Applying Genetic Programming to Evolve Learned Rules for Network Anomaly Detection

Chuanhuan Yin, Shengfeng Tian, Houkuan Huang, and Jun He

School of Computer and Information Technology,
Beijing Jiaotong University, Beijing, 100044, China
{chhyin, sftian, hkhuang}@center.njtu.edu.cn,
j.he@cs.bham.ac.uk

Abstract. The DARPA/MIT Lincoln Laboratory off-line intrusion detection evaluation data set is the most widely used public benchmark for testing intrusion detection systems. But the presence of simulation artifacts attributes would cause many attacks in this dataset to be easily detected. In order to eliminate their influence on intrusion detection, we simply omit these attributes in the processes of both training and testing. We also present a GP-based rule learning approach for detecting attacks on network. GP is used to evolve new rules from the initial learned rules through genetic operations. Our results show that GP-based rule learning approach outperforms the original rule learning algorithm, detecting 84 of 148 attacks at 100 false alarms despite the absence of several simulation artifacts attributes.

1 Introduction

Intrusion detection is an important facet of computer security. It has been extensively investigated since the report written by Anderson [1]. An intrusion detection system (IDS) can detect hostile attacks by monitoring network traffic, computer system sources, audit records, or the access of the file system. There are two models in intrusion detection context, one of which is misuse detection while the other is anomaly detection. The former technique aims to develop models of known attacks, which can be detected through these models. The latter intends to model normal behaviors of systems or users, and any deviation from the normal behaviors is regarded as an intrusion. Due to their nature, misuse detection has low false alarms but its major limitation is that it can't detect novel or unknown attacks until the signatures of attacks are appended to the intrusion detection system, on the contrary, anomaly detection has the advantage of detecting novel or unknown attacks which can't be detected by misuse detection but it has the potential to generate too many false alarms.

With the widespread use of the Internet, intrusion detection systems have become focused on attacks to the network itself. Network intrusion detection system has been developed to detect these network attacks, which can't be detected by host intrusion detection system, by means of examining network traffic. There are a number of network intrusion detection systems using misuse detection or anomaly detection.

Because the models of normal traffic are hard to be obtained, network intrusion detection systems like SNORT [2] typically use misuse detection, matching characteristics of network traffic to the characteristics of known attacks in their database. This method detects known attacks in a very low false alarms rate while omits novel or unknown attacks against intrusion detection systems' database. An alternative approach is anomaly detection, which models normal traffic and regards any deviation from this model as suspicious. But due to its high false alarms rate, anomaly detection hasn't been applied in most commercial intrusion detection systems, leading to the research on network anomaly detection.

This paper focuses on rule learning for network anomaly detection system. We evolve rules learned from the training traffic by using Genetic Programming (GP) [3], and then we use the evolved rules to differentiate attacks traffic from normal traffic.

We present an algorithm called LERAD-GP (LEarning Rules for Anomaly Detection based Genetic Programming) to generate and evolve rules for detecting attacks. LERAD-GP is a variation of LERAD, which has been presented by Mahoney et al. to find the relation of the attributes in network connections [4]. In the process of experimenting, we found that LERAD-GP outperforms LERAD in detecting attacks from modified data set, in which the simulation artifacts had been removed. The presence of simulation artifacts would lead to overoptimistic evaluation of network anomaly detection systems [5]. At the same time, we introduce a simple method to remove simulated artifacts, and validate the effectivity of this remove approach.

The rest of the paper is organized as follows. Section 2 describes an overview of related work in network anomaly detection and Genetic Programming. Section 3 describes the GP algorithm. Section 4 discusses how to use GP to evolve learned rules. Section 5 presents the evaluation of the evolved rules using test dataset and discusses the experimental results. Section 6 concludes the paper.

2 Related Work

Network anomaly detectors look for unusual network traffic by taking some attributes of traffic into account. Some information in network packets such as IP addresses and port numbers can be used for modeling network normal traffic. ADAM (Audit Data and Mining) monitors port numbers, IP addresses and subnets, and TCP state to build normal traffic models which can be used to detect suspicious connections [6]. Like ADAM, SPADE (Statistical Packet Anomaly Detection Engine) monitors addresses and ports to achieve detection [7]. Nevertheless, these few attributes of traffic are far from enough to model network traffic. There are more network attributes should be used to distinguish between hostile and benign traffic.

To use more features of traffic, Mahoney presented several methods in his paper [8]. They are PHAD, ALAD, LERAD, and NETAD [4, 9, and 10]. PHAD (Packet Header Anomaly Detector) [9] is a system that learns the normal range of values for 33 fields of the Ethernet, IP, TCP, UDP, and ICMP protocols. It is implemented by using simple nonstationary models that estimate probabilities based on the time since the last event rather than the average rate of events which is often applied to estimate probabilities. Different from PHAD, ALAD (Application Layer Anomaly Detection) [9] assigns a score to an incoming server TCP connection. It is configured to detect network attacks against the fixed victim, and it distinguishes server ports (0-1023) from client ports (1024-65535). After a testing of a number of attributes and their combinations, Mahoney et al. select five because of their best performance. They are P(source IP address | destination IP address), P(source IP address | destination IP address, destination TCP port), P(destination IP address, destination TCP port), P(TCP flags | destination TCP port), and P(keyword | destination TCP port). During the training stage, these probabilities are estimated and then used to obtain the anomaly score of connections in detecting. As with PHAD, the anomaly score is relevant to the time since the last event. LERAD (LEarning Rules for Anomaly Detection) [4] is a system that learns rules for finding rare events in nominal time-series data with long range dependencies. It constitutes an improvement of the two previous methods by using a rule learning algorithm. LERAD is able to learn important relationships between attributes of benign traffic, and use them to detect hostile traffic. NETAD (NEtwork Traffic Anomaly Detector) [10] models the most common protocols (IP, TCP, Telnet, FTP, SMTP, and HTTP) at the packet level to flag events that have not been observed for a long time. It is based on the consideration of fist 48 bytes of the packet, each of which is treated as an attribute with 256 possible values. The last algorithm can detect anomaly values ever seen in training phase while the others regard values seen in training phase as normal.

One of the major contributions of Mahoney et al. is that they presented a time-based model appropriate for bursty traffic with long range dependencies [8]. There had long been assumed that network traffic could be modeled by a Poisson process, in which events are independent of each other. Therefore, some anomaly detectors like ADAM and SPADE regard the average rate of events x in training as the probability of x. However, this may be inappropriate in this context. Paxson et al. showed that many network processes are self-similar or fractal [11]. In order to depict the features of network traffic, Mahoney et al. proposed a time-based model. Furthermore, Mahoney et al. presented a continuous model to monitor previously seen values of attributes. Using these two models, Mahoney et al. showed that LERAD and NETAD performed quite well on the dataset of DARPA 1999 [12]. They used the inside sniffer traffic from week 3 of the DARPA 1999 valuation dataset for training dataset, and regarded week 4 and 5 as testing dataset. Table 1 shows their results at 100 false alarms.

Table 1. The numbers of detected attacks of LERAD and NETAD at 100 false alarms

Approach	**Detections**
LERAD	117/177
NETAD	132/177

Mahoney et al. found the presence of simulation artifacts by integrating real network traffic into DARPA 1999 dataset. They deem that the simulation artifacts would lead to overoptimistic evaluation of network anomaly detection systems. In order to eliminate the impact of simulation artifacts of original network traffic dataset, Mahoney et al. collected a real inside traffic dataset which they denoted as FIT dataset. After preprocessing FIT, they mixed the DARPA 99 dataset with FIT dataset.

After that, they modified their detection algorithms to detect attacks of mixed dataset. Table 2 shows their detection results of Probe, DoS (denial of service), and R2L (remote to local) attacks on mixed dataset.

Table 2. Probe, DoS, and R2L attacks detected at 100 false alarms on mixed dataset

Approach	**Detections**
PHAD	24/148
ALAD	13/148
LERAD	30/148
NETAD	42/148

From their depiction, the performance of their methods on mixed dataset is far worse than that on DARPA 1999 dataset. But the appropriate explanation hadn't been presented. After analyzing their works on detection, we deem that there are several reasons for the worse performance of these four methods on merged dataset.

1. FIT was not free of attacks. Mahoney et al. examined the traffic manually and tested it using SNORT [3], and they found that at least four attacks existed in the FIT dataset. Moreover, it is sure that there are more attacks not found by them. Therefore, their training was based on a dataset contained a few unknown or known attacks. The noisy attacks would influence the training phase, resulting in a deviation from normal model which would be obtained from attack free training dataset. The deviated model led to a decrease in the attacks detection rates at the same false alarms rate.
2. The number of packets in mixed dataset was about double of that in DARPA 1999 dataset. Both traffics will generate false alarms, resulting in a higher error rate than either traffic source by itself. However, the evaluation criterion is the same as that of original traffic, resulting in the lower detection rates.
3. Mahoney et al. claimed that NETAD can model previously seen values, and that this detector can be used for training dataset which contains some attacks. However, from the above results we know that the model of NETAD isn't appropriate for this environment. All of the four detectors performed poorly if the training dataset contains some attacks.

GP has been used to solve many problems since it was developed by Koza [3]. It is also widely used in intrusion detection. Crosbie et al. employed a combination of GP and agent technique to detect anomalous behaviors in a system [13]. They use GP to evolve autonomous agents for detecting potentially intrusive behaviors. But communication among these autonomous agents is still an issue. Su et al. used GP to generate the normal activity profile by modeling system call sequences [14]. If the tested sequences deviate from the normal profile, their system denotes the process as intrusion. Lu et al. proposed a rule evolution approach based on GP to evolve initial rules which were selected based on background knowledge from known attacks [15]. After that, they used initial and evolved rules to detect known or novel attacks on a network. The results showed that the rules evolved based on knowledge of known attacks could detect some novel attacks. But the initial rules they used must be manually specified by domain experts.

3 Overview of GP

3.1 GP

GP is a further extension of Genetic Algorithm (GA) [16, 17]. In contrast to GA, GP typically operates on a population of parse trees which usually represent computer programs. A parse tree is composed of internal nodes and leaf nodes. The internal nodes are called primitive functions while the leaf nodes are called terminals. The terminals can be regarded as the inputs to the program being induced. There are independent variables and the set of constants in the terminals. The primitive functions can form more complex function calls by combining the terminals or simpler function calls. Solving a problem is a search through all the possible combinations of symbolic expressions defined by the programmer.

3.2 GP Operators

GP can be used to evolve rules which are of the form "if antecedents then consequent". In rule evolving, three genetic operators which are called crossover, mutation, and dropping condition can be used [18].

Crossover is a sexual operation that produces two children from two parents. A part of one parent is selected and replaced by another part of the other parent. Mutation is an asexual operation. A part in the parental rule is selected and replaced by a randomly generated part.

Dropping condition can be used to evolve new rules. It selects randomly one condition, and then turns it into "any", resulting in a generalized rule. This operator is a new genetic operator, which is proposed to evolve new rules [18]. For example, the rule

if condition 1 and condition 2 then consequence.

can be changed to

if condition 1 and any then consequence.

3.3 Fitness Function

A fitness function is needed to evaluate evolved rules. In this context, we use the fitness function based on the support-confidence framework and the number of antecedents. Support measures the coverage of a rule while confidence factor (*cf*) represents the rule accuracy. If a rule has the format of "*if A then B*", its confidence factor and support are defined as follows:

$$cf = |\text{A and B}| \,/\, |\text{A}|;\ support = |\text{A and B}|/\text{N}, \tag{1}$$

where |A| is the number of records that only satisfy antecedent A, |B| is the number of records that satisfy consequent B, |A *and* B| is the number of records that satisfy both antecedent A and consequent B, and N is the size of training dataset.

As described in [18], some rules may have high accuracy but the rules may be formed by chance and based on a few training examples. This kind of rules does not have enough support. To avoid the waste of evolving those rules of low support, fitness function is defined as:

$$fitness = \begin{cases} support & \text{if } support < min\_support \\ w_1 \times support + w_2 \times cf & \text{otherwise} \end{cases}, \tag{2}$$

where *min_support* is a minimum threshold of *support*, the weights w_1 and w_2 are user-defined to control the balance between the confidence and the support in searching.

Finally, we use the modified fitness as follows:

$$modified\_fitness = fitness/r, \tag{3}$$

where r is the number of antecedents in the rule. The idea behind the division by r is that the more antecedents in a rule, the easier the rule will be removed in the validation stage of the algorithm described in next section.

4 Using GP to Generate New Rules

We use a rule learning method which is similar with LERAD to generate the initial rule set. LERAD learns conditional rules over nominal attributes. After evolving rules we obtain a rule set, which can be used to evaluate connections from test dataset and then to detect malicious connections.

The antecedent of a rule is a conjunction of equalities, and the consequent is a set of allowed values, e.g. *if port = 80* and *word3 = HTTP/1.0 then word1 =GET or POST*. Allowed values means that it is observed in at least one training instance satisfying the antecedent. During testing, if an instance satisfies the antecedent but the consequent isn't one of the allowed values, then an anomaly score of *ts/n* is calculated, where t is the time since the last anomaly by this rule, s is the number of training instances satisfying the antecedent, and n is the number of allowed values. In fact, the reciprocal of the anomaly function is the product of two probabilities, *1/t* and *n/s*. Therefore, this anomaly score can be used to identify rare events: those which have not occurred for a long time (large t) and where the average rate of "anomalies" in training is low (small *n/s*). For all violated rules, a total anomaly score is summed. If the summed score exceeds a threshold, an alarm is generated.

The LERAD based GP algorithm is proposed as follows:

1. Rule generation. Randomly select M pairs of training instances from a subset S which is also randomly sampled from the training dataset. Then use these M pairs of instances to generate initial rule set R, in which each rule satisfies both instances with s/n = 2/1. Each pair of instances generate up to L rules.

2. Rule evolving. Evolve rules from R using crossover and dropping condition operators. The mutation operator isn't used because the mutated rules may become illegal, e.g. *if port = 80 then word1 = GET*, may mutated to: *if port = HTTP/1.0 then word1 = GET*.

3. Testing coverage. Discard rules from R to find a minimal subset of rules, which cover all instance values in S, favoring rules with higher *modified-fitness* over S.

4. Training. Make sure each of the consequent values of each rule in R is observed at least once when the antecedent is satisfied. If the predefined termination criterion is satisfied, then turn to step 5, otherwise to step 2.

5. Validation. In the whole training dataset, if an instance satisfies the antecedent of a rule but the consequent isn't one of its allowed values, then remove this rule. The reason for rule removing is because the invalidation of this rule has been verified via the unsatisfied instance.

6. Testing. For each instance in testing, assign an anomaly score of $\sum ts/n$ for the violations. The higher score means more suspicious of attacks.

5 Experiments and Results

We test our method using the 1999 DARPA/Lincoln Laboratory intrusion detection evaluation data set, a widely used benchmark using synthetic network traffic [12]. Only the inbound client traffic is used for our experiments because the targets of most R2L attacks, as well as probes and DoS attacks are various servers, leading to the abnormality in the inbound traffic. Our method was trained on week 3 and extra week 3, which contains no attacks, and tested on weeks 4 and 5.

Firstly we reassemble TCP connections from traffic packets. LERAD used 23 attributes of reassembled TCP connection to generate rules and detect suspicious connections. But according to the analysis of Mahoney et al. [5], there are some suspected artifacts attributes in the DARPA 1999 dataset, including client source IP address, TTL, TCP window size, and TCP options. For the purpose of removing artifacts attributes, we omit the above four attributes from reassembled TCP connections, resulting in 16 attributes being remained. The remained 16 attributes are date, time, last two bytes of the destination IP address, source port, destination port, log base 2 of the duration time, log base 2 of the length, and first 8 words of the payload. Because simulation artifacts would help intrusion detectors to detect attacks, some attacks become hardly detected after removing of artifacts attributes. So the detection rate of anomaly detector will be inevitable decreased if we want to keep the same false alarms as before. Nevertheless, the influence of artifacts attributes can be eliminated using this simple removal approach.

We set the sample size $|S| = 200$ and draw $M = 1000$ sample pairs, generating up to $L = 4$ for each rule. As to GP, The rates of crossover and dropping condition operations are 0.6, 0.001 respectively for each rule.

In our experiments, attack is counted as detected if it is "in-spec" (the attack are supposed to be detected) and the detector correctly identifies the IP address of the victim or attacker and the time of any portion of the attack interval within 60 seconds. Out of spec detections are ignored. Duplicate detections of the same attack are counted only once, but every false alarm is counted. Our anomaly detector is designed to detect 148 in-spec attacks, including probe, DoS, and R2L because there is evidence for these attacks in the inside sniffer traffic according to the truth labels.

We also test NETAD and LERAD in the dataset in which four suspected artifacts attributes are omitted. Moreover, for the sake of comparison, we excerpt the results of [7], in which NETAD and LERAD tested on the mixed dataset. Just the same as Mahoney et al., We refer to them as NETAD-C and LERAD-C respectively. Therefore, we compare the performance of LERAD-GP with that of the other two algorithms, which are tested in both modified DARPA 1999 and the mixed dataset in [4]. The results are as follows.

Table 3. Probes, DoS, and R2L attacks detected by LERAD, NETAD, and LERAD-GP at 100 false alarms

Category	**Total**	**LERAD-C**	**NETAD-C**	**LERAD-GP**	**LERAD**	**NETAD**
Probe	34	7(21%)	12(35%)	23(68%)	21(62%)	8(23%)
DoS	60	5(8%)	11(18%)	34(57%)	26(43%)	15(25%)
R2L	54	18(33%)	18(33%)	27(50%)	19(35%)	11(20%)
Total	**148**	**30(17%)**	**41(28%)**	**84(57%)**	**66(45%)**	**34(23%)**

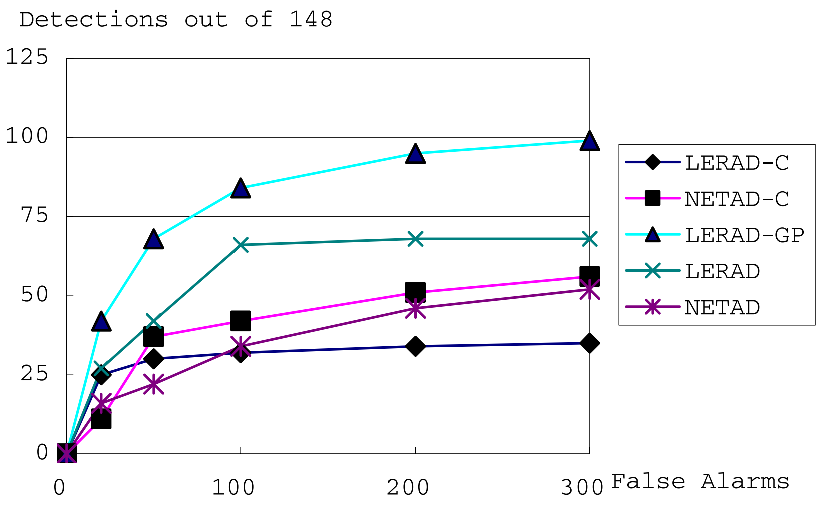

Fig. 1. Probe, DoS, and R2L attacks detected by LERAD, NETAD, and LERAD-GP as 0 to 300 false alarms (*0 to 30 per day*)

6 Conclusion

In this paper, we have presented and evaluated a GP-based rule learning approach for detecting attacks on network. We also introduced a simple removal approach to simulated artifacts attributes, which are regarded as the reason for overoptimistic evaluation of network anomaly detectors. The effectivity of this simple removal has been validated through the experiments conducted in this paper. We can see that LERAD-GP outperforms the original LERAD algorithm. The results show that GP can optimize rules by crossover and dropping condition operations for anomaly detecting.

However, there are some limitations in LERAD-GP. One is that the algorithm needs two passes during training, resulting in the inefficiency of detector. Another is that it requires training dataset which is attack-free whereas explicit training and testing data would not be available in a real setting. How to model network traffic on the training dataset which is mixed with attacks is still an open issue.

Acknowledgements

This paper is supported by the National Natural Science Foundation of China (No. 60442002, No. 60443003) and the Science and Technology Foundation of Beijing Jiaotong University (No. 2004SM010).

References

1. Anderson, J.P.: Computer Security Threat Monitoring and Surveillance. Technical Report, Fort Washington, PA (1980)
2. Roesch, M.: Snort: Lightweight intrusion detection for networks. In: Proc. of USENIX Large Installation System Administration Conference (1999)
3. Koza, J.R.: Genetic Programming. MIT Press (1992)
4. Mahoney, M.V., Chan, P.K.: Learning Rules for Anomaly Detection of Hostile Network Traffic. In: Proc. of International Conference on Data Mining (2003)
5. Mahoney, M.V., Chan, P.K.: An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In: Proc. of International Symposium on Recent Advances in Intrusion Detection (2003) 220-237
6. Barbara, D., Couto, J., Jajodia, S., Popyack, L., Wu, N.: ADAM: Detecting Intrusions by Data Mining. In: Proc. of IEEE Workshop on Information Assurance and Security (2001) 11-16
7. Hoagland, J.: SPADE, http://www.silicondefense.com/software/spice/ (2000)
8. Mahoney, M.V.: A Machine Learning Approach to Detecting Attacks by Identify-ing Anomalies in Network Traffic. Ph.D. dissertation, Florida Institute of Technology (2003)
9. Mahoney, M.V., Chan, P.K.: Learning Non-stationary Models of Normal Network Traffic for Detecting Novel Attacks. In: Proc. of ACM Special Interest Group on Knowledge Discovery in Data and Data Mining (2002) 376-385
10. Mahoney, M.V.: Network Traffic Anomaly Detection Based on Packet Bytes. In: Proc. of ACM Symposium on Applied Computing (2003)
11. Paxson, V., Floyd, S.: Wide area traffic: the failure of Poisson modeling. IEEE/ACM Transactions on Networking **3** (1995) 226 – 244
12. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA Off-Line Intrusion Detection Evaluation. Computer Networks **34** (2000) 579-595
13. Crosbie, M., Spafford, G.: Applying Genetic Programming to Intrusion Detection. In: Proc. of AAAI Fall Symposium on Genetic Programming (1995)
14. Su, P.R., Li, D.Q., Feng, D.G.: A Host-Based Anomaly Intrusion Detection Model Based on Genetic Programming. Chinese Journal of Software **14** (2003) 1120-1126
15. Lu, W., Traore, I.: Detecting New Forms of Network Intrusion Using Genetic Programming. Computational Intelligence **20** (2004)
16. Yao, X.: Evolutionary Computation: Theory and Applications. World Scientific, Singapore (1999)
17. Tan, K.C., Lim, M.H., Yao, X., Wang L.P. (Eds.): Recent Advances in Simulated Evolution and Learning. World Scientific, Singapore (2004)
18. Wong, M.L., Leung, K.S.: Data Mining Using Grammar based Genetic Program-ming and Applications. Kluwer Academic Publishers (2000)