Modeling NIDS evasion with Genetic Programming

S. Pastrana[1], **A. Orfila**[1], **and A. Ribagorda**[1]
[1]IT Security Group, Carlos III University of Madrid, Leganes, Madrid, Spain

Abstract— *Nowadays, Network Intrusion Detection Systems are quickly updated in order to prevent systems against new attacks. This situation has provoked that attackers focus their efforts on new sophisticated evasive techniques when trying to attack a system. Unfortunately, most of these techniques are based on network protocols ambiguities [1], so NIDS designers must take them into account when updating their tools. In this paper, we present a new approach to improve the task of looking for new evasive techniques. The core of our work is to model existing NIDS using the Genetic Programming paradigm. Thus, we obtain models that simulate the behavior of NIDS with great precision, but with a much simpler semantics than the one of the NIDS. Looking for this easier semantics allows us to easily construct evasions on the model, and therefore on the NIDS, as their behavior is quite similar. Our results show how precisely GP can model a NIDS behavior.*

Keywords: Intrusion Detection; Evasion; Network security; Genetic Programming

1. Introduction

Information Technologies have become a critical component of global economy in the last few years. Their protection against hostile actions determine how fast information society and communications will evolve. Security measures are normally classified as: preventive, detective , corrective and recovery. The most convenient are the first, but their cost is the highest and they do not assure to disable totally the risk, so it is preferable to distribute resources over all the techniques. Thus, a called *Perimeter Defense* should be performed, in which different protection barriers (preventive, detective, corrective and recovery) must be placed into the IT systems.

Intrusion Detection Systems (IDS) are software or hardware tools that automatically scan and monitor events that take place in a computer or network, looking for intrusive evidences [2]. An intrusion is any attempt to compromise the confidentiality, integrity or availability of information, endangering the security of a computer or network. Network Intrusion Detection Systems (NIDS) take as input network traffic and look for intrusion evidences on complete network segments. In this scenario, an evasion can be defined as any technique that modifies a detectable attack into any other form in order to avoid being detected. The overall idea is to perform some changes to cause that the NIDS does not process the entire attack packet, remaining so undetected. NIDS normally are, in conjunction with firewalls, one of the first objective to deal with when someone is trying to attack a system. That implies that attackers try to develop sophisticated techniques to avoid being detected. In general, NIDS do not give real time information about what is happening, but they log alerts. Human security auditors are then who have to analyze those alerts searching for hostile activity. If the NIDS gives erroneous information, auditor can be distracted and would not be able to focus their efforts in the real attack.

Currently, proposed evasive techniques are based in ambiguities present in transport and network layer protocols (mainly TCP and IP) [1]. Those ambiguities provoke that different systems interpret the protocols in a different way. An attacker attempting to evade NIDS detection can modify the transmitted packets in such a way that lead into a situation where a system has different information than another one. When using NIDS, an evasion can appear if the NIDS and the monitored endpoint interpret protocols in a different way, so the information processed is different in both systems.

Researching in evasive techniques is, along with the discovery and detection of new kind of attacks, the principal tool to improve the effectiveness of the NIDS. Currently, the fast adaptation of the NIDS against new attacks provokes that attackers try to perform evasive techniques (more stealthy and hard to detect) instead of directly exploiting those new attacks. Thus, a security administrator is not aware to have been evaded until posterior forensic analysis of the compromised system, when probably the damage has been done. That is the main motivation of our work, whose primordial objective is to discover new forms of NIDS evasive techniques and, in consequence,improve the NIDS preventive mechanisms against them.

Our idea is to construct models, using Genetic Programming [3], that simulates, as faithful as possible, the NIDS behavior. This model allows us to understand in an easy way how the NIDS works. Furthermore, if NIDS are proprietary, we could have an approximation about its internal functionality. The obtained model is supposed to have an easy semantics, so it would be easier to develop evasive techniques over it than directly trying over the NIDS. Due to the model is assumed to work nearly equal than the NIDS under study, those evasive techniques should work in both the model and the NIDS. In this paper we present an overview of the methodology to be implemented. Also a proof of concept is given, where we have modeled, using

Genetic Programming techniques , a C45 [4] based NIDS (which works well under a port scan scenario).

The remainder of this paper is structured as follows. Section 2 presents the state of the art. Section 3 shows the design details of our work. Next, in section 4 we explain the experimental setup performed, and the results are given in section 5. Finally, the conclusions and future work are discussed in section 6.

2. Related Work

NIDS evasive techniques were first proposed by Ptacek an Newsham in 1998 [1]. In their seminal paper, the authors stressed two main problems in some network protocols when using NIDS. The first is the existence of some ambiguities in the TCP and IP protocols. Those ambiguities allow systems to interpret on their own way how to implement some characteristics of those protocol. For instance, they do not determine what should be done when a packet encapsulates an erroneous checksum field in its TCP header. An evasion succeeds when NIDS ignore packets which are going to be processed on the endpoint systems or vice versa. The second problem presented by Ptacek and Newsham is that some NIDS are vulnerable to Denial of Service (DoS) attacks. An attacker sends several fake hostile packets to the NIDS provoking it to log all the alerts, in such a way that becomes overloaded. In this scenario, the NIDS may not process all the incoming packets, an the attacker could exploit that situation to perform a real attack over the endpoint. Several tools has been implemented with the aim of generate evasive traffic, thus exploiting the properties exposed above. For example, fragroute [5] intercepts network traffic and modifies the packets before forwarding them to their destination, or idsprobe [6], which generates traffic data from original traces.

Current research in techniques designed to prevent evasions are based mainly on network traffic modification, in order to remove the ambiguities of protocols. Thus, a common interpretation of them is established between the NIDS and the endpoint. Watson et. al [7] have proposed a system called Protocol Scrubbing that generates well formed TCP data from traffic. With that, there is only one way to process the information. Handley et al. [8] introduced the concept of traffic normalizers, which are intermediate elements that are located in networks to remove possible ambiguities before being exposed to the NIDS. Because some of the evasive techniques are based on packet fragmentation and reassembly, the state of each connection and the previous packets must be stored and processed, in order to analyze the consistency of connections. That consumes a large quantity of resources, leading into a bottleneck when working with high speed networks [9] .

Other solutions which does not require traffic modification have been proposed. Varguese et al. [10] present an idea based on dividing the entire signature of the NIDS into single smaller strings and use a fast path to find matches with them. If a match is found, then packets are given to a slower, more effective path to inspects the packet in a deeper way. Shankar and Paxon [11] proposed a system that informs the NIDS about the network topology and the interpretation policy of the endpoint being monitored. Thus, the NIDS can adapt its configuration taking into account that information. Snort [12] has adopted this technique in the IP processor (frag3) of its last release. Finally, Antichi et al. [13] propose the use of Bloom Filters to look for signature matching over single received packets without the need of reassembly. These systems improve the efficiency of the NIDS and never gives false negatives (they detect all), but increments largely the number of false positives.

Genetic Programming [3] has been proved to be a good paradigm in the scenario of the NIDS development ([14], [15], [16], [17], [18], [19]). The reason why is because this scenario is continually changing, and new rules or patches are needed. GP can help into the searching process of new solutions in conjunction with the human analysis.

3. Design

In this section we present an overall description of the proposed activity. The main objective is to design and discover new evasive techniques over a NIDS. For that purpose, we use Genetic Programming to model NIDS behavior. Because the model has a simpler semantic than the NIDS, we will try to search evasive techniques over it. We suppose that, if the evasions succeed in the model, it will also do in the NIDS.

Primary, we will generate our own NIDS. Several open source NIDS are available to be used here, but the focus of our work is not to generate a detection engine that works properly under certain conditions, but to generate a model as similar as possible with the NIDS under study. For that purpose, we use a simple, easy to generate, C45 [4] based model. C45 is a classification algorithm that generates a classifier in form of a tree. Once the NIDS is created, we generate the dataset to be used in the modeling (i.e., training and testing) phase.

In order to generate the new model, we need some network traffic dataset. One possibility is to generate our own dataset, in a controlled environment. However, this work is out of the scope of this paper. In section 4 we discuss which dataset we have finally used.

Afterwards, the NIDS is fed with the dataset. The output given is registered and analyzed. Normally most attack traffic will be detected and maybe normal traffic will generate some false alarm. The NIDS will give us one output for each packet, i.e., normal or attack. This output is then appended at the end of the corresponding trace in the dataset, generating traces with the following form:

$$F_1, F_2, F_3, ..., F_N, L, O$$

where each F_i is a field of the packet or connection (for example, the source port, the flag bits, the ip source, etc.) , L is the label which indicates the kind of data (normal, attack or evasive) and O is the output given by the NIDS (normal or intrusion).

We have thus obtained a complete dataset containing several packet traces (both malicious and normal) with their corresponding output given by the NIDS. We use it to model the NIDS with GP. A GP model (individual) consists of a tree where intermediate nodes (which are not leafs) are operators and the end nodes (leafs of the tree) are fields (like the source port, the connection timeout, etc). The output that the model gives should be binary, that is, labeling the data as intrusion (1) or normal (0). We must set the values of each GP parameter, and then run the algorithm (both training and testing phase), obtaining the final model.

A manual optimization of the model is then performed, because many times tree models given by GP are not easy to interpret, and some variable names are not always linked with the field names of the traces (packets). The tree obtained has normally redundant branches or nodes, so performing a pruning phase could be also interesting to improve the efficacy of the model. Although the improvement of the efficacy is not a direct objective to be satisfied, the prune of the tree could largely simplify the model semantics, which is in fact the core of this methodology. The output of this phase must be a model easy to understand and interpret, whose behavior must be as similar as possible with the one of the NIDS. Once the model has been generated and optimized, we can prove some new evasive techniques, on the basis of the model semantics, less complex than the one of the original NIDS. We suppose that, if evasions succeed in the model, they will also do in the NIDS.

4. Experimental setup

We have performed our experiments using internal enterprise traffic from the Lawrence Berkeley National Laboratory (LBNL) Dataset [20]. The dataset was systematically anonymised and only the resulting packet header traces (presented in tcpdump/pcap format) are publicly available. Two kinds of files for the traces are provided, anonymised separately. One corresponds to the non-scanning traffic and the other to the scanning traffic. Because we need several datasets (for both training and testing the model), we have used five intervals of the entire dataset, corresponding to hours between 13:00 and 15:00 hours of the five available days[1]. We have preprocessed the datasets remove both source and destination ip addresses, filtering the dataset in order to obtain only the TCP packets (thus, discarding the UDP data) and merging the non-scanning and scanning traces from the same period into a single text file (thus, obtaining 5 datasets). We have also labeled the traces, appending a character to indicate if each trace corresponds to a normal or scanning packet.

[1]2004/10/04, at 13:23 hours 2004/12/15 at 12:42 hours, 2004/12/16 at 13:17 hours, 2005/01/06 at 13:25 hours and 2005/01/07 at 13:25 hours

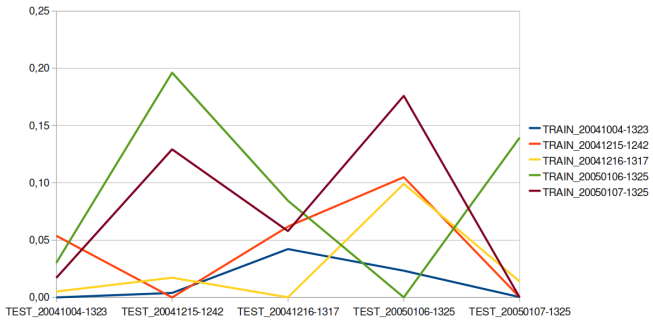

Fig. 1: C45 classification error over the 5 datasets

Due to the fact that we have 5 different dataset (corresponding to the 5 available days), we use each of those dataset once as the training dataset, being the remainder the testing dataset. Thus, we obtain five different models, corresponding to the five different datasets.

The NIDS we want to model is a C45 based NIDS. To build it, we have used the weka tool [21]. We have developed 5 NIDS, one for each training set. That is, we have trained with one of the previously obtained sets, and tested with the remainder, thus five times, obtaining five models with their corresponding test results. Figure 1 shows the degree of accuracy achieved for each training dataset.

The medium test classification error is 7 %, which is not a great result, but as we have told we are not interested in the intrusion detection accuracy, but in the accuracy when modeling an existent NIDS.

Before running the GP algorithm, the value of some parameters must be established. For such a purpose, a cross validation technique is used [22]. This technique can be summarized as follows:

1) Divide the entire training set into k smaller sets (called folds).
2) Establish a random value for each parameter that is going to be set. It is recommended to delimit the valid values to avoid incoherences (for example, an incoherent value could be a number of generations lower than ten).
3) With these parameter values, and for each fold k:
   a) Merge all the folds but k into a set.
   b) Train the model using as training set the previous one.
   c) Test the generated model using the fold k (whose traces does not belong to the training set).
   d) Store the results to be processed later.
4) Return to step 2 and repeat the process as often as desired. To assure enough variability in the values, we recommend to repeat the process at least 4 x n times,

where n is the number of parameters that are being tuning

Once the process is finished, we need to analyze the results to obtain the combination of parameters that gives the best results. The parameters that we wanted to set using this algorithm along with the interval accepted values are:

- Crossover rate. Establish the percentage of individuals on the population which will be selected to be crossed. We have let any value between 0 and 100.
- Mutation rate. Establish the percentage of individuals on the population which will be selected to be mutated. We have delimited to take random values between 0 and 50.
- Size of the population. Number of individuals in each population of the algorithm. Values are restricted to be between 500 and 1300
- Number of generations. The number of generations that the population will "live". Normally, as larger as better, but if the evolution remains stagnant it can be inefficient to use big values. The range of possible values given is from 60 to 200
- Tournament size. Is the number of individuals to be selected to perform the tournament selection (see [3] for more information). Values can oscillate between 3 and 8

We have thrown 25 experiments, so we have tried with 25 different configurations. The best results are presented in Table 1, so it will the one that we have posteriorly used in the real modeling.

Table 1: Values obtained using the cross-validation technique for each parameter

Name	Value
Number of generations	132
Size of population	670
Size of tournament	6
Crossover rate	82%
Mutation rate	29%

The fitness function determines which Individual is better when comparing two or more of them. It is employed in the tournament selection phase when determining the sorted classification of the selected individuals in the competition. The main objective is to obtain a model that classifies the traces as similar as possible as the C45 based NIDS, so, when comparing individuals, we have established as fitness function the classification error, defined as

$$fitness1 = \frac{\#incorrectly\_classified\_instances}{\#instances}$$

In order to obtain easy-to-understand models, we have established a maximum tree depth of 5, so the resultant individuals will have a small size.

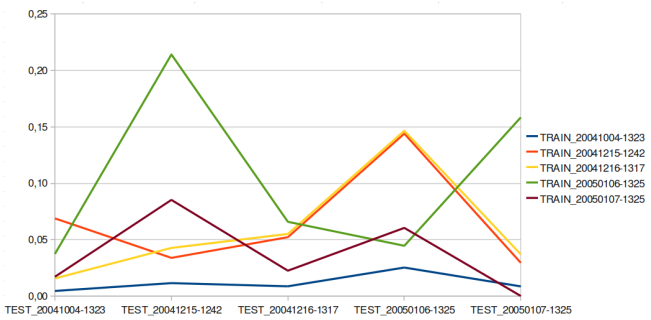

Fig. 2: Classification error when classifying a C45 based NIDS with GP. Best results for each dataset.

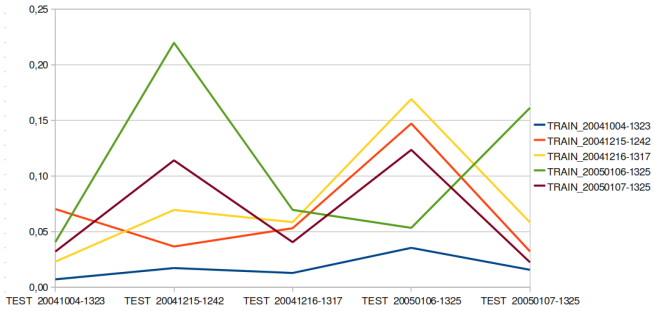

Fig. 3: Classification error when classifying a C45 based NIDS with GP. Average results for each dataset

5. Results

Figures 2 and 3 show the classification error for both the best individual and the average of the 15 individuals obtained respectively. In each figure we can see the test results for each model (each train represents one model, 5 in total). The average test results indicates a classification error equal of 5% for the first and 10 % for second. In both figures we can advise that the dataset of the December 15th, 2004 is quite singular. That is because it's the smaller one, and it has only 7 attack traces and 33580 in total, so its prevalence is much lower than the rest.

We have obtained 5 different models, one for each training set. For instance, we show in the figure 4 the best model found when training with the 20041004_1323 dataset. Note that it has only 14 nodes, in front of the 21 nodes given by the C45, so we have reduced the complexity of the model in a 33 % with an accuracy of 95 %. With that, we are able to design evasive techniques in a easier way than looking for the more complex C45 tree. We have studied the model, and we have been trying to evade the detection in several ways. This process could have been done automatically, but this task is out of the scope of this paper (we will automatize the process in future work). Finally, we have realized that if we change the *src_port* field, resetting it with the value 0, we provoke that the C45 based NIDS fails when trying to detect some attack traces, i.e, under certain conditions

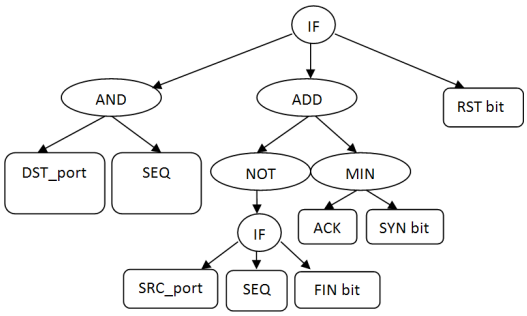

Fig. 4: Example of model obtained using GP

we have evaded the detection. Obviously most of the TCP implementations do not takes into account packets being sent from the port 0, but this is a simple proof of concept of how easily can be a NIDS evaded modeling its behavior with GP.

6. Conclusions and future work

In order to protect systems against new forms of attacks, a fast update is one of the main task to perform by NIDS designers when those new attacks are discovered. However, nowadays, attackers are using new skills when trying to attack IT systems. In order to pass unnoticed when attacking the systems, they must evade the detection engines that monitor those systems. Currently, the major evasive techniques are based on ambiguities when interpreting the network protocols by different systems. Most of these techniques are public and documented, so current NIDS designers are warned about that and can act in consequence. However, the apparition of new evasive techniques would be disastrous for the system administrators who think that all their NIDS are correctly working, because they can be evaded. In this paper we have presented a new methodology that improves the task of looking for new forms of evasion, thus allowing systems administrators to be warned before the attackers could exploit them. We have seen that modeling a NIDS with GP can help when trying to understand the behavior of it (even impossible when we do not dispose the source code). We have modeled a simple NIDS based on the C45 classification algorithm. Our results shows that, with an accuracy of 95 %, we can model the behavior of the NIDS reducing the complexity in a 33 %. With that, it could be easy to look for new evasive techniques. Moreover, looking at the model we have modified some attack traces from the original dataset and we have evaded the detection by the C45 based NIDS. We show how, if the modeling process is correctly done, an evasion thought the model can succeed thought the original NIDS.

We are now challenged to apply the entire methodology to a real NIDS, and to compare the complexities of our model with the original behavior of the NIDS in the cases that is possible (i.e., when the NIDS are open source). We will try also to improve the GP evolution with new fitness functions, new operators or new methodologies.

References

[1] T. H. Ptacek and T. N. Newsham, ”Insertion, evasion and denial of service: Eluding network intrusion detection”, Technical report, 1998.

[2] R. Bace and P. Mell, "NIST Special Publication on Intrusion Detection Systems," 800-31, 2001.

[3] J. R. Koza, ”Genetic Programming: On the Programming of Computers”, M. Press, Ed.Cambridge, MA, USA, 1992.

[4] J. R. Quinlan, ”C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)”, Morgan Kaufmann, 1993

[5] D. Son. (2002) Fragroute. [Online]. HYPERLINK ”http://www.monkey.org/ dugsong/fragroute/”

[6] L. Juan, C. Kreibich, C.-H. Lin, and V. Paxson, ”A Tool for Offline and Live Testing of Evasion Resilience in Network Intrusion Detection Systems”,in *DIMVA ’08: Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Paris, France, 2008, pp. 267-278.

[7] D. Watson, M. Smart, R. G. Malan, and F. Jahanian, ”Protocol scrubbing: network security through transparent flow modification”, in *IEEE/ACM Transactions on Networking*, vol. 12, pp. 261–273, 2004.

[8] M. Handley, C. Kreibich, and V. Paxson, ”Network instrusion detection: Evasion, traffic normalization and end-to-end protocol semantics”, in *Proceedings of the 10th Conference on USENIX Security Symposium*, Volume 10, 2001, p. 9.

[9] M. Vutukuru, H. Balakrishnan, and V. Paxson, ”Efficient and Robust TCP Stream Normalization”,in *SP ’08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2008, pp. 96–110.

[10] G. Varghese, J. A. Fingerhut, and F. Bonomi, ”Detecting evasion attacks at high speeds without reassembly”,in *SIGCOMM ’06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, Pisa, Italy, 2006, pp. 327–338.

[11] U. Shankar and V. Paxson, ”Active Mapping: Resisting NIDS Evasion without Altering Traffic”,in *SP ’03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2003, p. 44.

[12] M. Roesch, ”Snort - Lightweight Intrusion Detection for Networks”, in *LISA ’99: Proceedings of the 13th USENIX conference on System administration*, Seattle, Washington, 1999, pp. 229–238.

[13] G. Antichi, D. Ficara, S. Giordano, G. Procissi, and F. Vitucci, ”Counting Bloom Filters for Pattern Matching and Anti-Evasion at the Wire Speed”, in *IEEE Network Magazine of Global Internetworking*, vol. 23, no. 1, pp. 30-35, Feb. 2009.

[14] A. Orfila, J. M. Estevez-Tapiador, and A. Ribagorda, ”Evolving High-Speed, Easy-to-Understand Network Intrusion Detection Rules with Genetic Programming”, in *EvoWorkshops ’09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, TÃijbingen, Germany, 2009, pp. 93–98.

[15] J. Blasco, A. Orfila, and A. Ribagorda, ”Improving Network Intrusion Detection by Means of Domain-Aware Genetic Programming”,in *Proceedings of the 5th International Conference on Availability, Reliability and Security ARES 2010*, Krakow, Poland, 2010.

[16] G. Folino, C. Pizzuti, and G. Spezzano, ”GP Ensemble for Distributed Intrusion Detection Systems”,in *ICAPR, 2005*, pp. 54-62.

[17] S. Mukkamala, A. Sung, and A. Abrham, ”Modeling intrusion detection systems using linear genetic programming approach”, in *IEA/AIE’2004: Proceedings of the 17th international conference on Innovations in applied artificial intelligence*, Ottawa, 2004, pp. 633–642.

[18] S. Peddabachigari, A. Ajith, C. Grosan, and J. Thomas, ”Modeling intrusion detection system using hybrid intelligent systems”, in *Journal in Network Computer Applications*, vol. 30, no. 1, pp. 114-132, 2007.

[19] M. Crosbie and E. Spafford, ”Applying Genetic Programming to Intrusion Detection”, in *Working Notes for the AAAI Symposium on Genetic Programming*, 1995, pp. 1-8.

[20] Lawrence Berkley National Laboratory and ICSI. (2005) LBNL/ICSI Enterprise Tracibng Project. [Online]. www.icir.org/entreprise-tracing/

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The WEKA Data Mining Software: An Update", in *SIGKDD Explorations*, Volume 11, Issue 1,2009

[22] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", in *IJCAI: International Joint Conference on Artificial Intelligence*, Volume 2, Issue 1, 1137–1143, 1995