ISSN : 0974 - 7435 Volume 8 Issue 2

BioTechnology
An Indian Journal

FULL PAPER

BTAIJ, 8(2), 2013 [215-218]

Study of designing the trust management system

Cao Yonghui
School of Economics & Management, Henan Institute of Science and Technology, (CHINA)
School of Management, Zhejiang University, (CHINA)

ABSTRACT

In this paper, two additional components were required in order to test the TMS. The first component was a behavior model that would provide the simulated reports and observations on the performance of DCE members. The second component was a set of policies that the TMS used to protect itself from abuses commonly used against reputation-based systems. The combination of these two items allowed us to examine the response of the TMS to a variety of simulated network and behavior conditions.

KEYWORDS

Trust management system;
Behavior modeling;
System metrics.

INTRODUCTION

Designing the TMS (Trust Management System) was only the first step on the road to establishing the TMS as a viable access control mechanism. Two additional components were required in order to test the TMS. The first component was a behavior model that would provide the simulated reports and observations on the performance of DCE members. The second component was a set of policies that the TMS used to protect itself from abuses commonly used against reputation-based systems. The combination of these two items allowed us to examine the response of the TMS to a variety of simulated network and behavior conditions.

BEHAVIOR MODELING

The first step to simulating the TMS was to create a model of how network members behaved. The model needed to first determine what the member's actual behavior was. Members were divided into three broad behavior types: Good, Selfish, and Bad. Because the system operated on the observation and reporting of behavior, the simulation then had to determine how that behavior would be reported based on the observer's behavior type.

Given that people's basic inclination was to exhibit stable behavioral patterns, a finite state machine was constructed to model a member's behavior. In our model we emphasized the states rather than the transitions or stimuli, as our interest lay in the behavior exhibited during the state rather than in the significance of the state transition. Users started their behavior model in an initial or "natural" state. Behavior was based on the state that a member occupied at the time of making the observation. This model, shown in Figure 1, emphasized the probability that a member would change their behavior. Stable behavior types, such as the Good or Bad type, would start in their natural state demonstrating their dominant behavior. Selfish behavior types would start in a randomly selected state with positive behavior. The probability that a member would change states is given

FULL PAPER

in TABLE 1.

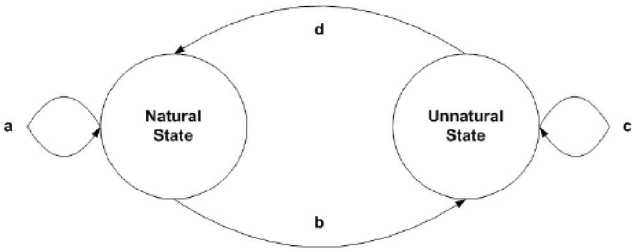

Figure 1 : Finite state machine behavior model

The state change probabilities (b and c) were selected to provide variance in individual behavior without creating random, irrational activity[1]. By moving the state change probabilities, we predicted the TSM's actions. The system approached a trivial state of complete predictability as these probabilities approached 0. Setting the probabilities closer to 50% to encourage state change reflected random activity that did not model human behavior. The settings in TABLE 1 were selected to provide the network with a stable population of members that demonstrated willingness to join a collaborative enterprise.

TABLE 1 : Behavior Model State Change Probabilities

Behavior Type	Initial State	a	b	c	d
Good	Good	90	10	10	90
Selfish	Random	70	30	30	70
Bad	Bad	90	10	10	90

In this case, the term stable implied that there was a low standard deviation in reputation indices. Users whose reputation index displayed a high standard deviation were displaying unrealistic behavior. The goal was for "good" users to show positive behavior the majority of the time. Figure 2 shows the impact of changing the settings on the average reputation index value of good members. The left side of the graph demonstrates that a stable, good user had an RI suitable for extending trust. As the parameters were changed to introduce more erratic behavior, good users changed states too frequently to demonstrate sustained positive behavior and thus earn trust.

The next part of the behavior script used the same behavior model to predict how the observed behavior would be reported. TABLE 2 shows how the cross mapping of behavior types to current behavior states yielded observations. In most cases, the observer reported the behavior truthfully but, in the cases of a Bad user, there was the chance that the report would be misleading. During testing, an element of collusion was added by mandating that the Bad user always report positive behavior for its confederates but followed the TABLE for all other observations.

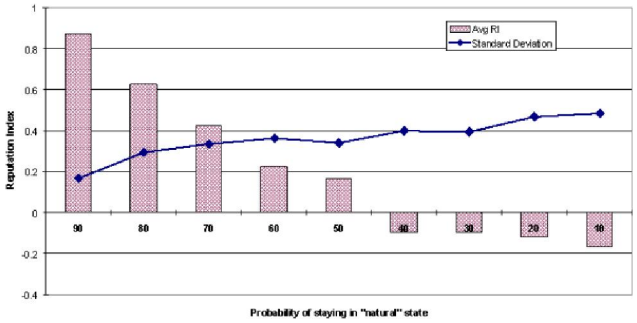

Figure 2 : Analysis of behavior model settings

TABLE 2 : Behavior Reporting Based on Behavior Type and State

If a	Is in its	It will report
Good or Selfish	Natural state	The behavior as observed
	Unnatural	The opposite of what was
Bad user	Natural state	The opposite of what was
	Unnatural	The behavior as observed

GAMING THE SYSTEM

The second step in simulating the TMS was to construct a set of operational procedures that would allow the system to eliminate or diminish the impact of common abuses of a trust-based system. A commonly perceived flaw in reputation systems was the ability for a user to "game" or manipulate their reputation to gain an unfair advantage. Gaming reputations undermined trust in the overall system. Users that knew their reputation often went to great lengths to maintain a high reputation, for both legal and illegal purposes. In KARMA[2], nodal entities successfully gamed the system to raise their overall KARMA rating to achieve greater access levels on a virtual bulleting board. EBay users inflated their client/seller ratings, which attempted to demonstrate accumulated trustworthiness based on transaction success, through a number of schemes like stacking[3].

This research used a Soft Security concept termed *Dissuade Reputation*. This reputation system sought

to isolate rather than exile misbehaving community members. Dissuade Reputation was used extensively in online communities such as Slashdot[4] and Kuro5hin[5]. As with our TMS, "it's possible to get an elevated K5 status (a term for reputation) but it takes continued effort to keep it."

In an article on online reputation, Derek Powazek[6] maintained that users only cared about their karma score if they knew what it was. This assertion was also stated by Ben Salem[7], who pointed out the difficulty in manipulating an unknown quantity. With any score or scoring system, there was the danger that community members would compete to get the highest score, manipulating what should have been a control into a game. If the game (i.e., gaining karma) was too easy or predictable, the control aspect of karma lost its meaning. This phenomenon was evident on Slashdot, with its use of karma points to gauge user activity and site privileges. As Slashdot users raced to increase their karma, the administrators made the decision to cap the amount of karma points any individual could accumulate. Rather than dissuade gaming, capping karma created a new type of gamer called a "karma whore." These individuals strove to reach the karma cap as quickly as possible, then dissipating their karma through malicious or disruptive behavior before starting all over again.

Another popular online community is Kuro5hin. This site concealed the user's mojo score (its name for reputation). Users gained a perception of their standing only by observing their privilege levels. Concealing reputation eliminated the gaming aspect but users may become frustrated and disruptive if the system prevents them from accessing resources.

Because an individual's reputation varied amongst his peers, direct "one on one" reputation gaming as is exhibited on eBay or Amazon.com was impossible in our TMS. In our TMS, every user (e.g., Alice) maintained an individual reputation index for each associated peer (e.g., Bob). This reputation index ($RI_A(B)$) was never shared as a single value. Instead, when Alice was asked to recommend Bob, she provided evidence in the form of non-reputable behavior grades. The requester was then left to calculate their own reputation index for Bob. Bob never knew what Alice thought of him, only that she granted or denied him access to her resources.

SYSTEM METRICS

The TMS was an access control system. We determined that we could assess its efficiency 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 calculation. 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[8].

We examined the ratio R of correct answers to false negative and false positive answers, shown in Equation 1a. 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.

$$R = (D - (P + N)) / D \quad \textbf{(1a)}$$

When tabulating access control decisions, we differentiated between false positives and false negatives, as Bryce did. We did this differentiation 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 (ω) was a value selected to represent this difference in cost and added to the previous equation. The result is shown in Equation 1b.

$$R = (D - (P + \omega N)) / D \quad \textbf{(1b)}$$

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 FI that needed to be sent between trusted peers to gain enough information to determine a trust-

FULL PAPER

worthiness decision on a specific peer. Equation 2a illustrates how the system divided the number of Introduction transactions (*I*) by the RIW size. This computation assumed that the user would, in the worst case, attempt to fill their RIW before calculating a new associate's RI. This assumption is not as far-fetched as it may seem, especially if the number of reports was few.

$$C_C = I * |RIW| \quad \textbf{(2a)}$$

Storage Overhead (C_S) was defined as the number of FI each node stored to create a decision. Equation 2b determined C_S by multiplying the TS size (expressed in the number of stored ABRs) by the RIW size.

$$C_S = |TS| * |RIW| \quad \textbf{(2b)}$$

Adding the two costs together yielded the number of FIs maintained by the TMS over a period of time. Equation 2c used this result, divided by the number of correct access control decisions ($D - (P+N)$), to provide the total cost for each correct decision.

$$C = (C_C + C_S) / (D - (P + N)) \quad \textbf{(2c)}$$

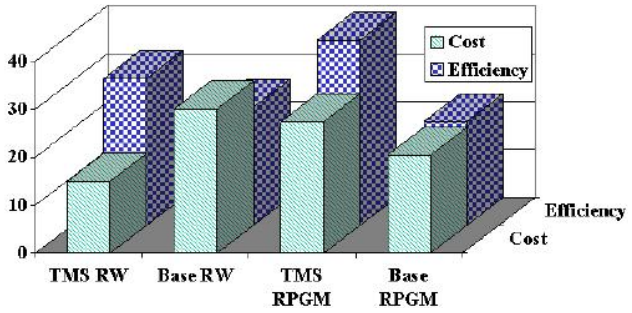

Figure 3 : Example Comparisons of Scenario Efficiencies versus Costs

When we developed the test scenarios, each scenario had independent values for *R* and *C*, as shown in the example in Figure 3. We called these values R_{Sx} and C_{Sx}, where *x* was the scenario number. In analyzing R_{Sx}, we wanted a value as high as possible. The opposite was true of C_{Sx}, where we wanted the smallest number possible. Because the number of different scenarios was bounded only by the limits of imagination, we used Lo Presti's format[9] for devising and organizing test scenarios.

REFERENCES

[1] J.Laird, R.Wray; Variability in Human Behavior Modeling for Military Simulations. Proceedings of the 2003 Conference on Behavior Representation in Modeling and Simulation (BRIMS), Scottsdale, AZ, 1-10 **(2003)**.

[2] R.Krishnan, M.Smith et al.; Economics of Peer-to-Peer Networks. Journal of Information Technology Theory and Application, **5(3)**, 31-44 **(2003)**.

[3] C.Keser; Experimental games for the design of reputation management systems. IBM Systems Journal, **42(3)**, 498-506 **(2003)**.

[4] Commander Taco; Slashdot. Retrieved 15 December, 2005, from http://slashdot.or, **(2005)**.

[5] Kuro Shin; KuroShin. Retrieved 15 December, 2005, from http://www.kuro5hin.org, **(2005)**.

[6] D.Powazek; Gaming the system: How moderation tools can backfire. Retrieved 15 December, 2005, from http://designforcommunity.com/essay8.html, **(2003)**.

[7] N.Ben Salem, J.P.Hubaux et al.; Reputation-based Wi-Fi deployment protocols and security analysis. Proceedings of the 2nd ACM International Workshop on Wireless Mobile Applications, Philadelphia, PA, 29-40 **(2004)**.

[8] C.Bryce, 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, 289-304 **(2005)**.

[9] S.Lo Presti, M.Butler et al.; A Trust Analysis Methodology for Pervasive Computing Systems. Trusting Agents for trusting Electronic Societies. R.Falcone, S.Barber, J.Sabater, M.Singly Springer, **(2005)**.