A Functional Framework to Evade Network IDS

Sergio Pastrana
Carlos III University of Madrid
spastran@inf.uc3m.es

Agustin Orfila
Carlos III University of Madrid
adiaz@inf.uc3m.es

Arturo Ribagorda
Carlos III University of Madrid
arturo@inf.uc3m.es

Abstract

Signature based Network Intrusion Detection Systems (NIDS) apply a set of rules to identify hostile traffic in network segments. Currently they are so effective detecting known attacks that hackers seek new techniques to go unnoticed. Some of these techniques consist of exploiting network protocols ambiguities. Nowadays NIDS are prepared against most of these evasive techniques, as they are recognized and sorted out. The emergence of new evasive forms may cause NIDS to fail. In this paper we present an innovative functional framework to evade NIDS. Primary, NIDS are modeled accurately by means of Genetic Programming (GP). Then, we show that looking for evasions on models is simpler than directly trying to understand the behavior of NIDS. We present a proof of concept showing how to evade a self-built NIDS regarding two publicly available datasets. Our framework can be used to audit NIDS.

1. Introduction

Information Technology systems have become a critical component in organizations that manage a huge amount of personal and critical data. Guarding those systems from hostile actions should be the main goal when applying security measures, but the continuous evolution of technologies makes this task difficult.

Intrusion Detection Systems (IDS) are software or hardware tools that automatically scan and monitor events that take place in a computer or a network, looking for evidence of intrusion [1]. Network Intrusion Detection Systems (NIDS) just analyze network traffic captured on the network segment where they are installed. NIDS may seek for either anomalous activity (anomaly based NIDS) or known hostile patterns (signature based NIDS) on the network. Unlike firewalls they do not normally block packets, but raise an alert about the intrusion attempt.

Signature based NIDS are effective at detecting attacks they are prepared for (they may fail to detect zero-day attacks until their signatures become updated). This situation causes attackers to focus their efforts in finding evasions over the signatures of these systems. The concept of evasion was first proposed by Ptacek and Newsham [2]. The authors highlighted some ambiguities in network protocols (concretely TCP and IP) that can lead into a situation where NIDS and endpoint systems process packets in a different way. An evasion succeeds if the processing of the packets generates a different representation of the raw data in the NIDS and in the end systems. Data contained in TCP segments can encapsulate some attacks, but if the NIDS processes those segments differently from the endpoint, it will not be able to detect those attacks.

To the best of our knowledge no new evasive procedures have been published since [2]. Security administrators adapt their systems to detect those well-recognized problems presented by Ptacek. Thus, a possible emergence of new evasive techniques would be critical for systems that are supposed to be secure. This is the motivation of our work, in which we present a new approach to look for evasions over NIDS, giving a proof of concept in which we put into practice our framework using an artificial scenario.

The aim of our framework is to look for new evasive techniques by analyzing NIDS behavior. As this behavior is normally complex, and even obscure if the NIDS source is proprietary, our approach is to model it by means of Genetic Programming (GP) [3]. GP is a paradigm that evolves programs that are represented by trees, where intermediate nodes are functions and leafs are terminals. The evolution is performed by selecting the best individuals (by comparing them using a denominated fitness function) from an entire population to cross and mutate them, thus obtaining the next population. In the scenario of our work, we have evolved simple models (individuals) that behave as similar as possible to the original NIDS. Thus, a simpler representation of the NIDS is obtained. The GP model allows finding evasions otherwise difficult to find.

In a previous work [4], GP was used to model a simple NIDS with great accuracy, using a publicly

1530-1605/11 $26.00 © 2011 IEEE

available dataset. In this paper we present new improvements, performing evasions over that NIDS and corroborating the effectiveness of modeling NIDS with GP using another publicly available set.

The remainder of this paper is structured as follows. First, the state of the art on NIDS evasion is presented in section 2. Second, section 3 presents the framework. Then, section 4 describes the goals and requirements of this work. Experiments performed and results obtained are depicted in sections 5 and 6 respectively. Finally, our conclusions and future work are gathered in section 7.

2. State of the art

Evasions on NIDS were first proposed by Ptacek and Newsham in 1998 [2]. In this seminal paper, the authors highlighted the existence of some ambiguities in the TCP and IP protocols, which allow different systems to implement them in a different way. An evasion succeeds when NIDS ignore packets which are going to be processed on the endpoints or vice versa. For example, TCP does not specify what should be done with TCP packets containing an erroneous checksum field. Implementations of the TCP protocol can ignore, accept or reject those packets. As shown in Figure 1, an evasion could succeed if the NIDS implementation of the TCP protocol differs from the endpoint implementation.

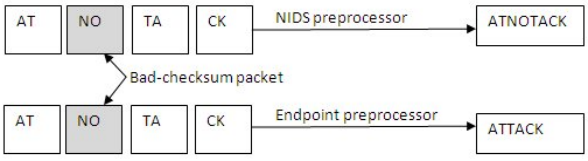

Figure 1: example of evasion. In this example, the NIDS preprocessor accepts the packet containing a bad-checksum field, while the endpoint does not, so the final structure after the preprocessing phase will be different.

Several tools have been implemented to exploit the properties exposed above. Examples are Fragroute [5], which intercepts network traffic and modifies the packets before forwarding them to their destination, and idsprobe [6], which is a tool that generates traffic data from original traces.

Many techniques have been designed to prevent evasions. Most of them are based on network traffic modification, to remove the ambiguities and establish a common understanding of the protocols for NIDS and endpoints. Watson et. al [7] propose a system called Protocol Scrubbing that generates well formed TCP data from traffic. A similar approach was proposed by Handley et al. [8], who introduced the concept of traffic normalizers. Those are intermediate elements that are placed in networks segments 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 the connection. This situation consumes a large quantity of resources, leading into a bottleneck when working with high speed networks [9].

Some other solutions that do not modify the traffic have also been proposed. Varguese et al. [10] present the idea of dividing the entire signature of the NIDS into single smaller strings. A fast path finds matches with them and a slower one inspects it deeper if any match is found. Shankar and Paxon [11] propose a system that reports the NIDS about network topologies and the interpretation policy of the endpoint being monitored. Thus, NIDS can adapt their configuration taking into account that information. Snort [12] has adopted this technique in its IP processor (frag3). Finally, Antichi et al. [13] propose the use of Bloom Filters to find signature matching over the single received packets without the need of reassembly. These systems are supposed to improve the efficiency of the NIDS and never gives false negatives (detects all), but increments largely the number of false positives.

A similar approach to our work has been presented by Mutz et. al [14]. The authors propose some kind of reverse engineering to obtain the signatures of non proprietary NIDS, so they became more vulnerable to attacks or evasions. Their principal goal is to show how keeping the signatures secret does not necessary increment the security of the NIDS. The main difference with our approach is that we do not focus on evading the NIDS directly but modeling it first.

Genetic Programming has been proven to be a good paradigm in the scenario of NIDS development [15,16,17,18,19,20]. The main reason is that the functions used by GP can be defined in advance for a particular scenario. Accordingly, it is appropriate for the intrusion detection domain. In addition, previous work [4] demonstrates the viability of using GP to model a simplified NIDS with an easy to understand syntax and semantics.

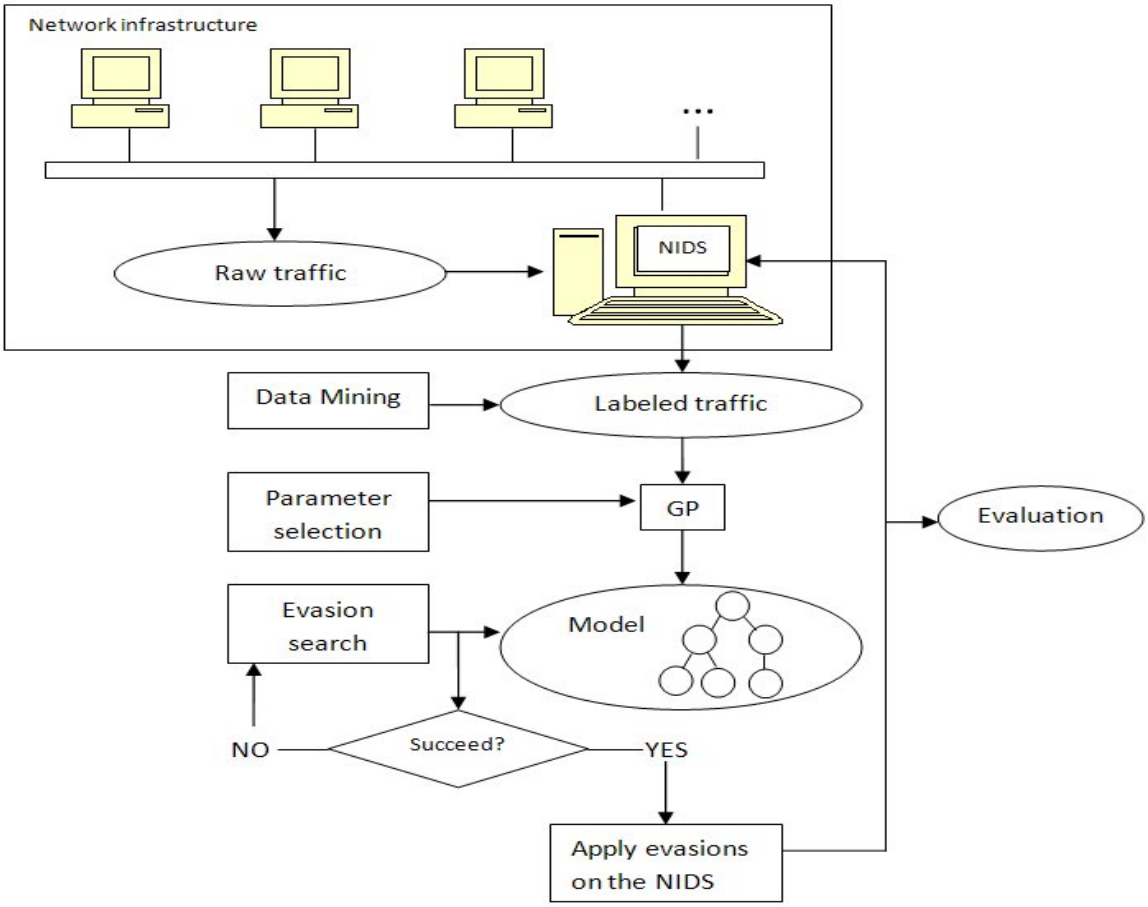

Figure 2: Overall scheme of the proposed framework

3. Framework

In this section we present an overall description of the proposed framework. Figure 2 shows a graphical description. The main objective is to look for new evasion techniques on a given NIDS. We use GP to obtain a model that classifies as similar as possible to the NIDS. Due to the use of a simple syntax, the GP model has a simpler semantics than the NIDS. Looking for evasive techniques over the model is easier than over the NIDS. If evasions succeed over the model, and given that this model may have a quite similar behavior than then original NIDS, it is likely that the evasions will also succeed over the NIDS. Our framework is composed of a set of tasks described in the following sections.

3.1. Generation of the datasets

The GP modeling process at issue requires a labeled dataset. This dataset must represent as well as possible real traffic. Due to the necessity of generating different traffic profiles, a controlled environment is required. Generated traffic should include normal (simple web requests, remote connections, web navigation, etc) and intrusive (malicious) traffic. Traffic is processed by means of data mining techniques to extract the most significant features. It also needs to be labeled in order to identify it as normal or hostile.

Obtained traffic should be exposed to the NIDS, which analyzes the dataset looking for intrusive actions. Output given by the NIDS is appended to its corresponding processed frame. Thus, the obtained dataset is composed of registers with the form:

$$F_1,F_2,F_3,\ldots,F_N,L,O$$

Where each F_i is the field i of the trace (for example, the source port, the flag bits, the amount of data exchanged, etc.), L is the label which indicates the nature of data (normal or attack) and O is the output given by the NIDS (normal or intrusion).

The overall dataset is then divided into smaller sets, one being the GP training subset and the remainder the GP testing subsets.

3.2. Modeling the NIDS

As we have mentioned, in our framework GP is used to model the behavior of NIDS. First, values for some Genetic Program parameters are established. This process can be made manually or automatically, for example using a cross-validation technique [21]. This technique consists of performing the GP modeling phase several times, by using different combination of parameters. The entire dataset is randomly divided into 10 subsets, called folds. Each training phase is performed with one fold, using the remainder to test the evolved model. When all folds have been used to train a different model, it is taken as final combination of parameters the one that gives the best results in test. The principal advantage of using this technique is that we explore several combinations of parameter values so we can assure that we are using an optimum values for them, as the training phase is performed with all the different subsets (folds) of the entire dataset, so it does no depends on an initial selection, but in the complete dataset.

The election of a good fitness function is a critical component when using GP. Because we are searching classification models, an optimum fitness function can be the classification error, that is, the rate that indicates how many traces has been correctly classified, not taking into account whether those traces are positives or negatives. Once the parameters are fixed, we obtain the NIDS models by training them with the entire training subset (which has to be considerably bigger than the remainder, used for testing). Then, we perform the test of the obtained models using the testing set. Results must be stored to be processed afterwards.

Because the GP search is heuristic, it is appropriate to perform the training phase several times, using different random seeds, taking the results for the best individual (the one that has produced the best test results) and the average of the individuals. Using different random seeds covers a bigger searching space.

A manual optimization of the model is then performed. The tree model obtained has normally redundant branches or nodes, so performing a pruning phase could be interesting to improve the efficiency of the model. Although the improvement of the efficiency is not a primary objective to be satisfied, the prune of the tree largely simplifies models semantics, which is in fact the core of this framework. The output of this phase must be a model easy to understand and interpret, whose behavior must be as similar as possible to the NIDS.

3.3. Analysis and design of evasive techniques

Once the model is obtained, it is analyzed in order to discover some points of the internal structure of the NIDS, thus conceiving an idea of its behavior. Mainly, the model indicates which are the fields that the NIDS takes into account to classify traces. This information is used to perform a brute force modification of those fields.

The idea is to automate the process by changing the value of the fields that are present in the GP model, generating new modified traces. Before changing the value, it should be assured that traces with the new value remain being attacks and still coherent with the protocols. For that purpose, a set of rules must be established and fulfilled, indicating which variables can be changed and which values can be set to them. New valid values are given for those fields in hostile traces which were previously detected by the NIDS (true positives), establishing a new dataset composed by old and new (modified) traces. Then, the NIDS is applied to those new modified traces. New false negatives would indicate that the evasions performed have been successful. The process is repeated for each field that appears in the GP model, and also multiple simultaneous changes (to more than one field at the same time) can be done.

4. Proof of concept. Specific goals

The two main objectives of the proof of concept presented are first, to corroborate that GP can be a good paradigm to model NIDS, and second, to find evasions over the NIDS analyzing the corresponding GP model. For that purpose, we have created a basic NIDS based on the C4.5 algorithm [22]. This algorithm is a supervised learning classifier whose output is a tree.

A simplification of the framework has been made to fulfill our goals. Instead of creating a specific dataset, in this work we have opted to use the only two publicly available datasets that are labeled (as normal or intrusive). Previous work [4] used one of them, the Lawrence Berkley National Laboratory (LBNL) dataset [23]. Results showed that with an accuracy of 96%, the behavior of a self-built NIDS can be modeled by reducing its complexity. In this work, we improve the study by using a different dataset, the KDD-99 [24], derived from raw traffic captured during MIT/LL 1998 evaluation [25]. A complete description of the fields can be found in [26]. The use of an extra dataset corroborates that the accuracy of using GP to model a NIDS is not limited

to one scenario and attack (the LBNL using port scanning attacks), but also to another that uses several kind of attacks. It is obvious that these datasets are both quite old, taking into account the fast growth of the complexity in the Information Technologies. However, they have been widely used in the literature [15,16,23] and they provide a huge set of labeled traces. Thus, it is useful for providing insight into the problem at issue and to analyze if the idea is sound.

After obtaining models for each dataset, we are challenged to find real evasions over the original C4.5 based NIDS, by firstly looking for them over each GP model. We look for evasions by modifying the value of one or more fields of the traces and exposing them to the original NIDS. We must choose fields and values in such a way that the traces remain coherent with protocols, being still attacks (for example, if we change the bit of some TCP flag in a port scanning attack, we are not evading the NIDS and attacking the endpoint, but transforming the malicious trace into a normal one). For that purpose, we need to analyze the nature of the attacks we are working with. It is also needed that traces to be modified are true positives. An evasion is considered successful if, after the modification of the trace, the NIDS does not detect it as an intrusion.

5. Experimental work

Figure 3 shows a scheme of the modeling phase. At first the datasets are prepared. The LBNL provides both normal and portscanning traffic, captured in various days at different hours. We use five different raw traffic files, processing them in order to take just TCP traffic. Thus, we establish five datasets containing labeled traces from both malicious and normal nature. These traces are composed of the fields (F_i) of the TCP header.

In the case of the KDD dataset, we have taken 10% of the original traces, preprocessed them in order to make the output binary (i.e. normal or intrusion) and normalizing the non-numerical fields.

We use the weka tool [28] to obtain the C4.5 based NIDS (step 1 in Figure 3). For that purpose, we randomly choose a subset of each dataset to perform the training phase, testing over the remainder. This testing phase provides, for each trace, the output given by the NIDS, i.e. if it has properly classified the trace or not. This information is appended to each trace, obtaining the final dataset (step 2 in the Figure 3). We perform another division of the dataset, in this case to obtain two new different subsets, one to be used in the GP training phase and another one to test the individuals (step 3 in the Figure 3).

Table 1 shows the performance of the NIDS created for both the LBNL dataset and the KDD. As can be observed, in the case of the LBNL, NIDS are tending to classify the traces as intrusive, so its detection rate and its false alarm rate are both very high. However, the NIDS built with the KDD has lower rates, which indicates that it is more likely to classify the traces as normal. So, given that the NIDS which are going to be modeled are very different in nature, the first goal of our proof of concept, which was to prove the feasibility of using GP to model NIDS goes a step further.

	Detection rate	False Alarm rate
LBNL	99%	65%
KDD	82.43%	0.1%

Table 1: Performance of the self-built, c4.5 based NIDS. In the case of the LBNL, the rates correspond with the average of the five raw traffic files.

GP works with trees that represent programs (called individuals in GP terminology) to be evolved. Each intermediate node of the tree is a function, and each leaf is a terminal. The terminals we use are both fields of the traces and real numbers. Table 2 shows the functions used for intermediate nodes. It is important to note that the election of these functions helps to produce simple individuals.

Name	Description
ADD	Addition of two numbers
AND	And logic operation between two numbers
DIV	Division between two numbers, or 0 if the denominator is 0
GT	Compares both values. Returns 1 if the first is greater, 0 in the other case
LT	Compares both values. Returns 1 if the first is lower, 0 in the other case
MULT	Multiplication of two numbers
OR	Or logic between 2 numbers
MAX	Returns the maximum of two values
MIN	Returns the minimum of two values
RL	Bitwise rotation in one position of a value
NOT	Not logic operation between two values
IF	Returns the second child if the first one is positive or 0, or the third child if the first one is negative

Table 2: GP function set

The models are created by first evolving them using a training subset (step 4 in Figure 3), using the remaining subsets to test whether the obtained models have a good performance with different traffic from the one used to evolve them (step 5 in Figure 3). In the training phase, that is, when

evolving the models, we use an Island Model [3], where different evolutionary environments are thrown in parallel. Thus, every 20 generations, the best individual of each environment (island) migrates to another island. Some of the fields obtained under an island are moved to another, in order to improve the variability of the global evolution.

One critical component in GP is that it performs a heuristic search. Accordingly seven different seeds have been used over each training subset, thus obtaining seven different evolved individuals. Then, a testing process is performed with each individual. In the following section the best and average result for each model is shown. Each individual represents one different NIDS model, and because they must be as simple as possible, a maximum depth of 4 is established. Therefore, each individual have at most 81 nodes including functions and terminals (the maximum number of children of a node, 3, at the power of 4).

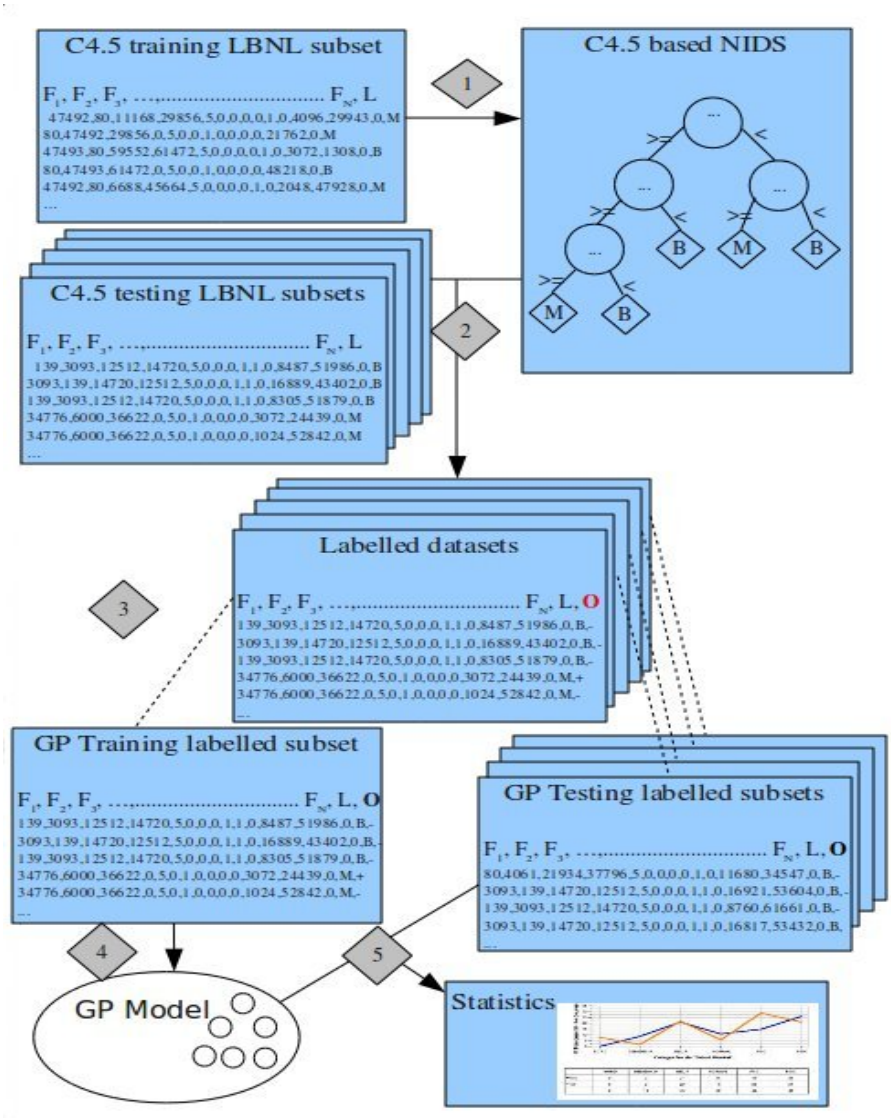

Figure 3. Graphical description of the experimental work performed. Firstly (1), the C4.5 based NIDS is created with one subset of the dataset (in LBNL, we create 5 different NIDS using the 5 different used files). Then (2), some other traffic files are presented to the NIDS in order to obtain its output for each trace, which is appended at the end of the traces. The labeled traces files are divided into a training file and several testing files (3). The training file is used to evolve (train) the GP model (4), which is later tested with the testing files in order to obtain statistics of its output (5).

As it was previously stated, one of the goals of this proof of concept is to corroborate that GP is a good paradigm to be used when modeling the NIDS. In order to compare with some other techniques, we have obtained models using two different techniques. Concretely, we have used the Naïve Bayes approach, which is a specific Bayesian classifier which assumes strong independence among fields [29] and whose output is not a tree, but a probabilistic model. The second method used is the C4.5 algorithm, which is the one used to create the NIDS under study, but limiting its maximal tree depth to 4 (as done in GP). It is obvious that the C4.5 algorithm will reach better results if its maximal depth would not be limited to 4, because it is the algorithm used to obtain the original NIDS. However, this limitation of the maximal depth is needed to assure that the complexity of the models to be compared is similar. Table 3 shows the classification error achieved by the three approaches when classifying the LBNL traces as done by the self-built, C4.5 based NIDS. As can be seen, the GP model achieves the best performance.

	GP	Naïve Bayes	C4.5
Classification error	4 %	37.4%	8.6 %

Table 3. Comparison of the three methods used to model the NIDS in terms of classification error for the LBNL dataset. The GP model is the best one of the seven different models obtained.

At this point (final step in Figure 3) several GP individuals are obtained for the LBNL dataset (35 individuals in total, five datasets and seven individuals for each one) and the KDD (one dataset, 7 individuals). In order to evade these models, we are interested in changing traces corresponding to true positives. We analyze the models manually looking for any field that, when changed, will make the NIDS to fail in the detection. It is possible that there is no possible change that causes the evasion of the NIDS. In this case, we should repeat the modeling process (by changing some field or the fitness function) in order to obtain another model over which we would look for new evasive methods.

6. Results

One of the objectives proposed in this work is to verify that using GP it is possible to model NIDS accurately, thus obtaining easier to understand models which, working as white box, allows us to understand the behavior of the NIDS and to find new ways to evade detection. Table 4 shows the statistics obtained in the GP modeling phase. The third column is calculated by dividing the number of nodes of the GP models by the number of nodes of the C4.5 tree. For both cases, we have calculated the medium nodes of all the obtained GP models (7 for the KDD case, and 35 for the LBNL). As we can see, we obtain a great complexity reduction with a slight classification error.

	Best	Average	Complexity reduction
LBNL	4 %	12.5 %	86%
KDD	3 %	8.6 %	98%

Table 4. Statistics of the GP classification phase. The first two columns show the classification error measured for the NIDS model for both the best individual and the average of the individuals obtained. The third column shows the reduction in complexity achieved by the average GP trees compared to the original C4.5 tree.

This reduction of complexity allows us to take a look at how the NIDS works. Realize that, being the C4.5 an algorithm whose output is a tree, it could be possible to understand its behavior by examining this tree. However, as Table 4 shows, GP trees are less complex than those of C4.5, being easier to understand. Once obtained the models, we look for possible evasive points over them. For instance, we show in Figure 4 an example of an obtained model for the KDD dataset. In this case, classification is made by dividing the variable 13 (the number of compromised conditions [24]) by the variable 6 (the number of data bytes from destination to source). As described in Table 2, if the denominator of the division is 0, a 0 is returned. In order to change that condition, we can change the variable 6 in order to modify the classification, thus setting it to a non-zero value.

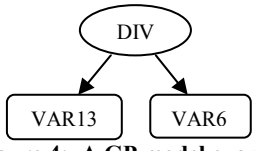

Figure 4: A GP model example

Before changing the value, we must realize that traces with the new value remains producing the original attack. For that purpose, we analyze a KDD attack, *Neptune*, which consists on a SYN Flooding attack. The attacker sends to the victim several SYN packets, indicating that a new connection is required.

The victim must store information about each new connection that is being established, storing it in a buffer which, after receiving several SYN packets, will be overloaded. This attack can be distinguished from normal traffic by looking for a number of simultaneous SYN packets destined for a particular machine that are coming from an unreachable host. A NIDS can monitor the size of the TCP connection data structure and alert a user if this data structure approximates its size limit [27].

The field dst_bytes, which we modify to perform the evasions, is not taken into account when generating the Neptune attack, so we can modify it in the traces without disturbing the condition of attack. On the one hand, sending traces with all the fields having the same value, but the destination bytes, we are performing the same attack. On the other hand, we have realized that changing to 100 the value of the field, the detection of the attack is evaded in all the traces of the training subset which belong to this attack. In summary, we are performing one attack that originally was detected by the NIDS, but after the changes, remains undetected, thus evading the NIDS detection engine. This situation, in a real environment, would be critical for systems that are receiving a huge amount of data, because if the NIDS is able to detect that a Neptune attack is happening, some actions could be taken. But if the detection does not succeed, that is, if the NIDS is evaded, the endpoint system buffer would be overloaded, thus producing a Denial of Service as the state of new benign incoming connections would not be stored.

Regarding LBNL, malicious traces are part of a port scanning attack. The aim of this attack is to probe the open ports of the victim system, in order to find open doors to perform a posterior attack. It is normally done by sending simple SYN packets over each of the ports that are being scanned, analyzing the response to that requests, so it can be determined whether the port is open, close or filtered (by a firewall) [30].

One variant of port scanning, used for example by Nmap [30] is based on the value of the window size field of a TCP header. This variant is called TCP Window Scan. The attacker examines the window size value on packets received with the RST bit set on, and depending on this value, it can be determined the nature of the system being attacked.

In a similar process as the one explained for the KDD dataset, we have realized that changing the window size value to 34 (see Figure 5) we can evade the NIDS detection of some attack traces (around a 69 % of them) in 2 of the 5 models we are working with. A deeper analysis of the models would allow us to improve the effectiveness of the evasions, for example by changing more than one field. The packets that we are modifying to evade the NIDS has the RST bit set to 0, so we can assure that they are not belonging to a TCP Window Scan attack. As we are changing the value of the Window Size field in those traces with the RST bit being 0, we assure that, after the change, the port scan attack is still succeeding, but in this case without being detected by the NIDS. This situation in a real environment would provoke that the port scanning phase, which is the previous step to any other more critical attack, could be performed with no alert from the NIDS.

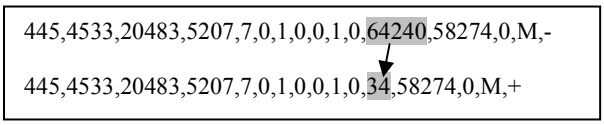

Figure 5: An evasion example, showing the original and modified trace. Last character indicates whether the NIDS has properly classified the trace (-) or not (+). Second last character shows that both traces are malicious (M). In boldface it is shown the change performed to the window size field.

7. Conclusions and future work

Currently, NIDS are prepared to detect a huge variety of attacks. Some of them, like Snort, take into account the possibility of being evaded with the techniques exposed by Ptacek and Newsham in 1998 [2]. However, they are not prepared to new evasive forms that can appear.

In this paper we present a new framework to look for evasions over a given NIDS. The core of the framework is to model the NIDS using Genetic Programming to obtain an easier to understand individual which works as similar as possible to the NIDS. This model allows the understanding of how the NIDS classifies network data. Once this model is obtained, we can look for some way of evading the NIDS detection by changing some of the fields of the packets. The final aim of using our framework is not to break the detection of the NIDS, but to analyze NIDS robustness.

We have tested our framework by using a simple NIDS based on the C4.5 algorithm over the only two publicly available datasets that have their records labeled. We have shown the effectiveness and degree of reduction of the complexity when using GP to model the behavior of the NIDS. Taking advantage of this reduction, we have provided evasions over the original C4.5-based NIDS. Concretely, we have found evasions that allow attackers to perform a SYN flooding attack and a port scanning attack to systems in such a way that the NIDS would not detect them.

This situation would provoke critical situations under a real scenario.

We have two main objectives for incoming work. One is to create our own dataset, as explained in the section 3.1, to perform the experiments. The other is to analyze if these techniques can be applied straightly to model a commercial NIDS.

Acknowledgment

This work was partially supported by CDTI, Ministerio de Industria, Turismo y Comercio of Spain in collaboration with Telefonica I+D, Project SEGUR@ CENIT-2007 2004.

8. References

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

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

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

[4] S. Pastrana, A. Orfila, and A. Ribagorda, “Modeling NIDS evasion with Genetic Programming”, on the Proceedings of The 2010 International Conference on Security and Management, SAM 2010, Las Vegas, Nevada, USA, July 11-15, 2010

[5] D. Son. (2002) Fragroute. [Online]. 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," 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," IEEE Network Magazine of Global Internetworking, vol. 23, no. 1, pp. 30-35, 2009.

[14] D. Mutz, C. Kruegel, W. Robertson, G. Vigna, and R. A. Kemmerer "Reverse Engineering of Network Signatures", in Proceedings of the AusCERT Asia Pacific Information Technology Security Conference, Gold, 2005

[15] 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, Tübingen, Germany, 2009, pp. 93--98.

[16] J. Blasco, A. Orfila, and A. Ribagorda, "Improving Network Intrusion Detection by Means of Domain-Aware Genetic Programming," in , Krakow, Poland, 2010.

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

[18] 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.

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

[20] 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.

[21] 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

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

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

[24] S. Hettich and S. Bay. (1999) The UCI KDD Archive. [Online]. http://kdd.ics.uci.edu

[25] I. Fraf, R. Lippmann, R. Cunningham, D. Fried, J. Kendall, S. Webster, and M. Zissman, “Results of DARPA 1998 offline intrusion detection evaluation”, 1998, DARPPA PI Meeting, Cambridge, Massachusetts, USA.

[26] W. Lee and S.J. Stolfo. “A framework for constructing features and models for intrusion detection systems”, ACM Transactions on Information and System Security, vol. 3, no. 4, pp 227-261, 2000

[27] A.C. Smith and K. Kendall Thesis, “A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems”, in Proceedings DARPA Information Survivability Conference and Exposition (DISCEX), 1999, 12--26

[28] 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

[29] N. Friedman, D. Geiger, M. Goldszmidt, “Bayesian Network Classifiers”, Machine Learning, vol. 29, issue 2, pp 131-163, 1997

[30] Nmap. [Online]. HYPERLINK "http://nmap.org/"