INL/CON-09-16248
PREPRINT

Computationally Efficient Neural Network Intrusion Security Awareness

2nd International Symposium on Resilient Control Systems 2009

Todd Vollmer
Milos Manic

August 2009

Computationally Efficient Neural Network Intrusion Security Awareness

Todd Vollmer
Idaho National Laboratory
Idaho Falls, Idaho, USA
denis.vollmer@inl.gov

Milos Manic
Department of Computer Science
University of Idaho at Idaho Falls
Idaho Falls, Idaho, USA
misko@ieee.org

***Abstract*—An enhanced version of an algorithm to provide anomaly based intrusion detection alerts for cyber security state awareness is detailed. A unique aspect is the training of an error back-propagation neural network with intrusion detection rule features to provide a recognition basis. Ethernet network packet details are subsequently provided to the trained network to produce a classification. This leverages rule knowledge sets to produce classifications for anomaly based systems. Several test cases executed on ICMP protocol revealed a 60% identification rate of true positives. This rate matched the previous work, but 70% less memory was used and the run time was reduced to less than 1 second from 37 seconds.**

Keywords - Site security monitoring, command and control systems, neural networks, backpropagation

I. INTRODUCTION

The work presented in this paper is based on a previously published paper [1] by the same authors. An entirely new improved solution to the same problem is detailed. This paper is composed of fresh material and describes the performance improvements of the Computationally efficient Neural Network Intrusion Security Awareness algorithm (CeNISA) as well as comparing results to the original.

An increased threat awareness of cyber attacks has been recognized by many entities around the world [2]. Computer based systems used within many critical infrastructures to monitor control systems are increasingly being connected to the Internet. This increases the potential threat status to these systems as they may be even more vulnerable than traditional Information Technology (IT) systems [3]. Attacks against these systems can endanger public safety and lead to large expenditures of capital. Therefore, exploration of solutions to provide security state awareness to control system operations personnel is of utmost concern. This paper presents a pattern recognition algorithm which maps novel attack vectors recognized by anomaly Intrusion Detection Systems (IDS) to known attack classifications.

There are numerous varieties of intrusion detection and prevention solutions. This paper is concerned with two network based intrusion systems: behavior and signature based. Both types are tasked with detecting unwanted, potentially harmful Ethernet traffic. However, one fundamental difference is in the implementation of the employed recognition system. These differences provide both strengths and weaknesses that can be exploited by attackers.

Rule based systems are analogous to virus protection software resident on personnel computers. Predefined rule sets capture characteristics of attack vectors. These sets perform well on known signatures but generally do not recognize novel attacks. In addition, minor variations in the signature of a known attack may not be noticed by the system. However, rules are developed and distributed by an active community and widely distributed. It is even possible to convert rules between different formats [4]. This provides a rich base of historical information that is readily available.

Anomaly IDS systems learn a model of normal behavior [5]. Certain key attributes are tracked and a representation of current activity is maintained. Deviations of this representation from the historical model are recognized and treated as suspicious activity [6]. These systems are capable of detecting new attacks or other abnormal systemic behavior that might be missed by rule based systems. Unfortunately, these new behaviors may be acceptable and simply were not present during the initial learning phase.

IDS rules have usually been designed for typical IT based systems. Control systems are more frequently incorporating these types of IT solutions and consequently inheriting their same weaknesses and strengths. This paper proposes a solution that takes advantage of the historical depth and breadth of rules developed for signature based systems combined with the recognition power of anomaly based systems.

The rest of the paper is organized as follows. Related work with intrusion systems is presented in section 2. Section 3 gives a detailed description of the problem associated with rule and anomaly based systems and provides background information on the neural network used to replace the original similarity algorithm. Section 4 introduces the CeNISA algorithm. Section 5 presents the achieved experimental results followed by the conclusion given in section 6.

II. RELATED WORK

IDS rules are used as a basis for network anomaly detection reporting in the HISA algorithm [1]. A simple similarity algorithm was developed mapping the network packet characteristics to an IDS rule fields. The detail of each anomaly packet was compared to all rule fields and a match was

Work supported by the U.S. Department of Energy under DOE Idaho Operations Office Contract DE-AC07-05ID14517, performed as part of the Instrumentation, Control, and Intelligent Systems Distinctive Signature (ICIS) of Idaho National Laboratory.

recorded as a Boolean value. After an exhaustive evaluation of all rules and packets, a summation of matches for each rule is computed. The category feature of the rule(s) with the largest match values were then presented as a classification.

Lee et al. describe data mining techniques to construct network behavior models that are both accurate and efficient doing real time processing [7]. These techniques were used to build network intrusion models. Patterns of normal network traffic were created and stored. As new network traffic was presented to the system, a pattern was created. A comparison of this pattern to the stored patterns determined if the traffic was considered abnormal. The concept of comparing a normal baseline to real time data is similar to CeNISA. However CeNISA uses signature rules to establish a normal baseline and a neural network for pattern comparison.

Three different event notification types are typically delivered by anomaly systems. A Boolean value indicating an anomaly or not is the simplest [8]. If labeled data is available and used as input to a supervised network, an output representing the classes can be produced [9]. This type of system is limited to the classifications present in the training data. Finally, the system can recognize an anomaly event and produce a new signature rule that recognizes it [10].

III. Problem Description

Signature and anomaly based systems both have strengths and weakness when used in isolation. One issue with anomaly based systems is obtaining labeled data that is not artificially generated [11]. As was pointed out earlier, signature based solutions can miss new signatures or variations on known attacks. The focus of this paper is on combining the capabilities of both to overcome these issues. In addition, a classification is produced that is suitable for cyber security awareness. A description of IP/ICMP packets and SNORT® rules are provided next. The Snort rules are used as training input. The use of rule features as training vectors is a unique aspect of the CeNISA algorithm. As a proof of concept, ICMP network attack packets are created as test cases.

A. IP/ICMP

The Internet Protocol (IP) layer is a connectionless datagram delivery service defined in RFC 791. Any reliability or ordering is performed by the transport layer, i.e. TCP. Routing of the datagram is the responsibility of the IP layer as each instance contains a source and destination address. IPv4 defines a data structure containing this information [12]. Other protocol versions are not considered in this paper. The header portion of the datagram has many different fields. Those that are relevant to this discussion are described next.

Enabling fragmentation and reassembly, the IP identification field is a 16 bit value set to different values for each datagram. A datagram may be broken into smaller datagram's (fragmented) if its size exceeds the Maximum Transmission Unit (MTU) of a path. This field is not always set if there is no fragmentation possible.

Time to live (TTL) is an 8 bit field set by the originator. Each time the datagram is forwarded by a router the value is decremented by 1. Given the field size the maximum value is 255. If this value reaches 0 then the packet is discarded.

Forty bytes of optional data is allowed to follow the fixed size twenty byte header. Ten different options are defined for use: no-operation(NOP), end-of-list (EOL), loose source and record route (LSRR), strict source and record route (SSRR), Timestamp, Record route, Basic security, Extended security, Stream Identifier, Router alert.

The Internet Control Message Protocol (ICMP) is defined in RFC 792. ICMP messages are encapsulated within an IP datagram. The first four bytes of these messages have the same format. These bytes are composed of 3 fields: 8-bit type, 8-bit code and 16-bit checksum. Composition of the remaining bytes depends upon the message type.

The type field has 15 different possible values. This field in conjunction with the code field defines the message type. ICMP messages belong to one of two categories that encompass all the type and code combinations: query and error. Some messages that belong to the error category are handled differently.

The sequence number and identifier field can appear in address mask request and reply messages. They are both 16-bit fields and can be set to any value that fits in the range. The intent is for the responder to provide these fields back to the sender to allow for synchronization of messages

B. Snort Rules

Snort is an open source intrusion detection system that performs real time packet analysis. A major component of Snort is a flexible rule engine. These rules describe the characteristics in network traffic that are to be examined. Snort.org maintains two rule sets available for public use: community and proprietary. The community rules are contributed by users and are freely available. Proprietary rules are developed by Sourcefire and require registration and acceptance of terms and conditions. Other sources of rules on the Internet include those by Emerging Threats [13].

The Snort rule specification defines two primary rule components: a rule header and rule option. The header is required and defines the rule action, protocol, IP addresses and port information. There are many rule options that provide power and flexibility for detection of payload and non-payload data. For example the content keyword provides a mechanism to match specific byte values in the data portion of a packet [14].

One of the rule option keywords is classtype. This keyword is used to mark a rule defining a specific attack type as belonging to a more generic attack class. The definitions of these attack classes reside in the classification.config file distributed as part of the Snort package and are replicated in Table 1. Including these as part of the Snort package provides a common basis for rule development and helps ensure consistency. All of the rules provided by snort.org and emerging threats were marked with a classification. These classes are the foundation for reporting state awareness by CeNISA.

TABLE I. SNORT CLASSIFICATIONS

Classtype	Classtype
attempted-admin	rpc-portmap-decode
attempted-user	successful-dos
kickass-porn	successful-recon-largescale
policy-violation	successful-recon-limited
shellcode-detect	suspicious-filename-detect
successful-admin	suspicious-login
successful-user	system-call-detect
trojan-activity	unusual-client-connection
unsuccessful-user	web-application-activity
web-application-attack	icmp-event
attempted-DOS	misc-activity
attempted-recon	network-scan
bad-unknown	not-suspicious
default-login-attempt	protocol-command-decode
denial-of-service	string-detect
misc-attack	unknown
non-standard-protocol	tcp-connection

C. EBP Network

A multi-layer feed forward Error Back Propagation network (EBP) provides the primary performance improvement for CeNISA. EBP networks are a well researched supervised learning method introduced by Werbos in 1974. The power of a multilayer neural network lies in its ability to model multidimensional nonlinear problems. The learning vectors are presented as input and the calculations from each layer are fed forward to the next layer in the network. Results from the final layer are calculated and compared to the desired output producing an error measurement. This error information is propagated back from the output layer to the inner layers.

The input vectors or features may need to be adjusted. Preprocessing of input data is one of the most important steps in development of a neural network solution [15]. The data set may be missing values or valuable information can be obscured by an excessive number of attributes. In addition, the numerical values of the data are normalized to help equate the strength of the variables.Nominal data, such as colors, can be mapped to numerical equivalents and normalized as well. The data points that most affect the solution are optimal candidates for inputs while others are discarded. If too much information is removed, the resulting prediction ability will be affected.

The middle layers of nodes are added with a non-linear activation function. An activation function determines the output of a given node. The sigmoid function is typically used in the form of a hyperbolic tangent. This function is differentiable which is required to calculate back propagation.

The final layer's output is compared to the desired and an error is calculated. The gradient of this error function with respect to the weights is used to adjust the weights applied to each input of a node. Starting with the layer closest to the output nodes and working backwards. This process is repeated until overall performance of the network satisfies some user defined limit.

The general process for creating an EBP network is described as follows:

1. Define a feature vector, gather the training data and present it to the network as input.
2. Determine the number of hidden and output layers.
3. Using an input feature vector compare the network's output to the desired output. Calculate the error from each output node.
4. Incrementally adjust the weights of each node using the error calculations as a basis of calculation.
5. Using the error calculations for each node, feed the values back through the layers adjusting weights accordingly.
6. Repeat steps 3 – 6 until some acceptable error level is reached.

IV. CENISA ALGORITHM DESCRIPTION

The goal of CeNISA is to present information characterizing an unknown attack to a cyber security consumer. A unique approach utilizing historical rule definitions to find a close match is described. The algorithm consists of two critical phases: 1. Building the EBP network. 2. Extraction of data points from network data and subsequent presentation to the trained network. The Neural Network consists of three fully connected layers: input, hidden and output. Each major functional area is described in detail in the following sections.

A. Network Design

There are numerous options available for rule definitions but an analysis of existing ICMP rules showed that in practice only eleven are in use. Of these possible values, three were excluded byte_test, content and threshold. The first two specify specific comparisons of payload information that are inappropriate inputs for a neural network. The threshold option specifies Snort specific behavior on rule alerts and does not provide useful information. Table 2 specifies the nine features (eight plus the classification) chosen for inclusion in the training feature vector.

TABLE II. TRAINING FEATURES.

Feature	Description
dsize	packet payload size
icmp_id	ICMP ID field
icmp_seq	ICMP sequence
icode	ICMP code field
id	IP id field
ipopts	IP option field
itype	ICMP type field
ttl	IP time-to-live
class	Snort classification

Normalization of these input vectors is accomplished according to (1) where x_i is a feature instance and x represents the set of a feature type.

$$x' = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{1}$$

This normalized input is passed to the next layer in the network. The net input of node *i* in layer *k+1* is calculated as

$$n^{k+1}(i)=\sum_{j=1}^{Sk} w^{k+1}(i,j)a^{k}(j)+b^{k+1}(i). \qquad (2)$$

Here *Sk* denotes the number of nodes in layer *k*, $w^{k+1}(i,j)$ is the weight of the connection from neuron *j* in layer *k*, $b^{k+1}(i)$ is the bias of neuron *i* and $a^{k}(j)$ is the output from neuron *j* in layer *k*.

The output of node *i* in layer *k*+1 is

$$a^{k+1}(i)=f^{k+1}\left(n^{k+1}(i)\right). \qquad (3)$$

where f^{k+1} is the activation function of neuron *i*. In eHISA the hidden node layer cardinality matches that of the input layer which is nine. Fig. 1 shows the relationship of inputs and layers. It should be noted that the nodes are not fully connected in the figure to avoid cluttering the image.

The training set only contains 8 of the possible 38 classifications therefore the output layer consists of 8 nodes. Each output node represents a possible classification. An error term is calculated using the sum of squares function where o is the network node output and d is the desired output.

$$E=\tfrac{1}{2}\sum_{n=1}^{c}\left(o_n-d_n\right)^2 \qquad (4)$$

After the value for the output and error is computed, weight adjustments for the hidden and output layers are calculated.

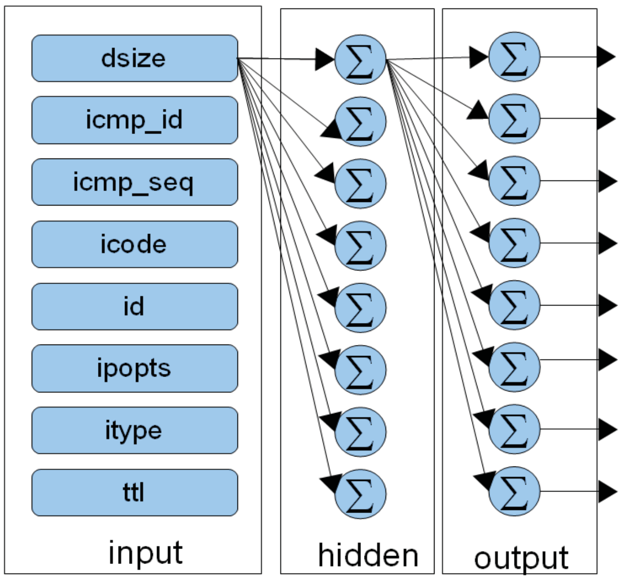

Figure 1. Network Diagram

Equation 5 provides the delta to apply to each weight; alpha is the learning rate *np* is the number of input patterns. An alpha of 0.3 was used for training.

$$\Delta w_p=\alpha\bullet\sum_{o=1}^{1}\sum_{p=1}^{np}\left[\left(d_{op}-o_{op}\right)F'\{z_p\}f'(net_p)x_p\right] \qquad (5)$$

After a complete set of updates for each training input pattern, the sequence of calculating outputs and feeding error information back through the network layers is reiterated 1000 times. These weights are saved and used for evaluation of extracted network features after the training period is complete.

In order to validate the recognition power of the described network a 10-fold cross-validation scheme is used. The input vectors are divided into 10 subsets of approximately equal size which was 13 for this instance. The network is trained 10 times, each time leaving out one of the subsets. The subset left out is subsequently used as the test data to judge the network error.

B. Data Evaluation

The network packet data is identified by an outside anomaly detection routine and is passed to the CeNISA algorithm for classification. The classification process consists of extracting the features found in Table 2 and presenting the values as input to the EBP network.

The processing of the packets is described next in a pseudo coding style.

```
Open Pcap file
Initialize feature structure
Loop
        strip Ethernet encapsulation
        decode IP content information
        decode ICMP content
        store values in feature list
        call EBP Network function
        store results
End loop
```

V. Results

A. Test Data

To show a proof of concept, the test data consisted of Snort ICMP rules and generated network packets. The rules sets used as input came from two sources, snort.org (VRT and community) and Emerging Threats [13] and combined for a total of 16,181 rules. Of these rules, 145 were ICMP specific and were categorized into 7 classifications.

The Nemesis network packet crafting and injection tool was used to create the ICMP test packets [16]. Nemesis is well suited for reproducing test scenarios as its output production is consistent and repeatable. It is capable of creating ARP, DNS, ETHERNET, ICMP, IGMP, IP, OSPF, RIP, TCP and UDP packets. A command line example creating an ICMP packet is shown next.

```
> nemesis –I 8 –s 14611 –c 123
```

Five ICMP network packets were created. The command line used for each packet was designed to create packets that belonged to different class types. The class types and command line specifics are shown in Table 3. These packets matched those created for testing in [1] and were stored in pcap format.

B. *EBP Results*

The original ICMP rule set of 145 was reduced to 129 instances. Duplicate and ambiguous rules (due to feature selection) were removed. Table 4 details the class types and associated counts of the rules. A fully trained network using these rules correctly identified 75% of the snort rule classifications. The confusion matrix shown in Table 5 provides the classification details. The class values have been abbreviated for space considerations. As can be seen from the matrix, a large number of rules are marked as miscellaneous. This may have caused overfitting of the solution to this classification.

The confusion matrix only shows the results from utilizing the Snort rules. The next step involved presenting the features from the five ICMP test network packets. The network correctly identified denial-of-service, misc-activity and attempted-recon for a 60% success rate. These classes represent the majority of the training vectors and should be the most recognizable to the network.

TABLE III. ICMP PACKET DETAILS.

Packet Details	Class Type
-i 0 -s 0 -e 667	attempted-dos
-i 8 -s 0 -e 666	attempted-recon
-i 5 -c 0	bad-unknown
-i 3 -c 2	attempted-user
-i 8 -s 14611 -c 123	misc-activity

TABLE IV. ICMP RULE CLASSIFICATIONS.

Class Type	Count (total 129)
denial-of-service	22
misc-activity	81
attempted-recon	11
trojan-activity	9
bad-unknown	4
attempted-user	1
network-scan	1

TABLE V. EBP CONFUSION MATRIX

	a	b	c	d	e	f	g
a:misc	74	2				1	4
b:trojan	5	4					
c:network	1						
d:bad	3	1					
e:a-user	0						1
f:recon	6					4	1
g:dos	7						15

C. *Original HISA Results*

The HISA algorithm used, as test data, the same 145 ICMP rules and generated ICMP network packets mentioned previously. After proving the correctness via a base case, the rule definitions that matched the generated packets were removed. Running the test packets through the system, without the matching rules, exercises the similarity algorithm's capability to recognize new threats. It also provided a truth set of classifications to calculate true matches. The results are shown in Table 6 [1]. The system considered a match of 80% or greater as a success. A 60% true positive match rate was achieved on the five test cases.

TABLE VI. ICMP RULE CLASSIFICATIONS.

Correct Class Type	Identified Class Type	% Match
attempted-dos	attempted-dos(3)	100%

attempted-recon	attempted-recon(4)
	network-scan(1)
		80%

bad-unknown	bad-unknown(2)
	attempted-recon(3)
	misc-activity (28)
		6%

trojan-activity	attempted-user(1)	0%
misc-activity	misc-activity(1)	100%

D. *CeNISA versus HISA*

In comparison, using the test cases as a basis, the modified algorithm using an EBP network provided the same accuracy (60%) as the original. However, a large improvement in efficiency with regards to run time and memory usage was realized as can be seen in Fig. 2. The HISA algorithm had a runtime of 37 seconds and utilized 100 MB's of RAM. The enhanced version took less than 1 second and required 30 MB's of RAM. This execution time does not take in to consideration the training phase which is a one time cost.

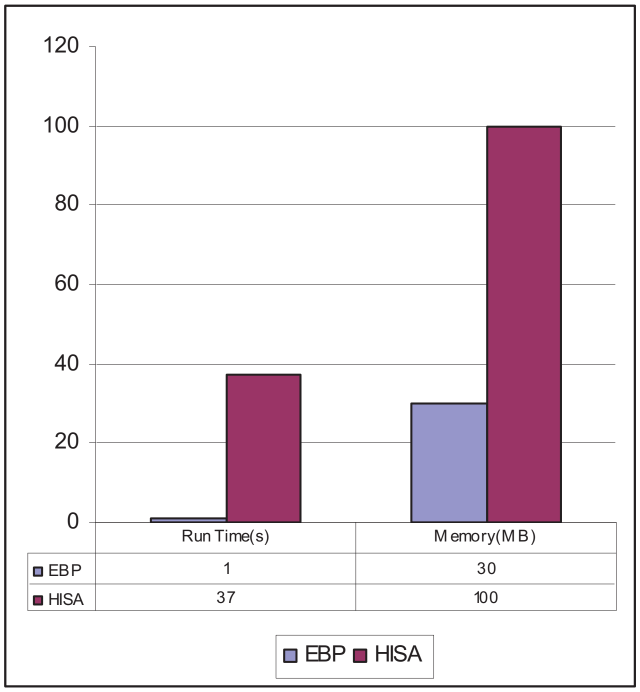

Figure 2. Performance

HISA stores the rule information as a data structure in memory and compares the network characteristics to values in the structure. The equivalent functionality in CeNISA is the EBP network. The rule attack knowledge is effectively stored within the weights between layers.

VI. CONCLUSION

An enhanced algorithm for presenting anomaly based intrusion detection alerts based on an error back-propagation neural network was presented. Rules developed for the Snort IDS and their default classifications were used as input vectors. The results of the network were compared with a previously derived similarity algorithm that solved the same problem and utilized the same input vectors. The similarity algorithm utilized a simple match summation process. The resulting output for both designs was subsequently presented to the user as an indicator of system cyber security status. An identification rate of 60% demonstrated the effectiveness of the proposed algorithm on test ICMP data. Although the test identification rates were identical, the EBP network based solution required 70% less main memory and executed 37 times faster.

The ICMP Snort rule set used as the basis for the identification scheme has a strong bias towards the miscellaneous category. As can be seen in the result section, the three categories that were identified correctly had the highest representation in the rules. Conversely, the false positives had the fewest rule counts. Addition of more rules to these under represented classes may improve the positive identification rate.

REFERENCES

[1] T. Vollmer, M. Manic, "Human Interface for Cyber Security Anomaly Detection Systems", HSI2009, 2nd Intl. Conf. on Human Systems Interactions, Catania Italy, May 28-30, 2009.

[2] B. Gellman, "Cyber-Attacks by Al Qaeda Feared." Washington Post. Online: http://www.washingtonpost.com/ac2/wp-dyn/A50765-2002Jun26/

[3] J. Meserve, "Sources: Staged cyber attack reveals vulnerability in power grid."CNN. Online: http://www.cnn.com/2007/US/09/26/ power.at.risk/.

[4] S. T. Eckmann, "Translating Snort rules to STATL scenarios", Proc. Recent Advances in Intrusion Detection, 2001

[5] A. K. Gosh, A. Schwartzbard, M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection", In Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, pp. 51-62, April 1999

[6] O. Linda, T. Vollmer, M. Manic, "Neural Network Based Intrusion Detection For Critical Infrastructures", IJCNN09, Int. Joint INNS-IEEE Conf. on Neural Networks, Atlanta, Georgia, June 14-19, 2009.

[7] W. Lee, S Stolfo, K Mok, "Mining in a Data-flow Environment: Experience in Network Intrusion", KDD-99, San Diego, CA USA, July 1999

[8] S. Zanero, S. Savaresi, 'Unsupervised learning techniques for an intrusion detection system', SAC 04 March 14-17, Nicosia, Cyprus

[9] L. Khan, M. Awad, B Thuraisingham, 'A new intrusion detection system using support vector machines and hierarchical clustering', The VLDB Journal, Vol. 16 pp. 507-521, 2007.

[10] Y. Huang, L. Wenke, 'A Cooperative Intrusion Detection System for Ad Hoc Networks', in Proc. 1st ACM workshop on security of ad hoc and sensor networks, Fairfax, Virginia, 2003 pp.135-147

[11] S. J. Stolfo, W. Lee, P. K. Chan, W. Fan, E. Eskin, SIGMOD Record, Vol. 30, No. 4, Dec. 2001

[12] W. R. Stevens, TCP/IP Illustrated, Volume 1, Addison Wesley Longman Publishing CO., Inc., Redwood City, CA, 2003

[13] http://www.emergingthreats.net

[14] M. Roesch. Writing Snort Rules: How To write Snort rules and keep your sanity. http://www.snort.org.

[15] C. M. Bishop, Neural Networks for Pattern Recognition, New York:Oxford Press,1995

[16] J. Nathan, Nemisis online: http://nemesis.sourceforge.net