A Survey of Intrusion Detection Systems using Evolutionary Computation

Sevil Sen*

Department of Computer Engineering,
Hacettepe University, Ankara, TURKEY
e-mail : ssen@cs.hacettepe.edu.tr

Abstract

Intrusion detection is an indispensable part of a security system. Since new attacks are emerging every day, intrusion detection systems (IDS) play a key role in identifying possible attacks to the system and giving proper responses. IDSs should adapt to these new attacks and attack strategies, and continuously improve. How to develop effective, efficient and adaptive intrusion detection systems is a question that researchers have been working on for decades. Researchers have been exploring the suitability of different techniques to this research domain. The evolutionary computation inspired from natural evolution is one of the approaches increasingly studied. Some characteristics such as producing readable outputs for security experts, producing lightweight solutions, providing a set of solutions with different trade-offs between conflict objectives, make these techniques a promising candidate for the problem. In this study, we survey the proposed intrusion detection approaches based on evolutionary computation techniques found in the literature. Each major research area on intrusion detection is investigated thoroughly from the evolutionary computation point of view. Possible future research directions are also summarized for researchers.

Keywords: network security, intrusion detection, evolutionary computation, genetic programming, genetic algorithms, grammatical evolution, multi-objective evolutionary computation.

**corresponding author*
Department of Computer Engineering, Hacettepe University
06800 Ankara/TURKEY
Tel : +90 312 297 75 00
Fax : +90 312 297 75 02

1. Introduction

Intrusion detection systems (IDSs), aptly called the 'second line of defense', play a key role in providing comprehensive security. Since it is difficult to develop a complete solution for the prevention of attacks, especially on complex systems, and attackers are always trying to find new ways to bypass these prevention mechanisms, intrusion detection systems have become an inevitable component of security systems. IDS comes into the picture after an intrusion has occurred. The main roles of an intrusion detection system are to detect possible threats to the systems, and to give proper responses such as notifying security experts, terminating damaging network connections, and other similar means.

Intrusion detection has been a popular research topic in the security field since Denning proposed an intrusion detection model in 1987 (Denning, 1987). Many techniques have been introduced to detect intrusions effectively and efficiently; so the security goals of a system -confidentiality, integrity, and availability- could be satisfied. Researchers have been working on finding answers to the following questions: How to detect attacks effectively and efficiently, which responses to give against detected attacks, how to continuously adapt to new attack strategies, and such like. Major research areas on intrusion detections could be summarized as follows (Lundin and Jonsson, 2002): foundations, data collection, detection methods, response, IDS environment and architecture, IDS security, testing and evaluation, operational aspects, and social aspects. In this study, we examine the evolutionary computation-based approaches proposed for each research area on intrusion detection.

Evolutionary computation is a subfield of artificial intelligence inspired from natural evolution. It has been successfully applied to many research areas such as software testing, computer networks, medicine, and art. Intrusion detection is the most studied area in the security domain, and various intrusion detection techniques already exist in the literature. The following characteristics of evolutionary computation attract researchers to investigate these techniques on intrusion detection: generating readable outputs by security experts, ease of representation, producing lightweight solutions, and creating a set of solutions providing different trade-offs between conflict objectives, such as detection rate vs. power usage. Furthermore, EC does not require assumptions about the solution space (Fogel, 2000). There are many promising applications of evolutionary computation on intrusion detection. It is especially suitable for resource-constrained and highly dynamic environments, due to their need of solutions satisfying multiple objectives. In this study, the main proposed solutions in the literature are looked at in detail. For example, how candidate solutions are represented, how evolved solutions are evaluated, which datasets are used, what advantages and disadvantages the proposed solutions have, are all presented.

Although some areas, such as detection methods, have already been extensively studied, there are only a few studies on areas such as testing and evaluation, and response. This study covers all research areas of intrusion detection from the evolutionary computation point of view. The suitability of proposed

solutions is discussed for each problem. Furthermore, some future directions for researchers are given at the end of the study. To sum up, this research outlines the main issues of intrusion detection and the proposed solutions based on evolutionary computation in the literature, and discusses the potential of evolutionary computation for intrusion detection.

The chapter is organized as follows. The fundamentals of intrusion detection, the possible research areas on intrusion detection are presented in Section 2, and an introduction to evolutionary computation is given in Section 3. Section 4 then outlines the evolutionary computation-based approaches proposed for intrusion detection; classified according to the research area they contribute to. Finally, the conclusions of the study and the future directions for researchers are summarized in Section 5.

2. Intrusion Detection Systems

Intrusion detection system (IDS) is an indispensable part of network security. It is introduced as a system for detecting intrusions that attempt to compromise the main security goals, confidentiality, integrity, and availability, of a resource. The development of an IDS is motivated by the following factors:

- Most existing systems have security flaws that render them susceptible to intrusions. Finding and fixing all these deficiencies is not feasible (Denning, 1987), and in particular, complex systems are prone to errors which could be exploited by malicious users;
- Prevention techniques are not sufficient. It is almost impossible to have an absolutely secure system (Denning, 1987). IDS comes into the picture when an intrusion has occurred and cannot be prevented by existing security systems;
- Since insider threats are generally carried out by authorized users, even the most secure systems are susceptible to insiders. Furthermore, many organizations express that threats from inside can be much more harmful than outsider attacks (CERT, 2011);
- New intrusions continually emerge. Therefore security solutions need to be improved or introduced to defend our systems against novel attacks. This is what makes intrusion detection such an active research area.

An IDS detects possible violations of a security policy by monitoring system activities and respond these violations according to the policy. An IDS could be called host-based IDS (HIDS) or network-based IDS (NIDS) according to the system that it monitors. If an attack is detected when it enters the network, a response can be initiated to prevent or to minimize damage to the system. Moreover, the prevention techniques could be improved with the feedback acquired from intrusion detection systems. Security solutions do not operate on their own as used to be the way. Nowadays, prevention, detection and response mechanisms generally communicate with each other in order to protect the system from complex attacks.

There are generally two metrics employed in order to evaluate intrusion detection systems: detection rate and false positive rate. Detection rate represents the ratio of malicious activities detected to all malicious activities. A missed intrusion could result in severe damage to the system. False positives indicate normal activities which are falsely detected as malicious by IDS. A low false positive rate is just as important as a high detection rate. When an intrusion is detected, it usually raises an alarm to the system administrator. High false positives result in excessive burden to the administrator and as a result, might not be analyzed by security experts in real time. Another metric called intrusion capability metric (C_{ID}) was introduced in 2006 in order to evaluate intrusion detection systems (Gu et al., 2006). The authors define C_{ID} as the ratio of the mutual information between IDS input and output to the entropy of the input. It naturally includes both the detection rate and the false positive rate. Even though many approaches still use the conventional intrusion detection metrics (*i.e.* detection and false positive rates), C_{ID} has important characteristics to compare IDSs and, it is expected to be more commonplace in the near future.

2.1. IDS Components

The three main components of an intrusion detection system, data collection, detection, and response, are depicted in Figure 1. The data collection component is responsible for the collection and pre-processing of data tasks, such as transforming data to a common format, data storage, and sending data to the detection module (Lundin and Jonsson, 2002). Various data from different sources such as system logs, network packets, MIB (Management Information Base) data could be collected and formatted to send to the intrusion detection module.

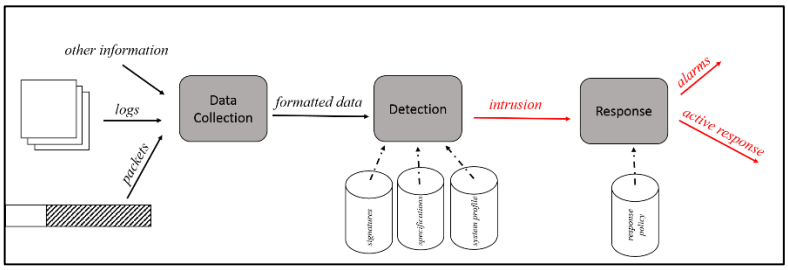

Figure 1. *IDS Components*

The detection module analyzes and processes the formatted data obtained from the data collection model in order to detect intrusion attempts, and forwards the events flagged as malicious to the response module. There are three intrusion detection techniques: anomaly-based, misuse-based, and specification-based. Anomaly-based intrusion detection technique defines the normal behaviors of the system, such as usage frequency of commands or system calls, resource usage for programs etc. The activities falling out of the normal behaviors of the system are labelled as intrusions. Various techniques have been applied for anomaly detection, such as classification-based (*e.g.* neural networks, naive Bayes, support vector machines), clustering-based techniques. Since the normal behavior could change

over time, one of the biggest challenges in this approach is to define the normal behavior of a system. It is particularly challenging in highly dynamic networks, such as mobile ad hoc networks (MANETs), vehicular ad hoc networks (VANETs). Another disadvantage of this technique is the high number of false positives. How to update the system profile automatically is another challenge. Concept drift, the problem of distinguishing malicious behaviors from the natural change in user/system behaviors, is an issue in anomaly-based detection systems. The conventional approaches mainly overcome this issue through the updating of user/system profiles. It is particularly crucial for the ongoing detection of attackers. The updating system generally uses unlabeled data in retraining due to the large amount of data. Therefore, the updating system has to trust the decisions that the anomaly-based detection system makes. For instance, if the detector misses an intrusive behavior, it will be added to the training data as benign datum. The ability of their adaptation to the concept drift depends on the accuracy of the detector. The authors showed that misclassified instances included in updating could considerably decrease the performance of anomaly-based detection approaches (Sen, 2014).

Misuse-based (or signature-based) intrusion detection systems are based on defined signatures in order to detect known attacks. It is the most commercially employed approach due to its efficiency. Although it has a low false positive rate, the biggest disadvantage of this approach is that it cannot detect novel attacks and unknown variants of existing attacks. Many proposed approaches have a low resilience against even the simplest obfuscation techniques. Another issue is to frequently update an attack signatures database. Since large numbers of attacks are introduced every day, the function of automatically generating new signatures is an essential characteristic of an IDS. Nowadays, both misuse-based and anomaly-based intrusion detection techniques are employed together. While the misuse-based systems are efficient in detecting known attacks, anomaly-based detection systems are employed to detect attacks missed from these systems.

The last intrusion detection technique is a specification-based method, in which attacks are detected as violations of well-defined specifications of a program/protocol. Since its introduction in 2001 (Uppuluri and Sekar, 2001), this technique has mainly been used for ad hoc networks. It both detects known and unknown attacks with a low false positive rate (Uppuluri and Sekar, 2001). Since the routing protocols proposed for ad hoc networks are vulnerable to attacks, due to their dynamic and collaborative nature, the specification-based intrusion detection is quite suitable for such networks. It is the most employed technique in ad hoc networks and, proposed as a way for different types of ad hoc routing protocols to be kept up to date. However this technique cannot detect Denial of Service (DoS) attacks, since these types of attacks follow the system specifications. Generally, it cannot detect legitimate activities, even if they are unusual (Uppuluri and Sekar, 2001). Another disadvantage of this technique is the requirement to define specifications for each protocol used in the system. Therefore it does not attract too much interest in wired networks due to this time consuming task requirement.

When an event is classified as malicious, it is sent to the response module. The module behaves according to the response policy defined. Intrusion detection responses are divided into two groups (Axelsson, 2000): active and passive responses. There are still many systems which give only passive responses; notifying the proper authority. On the other hand, the damage of an intrusion is tried to be mitigated or prevented by controlling either the attacked system or the attacking system in an active response (Axelsson, 2000). Blocking the IP address attacking the system, or terminating network connections for a while, are examples of commonly used active responses. These types of system are typically called Intrusion Prevention Systems (IPSs).

2.2. Research Areas and Challenges in Intrusion Detection

Intrusion detection has been an appealing research area since Denning first introduced a formal model for the problem (Denning, 1987). Intrusion detection is a challenging research area due to its very nature and a great deal of research has emerged in this domain. Lundin *et al.* (Lundin and Jonsson, 2002) classify major research areas on intrusion detection as follows: foundations, data collection, detection methods, response, IDS environment and architecture, IDS security, testing and evaluation, operational aspects, and social aspects.

Foundations cover the research carried out on intrusions, intruders, and vulnerability. The main challenge here is to update intrusion detection systems against emerging new attacks every day. A good IDS must perform continuous adaptation to new attacks, changes in the system, and the like. *Data collection* deals with selecting data sources and features, how to collect data, logging and formatting data. One of the main problems of IDSs is to analyze and process highly imbalanced and large amounts of network data efficiently. Researchers mainly work on selecting appropriate features for intrusion detection and, reducing redundant features. The majority of research has been carried out on *detection method*s. The main challenges of each detection techniques are given in detail in the previous section. Difficulty of differing normal data from abnormal data and, developing systems that are robust against unknown attacks are among the most important ones. Studies on *Response* aim to answer the following questions: how to respond to detected intrusions (i.e. passively or actively, temporarily or permanently), and how to represent detected intrusions to the proper authority.

How to distribute IDS agents and facilitate interoperability between IDS agents are sub research areas in *IDS environment and architecture*. It is a particularly active research area in networks with a lack of central points where we could monitor and analyze all network data. Three main intrusion detection architectures are proposed for such networks: stand-alone, distributed and collaborative, and hierarchical. There are few studies that define information exchanges between IDS agents. Mobile agents, which carry both data and software from one system to another system autonomously and continue its execution on the destination system, are another way of communication with many advantages, such as reducing the network load, and adapting dynamically (Lange and Oshima, 1999).

IDS security is related to protecting IDS communication and IDS itself from attacks. This 'secure security' concept is especially important in critical domains such as healthcare and tactical systems. The studies on this immature research area have accelerated in recent years, and a survey on adversarial attacks against IDSs was recently proposed (Corona et al., 2013). *Testing and evaluation* takes into account how to evaluate IDSs. There are many comparisons available in the literature. The KDD dataset (Lippman et al., 2000) is considered as benchmark data in these studies. *Operational aspects* cover technical issues such as maintenance, portability, upgradeability of IDSs. *Social aspects* are related to ethical and legal issues of deploying IDSs (Lundin and Jonsson, 2002). Operational and social aspects are excluded due to their irrelevance in this study.

3. The Method: Evolutionary Computation

Evolutionary computation (EC) is a computational intelligence technique inspired from natural evolution. An EC algorithm starts with creating a population consisting of individuals which represent solutions to the problem. The first population could be created randomly or fed into the algorithm. Individuals are evaluated with a fitness function, and the output of the function shows how well this individual solves or comes close to solving the problem. Then some operators inspired from natural evolution such as crossover, mutation, selection, and reproduction are applied to individuals. Based on the fitness values of newly evolved individuals, a new population is generated. Since the population size has to be preserved as in nature, some individuals are eliminated. This process goes on until the termination criteria is met. Reaching the number of generations defined is the most used criteria to stop the algorithm. The best individual with the highest fitness value is selected as the solution. The general steps of an EC algorithm is shown below.

```
initilize population
evaluate the fitness value of each individual
while the optimal solution is not found and
      the number of generations defined is not reached
            select parents
            apply genetic operators to the selected individuals
            evaluate fitness values of new individuals
            select individuals for the next generation
end while
return the best individual
```

There are various EC techniques such as Genetic Programming (GP), Genetic Algorithms (GA), Grammatical Evolution (GE), Evolutionary Algorithms (EA), and the like. These techniques generally differ from each other based on how to represent the individuals. For example, while GP uses trees, GE uses BNF grammar in order to define individuals.

One of the most popular EC techniques in the literature is GP. Since introduced by Koza (1992), it has been applied to many problems, and shown that GP produces better solutions for complex problems than humans do. In GP, crossover operator swaps subtrees of two individuals, mutation operator

exchange a subtree with another tree created. One of the problems in GP is bloating which is the uncontrolled growth of the average size of trees in a population. It is generally controlled by limiting the depth of the individuals. However bloating could show a positive effect on some problems. While GP uses trees, genetic algorithms (GA) represent each individual as an array of bits called chromosomes. In GA, the genetic operators are applied on subarrays of individuals selected.

Grammatical evolution (GE) is a technique, inspired largely by the biological process of generating a protein from the genetic material of an organism, which allows us to generate complete programs in an arbitrary language by evolving programs written in a BNF grammar (Ryan *et al*., 1998). However GE technique performs the evolution process on variable-length binary strings, not on the actual programs (O'Neill and Ryan, 2003). This transformation from variable-length binary strings to actual programs, provides mapping from the genotype to the phenotype as in molecular biology. One of the benefits of GE is that this mapping simplifies the application of search to different programming languages and other structures. Another benefit of GE from the security point of view is the production of readable outputs in well-defined grammars.

Multi-objective evolutionary computation (MOEC) is used to create solutions satisfying more than one objective. It creates a set of solutions providing different trade-offs between objectives. Another way to achieve that is to create a weighted fitness function. Since the population-based nature of evolutionary algorithms allows the generation of several elements of the Pareto optimal set in a single run, evolutionary algorithms are highly motivated for solving problems with multi-objectives (Coello et al., 2007).

4. Evolutionary Computation Applications on Intrusion Detection

The section outlines and discusses some representative examples of the EC applications to intrusion detection. Each solution is classified according to the subfield of intrusion detection it contributes to. Please note that some solutions could be placed in more than one subfield.

4.1. Foundations

This section covers the studies carried out on intrusions. Attackers have two prime motivations: to damage the targeted system and to avoid being identified. In order to achieve their goals, they continuously create new attacking strategies. On the other hand, intrusion detection systems generally are unable to detect these 'new' attacks. Particularly, misuse-based detection systems are ineffective against new attacks or unknown variants of existing attacks. Therefore, researchers have been also working on developing new variants of existing attacks automatically, using evolutionary computation techniques, and as a result, security solutions could be reassessed and strengthened. As far as we know, the first work on this area was proposed by (Kayacik et al., 2005a). The authors aim to evolve successful stack overflow attacks by employing grammatical evolution. The problem is represented as a simple C

program that determines the size of the NoOP sled; the main evasion technique used by attackers, the offset and the number of desired return addresses which state the address of the shellcode aimed to be executed by the attacker. Six criteria based on the success of the attack, the size of the NoOP sled, and the accuracy of the desired return address are incorporated in the fitness function. In some experiments, they also utilize fitness sharing in order to eliminate similar individuals in the population. Moreover, they carry out some experiments in order to discourage long NoOP sleds which helps evasion of the evolved attacks. Snort (Roesch, 1999; Snort, 2014) is chosen as an exemplar IDS in order to evaluate their results. The results show that GE techniques are applied successfully in order to generate both successful and evasive stack overflow attacks. It is observed that the number of invalid programs among the evolved ones is quite small. However it is stated that GE representation of malicious programs was not good enough to modify register references (Kayacik et al., 2005b). Therefore the same authors employ Linear GP (LGP) to generate new buffer overflow attacks. At this time, the malicious programs are represented in Assembly language. Instructions (opcodes and their operands) are represented by LGP. The positive effect of bloating property of GP is observed in the results. While bloating is generally aimed to be prevented to increase readability of GP outputs in other domains, it is used here to hide malicious code parts in evolved programs. The authors also analyze the effects of different instruction sets (arithmetic, logic, etc.) in the representation to the results.

Linear GP is also used to automatically generate mimicry attacks, in which the exploits are represented as a sequence of system calls (Kayacik et al., 2009). The proposed method follows a black-box approach in which the attacker only has information about the output of the anomaly-based IDS. Four anomaly-based IDSs (Stide, Process Homeostasis, Process Homeostasis with Scheme Masks, and Markov Mode) are employed to obtain the anomaly rate used in the fitness function, together with the attack success ratio and attack length. The results show that mimicry attacks evolved with GP produce lower anomaly rates than the original attacks. Nonetheless, none of the attacks produced were completely undetected. An extended study compares the evolved attacks based on the black-box approach, with the created attacks based on the white-box approach in which the attacker has the knowledge of the internal behavior of IDS (Kayacik et al., 2011). They also include one additional objective into the fitness function: delay (a particular type of response against an identified attack). Even a detector that has a low anomaly rate could prevent an attack by deferring the attack with long delays. The evolved attacks have comparable results with the white-box approach, but the latter produces lower anomaly rates. Nonetheless, the black-box approach generates many attacks with different trade-offs in a Pareto front, while the white-box approach has one exploit per attack, detector pair. Finally, an attacker might not have easy access to the detector where he could cost-effectively gain information about its internal behavior. The features make EC attractive for evolving evasive attacks, and are summarized as ease of representation, multi-objective optimization, and natural obfuscation. Besides the studies given here, in the literature, there are applications of EC on generating variants of known malwares (Noreen et al., 2009).

4.2. Data Collection

Studies applying evolutionary computation techniques in this research domain mainly focus on feature selection and feature reduction. The choice of features is very important for any research domain. On the one hand, the features generally contain sufficient and expressive information that is enough to generate effective models; but on the other hand, too many features could in fact confuse the learning algorithm and degrade its performance. Therefore, superfluous features should be eliminated while keeping necessary ones. This elimination speeds up both the feature extraction, by only spending time to get necessary features, and the training process by reducing the search space. From the intrusion detection point of view, features could also give some information about attack and attackers' behaviors. If we reduce the number of features, we could better understand the attack motives and techniques.

We mainly divide feature selection approaches into two groups: filter and wrapper approaches. Wrapper approach selects features according to the performance of the learning algorithm. Filter approach only gives general insights into the features based on some measures, and does not take into account the performance of the algorithms. The studies, based on feature selection for intrusion detection in the literature, mainly follow the wrapper approach.

As far as we know, the first wrapper approach using genetic algorithms for intrusion detection was proposed in (Helmer et al., 1999), and extended by the same authors in (Helmer et al, 2002). The authors used RIPPER algorithm for learning and showed that the performance of the algorithm is not affected even when the features used in training are reduced to half. In (Hofmann et al., 2004), the number of features is decreased from 187 to 8 features by employing evolutionary algorithms. In (Kim et al.,2005), GA is employed to select both the optimal features and the optimal parameters for a kernel function of SVM for detecting DoS attacks on the KDD dataset (Lippman et al., 2000). The results provided a better detection rate than the approaches that only adopted SVM in the IDS. A similar approach in order to improve the performance of SVM, by reducing the number of features, is also proposed for detection of a specific DoS attack against ad hoc networks (Sen and Dogmus, 2012). A recent GA application (Ahmad et al., 2014) works on principal components instead of working on features directly in order to both increase the performance of SVM and to use less number of features. Principal components are computed using Principal Component Analysis (PCA), a conventional technique for feature subset selection.

Evolutionary computation techniques for feature selection/reduction could also be employed with or after other techniques. In some approaches, it is combined with a filter approach, CFS (correlation-based feature selection) (Shazzad and Park, 2005; Nguyen et al., 2010), especially in the presence of a high number of features, as in the KDD dataset. An approach to increase the performance of SVM is introduced (Shazzad and Park, 2005) in which the feature set is first evaluated by a filter method called

CFS. There is also GA-based wrapper approaches proposed for different classification techniques such as decision trees (C4.5) (Nguyen et al., 2010; Stein et al., 2005), BayesNet (Nguyen et al., 2010), artificial neural networks (Mukamala et al., 2004), fuzzy data mining (Bridges and Vaughn, 2000) and the like.

The authors determine the weight of features for k-nearest neighbor classifier in (Middlemiss and Dick, 2003). They run GA many times and get the average of the weighted feature sets. Unlike feature selection, they include all features in the training since the final averaged feature set does not include any zero weights. They analyze the top five ranked features for each attack class in the KDD dataset. Moreover, they carry out additional experiments to remove the five features with the highest number of zero weights. However they show that removing zero-weighted features decreases the performance of the classifier. This situation could be discovered by changing the fitness function by taking the number of features removed into account. Furthermore, we could apply EC-based approaches in order to eliminate costly features which require a considerable amount of time to compute, especially when monitoring a large amount of traffic. In order to analyze features, Mukkamala *et al.* (2004) removes one feature out in each run of LGP. The results help to see the effects of each feature on the results.

4.3. Detection Techniques and Response

A great deal of research has emerged in the intrusion detection field, however the majority of research has been carried out on detection techniques. Various techniques such as statistical approaches, expert systems, SVM (support vector machines), ANN (artificial neural networks), evolutionary computation, and the like have been proposed for the problem. In recent years, computational intelligence methods have attracted a considerable interest due to their characteristics suitable for intrusion detection such as adaptation, fault tolerance, high computational speed and error resilience in the face of noisy information (Wu and Banzhaf, 2010). In this section, we focus on evolutionary computation techniques proposed in the literature. Evolutionary computation is one of the promising approaches on intrusion detection due to some advantages over other classical mechanisms: producing lightweight detection rules and the simple output which is understandable by security experts (Orfila et al., 2009).

4.3.1. Intrusion Detection on Conventional Networks

The first genetic programming (GP) application to intrusion detection is given by Crosbie and Stafford in 1995 (Crosbie and Stafford, 1995). The main idea in that research is to train autonomous agents based on the features related to network connections and the functions (arithmetic, logical, conditional) given to detect intrusive behaviors. An agent is evaluated with a fitness function which compares the output of the agent with the expected output. A weight parameter is also included in the fitness function to penalize the agents based on the difficulty of detecting an intrusion. Obvious intrusions that are misclassified during the evolution process are heavily penalized. This system is proposed to detect port flooding, port walking, probing, and password cracking attacks.

Applications of evolutionary computation techniques to intrusion detection on conventional networks have usually employed either genetic programming (GP) or a genetic algorithm (GA). GASSATA (Me, 1998) is one of the earliest works that employ GA to intrusion detection. It is a misuse-based detection system, using GA in order to detect 24 known attacks which are represented as sets of events (i.e. user commands). Since the system does not locate attacks precisely, a security expert is needed to analyze the audit trail and to pinpoint the attacks. As an improvement on GASSATA, a host-based IDS was introduced by (Diaz-Gomez et. al., 2005a). The authors propose an improved fitness function which is more general and independent of the audit trail. The results show that the model evolved with the improved fitness function produce no false positives and a low number of false negatives. The mathematical justification of the fitness function proposed is presented by (Diaz-Gomez et. al., 2005b).

One of the first proposals is the Network Exploration Detection Analyst Assistant (NEDAA), another GA-based approach (Sinclair et al.,1999). In NEDAA, automatically generated intrusion detection rules by GA and decision trees are fed into to a deployed IDS. In the GA component, a rule (a chromosome) is represented by the source IP address and port, the destination IP and port, and the protocol. Each value in the chromosome could be a number specified in the range or a wild card. A simple chromosome representation for a rule (source ip: 193.140.216.*, source port: 1454, destination port: 53, protocol: 2-TCP) is shown in Figure 2. The fitness function is evaluated based on the performance of each rule on a pre-classified dataset (normal and anomalous connections). However this GA application generates only one rule in each run. Since one single rule is not enough to identify different types of anomalous connections, the authors transfer the problem from finding global maxima to multiple local maxima of the fitness function by employing niching techniques. The rules generated by GA are compared with the rules generated by using decision trees in this study. While GA produces a set of rules, decision trees generate a single meta-rule with a number of different rules. This study investigates the use of GA on generating intrusion detection rules automatically, however it does not present any experimental results. The authors discuss future plans on the alteration of GA component in NEDAA such as finding complex rules in order to detect advanced attacks disseminated in space and time, generating rule chaining, and generating dynamic rules in order to detect new attacks.

1	9	3	1	4	0	2	1	6	*	1	4	5	4	5	3	2

Figure 2. *A simple chromosome representation of a rule*

The first GE application to detect attacks is employed on the KDD dataset (Wilson and Kaur, 2007). Abraham and Grosan (2007) employ two GP techniques, namely Linear GP (LGP) and Multi Expression Programming (MEP) on the same dataset. While MEP, a technique allows to encode multiple expressions, is more successful on identification some attack types, LGP is better in detection of other types. Tahta *et al.* (2014) employ GP in order to differentiate malicious peers from benign ones in peer-

to-peer (P2P) networks. They run P2P simulation for each individual to see how derived solutions are effective in preventing malicious peers from participating in the network.

The techniques based on artificial intelligence have the ability to solve complex problems cost-effectively. However they face training a model on imbalanced and large datasets in intrusion detection. Some researchers investigate the suitability of EC to work on large datasets (Dam et al., 2005). Another study proposes to use RSS-DSS (random subset selection - dynamic subset selection) algorithms in order to train GP computationally efficient (Song et al., 2005). The solution is useful in terms of both fitting training data into the memory and processing large amounts of data. RSS algorithm randomly selects a block of data from KDD which includes approximately half a million patterns. Then a subset of the block is processed with DSS algorithm, then this subset is given to the GP algorithm for evolution. The subset selection process is parameterized. The results make GP algorithm a very practical solution for intrusion detection by showing that performing one run takes only 15 minutes on a PC. The authors also analyze different fitness functions based on the recognition that different types of attacks are not uniformly distributed in the dataset. Using parallel genetic algorithms is another way of speeding training time up for complex problems with large datasets (Abadeh et al., 2007a).

A system which evolves attack signatures, by using GP, is proposed in (Lu and Traore, 2004). The results show that the derived rules are better at detecting both known and unknown attacks. Another signature-based intrusion detection was proposed recently (Gomez et al., 2013), which generates attack signatures automatically and works in an integrated manner with Snort. Multi-objective evolutionary algorithms are employed to obtain a set of solutions providing different trade-offs between false positives and false negatives. The performance of a weighted single fitness function is also shown in the results. Another recent approach uses Genetic Network Programming (GNP) in order to develop models both for misuse-based detection and anomaly-based detection (Mabu et al., 2011). GNP uses graphs in order to represent individuals in evolutionary computation. Evolutionary computation could be integrated with other techniques. For example, clustering genetic algorithm introduced in (Zhao et al., 2005) classifies activities into groups by employing clustering techniques in the first phase, then employs GA in order to distinguish normal activities from abnormal ones in the clusters. Another example is to represent individuals as fuzzy if-else rules, and then apply GA on these rules (Abadeh et al., 2007b). These are just some representative examples among the many EC applications to intrusion detection to be found in the literature.

4.3.2. Intrusion Detection on Wireless and Resource-Constrained Networks

Although evolutionary computation techniques have many potential application areas for wireless and ad hoc networks such as data aggregation, forming clusters and security, there are few studies on intrusion detection. Especially, MOO (multi-objective optimization) techniques are an apt candidate for problems on ad hoc networks, due to their very characteristics such as nodes with limited energy,

dynamic topology and the like (Sen and Clark, 2009a). Hence, the possible trade-offs to make between functional (*i.e.* accuracy) and non-functional (*i.e.* power usage, bandwidth) properties of solutions could exist. Please also note that the solutions proposed for these highly-constrained and dynamic networks should be lightweight.

The first work based on evolutionary computation techniques proposed for intrusion detection in mobile ad hoc networks uses grammatical evolution in order to detect attacks against mobile ad hoc networks such as dropping, ad hoc flooding, and route disruption attacks (Sen and Clark, 2009b). It is shown that the evolved programs detect both flooding and route disruption attacks successfully on simulated networks with varying traffic and mobility patterns. Furthermore, the grammars presented for each attack type are in a readable format so that security experts could easily understand the attack signatures. The authors extend their work by taking into account other objectives beside the intrusion detection capability of evolved programs. In (Sen and Clark, 2011), they propose a power-aware intrusion detection system in order to obtain a set of optimal solutions offering different trade-offs between detection ability and power usage by employing the SPEA2 algorithm (Zitzler et al., 2001). The programs are also compared with hand-coded detection programs which produce almost perfect detection rate with non-negligible false positive rates for networks under high mobility (Sen and Clark, 2011). On the other hand, evolutionary computation techniques not only produce acceptable false positive rates but also consider non-functional properties of intrusion detection programs. The authors also explore a suitable intrusion detection architecture for MANETs. Two different architectures are considered: stand-alone, distributed and cooperative architecture in the neighborhood. Other architecture proposals (Hassanzadeh and Stoleru, 2011b; Hassanzadeh and Stoleru, 2013) employing multi-objective evolutionary computation techniques also exist in the literature. The details of the architectures are presented in detail in the *IDS Architecture* section.

Evolutionary computation is also an emerging research area for wireless sensor networks. The first study on this area (Johnson et al., 2005) investigates a suitable framework for performing genetic programming on a wireless sensor nodes. EC could be a good candidate for many NP-hard problems in the field. Some problems are given as follows (Nan and Li, 2008): node placement and layout optimization, energy efficient routing, clustering, scheduling, position estimation, analyzing the lifetime of sensor networks and information fusion. In the literature, few approaches exist for intrusion detection (Khanna et al., 2007; Khanna et al., 2009). These approaches, as covered in the *IDS Architecture* section, mainly focus on optimally placing monitoring nodes.

A successful application of GP against a type of DoS attack (the deauthentication attack) at the link layer on WiFi networks is given in (Makanju et al., 2007). It is shown that GP is able to detect both the original attack and its modified versions with almost 100% detection rate and with less than 1% false positive rate.

4.4. IDS Architecture

There are few studies exploring the suitability of evolutionary computation techniques on how to place IDSs on a network. On the contrary, multi-objective evolutionary computation is a favorable candidate for taking into account different objectives while deploying IDSs.

Chen *et al.* (Chen et al., 2010) discovers the applicability of GA with multi-objective optimization techniques in order to place IDS sensors by satisfying conflict objectives. Various attack detection rates are traded-off with false alarm rates and costs. Probing and information gathering attack scenarios are evaluated based on the assumption that a placement for a type of attack could not be optimal for another type of attack. The following four objectives are tried to be satisfied: the number of sensors, detection rate, false alarm rate and monitoring costs calculated based on the monitored traffic. The experimental results show that different trade-offs are possible for IDS sensor placement. While the nodes have equal characteristics such as detection capabilities, the level of risk against them in the experiments, exploring the placement of IDS sensors with different characteristics could be investigated in the future.

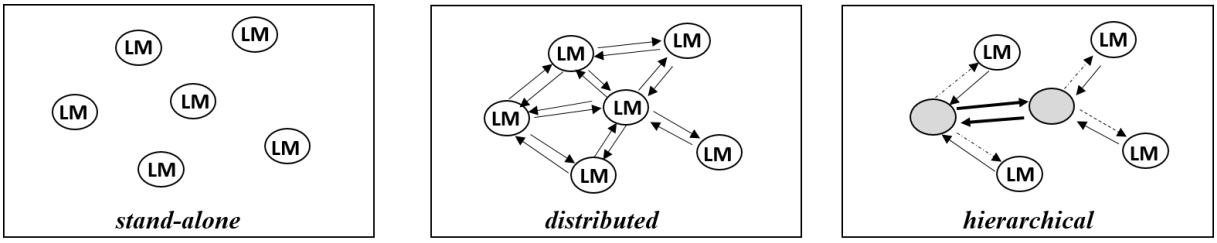

Figure 3. *IDS Architectures*

One of the most eligible areas to apply multi-objective optimization techniques for IDS placement is ad hoc networks in which intrusion detection systems must be distributed and cooperative due to their very nature. Different architectures are proposed in the literature: stand-alone, distributed and hierarchical as shown in Figure 3. While stand-alone architecture consists of only local monitoring (LM) nodes, nodes communicate with each other in order to obtain data and to distribute alarms in a distributed architecture. To achieve global detection of attacks, distributed and hierarchical architectures must attract the attention of researchers. In hierarchical architecture, while some nodes are only responsible for local detection, some particular nodes are more responsible than others by carrying out data aggregation and making decisions. The choice of these particular nodes is generally performed as random in order to provide security. Other criteria are residual energy, degree of connectivity, and the like. The first approach exploring an intrusion detection architecture suited to the distributed and resource-constrained environments is presented in (Sen and Clark, 2011). A distributed and collaborative architecture in a neighborhood where nodes communicate with one-hop away nodes in order to carry out intrusion detection is analyzed and compared with stand-alone architecture. While the classification accuracy is aimed to be maximized, the number of nodes in cooperation is minimized in order to save both energy

and bandwidth. While the classification accuracy is denoted with detection and false positive rate, the usage of bandwidth is denoted by the number of nodes in cooperation. The energy consumption is modeled mathematically (Feeney, 2001). The potential use of evolutionary computation techniques to discover complex properties of MANETs (such as limited power and bandwidth) are shown in the results.

Another important contribution of IDS deployment on MANETs by using GA is proposed in (Hassanzadeh et al, 2011b). It is proven that optimal monitoring node selection problem is NP-hard (Hassanzadeh et al., 2011a). Therefore MOO techniques are proposed to assign intrusion detection nodes in a cluster tree organization, based on some criteria (energy consumption, event reporting delay, network coverage, and quality of data). Distinct roles are defined for intrusion detection and each node is designed to perform only one role. It is shown that MOO techniques are necessary to optimize objectives simultaneously without negatively effecting others. Since the execution time of GA could be high, especially for large networks, an extra phase is introduced in their extended study (Hassanzadeh et al., 2013). In the two-phase hybrid algorithm for MOO, a set of suitable nodes as leaders (the nodes with the highest responsibilities) are determined in the first phase, and roles for *n* hop away neighbors of each leader is assigned by using GA in the second phase. It is empirically shown that hybrid algorithm outperforms only GA solution, in terms of both execution time and the optimality of the solutions evolved in large networks. While the experiments are carried out with nodes performing misuse-based intrusion detection, it is applicable for any type of detection technique. Khanna et al. (2009) also takes into account the security of monitoring nodes beside other objectives, node coverage and residual battery power. A weighted fitness function is employ to assess the individuals generated by the GA algorithm. The detection of compromised nodes is fasten up with this approach.

4.5. IDS Security

IDS itself could be the target of attacks. Corona *et al.* (Corona et al., 2013) divides attack goals against IDSs into six categories: evasion, overstimulation, poisoning, DoS, reverse hijacking and reverse engineering. The authors suggest to focus on techniques based on adversarial machine learning, since there is an expected increase in the machine learning techniques proposed for intrusion detection. The applications of evolutionary computation mainly focus on evasion attacks (Kayacik et al., 2011) as covered in the Foundations section. Pastrana *et al.* employs GP in order to reverse engineer the IDS behavior (Pastrana et al., 2011). It is assumed that the attacker does not know the internal behavior of the targeted IDS. One of the main reasons in employing GP in this study, is to generate easy-to-understand models. In order to achieve this objective, the depth of GP trees is restricted to five. The evolved IDS models (*i.e.* attack signatures) are used to generate evasion attacks against a C4.5-based intrusion detection system. The authors make legitimate modifications to evolved signatures in order to

avoid being detected while actualizing the attack. The modifications are not made automatically, but it is shown that the GP evolves successful IDS models similar to the IDS under study.

Recently, the adversarial capabilities against intrusion detection networks (IDNs) are presented in (Pastrana, 2014). IDNs consists of various monitoring and detection nodes, distributed through the network in order to be able to detect distributed attacks by aggreating/correlating data obtained from various sources. The study is one of the most comprehensive analysis on lDNs to evaluate the risks against these networks. Some countermeasures are considered based on this analysis. The trade-offs between cost and risk for each solution is represented in a Pareto front by employing the SPEA2 algorithm (Zitzler et al., 2001). So, the security experts could decide upon the placement of countermeasures.

4.6. Testing & Evaluation

Testing mainly covers issues related to setting benchmarks and creating datasets for evaluating IDSs objectively; however, it is out of the scope of this survey. In this section, we will outline the studies which compare different types of IDSs, particularly EC proposals. One of these studies is proposed by Abraham *et al.* (Abraham and Grosan, 2007), in which they compare GP performance on intrusion detection with decision tree and support vector machines approaches on the KDD dataset (Lippman et al., 2000), which is the most widely used benchmark evaluation data. They said that GP outperforms both techniques. It is noted that SVM performs well on detection for all kinds of attacks in the dataset. Mukkamala *et al.* (Mukkamala et al., 2004) compared linear GP with SVM and resilient back propagation neural networks (NN) on the KDD dataset. LGP presents better detection accuracy than the other two techniques, however it is a costly approach in terms of training time. Similar to the other comparisons (Abraham and Grosan, 2006; Pastrana et al., 2012), SVM is the best approach that is closest to GP. The authors also underline the possible effects of the parameters such as the program size, and the mutation and crossover rates of the results.

A comprehensive study on evaluating different classification algorithms for intrusion detection in mobile ad hoc networks is given in (Pastrana et al., 2012). The four popular attacks on ad hoc networks, dropping, blackhole, flooding and forging, are used in the experiments. The following six classifiers are compared: MultiLayer Perceptron (MLP), a linear classifier, the Gaussian Mixture Model (GMM), the naive Bayes classifier, Support Vector Machine (SVM) model and Genetic Programming (GP). The networks using AODV routing protocol (Perkins and Royer, 1999) is simulated, however different mobility levels are not taken into account. In the results, two classifiers, namely SVM and GP, surpass other techniques. Based on the results, it is suggested to use GP in order to differentiate attacks from normal behaviors (two-class classification). In the multi-class case, SVM performs better than GP. Sen and Clark (2011) also compared their EC-based intrusion detection proposals for MANETs with manually crafted programs. A performance comparison of the techniques GP and GE on the same attacks

can also be seen in the study. Makanju *et al.* (Makanju et al., 2007) compare the performance of GP on the detection of a DoS attack at the link layer on WiFi networks with Snort-Wireless, a widely used misuse-based IDS. The results show that while the variants of the attacks cannot be detected by Snort, GP shows a high detection rate with negligible false positive rate.

5. Conclusion and Future Directions

In this study, we present a survey on evolutionary computation techniques applied to intrusion detection. Each research area on intrusion detection is covered in detail. In this section, the highlights of the study are given.

Most studies in the literature are carried out for the development of effective and efficient intrusion detection methods. It is the same from the evolutionary computation-based approaches point of view. These approaches attract significant interest from researchers by generating readable outputs. It is a critical characteristic because security experts mainly tend to use the systems they can understand. While grammatical evolution techniques produce outputs in the specified grammar (Wilson and Kaur, 2007; Sen and Clark, 2009b), the outputs of genetic programming could become more readable by limiting the depth of GP trees evolved (Orfila et al., 2009). The comparison studies show that evolutionary computation techniques mainly perform equal to or greater than other machine learning techniques in this domain. The outputs of EC techniques are comparable with SVM, the technique showing the best performance in some studies.

Using fewer features is shown as another benefit of EC. We could also employ EC technique in order to select the feature set which best represents the problem, or is the most cost effective. Furthermore, multi-objective evolutionary computation techniques are among the best candidates for resource-constrained or dynamic environments in which different objectives such as less power usage, less communication between IDS agents need to be satisfied. Some studies also show that the outputs of EC techniques are lightweight.

There are few studies on IDS architecture. How to deploy intrusion detection nodes effectively and efficiently is an area worth investigating. Moreover, the proposed approaches should not introduce new vulnerabilities to the system, therefore proposed architectures should also be evaluated in terms of security. Furthermore, the optimal solution of the problem could change over time. For example, the location of nodes should adapt to the system due to changes in mobility, traffic, and resources in some environments. The applicability of evolutionary computation techniques could be investigated for such cases.

Testing is another area that needs to be explored. Even though studies which evolve new attacks or unknown variants of attacks exist in the literature, how to use a derived attack in order to develop our solutions has not been considered. It is believed that better solutions could be developed when evolved

new attacks are taken into account. That's why a co-evolutionary arms race mechanism could be employed to concurrently develop both intrusions and intrusion detection systems. Even though, co-evolutionary computation has a potential application area in security, there are few applications in this area (Ostaszewski et al., 2007).

Research on IDS security has accelerated in recent years. Attackers search for new ways to attack security solutions and to escape from such solutions. Especially, the security of machine learning-based IDSs should be investigated (Corona et al., 2013). As far as we know, there is no study exploring how EC based solutions are resilient to attacks such as evasion and DoS attacks.

To sum up, IDS is a research area extensively explored. However, some subfields of intrusion detection such as IDS testing and IDS security still need to be the subject of further studies. With the introduction of new attack strategies, new types of network, and the like, it is a field that has been evolving continuously. As shown in this study, evolutionary computation techniques are suited to developing effective, efficient and adaptive IDS. The suitability of evolutionary dynamic optimization and co-evolutionary computation techniques are promising areas that could be explored in future studies.

Acknowledgements

This study is supported by the Scientific and Technological Research Council of Turkey (TUBITAK -Project No: 112E354). We would like to thank TUBITAK for its support.

REFERENCES

1. Abadeh, M.S., Habibi, J., Barzegar, Z., Sergi, M. (2007a). A parallel genetic local search algorithm for intrusion detection in computer networks. Engineering Applications of Artificial Intelligence, Vol. 20, pp. 1058-1069.
2. Abadeh, M.S, Habibi, J., Lucas, C. (2007b). Intrusion detection using a fuzzy genetics-based learning algorithm. Journal of Network and Computer Applications, Vol. 30, pp. 414-428.
3. Abraham, A., Grosan, C. (2006). Evolving intrusion detection systems. Genetic Systems Programming: Theory and Experiences, Vol. 13, pp. 57-79, Springer.
4. Abraham, A., Grosan, C., Martiv-Vide, C. (2007). Evolutionary design of intrusion detection programs. International Journal of Network Security, Vol. 4, pp. 328-339.
5. Ahmad, I., Hussain, M., Alhgamdi, A., Alelaiwi, A. (2014). Enhancing SVM performance in intrusion detection using optimal feature subset selection based on genetic principal components. Neural Computing and Applications, Vol. 24, Issue 7-8, pp. 1671-1682.
6. Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Department of Computer Engineering, Chalmers University of Technology.
7. Bridges, S.M., Vaughn, R.B (2000). Fuzzy data mining and genetic algorithms applied to intrusion detection. *In* Proc. of the 23rd National Information Systems Security Conference, pp. 13-31.
8. Chen, H., Clark, J.A., Shaikh, S.A., Chivers, H., Nobles, P. (2010). Optimising IDS Sensor Placement, *In* Proc. of International Conference on Availability, Reliability, and Security.
9. Coello, C.A.C, Lamont, G.B., Linkveldhuizen, D.A.V. (2007). Evolutionary algorithms for solving multi-objective problems, Springer.
10. Corona, I., Giacinto, G., Roli, F. (2013). Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. Information Sciences, Vol. 239, pp. 201-225.
11. Crosbie, M., Spafford, E.H. (1995). Applying genetic programming to intrusion detection. *In* Working Notes for the AAAI Symposium on GP, pp. 1-8.
12. CERT, Cybersecurity Watch Survey (2011). http://www.cert.org/insider threat/
13. Dam, H. H., Shafi, K., Abbass, H.A. (2005). Can evolutionary computation handle large datasets? A study into network intrusion detection. *In* Proc. of Australian Conference on Artificial Intelligence, LNAI 3809, Springer, pp. 1092-1095.
14. Denning, D. (1987). An intrusion detection model, IEEE Transactions on Software Engineering, Vol. 13(2), pp. 222-232.
15. Diaz-Gomez, P.A., Hougen, D., (2005a). Improved off-line intrusion detection using a genetic algorithms. *In* Proc. of the Seventh International Conference on Enterprise Information Systems, pp. 66-73.
16. Diaz-Gomez, P.A., Hougen, D. (2005b). Analysis and mathematical justification of a fitness function used in an intrusion detection system. *In* Proc. of the Genetic and Evolutionary Computation, pp. 1591-1592.

17. Feeney, L.M. (2001). An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. Mobile Networks and Applications, Vol. 6, pp. 239-249.
18. Fogel, D.B. (2000). What is evolutionary computation? IEEE Spectrum, Vol. 37, pp. 28–32.
19. Gomez, J., Gil, C., Banos, R., Marquez, A.L., Montoya, F.G., Montoya, M.G. (2013). A Pareto-based multi-objective evolutionary algorithms for automatic rule generation in network intrusion detection systems, Soft Computing, Vol. 17, pp. 255-263.
20. Gu, G., Fogla, P., Dagon, D., Lee, W., Skoric, B. (2006). Measuring intrusion detection capability: an information-theoretic approach. *In* Proc. of the 2006 ACM Symposium on Information, Computer and Communications Security, ACM, pp. 90-101.
21. Hassanzadeh, A., Stoleru, R., Shihada, B. (2011a). Energy efficient monitoring for intrusion detection in battery-powered wireless mesh networks. *In* Proc. of ADHOC-NOW, LNCS 6811, Springer, pp. 44-57.
22. Hassanzadeh, A., Stoleru, R. (2011b). Towards optimal monitoring in cooperative ids for resource constrained wireless networks. *In* Proc. of the 20th International Conference on Computer Communications and Networks.
23. Hassanzadeh, A., Stoleru, R. (2013). On the optimality of cooperative intrusion detection for resource constrained wireless networks, Computers & Security, Vol. 34, pp. 16-35.
24. Helmer, G., Wong, J., Honavar, V., Miller, L. (1999). Feature selection using a genetic algorithm for intrusion detection. *In* Proc. of the Genetic and Evolutionary Computation Conference, Vol. 2.
25. Helmer, G., Wong, S.K., Honavar, V., Miller, L. (2002). Automated discovery of concise predictive rules for intrusion detection. The Journal of Systems and Software, Vol. 6, pp. 165-175.
26. Hofmann, A., Horeis, T., Sick, B. (2004). Feature selection for intrusion detection: an evolutionary wrapper approach. In Proc. of IEEE International Joint Conference on Neural Networks, pp 1563-1568.
27. Johnson, D.M., Teredesai, A.M., Saltarelli, R.T. (2005). Genetic programming in wireless sensor networks. *In* Proc. of the European Conference on Genetic Programming, LNCS 3447, pp. 96-107.
28. Kayacık, H.G., Heywood, M.I., Zincir-Heywood, A.N. (2005a). Evolving successful stack overflow attacks for vulnerability testing. *In* Proc. of the IEEE 21st Annual Computer Security Applications Conference (ACSAC).
29. Kayacık, H.G., Heywood, M.I., Zincir-Heywood, A.N. (2005b). On evolving buffer overflow attacks using genetic programming. *In* Proc. of the 11th Genetic and Evolutionary Computation Conference (GECCO-2006).
30. Kayacik, H. G., Zincir-Heywood, A. N., Heywood, M. I., Burschka, S. (2009). Generating mimicry attacks using genetic programming: a benchmarking study. *In* Proc. of the 2009 IEEE Symposium on Computational Intelligence in Cyber Security (CICS-2009).

31. Kayacik, H. G., Zincir-Heywood, A. N., Heywood, M. I. (2011). Evolutionary computation as an artificial attacker: generating evasion attacks for detector vulnerability testing. Evolutionary Intelligence, Vol. 4, issue 4, pp. 243-266.
32. Khanna, R., Liu, H., Chen H. (2007). Dynamic optimization of secure mobile sensor networks: a genetic algorithm, *In* Proc. of IEEE International Conference on Communications.
33. Khanna, R., Liu, H., Chen H. (2009). Reduced complexity intrusion detection in sensor networks using genetic algorithm, *In* Proc. of IEEE International Conference on Communications.
34. Kim, D.S., Nguyen, H., Park, J.S. (2005). Genetic algorithm to improve SVM based network intrusion detection system. *In* Proc. of the 19th International Conference on Advandced Information Networking and Applications.
35. Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection, MIT Press, Cambridge, MA.
36. Lange, D. B., Oshima, M. (1999). Seven good reasons for mobile agents. Communications of the ACM, Vol. 42, No. 3, pp. 88-89.
37. Lippman, R. P., Haines, J.W., Fried D.J., Korba, J., Das, K. (2000). Analysis and results of the 1999 darpa off-line intrusion detection evaluation. *In* Proc. of the International Symposium on Recent Advances in Intrusion Detection, LNCS 2212, Springer, pp. 162-182.
38. Lu, W., Traore, I. (2004). Detecting new forms of network intrusion using genetic programming. Computational Intelligence, Vol. 20, no. 3.
39. Lundin, E., Jonsson, E. (2002). Survey of intrusion detection research. Technical Report 02-04, Department of Computer Engineering, Chalmers University of Technology.
40. Mabu, S., Chen, C., Lu, N., Shimada, K., Hirasawa, K. (2011). An intrusion detection model based on fuzzy class-association-rule mining using genetic network programming. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 41, No. 1, pp. 130-139.
41. Makanju, A., LaRoche, P., Zincir-Heywood A.N. (2007). A comparison between signature and GP-based IDSs for link layer attacks on WiFi networks. *In* Proc. of the 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications.
42. Me, L. (1998). GASSATA: A genetic algorithm as an alternative tool for security audit trails analysis. *In* Proc. of the International Symposium on Recent Advances in Intrusion Detection.
43. Middlemiss, M.J., Dick, G. (2003). Weighted feature extraction using a genetic algorithm for intrusion detection. In Proc. of the 2003 Congress on Evolutionary Computation, pp. 1669-1675.
44. Mukkamala, S., Sung, A.H., Abraham, A. (2004). Modeling intrusion detection systems using linear genetic programming approach. *In* Proc. of the 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, LNCS 3029, Springer, pp. 633-642.
45. Nan, G., Li, M. (2008). Evolutionary based approaches in wireless sensor networks: a survey. In Proc. of the 4th International Conference on Natural Computation.

46. Nguyen, H., Franke, K., Slobodan, Petrovic (2010). Improving effectiveness of intrusion detection by correlation feature selection. In Proc. of the International Conference on Availability, Reliability, and Security, pp. 17-24.
47. Noreen, S., Murtaza, S., Shafiq, M.Z., Farooq, M. (2009). Evolvable malware. *In* Proc. of the 14th Genetic and Evolutionary Computation Conference (GECCO-2009).
48. O'Neill, M., Ryan, C. (2003). Grammatical evolution: evolutionary automation programming in an arbitrary language, Kluwer Academic Publishers.
49. Orfila, A, Estevez-Tapiador, J.M., and Ribagorda, A. (2009). Evolving high-speed, easy-to-understand network intrusion detection rules with genetic programming, *In* Proc. of EvoWorkshops on Applications of Evolutionary Computations, LNCS Series, Vol. 5484, pp. 93-98, Springer.
50. Ostaszewski, M., Seredynski, F., Bouvry, P. (2007). Coevolutionary-based mechanisms for network anomaly detection. Journal of Mathematical Modelling and Algorithms, Vol. 6, Issue 4, pp. 411-431.
51. Pastrana, S., Mitrokotsa, A., Orfila, A., Peris-Lopez, P. (2012). Evaluation of classification algorithms for intrusion detection in MANETs. Knowledge-Based Systems, Vol. 36, pp. 217-225.
52. Pastrana, S., Orfila, A., Ribagorda, A. (2011). Modeling NIDS evasion with genetic programming. *In* Proc. of the 9th International Conference on Security and Management.
53. Pastrana, S. (2014). Attacks against intrusion detection networks: evasion, reverse engineering and optimal countermeasures. PhD Thesis, Computer Science and Engineering Department, Universidad Carlos III De Madrid,
54. Perkins, C.E., Royer, E.M. (1999). Ad-hoc on-demand distance vector routing, In Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, IEEE, pp. 90-100.
55. Roesch, M., Snort – lightweight intrusion detection for networks (1999). *In* Proc. of the 13th USENIX Conference on System Administration Conference, pp. 229-238.
56. Ryan, C., Colline, J.J., O'Neill, M. (1998). Grammatical evolution: evolving programs for an arbitrary language. *In* Proc. of the First European Workshop on Genetic Programming, LNCS 1391, Springer, pp. 83-95.
57. Sen, S., Clark, J.A. (2009a). Intrusion detection in mobile ad hoc networks. Guide to wireless ad hoc networks, Springer, Chapter 17, pp. 427-454.
58. Sen, S., Clark, J.A. (2009b). A grammatical evolution approach to intrusion detection on mobile ad hoc networks, *In* Proc. of the Second ACM Conference on Wireless Network Security, pp. 95-102.
59. Sen, S, Clark, J.A. (2011). Evolutionary computation techniques for intrusion detection in mobile ad hoc networks. Computer Networks, Vol. 55, Issue 15, pp. 3441-3457.
60. Sen, S., Dogmus, Z. (2012). Feature selection for detection of ad hoc flooding attacks. Advances in Computing and Information Technology, Vol. 176, Springer, pp. 507-513.
61. Sen, S. (2014). Using instance weighted naive Bayes for adapting concept drift in masquerade detection. International Journal of Information Security.

62. Shazzad, K.M., Park, J.S. (2005). Optimization of intrusion detection through fast hybrid feature selection. In Proc. of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies.
63. Sinclair, C., Pierce, L., Matzner, S. (1999). An application of machine learning to network intrusion detection, *In* Proc. of the 15th Annual Computer Security Applications Conference, pp. 371-377.
64. Snort, https://www.snort.org/ (*accessed on* July, 2014)
65. Song, D., Heywood, M.I., Zincir-Heywood A.N. (2005). Training genetic programming on half a million patterns: An example from anomaly detection, IEEE Transactions on Evolutionary Computation, Vol. 9(3), pp. 225-239.
66. Stein, G., Chen, B., Wu, A.S., Hua, K.A. (2005). Decision tree classifier for network intrusion detection with GA-based feature selection. *In* Proc. of the 43rd ACM Annual Southeast Regional Conference, Vol. 2, pp. 136-141.
67. Tahta, U.E., Can, A.B., Sen, S. (2014). Evolving a trust model for peer-to-peer networks using genetic programming, *In* Proc. of EvoStar, LNCS Series, Springer. *(to appear)*
68. Uppuluri, P, Sekar, R. (2001). Experiences with specification-based intrusion detection. *In* Proc. of the Recent Advances in Intrusion Detection, LNCS 2212, Springer, pp. 172-189.
69. Wilson, D., Kaur, D. (2007). Knowledge extraction from KDD'99 intrusion data using grammatical evolution, WSEAS Transactions on Information Science and Applications, Vol. 4, pp. 237-244.
70. Wu, S. X., Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: a review, Applied Soft Computing, Vol. 10(1), pp. 1-35.
71. Zhao, J., Zhao, J., Li, J. (2005). Intrusion detection based on clustering genetic algorithm. *In* Proc. of the Fourth International Conference on Machine Learning and Cybernetics, IEEE, pp. 3911-3914.
72. Zitzler, E., Laumans, M., Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Swiss Federal Institute of Technology.