Zone-Based Intrusion Detection for Mobile Ad Hoc Networks*

Bo Sun
Dept. of Computer Science
Texas A&M University
College Station TX 77843-3112
b0s6067@cs.tamu.edu

Kui Wu
Dept. of Computer Science
University of Victoria
BC, Canada V8W 3P6
wkui@cs.uvic.ca

Udo W. Pooch
Dept. of Computer Science
Texas A&M University
College Station TX 77843-3112
pooch@cs.tamu.edu

ABSTRACT Intrusion Detection Systems (IDSs) for Mobile Ad hoc NETworks (MANETs) are indispensable since traditional intrusion prevention based techniques are not strong enough to protect MANETs. However, the dynamic environment of MANETs makes the design and implementation of IDSs a very challenging task. In this paper, we present a non-overlapping Zone-Based Intrusion Detection System (ZBIDS) that fits the requirement of MANETs. On the local detection part, we present a general intrusion detection agent model and propose a Markov Chain based anomaly detection algorithm. We focus on the protection of MANET routing protocols and present the details regarding feature selection, data collection, data preprocess, Markov Chain construction, classifier construction and parameter tuning. We demonstrate that local detection alone cannot achieve desirable performance. Therefore, we further propose a collaboration mechanism among ZBIDS agents and an aggregation algorithm used by gateway nodes. With alert information from a wider area, gateway nodes' IDS can effectively suppress many falsified alerts and provide more diagnostic information about the occurring attacks. Security officers can have a general understanding about the attacks using the proposed MANET Intrusion Detection Message Exchange Format (MIDMEF). We carry out extensive simulation to evaluate the performance of ZBIDS at different mobility levels. Sim-

This paper is the extension of our conference papers [3,4].

ulation results show that ZBIDS can achieve desirable performance and meet the security requirement of MANETs.

I. Introduction

The unique characteristics of Mobile Ad hoc NETworks (MANETs), such as arbitrary node movement and lack of centralized control, make them vulnerable to a wide variety of outside and inside attacks [1]. How to provide effective security protection for MANETs has become one of the main challenges in deploying MANET in reality. Intrusion prevention techniques, such as encryption and authentication, can deter attackers from malicious behavior. But prevention based techniques alone cannot totally eliminate intrusions. The security research in the Internet demonstrates that sooner or later a smart and determined attacker can exploit some security holes to break into a system no matter how many intrusion prevention measures are deployed. Therefore, intrusion detection systems (IDSs), serving as the second line of defense, are indispensable for a reliable system. IDSs for MANETs can complement and integrate with existing MANET intrusion prevention methods to provide highly survivable networks [1].

Nevertheless, it is very difficult to design a once-for-all detection model. Instead, an incremental enhancement strategy may be more feasible. A desirable detection model should at least include mechanisms against known attack types. In addition, it should provide a scheme to easily add new security features in the future. Based on this consideration, we target at a specific type of attack, *routing disruption attack*, as the preliminary step toward the goal of providing comprehensive security protection.

This paper has made four contributions in the IDS research for MANETs. First, it proposes a local detection engine. Because of the distributed nature of MANETs, it is desirable that an Intrusion Detection agent is attached to each node. Each IDS agent performs the intrusion detection task *locally* and *independently*. It uses *trusted* information to monitor the node's local activities for abnormal behavior. Regarding the local detection engine, we present the details of feature selection, data collection, data preprocess, Markov Chain (MC) construction, classifier construction, and performance tuning. Second, we present a framework for the collaboration of distributed detection agents. The distributed local IDS agents without collaboration

are likely to have high false positive rates under the dynamic MANET environment. Therefore, a suitable framework is required to facilitate the agents' cooperation in order to improve the performance. For such purpose, a two-level nonoverlapping Zone-Based Intrusion Detection System (ZBIDS) is proposed to meet the unique requirement of MANETs. With ZBIDS, the network is divided into nonoverlapping zones and each IDS agent broadcasts the locally generated *alerts* inside the zone. Gateway nodes (also called interzone nodes, which have physical connections to nodes in different zones) are responsible for the aggregation and correlation of locally generated alerts. Third, an algorithm is proposed to utilize the attribute similarity to aggregate locally generated alerts. With such alert aggregation, the IDS performance in terms of false positive rate and detection rate can be improved. Finally, to facilitate the interoperability of distributed detection agents, we present an alert format - MANET Intrusion Detection Message Exchange Format (MIDMEF), which conforms to IDMEF [20] standard.

The rest of the paper is organized as follows. In Section II, we illustrate the attack model. Section III describes our assumptions in constructing ZBIDS. Section IV outlines ZBIDS and describes the internal model of the intrusion detection agent. Section V presents a Markov Chain based anomaly detection algorithm. Section VI presents ZBIDS and an aggregation algorithm. Section VII provides simulation results. Section VIII introduces related work in wired and wireless IDS, and Section IX concludes this paper.

II. Threat Model

Routing protocols are the cornerstone of MANETs since they support the basic message forwarding service to MANET applications. Because of this, they are often the targets of various attacks. Therefore, in this paper, we focus on the protection of MANET routing protocols. To be specific, we aimed at defending against one of the most important active attacks: *routing disruption* attack.

In MANETs, it is very easy for the attacker to perform *routing disruption* attacks. For example, the attacker can generate randomly-constructed routing control packets (e.g., falsified Routing REPly (RREP) in Dynamic Source Routing protocol) and disseminate them into the network. Such falsified routing information can prevent the source node from establishing a correct path and can also disrupt the routing logic in the

whole network.

Because of the arbitrary movement of nodes, link breakages happen frequently. Therefore, it is difficult for the attacker to target one specific victim all the time. This will lead to the phenomenon called “partial” victims because the victim will not receive all the falsified routing control packets. The degree of *partial* depends on mobility levels, mobility patterns and locations of mobiles, etc. On one hand, this phenomenon alleviates the damage caused by an attacker to a specific victim. On the other hand, it makes intrusion detection more difficult since it obscures the difference between real attacks and normal link breakages.

III. Assumptions

First, we assume the local IDS agent is secure. Current technologies in resistant software/hardware [6] make it very hard to crack the embedded secrets. Therefore, we do not need to consider the security issues of IDS agents themselves. Actually, security of IDS agents presents another challenge for MANETs and is beyond the scope of this paper. In the case that tamper resistant techniques are difficult to provide, the compromised IDS agents do not intend to send error alert information. Otherwise, this kind of activeness would make them easy to be detected. We further assume the information exchanged among IDS agents cannot be compromised. This could guarantee the correct execution of the aggregation algorithm.

Second, in the context of intrusion detection, we assume that normal and abnormal behaviors have distinct manifestations. Actually, in MANETs, because of their dynamic nature, it is quite often for a normal node to send out falsified routing control packets. For example, because of mobility-induced errors, when a node generates routing control packets reacting to some routing information request, this information may be obsolete because of delay in message transmission. Therefore, we assume that the attacker sends out many routing falsified routing control packets in order to *effectively* disrupt the routing control logic. This is a valid assumption under intrusion detection. Moreover, it has been demonstrated that mobility could be used to develop suitable mechanisms to enhance security [5]. Based on this, one or few routing control packets could hardly incur severe damage to the whole system.

Third, we assume that attackers use their own address to initiate the attack. With efficient neighbor monitor mechanisms, the attacker could be easily identified if he changes his address dynamically.

IV. Zone-Based Intrusion Detection System

A. ZBIDS Framework

It is obvious that local detection alone cannot guarantee satisfactory performance because of limited security information obtained by each IDS agent. What's more, we may experience *alert flooding* problems given the distributed nature of MANETs. Therefore, a suitable framework is needed to integrate the alert information from a wider area. Moreover, attacks are likely to generate multiple related alerts. For example, because of the broadcast nature of radio channel, there may exist many victims suffering from same falsified routing control packets. The triggered alerts should have high correlations correspondingly. Therefore, it is desirable to treat them together.

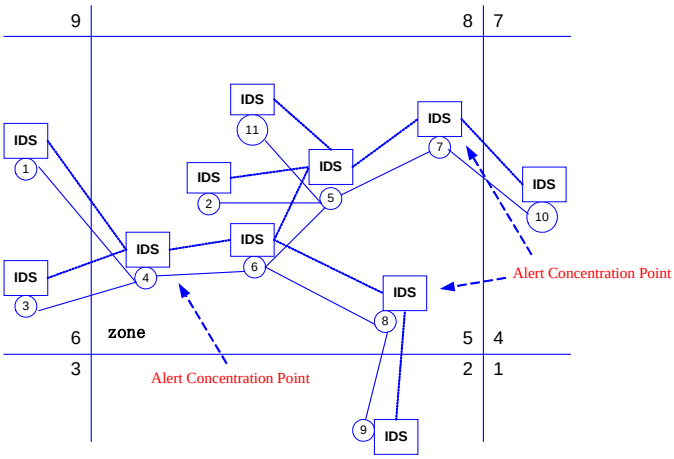


Fig. 1. The Zone-Based Intrusion Detection System (ZBIDS) for MANETs.

Based on the above considerations, we adopt a nonoverlapping zone based framework. The whole network is divided into nonoverlapping zones. The formation and maintenance of zones are beyond the focus of this paper. We assume the existence of such a framework. This could be done easily based on techniques like geographic partitioning[8]. As illustrated in Fig. 1, there are two categories of nodes in ZBIDS: *intrazone* nodes and *gateway* nodes (also called *interzone* nodes). If one node has a physical connection to a node in a

different zone, this node is called a gateway node, for example, node $4, 7, 8$ in Fig. 1. Otherwise, it is called an intrazone node.

Only gateway nodes can generate *alarms*. They collect the local *alerts* broadcast from the intrazone nodes and perform aggregation and correlation tasks to suppress many falsified *alerts*. There may exist more than one gateway node in a single zone, all of which perform the alert aggregation task simultaneously. In this way, we can avoid the single point of failure. Note that in this paper, alerts and alarms have different meaning. Alerts indicate possible attacks and are generated by local IDS agents, while alarms indicate the final detection decision and can be generated only by gateway nodes.

B. Internal Model of IDS Agent

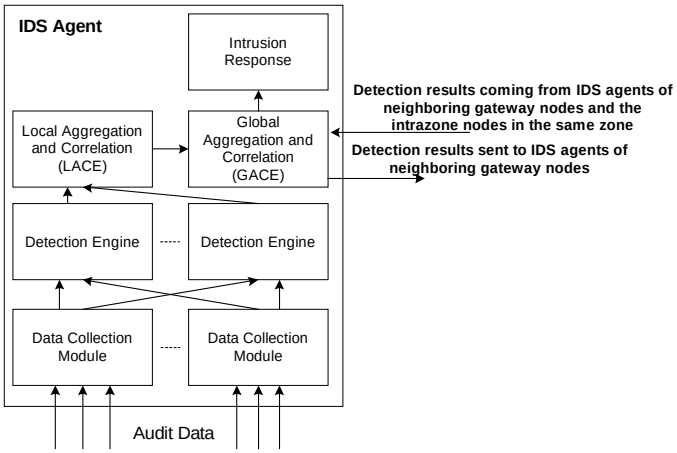


Fig. 2. Diagram of an IDS Agent.

Fig. 2 illustrates the internal diagram of an IDS agent. In the IDS agent, the data collection module collects security related information from nodes within wireless communication range and preprocesses the information to the input format of detection engines. Different local detection techniques can be deployed in detection engines and each of them could detect a class of attacks. We do not expect that a single technique could detect all possible intrusions. Instead, the deployed detection techniques should cooperate with each other to improve the overall performance. The functionality of Local Aggregation and Correlation Engine (LACE) is to locally aggregate and correlate the detection results of detection engines. Global Aggregation

and Correlation Engine (GACE) in *gateway* nodes is to aggregate and correlate the detection results from local nodes in order to make final decisions. They can also cooperate with neighboring gateway nodes to further exchange information. For an *intrazone* node, its GACE is to locally broadcast the alert information inside the zone. After an attack is identified, based on different attack types, the Intrusion Response Module (IRM) could take corresponding measures, such as identifying the intruders, reinitiating the communication channels, and excluding the compromised nodes from the networks.

V. Anomaly Detection in MANETs

A. Feature Selection

We need to select effective features in order to construct good IDS models. Features are security related measures which could be used to reflect subject activities. In this paper, because we focus on the protection of MANET routing protocols, we need to select features which can reflect MANET routing activities. Due to the unique characteristics of MANETs, each node acts as a router and each node maintains a routing table in order to relay messages for other nodes. For a realistic application, the users' mobility pattern often has some regularity. For example, each user may have a destination in mind and follows his favorite routes. Therefore, the routing table of each user may follow a regular change pattern. The legitimate changes of routing caches are *reliable* and can be *locally* collected. Therefore, like in [1], we define the normal updates of node's routing table as the normal profile.

For each feature, we further measure the *relative entropy* [11] between training data and test data (denoted as RE_{test} henceforth) and the *relative entropy* between training data and intrusion data (denoted as $RE_{intrusion}$ henceforth) at each mobility level in order to demonstrate the effectiveness of the selected features. *Relative entropy* is a measure of the "distance" between two probability mass functions [11]. For anomaly detection, in order to achieve high performance, it is better that RE_{test} is small (indicating the similar behavior between training data and test data) while $RE_{intrusion}$ is large (indicating significant difference between training data and intrusion data) [19].

For test data and intrusion data, we use the limited sample size to reflect recent subject activities. For a sample of size n, if the number of occurrence of an item c is n_c, the occurrence probability of c is calculated as n_c/n. Because of the different amounts of training data and test data (or intrusion data) items, it is possible that some item appearing in training data does not appear in test data. Therefore, we cannot calculate the divergence of the two probability distributions because the denominator is 0 when calculating relative entropy. To cope with this *zero-frequency* problem, we adopt the popular *Jelinek-Mercer* smoothing method [12]. That is, for item c, if its probability is 0 in test data (or intrusion data), we use the following formula to modify its probability:

$$p_\lambda(c) = \lambda q(c) + (1-\lambda)p(c), \quad 0 \leq \lambda \leq 1.$$

where $q(c)$ is the probability of c in test (or intrusion) data for a given history, and $p(c)$ is the probability of c calculated from training data. λ is an interpolation coefficient which is used as a balancing weight between the observed probability in test data (or intrusion data) and the probability calculated from training data.

Based on our experiment results, we use the following features that are sensitive to *routing disruption* attacks: *PCR* - percentage of the change in route entries, and *PCH* - percentage of the change in number of hops. *PCR* represents the deleted and increased routing entries in a certain time period. *PCH* indicates the changes of the sum of hops of all routing entries in a certain time period.

B. Data Preprocess

We utilize *Vector Quantization* (VQ) [10] to discretize the raw continuous data. VQ algorithm could be used to discretize raw data while minimizing the average distortion. Based on the *nearest neighbor* condition and *centroid* condition, VQ could be used to represent raw data accurately. We use the commonly used *distortion* measure - *squared-error* measure as the distortion distance. In practice, people usually resort to the suboptimal VQ algorithm - the Linde-Buzo-Gray (LBG) algorithm [10] for easy calculation of the codebook. After working out the codebook, each raw item is mapped to some codevector to which it has the smallest Euclidean distance.

C. Markov Chain Based Intrusion Detection

It is possible that some discretized values have very small probabilities. This kind of values does not represent the normal routing activities. Therefore, for data items whose probability is below some *threshold*, we convert them to a common "rare" symbol.

We construct a Markov Chain from the discretized routing table changes. Using the immediate previous w consecutive events (the routing table changes), also called the $from\_state$, we predict the transition probability of the next state, to_state. To do so, we use a window of size w sliding through the discretized training data, each time by one position. During the scanning process, for each *from_state*, we also record its transition times to different *to_states* respectively.

To speed up the search process, we can use a Hash table H to store the $transition(from\_state, to\_state)$. When a $from\_state$ is not found in H, we insert it into H and associate it with a counter 1. Otherwise, the associated counter increases by 1. If the $transition(from\_state, to\_state)$ is not in H, it is inserted into H and associated with a counter 1. Otherwise, its associated counter increases by 1.

The probability of the $transition(s_1, s_2)$ is then calculated as: $P(s_1, s_2) = N(s_1, s_2)/N(s_1)$. Here $N(s_1, s_2)$ is the number associated with the transition (s_1, s_2). $N(s_1)$ is the total number associated with the $from\_states_1$. Based on our calculation, the higher the $P(s_1, s_2)$, the more likely the transition is normal. The whole process to construct the MC is illustrated in Fig. 3(a).

D. Classifier Construction

An IDS is essentially a classifier. We utilize the transition probability aggregated over the recent past activities to construct a classifier. After initializing two real numbers A and B to zero, we shift the window with size w in the *locality frame* to search for $from\_state$ and to_state, each time by one position. If the $transition(from\_state, to\_state)$ is found in MC, A increases by 1, and B increases by $F = 1 - P(from\_state, to\_state)$, where $P(from\_state, to\_state)$ is obtained from the MC. Therefore, F sums up all the transition probabilities from the $from\_state$ that are not equal to $(from\_state, to\_state)$. If the transition is not found in the MC, A increases by 1, and B increases by some predefined *penalized value*.

```
Procedure Construct_Markov()

Input   D: raw data set;
Output  H: hash table that stores the states and the associated counters;

Begin

Use VQ algorithm to preprocess D;
Generate rare symbols, and convert D to a set of traces, denoted as M;

For each ψ ∈ M
{
    from_state is set to the first w symbols of ψ;
    shift ψ left by w positions;

    While (not reaching the end of ψ)
    {
        to_state is set to the first symbol in ψ;
        shift ψ left by one position;

        If (from_state ∉ H)
        {
            from_state → H;
            the counter of from_state in H is set to 1;
        }
        Else
            increase the counter of from_state by 1;

        If (transition(from_state, to_state) ∉ H)
        {
            (from_state, to_state) → H;

            the counter of (from_state, to_state) is set to 1;
        }
        Else
            increase the counter of (from_state, to_state) by 1.

        shift from_state left by one position;
        append to_state to the end of from_state;

    } /*for While*/
} /*for For*/

End.
```

(a) Markov Chain Model Construction.

```
Procedure Markov_to_Classifier(MC, ζ)

Input:    MC: constructed markov chain;
          ζ: a trace;
Output:   Normal or Anomalous;

Begin

i = 1; A = 0; B = 0; μ (ζ) = 0;

While ( i < the length of ζ)
{
    from_state is set to sequence δ[i] = (ζ[i], ζ[i+1], …ζ[w+i-1]).
    to_state is set to ζ[w+i];
    If (transition(from_state, to_state) is in MC)
    {
      A = A + Σ_{s∈Next(from_state)} P(from_state, s)   = A + 1;
      /*Next(from_state) indicates all the to_state associated with the current
        from_state*/
      B = B + Σ_{s∈Next(from_state)∧(s≠to_state)} P(from_state, s)
        = B + 1 - P(from_state, to_state)
    }
    else
    /* (from_state, to_state) is not the transition of MC */
    {
      A = A + 1;
      B = B + z;
    }

    Adjust A and B over the past locality frame;

    i++;
    μ(ζ) = B / A;
    If (μ(ζ) >= r)
        Return Anomalous;

} /*for While*/

Return Normal;

End.
```

(b) Classifier Construction.

Fig. 3. Pseudocode to Construct the Markov Chain Model and the Classifier.

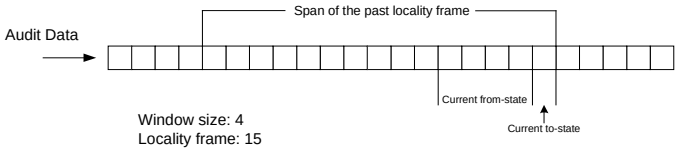


Fig. 4. Relation among from_state, to_state, window size, and locality frame.

In this way, B/A represents the *average distance* between the current trace and the MC. It measures how well the MC predicts the current activities. A lower B/A indicates that the current activity is more likely to be normal. If B/A is larger than a measure r, the *alert threshold*, this trace is identified as abnormal. Because B/A is calculated over the *locality frame*, it could suppress unexpected sudden changes and thus reduce the false positive rate.

The relationship among the $from\_state$, to_state, window size w, and the *locality frame* is illustrated in Fig. 4. The algorithm used to construct the classifier is illustrated in Fig. 3(b).

E. Parameter Tuning

a) Window size w: w is used to measure the accuracy of MC. As w increases, the algorithm depicted in Fig. 3(b) constructs a better model because it considers more historical data. However, the classical overfitting problem will appear for a very large w because the constructed MC models the training data "too well". Considering this trade-off, we utilize *conditional entropy* (*CE*) [11] to tune w. *CE* could be used to explore the temporal and sequential characteristics of audit data and determine the uncertainty for the current data item given the previous w items. We measure *CE* at different mobility levels and choose w when *CE* calculated with the w does not drop dramatically.

b) Penalized value z and *alert threshold r:* For a trace $\alpha = \{X_1, X_2, \ldots\}$, let $\mu(\alpha) = B/A$. Intuitively, it measures the *distance* between the MC and the current activities. A smaller $\mu(\alpha)$ indicates that the current activity is likely to be normal.

The discrepancy $D_t(\alpha)$ over the *locality frame* with length L is defined as:

$$D_t(\alpha) = \sum_{i=L}^{|\alpha|} \mu_i(\alpha)/(|\alpha| - L + 1) \tag{1}$$

where $\mu_i(\alpha)$ is the average μ over the locality frame $\{X_{i+1-L}, X_{i+2-L}, \ldots, X_i\}, \quad i \geq L$.

For a given normal trace set T_t, its discrepancy $D_t(T_t)$ is:

$$D_t(T_t) = \sum_{\alpha \subseteq T_t} D_t(\alpha)/|T_t| \tag{2}$$

where $|T_t|$ denotes the number of traces in T_t.

For intrusive activities, we define the discrepancy $D_a(\alpha)$ over the locality frame with length L as:

$$D_a(\alpha) = Max(\mu_L(\alpha), \mu_{L+1}(\alpha), \ldots, \mu_{|\alpha|}(\alpha)). \tag{3}$$

$\mu_i(\alpha)$ has the same meaning as before. We use Max here because it is possible that normal data is mixed together with abnormal data if the attacks are performed intermittently.

For a given trace set T_a of intrusive activities, its discrepancy $D_a(T_a)$ is:

$$D_a(T_a) = \sum_{\alpha \subseteq T_a} D_a(\alpha)/|T_a| \tag{4}$$

where $|T_a|$ denotes the number of traces in T_a.

We tune z until $(D_a(T_a) - D_t(T_t))$ is above a certain value so that we can easily distinguish a normal trace from a trace with intrusive behaviors.

Alert threshold r is used to make a balance between the false positive rate and the detection rate. A smaller value of r will lead to quicker detection of the intrusion. However, it may lead to a high false positive rate. A larger value of r will suppress more false positives. But it may miss the detection of some intrusions. We set r to be a weighted sum of $D_a(T_a)$ and $D_t(T_t)$ as:

$$r = h_a * D_a(T_a) + h_t * D_t(T_t), \quad h_a + h_t = 1, h_a > 0, h_t > 0$$

VI. Alert Aggregation

Local IDSs usually have high false positive rate and are subject to *alert flooding*, because local IDSs alone cannot collect enough security related information. To solve this problem, the internal relationship among the generated alerts should be taken into consideration. As such, we present an alert aggregation algorithm in this section.

A. Collaboration Mechanism

When no local alert is generated by a local IDS agent, this IDS agent will take no actions. When a local alert is generated, the local IDS agent simply broadcasts the alert inside the zone. To save bandwidth, we do not use global broadcast. In this way, the *gateway* nodes can receive necessary security-related information. If gateway nodes receive the locally broadcast alerts in some period, they need to perform alert aggregation. Furthermore, neighboring gateway nodes should exchange aggregated information with each other in order to get information from a wider area. In ZBIDS, only gateway nodes could generate alarms. Local IDSs attached to local nodes could only generate alerts based on their local information and propagate these alerts

inside the zone. Note that as we have mentioned before, alerts and alarms have different meanings: alarms are final detection decision made by gateway nodes, while alerts indicates potential attacks raised from local IDS agents.

B. Aggregation Mechanism

1) Class Hierarchy of the Alerts: A suitable alert format is required for the collaboration among IDS agents. It could also facilitate better interoperability of future IDS systems. Intrusion Detection Working Group (IDWG) has proposed the Intrusion Detection Message Exchange Format (IDMEF) [20] - an alert format which is suitable for wired IDSs. We modified it to fit the requirement of MANETs. The modification includes adding some new classes (e.g., Zone class) and attributes related to MANETs, deleting some unwanted classes (e.g., User class, Process class) and attributes, and modifying the definition of some classes and attributes (e.g., Location attribute). The alert class hierarchy for ZBIDS - $MIDMEF$ (MANET Intrusion Detection Message Exchange Format) is depicted in Fig. 5 using the UML notation.

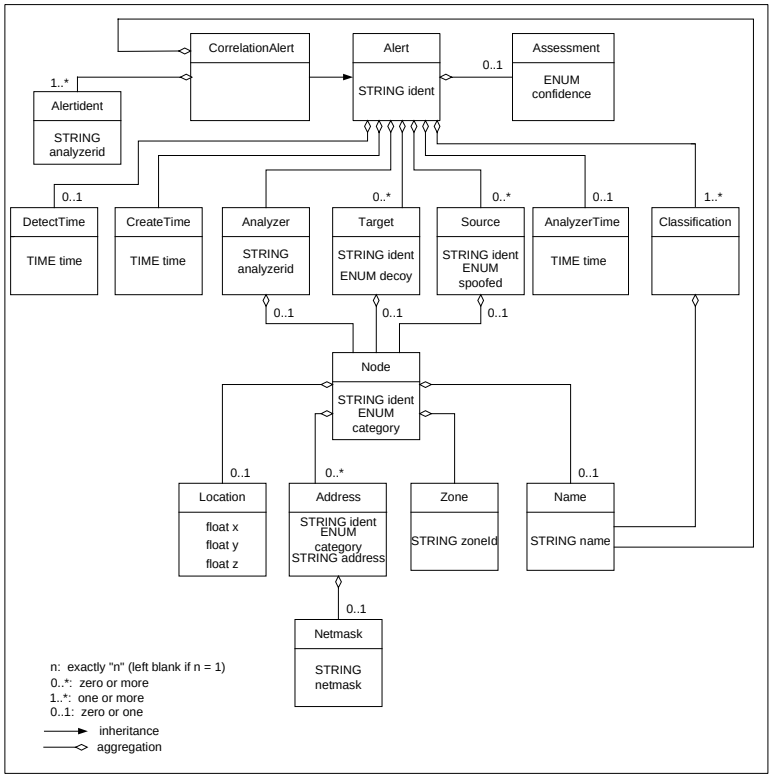


Fig. 5. MANET Intrusion Detection Message Exchange Format (MIDMEF).

Procedure Determine_P()
Input: test trace, attack trace, $P_{routing\_abnormal}$
Output: P
Begin
Test trace:
For each gateway node G
  $P_{Gt} = 0$;
  For each time interval of G that receives local alerts
    For all P_{ti}
      If $P_{ti} > P_{routing\_abnormal}$
      then $P_{Gt} = P_{Gt} + P_{ti}$
    End For
  End For
  /* m_t is the number of time intervals of G that receives local alerts */
  $P_{test\_sum} = P_{test\_sum} + P_{Gt}/ m_t$
End For
/* N_{test} is the number of gateway nodes that receive local alerts */

$$P_{test} = \frac{P_{test\_sum}}{N_{test}}$$

Attack trace:
For each gateway node G
  For each time interval of G that receives local alerts
    Compute the sum of the probability of attacker source addresses P_{Ga}
  End For
  /* m_a is the number of time intervals of G that receives local alerts */
  $P_{attack\_sum} = P_{Ga} / m_a$
End For
/* N_{attack} is the number of gateway nodes that receive local alerts */

$$P_{attack} = \frac{P_{attack\_sum}}{N_{attack}}$$

$P = h_t * P_{test} + h_a * P_{attack.}$
END

Fig. 6. Pseudocode to Decide P.

2) Aggregation Algorithm: Aggregation algorithms are based heavily on local IDSs. Existing aggregation algorithms usually assume the accurate information provided by local IDSs, and their execution is triggered by the receipt of local alerts [28], [29], [30]. Our case is different in that our aggregation algorithm is executed *periodically* to reduce the computation load of gateway nodes. If there are no local alerts received in the past period, no action is taken by gateway nodes. Also, our aggregation algorithm does not assume accurate information provided by local IDSs.

The gateway node mainly utilizes the following information:

- *Classification* similarity: It indicates the name of the attack and is provided by the *classification* field of MIDMEF. In our context, it should be "$Routing\_Disruption$".
- *Time* similarity: The entity *DetectTime* and *CreateTime* of MIDMEF could be used to provide time information. *DetectTime* indicates the time when the attack happens, and *CreateTime* indicates the time when the attack is detected. If the temporal difference between the *CreateTime* of a newly received local alert and the current time of the gateway node exceeds some predefined delay, this local alert is ignored.
- *Source* similarity: It indicates the possible attack sources.

Our aggregation algorithm mainly utilizes *source* similarity. During the normal routing discovery phase, the node which receives a RREQ (Routing REQuest) packet will reply with a RREP (Routing REPly) packet if it has a valid path to the destination. Given a random network topology, the distribution of the source addresses within these RREP packets during a given time period would be even. That is, most of the time, there is no strong bias for a given source address.

Nevertheless, when attacks happen, the attacker tends to send many falsified RREPs into the network. Therefore, the attacker's address is likely to dominate the source address distribution. The MC based local IDS records the routing control packets and their aggregated probability distribution is locally broadcast with the local alert specified using $MIDMEF$. The gateway nodes aggregate the locally broadcast information and compute the source address distribution over the past period. In this way, we can differentiate the attacker from the normal nodes: if the probability from some particular node exceeds a predefined threshold P, this address is then identified as an attacker. Note that an attacker cannot frequently change its IP address to send out fake messages. Otherwise, it can be detected easily by its neighbors.

A proper value P is of vital importance to the performance of our aggregation algorithm. In theory, P depends on attack intensity, attacking time, network topology, etc. If the value of P is set too low, the gateway nodes could identify the attack more quickly and can achieve a high detection rate. However, a small P value will result in a high false positive rate. In contrast, if the value of P is set too high, the

gateway nodes may have a low false positive rate. But the detection rate will be reduced as well. Hence, we set P in the following way.

In normal cases, for a given gateway node, if local alerts are received in a given time period, we first pick those source addresses whose aggregated probability is larger than $P_{routing\_abnormal}$. We denote these probabilities as P_{t_i} ($i = 1, 2, \ldots, n_t$).

Suppose that a given gateway node G has m_t time periods in which it receives local alerts. We compute the average of $P_{t_i}, (i = 1, 2, \ldots, n_t)$ over these m_t periods as:

$$P_{G_t} = \sum_{j=1}^{m_t} \sum_{i=1}^{n_t} P_{t_i}/m_t.$$

P_{G_t} represents the irregularity of source address distributions of routing control packets when the system is at normal status. Given test traces, we compute its average over all gateway nodes:

$$P_{test} = \sum_{\forall\ gateway\ nodes} \frac{P_{G_t}}{the\ number\ of\ gateway\ nodes}.$$

Given the trace of intrusive activities, we first compute the attack address distributions contained in the routing control packets. We denote these probabilities as P_{a_i} ($i = 1, 2, \ldots, n_a$).

Suppose that a given gateway node G has m_a time periods in which it receives local alerts. We compute the average of $P_{a_i}, (i = 1, 2, \ldots, n_a)$ over these m_a periods as:

$$P_{G_a} = \sum_{j=1}^{m_a} \sum_{i=1}^{n_a} P_{a_i}/m_a.$$

P_{G_a} represents the source address distribution of the routing control packets in the gateway node G during the attack time. Given traces of intrusive activities, we compute the average of P_{G_a} over all gateway nodes:

$$P_{attack} = \sum_{\forall\ gateway\ nodes} \frac{P_{G_a}}{the\ number\ of\ gateway\ nodes}.$$

We set P as:

$$P = h_t * P_{test} + h_a * P_{attack}, \quad h_t > 0, h_a > 0, h_t + h_a = 1.$$

For the aggregation algorithm, if a gateway node does not receive any alert during one period, it will take no actions. Otherwise, it first sums up the aggregated probabilities of those source addresses whose probability is larger than $P_{routing\_abnormal}$. If the resultant value is less than P, the gateway node will not generate alarms. Otherwise, the gateway node will generate alarms and provide attacker information based on the probability distribution of source addresses. The whole process if illustrated in Fig. 7.

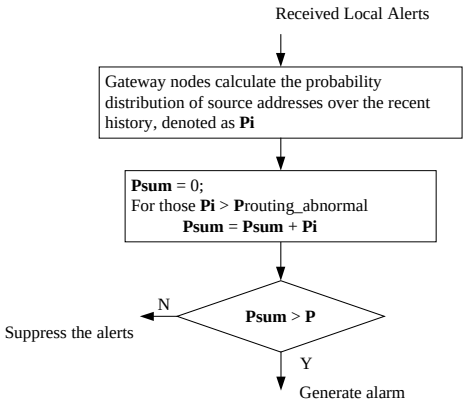


Fig. 7. Aggregation Algorithm.

It is possible that the detection sensitivity of our aggregation algorithm could decrease with the increase of the number of attackers. This is because when there are more attackers in the network, no single attacker's address would dominate the address distribution. Therefore, it would be difficult for the aggregation algorithm to make correct decisions. This situation is worsened when two or more attackers collude to attack the same objective at the same time. However, it is still possible that the victims do not overlap completely, whose addresses could contribute to the effectiveness of our aggregation algorithm.

VII. Simulation Study

A. Simulation Model

1) Simulation Platform and Parameter Settings: We use a simulation model based on GloMoSim [9]. The parameters used in the simulation are described in Table I. When we simulate a routing disruption

attack, the attacker is uniformly chosen from the simulated nodes.

2) Data Sets: Three types of data need to be generated: training data, testing data, and anomaly data. We use different pause time to represent different mobility scenarios. At each mobility level, we select 4 different random seeds. For a given mobility scenario and a given random seed, we run the simulation 400 minutes, and collect the normal data from all nodes to generate a normal data trace. From each normal data trace, we use its last 40 minutes part as testing data. The rest of the normal data are used as training data. We simulate the route disruption attack in order to collect anomaly data. To evaluate the performance of IDS, we further generate a different set of normal and abnormal data. This procedure is illustrated in Fig. 8.

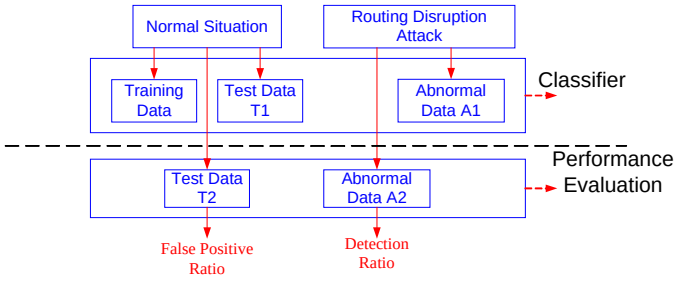


Fig. 8. Data Set used to Construct Classifier and Evaluate Performance.

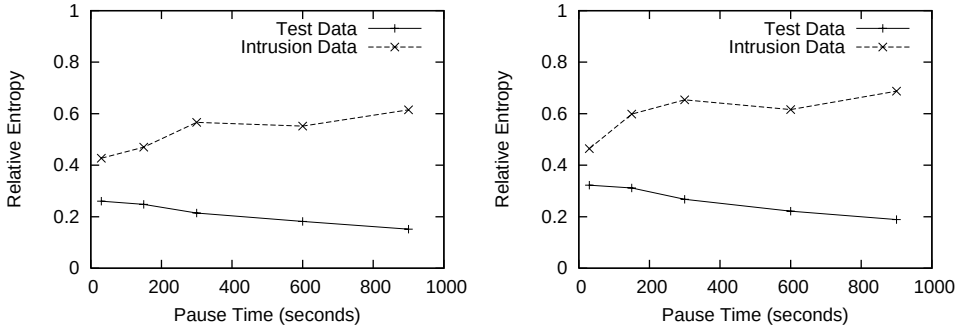
3) Performance Metrics:

- **False positive rate**: For local IDS, it is defined as the percentage of decisions in which normal data are flagged as anomaly. We present the average result of 10 runs as well as the confidence interval of 95%. For ZBIDS, it is computed as the ratio of the total number of false alarms over the total number of decisions made by gateway nodes.
- **Detection rate**: For local IDS, it is computed as the ratio of the total number of victims that correctly detect attacks over the total number of victims in the anomalous data. For ZBIDS, detection rate is measured from dividing the total number of gateway nodes that actually generate alarms by the number of gateway nodes that should generate alarms.
- **MTFA** (Mean Time of First Alarm): It is defined over anomalous traces and measures how fast the classifier detects the attack. Given an anomalous trace ξ, if we suppose the attack start location is L_a and

our local IDS generates its first alert after scanning the L_d-th symbol, then the $MTFA$ corresponding to ξ normalized by the length (denoted as L) of the locality frame is given by $MTFA(\xi) = (L_d - L_a)/L$.

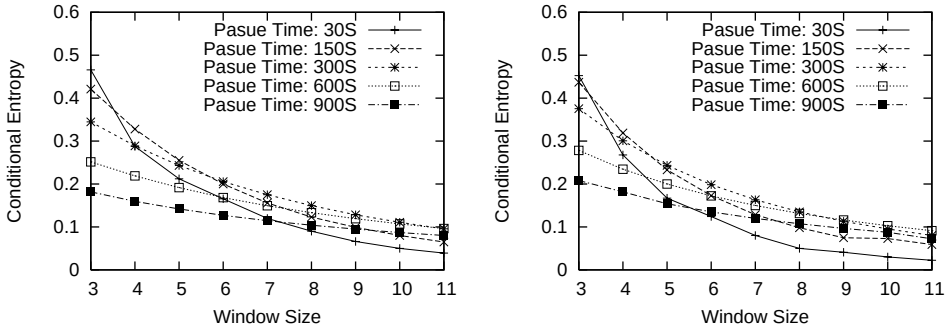
B. Simulation Results

1) Relative Entropy: Fig. 9 illustrates RE_{test} and $RE_{intrusion}$ of feature *PCH* and *PCR*. We can see that RE_{test} is smaller than $RE_{intrusion}$, suggesting that *PCH* and *PCR* are suitable features and can be used to construct anomaly detection models. We can also see that audit data at different mobility levels has different RE_{test}. Audit data at higher mobility level is more irregular. Therefore, when mobility is high, RE_{test} is larger and the difference between RE_{test} and $RE_{intrusion}$ is smaller. This explains from one aspect why the performance of anomaly detection under higher mobility is not good.



(a) Feature PCH. (b) Feature PCR.

Fig. 9. Relative Entropy of Feature *PCH* and *PCR*.



(a) Feature PCH. (b) Feature PCR.

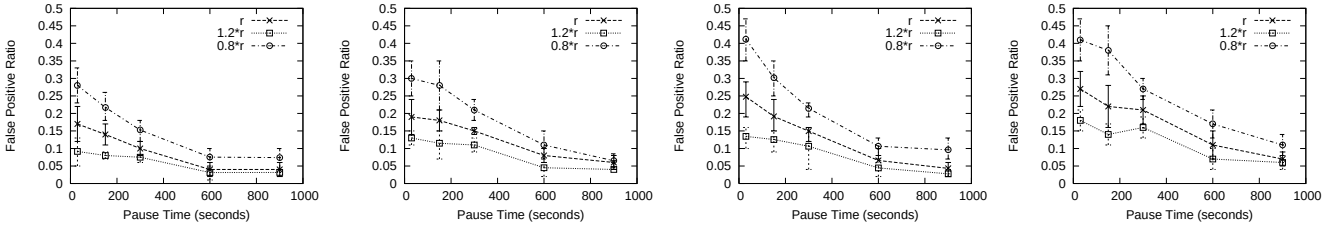
Fig. 10. Conditional Entropy of Feature *PCH* and *PCR*.

2) Conditional Entropy: We can see from Fig. 10 that audit data under different mobility levels has different regularity. This is obvious as audit data at higher mobility levels are more dynamic. Based on this, we need to construct different detection models corresponding to different mobility levels.

We can see that *CE* drops as w increases. When w is larger, less uncertainty exists for current activities, and thus *CE* drops. We also observe that *CE* does not drop dramatically when w is 4 or 5. Therefore, we set w to 4 or 5 in order to achieve a good trade-off between the accuracy and computation cost.

When w is 4 or 5, with the decrease of mobility, *CE* also decreases. This is because data at lower mobility is more regular compared to data of higher mobility. It is also the reason why the performance of MANET IDSs at lower mobility is better.

3) False Positive Rate: From Fig. 11, we can see that the false positive rate increases with the decrease of the *alert threshold* r. When r decreases, it is easy for the *alert signal* to exceed r, and thus more false alerts are generated.



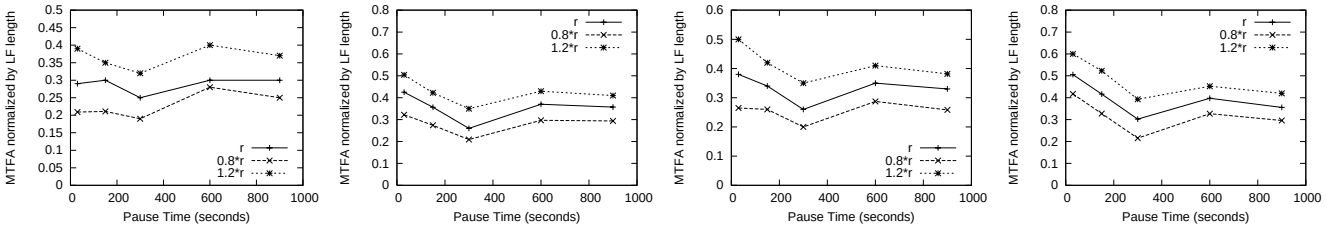
(a) PCH, window size=4. (b) PCH, window size=5. (c) PCR, window size=4. (d) PCR, window size=5.

Fig. 11. False Positive Rate of IDS Agent Constructed using *PCH* and *PCR* with Different Window Size.

With the decrease of mobility, the false positive rate decreases. This is because audit data at low mobility demonstrates high regularity. When mobility is low, routing tables will have less dynamic changes. Therefore, it is more accurate to characterize their activities, resulting in low false positive rates.

Comparing Fig. 11(a) with Fig. 11(b), when w is larger, the false positive rate increases. This is because, for the IDS with a larger w, it is more accurate and is more sensitive to unexpected abnormal changes. A small fluctuation of the routing table changes will make local IDS to generate an alert. This could also be illustrated by comparing Fig. 11(c) with Fig. 11(d).

Comparing Fig. 11(a) with Fig. 11(c), we observe a slight performance improvement of the classifier constructed using *PCH* compared to that using *PCR*. *PCH* characterizes not only the number of routing entry changes, but also the change of each routing entry. Therefore, it is a more accurate feature than *PCR*. This can also be observed by comparing Fig. 11(b) with Fig. 11(d).



(a) PCH, window size=4. (b) PCH, window size=5. (c) PCR, window size=4. (d) PCR, window size=5.

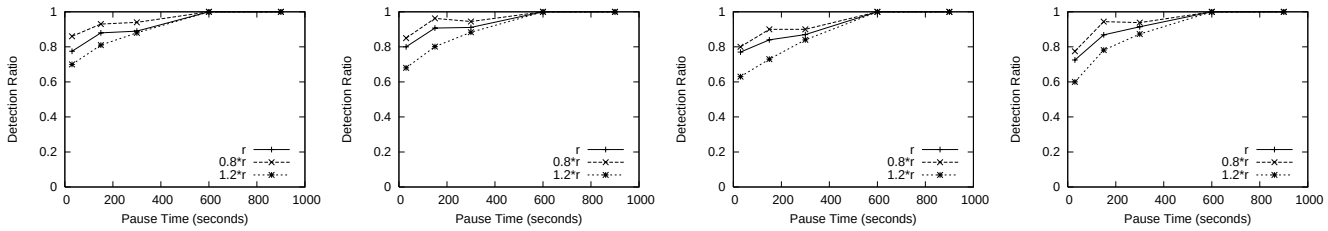
Fig. 12. MTFA of IDS Agent Constructed using *PCH* and *PCR* with Different Window Size.

4) MTFA: From Fig. 12, with the increase of r, MTFA increases. With a larger r, the detector needs longer malicious traces for the *alert signal* to exceed r, leading to a larger MTFA.

Comparing Fig. 12(a) with Fig. 12(b), we can observe a slight increase of MTFA with the classifier corresponding to window size 5. When w is larger, MC could characterize the behavior more accurately. This could contribute to a larger r, and in turn requires a longer history for the *alert signal* to exceed r. The same is true if we compare Fig. 12(c) with Fig. 12(d).

Comparing Fig. 12(a) with Fig. 12(c), we can observe that MTFA of *PCR* shows a larger value compared to that of *PCH*. Normal profile constructed using *PCH* contains more information of DSR routing caches, and is thus more accurate to characterize routing activities, making the distinction between the normal behavior and the abnormal behavior easier. Thus a shorter history is sufficient for the *alert signal* to exceed the *alert threshold r*. The same is true if we compare Fig. 12(b) with Fig. 12(d).

5) Detection rate: As we can see from Fig. 13, the detection rate increases with the decrease of r. Because when r decreases, it is easier for the *alert signal* to exceed it. We also observe that the detection rate increases with the decrease of mobility. When mobility is lower, routing table changes are more irregular. Thus it is easier for the classifier to identify the abnormal behavior.



(a) PCH, window size=4. (b) PCH, window size=5. (c) PCR, window size=4. (d) PCR, window size=5.

Fig. 13. Detection Rate of IDS Agent Constructed using *PCH* and *PCR* with Different Window Size.

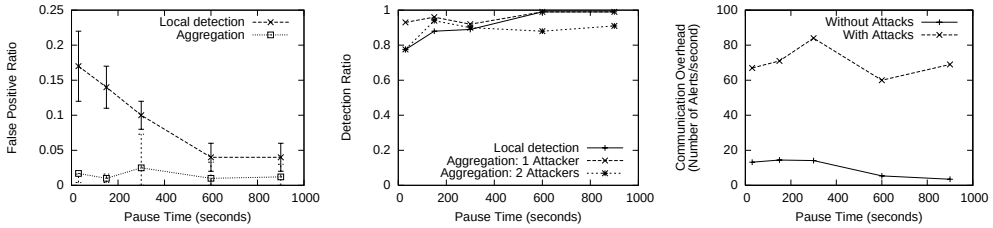
When mobility is high, the detection rate is relatively low. This is mainly due to the "partial" victims. It is very difficult to detect "partial" victims because they only receive very few routing control packets in the whole intrusion session. This makes it very difficult to differentiate normalcy caused by mobility induced errors and malicious behaviors at high mobility.

Comparing Fig. 13(a) with Fig. 13(b), we can see that when w increases, the detection rate corresponding to the same mobility also increases slightly. When w increases, MC could characterize the normal behavior more accurately because it incorporates more historical information. It could thus detect more subtle abnormal changes and increase the detection rate. In our detection model, an abnormal transition that is an invalid transition in the MC model with a smaller window size is an invalid transition in the MC model with a larger window size either. However, an abnormal transition that is an invalid transition in the MC model with a larger window size could be a valid transition in the MC with a smaller window size. The same is true if we compare Fig. 13(c) with Fig. 13(d).

Comparing Fig. 13(a) with Fig. 13(c), we realize that the classifier constructed using the feature *PCH* results in a higher detection rate compared to the classifier using the feature *PCH*. This demonstrates that *PCH* is better than *PCR* in terms of detection rates. Normal profile constructed using *PCH* contains more information of DSR routing caches, and is thus more accurate to characterize routing activities. The same is true if we compare Fig. 13(b) with Fig. 13(d).

C. Simulation Results of ZBIDS

We observe that local IDSs constructed using *PCH* demonstrate better performance results. Therefore, we use the local IDS constructed with *PCH*. w is set to 4.



(a) False Positive Rate. (b) Detection Rate. (c) Communication Overhead.

Fig. 14. Performance of ZBIDS.

1) False positive rate: As shown in Fig. 14(a), the aggregation algorithm of gateway nodes achieves much lower false positive rates than local IDS. This is because gateway nodes could collect information from a wider area and make more accurate decisions. By analyzing the aggregated alert information, gateway nodes can effectively eliminate the unexpected yet normal changes of MANETs. In this way, many false alerts can be suppressed. This is the great advantage of alert aggregation.

2) Detection rate: As we can see from Fig. 14(b), the aggregation algorithm achieves higher detection rates than local IDS. The attacker could cause "partial victims" and "full victims" at the same time. With aggregation, both "partial victims" and "full victims" could contribute to the detection of the attackers, resulting in a high detection rate.

However, our aggregation algorithm may still miss the detection of some attackers. This is because it is still possible that some normal control packets dominate the source address distribution. For example, in the routing discovery phase, we may experience some routing packet burst. This may depress the generation of true alarms. This phenomenon is more obvious with the increase of mobility.

If there exist many attackers, the probability of a single attacker address appearing as the *source* address decreases. This may lead to the decrease of the detection rate. However, in the case where different attackers have different victims and their attack times do not overlap, our ZBIDS can still achieve high detection rate.

3) Communication Overhead: The extra communication overhead introduced by ZBIDS is caused by propagating the local alerts of intrazone nodes. We measure the communication overhead as the number of transmission of all local alerts in a given time period. For a given local alert, it will be broadcast in the same zone. As shown in Fig. 14(c), when there are attacks in the network, the communication overhead is high because of the increased number of generated local alerts. When there are no attacks in the network, the communication overhead decreases with the decrease of mobility. This is because when mobility is low, local IDSs demonstrate better performance in terms of false positive rates, and thus the number of alerts locally propagated in the zone is reduced.

D. Global View of Attacks

In ZBIDS, the gateway nodes can provide a wider view of the attack happening in the network. With MIDMEF, an example of one possible aggregated alert is depicted in Fig. 15. This example shows that local IDSs attached to node 1, 4 and 6 generate local alerts and these alerts are aggregated into a $CorrelationAlert$. We can conclude that these nodes are the victims. $Source$ indicates the identification of the attacker: node 22. This makes it easy to track the offending mobile node.

```
CorrelationAlert
        name
                Routing_Disruption_Attack
        alertident
                analyzerid      1
        alertident
                analyzerid      4
        alertident
                analyzerid      6
        source
                22
END
```

Fig. 15. An Example of One Aggregated Alert.

VIII. Related Work

There are two important intrusion detection techniques: *misuse detection* and *anomaly detection*. A good taxonomy of existing technologies is presented in [18]. Research on IDS began with a report by Anderson [21] followed by Denning's seminal paper [22]. Expert system [23] [27], pattern recognition [24], and state

transition analysis [25] [26] have been used to construct misuse detection techniques. Statistical approaches [23] and MC based approaches [14] [16] have been used to construct anomaly detection techniques.

Several alert aggregation and correlation techniques [28] [29] [30] [31] [32] [33] have been proposed to facilitate intrusion analysis. Cuppens *et al.* [29] [30] use $Lambda$ language to specify attack scenarios and use Prolog to correlate alerts. In [28], an aggregation and correlation component is built in Tivoli Enterprise Console. In [31], a probabilistic method is used to correlate alerts using the similarity among their features. Ning *et al.* [32] develop three utilities to facilitate the analysis of large sets of correlated alerts. In [33], a formal data model $M2D2$ is proposed to make use of the available information.

Compared to the IDS research for the Internet, relatively few research efforts have been made to MANET IDSs. In [34], Samfat *et al.* propose IDAMN (Intrusion Detection Architecture for Mobile Networks) that includes two algorithms to model the behavior of mobile users. Marti *et al.* [13] propose to install *watchdog* and *pathrater* to identify routing misbehavior in MANETs. In [1], a general architecture for intrusion detection and intrusion response is proposed for MANETs. In [2], a data mining based method that performs the "cross-feature" analysis to capture the normal traffic patterns over MANETs is introduced. In [35], yian *et al.* provide more details on attack types and sources.

IX. Conclusion and Future Work

This paper presents the design of a nonoverlapping Zone-Based Intrusion Detection System (ZBIDS) for MANETs. We present details of constructing the Markov Chain based local anomaly detection model, including feature extraction, data preprocess, detection engine construction, and parameter tuning. Our simulation results demonstrate that the local anomaly detection model works well in low mobility environment. With high mobility, the local detection model and the aggregation algorithm under the zone-based framework should complement each other to form an effective and complete MANET IDS.

We plan to investigate more attack scenarios in MANETs, not only at the routing layer, but also at other layers. Analysis of the threats could help to construct more security-related features and misuse based intrusion detection systems. This, in turn, will help us design better aggregation and correlation algorithms.

REFERENCES

[1] Y. Zhang and W. Lee, "Intrusion Detection in Wireless Ad Hoc Networks," *the 6th Annual International Conf. on Mobile Computing and Networking (ACM MobiCom'00)*, Boston, MA, Aug., 2000, pp. 275-283.

[2] Y. Huang, W. Fan, W. Lee, and P. S. Yu, "Cross-Feature Analysis for Detecting Ad-hoc Routing Anomalies," *Proc. of the 23rd International Conference on Distributed Computing Systems*, Providence, RI, May 2003, pp. 478-487.

[3] B. Sun, K. Wu, and U. Pooch, "Routing Anomaly Detection in Mobile Ad-Hoc Networks," *IEEE International Conference on Computer Communications and Networks (ICCCN'03)*, Dallas, TX, 2003, pp. 25-31.

[4] B. Sun, K. Wu, and U. Pooch, "Alert Aggregation in Mobile Ad-Hoc Networks," ACM Wireless Security (WiSe'03), San Deigo, CA, pp. 69-78.

[5] S. Capkun, J. P. Hubaux and L. Buttyn, "Mobility Helps Security in Ad Hoc Networks," in *Proc. of the 4th ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2003)*, Annapolis, MD, June, 2003.

[6] H. Goto, M. Mambo, H. Shizuya, and Y. Watanabe, "Evaluation of Tamper-Resistant Software Deviating from Structured Programming Rules," in *Information Security and Privacy (ACISP2001)*, Lecture Notes in Computer Science 2119, Springer-Verlag, 2001, pp.145-158.

[7] J. Broch, D. Johnson, and D. Maltz, The Dynamic Source Routing Protocol for Mobile Ad hoc Networks, *http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt*, Feb. 2002, IETF Internet Draft.

[8] M. Joa-Ng and I. Lu, "A Peer-to-Peer zone-based two-level link state routing for mobile Ad Hoc Networks," in *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, Aug., 1999, pp. 1415-1425.

[9] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a Library for Parallel Simulation of Large-Scale Wireless Networks," *Proceedings of the 12th Workshop on Parallel and Distributed Simulations (PADS '98)*, Banff, Canada, May 26-29, 1998, pp. 154-161.

[10] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, Jan. 1980, pp. 702-710.

[11] T. Cover and J. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.

[12] C. Zhai and J. Lafferty, "A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval," in *Proceedings of the 24 th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, New Orleans, LA, pp. 334-342, 2001.

[13] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *the 6th Annual International Conf. on Mobile Computing and Networking (ACM MobiCom'00)*, Boston, MA, Aug. 2000, pp. 255 - 265.

[14] S. Jha, K. Tan, and R. Maxion, "Markov chains, classifiers, and intrusion detection," in the *Proc. of 14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, Canada, 2001, pp. 206-219.

[15] M. Nassehi, "Anomaly detection for markov models," *Technical Report* RZ 3011(#93057), IBM Research Division, Zurich Research Laboratory, March, 1998.

[16] Y. Nong, "A Markov Chain Model of Temporal Behavior for Anomaly Detection," in *Proc. of the 2000 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, June 6-7, 2000, pp. 171-174.

[17] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Proc. of 1999 IEEE Symposium on Research in Security and Privacy*, 1999, pp. 133-145.

[18] H. Debar, M. Dacier, and A. Wespi, "A Revised Taxonomy for Intrusion-Detection Systems," Annales des Tlcommunications, vol. 55, 2000, pp. 361 - 378.

[19] W. Lee, and D. Xiang, "Information-theoretic measures for anomaly detection," in *Proc. 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, May 13-16, 2001, pp. 130-143.

[20] D. Curry and H. Debar, "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition," *Internet Draft*, June, 2002.

[21] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," Technical Report, James P. Anderson Co., Fort Washington, PA, April, 1980.

[22] D. E. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol. 13, no. 7, pp. 222-232, Feb. 1987.

[23] P. Porras and A. Valdes, "Live Traffic Analysis of TCP/IP Gateways," *Proceedings of the 1998 ISOC Symposium on Network and Distributed System Security (NDSS'98)*, San Diego, CA, March 1998.

[24] Internet Security Systems, "RealSecure Network Protection," Nov. 2003, Available at $http://www.iss.net/products\_services/enterprise\_protection/rsnetwork$.

[25] P.A. Porras and R. Kemmerer, "Penetration State Transition Analysis C a Rule-Based Intrusion Detection Approach," *Proceedings of the 8th Annual Computer Security Application Conference*, pp. 220-229, Nov. 1992.

[26] K. Ilgun, "Ustat: A Real-time Intrusion Detection System for Unix," *Proceedings of IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 16-28, May, 1993.

[27] U. Lindqvist, and P.A. Porras, "Detecting Computer and Network Misuse through the Production-Based Expert System Toolset (P-BEST)," *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, Oakland, CA, pp. 146-161, May 9-12, 1999.

[28] H. Debar and A. Wespi, "Aggregation and Correlation of Intrusion-detection Alerts," in *Proc. of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, Davis, CA, 2001, pp. 85-103.

[29] F. Cuppens, "Managing Alerts in a Multi-intrusion Detection Environment," in *17th Annual Computer Security Applications Conference*, New Orleans, Louisiana, Dec. 10-14, 2001.

[30] F. Cuppens, and A. Mige, "Alert Correlation in a Cooperative Intrusion Detection Framework," in *IEEE Symposium on Security and Privacy*, Berkeley, CA, 2002, pp. 202-215.

[31] A. Valdes and K. Skinner, "Probabilistic Alert Correlation," in *Proc. of the Fourth International Workshop on the Recent Advances in Intrusion Detection* (RAID2001), Davis, USA, October 2001.

[32] P. Ning, Y. Cui, and D. S. Reeves, "Analyzing Intensive Intrusion Alerts Via Correlation," in *Proc. of the 5th International Symposium on Recent Advances in Intrusion Detection* (RAID 2002), LNCS 2516, Zurich, Switzerland, Oct., 2002, pp.74-94.

[33] B. Morin, L. M, H. Debar, and M. Ducass, "M2D2: A Formal Data Model for IDS Alert Correlation," in *Proc. of the 5th Int. Symp. on Recent Advances in Intrusion Detection (RAID'02)*, LNCS 2516, Zurich, Switzerland, Oct. 16-18, 2002, pp. 115-137.

[34] D. Samfat, and R. Molva, "IDAMN: An Intrusion Detection Architecture for Mobile Networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7 , pp. 1373-1380, Sept. 1997.

[35] Y. Huang, and W. Lee, "A Cooperative Intrusion Detection System for Ad Hoc Networks," *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*, Fairfax VA, October 2003.

TABLE I

SIMULATION PARAMETERS

Parameters	Values
Channel capacity	2Mbps
Channel model	Free space propagation model with a threshold cutoff
Transmission range	250m
MAC layer	Distributed Coordination Function of IEEE 802.11
Number of nodes	30
Moving region	1000m X 500m
Mobility model	Random waypoint model
Minimum speed	3m/s
Maximum speed	5m/s
Traffic pairs	8 pairs with CBR traffic
Interval transmission time	0.25s
Data packet size	512bytes
Buffer size	128 packets
Data collection interval time	3 seconds