Performance Evolution of Intrusion Detection system on MANET Using Genetic Evolution

Kirti Singh
M.Tech Scholar
SBCET, Jaipur, Rajasthan

Poonam Yadav
Asst. Professor
GITM , Gurgaon, Haryana

Abstract:

Mobile ad hoc networks (MANETs) are one of the best ever growing areas of research. By providing communications in the absence of fixed infrastructure MANETs are an attractive technology. However this edibility introduces new security threats. The traditional way of protecting networks is not directly applicable to MANETs. Many conventional security solutions are ineffective and inefficient for the highly dynamic and resource-constrained environments where MANET use might be expected. In this paper we solving security issue in Mobile Adhoc Network using Evolutionary Computation that will be discover complex properties of mobile ad hoc networks and evolve intrusion detection programs suitable for this new environment. Programs evolved using Grammatical Evolution techniques which is part of Evolutionary Computation, will be able to detect specific routing attacks on mobile ad hoc networks.

Keywords: Mobile Adhoc Network, AODV, NS2.

1. INTRODUCTION:

Intrusion is set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource and an intrusion detection system (IDS) is a system for the detection of such intrusions. There are three main intrusion detection techniques working in the literature anomaly-based, misuse based, and specification based. All intrusion techniques have their own strengths and weaknesses. One of the most commonly proposed intrusion detection techniques in MANETs is specification-based intrusion detection, where intrusions are detected as runtime violations of the specifications of routing protocols. This technique has been applied to a variety of routing protocols in MANETs.

2. PROBLEM STATEMENT:

Evolutionary computation has already showed considerable promise for creating IDS components for conventional networks, but has seen very little application in the MANET domain. Accordingly, we propose to investigate its potential in the new area. The

main issue we consider in this paper is that Evolutionary Computation will be able to discover complex properties

of mobile ad hoc networks and evolve intrusion detection programs suitable for this new environment. Programs

evolved using Genetic Programming and Grammatical Evolution techniques will be able to detect specific routing attacks on mobile ad hoc networks (namely ad

hoc flooding, route disruption, and dropping attacks) effectively.

3. EVOLUTION OF IDS IN MANETs:

Here we are going to show the Evolution of IDS using evolutionary computation techniques like Grammatical Evolution (GE).Before considering the evolution computation technique we first focus on collaborative routing protocols (AODV) as an exemplar protocol for in this paper.

3.1 Ad-Hoc On Demand Routing Protocol (AODV):

This paper the Ad hoc On-Demand Vector routing protocol is employed. Operations of AODV described in order to allow a better understanding of the routing attacks. AODV is a reactive routing protocol, discovering routes only when they are needed. "It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within ad hoc network". Three main messages in AODV: Route Request (RREQ), Route Reply (RREP), and Route Error (RERR) messages. When a node wants to communicate with another node in the network and does not have a fresh route for destination, it broadcasting a RREQ message. Intermediate nodes that receive this request either send a RREP to the source node if they have a fresh route to the destination node and the \"destination\ only" flag is not set, or forward the RREQ message to other nodes. If the request packet has been previously forwarded by this Intermediate node, it is silently dropped. When the destination node receives a RREQ for itself, it sends back a RREP message on the reverse route.

3.2 Attacks on AODV:

The three attacks (targeting at AODV) whose detection is the subject of this paper are introduced below.

- Dropping Attack
- Ad Hoc Flooding Attack
- Route Disruption Attack

Dropping Attack: In a dropping attack malicious node(s) drop data packets not destined for themselves with the aim of disrupting the network connection. Selfish nodes also drop data packets not destined for themselves to preserve their resources. Dropping attacks result in reduced network performance by causing the retransmission of data packets.

AdHoc Flooding Attack: In Adhoc flooding attack the attacker exploits property of link breakages and the route discovery mechanism by broadcasting a lot of route request messages for randomly selected nodes. The attacker aims to consume the resources of the nodes and the network. We believe that high mobility makes it difficult to distinguish a flooding attack from benign behavior on a network, since it may also cause a high number of route request packets in the network. In the simulation, the attacker broadcasts 20 route request packets in a row as in.

Route Disruption Attack: In a route disruption attack, the attacker sends route reply messages to the victim node without receiving any route request messages. There is no mechanism that checks route request-reply flow in AODV. Even nodes overhearing a route reply message can update their routing tables if the message carries a fresh route. Instead of sending route replies for random destination nodes, the attacker chooses one of its neighbors as a victim. Since the attacker is the victim node's neighbor, he already knows about the active routes of the victim through the routing. The attacker sends fresh route reply messages (with higher destination sequence numbers) to this node. Since the route reply messages sent by the attacker are fresher, the victim node updates his routing table with the routes that the attacker advertises. Hence the attacker is able to disrupt the victim's active routes and puts himself into the victim's routes. As stated in [89], one or just a few routing control packets could hardly impose severe damage to the system. So, in the simulation the attacker sends 5-10 route reply packets to the victim in a time interval (a second). This attack has been extended to 3 and 5 seconds in further simulations where the attacker achieves his goal slowly and makes the detection of his malicious behaviour difficult.

4. EVOLUTIONARY COMPUTATION

Evolutionary Computation (EC) is a research area inspired by natural evolution. It is loosely based on the process of Darwinian survival of the fittest, where individuals are competing with each other for survival and reproduction in an environment that can only host limited number of individuals. Evolutionary computation uses this approach to create computer programs for a given problem automatically, where candidate solutions of the problem correspond to the individuals, and the best programs correspond to the fittest individuals in a population in natural evolution. The pseudo code of the general steps in evolutionary computation is given below.

```
Initialize population
While termination criterion not satisfied do
Execute and evaluate fitness value of each individual
Apply genetic operators (selection, crossover, mutation, etc.) to the individuals
Create new population
End while
Return best-of-run individual
```

EC starts with generating a population of individuals (usually randomly) which are candidate solutions for the target problem. Each individual is evaluated and assigned a fitness value that indicates how well this candidate solves the problem at hand. Until a termination criterion is satisfied, new populations are generated iteratively by using selection, crossover, and mutation operators as in natural evolution.

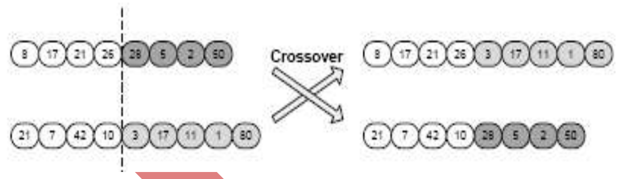
The main components of an evolutionary algorithm are outlined as follows:

- Representation of Individuals
- Fitness (Evaluation) Function
- Initialization
- Selection Mechanism (Parent Selection)
- Variation Operators
- Replacement Mechanism (Survivor Selection)
- Termination

4.1Grammatical Evolution:"Grammatical Evolution is a system that can be used to automatically generate programs in any language".GE is not the first technique using grammars, But it present a unique way of using grammars in an evolution process. GE evolves programs written in a BNF grammar. BNF (Backus-Naur Form) is a formal way to describe a language by defining a set of rules. A BNF described as a quadruple:[T, N, P, S]. T is a set of terminal symbols; N is a set of non-terminal symbols. P provides mapping from non-terminal symbols to sequences of terminal or non-terminal symbols. S is the start symbol where mapping starts.

Initialization: GE utilizes sensible initialization to create the initial population. Sensible initialization is based on ramped half and half initialization in GP but generates derivation trees of equivalent size.

Variation Operator: GE used one point crossover by default. In one-point crossover, two points are Selected on each parent randomly and their genetic contents are exchanged. In mutation operator point mutation which mutates each bit of a genome with a given probability is employed. It is possible that more than one bit making up a cordon value will be changed.



Crossover Operator on Grammatical Evolution

4.2 Simulation Model:

In this paper the networks are simulated by ns-2 to evaluate the performance of evolved programs by using GE. In this model, each node moves from its current location to a random new location with random speed and pause time within determined speed/pause time limits. Different network scenarios are created with different mobility levels and traffic loads. The parameters of the network simulation are given in Table 5.1.

Network Dimension	1100 × 500
Number of nodes	40
Packet Traffic	TCP with 25 and 30 Connections
Speed	0-22 m/sec ,pause time 42, 20 , 5 sec to simulate low, medium and high mobility respectively
Routing Protocol	AODV
Radio Propagation	Two- ray ground model with 250m transmission range
Local link connectivity	AODV periodic Hello Message
Simulation Time	550 sec(training), 2000 sec Testing

4.3 Performance of Grammatical Evolution:

The GE algorithm is run ten times on the training data and the best result of ten runs is evaluated on simulated networks. Performance of the evolved program for each attack on the network with 25 TCP connections. The false positive rates on the same network under no attack and on the stable network (no mobility) are also evaluated. The false positive rates changes due to mobility and the traffic load. Other factor such as network topology, traffic and mobility pattern is also affecting the results. In Figure 5.1 the numbers of route request packets on a normal network and on a network under ad hoc flooding attack are presented. It shows little difference in the numbers of route request packets among the networks with different mobility levels. The network under low/medium mobility may also broadcast a lot of route request packets due to its topology, its mobility and traffic patterns to build and preserve its active routes.

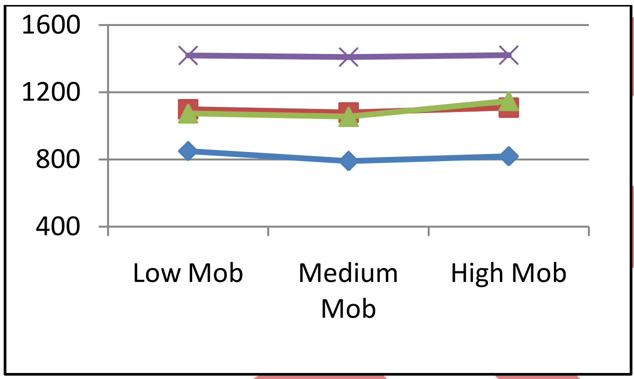


Fig: 5.1 Route Request Packets on Simulated Networks

In the results given in Table 5.2 the false positive rate of the detection program for the ad hoc flooding attack on the stable network is higher than for the mobile network. Since the programs are evolved under medium mobility, this is quite expected. Furthermore lots of route request messages are noticed in the stable network. The nodes who cannot communicate with their destination nodes broadcast route request messages frequently to find routes. As a consequence lots of route request packets under no mobility the system results in high false positive rate.

Table: 5.2 The Performance of GE on a Network with Medium Mobility/Traffic

Attack	Detection	False Positive	False positive Without attack (with Mobility)	False Positive Without attack (No Mobility)
Dropping Attack	100%	8.50%	8.83%	10.35%
Flooding Attack	99.80%	2.2%	2.15%	5.10%
Route Disruption	99.20%	0.85%	0.80%	0.60%

5. CONCLUSION:

This paper investigates the use of evolutionary computation techniques, specifically GE to evolve intrusion detection programs for MANETs. Since MANETs have become the target of new attacks which exploit the cooperative nature of routing protocols, we have aimed to detect specific attacks targeting routing protocols. It is the first application of evolutionary computation to intrusion detection in MANETs. The performance of programs evolved using GE is evaluated on simulated networks with varying mobility and traffic patterns. Both techniques show a good performance for detecting ad hoc flooding and route disruption attacks. This paper consists of the detailed descriptions of how to apply evolutionary computation techniques to the detection of the following attacks: ad hoc flooding, route disruption (and its variations), and dropping attacks.

6. REFERENCES:

[1] ns-2: The network simulator. http://www.isi.edu/nsnam/ns.

[2] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications,2:483{502, 2009.

[3] S. Laniepce, J. Demerjian, and A. Mokhtari. Cooperation monitoring issues in adhoc networks. In Proceedings of the International Conference on Communications and Mobile Computing, pages 695{700, 2006.

[4] A. E. Eiben and J. E. Smith. Introduction to Evolutionary Computing. Springer, 2003.

[5] S. Mehfuz and M. N. Doja. Swarm intelligent power-aware detection of unauthorized and compromised nodes in MANETs. Journal of Arti_cial Evolution and Applications, 2008.

[6] T. Srinivasan, V. Mahadevan, A. Meyyappan, A. Manikandan, M. Nivedita, and N. Pavithra. Hybrid agents for power-aware intrusion detection in highly mobile ad hoc networks. In Proceedings of the International Conference on Systems and Network Communication. IEEE Computer Society, 2006.

[7] T. Weise. Genetic programming for sensor networks. Technical report, Distributed Systems Group, Fachbereich 16: Elektrotechnik/Informatik, University of Kassel, 2006.

[8] D. R. White, J. Clark, J. Jacob, and S. Poulding. Evolving software in the presence of resource constraints. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'08). Springer, 2008.e