A Bio-Inspired Trust Framework in Wireless Ad Hoc Networks

Vikram Kanth
Naval Postgraduate School
vkkanth@nps.edu

John McEachen
Naval Postgraduate School
mceachen@nps.edu

Murali Tummala
Naval Postgraduate School
mtummala@nps.edu

Abstract

Cyber attacks are amongst the most serious threats facing people and organizations. In the face of the increasing complexity and effectiveness of these attacks, creative approaches to defense are required. Groups of insects survive due to their use of collaborative approaches with the unique ability to detect anomalies using primarily local data and very limited computational resources (i.e., limited brain power). These attributes are even more crucial for wireless ad hoc networks where the number of nodes and connections between those nodes are ephemeral. We propose a trust framework inspired by the detection mechanisms exhibited by bee swarms in which a wireless node can only observe and leverage the actions of their neighbors rather than the global knowledge of the network to make decisions. This leveraging of local knowledge is an important aspect of trust in wireless networks in which global state information is difficult to encapsulate. We also utilize models from binary voting to present a mathematical model for our bee-inspired trust framework in wireless ad hoc networks.

1. Intrusion Detection, Neighboring Nodes, and Trust

Cyber threats present some of the most pervasive challenges in today's security landscape. Reports from the Governmental Accountability Office (GAO) showed that federal government agencies reported 35277 cyber security incidents in 2017 and 31107 incidents in 2018 [1, 2]. According to the Identity Theft Resource Center, there were 1473 reported data breaches exposing over 164.68 million sensitive records in 2019 [3]. In the face of these stark statistics, it is clear that there is a need for more effective defensive tools.

One of the most commonly utilized defensive tools is the intrusion detection system (IDS) [4]. As the name implies, the purpose of these systems is to detect intrusions by identifying anomalous behavior within individual hosts or in a network. Broadly, there are two categories of IDSs, network-based and host-based. A network-based IDS uses network behavior such as traffic volume or diversity to detect abnormal network behavior. A host-based IDS uses local events like unusual program execution or log-in attempts to trigger an alert regarding a potential attack. These techniques have been successful in monitoring single machines and single networks. Unfortunately, the nature of cyber attacks is evolving and require more sophisticated defenses to detect them. Highly distributed attacks like distributed denial of service (DDoS) attacks can be used to camouflage virus and malware installation for nefarious purposes [5]. In order to counter these threats, cyber defenses have leveraged the idea of information sharing between hosts and networks in order to create collaborative systems capable of rapid detection and response to sophisticated cyber threats [5, 6]. These types of systems are called collaborative intrusion detection systems (CIDS).

While CIDS and collaboration are good approaches to improving cyber defense, they raise several issues. The first is a question of network structure and how nodes should share information amongst themselves. There are three approaches to CIDSs, centralized, hierarchical, and distributed. A detailed analysis of these various approaches can be found in [6]. In a perfect world, every node in a network would share information with every other node in a fully distributed system. Unfortunately, as a network grows larger, the overhead required to implement such a system can become unmanageable (connections between nodes scale to $\frac{n(n-1)}{2}$ where n is the number of nodes).

These issues are even more difficult to resolve in wireless ad hoc networks. These types of networks introduce additional concerns such as limited battery life, an unreliability of links, and a changing network architecture that make defense paradigms more difficult [7]. Figure 1 captures the essence of the problem. At any given time, not all of the links are necessarily active.

URI: https://hdl.handle.net/10125/71446
978-0-9981331-4-0

In fact, at this specific time t, Node f (depicted by the dashed circle) has been dropped from the network.

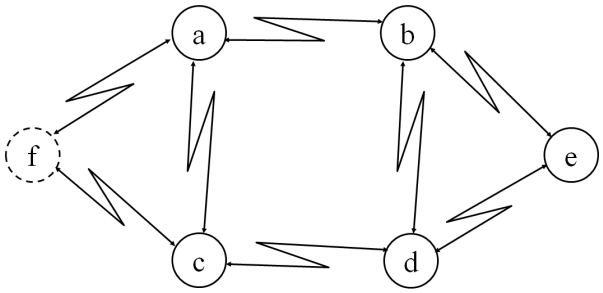

Figure 1. Snapshot of the Configuration of a Wireless Network at time t

In a dynamic wireless ad hoc environment where both the number of nodes and the links between them can change, it is difficult to acquire and process global state information for defense purposes. The question then becomes, for a particular wireless node, what other nodes provide the most useful information for cyber defense?

In order to answer this question, we briefly examine how insect colonies, particularly honeybees, make collective decisions. Insects exhibit a powerful ability to make collective decisions with relatively little communication, very little computational power, and in short amounts of time. Most importantly, they make these decisions using local, not global information. Insect swarms are in many ways similar to a wireless network. There are a variable number of scouts communicating information to dynamic groups of workers. This idea is analogous to nodes in a wireless network which may or may not be actively connected to the network at a specific time. Furthermore, nodes that are connected to the wireless network have to be able to process information from a variable number of other nodes.

In this type of environment, insects have to make correct collective decisions to ensure survival. In his book Honeybee Democracy, Seeley points out that a bee can only observe and react to her neighbors and thus operates only with local information without the full global context [8]. Bees choose foraging sites and potential new hive sites by means of a waggle dance. Scout bees travel in all directions and upon returning to the hive, indicate the direction and "goodness" of their discovered site to the remainder of the bees by dancing. The speed and intensity of the dance indicate how good a potential site is. Any scout bee that observes that dance can visit that site. Over time, good sites are visited more frequently and bad sites are discarded. Thus, using only local information, bees are able to make effective collective decisions.

The idea that nodes can use their close neighbors to make decisions is not particularly new. Modern routing algorithms like OLSR [9] have made extensive use of this type of logic. More specifically, several different works have used insect behaviours to make a similar point. A survey of these works can be found in [10]. Wedde *et al.* proposed a fault-tolerant routing algorithm using "foraging zones" to update local routing algorithms based on the communication exhibited by honey bees [11]. Korczynski *et al.* propose a non-parametric exchange of levels of concerns from neighborhoods of nodes to adjust the sensitivity of local detection algorithms based on bee communication methods [5]. Information from a node's neighbors can prove useful to the intrusion detection capabilities of a node.

Another critical component of information sharing systems is trust. There is a need for a mechanism to evaluate the trustworthiness of nodes that are sharing information, as a malicious node can send out false data that can degrade the performance of any collaborative approach [12]. Trust in wireless and mobile networks has been a subject of intense scrutiny [13] due to the aforementioned additional challenges presented by a dynamic network. Any potential mitigation strategies must begin with a definition of trust. Given an entity A, an entity B, and a behavior X, a useful definition is A's subjective estimation of the probability that B displays the behavior X [14]. There are a litany of previous works that address trust and the related problem of collective decision making in the cyber environment. Consensus theory [15] and game theory [16] are two broad fields that have provided strategies to address this deficit of trust.

In this paper, we borrow a common trust mechanic to assess the trustworthiness of a node at any given time. Simply put, if a neighbor node acts in an expected or beneficial manner, we trust it more. If the node acts in a manner detrimental to our interests, we decrease our trust in it and act accordingly. Again, this idea is nothing particularly new and has been used in trust models for cyber security [17] and in other fields such as game theory. Consider the example of the prisoner's dilemma in which two prisoners are arrested and have a choice to flip on each other, with the payoff table below [18].

		Prisoner A	
		Cooperate	Defect
Prisoner B	Cooperate	$1, 1$	$0, 4$
	Defect	$4, 0$	$3, 3$

Table 1. Payoff Table for Prisoner's Dilemma Game adapted from [18]

Based on the table, the purely rational decision is for both players to betray each other even though both players cooperating (staying silent) results in a better outcome for both of them. If this game is played repeatedly (the iterated prisoner's dilemma) the optimal strategy is tit-for-tat [19] or do the same thing that the other prisoner did in the last round. In other words, stay silent and cooperate with the other prisoner until they defect/betray and once they do, retaliate. This idea provides support for our trust mechanic, increase trust in a node if they behave beneficially and penalize the node if they behave detrimentally.

Another example of this mechanic can be seen in repeated binary voting. Kamhoua *et al.* proposed a game theoretic approach to mitigate malicious node behavior in binary decisions [16]. They proposed nodes building their future reputations using their past behaviors in a game theoretic construct using binary voting. We leverage this concept and portions of their model in Sections 2 and 3.

Our focus in this research area is to address trust in the information sharing process by tying together different concepts from existing cyber security and consensus work. In this paper, we propose a scheme informed by existing strategies that can be used in any network to establish trust ratings for nodes and their neighbors. In this way, those nodes can weigh information from their neighbors for the purposes of cyber defense.

2. Proposed Trust Framework

As noted in Section 1, the idea of neighbors informing their neighbors for the purpose of cyber defense is not new. In [5], the authors present the idea of non-parametric levels of concern that nodes pass amongst themselves that help to adjust the sensitivity of their local detection algorithms. However, their approach does not take into account the ability of a node to be compromised or simply incorrect in its analysis. We leverage a portion of the Repeated Binary Voting Algorithm proposed in [16] to establish reputation values for each node. In this way, a node can weigh levels of concerns from their neighbors. In [16], reputation is a quantity, where a node g has has reputation $R_g(t)$ at time t is given by:

$$R_g(t) = \begin{cases} 0.5 & t = 0 \\ (1-\gamma)R_g(t-1) + \gamma & \text{node vote correct} \\ (1-\gamma)R_g(t-1) - \gamma & \text{node vote incorrect} \end{cases} \quad (1)$$

The γ value is referred as the smoothing factor and is bounded $0 < \gamma < 1$. It is adjusted based on the desires of the defender. The larger the γ value is, the more the node values recent events versus the history that has been established. Kamhoua et al. recommend a γ value of 0.1 [16].

In [16], nodes vote on the state of nature, which was a node's belief in the percentage of the system nodes that was compromised. We adapt their model and provide different definitions for correct and incorrect votes. In [20], a level of concern represents non-parametric data regarding the likelihood of an attack. The nature of attack is unimportant as different nodes may have different intrusion detection mechanism. In their model, they make the assumption that if a node's neighbor is attacked, it is more likely for that node to be attacked shortly thereafter. We represent this as a conditional probability, $\Pr[\text{Node A is attacked} \mid \text{Node A's neighbor was attacked}]$. Thus, an indication of attack from a node's neighbor can be useful in adjusting the threshold values intrusion detection algorithm for the node. We use this idea to adjust the rules from Equation 1.

We define our model as follows. Let the network of nodes G be a connected graph defined as $G = \{g_i, i = 1, 2, ...n\}$ where n is the number of nodes in the graph. Each node has an associated list of neighbors $A = \{a_j, j = 1, 2, ...n-1\}$. Finally, let t represent a time interval. For maximal abstraction, we do not force a value for the length of the time interval, but rather reference a particular interval $t \geq 0$.

A neighbor node a_j has its reputation increased if it reports an attack and the node g_i is attacked. This is our definition of a correct vote. An incorrect vote is characterized by one of two cases. Either neighbor node a_j does not report an attack and node g_i is attacked or node a_j reports an attack and node g_i is not attacked. The second case in particular is important as if a neighbor node is not penalized for reporting false attacks, a malicious node would always report an attack. However, the reputation damage that is caused in the second case should be lower than that of the first case. Figure 2 summarizes these different cases.

With those requirements in mind, our adjusted rules are as follows:

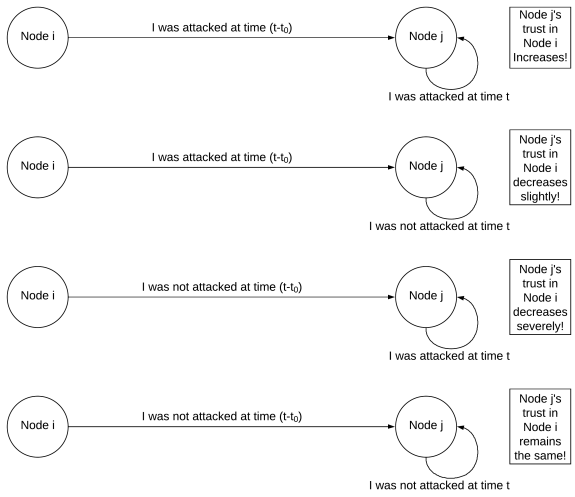

Figure 2. Overview of Correct and Incorrect Votes

$$R_g(t) = \begin{cases} 0.5 & t = 0 \\ (1-\gamma)R_g(t-1) + \gamma & \text{node vote correct} \\ (1-\gamma)R_g(t-1) - \gamma & \text{node vote incorrect case1} \\ (1-\gamma)R_g(t-1) - \alpha \times \gamma & \text{node vote incorrect case2} \end{cases} \quad (2)$$

The value α is a scale factor where $0 < \alpha < 1$ that can be adjusted to penalize a node for reporting an attack where one has not occurred. For the purpose of analysis we define a global p_{att} that represents the probability that any node g_i is attacked in a particular time interval t. The scale factor α should dependent on the value of p_{att}. We initially set $\alpha = 0.1 \times p_{att}$. Table 2 summarizes the effect of each of the cases on reputation.

		Node g_i	
		Attack	No Attack
Node a_j	Attack	$+\gamma$	$-\alpha \times \gamma$
	No Attack	$-\gamma$	0

Table 2. Effects of Correct and Incorrect Votes on Reputation

Table 2 describes the interaction between a pair of nodes. At every time interval t, each node g_i updates its relationship with its neighbors based on the rules described in Equation 2. Algorithm 1 presents a high-level description of these interactions.

.

Similar to [5], we make no assumptions about the type of attack or the detection algorithm implemented by a particular node. Hence, the functions $g.isAttacked(t)$ and $a.wasAttacked(t - t_0)$ are dependent on the IDSs that are utilized for each node. For simplicity of analysis, we assumed the best case scenario that each node detected attacks perfectly. It is easy to see that if this is not the case, trust amongst nodes will decrease. We can think of this as distrusting a node that is particularly bad at detecting attacks. Naturally, we would want to ignore input from such a node. Mathematically, this can be seen by examining the conditional probability $\Pr[g_i.isAttacked(t) \mid a_j.wasAttacked(t)]$. If the node a_j is bad at detecting attacks, this conditional probability will be lower than if the node was good at detecting attacks. As this case is the only case in which a neighbor node can increase its reputation, if the probability of this occurring decreases, the reputation of the neighbor node would also decrease. Similarly, if a node g is bad at detecting attacks, information from its neighbors does not make up for that fact.

Algorithm 1: Algorithm describing Trust Update Process at every time interval t

Timestep *Graph G, Time t, γ, α*
  inputs : Graph G, Time t, γ, α
  foreach *Node g in G* **do**
    $g.isAttacked(t) = (0, 1)$
    foreach *Node a in* $g[neighbors]$ **do**
      Update $g[trust][a]$ based on values of $g.isAttacked(t)$ and $a.wasAttacked(t - t_0)$ according to Table using γ and α.
  return G;

Another important element of the algorithm is the value of t_0. We make no assumptions about the duration of t or the number of time intervals t_0 that a node g is willing to accept. In our testing, we chose to use the simplest case, $t_0 = 1$. Additional considerations must be taken into account as t_0 increases. Specifically, one may want to add extra rules to prevent an adversary node from maintaining its reputation. These considerations are left for future work. Lastly, we make no assumptions about the type of node or network. This framework is as applicable to wired network as a wireless one. This is because the trust relationship between nodes is explicitly based on neighboring nodes and the prior relationship between them. As nodes disconnect and reconnect, the history of their relationships does not disappear and can still be leveraged by our framework.

3. Results

We implemented our trust framework on a Linux box running Ubuntu 18.04.1 and used the Python3 language. Graphs were implemented using the $Networkx$ package [21], specifically using the Newman-Watts-Strogatz small-world graph generator [22]. For the purposes of initial framework analysis, we used a static graph with static links.

Our first set of experiments was designed to test the effects of varying the conditional probability $\Pr[g_i.isAttacked(t) \mid a_j.wasAttacked(t)]$. We used values of 0.4, 0.6, and 0.8 as our conditional probabilities with a graph consisting of 10 nodes. The global probability of attack was 0.1. Figure 3 shows the configuration of the wireless network graph.

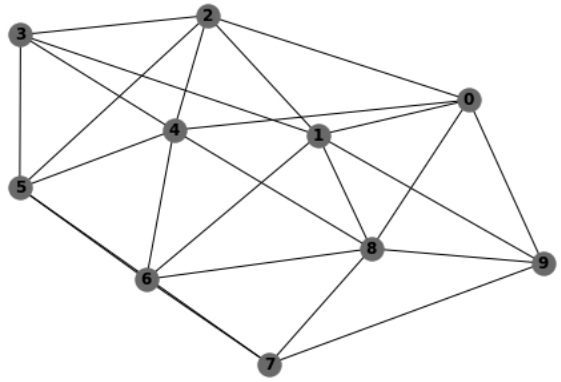

Figure 3. Initial Configuration of Wireless Network Graph with 10 Nodes

Figure 4 presents a comparison of the different conditional probabilities. An arbitrary node and its neighbors were chosen for this figure. In this case, the trust relationship over 500 timesteps between Node 1 and Node 9 was examined. This figure reveals a couple of very important points. As the conditional probability increases, the trust in a neighbor node increases. This observation is simple but very powerful. If an attack on a node g is uncorrelated with a previous attack on one of its neighbors, then the node should not use indications from that neighbor to inform its detection algorithm. This is clearly demonstrated by the upward average trend in reputation as the conditional probability $\Pr[g_i.isAttacked(t) \mid a_j.wasAttacked(t)]$ increases. In this implementation, we capped the lower bound for reputation at 0 instead of allowing the reputation to be negative. Capping the lower bound at 0 only had an effect on the $p = .4$ case as trust values routinely went negative. As the conditional probability $\Pr[g_i.isAttacked(t) \mid a_j.wasAttacked(t)]$ increases, fewer timesteps exhibit a negative trust value. In Figures 4(b) and 4(c), the reputation value never goes to 0. Similarly, we also capped the upper bound for reputation at 1.

Our next experiment was designed to see if the reputation value converged to a particular value over a number of runs. For reproducibility purposes, we ensured that the random number generator we used was seeded with the same value to generate the graphs in Figure 4. In this experiment, we ran our algorithm for 100 different progressions of 500 timesteps. We fixed the configuration of the graph and varied the attack events. We then averaged the results together and displayed them in Figure 5. Furthermore, we then plotted the average reputation for several different conditional probability values. Figure 5 and Figure 6 confirm the trend described in the first experiment. As the conditional probability increases, the average reputation value of the neighbor node increases. Furthermore, that trust value converges as the number of timesteps increases. This observation led us to vary our starting reputation value, which was originally set to 0.5 as was done in [16]. The value of $R_n(0)$ did not impact the converged value and only had a limited impact on any individual run.

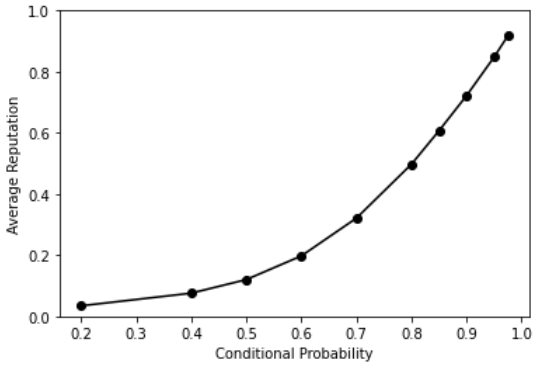

Figure 6. Average Reputation versus Conditional Probability $\Pr[g_i.isAttacked(t) \mid a_j.wasAttacked(t)]$

While the experiments above only tracked the trust relationship between Node 1 and Node 9 (the trust Node 1 had in Node 9 is not necessarily the same as the trust Node 9 has in Node 1), these results were observed throughout the network. Table 3 shows a snapshot of the $p = 0.8$ case at $t = 250$, and Table 4 shows the average reputation values of all of the nodes with the same p-value.

The value nan indicates that there was no edge connecting the two Nodes i, j. As noted before, reputation relationships are not reciprocal. Also, in this implementation, trust relationships are not transitive.

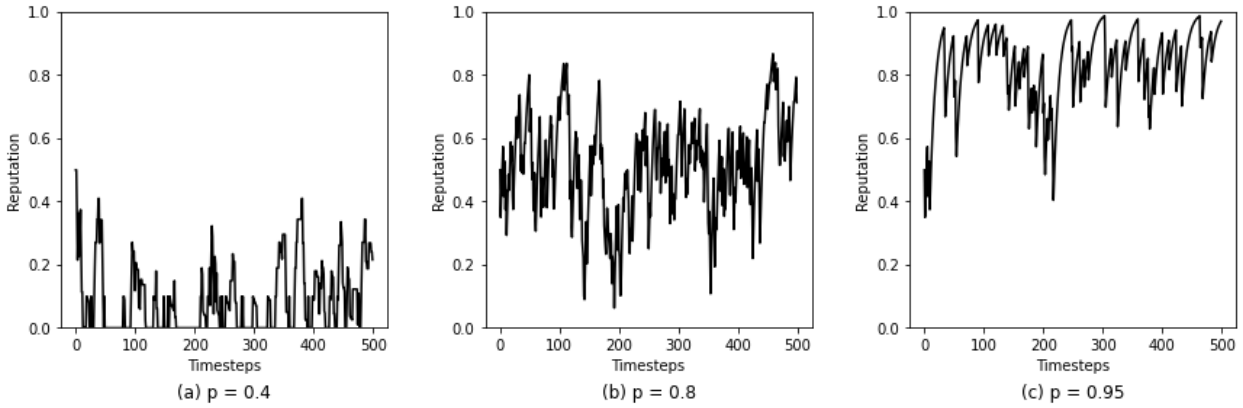

Figure 4. Comparison of Trust Profiles with Increasing Conditional Probability
$\Pr\left[g_i.isAttacked(t) \mid a_j.wasAttacked(t)\right]$

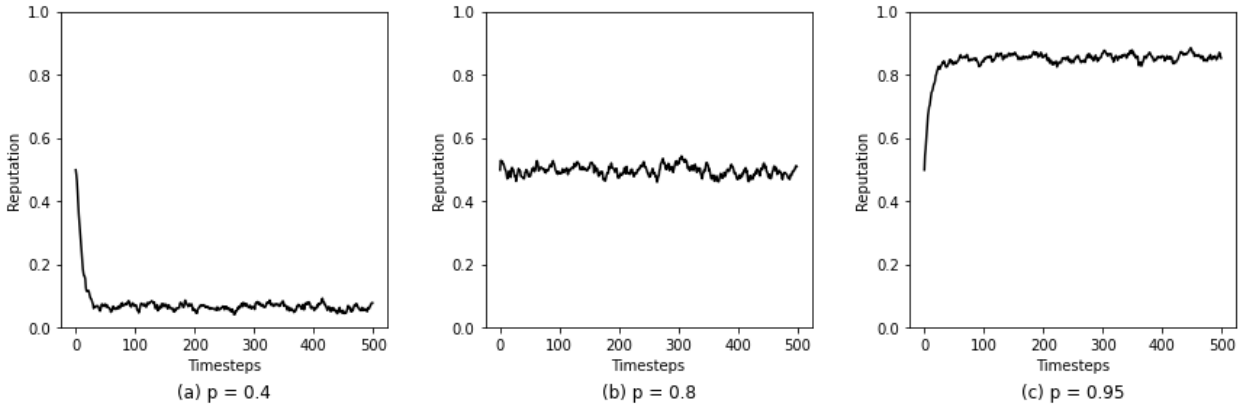

Figure 5. Comparison of Average Trust Profiles over 100 Runs with Increasing Conditional Probability
$\Pr\left[g_i.isAttacked(t) \mid a_j.wasAttacked(t)\right]$

	0	1	2	3	4	5	6	7	8	9
0	nan	0.41	0.32	nan	0.41	nan	nan	nan	0.27	0.41
1	0.34	nan	0.63	0.73	nan	nan	0.56	nan	0.49	0.25
2	0.59	0.53	nan	0.62	0.55	0.44	nan	nan	nan	nan
3	nan	0.60	0.45	nan	0.77	0.28	nan	nan	nan	nan
4	0.29	nan	0.42	0.79	nan	0.22	0.64	nan	0.64	nan
5	nan	nan	0.27	0.56	0.56	nan	0.38	0.10	nan	nan
6	nan	0.55	nan	nan	0.66	0.29	nan	0.19	0.50	nan
7	nan	nan	nan	nan	nan	0.57	0.50	nan	0.25	0.21
8	0.22	0.54	nan	nan	0.63	nan	0.56	0.32	nan	0.24
9	0.11	0.67	nan	nan	nan	nan	nan	0.21	0.60	nan

Table 3. Reputation Values for all Nodes at $t = 250$, $p = 0.8$

	0	1	2	3	4	5	6	7	8	9
0	nan	0.50	0.49	nan	0.53	nan	nan	nan	0.39	0.49
1	0.43	nan	0.52	0.51	nan	nan	0.52	nan	0.48	0.51
2	0.46	0.53	nan	0.51	0.52	0.44	nan	nan	nan	nan
3	nan	0.49	0.49	nan	0.50	0.42	nan	nan	nan	nan
4	0.42	nan	0.51	0.50	nan	0.44	0.50	nan	0.48	nan
5	nan	nan	0.48	0.51	0.53	nan	0.45	0.41	nan	nan
6	nan	0.52	nan	nan	0.47	0.46	nan	0.46	0.46	nan
7	nan	nan	nan	nan	nan	0.45	0.46	nan	0.46	0.46
8	0.41	0.51	nan	nan	0.49	nan	0.47	0.41	nan	0.48
9	0.44	0.51	nan	nan	nan	nan	nan	0.43	0.48	nan

Table 4. Average Reputation Values for all Nodes after $t = 500$, $p = 0.8$

That is to say, just because Node i trusts Node j and Node j trusts Node k, this does not mean that Node i trusts Node k. This type of logic could be implemented in future work.

Table 4 shows that although the reputation value at any given t might differ, the average reputation value remains the same around 0.5. This is the same value that was very clearly displayed in Figure 5(b).

Finally, we varied the number of nodes from 10 to 100 (with a global probability of attack of 0.01, which is 1 divided by the number of nodes) and observed the effects. We concluded that scaling up the number of nodes made no discernible difference in the reputation values as would be expected. In our model, each node's relationship with another node is somewhat independent

from any other node. Of course, there are second and third order effects that can have an impact on reputation values. For example, if Node i's neighbor j has a neighbor that is attacked, then Node j will be more likely to be attacked in the next time interval. Following that interval, if Node j is attacked , Node i will be more likely to be attacked and so on and so forth. A more detailed analysis of these effects is planned for future work.

4. Conclusion

We have proposed and implemented a trust framework to dynamically adjust reputation values of wireless nodes and their neighbors. Nodes tracked their level of trust in their neighbors based on whether they benefited from the attack indications that their neighbors provided. While we did not specifically examine dynamic changes to the network (dropping/adding nodes or links), our initial experiments show that this framework is capable of tracking reputation values over long periods of time and in a stable manner. It also appears scalable. These indications validate future research in this area for both wired and wireless networks.

The framework developed in this paper is by no means a finished product. There are still additional parameters that must be considered. Specifically, we do not fully explore varying the number of time intervals for the set of rules. We considered a positive indication of trust to be if Node g_i was attacked in time interval t and if one of its neighbors a_j indicated it was attacked in time interval $t-t_0$. We did not consider the cases where neighbor a_j was attacked in time interval $t-k*t_0$ where k is some arbitrary number or intervals. We believe that this would have a substantial effect on the system and would alter the set of rules based on the number of intervals.

We also do not fully explore how a malicious node would try to leverage our rules to maintain its reputation. Further analysis is necessary to identify optimal strategies for a malicious node and then to adjust our model accordingly.

This framework is a small piece in the overall CIDS challenge. Our next step would have to incorporate this reputation system into an existing CIDS model and explore its benefits on detection probability. Also not fully explored in this paper is the communication overhead of this system, though we believe it is small as nodes only communicate attack information with their neighbors and are passing small messages.

The sharing of information between cyber defenders is critical to preventing attacks against our cyber systems, especially our wireless systems. That information is only as good as the nodes that share it. We need better ways to track the trustworthiness of the nodes in our wireless networks. The framework proposed in this paper provides a simple and easily implementable way to track trust in adhoc networks. While it is not perfect, it shows promise in this area.

References

[1] U.S. Government Accountability Office, "Federal information security: Agencies and omb need to strengthen policies and practices," Washington, DC, USA. GAO Report No. GAO-19-545, 2019.

[2] U.S. Government Accountability Office, "Information security: Agencies need to improve implementation of federal approach to securing systems and protecting against intrusions," Washington, DC, USA. GAO Report No. GAO-19-105, 2018.

[3] Identity Theft Resource Center, "2019 end-of-year data breach report," 2019.

[4] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, pp. 805–822, 1999.

[5] M. Korczynski, A. Hamieh, J. H. Huh, H. Holm, S. R. Rajagopalan, and N. H. Fefferman, "Hive oversight for network intrusion early warning using DIAMoND: a bee-inspired method for fully distributed cyber defense," *IEEE Communications Magazine*, vol. 54, pp. 60–67, June 2016.

[6] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Comput. Surv.*, vol. 47, pp. 55:1–55:33, May 2015.

[7] B. Subba, S. Biswas, and S. Karmakar, "Intrusion detection in Mobile Ad-hoc Networks: Bayesian game formulation," *Engineering Science and Technology, an International Journal*, vol. 19, pp. 782–799, June 2016.

[8] T. D. Seeley, *Honeybee democracy*. Princeton: Princeton University Press, 2010.

[9] "Optimized Link State Routing Protocol (OLSR)," Tech. Rep. RFC3626, RFC Editor, Oct. 2003.

[10] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, pp. 61–85, June 2009.

[11] H. F. Wedde, M. Farooq, and Y. Zhang, "Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior," in *Ant Colony Optimization and Swarm Intelligence* (M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, eds.), (Berlin, Heidelberg), pp. 83–94, Springer Berlin Heidelberg, 2004.

[12] C. Fung, J. Zhang, I. Aib, and R. Boutaba, "Trust Management and Admission Control for Host-Based Collaborative Intrusion Detection," *Journal of Network and Systems Management*, vol. 19, pp. 257–277, June 2011.

[13] T. K. Kim and H. S. Seo, "A trust model using fuzzy logic in wireless sensor network," *World academy of science, engineering and technology*, vol. 42, no. 6, pp. 63–66, 2008.

[14] P. C. Bauer, *Three essays on the concept of trust and its foundations*. PhD thesis, Universität Bern, 2015.

[15] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI 99, (USA), p. 173186, USENIX Association, 1999.

[16] C. A. Kamhoua, K. A. Kwiat, and J. S. Park, "Surviving in Cyberspace: A Game Theoretic Approach," *Journal of Communications*, vol. 7, pp. 436–450, June 2012.

[17] G. Theodorakopoulos and J. S. Baras, "On trust models and trust evaluation metrics for ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 318–328, 2006.

[18] D. Bauso and Now Publishers, *Game theory: models, numerical methods and applications*. 2014. OCLC: 905837938.

[19] R. Axelrod and W. D. Hamilton, "The evolution of cooperation," *science*, vol. 211, no. 4489, pp. 1390–1396, 1981.

[20] M. Korczynski, A. Hamieh, J. H. Huh, H. Holm, S. R. Rajagopalan, and N. H. Fefferman, "DIAMoND: Distributed Intrusion Anomaly Monitoring for Nonparametric Detection," in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, (Las Vegas, NV, USA), pp. 1–8, IEEE, Aug. 2015.

[21] "NetworkX NetworkX documentation." `https://networkx.github.io/`.

[22] M. Newman and D. Watts, "Renormalization group analysis of the small-world network model," *Physics Letters A*, vol. 263, pp. 341–346, Dec. 1999.