A Grammatical Evolution Approach to Intrusion Detection on Mobile Ad Hoc Networks

Sevil Şen
Department of Computer Science
University of York
York, UK
ssen@cs.york.ac.uk

John A. Clark
Department of Computer Science
University of York
York, UK
jac@cs.york.ac.uk

ABSTRACT

In recent years mobile ad hoc networks (MANETs) have become a very popular research topic. By providing communication in the absence of a fixed infrastructure they are very attractive for many applications such as tactical and disaster recovery operations and virtual conferences. On the other hand, this flexibility introduces new security risks. Moreover, different characteristics of MANETs make conventional security systems ineffective and inefficient for this new environment. Intrusion detection, which is an indispensable part of a security system, presents also a particular challenge due to the dynamic nature of MANETs, the lack of central points, and their highly constrained nodes. In this paper, we propose to investigate the use of an artificial intelligence based learning technique to explore this difficult design space. The grammatical evolution technique inspired by natural evolution is explored to detect known attacks on MANETs such as DoS attacks and route disruption attacks. Intrusion detection programs are evolved for each attack and distributed to each node on the network. The performance of these programs is evaluated on different types of networks with different mobility and traffic patterns to show their effects on intrusion detection ability.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*
; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*
; I.2 [**Artificial Intelligence**]: General

General Terms

Design, Experimentation, Performance, Security

Keywords

Mobile ad hoc networks, security, intrusion detection, grammatical evolution, artificial intelligence

WiSec'09, March 16–18, 2009, Zurich, Switzerland.

1. INTRODUCTION

A mobile ad hoc network (MANET) is a self-configuring network of mobile nodes connected by wireless links. These networks have no fixed and pre-established infrastructure such as centralized management or base stations in wireless networks. The union of nodes forms an arbitrary network topology that changes frequently due to the mobility of the nodes. Limited range wireless communication and high node mobility mean that the nodes must cooperate with each other to provide essential networking, with the underlying network dynamically changing to ensure needs are continually met. A routing protocol in such a network is responsible for finding routes and providing communication between end points through cooperating intermediate nodes. In recent years, MANETs have become a very popular research topic. Since they provide communication even in the absence of a fixed infrastructure, they are very attractive for many applications such as rescue operations, tactical operations, environmental monitoring, conferences, and the like.

MANETs by their very nature are more vulnerable to attacks than wired networks. The flexibility provided by the open broadcast medium and the cooperativeness of the mobile devices (which have generally different resource and computational capacities, and run usually on battery power) introduces new security risks. As part of rational risk management we must be able to identify these risks and take appropriate action. In some cases, we may prevent these risks cost-effectively. In other cases we may have to accept that vulnerabilities exist and seek to take appropriate action when we believe someone is attacking us. That's why, intrusion detection systems (IDSs), which monitor system activities and detect anomalies, are usually used to complement other security mechanisms.

Many intrusion detection systems have been proposed in the literature for wired networks but specific features of MANETs –mobility, resource-constrained nodes, having only local data, and limited bandwidth– make direct application of these approaches to MANETs impossible. Therefore, new approaches have been developed or existing approaches have been adapted for MANETs. In this paper, we propose to investigate the use of an artificial intelligence based learning technique to explore this difficult design space. The grammatical evolution (GE) technique inspired by natural evolution is explored to detect known attacks on MANETs such as DoS and route disruption attacks. This is an extension of our earlier research which used GE to differentiate malicious dropping from benign dropping due to mo-

bility on MANETs. The details of applying GE to intrusion detection on MANETs are presented below. For each attack type, intrusion detection programs are evolved and distributed to each node on the network. The performance of these programs is evaluated on simulated networks with varying mobility and traffic patterns. Furthermore, memory requirements of evolved programs are considered for this environment with scarce resources. The tradeoffs between the intrusion detection ability of the programs and their memory usage are demonstrated.

The paper is organized as follows. Section 2 presents related work in the area of intrusion detection on MANETs and evolutionary computation applications on intrusion detection. An overview of grammatical evolution is given in Section 3. In Section 4 the attacks on MANETs are defined, and our approach using GE to detect these attacks is given together with experiment results. Section 5 concludes.

2. RELATED WORK

In the first proposed IDS for MANETs each node has an IDS agent responsible for local detection, and collaborates with neighbouring nodes for global detection [28]. Statistical anomaly-based detection is chosen over misuse-based detection in that research, since expert rules can detect only known attacks and the rules cannot easily be updated across a wireless ad hoc network. Another proposed anomaly-based detection approach on MANETs [22] is Zone-Based IDS where the network is divided into zones based on geographic partitioning. The nodes in a zone are grouped into intrazone nodes and interzone nodes which work as bridges to the other zones. Each node in a zone is responsible for local detection and sending alerts to interzone nodes which make the final decisions. They also introduce MIDMEF (MANET Intrusion Detection Format) which defines the format of information exchange between IDS agents. Another approach which constructs an anomaly-detection model automatically by extracting the correlations among monitored features is proposed in [12]. Furthermore, simple rules are introduced to determine attack types and sometimes attackers after detecting an attack using cross-feature analysis.

One of the most commonly proposed intrusion detection techniques on 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 on MANETs such as AODV, OLSR [24][23]. There are also a few signature-based IDSs developed for MANETs. One of them is proposed in [25], which is based on a stateful misuse detection technique and defines state transition programs for known attacks on AODV. In [11], known attacks are formulated as cases for exact/similarity matching on the packet level. Some approaches integrate different intrusion detection techniques to increase effectiveness of the system. In [25], an IDS is proposed which uses a specification-based technique for attacks that violate the specifications of AODV directly and an anomaly-based technique for other kinds of attacks such as DoS. Some researchers suggest using anomaly-based and misuse-based detection system together [22][21]. In [14], a stationary secure database is proposed to maintain the latest attack signatures and patterns of normal user behaviours. Since wireless nodes can overhear traffic in their communication range, promiscuous monitoring is also used to detect some kind of attacks such as dropping and modification attacks on MANETs [13][7][16][6]. Mobile agents have been suggested as another way to provide communication between IDS agents [14].

Applications of evolutionary computation techniques to intrusion detection generally use either genetic programming (GP) or genetic algorithms (GA) to evolve intrusion detection rules for wired networks [5][10][27][18]. Their results are promising. In [4], there is a comparison of genetic programming techniques with some other machine learning techniques (Support Vector Machines and Decision Trees) for intrusion detection. The results show that genetic programming techniques outperform other techniques and it is a lightweight approach. Recent promising research has applied grammatical evolution to detect attacks on wired networks [26]. This suggests that an attempt to move a GE approach to the highly challenging MANET setting is a highly worthy subject if research.

3. GRAMMATICAL EVOLUTION

Evolutionary computation mimics the processes of natural evolution to find "fit" solutions to posed problems. A population of individuals that are candidate solutions for the target problem is generated (usually randomly). Then, each individual is evaluated and assigned a fitness value that indicates how well this candidate solves or comes close to solving the problem at hand. Until a termination criterion is satisfied, new populations are generated iteratively by using selection, crossover, and mutation operators. Selection picks individuals for mating based on the fitness value from the current population. Crossover mates selected individuals from the current population to generate new individuals. Mutation changes selected individuals to introduce diversity into the population. These operators are used to provide better solutions in the new population. The general steps in evolutionary computation are shown below.

```
initialize population
while termination criterion not satisfied do
    execute and evaluate fitness value of each individual
    apply genetic operators (crossover, mutation, etc.) to
    the individuals
    create new population
end while
return best-of-run individual
```

Grammatical Evolution (GE) is an evolutionary computation technique evolving programs written in a BNF grammar which allows generating computer programs in an arbitrary language to be generated [19]. Although our approach to search for excellent (fit) programs does indeed use a population based evolutionary computation, approach as described above, we could also use alternative optimisation or search techniques. The core idea of GE relates to how simple integer sequences can be interpreted as programs and this is now described below.

BNF (Backus-Naur Form) is a formal way to describe a language. It can be simply described as a set of rules. A BNF system is described a quadruple:T, N, P, S. T is a set of terminal symbols, which are concrete terms in the grammar. N is a set of non-terminal symbols, which are place-holders used in the generation of terminals by using the set of production rules P. P provides mapping from non-

terminal symbols to terminal or non-terminal symbols. S is the start symbol where mapping starts. A BNF grammar for the symbolic regression problem [17] is given in Table 1. The symbols enclosed by brackets (<>) are non-terminals, others are terminals. The productions of a rule assigned by '::=' are separated with '|'.

Table 1: BNF grammar for symbolic regression

S = <expr>
<expr> ::= <expr><op><expr> | (<expr><op><expr>) | <pre-op>(<expr>) | <var>
<op> ::= + | - | / | *
<pre-op>::= sin | cos | exp | log
<var> ::= X | 1.0

In GE genomes are represented by variable-length binary strings. A group of 8 bits in a genome produces a codon value, which is used to choose a rule from a BNF grammar. Assume that the genome of an individual, which is mapped from a binary string to an integer string, is:

220	35	47	68	137	55	144	22	46	178

Left-most non-terminal expansion is used at any stage. The first codon value is used to choose the production rule from the start rule S. Since S has one production expr, 220 is used to choose one of the productions of expr according to the formula below:

$$Rule = (codon\ val.) MOD (\#\ of\ productions\ of\ the\ non{-}terminal) \quad (1)$$

expr has 4 productions, so 220 MOD 4=0 is calculated and the production "<expr><op><expr>" is selected accordingly (production options are numbered starting from 0). Then, next codon value 35 is used to choose from expr rule, since it is the first non-terminal at this point. So, third (35 MOD 4=3) production of the expr, "<var>", is selected. The individual becomes "<var><op><expr>", the next codon value 47 is used to choose the (47 MOD 2=1) production of "<var>" rule to give "1.0<op><expr>". The process continues until there are no mapped non-terminal symbols. If there are still non-terminal symbols to be expanded and all integers in a string are used, the interpretation simply returns to the beginning of the genome again and starts re-reading the string. This process is called wrapping. However, there is a threshold value for such wrapping. If this value is exceeded and there are still unmapped non-terminals, this individual is assumed invalid and assigned the lowest fitness value.

4. GRAMMATICAL EVOLUTION IN INTRUSION DETECTION ON MANETS

4.1 Attacks

In this paper, we use grammatical evolution to evolve intrusion detection programs for known attacks against routing protocols on MANETs. AODV [9], which is one of the most commonly used on-demand routing protocols on MANETs, is used as an exemplar routing protocol. The attacks on AODV considered in this paper are explained below.

4.1.1 Dropping Attack

In the dropping attack scenario malicious nodes drop data packets not destined for themselves for a given time interval to disrupt the network connection. Since malicious nodes need to be on a routing path to drop data packets, they have little reason to drop routing protocol control packets such as RREQ, RREP, and RERR messages used in route discovery and maintenance mechanisms of AODV. So, it is assumed that malicious nodes do not drop routing protocol control packets. Dropping data packets reduces network performance by causing the retransmission of the packets, the discovery of new routes to the destination, and the like. Furthermore, they can prevent the end-to-end communications between nodes if the dropping node is at a critical point. In the simulation, malicious nodes drop each packet they receive in 3 second intervals.

While packet losses usually occur due to congestion in wired networks, there can be other causes on MANETs such as wireless link transmission errors and mobility [15]. Mobility is given as the major cause of packet losses (60%) on AODV [15]. So, we mainly aim to differentiate packet dropping due to malicious behaviour from packet dropping due to mobility.

4.1.2 Route Request Flooding Attack

Network topology changes frequently on MANETs due to mobility. Moreover link breakages are very common in wireless networks. These may result in making existing routes inactive and discovering new routes by route request packets. Route request messages are sent when nodes need a new route on reactive routing protocols such as AODV. Evidently, mobility may increase the number of route request packets on the network. In the flooding attack scenario, the attacker exploits this property of 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. In the simulation, the attacker broadcasts 20 route request packets in a row as in [9].

4.1.3 Route Disruption Attack

In this attack scenario, the attacker sends route reply messages to the victim node without receiving any route request messages from that node. Instead of sending route replies for random destination nodes, the attacker chooses one of its neighbours as a victim and sends route reply messages (with higher destination sequence number) to this node for disrupting the active routes in its routing table. Since the attacker is the victim node's neighbour, he already knows about the active routes of the victim through the routing control packets broadcasted by him. As it is stated in [22], one or few routing control packets could hardly incur severe damage to the system. So, in the simulation the attacker sends 5-10 route reply packets to the victim in one 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.

Separate programs are evolved by GE to detect each attack against MANET explained above and the evolved programs are distributed to each node on the network. Attacks are detected by the nodes that the attacks affect directly. In dropping attacks, the monitor nodes are those who send/forward packets to the next node. These mon-

itor nodes listen promiscuously to determine whether the next node forwards the packets and try to differentiate the malicious dropping from benign dropping. However, promiscuous monitoring fails in ambiguous collisions and receiver collisions [16]. In our previous research, we use TCP acknowledgements instead of promiscuous monitoring to indicate dropping on the network [20]. In flooding attacks, the nodes who are flooded by route request messages detect the attack. In route disruption attacks, the victim node is assumed to detect malicious change in its routing table.

4.2 Grammatical Evolution

In GE, a problem is defined with the grammar and the fitness function. The grammar used to evolve programs to detect the attacks on MANETs and raise an alarm is defined in Table 2. *libge* [2] library is used for the evolution process.

Table 2: BNF grammar used for the problem

S = <code>
<code> ::= if(<cond>) {raise_alarm()}
<cond> ::= <cond><set-op><cond> | <expr><relop><expr>
<expr> ::= <expr><op><expr> | (<expr> <op><expr>) | <pre-op>(<expr>) | <pre-op2>(<expr>) | <var>
<op> ::= + | - | / | *
<pre-op>::= sin | cos | log | ln | sqrt | abs | exp | ceil | floor
<pre-op2>::= max | min | pow | percent
<rel-op>::= < | ≤ | > | ≥ | == | !=
<set-op> ::= and | or
<var> ::= feature set in Appendix A

Features used in the grammar are given in Appendix A. We use both mobility-related features as well as packet-related features as input to the evolution system. While some of these features give information about mobility directly (such as changes in the number of neighbours), some of them can be the result of mobility (such as routes added in the last period). Packet-related features include routing protocol control packets and transport protocol packets. Some features are used only for particular attacks such as data packets not forwarded by the next node for dropping attacks, and the average hop count feature for route disruption attacks. All features are gathered every time interval by each node.

GE parameters The parameters used in GE are given in Table 3.

Table 3: GE Tableau for detecting known attacks on MANETs

Objective:	Find a computer program to detect flooding, route disruption and dropping attacks on MANETs
Non-Terminal Operators:	The binary operators +,-,*, /, pow, min, max, percent The unary operators sin, cos, log, ln, sqrt, abs, exp
Terminal Operators:	The feature set in Appendix A
Fitness cases:	The given sample of network data marked malicious or non-malicious
Raw Fitness:	The detection rate over the fitness cases subtract the false positive rate over the fitness cases
Standardised Fitness:	Same as raw fitness
Parameters	Populations Size = 100 Termination when Generations= 2000 Prob. Mutation = 0.01 Prob. Crossover = 0.9 Steady State

The **fitness function** is one of the most important parameters in evolutionary computation, since it evaluates how good the solution is. In our experiment, we use a fitness function based on the detection rate and the false positives rate to evaluate effectiveness of a proposed system.

The detection rate shows the ratio of correctly detected intrusions to the total intrusions on the network. The false positive rate shows the ratio of normal activities that are incorrectly marked as intrusions to the total normal activities on the network. A low false positive rate is as important as high detection rate for a good intrusion detection system.

$$detection = \frac{correctly\ detected\ attacks}{total\ attacks} \quad (2)$$

$$false\ pos. = \frac{normal\ activities\ incorrectly\ detected\ as\ attacks}{total\ normal\ activities} \quad (3)$$

$$Fitness = detection - false\ positive \quad (4)$$

4.3 Experimental Results

The evolved programs are evaluated on the networks simulated by ns-2 [3]. Mobility of the nodes is simulated by the Random Waypoint model which is created using BonnMotion [1]. In the Random Waypoint model, each node moves from its current location to a random new location with random speed and pause time in determined speed/pause time limits [8]. Different network scenarios are created with different mobility levels and traffic loads. 50 nodes are placed in a topology of 1000m by 500m. Both TCP and UDP/CBR traffic are used for communication. The maximum of connections is set to either 20 or 30 to simulate different traffic loads. The maximum speed of nodes is set to 20 m/sec and the pause time between movements is set to 40, 20, and 5 sec to simulate low, medium, and high mobility respectively. AODV is chose as a routing protocol and AODV periodic hello messages are used for local link connectivity. The simulations run 5000 seconds for training and 2000 seconds for testing.

The algorithm is evolved using the training data collected from a network under medium mobility with 30 TCP connections. The same network with attacks and without attacks is used together for training to reduce false positives. The best result of ten runs is chosen for each attack type and evaluated on different network scenarios. Table 4 shows the performance of the evolved program for each attack on the network under medium mobility with 30 TCP connections. The false positive rates on the same network under no attack and on the stable network under no attack are also given in this table.

The results show that the evolved program for dropping attack has a high false positive rate. False positives which occur in 5 seconds by the same node after raising an alarm are ignored, since the effect of the dropping might continue. As we mentioned before, packet losses occur frequently on MANETs due to congestions, wireless link transmission errors, and mobility. So, the features beyond the routing protocol are also needed to detect malicious dropping effectively. The system can be improved by adding these features to the evolution. The false positive rate of the detection program for the flooding attack on the stable network is higher than the mobile network. Since the programs are evolved under medium mobility, this is quite expected. It is seen that the false positive rate of the detection program for route disruption attack on the network under attack is slightly more than the network without any attacks. The effect of this attack might take longer on the network.

Table 4: the experimental results

Attacks	Detection	False Positive	False Positive without attack	False Positive without attack no mobility
Dropping Attack	100%	8.46%	8.81%	10.30%
Flooding Attack	99.86%	2.00%	2.19%	5.11%
Route Disruption	100%	0.83%	0.81%	0.59%

For each attack, the performance of the evolved programs is tested on the network under different mobility and traffic levels. The results for each attack are given in Table 5. The results demonstrate that grammatical evolution approach shows a good performance on intrusion detection in mobile ad hoc networks. The detection rate is high for both the flooding attack and the route disruption attack, since the results of these attacks are quite evident on the network. However they are not easily differentiated from normal network operations under high mobility and high traffic.

Table 5: the results under varying mobility and traffic levels

Network Scenarios	Flooding Attack		Route Disruption Attack	
	DR	FPR	DR	FPR
low mobility low traffic	99.81%	0.29%	100%	0.41%
low mobility medium traffic	98.54%	1.72%	100%	0.88%
medium mobility low traffic	99.86%	0.36%	100%	0.4%
medium mobility medium traffic	99.86%	2.00%	100%	0.83%
high mobility low traffic	99.96%	0.66%	100%	0.44%
high mobility medium traffic	98.66%	1.73%	100%	0.76%

The false positive rate changes due to the mobility and the traffic load. Other factors such as network topology, traffic and mobility patterns also affect the results. For instance, we expect that the number of route request packets is much higher in high mobility networks, since mobility can make existing routes inactive frequently. In Figure 1 the numbers of route request packets on a normal network and on a network under 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.

The route disruption attack is also implemented in 1, 3, 5 seconds on the same networks and trained separately by collecting data each 1, 3, 5 seconds respectively. The results for the best individuals (with the highest detection rate, 100%) are presented in Figure 2. It is clear that the false positive rate is increased proportional to the attack distribution time. The best individual with the highest detection rate is chosen here to compare attack patterns with each other easily. There are also individuals who have lower false positive rate but also a lower detection rate among the evolved programs. It is a trade-off between detection rate and false positive rate to decide the best individual.

For each attack type, the evolved programs and the memory usage of each program (detect an attack and call the response function) are given below. Memory usage of each program is calculated according to the formula below:

$$mem = program\ binary\ size + variables * sizeof(float) \quad (5)$$

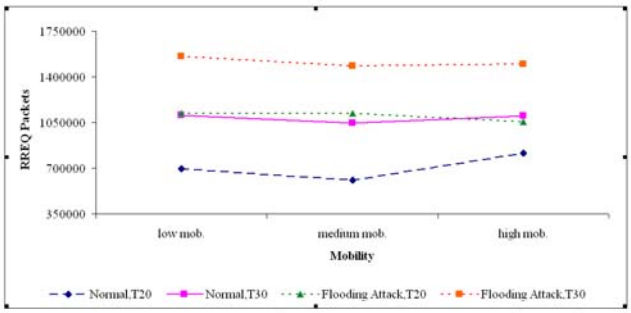

Figure 1: Route Request Packets on the networks under different mobility levels

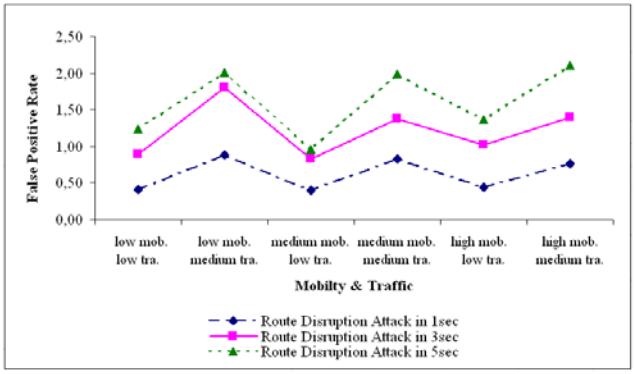

Figure 2: Performance of the evolved programs for route disruption attack

Program size of the evolved programs is also analyzed, since mobile nodes can have scarce resources. Efficiency is as important as effectiveness on intrusion detection in such a resource-constrained network. In Figure 3, the best individuals of each run are given with their program size, detection rate, and false positive rate on the training data. Program size is calculated by adding the number of terminals and the number of functions in the program. It is seen that there are programs that have similar accuracy but different program size. There is a trade-off between resources consumed by programs and detection efficacy.

Lastly the relation between accuracy of evolved programs for flooding and route disruption attacks and number of generations is shown in the figure below.

Table 6: programs (best individuals) evolved by GE for each attack

Attack Type	Evolved Program	Memory Usage
Flooding	sent_rrepPs + exp(frw_aodvPs - updatedroutes * pow(frw_rreqPs, added_repairedroutes)) > no_neighbours	≃7.52kB
Route Disruption	log(exp(invroutes_other)) + recv_aodvPs - percent(exp(init_aodvPs), sent_rreqPs) + repairedroutes > recvB_rerrPs	≃7.79kB

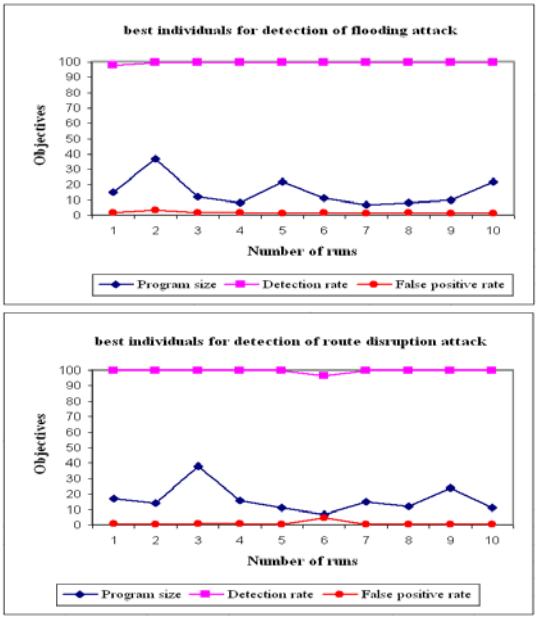

Figure 3: Best individuals of each run versus IDS objectives

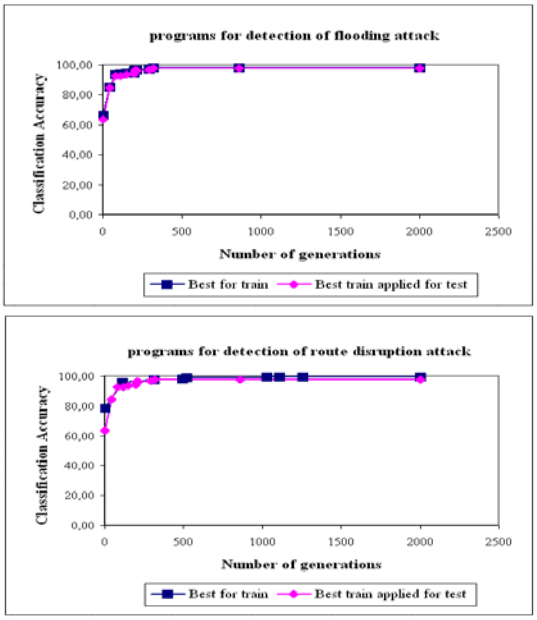

Figure 4: Relation between classification accuracy and number of generations

5. CONCLUSIONS AND FUTURE WORK

MANETs are a highly promising new form of networking. However, they are more vulnerable to attacks than wired networks. Furthermore, existing intrusion detection systems are not applicable to this highly dynamic environment. We propose to use a promising technique from artificial intelligence to synthesise the most appropriate intrusion detection programs for this challenging network type. Grammatical evolution essentially "grows" intrusion detection programs by evaluating populations of potential programs and subjecting them to a variety of genetically inspired operators. Our grammatical evolution technique shows good performance on evolving efficient detectors for known attacks against routing protocol on MANETs such as flooding, and route disruption attacks. The work presented here is an extension of our earlier research which aims to distinguish malign and benign packet dropping using the same technique. An evolved program's effectiveness is evaluated under some variations applied to the attack patterns, the mobility and traffic levels of the network. The program size of best individuals is also presented and compared with their effectiveness.

We consider only detection rate and false positive rate in our experiments to evaluate our programs. However, other non-functional properties of the programs such as program size, power consumption should be taken into account for MANETs where the nodes might well be highly resource-constrained. Although we have not done so here, evolutionary techniques have ability to handle messy multi criteria tradeoffs, such as those we have to deal with. In the future, we are planning to explore optimal trade-offs between these objectives of intrusion detection on MANETs: detection and false positive rate, resources used by programs (e.g. memory, power). We believe that artificial intelligence based approaches to program synthesis such as grammatical evolution are of significant potential benefit for the evolution of IDS programs for challenging environments and we encourage the research community to explore their use.

6. REFERENCES

[1] Bonnmotion: A mobility scenario generation and analysis tool.

[2] libge: A c++ library for grammatical evolution.

[3] Ns-2: The network simulator.

[4] A. Abraham and C. Grosan. Evolving intrusion detection systems. In *Genetic Systems Programming: Theory and Experiences*, volume 13, pages 57–79. Springer, 2006.

[5] A. Abraham, C. Grosan, and C. Martiv-Vide. Evolutionary design of intrusion detection programs. *Int. Journal of Network Security*, 4:328–339, 2007.

[6] F. Anjum and R. Talpade. Lipad: lightweight packet drop detection for ad hoc networks. In *60th IEEE Vehicular Technology Conference Proceedings*, pages 1233–1237. IEEE, 2004.

[7] S. Buchegger and J.-Y. L. Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing Proceedings*, pages 403–410. IEEE Computer Society, January 2002.

[8] 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, 2002.

[9] E. R. C.E. Perkins. Ad-hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computer Systems and Applications Proceedings*, pages 90–100, 1999.

[10] M. Crosbie and G. Stafford. Applying genetic programming to intrusion detection. In *AAAI Symposium on Genetic Programming Proceedings*, 1995.

[11] R. Guha, O. Kachirski, D. Schwartz, S. Stoecklin, and E. Yilmaz. Case-based agents for packet-level intrusion detection in ad hoc networks. In *Proceedings of the 17th International Symposium on Computer and Information Sciences*, 2002.

[12] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *In Proc. of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[13] J. P. A. J. J. Parker, J. Undercoffer. On intrusion detection and response for mobile ad hoc networks. In *23th IEEE Int. Performance Computing and Communications Conference Proceedings*, 2004.

[14] O. Kachirski and R. Guha. Effective intrusion detection usign multiple sensors in wireless ad hoc networks. In *Proceedings of the 36th IEEE International Conference on System Sciences*, 2003.

[15] Y. Lu, Y. Zhong, and B. Bhargava. Packet loss in mobile ad hoc networks. TR 03–009, Dept. of Computer Science, Purdue University, April.

[16] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *In Proc. of ACM Int. Conf. on Mobile Computing and Networking (MOBICOM)*, pages 255–265, 2000.

[17] M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Springer, 2003.

[18] M. Pillai, J. Eloff, and H. Venter. An approach to implement a network intrusion detection system using genetic algorithms. In *SAISCIT Proceedings*, pages 221–228, 2004.

[19] C. Ryan, J. Colline, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *1st European Workshop on Genetic Programming Proceedings, LNCS 1391*, pages 83–95. Springer, 1998.

[20] S. Sen and J. A. Clark. Evolving intrusion detection rules on mobile ad hoc networks. In *Proceedings of the PRICAI, LNAI 5351*, pages 1053–1058. Springer, 2008.

[21] A. Smith. An examination of an intrusion detection architecture for wireless ad hoc networks. In *Proceedings of the 5th National Colloquium for Information System Security Education*, 2001.

[22] B. Sun, K. Wu, and U. Pooch. Zone-based intrusion detection for mobile ad hoc networks. *Int. Journal of Ad Hoc and Sensor Wireless Networks*, 2(3), 2003.

[23] C. Tseng, S.-H. Wang, W. Lee, C. Ko, and K. Lewitt. Demem: Distributed evidence driven message exchange intrusion detection model for manet. In *In Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06)*, pages 249–271. Springer, 2006.

[24] C.-Y. Tseng, P. Balasubramayan, C. Ko, R. Limprasittiporn, J. Rowe, and K. Lewitt. A specification-based intrusion detection system for aodv. In *In Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2003.

[25] G. Vigna, S. Gwalani, K. Srinivasan, E. M. Belding-Royer, and R. A. Kemmerer. An intrusion detection tool for aodv-based ad hoc wireless networks. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 16–27, Washington, DC, USA, 2004. IEEE Computer Society.

[26] D. Wilson and D. Kaur. Knowledge extraction from kdd'99 intrusion data using grammatical evolution. *WSEAS Transactions on Information Science and Applications*, 4:237–244, February 2007.

[27] C. Yin, S. Tian, H. Huang, and J. He. Applying genetic programming to evolve learned rules for network anomaly detection. In *ICNC Proceedings, LNCS 3612*, pages 323–331. Springer, 2005.

[28] Y. Zhang, W. Lee, and Y. an Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks Journal (ACM WINET)*, 2(5), September 2003.

APPENDIX

A. THE FEATURES

Features	Explanation
neighbours	no. of neighbours
added_neighbours	no. of added neighbours
removed_neighbours	no. of removed neighbours
active_routes	no. of active routes
repaired_routes	no. of routes under repair
invalidated_routes	no. of invalidated routes
addedroutes_disc	no. of added routes by route discovery mechanism
addedroutes_notice	no. of added routes by overhearing
updated_routes	no. of updated routes (modifying hop count, sequence number)
added_repairedroutes	no. of added routes under repair
invroutes_timeout	no. of invalidated routes due to expiry
invroutes_other	no. of invalidated routes due to other reasons
avg_hopcount	average no. of hop counts of active routes
recv_rreqPs	no. of received route request packets destined to this node
recvF_rreqPs	no. of received route request packets to be forwarded by this node
send_rreqPs	no. of broadcasted route request packets from this node
frw_rreqPs	no. of forwarded route request packets from this node
recv_rrepPs	no. of received route reply packets destined to this node
recvF_rrepPs	no. of received route reply packets to be forwarded by this node
send_rrepPs	no. of initiated route reply packets from this node
frw_rrepPs	no. of forwarded route reply packets from this node
recvB_rerrPs	no. of received broadcast route error packets (to be forwarded or not)
send_rerrPs	no. of broadcasted route error packets from this node
recv_aodvPs	no. of received total routing protocol packets
recvF_aodvPs	no. of received total routing protocol packets to be forwarded
send_aodvPs	no. of initiated total routing protocol packets from this node
frw_aodvPs	no. of forwarded total routing protocol packets by this node
dropped_dataPs	no. of data packets not forwarded by the next node