ISSN : 0974 - 7435 Volume 8 Issue 4

BioTechnology
An Indian Journal

FULL PAPER

BTAIJ, 8(4), 2013 [454-458]

Study of verification of the trust score module

C.A.O.Yonghui
School of Economics & Management, Henan Institute of Science and Technology, (CHINA)
School of Management, Zhejiang University, (CHINA)

ABSTRACT

The TS represented a number of contributions to the state of the art in trust and access control systems. The presence of the TS enabled the TMS to maintain and update the behavior history of a user's associates. Additionally, linking the concepts of network density to memory size optimized the storage cost of holding trust credentials when compared to un-managed or memory-less systems. The analysis presented in this paper showed that the TMS minimized the number of credential exchanges it had to perform because the contents of the TS were actively managed.

KEYWORDS

Trust Store;
Memory store management;
Environmental impacts.

VERIFICATION: GOALS AND OBJECTIVES

Verification determined that each module had correctly transformed its inputs into the expected output. This testing involved isolating each function of each module by the arrangement of input and processing parameters. Specific outputs were then analyzed to check their correspondence to expected results. In general, verification testing revealed that the modules worked in accordance with the requirements.

Once basic verification was complete, performance boundary analysis was conducted to ascertain under which conditions the module operated best and under which conditions performance was impaired. Once these expectations were met, the modules were combined and the system validated. The following sub-sections provide the analysis of verification testing. These sub-sections follow a standard methodology (Bryce, Dimmock et al. 2005) of:

- Defining the role of each component.
- Analyzing the component to determine how module failure or impaired performance influences overall system functioning.

TRUST STORE

The implementation of a system memory represented a step forward in state of the art access control because the TS was actively managed to update trust records on a continuous basis. Current systems fell into one of two categories: those with passive storage and those with session-based storage. Those with passive storage collected credentials for associates, usually in the form of authorization certificates. When the storage space was full, none of these systems had defined management schemes to eliminate unneeded credentials and make space for new certificates. Systems with session based storage kept values, like reputation and threshold values, in program memory. The trust or access control system was vulnerable while it learned about its associates. This vulnerability presented itself each and every time the system started and was considered grave enough to accept the storage and processing costs of implementing the TS.

The TS provided a finite memory for the TMS. The TS held the behavior history of peers and provided the system the ability to remember and incorporate the previous actions of associates into its access control decisions. The TS stored the records of associates' behavior in the form of Atomic Behavior Records (ABRs) and managed the memory to limit the number of reintroductions the TMS performs. The TS also provided the store from which a node made recommendations and referrals to its trusted peers.

Without the TS, the system would be forced to work on a by-session basis, remembering associates only for the term of the current session. If the TS was too small, a user could be overwhelmed performing introductions and be kept from performing actual information sharing. If the TS size was too large, a user spent too much time searching the store for associates that might or might not be present.

A properly sized TS management algorithm:

- Should minimize the number of "re-introductions" that a node required.
- Should optimize storage of the trust and identity information to allow a node to remember as many peers as possible.

Testing was conducted in network environments that contained 100% "good" or behaving users. By removing the possibility of encountering undesirable associates, we presented a testing scenario where a user would want to interact with every node it came into contact with. The following sections examine the effects of changing various parameters on determining the optimum amount of memory allocated to a TS.

GENERAL TESTING

General testing was performed to establish the sizing of the TS under different network population characteristics. We called the relationship between population and transmission range network density, as it characterized the number of peers within a node's transmission range.

The TS should be sized to prevent the need for reintroductions. Reintroductions represented work that had already been accomplished but had been deleted and needed to be performed again. At the same time, the TS needed to be small enough to prevent saving lots of people that we no longer interacted with. We wanted to link the size of the TS to the network density, rather than a "one size fits all" approach, so that we were optimizing for all densities rather than picking a size that suited for the densest networks and assuming the cost of excess memory usage when operating in a sparse network. Finally, we wanted a memory management technique to allow us to forget older and presumably unused associates' information.

Memory store management

The first set of tests sought to determine the memory management algorithm that minimized the number of introductions. The tests compared the number of introductions in static and random waypoint mobility (Camp, Boleng et al. 2002) situations, varying the size of the TS in successive runs. Two algorithms, First In First Out (FIFO) and Least Recently Used (LRU) were compared. These specific algorithms were used because they did not require additional timing information, used by algorithms that were based on usage frequency. This timing information was not available to the TMS, as all frequency information was lost when an associate was "forgotten."

The FIFO algorithm placed an associate's ABR at the head of the TS queue. As newer associates were encountered, older ABRs were pushed toward the end of the queue. When an ABR was pushed from the queue, that associate was "forgotten."

The LRU algorithm required more queue manipulation than FIFO. While new ABRs were added at the head of the queue, as in the FIFO scenario, LRU moved ABRs to the head of the queue each time an ABR was accessed. Other ABRs were pushed down to fill the space. ABRs that were pushed out of the queue were forgotten.

Figure 1 shows the performance of the FIFO algorithm against the LRU algorithm in a mobile network of twenty six and fifty two users. Tests used different memory management algorithms on the same simulation script to determine the number of introductions a

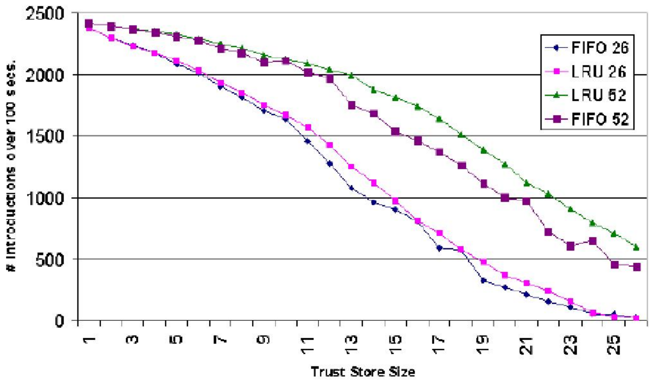

Figure 1 : Trust Store Management Algorithm Comparison

user needed to perform given a specific TS size. The goal was to determine which algorithm was most efficient at maintaining the ABRs of the associates the user needed to speak with.

In both cases, the FIFO algorithm required fewer introductions than the LRU implementation over the period of the simulation. As the TS size increased, the FIFO curve decreased faster than the LRU curve. Detailed analysis of the TS contents determined that FIFO worked better than LRU because of the node's mobility in the simulation. LRU focused on a node's current associates and forgot old associates right before the mobility model brought them back into range, thus requiring the node to reintroduce itself. Based on this testing, subsequent tests used only the FIFO memory management algorithm in the TS.

Trust store size determination

Efficiency, in the case of the Trust Store management problem, was defined as limiting the number of "re-introductions" that a node required. Re-introductions were cases where a node forgot a peer that it once had security associations with and had to go through the entire introduction process, as if the two nodes had never dealt with each other before. Through testing, we wanted to find the smallest trust store that minimized the number of introductions the system had to perform.

Sizing tests performed using a random waypoint mobility simulation showed that the optimum size for a trust store was approximately 30% of the total number of users of the network (N), as shown in Figure 2. This optimum was determined by comparing the increasing amount of memory required for the ABRs against the number of introductions required in a network of a certain density. Sparse networks (e.g., those with less than 100 members per square kilometer) required a user to store a higher percentage of associates' credentials due to mobility. Once a certain density was reached (e.g., 100 mobile users in a 1000 x 1000 meter area), however, the tradeoff point stayed near the 30% mark. This meant that, if a user could know how many nodes were in the network, he could size their system memory to minimize reintroductions and maintain only those credentials that he needed in the near future.

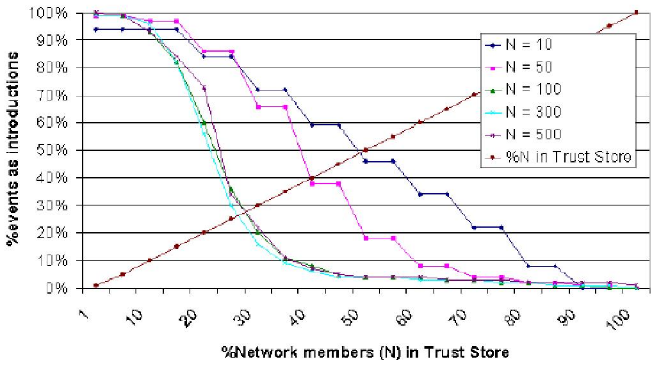

Figure 2 : Trust Store Sizing Test

Since the ad-hoc nature of a DCE prevented anyone from knowing the number of users in the network at any time, we developed the concept of network density sampling, shown in Equation 1.

$$D = (N\pi r^2) / A \qquad (1)$$

To use Equation 1, a node entered a network and listened for the beacons of its neighbors. In a hybrid environment (i.e., one having wired and wireless users), the node might sample the headers of IP packets. The user knew its *r* (the transmission radius) and assumed *A* (the area of the mobility area) to be a standard one square kilometer. As it listened, the node counted *N* (the number of nodes it could hear) and calculated *D*, the network density.

From this equation we created three representative network densities: sparse, medium and dense. In Figure 2, the sparse networks had ten members in a square kilometer, resulting in a *D* of 1. With this density, the node had to maintain at least 50% of the network members (i.e., five ABRs) in its memory to achieve the desired optimization of reintroductions. Similar tests showed that medium networks (50 members) had a *D* of 5 and needed to maintain 40% of the network members in the TS. Dense networks had a *D* of 10. Once Equation 9 indicated the presence of a dense network (i.e., a network with more than 100 users in a square kilometer), the user set the TS size to 30% of the number of members it sampled upon entering the network. During operation or at any point the node detected that it had to reintroduce itself more than indicated by the density factor, it re-sampled the nodes and reset the TS size.

ENVIRONMENTAL IMPACTS

While it might seem feasible to accept the cost of a large TS, tests in a 100 user static network showed that larger stores had less efficiency than smaller, more optimally sized ones. Figure 3 shows how increasing the store size for a user with a 100 meter transmission range lowers the cost significantly but actually decreases the efficiency from 99% when the |TS|=10 to 82% when the |TS|=100. Since the actual cost of introductions in a static environment was minimal due to the increased storage and communications capacity of the nodes, there was little point in decreasing costs at the expense of losing almost 18% efficiency.

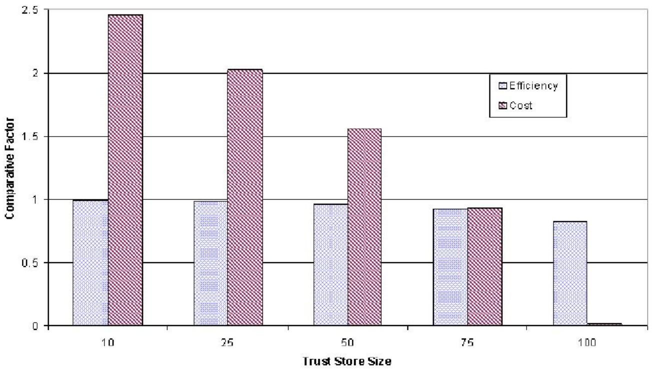

Figure 3 : Effects of Trust Store Size on Efficiency and Cost

The decrease in efficiency seen with larger TS sizes was traced to the effects of the introduction process. While trying to minimize the number of introductions performed, testing proved that it was not beneficial to do away with them completely. The introduction process had the effect of flushing out the ABRs of old associates, in effect restoring them with more current information and improving the fidelity of the reputation and, therefore, the efficiency calculations it was required to perform.

The performance of the TS was directly proportional to the number of introductions. Network density and mobility affected the number of introductions as a result of increased interactivity. Since, as discussed above, changing the TS size had a large impact on the cost of interactivity, sizing the store to decrease costs had to be coupled with the acceptance of decrease in efficiency.

CONTRIBUTIONS AND CONCLUSION

The TS represented a number of contributions to the state of the art in trust and access control systems. The presence of the TS enabled the TMS to maintain and update the behavior history of a user's associates. Additionally, linking the concepts of network density to memory size optimized the storage cost of holding trust credentials when compared to un-managed or memory-less systems. The analysis presented in this section showed that the TMS minimized the number of credential exchanges it had to perform because the contents of the TS were actively managed. This performance increase was reflected in a lower communications cost incurred while operating the TMS.

While the TS provided a number of benefits as implemented, there remained room for improvement in certain situations. An unfortunate aspect of more mobile networks was the situation where a user forgot important associates or resource providers, since older ABRs were forced out of the TS by new acquaintances. This deficiency might be addressed through the use of a "passive" store that was user-selected and identity-based. The passive store would not be subject to the same memory management algorithm and would allow a peer to remember his supervisor and/or a set of resource providers without having to be reintroduced to them on a regular basis. The danger with the implementation of the passive store would be the dated information contained in these ABRs. The suggestion was made that ABRs stored in the passive store would contain only first hand and Key Management System (KMS) observations to minimize this deficiency.

The TS existed to provide storage for the behavior grades of network peers that a user wanted to associate with. Information, in the form of ABRs, was provided upon request to the reputation scaling process whenever the system needed to calculate an associate's trustworthiness.

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] KuroShin; 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)**.