A Neural Network Application for Attack Detection in Computer Networks

Lília de Sá Silva *lilia@dss.inpe.br*
Adriana C. Ferrari dos Santos *aferrarisantos@directnet.com.br*
José Demisio S. da Silva *demisio@lac.inpe.br*
Antonio Montes *montes@lac.inpe.br*

Instituto Nacional de Pesquisas Espaciais – INPE
Av. dos Astronautas, 1758, Jardim da Granja
São José dos Campos, SP, BRAZIL

Abstract – **This work presents a network intrusion detection method, created to identify and classify illegitimate information in TCP/IP packet payload based on the Snort signature set that represents possible attacks to a network. For this development a type of neural network named Hamming Net was used. The choice of this network is based on the interest to investigate its adequacy to classify network events in real-time, due to is capability to learn faster than other neural network models, such as, multilayer perceptrons with backpropagation and Kohonen maps. A Hamming Net does not require exhaustive training to learn. TCP/IP packet payloads were used as input pattern to the Hamming Net and Snort signature as exemplar patterns. The challenges faced to model the input and exemplar data and the strategies adopted to capture and scan relevant data in TCP/IP packets and in Snort signatures are described in this paper. In addition, the application architecture, the processing stages and some test results are presented.**

1. Introduction

Different tools, used to monitor users and system events in order to detect illegitimate and abnormal activities in the network, named Intrusion Detection Systems (IDS), have been implemented considering the use of Artificial Intelligence (AI) technologies.

Neural networks models have been become a promising AI approach to improve the search for malicious events or invasion attempts in computer networks due to their capabilities to compact knowledge representation.

Instead of processing program instructions sequentially, neural network based models on simultaneously explore several hypotheses, making the use of several computational interconnected elements (neurons) [4], this parallel processing may imply time savings in malicious traffic analysis.

The initial motivation for this work was to study the feasibility of using the Hamming Net neural network if fast classification of illegitimate activities during the monitoring of the computer network traffic.

The use of the Hamming Net is due to the fact this network may be viewed as a fast pattern matcher that computes the distance between an input pattern vector and several patterns stored in its weights during the training phase, which consists of direct inserting of the patterns in the weights. This may imply fast update for novel detected intrusion patterns. Fast processing may mean faster packet data classification, allowing rapid response in case of incidents. Moreover, for each input (TCP packet content) the Hamming Net finds the most similar class, according to a pre-defined similarity threshold, providing a great flexibility and fault tolerance by finding small attack variations.

During our study the necessity of restrict the development scope was verified, due to some problems real-time packet content caption. So, it was decided to primarily explore the use of Hamming Net in the classification of data from packet content in off-line mode, to test the feasibility of the method. Thus, files with real events content were obtained and prepared to simulate a scenario as close as possible to the real environment. Thus, the main objective of this work is to create an application to detect post-mortem malicious content in TCP/IP packet payload using the Hamming Net classifier neural network. The signature patterns were obtained from the Snort.

In section II a brief review of intrusion detection systems and description of the Snort basic elements are presented. In section III a summary related to the neural network use for data classification as well as the Hamming Net operation are described. The structure of the implemented system and final considerations, including ideas for future work, are found in the sections IV and V, respectively.

2. Intrusion Detection System

Intrusion Detection Systems (IDS) are important security tools used to detect possible network attacks, inappropriate,

0-7803-8359-1/04/$20.00 ©2004 IEEE

incorrect or abnormal activities, and to alert the occurrences to network administrators.

There are two kinds of IDS's that are classified according to the targets of monitoring [1]:

- *Host-Based* IDS uses the log files or audit trails to monitor the system. It is used to monitor accesses and modifications in critical system files, changes in users' privileges, processes, programs that are being executed, and CPU usage, among others.
- *Network based* IDS monitors the system through the capture and analysis of package headers and contents, which are compared with known patterns or signatures previously stored in rules or signatures files.

Considering the detection method [8] there is another classification:

- IDS based on Signature or Misuse-based, that seeks for incidents of intrusion scenarios or signatures and requires previous knowledge of the intrusion nature;
- IDS based on behavior or Anomaly-based, that considers unknown intrusion nature and event different from normal behavior (profile) to be malicious activity.

The IDS developed in this work is based on Network and Signature, and the considered signatures were collected from Snort.

A. *Snort*

Snort is an open code IDS based on network and signature that combines rules and pre-processors to analyze the network traffic. The Snort intrusion detection process applies rule filters to each package in order to find attack signatures [2].

With Snort, users can evaluate and modify the large existent rule set. Nowadays, Snort has approximately 3000 rules. It is not recommended that all rules be used in Snort installation, because as more rules are used, the traffic inspection becomes slower. The analyst may decide to use only those rules that are interesting to the network environment observed.

1) A rule Snort Anatomy

A Snort rule is divided into two parts: the header and options, according to Figure 1.

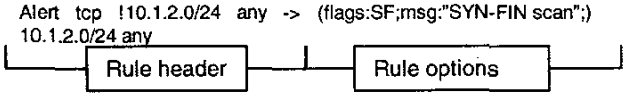

FIGURE 1. Snort rule composition.

The 'Rule header' part contains the hosts and ports identification to be applied to the 'Rule options' part [2]. The rule options define the parameters to be considered in the search of malicious patterns. This includes the information of TCP payload header or content. All conditions specified in both, rule header and options, should be true to dispatch an alert message or some other event.

The 'Rule header' in Figure 1 presents a keyword dispatched (for example, an alert) corresponding to an event when the rule matches against the input pattern and also presents the information regarding the values within the package header and also presents the information related to the values contained in the package header (for example, network IP address). In this rule, an abnormal pattern is detected for TCP network traffic coming from any port of a network different of 10.1.2.x to 10.1.2.x port network.

In the 'Rule options' part, the specification of the package attributes is presented. In this case, the pair of anomalous TCP flags, SYN and FIN, is searched and if found a message "SYN-FIN scan" is displayed together with an alert message.

Snort dispatches an event when the first rule matches the package and does not examine the remaining rules. The sequence in which the rules are placed in the rule files is important. By default, all rules are organized by its action value in the following order: alert, pass and log. However, Snort accomplishes a posterior ordering to group identical headers.

2) Content Option

The '*Content* option' is one of the most powerful options in a Snort rule because it provides the information to be investigated on a package payload.

The search on the payload content is considered to be computationally expensive, that is, the processing to search data can be very slow if it is not built in an intelligent way. Although Snort developers have maximized the efficiency of the search algorithm, the operation is still expensive when compared to a more exact task, like matching a value within the header field. This occurs because the existing value in the header field is known to be 4 bytes long, while the payload is generally very large, requiring a long time to be scanned. Figure 2 presents the format of a TCP/IP package. The interesting field for this work is the 'TCP payload' that contains the string '*content*' to be compared with the Snort rules.

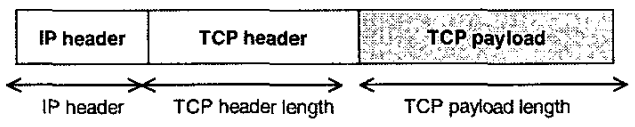

FIGURE 2. TCP/IP Packet Parts.

'String content' (signatures) can be represented as a text, in hexadecimal format, or a combination of both. Text strings are included between quotation marks (""). The hexadecimal code is defined by bars (|). Figure 3 illustrates an example of Snort rule with the string content " anonymous ".

Sample rule:

```
alert tcp any any -> 192.168.5.0/24 21   \
(msg: "Attempted anonymous ftp access";  \
content: "anonymous"; offset: 5;)
```

Sample output:

```
[**] Attempted anonymous ftp access [**]
04/24-12:11:08.724441 192.168.143.15:3484 -> 192.168.5.16:21
TCP TTL:64 TOS:0x10 ID:30124  DF
***AP*** Seq: 0x93EE0A87   Ack: 0x88352E61   Win: 0x7D78
TCP Options => NOP NOP TS: 112024246 27551686
55 53 45 52 20 61 6E 6F 6E 79 6D 6F 75 73 0D 0A  USER anonymous..
```

FIGURE 3. Snort rule example [2].

Format:

```
content: <'value'>;
content: <'value'>; content: <'value'>;
```

Sample rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 \
(msg: "EXPLOIT BIND tsig Overflow Attempt"; \
content: "|00 FA 00 FF|"; content: "/bin/sh";);
```

Sample output:

```
02/22-15:33:19.472381 ATTACKER:1024 -> VICTIM:53
UCP TTL:64 TOS:0x0 ID:6755 IpLen:20 DgmLen:538
Len: 518

<lines omitted to condense output>

00 3F 00 E8 72 FF FF FF 2F 62 69 6E 2F 73 68 00 .?..r.../bin/sh.
0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D ................
1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D .. !"#$%&'()*+,-
2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C EB ./0123456789:;<.
07 C0 00 00 00 00 00 3F 00 01 02 03 04 05 06 07 .......?........
08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 ................
18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 ........ !"#$%&'
28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 ()*+,-./01234567
38 39 3A 3B 3C EB 07 C0 00 00 00 00 3F 00 01 89 89:;<.......?...
02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 ................
D8 FA FF BF D8 F7 FF BF D0 7C 0D 08 04 F7 10 40 .........|.....@
22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 "#$%&'()*+,-./01
32 33 34 35 36 37 38 39 3A 3B 3C EB 07 C0 00 00 23456789:;<.....
00 00 00 3F 00 01 02 03 04 05 06 07 08 09 0A 0B ...?............
0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B ................
1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B .... !"#$%&'()*+
2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B ,-./0123456789:;
3C EB 07 C0 00 00 00 00 00 00 00 FA 00 FF <..............
```

FIGURE 4. Option with multiple *contents* [2].

Also, the multiple 'content options' can be found, as illustrated in the Figure 4. The values associated with this field can appear in any order within the payload. There is another case where there are multiple string content and if any of them match to a search pattern, a rule is dispatched.

The output presented on the left side of the Figure 4 shows the hexadecimal characters contained in the observed payload, followed by the ASCII interpretation on the right side. The created rule searches the UDP traffic outside the protected network with destination to host port 53. Specifically, the existence of two strings is searched: first, the string represented for a hexadecimal format, '00 FAN 00 FF', and the second, the text '/bin/sh'. Both strings can appear in the payload in any order.

3. Neural Network for Data Classification

Neural networks are composed of several computational non-linear elements (neurons) operating in a parallel and massively distributed way, connected through synaptic weights, which suffer adaptation to improve the network performance [6]. Because of the massive parallelism process, the neurons are able to provide high processing rates. Another important characteristic is the high fault tolerance, due to the great amount of neurons and connections [9].

Some studies related to the use of neural networks to classify unknown data, in order to indicate abnormal patterns (possible intrusion events) in a large amount of data from the network, have been conducted [5,6]. The neural network classifier has to determine which of *M* classes better represents a certain unknown input pattern. This work focuses the Hamming Net for fast classification.

The Hamming Net was chosen because it does not require exhaustive training and it may easily learn novel non-predicted patterns without requiring retraining over the whole training set. In addition, it requires less connections (linear connection growth related to the input data number), when compared with other neural network classifiers, such as the Hopfield network, which has a quadratic growth. Moreover, a consideration related to the use of the Hamming Net on Intrusion Detection is that the matching that this network always provides, between packet data and signature, even being an approximated similarity, allows to obtain possible attack attempts or new attack information. However, a study to verify the amount of false positive events returned must be conducted.

A. Hamming Net

The Hamming Net works in cooperation with a MAXNET to identify the class that contains a given input pattern. The pattern is identified through a stored prototype-pattern set (exemplar data set or classes). The input data pattern is associated to the most similar class, in other words, the class that has the smallest Hamming distance. The Hamming distance [3] is the number of bits within an input pattern that does not match the corresponding bits in an exemplar pattern.

Figure 5 presents the Hamming Net basic structure including the MAXNET network.

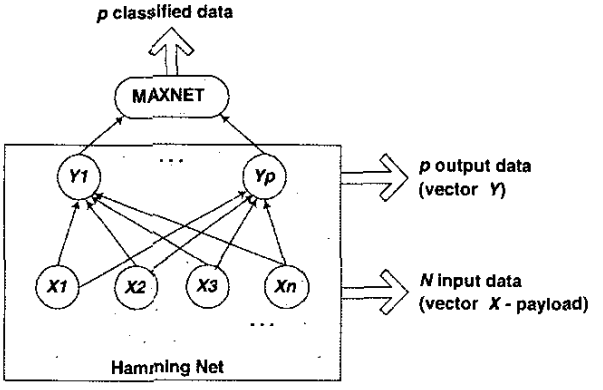

FIGURE 5 *Hamming Net* and *MAXNET* basic structure.

If the network is supposed to identify input patterns based on p different classes, both the Hamming Net and the MAXNET need to have p outputs. The Hamming Net outputs the distance between the input vector and each store pattern [7]. The Hamming Net output data are the input data for the MAXNET that modifies the input values simply by setting to zero all the other values, except the only one that corresponds to the target pattern class.

4. Developed application

This application was developed in four phases: Hamming Net implementation, treatment of the exemplar data, treatment of the input data, and data classification in Hamming Net.

In the first phase, the *Hamming Net* and *MAXNET* neural networks were implemented based on the algorithm placed in [3].

The second stage (started at point 1 Figure 6) consisted of creating the exemplars file through the "contents" (signatures) extracted from each line of Snort rule file, converting the string data to bipolar format, and storing all signatures size in the "exemplars" file.

In this stage, a study about the TCP/IP packet (data format, and retrieving interesting data) was conducted. Through the samples of the *Snort* patterns and *tcpdump* information, it was certified that the TCP payload data is the most important TCP/IP packet data. In this part, attack information may be found. Another information is that exemplar data may be obtained from Snort rule files.

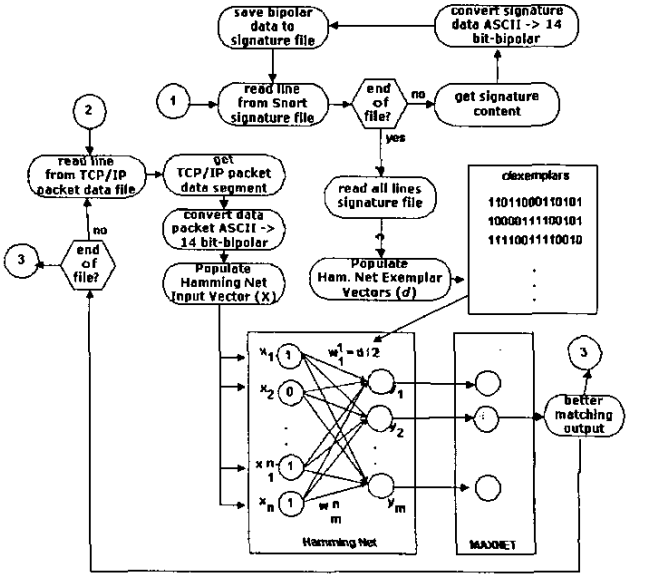

FIGURE 6 System Basic Processing.

At first, the intention was to work with data in real-time, but due to the difficulties found it was decided to use data obtained off-line to test the possibility and adequacy of using the Hamming Net as a classifier of anomalous data, as a preliminary study.

The payload file is manually built through illegitimate data from exemplar file and a normal data set. In each payload, the line data string is scanned in parts to get the TCP/IP packet payload data segment, which is the input for the Hamming Net. This loop continues through all the lines of the payload file.

The third step starts at point 2 in Figure 6. The data string from the payload file line is scanned by a mask of variable size according to the signatures size in the exemplars file. This mask slides one character per time. Then, each payload part in which the mask passes is converted to bipolar format and introduced in the Hamming Net for processing.

When input data is inserted in the Hamming Net, the weights have been configured using the exemplar data. For an input vector **X** and the outputs **Y** are computed. The Hamming Net provides the comparison between input data (TCP/IP packet payload part) and exemplars data (Snort signatures), making use of fixed weights. The MAXNET network receives the Hamming Net outputs and outputs the largest value, gradually eliminating those outputs that are smaller than the maximum value. The process is carried until only one non-zero output remains, when the data classification process is finished. The end of this processing is indicated by point 3 in Figure 6.

For each input pattern the network finds the most similar class. Even so, this similarity can be despicable by the difference among corresponding bits of the input and exemplar data. To avoid this situation, in this work a threshold was established to classify an input as "malicious information" only if it perfectly matches (same value for all bits) with one of the exemplars.

A. Considerations

The input data (originally a data string) were pre-processed and converted to bipolar data to provide a unique, known and small input data for the Hamming Net.

In order to achieve unique size for the bipolar data, independent of the data string size observed, a fold-shift hashing method was used [11]. For this work, a 72-character array to store the signature data string (each line of the exemplar file) was sufficient. The data string array was divided into 12 blocks with 6 characters and each block was processed using a hashing table. So, it was possible to create a unique-key with decimal 4-digits, which was then converted to 14-bits bipolar data.

To facilitate the application test, several input data (payload data files) were used. They were emulated through normal and abnormal data of the Snort rule files. Each line of a payload data file contains illegitimate or normal information. Payload data, in a certain interval of time, were also obtained by using the filter named *oknbashsniffer*. Each line part of the payload file was converted from string to bipolar format through the implemented conversion procedure.

B. Network Architecture

This application required a Hamming Net architecture composed of 14 inputs (vector **X**), N classes (vectors $\boldsymbol{d}$), named exemplar data or signatures, where N corresponds to the total number of signatures (intrusion pattern) considered and N output neurons (vector **Y**) corresponding to the Hamming Net output, as presented in Figure 6.

For a more sophisticated application, the number of the exemplar neurons should be approximately 3000, considering the whole Snort signature set. However, as more signatures are used, the application may become slower. Thus, it is fundamental to limit the signature number according to the characteristics and requirements of the observed network environment.

All the weight values for a neuron j of the Hamming Net are calculated by $w_{ji}=d_i/2$, where d_i corresponds to the i^{th} element of the exemplar vector j. Notice that the learning of the Hamming Net consists of the direct insertion of the knowledge (the Snort rules) in the network weights. The bias value is set to number of input data divided by 2. The weights and bias are constant for all neurons.

The Hamming Net neural network produces a set of m outputs (vector **Y**) that contains the m distances measurements computed between the input vector and the exemplars stored in the neural network.

The output of the Hamming Net is first complemented and then presented as input to the MAXNET, which will strengthen the largest value and will eliminate the others. In other words, only one neuron in MAXNET will be the winner corresponding to the exemplar index that matches the input.

C. Test Results

Through this application, three main experiments were analyzed with different payload and exemplar data files and the results are presented in the Table 1.

TABLE 1. Application Results

Data x Experiments	File: Payloads.txt	File: Exemplars.txt (extracted from 48 Snort files)	% matching
Experiment 1	TOTLIN 16 TOTANORM 10	NUMEXEMP 15 MAXLEN 72 TOTMASK 12	77%
Experiment 2	TOTLIN 16 TOTANORM 10	NUMEXEMP 448 MAXLEN 40 TOTMASK 36	70%
Experiment 3	TOTLIN 32 TOTANORM 28	NUMEXEMP 856 MAXLEN 43 TOTMASK 42	65%

The following legend is used in Table 1 to describe the information:

- TOTLIN – total of line in the payload file
- TOTANORM – total of line with abnormal data in the payload file
- NUMEXEMP – number of lines (signatures) in the exemplars file
- MAXLEN – length of the largest exemplar (in characters)
- TOTMASK – total of mask (equivalent to different exemplar size)

Three amounts of exemplar data were used for the tests: 15, 448 e 856, using signatures of 48 Snort rule files.

The matching percentage found indicates that, on average, 70% of well-known malicious input data within the payload file encounter its common pair in exemplar file and it is classified by the application as suspicious information. Besides, it was verified that if the number of exemplars is large, the network provides small convergence. Other parameters can affect the neural network results, such as the mask length; an issue that needs to be further researched in future works.

5. Conclusions

This paper describes a neural network approach to develop tool for computer network traffic monitoring in searching malicious and illegitimate information, based on available Snort attack information used to train a Hamming Net neural network.

The implementation results show the the developed application is a feasible technique to identify abnormal patterns in off-line mode.

A data reduction technique by using a unique 4-digit word, has favored the creation of 14-bit bipolar data (inputs and exemplars). This constant size for the input data set was well satisfactory for the network.

The Hamming net matches input pattern (TCP/IP packet payload segment) with exemplar classes (malicious information). In this work, a threshold value was used to achieve a perfect matching (input data match 100% to exemplar data). Choosing an appropriate threshold for the similarity measurement to produce satisfactory results and improve the neural network efficiency without loss of important data is a challenge.

On average 70% of the input data containing illegitimate information was satisfactorily classified. However, the application code must be better tuned to achieve better data classification rates, for example, considering multiple contents in the network packet. One observable advantage was the fact the application identified abnormal network events quickly.

Some difficulties were faced implementing the Hamming Net based application described. Such difficulties were related to appropriately model the data to be processed on the Hamming Net architecture; to correctly treat the input data; to obtain and interpret the payload data in TCP/IP packets; and measure the efficiency of the neural network.

In these regards, future work shall be conducted attempting at expanding the number of signatures in the exemplar set; refining and improving the application code; working with different mask sizes to scan the packet payload; conducting tests with real data from computer network packets; improving the application to work in real time; and comparing the efficiency of the model with other intrusion detection models.

References

[1] NORTHCUTT, S. *Desvendando Segurança em Redes*. Rio de Janeiro: Editora Campus, 2002. ISBN 85-3562-1123-3.

[2] NORTHCUTT, S. *Network Intrusion Detection*. 3.ed. New York: New Riders Publishing, 2002. ISBN 0-73571-265-4.

[3] FAUSSET, L. *Fundamentals of Neural Networks: architectures, algorithms, and applications*. New York: Prentice Hall, 1994. ISBN 0-13-334186-0.

[4] SIMON, H. *Redes Neurais – Princípios e Prática*. 2.ed. Porto Alegre: Bookman, 2001. ISBN: 85-7307-718-2.

[5] HUNG, C.; LIN, S. Adaptative Hamming Net: A Fast-Learning ART1 Model without Searching. *Neural Networks*, v.8, n. 4, p. 605-618, 1995.

[6] LIPPMANN, R. P. An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, v. 4, p.4-22, abr. 1987.

[7] Downs, T. COMP3700 Machine Learning – 3E381 Neural Computing Lecture 8 Unsupervised Learning. Austrália, July, 2002. Available in: *http://www.itee.uq.edu.au/~comp3700/lectures/lecture8.pdf* Accessed in: 24 nov. 2003.

[8] HOFMEYR, S.A.; FORREST, S.; SOMAYAJI, A. Intrusion detection using sequences of system calls. *Journal of Computer Security*, v.6, p.151-180, 1998.

[9] RUSSEL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. New York: Prentice Hall. 1995. ISBN: 0- 13-103805-2.

[10] Index of /claremont/keller, 11 dec. 1997. Available in:<*http://www.cs.hmc.edu/claremont/keller/152-slides/272.html*>. Accessed in: 12 nov. 2003.

[11] MORAES, C. R. Estrutura de Dados e Algoritmos: Uma abordagem Didática. São Paulo: Editora Berkeley, 2001. ISBN 85-7251-585-2.