

# Validating a Trust-based Access Control System

William J. Adams<sup>1</sup> and Nathaniel J. Davis, IV<sup>2</sup>

<sup>1</sup>Department of Electrical Engineering and Computer Science,  
United States Military Academy, West Point, NY 10996

[Joe.adams@usma.edu](mailto:Joe.adams@usma.edu)

<sup>2</sup>Department of Electrical and Computer Engineering, Air Force Institute of Technology,  
Wright Patterson AFB, Dayton OH 45433 USA

[Nathaniel.Davis@afit.edu](mailto:Nathaniel.Davis@afit.edu)

**Abstract.** Over the last few years researchers have recognized the need for adaptive access control mechanisms for dynamic collaborative environments. As a result, several mechanisms have been proposed and demonstrated in academic literature. Although these mechanisms have been verified to perform as advertised, few of them have been validated to work within an operational environment. Using a decentralized trust-based access control system of their own design, the authors validated their system using a narrative technique to develop a realistic operational scenario. They tested the system within the scenario and then applied a cost and a success metric to the results to determine the efficiency of their mechanism. The results show how the authors' narrative approach and success metric combine to provide more efficient and effective analysis of how an access control mechanisms will perform when used in an operational environment.

**Keywords:** validation testing, access control, trust management

*The views expressed are those of the authors and do not reflect the official policy or position of the US Army, the US Air Force, the Department of Defense or the US Government.*

## 1. Introduction

During development, programmers typically verify a system's output to ensure that the system is performing as expected and producing credible results. To complete testing, however, a system must be validated to ensure that it performs reliably in situations that are present in its intended operational environment. In the case of a trust-based access control (TBAC) system, these situations include misbehaving users and temporary collaborations. This paper looks at the validation of a TBAC system

called the Trust Management System (TMS) and examines its performance in terms of effectiveness and cost to make correct decisions.

TBAC validation required more than getting the TMS to produce reputation or risk assessments. Our approach started by describing our system's expected operational environment and then deriving specific tasks that needed to be tested in that environment. After executing the tests, we applied specific metrics to measure the TMS' performance in that environment.

Dynamic collaborative environments (DCEs) formed to enable participants to share information while, at the same time, allow them to retain control over the resources that they brought with them to the coalition [1]. The trust management system (TMS) [2] developed through this research effectively implemented a decentralized access and permission management scheme. User permissions were determined using a combination of behavior grading and risk assessment without the need for preconfigured centrally managed roles or permission sets. Because the TMS tracked a user's behavior, using past behavior as an indication of future performance, no pre-configuration of users or resources was required.

The TMS also offered a unique ability to enforce multiple access levels without the burden of implementing and managing multiple cryptographic keys or hierarchies of roles. A user provided its peers customized views of its contents and services based on its trust profile and its individual assessment of the peer's trustworthiness. As the user's evaluation of a peer's reputation changed, the peer's access changed to safeguard the user's resources, restricting access to those peers that have contributed to the user's and the coalition's goals.

The contribution of this paper lies in its application of contextually derived objectives and requirements to validate a TBAC system. We use a narrative technique that is based on a realistic operational scenario. The scenario not only defines the operating environment but it also constrains testing so that results are pertinent and justified by real requirements. Having defined out test environment, we developed a success metric that assesses the TBAC system. Our results show that our TBAC implementation is far more effective and efficient than other current systems.

The rest of this paper is organized as follows. The next section describes our TMS, an implementation of a TBAC system, and then presents some related work in the field of trust-based access control and access control validation. Section 4 describes our validation test and the metrics used to gauge performance effectiveness. Finally, we conclude the paper and describe the future work that is ongoing with the TMS.

## 2. System Description

The TMS was developed to provide a trust-based privilege management mechanism in a fluid, collaborative environment. Users were initiated information sharing with a new peer in the DCE through an introduction process. This process in an exchange of lists of DCE members that can refer the user to strangers. Once introduced, the user collected behavior observations collected behavior observations from its trusted peers on members of the DCE called Feedback Items (*FI*). *FI* were weighted with the

reputation of the observer and placed in a temporally-ordered queue called a Reputation Indexing Window (*RIW*).

When a peer requested a resource, a user applied a quantitative method (called the 3Win method) to the *RIW* to compute a Reputation Index (*RI*) for that peer before allowing access to any of his or her resources. Once the *RI* was computed, the TMS stored the *RIW* in its trust store (*TS*). The *RI* was compared against the user's trust thresholds and the decision to extend or deny trust was made.

### 3. Related Work

Access control systems have been implemented to grant or deny the ability to use a resource or perform an operation in almost all computer systems. Before fielding, they have been verified to perform as expected given a wide range of statistically valid input. Few access control systems have been validated, however, because of the number and complexity of operating environments. One exception was dRBAC [3] that proposed an operational environment and then used this setting to derive the operational requirements for system testing.

TBAC systems used behavior grading to assess the trustworthiness of a prospective associate. They allowed or denied access based on a comparison of a quantitative reputation rating and a trust threshold. Previous work by the authors [2] discussed how the TMS was designed and verified to operate correctly. Other TBAC projects, such as SECURE [4] and Vigil [5] also verified the operation of their systems but stopped short of validating them in any realistic operational environment.

Validation testing was considered crucial to the success of a fielded system, as it provided the engineers and users some certainty that the system could withstand the demands of the specific operational environment and still perform as expected. Lo Presti [1] presented one method of using an operational scenario to derive user requirements for testing. The application of this method formed the first part of the validation process presented in this paper.

Once the requirements were derived, the system's success at accomplishing the tasks was measured quantitatively. Assessing the efficiency of an access control system [6] involved examining a ratio of three parameters: the number of correct decisions, the number of false positive decisions, and the number of false negative decisions. Linked to the efficiency rating, the cost of making decisions was also considered. This evaluation included the amount of memory and communications required by the system to make trust decisions. These criteria are explained in more detail in the next section.

### 4. Validation

Validation ensured that the system met the user requirements. In our case, validation guaranteed that the modules of the TMS worked together to make access control decisions correctly under a variety of network conditions. Validation differed from

verification testing in that the system was tested against operational requirements instead of purely quantitative comparisons.

The requirements used in validation testing came from two sources. The first source was verification testing. These requirements, derived in part from the analysis presented in previous work [2], placed the system in a test environment that simulated the target operational conditions. The points of failure identified in each module during verification testing were added to the validation test profile to determine the impact of a module's limitations on the system as a whole. The goal was that the system continued to operate or at least failed in a safe state when these points of failure were reached. For an access control system, such as the TMS, failing in the "closed to all" state was desirable, since it was better to deny access to everyone at the expense of false positive responses than to fail in the "open" position and suffer false negative responses, which were more costly.

The second source of validation requirements was an operational scenario. In our case, the scenario needed to involve mobile, collaborating users asking each other to share resources. Once the general conditions of the scenario were determined, we applied a narrative technique to construct the test environment for the system [1].

**Task 1:** A user should be able to enter the community.

Condition: A user enters a location with an established identity.

Standard: The user joins the community and can interact with altruistic users or the control plane until he or she establishes a reputation with other users.

**Task 2:** A user should be able to meet another user through the introduction process.

Condition: A community member meets another community member and wants to establish an association. Other members, known to one or both of the prospective associates as trusted peers, are available to provide references.

Standard: The prospective associates request and receive information on each other from their trusted peers. This information is processed to determine the reputation index of each other.

**Task 3:** A user should be able to move between sites (i.e., geographically separate sub-networks) and continue to operate.

Condition: A user enters a location with an established identity.

Standard: The user joins the community and can interact with established trusted peers, members of their own organization, altruistic users, or the control plane until he or she establishes a reputation with other users.

**Fig. 1.** Enumeration of Validation Testing Objectives

#### 4.1 Describing the Validation Test Objectives

First, the objectives that needed to be tested within the system were enumerated. These objectives addressed operational issues within the broad topic areas, such as mobility, network density, and general peer behavior. Objectives were expressed in

the form of task-condition-standard in order to be evaluated. Figure 1 presents three tasks that were included in the validation testing. The benefit of using the task-condition-standard format was that the task's context and the conditions for its success were explicitly described. This format was also extensible, so that tasks that could be performed in different contexts were identified, described, and evaluated separately. By describing the context of each task, we also helped build the operational scenario we used as the background to the test.

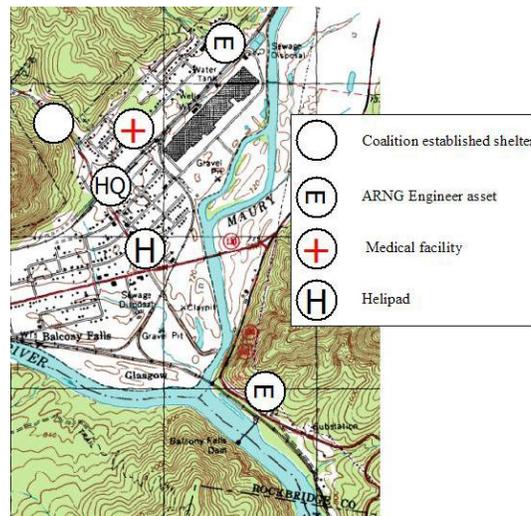


Fig. 2. Operational Scenario Map

#### 4.2 Operational Scenario

The scenario provided a framework for user requirements within realistic vignettes for the purpose of testing interaction. In this scenario, we detailed the composition and deployment of a notional disaster response task force. Since the system was specifically concerned with the access control of resources within a collaborative environment, users were assigned as resource providers in specific locations.

Figure 2 illustrates how a coalition might deploy in response to the notional emergency response situation. The operational scenario was developed using Training Scenario 2: Slow Building River Flood – Natural Disaster [7]. This scenario called for cooperation between a variety of government organizations and local volunteers to evacuate the inhabitants of a small riverside town and secure the town's infrastructure against damage. The local Emergency Management Services (EMS) office coordinated the efforts of law enforcement agencies, local fire department, volunteer rescue squad, and county health facility to form the coalition. EMS directed

the evacuation of the town’s inhabitants to the higher ground behind the town, where the Red Cross established a shelter. Medical units treated injured people and evacuated a senior citizen home, assisted by the helicopters and rescue squads. An Army National Guard (ARNG) engineer unit provided technical or specialist assistance to contain contamination from the town’s two sewage disposal sites and to reinforce the Balcony Falls Dam. The coalition formed using the Incident Command System (ICS) [8] and established a public information cell (PIC) to provide media services with information, in accordance with the ICS guidelines.

Location	EMS	ARNG	Fire/ Police	Red Cross
Shelter	👤 🖨️			👤
Tactical Ops Center (TOC)	👤 🖨️ 📄 📄 📄	👤	👤 🌐 📄 📄 📄	
Medical Facility			👤 🖨️ 📄 📄	👤👤👤
Helipad		👤 🌐 📄 📄		
Engineer 1 (Sewage Plant)	👤👤👤	👤	👤👤👤	
Engineer 2 (Balcony Falls Dam)	👤👤👤	👤		
Engineer 3 (Power Substation)	👤👤👤	👤		
Public Info Center (PIC)	👤 🌐		👤👤👤	👤

- 👤 People (👤👤👤 = transient)
- 🖨️ Printer
- 📄 Web access (HTTP/SSL)
- 🖨️ Intranet (HTTP)
- 🌐 Internet (HTTP)
- 📄 File server (HTTP)

Fig. 3. Coalition Locations and Resources

Although this scenario included severe inclement weather, members of the coalition and the outside populace were able to move about the scenario location. Communications, although unreliable, were present between the coalition locations and the unaffected “safe” areas.

Figure 3 provides an example of the resources that require access control. In terms of information resources, the coalition represented a hastily formed DCE. Users

possessed a variety of computing and communications platforms that utilized both wired and wireless communications. This research focused on the ability for users to access coalition resources and, similarly, for the resource owners to maintain control and protect their resources for the use of contributing coalition members. Validation testing analyzed the TMS from the role of the Tactical Operation Center's (TOC) file server to assess system performance.

Given the composition and deployment of the notional coalition, we distributed resources for coalition use. For example, the coalition might leverage the connectivity present at the police building to co-locate the TOC and coalition Headquarters (HQ). The community fire department and clinic would provide a location for the medical unit.

Finally, vignettes were written to frame the points within the scenario that tested the objectives. The scenario not only provided a realistic approach to developing the vignettes but also helped order the tests if need be. Lo Presti's narrative technique [1] mapped objectives to vignettes and this exercise is demonstrated in the next section.

Vignette	Task	1	2	3
Dave meets Alex, the task force engineer.		X		
Dave and Alex move to the Balcony Dam site to perform an assessment and rejoin network				X
Dave is introduced to Alex's colleague Bob at Balcony Dam. Bob shares his most recent assessment with Dave.			X	

Fig. 4. Mapping Tasks to Vignettes within the Scenario

### 4.3 Vignettes

A vignette described a scene within the scenario. Each vignette was developed to be as realistic as possible. Individual experiences contributed background details such as terrain, weather, and timing. Technical details were derived from more quantitative sources, however, and are described in Section 4.4.

A vignette established context within the test scenario in terms of time, location and actor participation. Most importantly, the vignette's description specified which task it was exercising for the purposes of the test. Figure 4 illustrates how three tasks (described in Figure 1) were tested within the scenario. Because the mapping of objectives to vignettes was done before the test started, the test ran through several vignettes in sequence, collecting data that was analyzed using the metrics described in the next section.

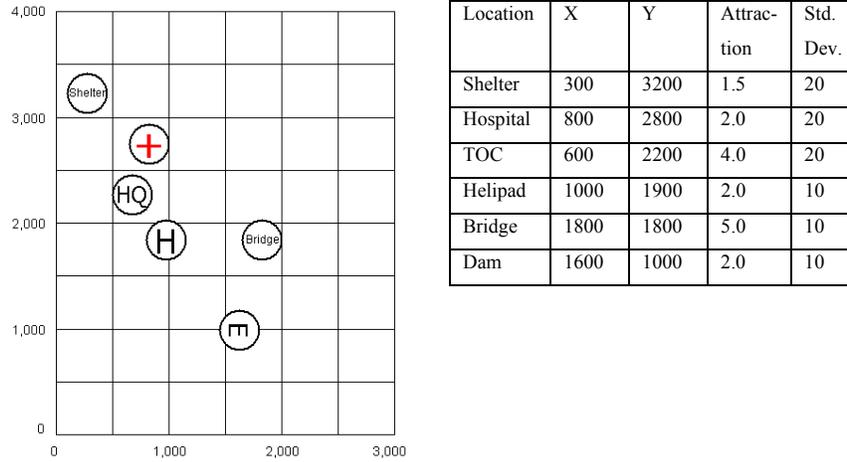


Fig. 5. Simulation Location Parameters

#### 4.4 Simulation Testing

Testing the TMS involved simulating user interaction in a mobile, dynamic environment. A four-step process was developed to create scripts that simulated behavior reporting and resource requests from the user’s peers. These scripts were used by both the TMS and the base system (described below) during the simulation.

The first step constructed a simulation area using parameters applicable to the operational scenario. The resulting Cartesian representation of the simulation area is shown in Figure 5. BonnMotion 1.3a [9] simulated node movement inside a 3,000 x 4,000 meter bounded area. Attraction points mimicked the effect of roads and facilities on nodal movement. Each Attraction Point was given an (x,y) coordinate, roughly corresponding to the map in Figure 2. The intensity value of the point weighted the attraction points so that a point with an x intensity level attracted nodes with a probability x times higher than an un-weighted point. Locations with higher intensity values were predicted to have heavier traffic. Nodes would approach an

Table 1. Individual Mobility Simulation Parameters

Duration	5000 secs.
Warmup	3600 secs.
Sim area	3000 x 4000 meters
Nodes	100
Speed	Min = 0.5 m/s Max = 10 m/s
Pause Time	60 sec. (Max.)

Table 2. Group Mobility Simulation Parameters

Average Nodes per Group	3 (Std. Dev. 2)
Group Change Probability	0.01
Distance to Group Center	2.5 meters

attraction point to a location within the point's standard deviation from a Gaussian distribution with a mean of 0 in meters.

The second step of the process involved three mobility models. The first model was static, meaning that the nodes (i.e., users) were homogeneously distributed over the simulation area and did not move. The second mobility model was the Random Walk (RW) model [10]. Simulations run using the RW model used the node speed and pause time parameters indicated in Table 1.

The reference point group mobility model (RPG) was used to simulate group mobility. In addition to the speed and pause parameters that it shared with the RW simulations, RPG required settings to describe the group dynamics in the simulation. These settings, shown in Table 2, show that the simulation had groups of one to five people. These groups were stable, in that the chance of people changing groups was low. Raising this probability skewed the RPG results toward those seen in individual mobility models, such as RW. The groups moved in open order, as the group members could be as far as 2.5 meters from each other. Each mobility model was executed on the same area mentioned above with and without attraction points. By executing the chosen parameters on the selected grid and mobility model, BonnMotion created a movement trace file for all the nodes in the network.

The third step in creating the scenario script fed the movement trace into BonnMotion's companion program, LinkDump. This program read the movement trace and applied a transmission range of 100 meters (selected to simulate 802.11b traffic) to determine when pairs of nodes could interact. The interaction file that was produced listed each node and its unidirectional communications link. Having each interaction listed twice reflected the "one-way" nature of the link. For example, if Alice could interact with Bob, two links were listed: "Alice to Bob" link was listed in Alice's part of the file and the "Bob to Alice" link was listed in Bob's portion. Having the links listed in this manner facilitated the next step, which was determining who could provide performance observations on whom.

The fourth and final step of the script generation process was to generate behavior and trust related network traffic. A reporting period was set and had each node generate a behavior grade once every ten seconds. A bin in a linked list represented each reporting period. Each bin was itself a linked list of behavior grades for that time period. A C++ program called Builder read the interaction list and populated the bins with observations and reports. These transactions placed associates in the TMS's Trust Store. Once an associate was known, the generated traffic represented the flow of behavior observations and reports.

As Builder read each link from the interactivity list, it called on the behavior model to determine a grade for the observed node for that reporting period. That grade was then adjusted based on the observer's behavior model. Once Builder had read the entire interactivity list and filled all of the appropriate bins, the behavior grades were written to a script file of network traffic formatted for the TMS.

Initializing the scenario required that the user (e.g., Joe) be introduced to someone by the KMS. Once Joe had an initial trusted peer (TP), he could participate in the scenario and make other TPs. This startup requirement was viewed as feasible; since Joe would be introduced to the people he would be working with when he arrived at the TOC, thus allowing Joe to start associating in the DCE.

Testing the TMS required a means of simulating service requests received by a resource providing DCE member from associates. Our simulation assumed the viewpoint of the server in the TOC and processed requests for files via a HyperText Transfer Protocol (HTTP) user interface. Modeling requests typically made to the resources illustrated in Figure 3, we examined the process of modeling a typical wireless system [11]. Given a generic inter-arrival rate we determined the number and period of resource requests in our notional scenario.

Requests were classified by the nature of information being sent or received. There were two general types of information: simple files and composite files. Simple files were single data type (text or graphics) files. Examples of these included email, web page (without graphics), or text files that were exchanged through an HTTP process. Composite files were multiple simple files linked together. Web pages with graphics were the most common examples. Each file type could come in one of three sizes. After determining the type and size of a request, the request duration was determined by approximating the times depicted in a “slow Internet” connection [12], again following Ost’s example.

**Table 3.** Probability and Duration of Resource Requests in a Simulated Collaborative Environment

Request Type	Probability	Duration (secs)
Small Simple File	0.6	1
Medium Simple File	0.1	2
Large Simple File	0.05	8
Small Composite file	0.15	1
Medium Composite File	0.075	6
Large Composite File	0.025	27

The test system simulated resource requests in a three step process. First, the system determined if there was a request being serviced. If the system was free, it checked to see if there was a request. Requests were serviced on a first come, first served basis, with no attempt being made to restore or save requests that might be lost if a system was busy. When there was a request, the system determined the type. The system was then flagged as busy for the duration specified for that type of request. The probability and duration for each type of request is shown in Table 3.

In order to provide a frame of reference for the results gathered during testing, a base system was constructed using the basic reputation aggregation equations and principles developed by Buchegger [13] and through the SECURE project [4]. The base system utilized an exponential weighted moving average equation for reputation scaling. It had fixed trust thresholds and exchanged reputation index values during a modified introduction process.

In addition to the work of the previously mentioned authors, the base system was equipped with a trust store-like reputation storage to enable the system to weight behavior grades upon receipt. During all tests, the same underlying interactivity traces and behavior models were applied during the creation of the test scripts. Although the simplicity of the base system appeared beneficial at first glance, testing

revealed serious deficiencies in its performance. The most notable deficiencies were found in the investigation of the success metric.

#### 4.5 Success Metric

The TMS was an access control system, so its efficiency was determined by examining how often the system correctly allowed access. The cost of making the decisions, in terms of communications and storage overhead, was also included in the determination. While acknowledging that the success metric of an access control system was comparative (i.e., one system performs better than another given a set of circumstances), we also experimented with critical settings to determine a feasible parameter range within which the system was effective.

In the most basic sense, the system was efficient when it correctly allowed access more often than it made incorrect decisions. Incorrect decisions came in two forms. False positive decisions occurred when a trustworthy user was incorrectly denied access. False negative decisions occurred when untrustworthy users were incorrectly allowed access [6].

We examined the ratio  $R$  of correct answers to false negative and false positive answers, shown in Equation 1.  $D$  was the total number of trustworthiness decisions the TMS was asked to make.  $P$  was the number of false positive answers and  $N$  was the number of false negative answers.

$$R = (D - (P + \omega N)) / D \quad (1)$$

We differentiated between false positives and false negatives and applied a weighting factor in recognition of the fact that the cost of a false positive was much less than the cost of a false negative. The cost weight ( $\omega$ ) was a value selected to represent this difference in cost and, in these experiments, was set to ( $\omega = 1$ ) to show the basic effectiveness of the TMS.

Having examined the efficiency of the TMS, we evaluated the overhead required by the system to render its decisions. The general intent of the overhead metric ( $C$ ) was to determine the cost of the level of efficiency. Two forms of overhead were included in the calculation of  $C$ .

Communications Overhead ( $C_C$ ) was defined as the number of Feedback Items ( $FI$ ) that needed to be sent between trusted peers to gain enough information to determine a trustworthiness decision on a specific peer. Equation 2 illustrates how the system divided the number of Introduction transactions ( $I$ ) by the size of the weighted queue of  $I$ , which is called the  $RIW$ . This computation assumed that the user would, in the worst case, attempt to fill their  $RIW$  before calculating a new associate's Reputation Index ( $RI$ ). This assumption is not as far-fetched as it may seem, especially if the number of reports was few.

$$C_C = I * |RIW| \quad (2)$$

Storage Overhead ( $C_S$ ) was defined as the number of  $FI$  each node stored to create a decision. Equation 3 determined  $C_S$  by multiplying the amount of memory designated for the TMS ( $TS$ ) by the amount of memory used to store reputations that are being actively calculated (e.g., the size of the  $RIW$ ).

$$C_s = |TS| * |RIW| \tag{3}$$

Adding the two costs together yielded the number of FIs maintained by the TMS over a period of time. Equation 4 used this result, divided by the number of correct access control decisions ( $D - (P+N)$ ), to provide the total cost for each correct decision.

$$C = (C_c + C_s)/(D - (P+N)) \tag{4}$$

When we executed the test scenarios, each scenario yielded independent values for  $R$  and  $C$ , as shown in the following charts. We called these values  $R(S)$  and  $C(S)$ , where  $S$  was the scenario number that was used. In analyzing  $R(S)$ , we wanted a value as high as possible. The opposite was true of  $C(S)$ , where we wanted the smallest number possible.

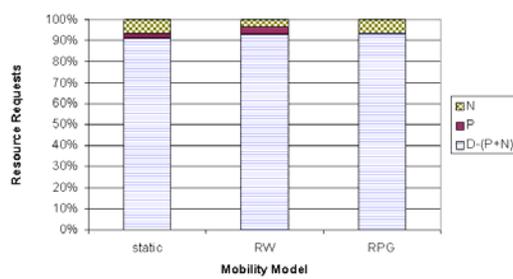


Fig. 6. Components of the Success Metric

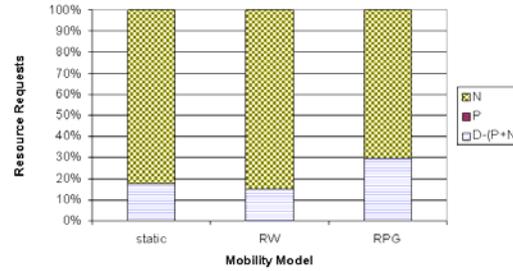


Fig. 7. Success Metric Components of the Base System Test

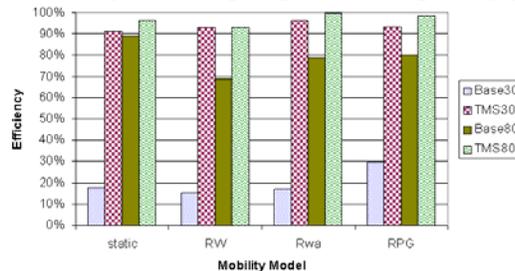
The a success metric, explained in Equation 1 expressed the number of correct decisions the system made as a ratio against the number of false positive ( $P$ ) and false negative ( $N$ ) decisions. Ideally, the column should be 100% correct (i.e.,  $P+N = 0$ ) to represent that the system answered all of the requests correctly. Barring this situation,

the goal was to minimize the number of false negative responses and then to eliminate the number of false positive responses.

Figure 6 shows the three components of the success metric. These tests, performed in a 100 node network with 30% misbehaving or non-contributing users, illustrated how well the TMS responded to resource requests in three mobility cases. The graph shows the proportional contribution of each response category to the overall success rate.

The TMS performed well in the static case, having 91% overall success in responses, but had moderate numbers of false positive and false negative responses. The overall success rate improved slightly in the RW case to 93% but the incidence of false positives almost doubled as a proportion of the incorrect responses. These false positive responses are of concern because they represent missed opportunities for information exchange and the possibility for a trustworthy peer to submit negative behavior reports on an otherwise “good” user. The RPG case was the most worrisome. Although the overall success rate increased to 94% and there were no false positive reports, the proportion of false negative reports doubled once again to represent 6% of the total number of requests. This testing illustrated the importance of examining the contributing components of the metric in addition to examining the overall percentage of correct responses.

The ratios presented in the previous tests are put into a better frame of reference when the TMS results are compared against those of the base trust system. Figure 7 shows how the base system performed. In addition to having a lower overall success percentage, the base system exhibited an extraordinarily high percentage of false negative responses. This high proportion was due to the lack of historical knowledge maintained by the TMS for dynamic weighting of behavior grades [2].



**Fig. 8.** Comparison of Base and Trust Management System Success Rates

The comparison between the TMS and the base system clearly showed the benefits of the 3Win method and the impact of dynamic grade weighting [2]. Figure 8 shows the comparison of success of the TMS and the base system in different mobility models. Tests using a general peer behavior condition of 30% misbehaving users, for example, are entitled TMS30 and Base30, respectively. While it had been expected that the base model performed would show less efficiency than the TMS, the poor success percentage in the static and RW models was surprising considering the general ratio of good users to bad was rather high. While the base system efficiency

increased slightly in the RW models with attraction points (RWa) and group mobility (RPG), it never demonstrated better than 30% efficiency.

As the proportion of bad users increased, the TMS efficiency remained at or over 90%. The base system reached its highest performance level when there were 80% bad users (see Figure 8, TMS80 and Base80, respectively). This case simulated a situation where the TBAC system was effectively presented with fewer trustworthy associates to select from.

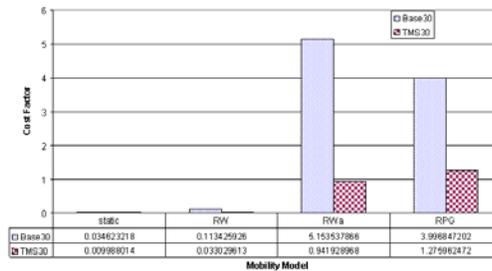


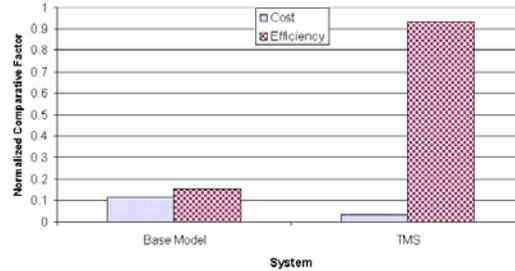
Fig. 9. Comparison of Cost Factors between TMS and Base System

#### 4.6 Cost Metric

Using Equation 4, the communications and storage costs of each system were combined to convey a sense of the behind the scenes requirements for making trust-based access control decisions. The TMS incurred a fixed cost of having to store and exchange all of the *FI* in its *RIW* while the base system only maintained an *RI*, so the general expectation was that the TMS would be at a disadvantage in this comparison.

What tests determined, however, was that the TMS cost was far lower than the cost incurred by the base system under the same test conditions, as shown in Figure 9. This phenomenon occurred because, while the amount of data exchanged by the base case during introductions was much smaller than used by the TMS, the number of introductions was an order of magnitude higher. In most cases, the difference between the base system and the TMS was 3:1 but, under RW mobility tests, the difference grew to four or five to one bias against the base system.

When success and cost were combined and displayed, the overwhelming efficiency of the TMS was reinforced. The TMS costs were several times less than those of the base system, while providing much higher efficiency. As discussed above, the combination of maintaining historical behavior grades, dynamic weighting of feedback items at every reputation calculation, and adjusting trust thresholds based on current levels of uncertainty have resulted in a much more robust trust system.



**Fig. 10.** Comparison of TMS Success Rate to Base Case System

Figure 10 illustrates this point using the RW mobility model and a network of 30% misbehaving users. The TMS displays low cost and high efficiency while the base system provided less success and more cost. Although changes to the base system might compensate for some of the deficiencies, the use of a memory-less computation method like the exponential weighted moving average puts and keeps the base system at a disadvantage. Furthermore, the implementation of adjustable thresholds and dynamic weighting in the TMS make it more flexible and able to adapt to a wider range of network conditions.

## 5. Conclusion and Future Work

Validation ensured that the system was ready for the intended operational environment. Using narrative techniques, we derived realistic requirements and assessed the TMS's efficiency and cost in meeting the demands of a TBAC system. The key to appreciating the impact of these results was that the findings would have less meaning were they not framed within a realistic operational scenario. While both TMS and the base system were verified to produce expected results, validation testing demonstrated that the TMS outperformed the base system in the expected operational setting. This conclusion could not have been determined without establishing the tasks to be accomplished, the conditions under which the task would be called for, and the standard to which the task would be accomplished successfully.

Throughout its development, the TMS was applied to inter-personal access control situations in mobile, often ad-hoc, networks. Currently, the TMS is being investigated for use as an inter-organizational access control mechanism. In this new incarnation, security policies and best business practices are applied to generate verifiable behavior observations. Studies are ongoing to create a framework for evaluating observed practices.

## References

- [1] Lo Presti, S., M. Butler, et al.: A Trust Analysis Methodology for Pervasive Computing Systems. *Trusting Agents for trusting Electronic Societies*. R. Falcone, S. Barber, J. Sabater and M. Singh, Springer (2005) 129 - 143
- [2] W. J. Adams. *Decentralized Trust-Based Access Control for Dynamic Collaborative Environments*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA (2006)
- [3] Freudenthal, E., T. Pesin, et al.: dRBAC: distributed role-based access control for dynamic coalition environments. *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002)*. Vienna, AU, 2 - 5 July 2002, (2002) 411-420.
- [4] Cahill, V., Shand, B., Gray, E., Bryce, C., Dimmock, N.: Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing 2* (2003) 52—61
- [5] Kagal, L., T. Finin, et al.: A framework for distributed trust management. *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-01)*, Workshop on Autonomy, Delegation and Control, 2001, Seattle, WA (2001) 73-80.
- [6] Bryce, C., N. Dimmock, et al.: Towards an Evaluation Methodology for Computational Trust Systems. *Proceedings of the Third International Conference in Trust Management (iTrust 2005)*, Paris, FR (2005) 289-304.
- [7] FEMA: Scenario and Incident Action Plan Catalog. Retrieved from <http://www.nwecg.gov/pms/forms/compan/iap.pdf> (1994)
- [8] FEMA: Incident Command System. Retrieved from <http://training.fema.gov/EMIWeb/IS/is195.asp> (2004)
- [9] de Waal, C. and M. Gerharz: BonnMotion. Retrieved from <http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/> (2005)
- [10] Camp, T., J. Boleng, et al.: A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications 2(5)* (2002) 483 - 502.
- [11] Ost, A.: *Performance of communication systems: a model based evaluation with matrix geometric methods*. New York, Springer (2001)
- [12] Heidemann, J., K. Obraczka, et al.: Modeling the performance of HTTP over several transport protocols. *Networking, IEEE/ACM Transactions on 5(5)* (1997) 616-630.
- [13] Buchegger, S. and J.-Y. Le Boudec: A Robust Reputation System for Mobile Ad-Hoc Networks. Lausanne, Switzerland, Ecole Polytechnic Federal de Lausanne (2003)