Stevens, M., Williams, P.D., Peterson, G.L., and Kurkowski, S.H., "Integrating Trust into the CyberCraft Initiative via the Trust Vectors Model," *IEEE Computational Intelligence Magazine*, vol. 3, no. 2, pp. 65-68, 2008. DOI: 10.1109/MCI.2008.4490264.

Integrating Trust into the CyberCraft Initiative via the Trust Vectors Model

Michael Stevens, Paul D. Williams, Gilbert L. Peterson, and Stuart H. Kurkowski

I. Introduction

Computer Network Defense (CND) is vital for the protection of information related to our national security. Operating systems are becoming more complex, higher bandwidth allows more volume and faster network attacks, and the number of networked devices is increasing exponentially. The need for an intelligent, automated mechanism to perform CND grows with the threats and the vulnerabilities. The CyberCraft Initiative is designing a command and control (C2) system for a fleet of agents to autonomously operate and defend the Air Force networks faster than humans can.[1] Analogous to an aircraft flying in cyberspace, the CyberCraft agent can load different software payloads for various missions such as Insider Threat Detection, Policy Enforcement, and Virus Detection and Remediation.

CyberCraft are composed of two components: Agents and Payloads. Agents are hardware or software constructs with limited functionality, essentially a three-way interface between the payloads, the host Operating System (OS), and the C2 structure. Payloads are *Sensors* that sample the environment, *Decision Engines* that decide what action to take, or *Effectors* which alter the environment. Currently these payloads are in the development stage, but will incorporate trust to provide confidence to the warfighter that the data is accurate, the decisions are correct, and actions predictably change the environment. The fundamental research question of CyberCraft is "What is required for a commander to trust a CyberCraft to autonomously defend military information systems?"

Trust will mainly be used by the decision engines to evaluate the data gathered by the sensors and determine which effectors to use to best change the cyber-environment to a desired state. The decision engines will need to use trust metrics to determine the actual state of the environment when being fed conflicting data by the sensors, mitigating spoofing attacks and other electronic subterfuge. Trust will also be used in a smaller degree by the sensors and effectors for self-diagnosis.

Trust is vital to the CyberCraft Initiative, and this research examines if the Trust Vector model is suitable to integrate trust into the CyberCraft fleet. We use the definition of trust from the Trust Vector model: *The firm belief in the competence of an entity to act dependably, reliably and securely within a specific context*. Integrating trust into the CyberCraft Initiative enables the commander to have a measure of confidence that a Cyber-operation executes as expected.

We hypothesize the Trust Vector model is an acceptable model for the distributed environment of the Cyber- Craft fleet, and that there are limits to the utility of historical data (i.e. data about previous trust and events that are stored to calculate trust in the future). This research determines whether or not this hypothesis is true by implementing the Trust Vector model into a small fleet of test agents.

This research finds that a modification of the Trust Vector model can be used to provide a metric of trust in a CyberCraft agent's actions and that there are limits to the utility of historical data. This article is derived from the Masters thesis of the first author [2] and from [3].

The rest of the paper is laid out as follows: Section 2 covers the Trust Vector model, modifications needed to fit the Trust Vector model to the Cyber-Craft fleet, and other work on trust in distributed systems. Section 3 discusses the two experiments that address the value of historical data to the Trust Vector. Section 4 contains conclusions drawn from the analysis and experiments and identifies other problem areas and optimization issues to be addressed by future research.

II. Background

This section focuses on the background of the Trust Vector model and other work in trust between distributed systems.

A. Trust Vector Model

The Trust Vector model [4] defines a unidirectional trust relationship between and agent and a remote entity as a vector with three components, where the value of the relationship is a real number that spans from complete trust (represented as +1) through neutral trust (0) to complete distrust (−1). Distrust differs from no trust in that distrust indicates a level of confidence that the information is incorrect rather than uncertainty about the veracity of the information. The model also incorporates a Trust Policy vector for assigning weights

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This research is sponsored by the Cyber Operations branch of AFRL at Rome Labs, NY, who also sponsor the CyberCraft Initiative.

M. Smith, P.D. Williams, G.L. Peterson, and S.H. Kurkowski are with the Air Force Institute of Technology, AFIT/ENG, 2950 Hobson Way, Wright-Patterson AFB, OH 45433-7765. (e-mail: {michael.smith, paul.williams, gilbert.peterson, stuart.kurkowski}@afit.edu)

to each component and a degradation function for degrading the value of trust over time. Figure 1 displays a graphical representation of the Trust Vector.

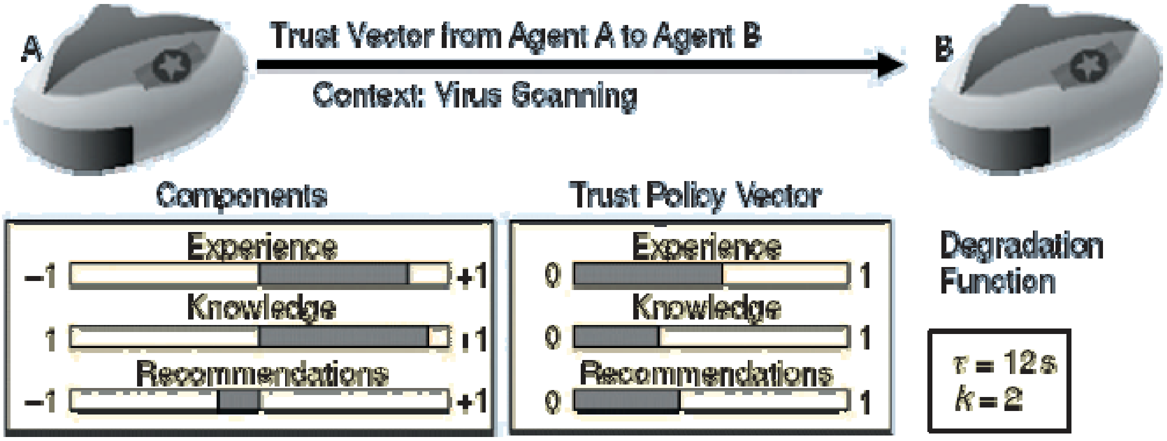

Fig. 1: Diagram of a trust vector.

Experience Component: The experience component of the original Trust Vector work [4] is based on the past performance of the remote agent in the given context. Each event is given a value of trust positive (+) or trust negative (−). This collection of events is then divided by time into intervals. The values of the events in each interval are summed to produce a single value for each interval, which are then summed to produce a single value.

We modify the experience component to use a range of values from −1 to +1 to describe a trust event, rather than just (−1) or (+1). This allows greater granularity to describe an event and is useful for calculating the experience component for the Trust Vector for Recommendations, as it is difficult to arbitrarily evaluate a recommendation as completely trustworthy (+1) or completely untrustworthy (−1).

Knowledge Component: The knowledge component represents an arbitrary value for the local agent’s knowledge about the remote agent’s abilities.

Recommendation Component: The recommendations component is updated through querying other agents about their trust with the remote agent. The recommendations received are then weighted by the local agent’s trust in the recommenders, summed, and normalized to give a value from −1 to +1.

Trust Policy Vector: The Trust Policy Vector is composed of the weights applied to each component of the trust vector to produce a single normalized value from −1 to +1. The value for each component ranges from 0 to 1, and the sum of all the weights is equal to 1.

Degradation Function: The Trust Vector model includes a degradation function to calculate the current value of trust based on a past value. If no new experience or recommendations are received, the trust value of a trust relationship asymptotically approaches 0 as time increases. The value (v) of the trust relationship (T) at time t_n is given by the equation

$$v(T_{t_n}) = v(T_{t_i})e^{-(v(T_{t_i})\Delta t)^{2k}},$$

where t_i is the time the trust value was calculated, Δt is the difference between t_i and t_n, and k is an integer ≥ 1.

To calculate the normalized trust relationship at the present time, the values of the previous trust vector and the present trust vector are weighted and combined to produce a single value for the trust at the current time.

B. Transactional Paradigm

The Trust Vector model does not explicitly state if it should be used in a *synchronous* or *asynchronous* paradigm. In a synchronous paradigm, multiple agents sample the same aspect of the environment, immediately exchange their results, and evaluate the results of the remote agents. This works well for the Trust Vector model, as each participating agent obtains a large amount of data from the other participating agents and a robust experience component can be built. When CyberCraft is implemented, it is unlikely that multiple agents can immediately sample the same aspect of the environment to evaluate the data produced by an agent, therefore an *asynchronous* or *transactional paradigm*, derived from Xiong and Liu’s paper [5], is more appropriate. Data produced by an agent is posted to an information store, and when another agent accesses that information, the second agent updates its trust in the producing agent and posts its new trust data to the store.

C. Other Work in Trust in Distributed Systems

Yahalom, Klein, and Beth’s Boolean trust model [6] models the passing of authentication data inside a network. This models the transitivity of trust that builds new trust relationships between entities, but distinguishes between directly calculated trust and trust passed from a recommender.

Beth, Borcherding, and Klein [7] expand on [6] by adding degrees of trust, based on the summation of positive experiences. Negative experiences cause the remote agent to become completely untrusted, where the Trust Vector model lowers the trust in the remote agent. Jøsang [8], postulates that trust in distributed systems should be based upon knowledge, which represents the information used to determine trustworthiness.

The Abdul-Rahman and Hailes’ trust model (A-R/H model) [9], [10] incorporates different contexts for trust. Trust is based on experiences and recommendations, and trust is not directly transitive (a recommendation is modified by the trust in the recommender). The A-R/H model also incorporates degrees of untrustworthiness, but trust values are discrete, rather than continuous.

Xiong and Liu’s PeerTrust [5] models trust in peer-to-peer (P2P) eCommerce transactions. PeerTrust provides a range of trust relative to a peer’s performance in transactions based on feedback received from the other peers involved, while including other factors (e.g., credibility of feedback and context). PeerTrust models trust in an asynchronous transactions, from which we derived the transactional paradigm covered in Section 2.2.

Ray, Chakraborty, and Ray [11] extended the Trust Vector model with *VTrust*, a trust management framework. *VTrust* stores trust relationships in a database that uses a modification of SQL (*TrustQL*) for queries. The $p - Trust$ model by Chakraborty and Ray [12] extends the Trust Vector model for the control of privacy in online transactions.

III. Experiments

Both experiments use a test network of five agents that exchange data and trust values. At a specific time, one agent begins to misinterpret the data and then interprets the data correctly again. We use the value of the trust vector between a correct agent and the misinterpreting agent as our metric for system performance.

A. *Experiment I: Limits of Historical Data for Calculating the Experience Component*

Experiment I tests the utility of the historical data in the calculation of the Experience Component. This is accomplished by altering the number of intervals kept (8, 9, 10, and 11 intervals kept) during different runs of the system using a deterministic set of input data, and comparing the results of each run.

Problem Statement: At what point does the weight of the interval diminish the value of the interval to the point that it is not worth storing the data?

Hypothesis: There is a point where the cost of storing the data outweighs the utility of historical data.

Analysis: To determine how much historical data must be stored, we first analyze how much the historical data contributes to the current trust value. Historical data is used to calculate the value of the experience component (which in turn is used to calculate the trust value of the trust vector), and in the current value of a previous trust vector.

TABLE I
DECREASING VALUE OF OLDEST INTERVAL.

Number of Intervals	8	9	10	11
Weight of Oldest Interval	2.8%	2.2%	1.8%	1.5%

The decision of how many intervals are to be kept depends on the trade-off between storage of each interval and the need for granularity of the Trust Values. The storage cost of each interval can be calculated by *Events per Interval* × *Bytes perevent* × *Number of contexts* × *Number of agents*

As stated in Section 2.1, the experience component is calculated by weighting the values of the intervals of the event history and summing the products of the weights and the values of the intervals. Table 1 shows the weight associated with the oldest interval kept for a given number of intervals.

Results: As the number of intervals kept decreases, the Trust Value changes faster and drops farther when discrepancies occur, as there is less historical data to counterbalance current data. The difference in trust values between 8 intervals kept and 9 intervals kept is 0.0357, greater than the difference between 9 intervals kept and 10 intervals kept (0.0321), which is greater than the difference between 10 intervals kept and 11 intervals kept (0.0287).

Our results support our hypothesis in that as data ages; the benefit provided by the older data is diminished to the point where the contribution to the current trust level is so small that keeping the data is not worth the storage cost. We do not attempt to identify a specific point to discard data, but provide a model which can provide recommendations based on the implementation of the Trust Vector model.

B. *Experiment II: Utility of Degradation Function*

Experiment II tests the utility of the historical data in the calculation of the current Trust Value. The current Trust Value is calculated by combining the time-dependant trust vector and the current trust vector (Section 2.1). The *period of decay* controls how long a time-dependant trust vector will retain its value. To determine a proper retention of data, we alter the *period of decay* to 1x, 1.5x, 2x, and 3x times the exchange rate using a deterministic set of input data.

Problem Statement: How long should the values of previous (time-dependant) trust vectors be stored before their utility is outweighed by their storage cost?

Hypothesis: The *period of decay* must be set to at least twice the expected time between trust calculations. If the period of decay is shorter than that, the degradation function decays the previously calculated trust value too rapidly to be useful.

Analysis: Using the Trust Vector model’s degradation function, higher initial values of trust degrade faster than lower initial values, which is counter intuitive. Inverting the $v(T_{t_i})$ term (trust value at time t_i) from the exponent of e changes the equation to $v(T_{t_n}) = v(T_{t_i})e^{-(v(T_{t_i})\Delta t)^{2k}}$, all trust values degrade at an equal rate.

Time periods are arbitrary, as the time difference between t_0

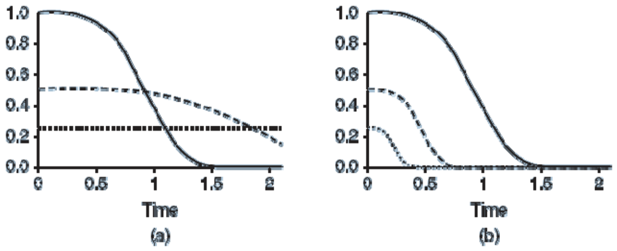

Fig. 2. (a) Higher values degrade faster using standard representation. (b) All values degrade at an equal rate using inverted v(*T*).

and t_1 (the *period of decay*) could be 10 seconds or 2 months. The *period of decay* should be set based on how often trust values are calculated, so that most calculations of the value of a trust relationship happen before the value of a previous trust vector has decayed to nearly zero.

Results: Our results shown in Figure 2, demonstrate that as the exchange rate approached the *period of decay*, even complete trust (value of +1) would degrade to the point that it was becoming irrelevant (+1 would degrade to +0.6 if the exchange rate and the *period of decay* were equal). We conclude that implementers of the Trust Vector model should set the *period of decay* to 1.5 × the exchange rate to avoid degrading the previously calculated trust value too rapidly.

IV. Conclusions and Future Work

This research supports the hypothesis that the Trust Vector model can be modified to fit the CyberCraft Initiative, and that there are limits to the utility of historical data. This research proposed some modifications and expansions to the Trust Model Vector, and identified areas for future research.

Contributions: This research identifies that the *transactional paradigm* models the environment of the CyberCraft fleet better than a *synchronous paradigm*. A modification to the degradation function is proposed, as well as defining the benefit of each event as a value between [−1, 1], rather than just as a trust-negative or trust-positive event. Experiment I identifies the degrading value of older intervals in the experience component. Experiment II identifies problems with the degradation function, both with the function itself and the need for defining a reasonable *period of decay*.

Future Work: There are a few modifications of the Trust Vector model that still need to be added. The largest of these additional modifications is determining the amount of data needed to populate a trust vector. In the transactional paradigm, an agent needs several transactions to build enough history to make accurate evaluations about the trustworthiness of a remote agent. Because the Trust Vector model does not rely on experience alone, the model is able to rely on the other two components to build a more robust picture of trust than by relying on experience alone. Also, the VTrust system [11] could be integrated as the Trust Management framework for the CyberCraft Initiative as another area for future work.

Scalability of the Trust Vector model to a CyberCraft fleet of over 1000 agents needs to be researched. We expect that with the transactional paradigm, the model will scale without storage issues or bandwidth depletion, as agents will be selective on building trust relationships with other agents.

Further expansions of the Trust Vector model include dynamic routing of network traffic, how to evaluate an event if no corroborating data is present, and the use of Trust Vectors with differing views that do not necessarily conflict.

References

[1] K. Jabbour and D. Bibighaus, “Cybercraft information workshop,” June 2006.

[2] M. Stevens, “Use of trust vectors in support of the cybercraft initiative,” *Master’s thesis, Air Force Institute of Technology*, 2007.

[3] M. Stevens and P. Williams, “Use of trust vectors for cybercraft and the limits of usable data history for trust vectors,” in *2007 IEEE Computational Intelligence for Security and Defense Applications (CISDA’07)*, Honolulu, HI, USA, 2007.

[4] I. Ray and S. Chakraborty, “A vector model of trust for developing trustworthy systems,” in *Proceedings of 9th European Symposium on Research in Computer Security (ESORICS’04)*, Sophia Antipolis, France, 2004.

[5] L. Xiong and L. Liu, “A reputation-based trust model for peer-to-peer ecommerce communities,” in *Proceedings of IEEE Conference on E-Commerce (CEC’03)*, pp. 275–284, 2003.

[6] R. Yahalom, B. Klein, and T. Beth, “Trust relationships in secure systems: A distributed authentication perspective,” in *Proceedings of the IEEE Computer Society Symposium on Security and Privacy, Oakland, California, USA, IEEE Computer Society*, pp. 150–164, 1993.

[7] T. Beth, M. Borcherding, and B. Klein, “Valuation of trust in open networks,” in *Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS’94)*, Brighton, U.K., pp. 3–18, 2004.

[8] A. Jøsang, “The right type of trust for distributed systems,” in *Proceedings of the 1996 workshop on New Security Paradigms*, 1996.

[9] A. Abdul-Rahman and S. Hailes, “A distributed trust model,” in *Proceedings of the 1997 workshop on New Security Paradigms*, pp. 48–60, 1997.

[10] A. Abdul-Rahman and S. Hailes, “Supporting trust in virtual communities,” in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS-33)*, Maui, Hawaii, USA, pp. 1769–1777, 2000.

[11] I. Ray, S. Chakraborty, and I. Ray, “Vtrust: A trust management system based on a vector model of trust,” in *Proceedings of 1st International Conference on Information Systems Security (ICISS’05)*, Sophia Antipolis, France, 2005.

[12] S. Chakraborty and I. Ray, “Allowing finer control over privacy using trust as a benchmark,” in *7th Annual IEEE Information Assurance Workshop (IAW’06), United States Military Academy*, West Point, NY, USA, 2006.