning of trust threshold values is non-trivial.

Some authors, such as Kamvar et al. [6], have proposed normalisation strategies for recommendations systems, in an attempt to address this.

We also assume authentication of identity, at least to the extent that a malicious node cannot masquerade as another more trusted node. This could be done, for example, by having nodes exchange public keys on their first interaction (in fact the public keys could be used as node identifiers). Further messages between those nodes could then be signed by the sending node's corresponding private key so that the recipient could at least be confident that the sender is the same entity as that for which it has been building up a trust profile. Verifying real identity is unimportant – in fact many entities may have the expectation of anonymity.

4.2 General Model

Topology: Let $V_i = \{1, \dots, N_i\}$ be the set of nodes of which node i is aware. Some of these nodes will be *adjacent* to i , normally by reason of network topology. This can be modelled by an adjacency vector, $A_i = (a_{i,j})_{j=1, \dots, N_i}$, where $a_{i,j} = 1$ if pair (i, j) are neighbours and $a_{i,j} = 0$ otherwise.

Services: Let $S = \{S_1, \dots, S_M\}$ be a set of services that may be provided. Each node j provides a set of services $S^j \subset S$. Some nodes will just be service consumers, so S^j will in those cases be empty.

Trust threshold: Each service S_x has an associated *trust threshold* t_x , $0 \leq t_x \leq 1$.

Representing Trust: We denote the local trust that node i has in node j as $T_{i,j}$. Each other node $k \in V_i$ will maintain its own local view of trust in j , which may, as we shall see, influence the local trust of node i in j . Note that $T_{i,j}$ is a *trust vector*. In the simplest case, this can just be a number, but such a number may have other related attributes, such as confidence in the trust score or recency, or the service(s) to which it relates.

Trust initialisation: In the case where i has no prior knowledge of j , we will have $T_{i,j} = x$ where x is the *default trust*.

Trust decision (using trust in service protection): When node j attempts to use service S_x provided by node i :

- Service use is permitted if $f_x(T_{i,j}) > t_x$, where the function f_x maps trust vector $T_{i,j}$ onto a scalar number in the range (0,1)
- Otherwise node j is blocked from using service S_x

Trust update following service usage: After a service usage event, by node j on node i :

- positive outcome: $T_{i,j}$ is increased according to some algorithm
- negative outcome: $T_{i,j}$ is reduced according to some algorithm

In general, following service usage, we update $T_{i,j}$ according to:

$$T_{i,j} \leftarrow f_e(T_{i,j}E) \quad (1)$$

where E is a vector of attributes of the service usage event and f_e defines how trust is updated based on service usage experience.

Trust update following referrals by a third party:

Node i may receive a message from a third party node, k , indicating a level of trust node j . This can be modelled as node i adopting some of node k 's trust level in node j , $T_{k,j}$. In general, following such a third party recommendation, we update $T_{i,j}$ according to:

$$T_{i,j} \leftarrow f_r(T_{i,j}, T_{i,k}, T_{k,j}) \quad (2)$$

where f_r is a function defining how trust is updated. This trust transitivity depends on $T_{i,k}$, as node i can be expected attach more weight to a referral from a highly trusted node.

4.3 Algorithms

Some algorithms for trust update. Both the way trust is updated based on service usage, and how referrals are issued and handled, have a profound impact on the usefulness of this kind of trust overlay system. The best algorithms will of course depend on network topology, node mobility, the range and types of service on offer, the attack threat model, and security requirements. The potential proportion of “bad guys” in the system (i.e. compromised nodes) has an impact, as does the scope for collusion between them. In this paper, we consider some candidate algorithms and how well they work in a selection of topology, security, and attacker collusion scenarios. How well they work is assessed in terms of system stability and improved resistance to attack.

Note that nodes are autonomous in our model and each might implement a different algorithm for updating and using trust. It can also be expected that a node's behaviour in terms of handling trust may change if it is hijacked.

The following are some examples of trust update algorithms:

Moving average: Each new assessment of trust is fed into a simple moving average, based on a sliding window. Direct experience can be given more weight than third party referrals in the averaging if desired. More advanced moving averages are also possible, where old data is “remembered” using data reduction and layered windowing techniques.

Exponential average: Each new assessment of trust is fed into a simple exponential average algorithm. Exponential averaging is a natural way to update trust as recent experience is given greater weight than old values, and no memory is required in the system, making it more attractive than using a moving average. Direct experience can be given more weight than referrals by using a different (higher) parameter. In our initial simulations, we update trust based on experience as follows:

$$T_{i,j} \leftarrow \alpha S + (1 - \alpha)T_{i,j} \quad (3)$$

where parameter α , $0 \leq \alpha \leq 1$, defines the rate of adoption of trust based on experience. Note that having $\alpha = 0$ means that the trust value is unaffected by the experience. Having $\alpha = 1$ means that local trust is always defined by the latest experience and no memory is retained. In general, the higher the value of α , the greater the influence of recent service usage on the trust value maintained for that node. Lower values of α encourage stability of the system. In our initial simulations, we update trust based on referrals as follows:

$$T_{i,j} \leftarrow T_{i,j} - \beta T_{i,k} (T_{i,j} - T_{k,j}) \quad (4)$$

where β is a parameter indicating the level of influence that “recommender trust” has on local trust, $0 \leq \beta \leq 1$. Note that, the larger the value of $T_{i,k}$, (i.e. the more i trusts k), the greater the influence of k 's trust in j on the newly updated value of $T_{i,j}$. Note that, if $T_{i,k} = 0$, this causes $T_{i,j}$ to be unchanged.

Exponential average, per service parameters: Parameters depend on service. Here, some services are more revealing than others of user's trustworthiness. Usage of a service that provides little opportunity for exploitation (say a service that allows read-only access to a system) shouldn't have much impact on trust in the user.

No forgiveness on bad experience: This is a draconian policy: a node behaving badly has its trust set to zero forever. This could be considered for critical services.

Second chance (more generally, n^{th} chance): Draconian policies are generally not a good idea. Intrusion detection and other security systems are prone to false alarms. Also, good nodes can be temporarily hijacked and should be given the opportunity to recover. Thus a suitable variation on the “no forgiveness” algorithms is to keep a count, perhaps over a sliding time window, of misdemeanours. Trust is set to zero, possibly forever, on n misdemeanours.

Hard to gain trust; easy to lose it: To discourage collusion between bad nodes, there is a case for making it hard to gain trust and easy to lose it. Thus attackers will require lots of effort to artificially increase their trust scores, either by repeated benign use of low-threshold services or by issuing repeated positive recommendations about one another. Even a little malicious activity will cause trust to fall significantly.

Use of corroboration: To prevent an attack by up to k colluding bad nodes, we could require positive recommendations from at least $k+1$ different nodes.

Some algorithms for collaboration. As already indicated, collaboration in our system is by the issuing of trust referrals. Certain rules are required to make this system workable in possibly large scale distributed environments. Some candidate approaches are as follows:

Random destination and frequency: At random intervals, a node i chooses a random other node that it knows about, j , to which to issue a referral. This approach ensures even node coverage over time, and scalability can be ensured by agreeing common upper bounds on the frequency of referrals.

On request: Node i only issues a referral to node j when asked for it. From node j 's perspective, there is no guarantee that node i will respond.

On service usage experience: Node i only issues a referral relating to node k just after a service offered by node i has been used by node k . This means that the referral is based on fresh experience.

Only on reduction in trust: Referrals are only issued on detection of malicious activity. In this case, the trust system would act like part of a distributed intrusion detection system (IDS).

Each node specifies own trust receiving preferences: Here, each node specifies how often and under what conditions spontaneous trust reports are sent to it. A node may only wish to receive referrals if trust has changed significantly, for example.

5 Experiments, Results and Discussion

In this section, we present and discuss the results of some experiments to evaluate the stability of this system and its effectiveness in detecting attacks, including the incidence of false alarms. In particular we examine the following:

- Impact of having multiple services with different trust requirements.
- Influence of network topology and how a node's neighbourhood is defined.
- Security: attacker interaction with the reputation system, especially collusion between bad nodes to attempt to artificially raise their trust scores, and strategies to mitigate such collusion.

5.1 Multiple Services with Different Trust Requirements

We consider the situation where there are multiple services with different trust thresholds. We firstly just have three services (A, B, and C) so that we can more easily see the effects. We assign quite distinct trust thresholds to these: 0.2, 0.5 and 0.9, respectively.

We initially consider a 20-node fully connected network; i.e. every node has access to every other node to attempt service usage and to share trust referrals. In the next subsection, we repeat these experiments with a sparser network topology. A default initial trust score of 0.25 is chosen so that all nodes will have initial access to service A but will not have access to services B and C. In effect, nodes need to earn trust based on their usage of service A before they are allowed access to services B and C. To keep things straightforward, there are just two types of nodes. In Fig. 1 below, of the twenty nodes, two are 'bad' and the remaining eighteen are 'good'. We increase the number of bad nodes in Fig. 2.

We initially consider a fully connected network – i.e. every node has access to every other node to attempt service usage and to share trust referrals. In the next subsection, we repeat these experiments with a sparser network topology.

In the figures that follow, we plot the percentage of "good usage allowed" – the percentage of service usage attempts by good nodes that succeed. We also plot the percentage of "bad usage allowed" that is, the percentage of bad usage attempts that succeed. Ideally 100% of good usage attempts and 0% of bad ones should succeed.

To make the situation more realistic, we also model the fact that IDSs do not always get it right. Occasionally an attack goes unnoticed (false negative) or, alternatively, benign activity raises an alarm (false positive). We use simple Gaussian

(normal) distributions to model the characteristics of service usage that are being monitored by the IDS. In the case of good nodes, a number is tagged onto each service usage using a normal distribution with mean 2.0 and standard deviation 2.0. Similarly, in the case of bad nodes, a number is attached to the service usage using a normal distribution with mean -2.0 and standard deviation 2.0. The sign of this number is then used by the pseudo-IDS to “detect” whether there has been an attack.

Figs. 1 and 2 show the percentages of good and bad usages allowed, over all three services, for a fully connected topology. The number of bad nodes is greater in Fig. 2.

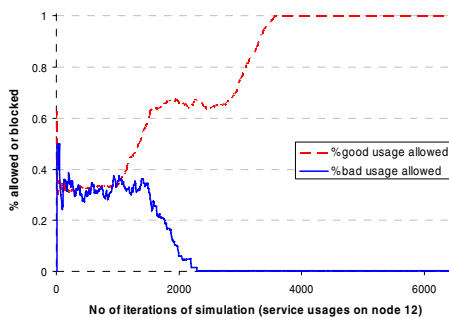


Fig. 1. % good usage allowed and % bad usage allowed, aggregated over all three services: 18 good nodes, 2 bad nodes, fully connected topology

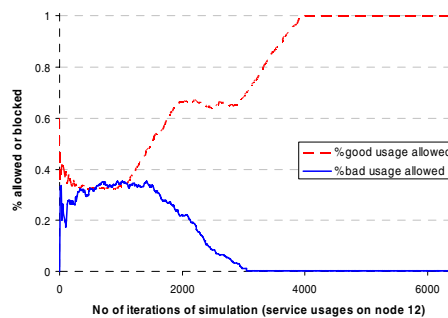


Fig. 2. % good usage allowed and % bad usage allowed, aggregated over all three services: 15 good nodes, 5 bad nodes, fully connected topology

In both these cases, the system works very well after trust has converged. When trust has been properly established, even with 25% of nodes behaving maliciously, all attempts by good nodes to gain access are allowed and the bad nodes are blocked. Note from the plateaus how the good nodes initially can mostly access just service A, then both services A and B, and finally all three services. Trust convergence takes a bit longer in the presence of bad nodes.

5.2 Influence of Network Topology and How a Node’s Neighbourhood Is Defined

In the examples shown above, the neighbourhood of a node is defined as containing every other node (i.e. a fully connected network). Ad hoc networks are of course often less well connected and thus it is useful to study topological effects. Fig. 3 shows a randomly generated twenty node planar topology where connectedness depends on distance. Note that nodes 19 & 20 are ‘bad’ nodes in the simulations shown here. Furthermore, node 5 is effectively “hidden behind” a bad node.

Fig. 4 shows the effects of topology on service access adjudication in the multi service environment described in the previous section. The main impact of this relatively sparse topology is that some nodes are more significantly impacted by the behaviour of bad nodes. The success rate for access attempts by good nodes does not quite reach 100%, due mainly to one node (node 5 in our example) being “hidden” behind bad node 19.

A main effect of having a sparser network topology is that convergence is slower. This will vary from node to node, and is less of a problem for node 12 than some others, as node 12 is fairly well connected. The higher the proportion of bad nodes, the more potential there is for a poor community view to be maintained of poorly connected but well-behaved nodes. Despite good behaviour, trust in some good nodes might never get very high and they may be blocked from some services.

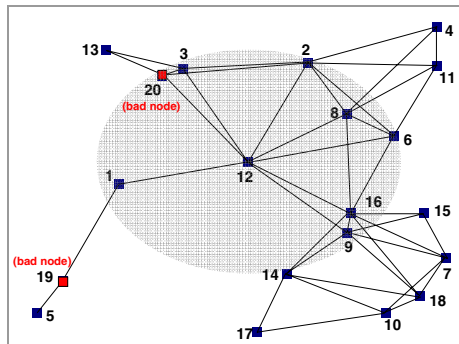


Fig. 3. Network topology used for simulations. The neighbourhood of node 12 is highlighted for illustration. Nodes 19 & 20 are bad nodes for some experiments.

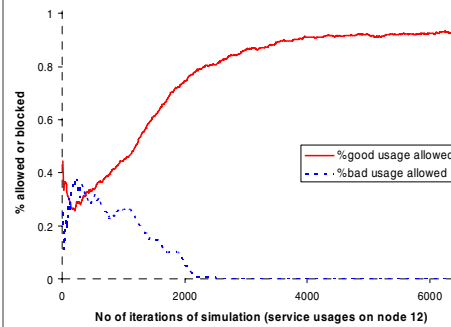


Fig. 4. % good usage allowed and % bad usage allowed, aggregated over all three services: 18 good nodes, 2 bad nodes, topology as in Fig. 3

5.3 Anti-collusion Strategies

Attacker interaction with the system is of particular interest, especially collusion between bad nodes to attempt to artificially raise their trust scores. Fig. 5 compares the success rate of bad nodes in attempting to gain access to services when they act independently with when they act in collusion. Without collusion, each bad node, as well as behaving badly when granted service access, sends out low trust referrals about all nodes. We model collusion as where bad nodes are aware of each other and send out high trust referrals about each other and low ones about other (good) nodes. Not surprisingly, they are more effective at gaining access when actively colluding.

Other strategies are possible to attempt to break the system. One possibility is for a bad node to act benignly for a time in order to build up its reputation and gain access to more sensitive services than would otherwise have been possible. Strategies for countering this include (i) applying a referrer trust threshold and (ii) making it harder to gain trust than to lose it. Fig. 6 shows how such strategies can help. Each of the plots shown in this figure is for the proportion of bad usage allowed by the system in the face of colluding bad nodes as discussed above. With either application of a referrer trust threshold (of 0.5) or applying a factor (of 5) that makes gaining trust harder than losing trust, a significant improvement is achieved.

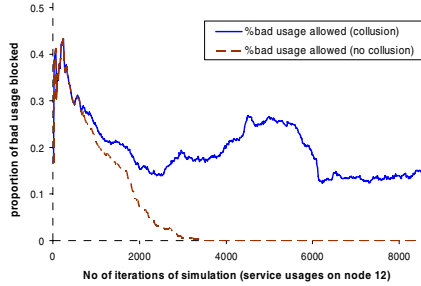


Fig. 5. % bad usage allowed where nodes actively collude compared with the same measure where nodes behave badly but do not work together. Aggregated over all three services: 15 good and 5 bad nodes.

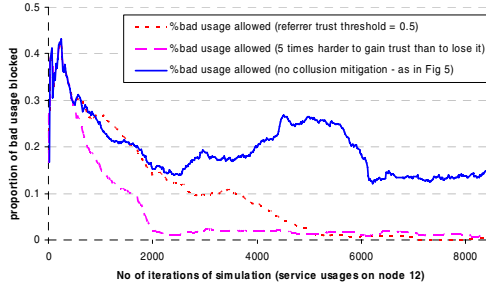


Fig. 6. % bad usage allowed where nodes actively collude. A comparison is shown between cases where there is no strategy to mitigate collusion and two anti-collusion approaches.

6 Conclusions and Future Work

We have described a collaborative approach to handling dynamic threats to services running on unstructured networks. In this, we have defined a dynamic system involving nodes, services and trust scores that appears to help to quickly and reliably isolate sources of attacks and restrict their access to the system. Services have associated trust thresholds – the more sensitive the service, the higher the threshold. New service users will initially just be able to access services with low thresholds, but repeated positive use of these can help the user gain sufficient trust to later access more sensitive services. The key to this kind of dynamic decentralised system is a stable means of updating trust scores that can learn from positive experiences and react to bad ones. Some inter-node services will be allowed and some will be blocked by either end, depending on the trust score it has determined for the other. Simulation results presented in section 5 are encouraging.

Our simulations to date have just used relatively simple strategies for sharing and updating trust information. There is significant scope to refine the dynamics of the system. New algorithms need to be developed and evaluated for trust updates. A protocol is required to specify how and when trust information is shared between nodes. Further experiments are needed to explore the effects of how node neighbourhoods are defined.

A fuller security assessment of this system is required. Further possible collusion strategies of bad nodes need to be examined. It is also assumed in our model that authentication is provided by underlying services, but the implications of this assumption needs to be evaluated in some detail. Account also needs to be taken of privacy, data integrity and availability issues.

Further work is also required to assess the performance implications of having this kind of trust overlay. Though the system was designed with scalability in mind (the use of neighbourhoods, and a lack of any centralised services), various aspects such as choice of neighbourhood need to be optimised.

Acknowledgements. This work is supported by the European Commission (*OPAALS* FP6 project) and by Science Foundation Ireland (*Foundations of Autonomics* project).

References

1. Jøsang, R., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 618–644 (March 2007)
2. Falcone, R., Castelfranchi, C.: Social trust: a cognitive approach. In: Castelfranchi, C., Tan, Y.-H. (eds.) *Trust and Deception in Virtual Societies*, pp. 55–90. Kluwer Academic Publishers, Dordrecht (2001)
3. Dingledine, R., Freedman, M., Molnar, D.: Accountability measures for peer-to-peer systems, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly (2000)
4. Douceur, J.: The Sybil attack. In: *Proc. Int'l. Workshop on Peer-to-Peer Systems* (March 2002)
5. Gambetta, D.: Can we trust trust? In: Gambetta, D. (ed.) *Trust: making and breaking cooperative relations*, pp. 213–237. Basil Blackwell (1988)
6. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: *Proc. 12th World Wide Web Conf.*, Budapest (May 2003)
7. Gupta, M., Judge, P., Ammar, M.: A reputation system for peer-to-peer networks. In: *Proc. 13th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)* (2003)
8. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P., Violante, F.: A reputation-based approach for choosing reliable resources in peer-to-peer networks, In: *Proc. 9th ACM conference on Computer and Communications Security (ASIACCS)* (November 2002)
9. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web, Technical report, Stanford Digital Library Technologies Project (1998)
10. Repantis, T., Kalogeraki, V.: Decentralized trust management for ad-hoc peer-to-peer networks. In: *Proc. MPAC 2006*, Melbourne (November 2006), <http://www.smartlab.cis.strath.ac.uk/MPAC/>
11. Zhou, L., Haas, Z.: Securing ad hoc networks. *IEEE Network* (November/December 1999)
12. Stajano, F., Anderson, R.: The resurrecting duckling: security issues for ubiquitous computing. *IEEE Computer (Supplement on Security & Privacy)*, pp. 22–26 (April 2002)
13. Yang, H., Shu, J., Meng, X., Lu, S.: SCAN: self-organized network-layer security in mobile ad hoc networks, *IEEE Journal of Selected Areas in Communications* (February 2006)
14. Buchegger, S., Le Boudec, J.-Y.: A robust reputation system for mobile ad-hoc networks, EPFL IC Technical Report IC/2003/50, EPFL (July 2003)
15. McGibney, J., Botvich, D., Balasubramaniam, S.: A Combined Biologically and Socially Inspired Approach to Mitigating Ad Hoc Network Threats. In: *Proc. 66th IEEE Vehicular Technology Conference (VTC)*, Baltimore (October 2007)
16. McGibney, J., Schmidt, N., Patel, A.: A service-centric model for intrusion detection in next-generation networks. *Computer Standards & Interfaces*, pp. 513–520 (June 2005)
17. Handorean, R., Roman, G.-C.: Secure service provision in ad hoc networks. In: *Proc. Int'l. Conf. on Service Oriented Computing (ICSOC)* (December 2003)