Power-Aware Intrusion Detection in Mobile Ad Hoc Networks

Sevil Şen, John A. Clark, and Juan E. Tapiador

Department of Computer Science,
University of York, YO10 5DD, UK
{ssen,jac,jet}@cs.york.ac.uk

Abstract. Mobile ad hoc networks (MANETs) are a highly promising new form of networking. However they are more vulnerable to attacks than wired networks. In addition, conventional intrusion detection systems (IDS) are ineffective and inefficient for highly dynamic and resource-constrained environments. Achieving an effective operational MANET requires tradeoffs to be made between functional and non-functional criteria. In this paper we show how Genetic Programming (GP) together with a Multi-Objective Evolutionary Algorithm (MOEA) can be used to synthesise intrusion detection programs that make optimal tradeoffs between security criteria and the power they consume.

Key words: Mobile ad hoc networks, intrusion detection, power-aware, evolutionary computation, genetic programming, multi-objective optimization.

1 Introduction

Intrusion is any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource [11] and an intrusion detection system (IDS) is a system for the detection of such intrusions. Since prevention techniques cannot be sufficient and new intrusions continually emerge, IDS is an indispensable part of a security system. An IDS is introduced to detect possible violations of a security policy by monitoring system activities and responding to those that seem intrusive. If we detect the attack once it comes into the network, a response can be initiated to prevent or minimise the damage to the system. An IDS also helps prevention techniques improve by providing information about intrusion techniques.

There have been many approaches proposed for intrusion detection. Intrusion detection methods are classified into three main techniques: anomaly-based, misuse-based, and specification-based. An anomaly-based technique profiles the symptoms of normal behaviours of the system such as usage frequency of commands, CPU usage for programs, and the like. It detects intrusions as anomalies, i.e. deviations from the normal behaviours. In the literature, various techniques have been applied for anomaly detection, e.g. statistical approaches, and artificial intelligence techniques like data mining and neural networks. Misuse-based detection compares known attack signatures with current system activities. It is generally preferred by commercial IDSs since it is efficient and has a low false positive rate. Both anomaly-based and misuse-based approaches have their strengths and weaknesses. Therefore, these techniques are generally employed together for effective intrusion detection. Specification-based technique is introduced as a promising alternative that combines the strengths of anomaly-based and misuse-based detection techniques [28], providing detection of known and unknown attacks with lower false positive rate. In this technique, a set of constraints of a program or a protocol are specified and intrusions are detected as runtime violations of these specifications.

Mobile ad hoc networks (MANETs) are a new type of networking which combine wireless communications with a high degree of node mobility. They also provide communication even in the absence of a fixed infrastructure. Because of the flexibility they provide, MANETs have become a very popular research topic and have been proposed for use in many areas such as rescue operations, tactical operations, environmental monitoring, virtual conferences, and the like. On the other hand they are more vulnerable to attacks than wired networks and introduce new security risks.

Due to MANETs' specific features, conventional IDSs cannot be easily applied to these environments. When an IDS is being designed for these networks, there are new issues which should be taken into account. Lack of concentration points, mobility and cooperativeness of nodes, and limited resources are the main issues. In this paper, we address the intrusion detection problem in MANETs.

1.1 Our Contributions

In this research, we propose a new approach which uses evolutionary computation techniques to explore the MANETs' complex design space. Programs are evolved using the Genetic Programming (GP) technique to detect known attacks against MANETs and evaluated on simulated networks with varying mobility and traffic patterns. It is shown that GP effectively detects known attacks, flooding and route disruption attacks, against AODV. However a good intrusion detection system (IDS) on MANETs should also be suited to resource-constrained environments. That is why we employ Multi Objective Evolutionary Computation (MOEA) techniques in order to discover trade-offs between non-functional and functional properties of programs, and optimise these objectives simultaneously during evolution. Our main contribution in this paper is to evolve a set of programs for each attack offering different trade-offs between intrusion detection ability of evolved programs and their energy usage. Moreover, we investigate if it is better to evolve separate programs for each attack or one program to detect both attacks.

The paper is organised as follows. Section 2 presents related work in the area of evolutionary computation applications on intrusion detection and proposed IDSs on MANETs. The problem, intrusion detection on MANETs, and the attacks considered in this research are defined in Section 3. Section 4 introduces the techniques (GP, MOEA) applied to the problem and also demonstrates the performance of intrusion detection programs evolved by using these techniques on simulated networks. In Section 5 power-aware intrusion detection is presented with the experimental results. Section 6 concludes the paper by summarising our main findings and pointing out some avenues for future research.

2 Related Work

Applications of evolutionary computation techniques to intrusion detection use generally either genetic programming (GP) or genetic algorithms (GA). A recent research which develops an IDS by using genetic programming is given in [6]. The main idea is to create an automatic intrusion detection algorithm based on the input features and the functions given. The output program is small, simple, and uses just a few input features where "most machine learning paradigms (artificial neural networks, support vector machines (SVM), decision trees) examine all input features to detect intrusions [6]". The results of the evaluation show that the approach is lightweight and effective, satisfying the main goals of an intrusion detection algorithm. The GP techniques used in that research are compared with some other machine learning techniques (SVM and Decision Trees) for intrusion detection in [5]. The results show that GP outperforms other techniques and it is a lightweight approach. There are also promising applications of GAs to intrusion detection [16][14]. Grammatical evolution, another evolutionary computation paradigm, has been proposed recently for intrusion detection in wired networks [32]. It allows the generation of computer programs in an arbitrary language and supports type safety by using a BNF grammar to represent the problem.

Even though there are many proposed IDSs for wired networks, MANETs specific features make direct application of these approaches to MANETs impossible. For that reason, researchers have proposed new approaches for intrusion detection in MANETs for the last decade. 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 such as AODV and OLSR [27][26]. There are also a few signature-based IDSs developed for MANETs. One of them, proposed in [29], is based on a stateful misuse detection technique and defines state transition programs for known attacks on AODV. In [29] 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. 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 [18][15][9]. Mobile agents have been suggested as another way to provide communication between IDS agents [13].

Few artificial intelligence based intrusion detection systems have been proposed to explore the complex spaces associated with MANETs. In the first proposed IDS for MANETs [34] statistical anomaly-based detection is chosen over misuse-based detection, since expert rules can detect only known attacks and the rules cannot easily be updated across a wireless ad hoc network. SVM Light and RIPPER classifiers are employed and compared in that research. In [25] Markov-chain based local anomaly detection model is proposed for a zone-Based IDS architecture where the network is divided into zones based on geographic partitioning. Another approach which constructs an anomaly-detection model automatically by extracting the correlations among monitored features is proposed in [12]. Furthermore, they introduced simple rules to determine attack types and sometimes attackers after detecting an attack using cross-feature analysis.

There is little research on applying evolutionary computation techniques to sensor networks and mobile ad hoc networks. In [30] the Distributed Genetic Programming Framework (DGPF) which automatically discovers distributed algorithms for given problems is introduced for sensor networks. The election problem (i.e. finding the maximum) is solved by using this framework and a multi-objective optimisation technique is employed on this problem to consider non-functional attributes such as code size, memory size, and transmission count for this resource-constrained environment. An application of grammatical evolution to intrusion detection for MANETs is proposed in our previous work [21]. The reader may refer to [8][22] for a detailed review of intrusion detection in MANETs.

3 Intrusion Detection in MANETs

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 introduces new security risks. Furthermore mobile nodes generally have different resource and computational capacities, and run usually on battery power. As part of a security strategy, we should detect these attacks and take appropriate action. Detection of such intrusions in an efficient way is the primary focus of this research. The attacks against MANETs considered in this research are given below. AODV [19], which is one of the most commonly used on-demand routing protocols for MANETs, is used as an exemplar routing protocol.

Route Request Flooding Attack. Network topology changes frequently in MANETs due to mobility. Moreover link breakages are very common in wireless networks. These may make existing routes inactive and cause new routes to be sought by issuing route request (RREQ) packets. Route request messages are sent when nodes need a new route in 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 our simulation the attacker broadcasts 20 route request packets in a row as in [19].

Route Disruption Attack. In this attack scenario, the attacker sends route reply messages (RREP) 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 broadcast by him. As stated in [25], 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 a time interval.

4 Evolutionary Computation Techniques In Intrusion Detection

Evolutionary computation provides a framework to create computer programs automatically. Evolutionary computation techniques are loosely based on the process of Darwinian survival of the fittest. They start by generating a population of individuals (usually randomly) which are candidate solutions for the target problem. 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 as in the natural evolution. These genetic operators are used to provide better solutions in the new population. Selection provides great opportunity for fitter individuals to survive. Whilst crossover mimics the exchange of DNA under sexual production, mutation mimics natural mutation causing new areas of the design space to explore.

4.1 Genetic Programming

In this research GP, one of the most employed evolutionary computation techniques in the literature, is employed to detect the flooding and route disruption attacks described above. Programs are evolved for each attack by this technique and then evaluated on different networks with varying traffic and mobility patterns.

A problem in GP is defined with the functions, variables and the fitness function. The set of variables used, which include mobility-related features as well as packet-related features of a node in the network, is given in Appendix A. 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 added routes in the last period). Packet-related features include the number of routing protocol control packets sent, received, forwarded by a node in a time interval. The average hop count feature is used only for the route disruption attacks. The functions used together with the major GP parameters are given in Table 1. *Population size* is the number of individuals in a population in any generation. *Generations* defines when (at which generation) the evolution process stops. *Crossover probability* shows how likely individuals selected for mating will exchange elements. *Reproduction probability* shows how likely an individual will be copied without any modification to the new generation. Tournament selection is one of the methods used for selecting individuals for mating. In this method, a group of individuals is chosen randomly from the population and the best individual from this group (i.e. the fittest) is selected as parent. *Tournament size* defines the number of the individuals in this group. *ECJ 18* [2] toolkit is used for the GP implementation. The parameters not listed here are the default parameters of the toolkit.

Table 1. GP parameter settings

Objective	Find a computer program to detect flooding and route disruption attacks against MANETs
Function set	+,-,*, /, pow, min, max, percent sin, cos, log, ln, sqrt, abs, exp, ceil, floor, and, or, comparison operators
Terminal set	The feature set in Appendix A
Populations Size	100
Generations	1000
Crossover Probability	0.9
Reproduction Probability	0.1
Tournament Size	7

The fitness function is very important in evolutionary computation since it evaluates how good the individual is. The fitness function used in the experiments is defined below. 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 high false positive rate will cause a good deal of time to be wasted and will likely destroy confidence in the IDS.

$$Fitness = detection\ rate - false\ positive\ rate \tag{1}$$

Each individual in GP is represented by a tree. Here we use strongly-typed GP, which enforces data type constraints and whose use of genetic functions and generic data types [17]. In order to evaluate an individual we translate the individual tree to a C program.

Experimental Results The networks are simulated by ns-2 [3] where mobility patterns of the nodes on the network are created using BonnMotion [1]. Different network scenarios are created with different mobility levels and traffic loads. 50 nodes are placed in a topology of 1000m by 500m. TCP traffic is used for communication. The maximum number 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 chosen as the 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.

We evolve separate programs for each attack. Intrusion detection programs are distributed to each node on the network. Each node gathers the features every time interval. We assume that attacks are detected by the nodes that the attacks affect directly. 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. Table 2 shows the performance of the evolved program (the best individual of ten runs of GP) for each attack type on networks with varying mobility and traffic patterns.

Table 2. Performance of the Genetic Programming technique on simulated networks

Network Scenarios	**Flooding Attack**		**Route Disruption Attack**	
	DR	FPR	DR	FPR
low mobility low traffic	99.81%	0.34%	100%	0.51%
low mobility medium traffic	99.24%	1.94%	100%	0.99%
medium mobility low traffic	99.95%	0.36%	97.06%	0.46%
medium mobility medium traffic	99.89%	1.88%	100%	0.88%
high mobility low traffic	99.79%	0.66%	100%	0.52%
high mobility medium traffic	98.62%	1.83%	100%	0.84%

Some conclusions can be drawn from these figures. Apparently, route disruption attacks seem to be easier to detect than flooding attacks. In all cases but one the detection rate (DR) is 100% and the false positive rate (FPR) is less than 1%. Note that in the case with medium mobility and low traffic perfect detection is not reached, but the FPR is low (0.46%). It seems reasonable to suppose that a 100% DR can be achieved with a small increase in the FDR. The results for flooding attacks are slightly "worse", attaining in almost all cases detection rates higher than a 99% while keeping the FPR reasonably low. Note that in both attacks the main difficulty seems to come from the traffic load: regardless of the mobility patern, the FPR for medium traffic is higher than for low traffic. This is a common characteristic of any detection technique which does not achieve a perfect detection, as the higher the traffic to be analysed, the higher the FPR.

4.2 Multi-Objective Optimisation (MOO)

Multi-Objective Optimisation (MOO) aims to optimise two or more, often conflicting objectives simultaneously. The solution to multi-objective optimisation generally is not unique. It is the set of optimal solutions called the Pareto set. In Pareto efficiency, an objective vector x is said to dominate another objective vector y (x $\succ$ y) if no criterion of x is no greater than the corresponding component of y and at least one criterion is less (lesser values are preferable).

$$x \succ y : if\ x_i \leq y_i\ for\ each\ i\ and\ x_j < y_j\ for\ some\ j \tag{2}$$

The Pareto front compromises the solutions that are not dominated by any other individuals. In other words, it includes the optimal solutions (non-dominated) which represent different trade-offs among the objectives.

Multi-Objective Evolutionary Computation. Multi-objective evolutionary computation (MOEA) allows us to combine multi-objective optimisation with evolutionary search. In our research, we explore the trade-offs between the detection and false positive rate of evolved programs by using MOEA techniques. We might not discover the optimal solutions for both metrics by using the fitness function described in the equation (1). The fitness of an individual can be high due to high detection rate or low false positive rate, or both. For example, the fitness of a program with 90% DR and 2% FPR is the same with a program which has 100% DR and 12% FPR. Therefore, the fitness of an individual (evolved program) is represented by two separate objectives here: detection rate and false positive rate. We aim to discover a set of optimal solutions between the objectives detection rate and false positive rate of evolved programs by using MOEA techniques.

SPEA2 [35] is one of the most popular MOEA algorithms. An implementation of SPEA2 which is an extension to ECJ [2] is utilised in this research. Figure 1 shows the Pareto fronts for each attack, which demonstrate the optimal solutions at the end of 500 generations. Each chart shows [1-DR] versus FPR which are the metrics we want to minimise simultaneously. There is a clear trade-off between DR and FPR: while FPR decreases, DR decreases too.

5 Power-Aware Intrusion Detection in MANETs

Nodes on MANETs can vary from hand held devices such as PDAs, cell phones, and the like, to laptops that have different resource and computational capaci-

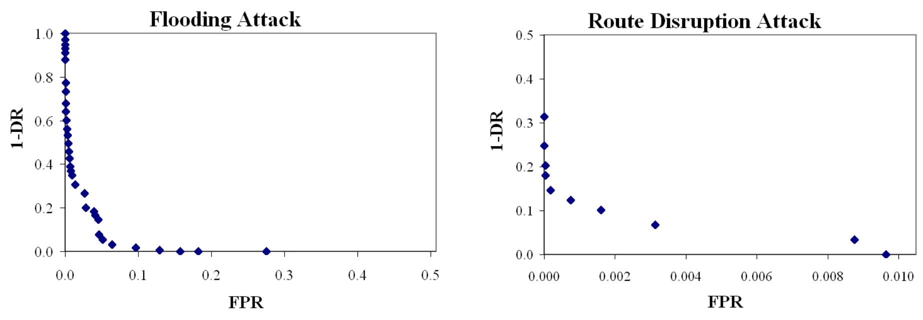


Fig. 1. Trade-offs between detection rate & false positive rate for each attack

ties. Moreover they usually run on battery power. The variety of mobile nodes generally with scarce resources affects proper working of intrusion detection systems running on these nodes. For instance, IDS agents might not be able to process every packet/alert due to limited resources. This is why efficiency is as important as effectiveness for intrusion detection in mobile networks. In the case of sensor networks, the power issue may become acute.

The proposed intrusion detection approaches for MANETs in the literature generally put the emphasis on IDS architecture which distributes functional tasks among nodes. Hence the resources used for IDS on the network are distributed as well. A hierarchical architecture is used by many proposed approaches. The network is divided into groups such as clusters, zones where some nodes (cluster heads, interzone nodes) have more responsibility than other nodes in the group. From the point of view of intrusion detection, each node in the cluster carries out local detection while cluster heads carry out global detection. While cluster heads are chosen based on some criteria such as connectivity, energy remaining, and the like in some approaches [24][13], other approaches make such choices randomly for security reasons [7][33]. Moreover, central management points are used in some approaches to do computationally intensive tasks like data mining [23].

In this research, we investigate evolving intrusion detection programs which also take into consideration the capability of nodes running these programs. We explore trade-offs between functional and non-functional properties of programs by using multi-objective evolutionary computation. Since power is one of the most crucial resources on mobile networks, both classification accuracy and energy consumption of programs are considered as objectives.

The Framework We employ multi-objective evolutionary computation techniques to optimise the following three objectives in our experiments: detection rate, false positive rate, and energy consumption of the program. To evaluate a program's energy consumption, we need to simulate the execution of the program. For that reason we use Wattch [10] which is a framework for analysing and optimising microprocessor power dissipation for specific architectures. Wattch integrates its power models with the SimpleScalar architectural simulator [4].

This new simulator utilising SimpleScalar and Wattch is called Sim-Wattch. We have made our power evaluations on Simple Scalar's PISA architecture.

Once again the ECJ implementation of SPEA2 algorithm is used to carry out multi-objective optimisation. In SPEA2, each individual in a population is represented by a tree structure. To analyse the energy consumption of an individual, we convert each individual to a C program and write to a file. In the transformation process from a tree to a C program, the functions used by the individuals and not included in the standard C library (e.g. percent function) are defined as macros. After the C file is created, it is compiled and run on the Sim-Wattch in order to simulate the execution of the program on PISA architecture and estimate the energy consumption of it. The energy consumption of the program together with its classification accuracy (the detection rate and the false positive rate) are assigned as the objectives of the individual. The individual takes place in the evolution process and survives in subsequent generations based on its performance on these objectives. The fitness function components are defined below. We aim to maximise these three objectives simultaneously.

$$f_1 = detection\ rate \tag{3}$$
$$f_2 = 1 - false\ positive\ rate \tag{4}$$
$$f_3 = 1/energy\ consumption \tag{5}$$

5.1 Experimental Results

Firstly we evolve programs to detect flooding and route disruption attacks on MANETs by using GP. The same GP parameters given in Table 1 are used. Only classification accuracy (1-(DR-FPR)) is targeted at this point. The best individuals of ten runs with their energy consumptions are given in Figure 2. This figure shows that while classification accuracy is high, energy consumption of the program gets higher as well for flooding attacks. On the other hand, this relation is not quite as straightforward for route disruption attacks. Analysing the best individuals evolved for route disruption attack shows that it is a simple attack and can be detected by small programs which generally have a tendency to consume lower energy. These experiments demonstrate that different trade-offs can be made between classification accuracy and energy consumption of the programs, and encourage us to find the acceptable trade-offs between these objectives using multi-objective optimisation. Furthermore, since the size of the programs can affect their energy consumption, we conduct experiments to evolve programs with different tree depths (17, 5). *Tree depth* defines the maximum size of the individuals (trees) evolved in GP. The effect of program size on evolved programs' detection ability and energy consumption can also be seen in Figure 2. Program size forces the programs to be smaller which can presumably result in less energy consumption. The programs evolved for route disruption attack with the same detection ability but lower energy consumption can be seen in the figure. The results are more dramatic for the flooding attack. It is seen how a good performance on detection of the attack can also be achieved with small-

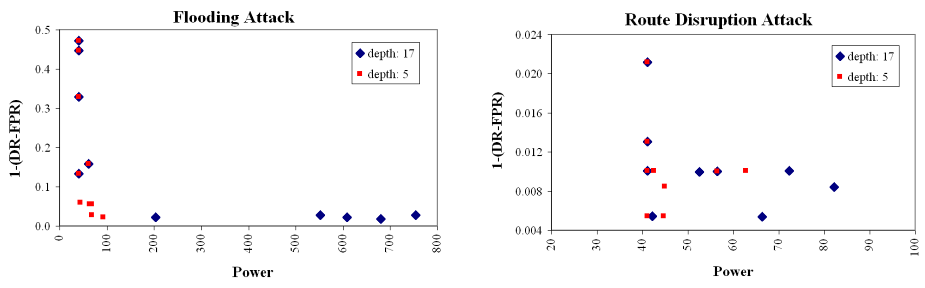
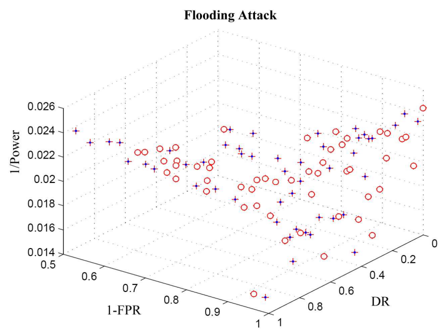


Fig. 2. Classification accuracy and energy consumption of the optimal evolved programs

sized programs. Nevertheless programs with bigger program size and accordingly higher energy consumption show a slightly better performance detection ability.

Another effect on the size of the individuals in GP is bloat. Bloat is a phenomenon whereby the size of individuals in a GP population increases dramatically over the duration of a run, largely due to redundant code [20]. The effect of bloat has also been noticed in our experiments where there are individuals with the same fitness but with different sizes due to code redundancy, which tends to evolve programs with higher energy consumption. Fortunately, "the archiving in SPEA2 is effectively elitist, and counteracts the emergence of bloat in GP, because a larger individual will only survive if it makes an improvement over the existing archive in at least one objective" [31]. This feature is very important in our experiments since it supports our goal to evolve small-sized programs, and programs with low energy consumption presumably.

In the second part of our experiments, we evolve programs for flooding and route disruption attacks separately by using multi-objective evolutionary computation. The parameters used are the same as in Table 1 except the population size (150) and SPEA2 archive size (100). Figure 3 shows the optimal solutions found for each attack at the end of 1000 generations. The circle points show the Pareto front. In the case of flooding attacks, Pareto front moves towards higher energy consumption for higher detection rate and lower false positive rate. It has been observed that it is the false positive rate that is most clearly affected by energy consumption. Allowing an increase in false positive rate causes decrease in energy consumption. For route disruption attacks, programs closer to optimum solution which have higher detection ability and lower energy consumption are achieved by using MOEA techniques. Moreover we have compared the energy consumption of programs which have highly-accurate detection ability with that of the programs evolved using GP in Figure 2. It is observed that programs with lower energy consumption stand out in the results obtained by MOEA techniques. Particularly for flooding attacks energy consumption is significantly reduced. Lastly we evolve programs to detect flooding and route disruption attacks together by using MOEA techniques. We aim to investigate if it is better to evolve one program to detect both attacks or evolve two programs each with half the resource usage. Figure 4 shows the 3D-Pareto front for the three objec-



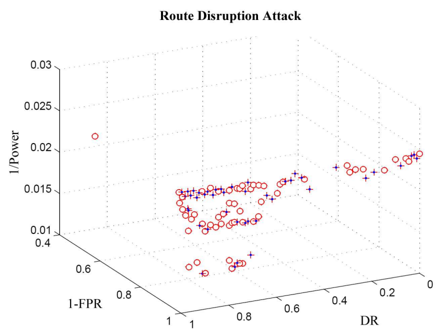


Fig. 3. 3D-Pareto front for detection of each attack with the three objectives: detection rate, false positive rate and energy consumption

tives. The results demonstrate that a detection program for both attacks can be more energy-efficient than two programs which detect these attacks separately, it does not show high classification accuracy as much as two programs do separately. There is a trade-off to be made based on the requirements of the MANET application used. In the results of five runs, there is no program evolved for detecting both attacks which simultaneously has detection rate and false positive rate (1-FPR) more than 94%. Table 3 shows some example programs evolved using MOEA. (There are many other programs on the Pareto front which have different trade-offs.)

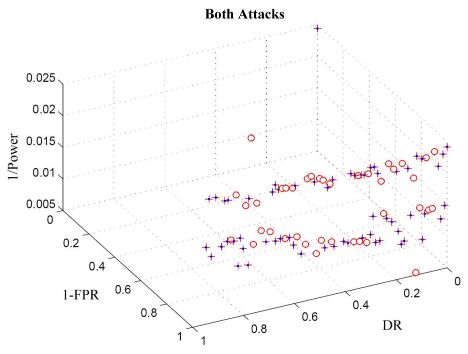


Fig. 4. 3D-Pareto front for detection both attacks with the three objectives: detection rate, false positive rate and energy consumption

Table 3. Example programs evolved by MOEA for each attack

Attack Type	**Evolved Program**	**DR**	**FPR**	**Energy Usage**
Flooding	(frw_aodvPs * frw_aodvPs) $>$ (4log(neighbours) + 5updated_routes)	98.65%	1.23%	65.42
Route Disruption	((2updated_routes - 2recv_aodvPs + active_routes) * recv_rrepPs $>$ (recv_aodvPs + updated_routes)	100%	0.63%	43.05
Both	(((updated_routes * init_aodvPs) $\leq$ frw_rreqPs) && (init_rrepPs $\neq$ recv_rrepPs) && (exp(updated_routes) $\neq$ recv_rrepPs)) $\|$ (updated_routes $<$ frw_rreqPs)	93.29%	4.65%	50.14

6 Conclusions and Future Work

We have evolved programs using GP to detect known flooding and route disruption attacks against AODV and have evaluated them on simulated networks with varying mobility and traffic levels. We used both single fitness functions and multiple fitness functions. We have shown how in some circumstances a multiple objective approach provides a more effective means of searching the tradeoff space. Our work is unusual in that we trade off security performance (detection and false positive rates) against resources (power). It is likely that for some types of networks (e.g. sensor networks) the ability to make good tradeoffs will be particularly important. Our techniques can be used to generate solution sets with the best (or near best) tradeoffs possible. A final choice between solutions making different tradeoffs rests with the designer. The inherent complexity of MANET operations makes it difficult to see how IDS programs with optimal tradeoffs could be obtained by standard system development practices. An optimisation based approach seems a natural and effective candidate for the problem we seek to solve. We recommend the use of GP and MOEA for further consideration by the IDS and MANET research communities.

References

1. Bonnmotion: A mobility scenario generation and analysis tool. http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/.
2. Ecj18: A java-based evolutionary computation research system. http://cs.gmu.edu/ eclab/projects/ecj/.
3. Ns-2: The network simulator. http://www.isi.edu/nsnam/ns.
4. Simplescalar. http://www.simplescalar.com/.
5. A. Abraham and C. Grosan. Evolving intrusion detection systems. In *Genetic Systems Programming: Theory and Experiences*, volume 13, pages 57–79. Springer, 2006.
6. A. Abraham, C. Grosan, and C. Martiv-Vide. Evolutionary design of intrusion detection programs. *Int. Journal of Network Security*, 4:328–339, 2007.

7. Y. an Huang, W. Fan, W. Lee, and P. S. Yu. Cross-feature analysis for detection ad-hoc routing anomalies. In *In Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS).*
8. T. Anantvalee and J. Wu. *A Survey on Intrusion Detection in Mobile Ad Hoc Networks*, chapter 7, pages 159–180. Springer, 2007.
9. 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.
10. D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *In Proceedings of the 27th International Symposiyum on Computer Architecture (ISCA-27)*, 2000.
11. D. Denning. An intrusion detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987.
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. 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.
14. Y. Liu, K. Chen, X. Liao, and W. Zhang. A genetic clustering method for intrusion detection. *Pattern Recognition*, 37, 2004.
15. 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.
16. L. Me. Gassata, a genetic algorithm as an alternative tool for security audit trails analysis. In *In Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID'98)*, 1998.
17. D. J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3:199–230, 1995.
18. J. Parker, J. Undercoffer, J. Pinkston, and A. Joshi. On intrusion detection and response for mobile ad hoc networks. In *23th IEEE Int. Performance Computing and Communications Conference Proceedings*, 2004.
19. C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computer Systems and Applications Proceedings*, pages 90–100, 1999.
20. 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.
21. S. Sen and J. A. Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks. In *In Proc. of Second ACM Conference on Wireless Network Security (WiSec'09)*, 2009.
22. S. Sen and J. A. Clark. *Intrusion Detection in Mobile Ad Hoc Networks*, chapter 17, pages 427–454. Springer, 2009.
23. 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.
24. D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C.-Y. Tseng, and T. Bowen. A general cooperative intrusion detection architecture for manets. In *In Proceedings of the 3rd International Workshop on Information Assurance*, pages 57–70, 2005.

25. 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.
26. 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.
27. 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.
28. P. Uppuluri and R. Sekar. Experiences with specification-based intrusion detection. In *In Proc. of the Recent Advances in Intrusion Detection (RAID'01), LNCS 2212*, pages 172–189. Springer, 2001.
29. 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.
30. T. Weise. Genetic programming for sensor networks. Technical report, 2006.
31. D. R. White, J. Clark, J. Jacob, and S. Poulding. Evolving software in the presence of resource constraints. In *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'08)*. Springer, 2008.
32. 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.
33. P. Yi, Y. Zhong, and S. Zhang. A novel intrusion detection method for mobile ad hoc networks. In *In Proceedings of Advances in Grid Computing (EGC'05), LNCS 3470*.
34. Y. Zhang, W. Lee, and Y. an Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks Journal (ACM WINET)*, 2(5), September 2003.
35. E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Swiss Federal Institute of Technology.

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