Security in MANETs using reputation-adjusted routing

Attila Ondi, Katherine Hoffman, Carlos Perez, Richard Ford, Marco Carvalho, et al.

Attila Ondi, Katherine Hoffman, Carlos Perez, Richard Ford, Marco Carvalho, William Allen, "Security in MANETs using reputation-adjusted routing," Proc. SPIE 7344, Data Mining, Intrusion Detection, Information Security and Assurance, and Data Networks Security 2009, 734406 (13 April 2009); doi: 10.1117/12.818894

Event: SPIE Defense, Security, and Sensing, 2009, Orlando, Florida, United States

Security in MANETs using Reputation-Adjusted Routing[1]

Attila Ondi[a], Katherine Hoffman[a], Carlos Perez[b], Richard Ford[a], Marco Carvalho[b], William Allen[a]
[a]FIT, 150 W. University Blvd., Melbourne, FL 32901
[b]IHMC, 40 South Alcaniz Street, Pensacola, FL 32502

ABSTRACT

Mobile Ad-Hoc Networks enable communication in various dynamic environments, including military combat operations. Their open and shared communication medium enables new forms of attack that are not applicable for traditional wired networks. Traditional security mechanisms and defense techniques are not prepared to cope with the new attacks and the lack of central authorities make identity verifications difficult. This work extends our previous work in the Biologically Inspired Tactical Security Infrastructure to provide a reputation-based weighing mechanism for link-state routing protocols to protect the network from attackers that are corrupting legitimate network traffic. Our results indicate that the approach is successful in routing network traffic around compromised computers.

Keywords: MANET, Security, Routing, Simulation

1. INTRODUCTION

Mobile Ad-Hoc Networks (MANETs) are critical key technology enablers for various scenarios that include, for instance, disaster relief and military combat operations. Security considerations in a MANET are a superset of those in wired networks. Possible frequent change of routes and the open and shared communication medium are among some of the characteristic properties of mobile wireless network environments that facilitate attacks such as easy eavesdropping and rush attacks, for example. It is imperative, therefore, to ensure the intended cooperation of MANET participants. The lack of a central authority makes verification of identities difficult, however.

Here we expand on our previous work in BITSI (Biologically Inspired Tactical Security Infrastructure) to create a reputation-based weighting mechanism for link-state routing protocols in MANETs. In this paper we make the relaxing assumption that each and every participating node in the MANET encompasses a trusted core (i.e. BITSI), the identity of which can be verified by neighboring nodes. Such assumption is not unrealistic in a combat scenario – the computers of an army can be equipped with a tamper-resistant hardware component that monitors and reacts to the actions of the computer. We focus here on how nodes can be protected from damage by adjusting the routing weights around the offending nodes – thus building security into routing without significantly needing to modify the underlying routing algorithm.

1.1 Overview of this work

After a short presentation of the essential background information we describe the proposed algorithm, which is illustrated on a simple scenario. We present the related work of other researchers in similar areas and then describe the setup of some initial experiments. The results are then presented and discussed. The paper is then finished with an overview of our proposed work on extending our approach.

2. BACKGROUND

The approach described in this work is one feature of a larger framework: the Biologically Inspired Tactical Security Infrastructure (BITSI). This framework has been developed to provide security in MANETs where traditional security solutions (e.g. firewalls, anti-virus software and intrusion detection systems) are unable to address the unique threats faced by ad-hoc networks [5]. To achieve this goal, BITSI uses a combination of *reputation*, *reinforcement learning* and *danger theory* to autonomously reconfigure and manage node stance to enhance security. BITSI is a trusted component on the computer and is operating at the kernel level. Perhaps the most significant difference between the goals of traditional security solutions and that of BITSI is that while traditional protection is focused on the defense of a specific

[1] This work was sponsored by the U.S. Army Research Laboratory - Cooperative Agreement W911NF-080200023.

system or group of systems, BTISI attempts to protect the *mission* of the computers comprising the network. The implication of this difference is that for BITSI – as long as the mission is still achievable – the compromise or damage of computers is acceptable (although not desired). It is important to note that the criticality of computers can change dynamically during the course of the mission; therefore the importance of protection can shift between computers.

BITSI is a tamper-proof trusted component on the system. Although this setup is unusual in everyday scenarios, it is easily envisioned in a military scenario, where all computers of an army are required to fulfill a number of requirements on the hardware before they are deployed on the battlefield.

2.1 HSLS

As the nodes in a MANET are not fixed, MANETs typically employ routing protocols that are specifically designed to support frequent changes in link characteristics and network topology. In this paper, we have chosen the Hazy-Sighted Link-State protocol (HSLS) [21]. HSLS is a proactive routing protocol, which means that routes are discovered and maintained continuously (as opposed to reactive routing protocols that discover routes on demand). HSLS is shown to have close-to-optimal asymptotic overhead from routing messages among both pro- and reactive algorithms [21]. Connections between nodes are discovered via the exchange of HELLO messages and are stored in a connectivity matrix. The link weights between a node and its neighbors are augmented by the elapsed time since the node received the last HELLO from the corresponding neighbor. The routing table can then be computed from the connectivity matrix using Dykstra's algorithm.

3. REPUTATION-ADJUSTED ROUTING

In HSLS nodes broadcast their incoming links (other nodes they heard from) and only learn about their outgoing links by receiving the incoming-link notifications of their neighboring nodes. The reason behind this is that communications are not always bidirectional – differences in antenna strength, elevation; absorption and reflection of signal carrying waves can result in various transmission ranges for different nodes. In order to be able to route around a node, the routing weights from neighboring nodes towards the undesired node must be higher than towards other possible routing candidates. In this paper, we explore the use of the link weights, advertised by the HSLS HELLO messages, to modify the routing costs to their neighbors based on reputation information.

Due to the distributed nature of the MANET, sharing reputation information is difficult and potentially open to abuse. To address this, considerable work has been carried out to reduce the overhead of reputation sharing and the impact of liars [18]. However, if we use the BITSI kernel at each node to store reputation (that is, each node carries with it the opinions others have of it), this group reputation can be easily used to modify the cost of routing weights. While this approach requires BITSI to act in a trustworthy way, the higher-level system design provides some protection from rogue nodes.

3.1 Algorithm

Whenever a node perceives damage[2] from a received packet, it issues a damage report to its BITSI component. Since the exact node that corrupted the packet is unknown, BITSI will reduce the reputation information equally[3] – the total reduction is based on the severity of the damage – for all nodes in the forwarding path[4]. Every time a node's reputation information is reduced, that node is notified of the new reputation value by the sender.

When a node receives a reputation adjustment message, it stores that information associated with the sender of the reputation reduction. The node then adjusts its routing link weights based on the combined reputation others have about them.

In order to forgive nodes that were blamed incorrectly, or blamed for damage that was not their fault, the reputation information is periodically incremented. For the sake of simplicity, reputation is incremented by a fixed value at regular intervals in the current implementation. This approach is unlikely to be optimal, and will be explored in future work.

[2] For a discussion on damage as used here, see the Related Work section of this paper.
[3] We are looking for distributing the blame unequally in light of additional information about the nodes.
[4] For the sake of simplicity we assume here that the actual routing path is embedded in the packet (i.e. the BITSI components on the forwarding nodes attach the address of the forwarder to the packet); however, our intuition is that this assumption is not necessary for the algorithm to work efficiently (see the Future Work section of this paper).

Every node stores two reputation scores for each other node that it has learnt about through HELLO messages. One score is the current node's perception of the other's reputation[5]; the other score is the perception of the other node about the current node's reputation (if known). These latter are combined to supplement the routing link weight. Both sets of reputations are periodically increased, using the increase algorithm described above, so that the sender and receiver are generally in agreement,

Although possibly more than one node is blamed for the damage, if the damage is consistently originating from the attackers, routing shifts to alternative paths through nodes with less tarnished reputations. If the path through which damage arrives changes, we expect that soon only the common elements in the path (usually the real attackers) will end up with reduced reputations.

It is important to realize that the originator of the packet might be the cause of the damage. In this paper we assume that this is not the case, however we intend to apply adequate measures to filter this case as well (see the Future Work section).

Another issue is related to large messages that are assembled from multiple packets. It is possible for individual packets to travel on different forwarding paths before reaching the destination. As damage might not be realized until the whole message is assembled, it is important to select the right packet for the reputation algorithm (see the Future Work section of this paper). A relatively easy – although possibly inefficient – solution is to bypass differentiation of individual packets and blame all nodes on all forwarding paths for the damage.

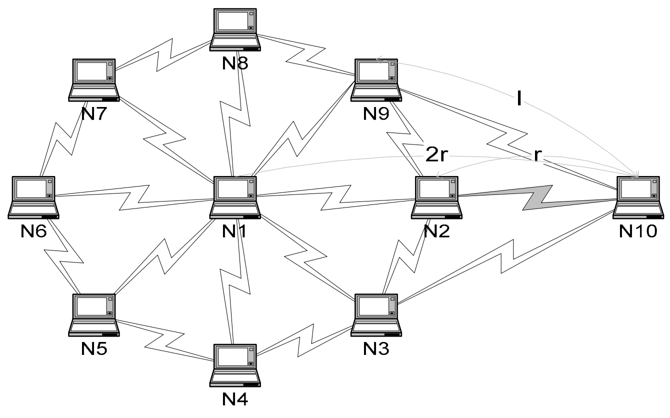


Figure 1: Simple circular movement scenario

3.2 Example

To show the idea in work, let us consider a simple scenario. The MANET consists of 10 nodes (*N1*…*N10*), of which *N2*-*N9* forms a circle or radius *r* with equal distance from each other; *N1* is stationary in the center of the circle; and *N10* is initially positioned outside the circle on the line connecting *N1* and *N2* at distance *r* from *N2*. Let the communication radius (*l*) of all nodes be equal such that $r<l<2r$ (Figure 1). *N10* periodically sends UDP-based messages to *N1* (e.g. status-report).

In this scenario, there is a connection between *N1* and *N2*…*N9*; for each $2\leq i\leq 9$, *Ni* is connected to at least two of its neighbors in the circle; furthermore, *N10* is initially connected to *N2*, *N3* and *N9*.

Let us also assume that *N10* is routing towards *N1* through *N2*, and furthermore, the application traffic leaving *N2* arrives corrupted at *N1*. In this very simple scenario, the packets that arrive at *N1* have been forwarded only by *N2*. Unless *N10* changes its routing table toward *N1*, or starts moving and leaves the communication area of *N2*, all traffic sent by *N10* will arrive corrupted at *N1*.

[5] Although it is possible to just send the amount of reduction, we opted to keep a copy of the reputation score on the victim as well in case the reputation modification turns out to be undeliverable (e.g. due to packet loss).

Now, let us consider how the situation changes if BITSI is active. As soon as the application on *N1* detects the corruption, it signals a damage report to the BITSI component residing on the same node, blaming all the nodes on the forward path from the sender (*N2* in this case). The reputations of the blamed nodes are decreased and each is issued a notification about their new reputation score by *N1*. Once *N2* receives its reduced reputation from *N1*, it modifies (increase) its incoming link weights based on the modified reputation score. When *N10* learns about the new link weights, its routing table is modified to relay through a less costly route (either through *N3* or *N9*), thereby bypassing the corruption from *N2*. As long as the weight for *N2* is higher than its alternatives (N3 and N9), the nodes route around *N2* (unless the only route is through it), thereby protecting themselves from further damage. After *N2* is forgiven, if it is still corrupting the traffic, its weights will increase as before and the network will be protected again.

3.3 BITSI parameters

There are five main parameters to the described algorithm: a) the value attributed to the damage, b) the scaling factor from damage value to blame information – i.e. how much reputation is reduced for a particular amount of damage, c) the scaling factor from reputation to routing weights – i.e. how important the reputation is when determining routes, d) the value of forgiveness and e) the time period for the forgiveness calculation.

We propose the *damage value* to be an integer number in the [0, 255] interval – the range was chosen so that it can be represented by one byte in the frame. This allows nodes to attribute different values to attacks based on the severity of the damage. Since damage is perceived on the application level (which could be part of BITSI just as well), the applications will have to be configured to match the role of the computer and the environment they are running on. For example, crashing a database server application might have more severe consequences to the mission on one machine than on another. It is also important to realize that if the application is not part of the trusted component, it is subject to suspicion as it can be reporting incorrect damage value or blaming innocent nodes. In order to properly address this issue, other mechanisms must also be employed (see Future Work).

The *damage-to-blame scale factor* needs to transform the reported damage value to the reputation range of [0, 1], therefore its maximum value is 1/255.

The exact value of the *reputation-to-routing weights scale factor* depends on the range of valid routing weights used by the routing algorithm and other mechanisms that affect the routing weights (e.g. weight modification based on the freshness of the routing information).

Choosing the right *forgiveness value* and *forgiveness frequency* has to be high enough to allow relatively quick recovery for incorrectly blamed nodes, but low enough that blame is not forgiven immediately. The frequency of forgiveness calculation can also change the effectiveness of the system. This is a major subject for future research.

Because our approach relies on the routing protocol itself as a defense mechanism, there is an implicit relationship between the responsiveness of the system and the frequency of route advertisements and route updates. While an adaptive routing algorithm can be used to enhance the effectiveness of the system, we will not consider this variable here. We will assume that routing advertisement and update protocols are fixed and not relevant to our analysis.

4. RELATED WORK

Our proposal has touched on many active areas of research. In this section we present a brief overview of the relevant work of others.

4.1 Secure routing

The literature on the topic of secure routing is abundant. Most of the proposals (e.g. [1, 2, 9]), however, try to identify – and often punish – nodes that do not participate in the routing protocol properly or fail to forward messages for which they are the next hop in the route. In order to achieve this, non-misbehaving nodes are actively listening to and analyzing messages from other nodes, even if they are not the intended handler of the message. In some of these, (e.g. [2]), the learned behavior of others is captured in a reputation and/or trust value that is later used to predict the behavior of others in the network. Since not all nodes are involved in direct communication with all other nodes, the reputation information is disseminated to help others make an educated guess on the expected behavior of a node they interact with, even if they have never interacted directly with that particular node before. The exact scheme of disseminating the reputation information is numerous [3, 24] and usually quite complex since the nodes cannot trust information received from others. This skepticism is greatly reduced when the use of trusted components can be assumed.

Few proposals modify the routing algorithms to change the weights of the links between nodes to encompass additional information. The few examples that do enlist this technique are aiming to ensure the quality of service (QoS) of message delivery – therefore they are concerned with optimal bandwidth allocation [13, 14].

4.2 Danger theory

Danger theory is a relatively new attempt to explain how the human immune system works [17]. Traditional models of immune response have focused on self/non-self recognition. However, this approach fails to account for the richness of the vertebrate immune response. To address this, Matzinger proposed a new model suggesting that the reactive immune system was modulated by the detection of damage at the cellular level. While this model has been subject to considerable discussion in biology, it is surprisingly useful when considering how a computer immune system might function.

Historically, artificial immune system (AIS) work [6, 10, 15] has typically focused on differentiating between self and non-self within either a network or a single computer. However, this distinction can be either difficult or unhelpful. For example, a document is data, and as such cannot usefully be considered non-self, as computers are designed to exchange data. However, this same document could contain macros that damage the machine, or an exploit that allows the execution of arbitrary code. Similarly, new programs are installed on computers all the time. Often these programs are updates of existing software, designed to address vulnerabilities. As such, while they are not “self” (they are new to the computer) they are certainly not items that an AIS should respond to. Thus, the Danger Theory model provides an interesting mechanism for handling new items: if a newly installed program causes no “damage” the AIS does not attempt to detect it. This approach has been used with some success by other researchers (see, for example, [4, 7, 22]), and it forms the basis of the detection system used by BITSI.

Our approach to damage is based on damage to the mission. Thus, it is based on high-level requirements for mission functionality. However, these requirements are broken down into increasingly lower-level descriptions, until individual nodes can detect conditions that are likely to damage the mission. For example, consider a pair of Nodes X and Y that are required by the mission to exchange targeting information. If Node Y begins to receive corrupted packets from Node X, then Y can be said to be experiencing damage. It is this damage we aim to address. We note that this is just one aspect of the BITSI system – a complete overview can be found in [5].

5. SIMULATION

The validity of our approach can be verified in several ways, the three main venues being real-world scenarios, emulation and simulation. Of these three we chose to employ simulation for its cost efficiency and ease of setup.

The choice of tool for carrying out simulations in MANETs is overwhelmingly the ns2 network simulator [19]. We opted to use Hephaestus, a discrete time-step simulator, instead – even though the level of abstraction is higher than that of ns2 – for many of our scenarios are handled more efficiently by that software. It has been shown elsewhere that the two software provide similar results in most MANET routing simulations [20].

5.1 Main scenario

To demonstrate the capabilities of BITSI, the following base scenario is considered. In an area of 200m x 200m, 9 scouts are moving constantly along the border, looking for enemies, with 1m/s random waypoint movement staying within 15m of the border. The scouts send status reports to a server that moves about the center 40m x 40m of the area with 1m/s random waypoint motion. There are also 20 other nodes that occupy the area and provide the routing infrastructure of the MANET between the scouts and the server. The routing nodes are moving within the central 170m x 170m area with 1m/s random waypoint movement. The scouts send their location and status information to the server every 2 seconds. The communication range of all nodes is set uniformly to 90m and the scenario is executed for 10 minutes simulated time. In all scenarios the first 30 simulated seconds were discarded from the measurements to mitigate the variability in the early routing-discovery state of the network.

Let us consider that malicious mobile code resides on some of the routing nodes (e.g. via installation of software from untrusted sources) that corrupts the scout status reports whenever they are routed through the compromised routers. In the following sections we will compare the amount of damage suffered by the server (by receiving corrupted information), under three different setups: a) no router is corrupted – this is a baseline result, b) when some of the routers are corrupted, but the BITSI system is inactive (e.g. not present), and c) when corruption is present and BITSI is active.

The basic scenario is modified in the following ways. Varying the ratio of corrupted nodes to intact nodes shows how BITSI behaves under varying amount of stress; varying the speed of movement of the nodes in the MANET demonstrates the system's reaction to rapid changes in routing; varying the node-density of the network explores the effectiveness of the system under different route-availability. For all scenarios, forty movement scenarios were generated. The simulation was run on each of the generated movement scenarios, thus the random placement and movement of nodes is mitigated by 40 repetitions.

6. RESULTS

Each dot on the following graphs represents the mean of the 20 runs. We measured the total number of non-corrupted packets received over the total number of packets sent[6] – this information indicates the amount of traffic the server received intact

6.1 Speed

Let us first consider the impact of movement speed over the effectiveness of BITSI. In these scenarios there were 30 nodes in the network among which one was corrupting traffic. The speed of movement varied from 1 m/s to 6 m/s.

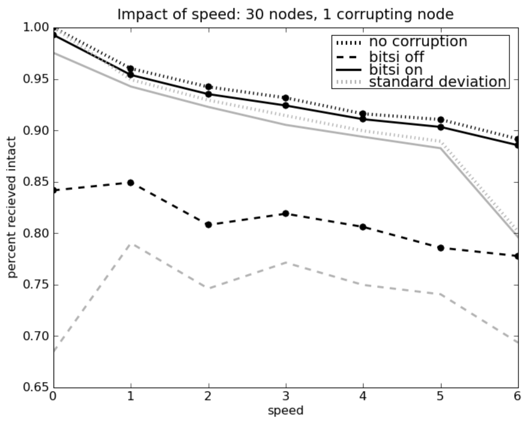


Figure 2: Impact of BITSI over varying movement speed.

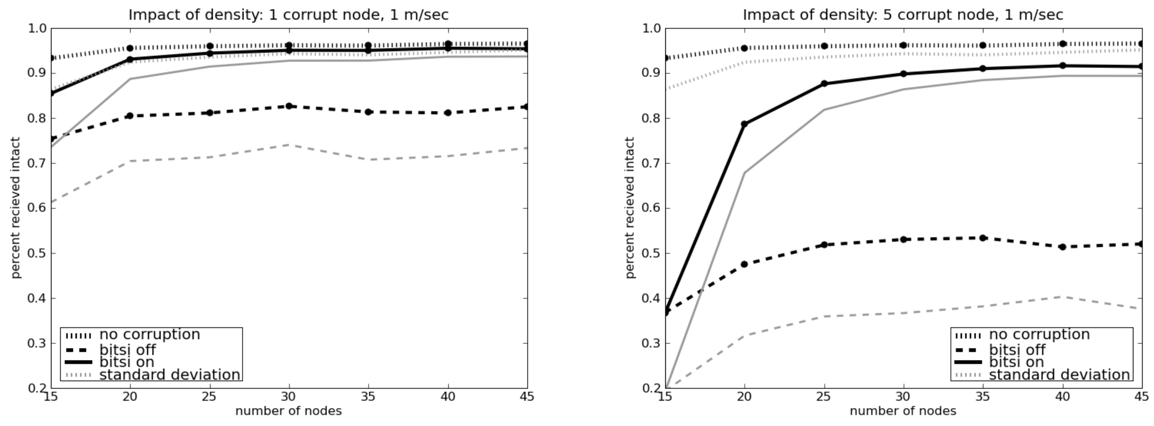
Faded lines show one standard deviation below the mean.

The speed of movement has a significant impact on this scenario – HSLS has a hard time maintaining routes when nodes are moving quickly (Figure 2). With 30 nodes in the simulation, at the lowest speed of 0 m/s, 100% (±0% std) of packets reached their destination when no corruption was present. With the corrupting node active and BITSI inactive, only 84% (±15% std) of the packets sent reached the server intact, whereas, with BITSI turned on, the ratio of non-corrupted packets received is raised to 99% (±2% std). As the nodes start to move, goodput gradually decreases with the increase in movement speed. At the highest simulated speed of 6 m/s, 89% (±9% std) of packets reached the server with no corruption present, which dropped to 77% (±8% std) when corruption was activated; however BITSI was able to increase the goodput back up to 88% (±9% std). Therefore, with BITSI running, performance is only very slightly reduced, compared to the no-corruption scenario, regardless of the movement speed.

6.2 Node density

Let us investigate next the effects of node density in the network. In these scenarios the number of nodes changed from 15 up to 45 with the movement speed held at 1 m/s. There were 1 and 5 corrupting nodes present in two, otherwise identical, setups. The simulations were run for 135 simulated seconds.

[6] The number of packets sent may vary from one movement scenario to another, as a scout will only attempt to send if its routing table indicates that a path exists to the server.



a) One corrupting node

b) Five corrupting nodes

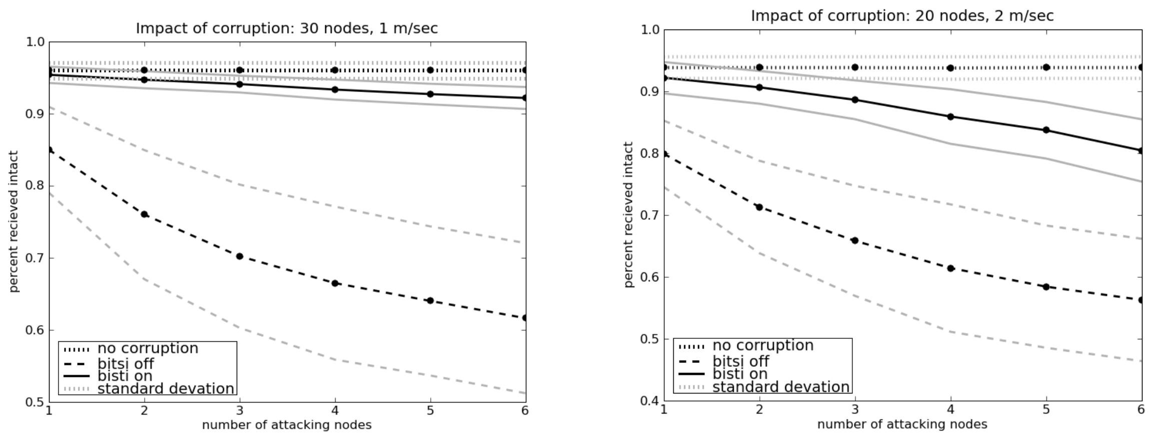
Figure 3: Impact of BITSI over varying node density.

Faded lines denote one standard deviation below the mean.

In the one corrupting node setup (Figure 3a), the number of nodes is not a significant factor, and BITSI restores the goodput to a level nearly equivalent to the scenario where corruption is not present, from the consistent twelve to seventeen percent reduction in non-corrupted packets received without BITSI. With five corrupting nodes (Figure 3b), BITSI is much less likely to be able to find alternative routes in low density situations. At 15 nodes (when all the 5 routers are corrupting traffic) BITSI is unable to improve the goodput, as there are no alternative, non-corrupting routes available. However, with as low as half of the routers non-corrupting (20-node scenario), BITSI improves the situation significantly (from a goodput of 48% (±17% std) when BITSI is not present to a goodput of 79% (±11% std) when BITSI is operational). Moreover, in the 30-node scenario, where a quarter of the routers are corrupting traffic, BITSI is just 8% below the no-corruption results, with a goodput of 89% (±4% std), while the goodput in the BITSI-off case is only 53% (±17% std). Further decreasing the ratio of corrupting to non-corrupting nodes does not change the results significantly.

6.3 Putting it together

Finally, we consider a scenario where the amount of corruption changes in the network. We evaluate the effectiveness of BITSI in two setups. In the first, there are 30 nodes in the network and the movement speed is 1 m/s. The second setup presents more challenge to BITSI with 20 nodes (less density) and higher movement speed of 2 m/s.



a) 30 nodes, movement speed of 1 m/s

b) 20 nodes; movement speed of 2 m/s

Figure 4: Impact of BITSI under varying corruption levels. Faded lines denote one standard deviation around the mean.

The cases where corruption is not present represent the base amount of intact traffic received by the server. The data indicates that the total amount of traffic able to reach the server is 96% (±1% std) in the first setup (Figure 4a) and 94% (±2% std) in the second setup (Figure 4b). The exact amount of traffic received depends primarily on the density of the scenario (i.e. the number of nodes) and on the speed of movement, as paths to the server can be lost.

In the scenario where corrupting nodes are present, but BITSI is inactive, the corruption quickly and steadily diminishes the percentage of packets received intact, down to 61% (±10% std) when there are six corrupting nodes (30% of the routers) in the first setup (Figure 4a). In the second setup, the percent of packets received intact is slightly below the results of the first setup, with 58% (±5% std) goodput when the number of corrupting nodes is 6 (60% of the routers), due to the higher movement speed (Figure 4b).

When corrupting nodes are present and BITSI is turned on, in the first scenario one to three corrupting nodes (5-15% of routers) make little difference to the system; the amount of traffic received intact for BITSI is within five percent of the no-corruption result. As the ratio of corrupting nodes increases (up to 30%), the performance is reduced, though still significantly above that of the scenario without the intervention of BITSI; 92% (±1% std) goodput with 6 corrupting nodes (Figure 4a). In the more challenging scenario, the performance of BITSI degrades faster, but still remains above that of the BITSI-off scenario; recovering the goodput to 80% (±5% std) with 6 corrupting nodes (Figure 4b).

7. CONCLUSIONS

Based on our experiment results we conclude that BITSI is capable of protecting the server from a large amount of corruption when the ratio of corrupt routers to the total number of intermediate routers in the network is not overwhelming (below 50%). This result is even better than our operating assumption for BITSI that the ratio of misbehaving nodes to the total number of nodes in the network is low [5], which we expect to be true on a battlefield. The speed of movement did not cause much degradation to the effectiveness of BITSI until the nodes moved too fast for HSLS to maintain the proper routes. Also, the more dense the network, the more alternatives are present for routing alternatives, which allows bypassing the corrupting nodes.

Based on these results, we conclude that route-adjusting via link-weights through BITSI is capable of significantly reduce (and sometimes almost eliminate) the impact of corruption in the network. In the next section we discuss viable extensions to this work and its integration to the BITSI framework.

8. FUTURE WORK

As the research on BITSI and the protections it can provide is in its early stages, there are many opportunities for further work. The most important areas are packet tracing, "forgiveness" of nodes, and integration with other BITSI features.

8.1 Packet tracing

Part of the current algorithm requires nodes to be able to determine the routing of the corrupted packets. However, in a typical MANET, this will be very difficult, as each node knows only the previous hop of the traffic. The most intuitive solution to this problem is to require the forwarders to attach a set of information potentially required in an investigation attempt. Even if the nodes are trusted to provide the correct information, the packet size could soon reach the point where it cannot be transmitted using the physical communication protocol in use.

Instead of packet tagging, a number of other approaches could be employed. Kim et al. [11] investigated the directionality (and ultimately the origin location) of traffic based on pattern and volume matching. To reduce the bandwidth requirement, the network is partitioned so that only overseers of partitions in the right direction continue the investigation.

Huang and Lee [8] built a trace-back path to identify potential areas of location for misbehaving nodes. They base their proposal on storing hash information about every packet a node forwards and later asking them to match the hash of the investigated packet. While this approach does not guarantee reporting the full route (due to change in the packet hash), it is capable of finding the vicinity of the last node on the forwarding path that either changed the packet or is refusing to comply to the investigation request. Strayer et al. [23] proposed a similar trace-back system based on Bloom filters.

Another possibility we propose to investigate is the use of the HSLS connectivity table on the victim node to attempt to reconstruct the forwarding path from the originator of the packet. While this approach might not be completely accurate (due to changes in routing further away from the victim), we anticipate that over time the node responsible for the damage is correctly identified (via loss of reputation).

8.2 Forgiveness policies

Forgiveness is especially necessary when dealing with damage done along a path, because BITSI cannot directly determine which node was the culprit, and so assigns negative points to all nodes along the path. BITSI, unlike the watchdog/pathrater [16] and many reputation systems that are based on it (e.g. [12]), does not use eavesdropping and promiscuous mode. This decision was made because the watchdog plan of listening to neighbors to determine if they are forwarding appropriately depends on bidirectional transmission, which may consume power unnecessarily, and on equivalent communication range between all nodes, which may not be a realistic assumption.

In the work presented in this paper, each node is regularly forgiven by a small set amount. Using this policy, damaging nodes can be expected to cycle in and out of paths as their reputation is recovered. Clearly, this is not ideal and new work will focus on evaluating new forgiveness strategies.

One alternative is to forgive the nodes that have lower reputations less than others. The lower a node's reputation is, the less it is forgiven. However, this may punish central or bridge nodes that do a lot of forwarding and so are on a lot of paths.[7] A more promising possibility is to factor in the proportion of nodes that have a reduced opinion, so that nodes that are marked down by most of the nodes for which they have worked should be forgiven less readily. This approach requires the node to keep track of other nodes for which it forwards packets.

8.3 BITSI

The work presented in this paper is part of a larger framework aimed to enable mission continuity in the presence of adversaries in the network. The BITSI framework will ultimately be able to cluster nodes in the network based on their behavior and state of software (compared to a known base installation base). This ability will allow us to efficiently address the issues that apply to this research, but were not addressed in this paper. Such scenarios include the case when the packets originated by a node represent damage (as opposed to being corrupted during forwarding) and when the application reporting damage to BITSI is not honest.

REFERENCES

[1] Abusalah, L., Khokhar, A., BenBrahim, G. and ElHajj, W., "TARP: Trust-aware routing protocol", Proc. ACM IWCMC'06, 135-140 (2006)

[2] Balakrishnan, V., Varadharajan, V., Lucs, P. and Tupakula, U. K., "Trust enhanced secure mobile ad-hoc network routing", Proc. IEEE AINAW'07, 27-33 (2007)

[3] Buchegger, S. and Boudec, J.-Y. L., "The effect of rumor spreading in reputation systems for mobile ad-hoc networks", Proc. WiOpt'03, (2003)

[4] Dasgupta, D. "Immuno-inspired autonomic system for cyber defense", Proc. Inf. Secur. Tech. Rep., vol 12, 4, 235-241 (2007)

[5] Ford, R., Carvalho, M. and Allen, W., "BITSI: A biologically-inspired adaptive defense framework", Proc. Adaptive and Resilient Computer Security Workshop (2007)

[6] Forrest, S., Hofmeyr, S., Somayaji, A. and Longstaff, T., "A Sense of self for Unix processes", Proc. IEEE Research in Security and Privacy, 120-18 (1996)

[7] Greensmith, J., Aickelin, U. and Twycross, J., "Articulation and clarification of the dendritic cell algorithm", Proc. ICARIS, 404-417 (2006)

[8] Huang, Y. and Lee W., "Hotspot-based traceback for mobile ad hoc networks", Proc. ACM WiSe'05, 43-54 (2005)

[9] Kargl, F., Klenk, A., Schlott, S. and Weber, M., "Advanced detection of selfish or malicious nodes in ad hoc networks", Proc. ESAS, 152-165 (2004)

[10] Khanna, R. and Huaping, L., "System approach to intrusion detection using hidden Markov model", Proc. IWCMC, 349-354 (2006)

[7] Although this can have a possibly desirable effect of pushing traffic from overwhelmed central nodes towards the less used peripheral ones.

[11] Kim, Y., Sankhla, V. and Helmy A. "Efficient traceback of DoS attacks using small worlds in MANET", Proc. IEEE VTC, vol 6, 3979-3983 (2004)

[12] Lavecchia, C., Michiardi, P and Molva, R, "Real life experience of Cooperation Enforcement Based on Reputation (CORE) for MANETs", Proc. IEEE REALMAN, (2005)

[13] Lee, K. J., Kim, M. S., Cho, S. Y. and Mun, B. I., "Delay-centric link quality aware OLSR", Proc. IEEE LCN'05, 690-696 (2005)

[14] Leguay, J., Conan V. and Friedman, T., "QoS routing in OLSR with several classes of service", Proc. IEEE PERCOMW'06, (2006)

[15] Leung, K., France, C. and Christopher, C., "Generating compact classifier systems using a simple artificial immune system", IEEE Trans. Systems, vol 37, 5, 1344-1356 (2007)

[16] Marti, S., Giuli, T. J., Lai, K. and Baker, M., "Mitigating routing misbehavior in mobile ad hoc networks", Proc. ACM MobiComm'00, 255-265 (2000)

[17] Matzinger, P., "Tolerance, danger and the extended family", Annu. Rev. Immunol., vol 12, 991-1045, (1994)

[18] Mundinger, J. and Boudec, J.-Y. L., "Analysis of a reputation system for mobile ad-hoc networks with liars", Journ. Perform. Eval., vol 6., 3-4, 212-226 (2008)

[19] The Network Simulator, http://www.isi.edu/nsnam/ns/, Page accessed on 01/26/2009.

[20] Ondi, A., Ford, R., Allen, W., Marin, G., Carvalho, M. and Perez, C., "MANET simulation and security: New wineskin for new wine?", Proc. ACM SE'08, (2008)

[21] Santiváñez, C. A. and Ramanathan, R., "Hazy-sighted link state (HSLS) routing: A scalable link state algorithm", BBN technical memo BBN-TM-I30I, (2001)

[22] Sarafijanović, S. and Boudec, J.-Y. L., "An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal and memory detectors", Proc. ICARIS, 342-356 (2004)

[23] Strayer, W. T., Jones, C. E., Tchakountio, F. and Hain, R. R., "SPIE-IPv6: Single IPv6 packet traceback", Proc. IEEE LCN'04, 118-125 (2004)

[24] Theodorakopoulos, G. and Baras, J. S., "Trust evaluation in ad-hoc networks", Proc. ACM WiSe'04, 1-10 (2004)