USING VISUAL ANALYTICS FOR WEB INTRUSION DETECTION

Xydas[1], G. Miaoulis[1], P.-F. Bonnefoi[2], D. Plemenos[2], D. Ghazanfarpour[2]

[1] Technological Educational Institution of Athens, 28 Ag. Spiridonos Str., 12210 Athens, Greece
(yxydas@teiath.gr, gmiaoul@teiath.gr}
[2] University of Limoges, XLIM Laboratory, CNRS, UMR 6172
83, rue d'Isle, 87000 Limoges, France
{bonnefoi@unilim.fr, plemenos@unilim.fr, ghazanfarpour@unilim.fr}

Abstract

Web sites are likely to be regularly scanned and attacked by both automated and manual means. Intrusion Detection Systems (IDS) assist security analysts by automatically identifying potential attacks from network activity and produce alerts describing the details of these intrusions. However, IDS have problems, such as false positives, operational issues in high-speed environments and the difficulty of detecting unknown threats. Much of ID research has focused on improving the accuracy and operation of IDSs but surprisingly there has been very little research into supporting the security analysts' intrusion detection tasks. Lately, security analysts face an increasing workload as their networks expand and attacks become more frequent. In this paper we describe an ongoing surveillance prototype system which offers a visual aid to the web and security analyst by monitoring and exploring 3D graphs. The system offers a visual surveillance of the network activity on a web server for both normal and anomalous or malicious activity. Colours are used on the 3D graphics to indicate different categories of web attacks and the analyst has the ability to navigate into the web requests, of either normal or malicious traffic. Artificial Intelligence is combined with Visualization to detect and display unauthorized web traffic.

Keywords: *Web Visual Analytics, Web Attacks Visualization, Web Intrusion Detection, Evolutionary Artificial Neural Networks, Network Security, Surveillance Aid*.

1. Introduction

The work of an ID analyst is a complex task that requires experience and knowledge. Analysts must continually monitor IDSs for malicious activity. The number of alerts generated by most IDS can quickly become overwhelming and thus the analyst is overloaded with information which is difficult to monitor and analyze. Attacks are likely to generate multiple related alerts. Current IDS do not make it easy for operators to logically group related alerts. This forces the analyst to look only at aggregated summaries of alerts or to reduce the IDS signature set in order to reduce the number of alerts. In the current version of Snort [1], an open source IDS available to the general public, there are more than 12000 signatures for network intrusion detection, over 2100 of which are web-related signatures. By reducing the signature set the analyst knows that although it reduces the false alarms it is also likely to increase the number of false negatives, meaning that he will not be able to detect actual attacks.

Organizations, companies and individuals are making every effort to build and maintain secure Web sites. The threat profile facing enterprises and organizations has undeniably shifted from network-layer exploits to more advanced attacks against applications, primarily Web and Web services applications.

According to a recent report published by the Common Vulnerabilities and Exposures (CVE) project [2], flaws in Web software are among the most reported security issues so far this year. Hackers are known to search for an easy target. Poorly configured or poorly written web applications are not only an easy target, taking the attackers straight to their goal, giving them access to data and other information, but can also be used to spread malicious software such as viruses, worms, Trojan horses and spyware to anyone who visits the compromised site. "Easy to learn" scripting languages enable anyone with an eye for graphic design to develop and code powerful web-based applications. Unfortunately, many developers only bother to learn the eye-catching features of a language and not the security issues that need to be addressed. As a result, many of the same vulnerabilities that were problematic for developers several years ago remain a problem today. This is perhaps why Cross-Site Scripting (XSS) is now the most common type of application layer attack, while buffer overflow vulnerability, the perpetual No. 1 attack, has dropped to fourth place. Two other web application vulnerabilities, SQL injections and PHP remote file inclusions, are currently ranked second and third [3].

To detect web-based attacks, intrusion detection systems (IDS) are configured with a number of signatures that support the detection of known attacks. Unfortunately, it is hard to keep intrusion detection signature sets updated with respect to the large numbers of continuously discovered vulnerabilities. Developing ad hoc signatures to detect new web attacks is a time-intensive and error-prone activity that requires substantial security expertise. To overcome these issues, misuse detection systems should be complemented by anomaly detection systems, which support the detection of new attacks. Unfortunately, there are no available anomaly detection systems tailored to detect attacks against web servers and web-based applications.

According to a survey [4], in the intrusion detection area visualization tools are needed to offload the monitoring tasks, so that anomalies can be easily flagged for analysis and immediate response by the security analyst. Information presented in a visual format is learned and remembered better than information presented textually or verbally. The human brain is structured so that visual processing occurs rapidly and simultaneously. Given a complicated visual scene humans can immediately pick out important features in a matter of milliseconds. Humans are limited in terms of attention and memory but they excel at the processing of visual information.

The lack of intelligent visualization tools for web Intrusion Detection has led us to design and create a prototype system. It is a surveillance aid for the web and security analyst providing him with an intelligent visual tool to detect anomalies in web requests by exploring 3D graphs and understand quickly the kind of undergoing attack by means of colours. The system looks into web requests to detect "fingerprints" which are special characters or chains of characters. These fingerprints are then passed to an expert system to decide if they constitute a malicious request or attack. The output of the expert system is then transformed into a 3D graph for visual interpretation. Web attacks can be either rejected by the web server or can be successful due to security weaknesses.
The expert system used for the web attack classification is a hybrid expert system, an Evolutionary Artificial Neural Network (EANN). It is a supervised multilayer Artificial Neural Network (ANN) combined with genetic algorithms.
The rest of this paper is organized as follows: section 2 presents related work, section 3 presents the modules of the visualization prototype ID system in details and section 4 describes the system's performance evaluation. Finally, concluding remarks appear in section 5.

2. Related Work

There is ongoing research on IDS systems especially on anomaly detection and profile or specification-based detection. This includes various statistical methods, artificial neural networks and data mining methods ([5],[6],[7]).
Interesting works on the detection of web-based attacks have been published in the last few years. Statistical methods have been used in [8] such as the multi-model approach for the detection of web-based attacks. A Bayesian parameter estimation technique for web session anomaly detection is described in [9] and DFA (Deterministic Finite Automata) induction has been applied in [10] to detect malicious web requests in combination with rules for reducing variability among requests and heuristics for filtering and grouping anomalies.
Recent works on application-level web security cover SQL and PHP code injections and XSS attacks. The authors in [11] combine a static analysis and runtime monitoring to detect and stop illegal SQL queries. In [12] a sound, automated static analysis algorithm is developed for the detection of injection vulnerabilities, modelling string values as context free grammars and string operations as language transducers. In [13] Noxes, a client-side solution is presented, which acts as a web proxy and uses both manual and automatically generated rules to mitigate possible cross-site scripting attempts. Additionally, in [14] Secubat, a web vulnerability scanner is described, which is a generic and modular web vulnerability scanner that, similar to a port scanner, automatically analyzes web sites with the aim of finding exploitable SQL injection and XSS vulnerabilities. Visual analytics have recently been applied in network monitoring [15], detection and analysis of protocol BGP anomalies [16] and Intrusion Detection [17].
Artificial Intelligence used for web intrusion detection is limited to Bayesian classifiers. In [18] an IDS system is presented based on a Bayesian classifier in the same vein as the now popular spam filtering software. This simple classifier operates as follows: First the input is divided into some form of unit which lends itself to being classified as either benign or malicious, this unit of division is denoted as a *message*. It is the responsibility of the user to mark a sufficient number of messages as malicious/benign beforehand to effect the learning of the system. The system is thus one of direct self learning. The message is then further subdivided into tokens. The tokens are scored, so that the score indicates the probability of the token being present in a malicious message, i.e. the higher the relative

frequency of the tokens occurrence in malicious messages, relative to its occurrence in benign messages, the more indicative the token is of the message being malicious. The entire message is then scored according to the weighted probability that it is malicious/benign, given the scores of the tokens that it consists of. A 2D tool named Bayesvis was implemented to apply the principle of interactivity and visualization to Bayesian intrusion detection. The tool reads messages as text strings and splits them up into the substrings that make the tokens. The major limitations of this system are the following: a) the training phase of the classifier is time-consuming as sufficient statistics for every type of web attack are needed for the efficient work of a Bayesian classifier. The training is also a laborious task as the operator has to perform manually the correction of false alarms. He/she starts by marking a few of the benign accesses and then he re-scores, re-sorts and repeats the process according to a predefined strategy, until the false positive rate arrives at an acceptable level, b) attacks against the web applications are not detected, such as backdoor intrusions and code injection attempts by high level applications such as SQL, Perl, Php, HTML and Java c) new attacks cannot be detected due to the absence of previous statistics d) only web logs, not real time web traffic, are processed.

Our work focused on creating an ongoing surveillance tool offering the security analyst a novel visual tool for monitoring and diagnostic needs. We would like to offer an online tool which is capable of dealing with real network traffic in addition to processing stored web logs. We used an unsupervised artificial neural network for grouping similar attacks into classes and an Evolutionary Artificial Neural Network for the web attack classification. In addition, we have expanded the signature method for ID to detect backdoor intrusions and code execution attempts by high level applications such as SQL, Perl, Php, HTML and Java. Attacks are classified automatically by the expert system, false alarms are very limited, new attacks not seen before are detected as well and simultaneous multiple attacks from different networks can be easily spotted on the screen from the IP source address labels and the colouring of the different attack classes. Additionally, the security analyst can examine in real time the malicious code of Perl, SQL or other high level language injections, Cross Site Scripting information and the code on new backdoor attempts such as worms and viruses.

In the first version of the prototype the classifier used was an Artificial Neural Network (ANN). In the final version of the prototype we used a hybrid expert system for the web attacks classification. We used an Evolutionary Artificial Neural Network (EANN), which is a multilayer Artificial Neural Network combined with Genetic Algorithms (GA) for weight optimization.
Finally, we must emphasize that the whole system is developed in Linux and all system modules are written in standard C language, offering speed and portability to any operating system and platform, even on small portable computers.

3. Materials and Methods

The visualization prototype system consists of the following modules: The data capture module, the pre-processor module, the knowledge base module, the graph generator module and the statistical analysis module. The data capture module selects data either online from the Internet traffic or offline from the web server logs. The pre-processor module examines the web requests to detect malicious traffic and its output is then forwarded to the knowledge base module to predict the type of unauthorized traffic. Then, both normal and malicious traffic are processed by the graph generator module for visualization. Additionally, all traffic is kept for statistical analysis. Fig. 1 shows the architecture of the visualization prototype system. Each module is described in detail below:

3.1 Data capture module

The two most popular web servers are Microsoft Internet Information Services (IIS) and the open source Apache web server. The IIS web server of the Library of the Technological Educational Institution of Athens was used in order to study the various types of attacks and to create the knowledge data base of the system. We captured real data with the *tcpdump* utility from June to the end of November 2005. Using only real data we could not have a complete set of various attacks, so we have completed the tests with web logs data of the last three years. Web logs covered all versions of the Microsoft IIS server, e.g V4 (Win NT 4.0), V5 (Win 2000), V6 and API (Win 2003). The size of real data was 95,298 web requests and the size of tested logs was 527,373, 620,033 and 23,577 events for the last three years respectively.

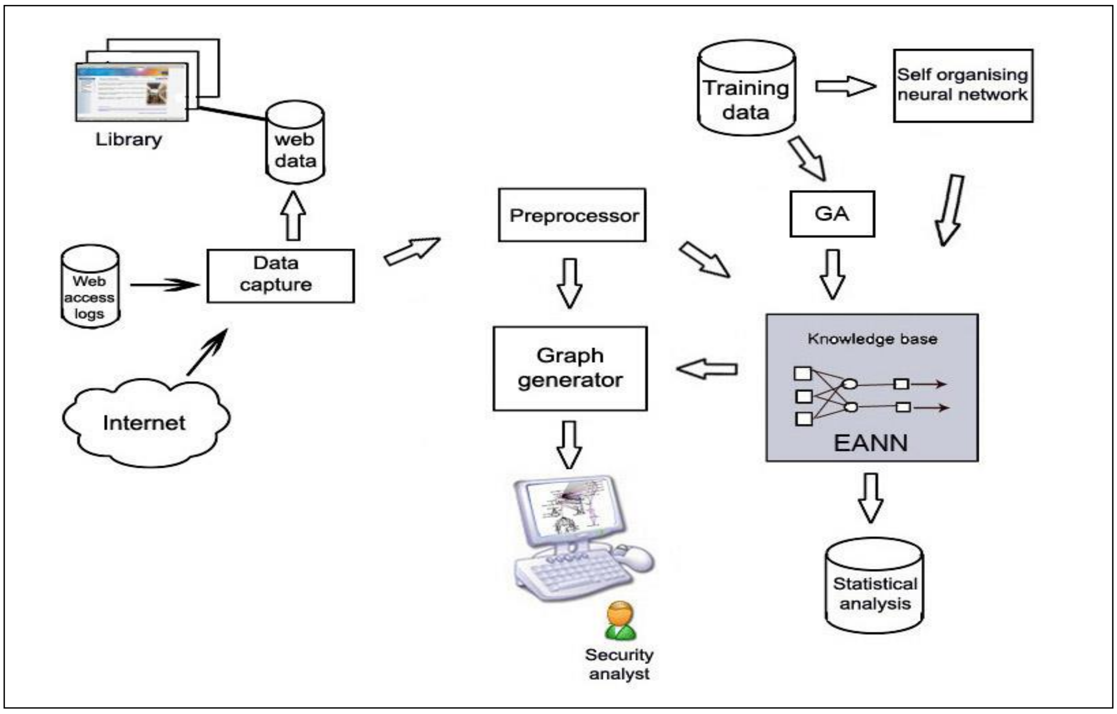

Figure 1. Visualization prototype ID system

3.2 Pre-processor module

The pre-processor analyses the web request and creates a feature vector of dimension 30. Fingerprints are detected checking their decimal or hexadecimal representation. The presence of a specific fingerprint in the web request is indicated in the feature vector as 1 (true) and its absence as 0 (false or unknown). An attack may have more that one "1" fired in its vector representation; furthermore an attack belonging to a specific attack class has at least one binary representation. The output of the pre-processor module is two files, one with the feature vector and one with the web request data. The feature vector will be the input to the expert system and the web request data will be forwarded to the graph generator module.

The extracted data is the most significant for online analysis such as the source IP address, the request option (GET, HEAD etc.) and the request payload. For example the pre-processor for the following malicious web request:

00:25:37 213.23.17.133 - HEAD /Rpc/..%5c..%5c..%5cwinnt/system32/cmd.exe /c+dir+c:\ 404 143 99 0 HTTP/1.0 - - - produces the following outputs:

1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (feature vector) and

213.23.17.133 HEAD /Rpc/..%5c..%5c..%5cwinnt/system32/cmd.exe /c+dir+c:\ (web request data).

3.3 Knowledge base module

3.3.1 Classes of Web attacks

Modern web servers offer optional features which improve convenience and functionality at the cost of increased security tasks. These optional features are taken in consideration in our design in addition to traditional types of web attacks (Unicode, directory traversal, buffer overflow, Server-Side Includes-SSI, Cross Site Scripting-XSS, mail and CGI attacks). Attacks against web applications such as code injections or insertion attempts are detected in the following programming languages HTML, Javascript, SQL, Perl, Access and Php. In addition IIS indexing vulnerabilities, IIS highlight, illegal postfixes, IIS file insertion (.stm), IIS proxy attempts and IIS data access vulnerabilities (msadc) are detected as well. All .asa, .asp and Java requests are tested for URI (Uniform Resource Identifier) legal syntax according to standards, meaning that a corresponding query not in the form <?key=value> is illegal. Trojan/backdoor upload requests are detected as well. These backdoors are left by worms such as Code Red, Sadmin/IIS and Nimda. Backdoor attempts for apache and IIS servers are detected when web requests ask for the corresponding password files (.sam and .htpasswd). Finally, command execution attempts are detected for both Windows (.exe, .bat, .sys, .com., .ini, .sh, .dll and other) and Unix (cat, tftp, wget, ls and other) environments.

A total of 30 fingerprints was used in the model to group all the different types of known web attacks

[19]. A detailed description of web attacks fingerprints is given in [20].
To classify the above web attack types a self-organizing neural network system has been used. The system was based on the famous Grossberg and Carperter's Adaptive Resonance Theory (ART1) [21]. ART1 algorithm is an unsupervised learning algorithm with biological motivations. Clustering algorithms are motivated by biology in that they offer the ability to learn through classification. Based on the Grossberg's *stability-plasticity dilemma* we cluster new concepts with older analogous ones and when we encounter new knowledge we create new clusters without destroying what has already been learned.
The ART1 neural network created 15 clusters or classes. These 15 classes were finally grouped manually into 9 as there was more that one class for command execution (Windows, Unix) and IIS type of attacks. It is interesting to notice that ART1 did not create a separate class for directory traversal and Unicode attacks because almost all of the web requests containing Unicode or traversal fingerprints (..\ or ../) always included another type of attack (e.g buffer overflow, command execution attempt, code insertions or other). So, directory traversal and Unicode attempts are not classified as separate attack classes. For historical reasons we included Unicode attempts into the Miscellaneous class.
The 9 final web attack classes are the following:

1) *Commands (CMD)*: Unix or Windows commands for code execution attempts.
2) *Insertions (INS)*: Application code injections (SQL, Perl, HTML, Javascript, Data Access).
3) *Trojan Backdoor Attempts (TBA)*: Attacks triggered by viruses and worms (Cod Red II, Sadmin, etc.).
4) *Mail (MAI)*: Mail attacks through port 80 (formail, sendmail etc.).
5) *Buffer overflows (BOV)*: Attacks corrupting the execution stack of a web application.
6) *Common Gateway Interface (CGI)*: Exploitation of vulnerable CGI programs.
7) *Internet Information Server (IIS)*: Attacks due to vulnerabilities of IIS.
8) *Cross Site Scripting (XSS)* attacks.
9) *Miscellaneous (MISC)*: Unicode, coldfusion and malicious web request options such as PROPFIND, CONNECT, OPTIONS, SEARCH, DEBUG, PUT and TRACE.

3.3.2 Training Data Quality
To measure the information which exists between the input data and the output data we had to calculate the *mutual information* between the two data sets. We want the network to take the input and remove all uncertainty about what the corresponding output should be. The amount of the original uncertainty we can remove depends on the mutual information present in the data. With an ideal training set, once we know the input value, there should be no doubt as to the correct output value: it should be the one value with a conditional probability, given the current input, of one; all other output values should have a probability of zero. As we cannot have an ideal training set, we need a measure of the average spread of conditional probabilities over the whole training set.
Let H denote the entropy of a set of events, X and Y the data sets of input and output respectively, H(X|Y) the conditional entropy of inputs given the outputs and I(X;Y) the mutual information between the input and the output data of the training set. We measured the H(X), H(Y) and I(X;Y) using a program to calculate the equations (1), (2) and (3) or (4):

$$H = \sum_{i=1}^{n} P_i * \log \frac{1}{P_i} \tag{1}$$

$$H(X\,|\,Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) * \log \frac{P(y_j)}{P(x_i, y_j)} \tag{2}$$

$$I(X;Y) = H(X) - H(X\,|\,Y) \tag{3}$$

$$I(X;Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) * \log \frac{P(x_i, y_j)}{P(x_i) * P(y_j)} \tag{4}$$

where n is the number of possible distinct input events, m the number of possible distinct output events and P_i is the probability of event i occurring out of the possible n events. Table 1 shows the results with the used training set.
As we can see: H(inputs) ≈ log(n) and H(outputs) ≈ log(m), so the used training set is a well balanced training set. The ratio I(input; output):H(output) ranges from 0 to 1 and would be high if a data set is

learnable. This ratio for our data set is equal to 0.805, which means that the data set used is learnable. However, it could be improved in the future.

Table 1. Data sets entropy and mutual information results

n	log(n)	m	log(m)	H(X)	H(Y)	H(X|Y)	H(Y|H)	I(X;Y)
49	3.891	9	2.197	3.512	2.160	1.777	0.420	1.740

3.3.3. Evolutionary neural expert system

If the pre-processor detects even one fingerprint its output is forwarded to an expert system for classification. In the first version of the prototype [22] we used an Artificial Neural Network (ANN) for classification. ANN's represent a class of very powerful, general-purpose tools that have been successfully applied to prediction, classification and clustering problems. The ANN used was a multilayer network with one hidden layer, using the *generalized delta rule with the backpropagation (BP) algorithm* for learning and the sigmoid function as activation function [23]. The input neurons were 30 (+1 the bias), the hidden neurons 10 (+1 the bias) and the output neurons 9, representing the 9 web attack classes.

In the final version of the prototype a hybrid expert system is used for the web attacks classification. We used an Evolutionary Artificial Neural Network (EANN), which is neural network combined with Genetic Algorithms (GA) for weight optimization. GA's are algorithms for optimization and learning, based loosely on several features of biological evolution. GA's do not face the drawbacks of the backpropagation (BP) algorithm, such as the scaling problem and the limitation of the fitness (error) function to be differentiable or even continuous. If the problem complexity increases, due to increased dimensionality and/or greater complexity of data, the performance of BP falls off rapidly. GA's do not have the same problem with scaling as backpropagation. One reason for this is that they generally improve the current best candidate monotonically, by keeping the current best individual as part of their population while they search for better candidates. Secondly, they are not bothered by local minima.

Let us consider the three-layer neural network of the prototype system. The parameters *n, m* and *l*, denoting the number of neurons of the three layers, are respectively 30, 10 and 9 for the prototype system. To find an optimal set of weights for the multilayer feedforward neural network we represented the problem domain as a chromosome. Initial weights are chosen randomly within some small interval [-0.5, 0.5]. The set of weights can be presented by a square matrix (Fig. 2) in which a real number corresponds to the weighted link from one neuron to another.

Each row of the matrix represents a group of all the incoming weighted links to a single neuron. This group can be thought of as a functional building block of the network [24] and therefore should be allowed to stay together passing genetic material from one generation to the next. To achieve this we associated each gene of the chromosome not with a single weight but with a group of weights, a row of the above matrix.

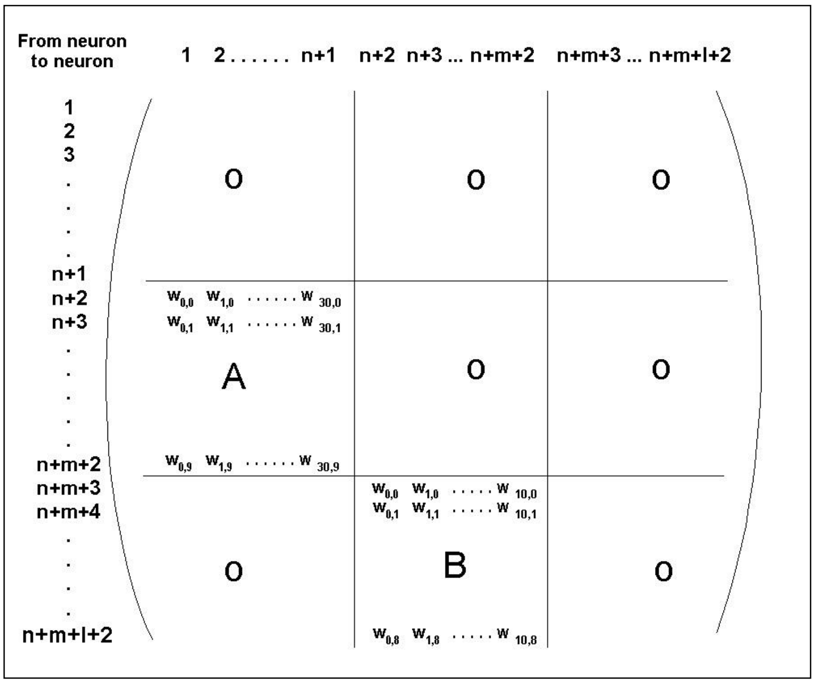
From neuron
to neuron
1 2 ...... n+1
n+2 n+3 ... n+m+2
n+m+3 ... n+m+l+2
1
2
3
n+1
n+2
n+3
n+m+2
n+m+3
n+m+4
n+m+l+2
O
A
B
$w_{0,0}$ $w_{1,0}$ ...... $w_{30,0}$
$w_{0,1}$ $w_{1,1}$ ...... $w_{30,1}$
$w_{0,9}$ $w_{1,9}$ ...... $w_{30,9}$
$w_{0,0}$ $w_{1,0}$ ..... $w_{10,0}$
$w_{0,1}$ $w_{1,1}$ ..... $w_{10,1}$
$w_{0,8}$ $w_{1,8}$ ..... $w_{10,8}$

Figure 2. ANN's weight connection matrix

The numbers of neurons are the same as in the original neural network. So, as in total there are 409 weighted links (31*10+11*9) between neurons, the chromosome has a dimension of 409 and a population member has been represented as follows:

M = < $w_{0,0}, w_{1,0} \dots w_{30,0},\ w_{0,1}, w_{1,1} \dots w_{30,1}\ \dots\ w_{0,9}, w_{1,9} \dots w_{30,9} \mid w_{0,0}, w_{1,0} \dots w_{10,0},\ w_{0,1}, w_{1,1} \dots w_{10,1}\ \dots\ w_{0,8}, w_{1,8} \dots w_{10,8}$ > ,

where the first part is the transposed matrix $W_{ih}[31,10]$ of weights between the input and the hidden layer (we string the rows together) and the second concatenated part is the transposed matrix $W_{ho}[11,9]$ of weights between the hidden layer and the output. Each member of the population was coded with the structure of the chromosome and a double real number for the fitness number.
The fitness function for evaluating the chromosome's performance was the sum of squared errors (SSE), used in the training phase of the BP algorithm. The smaller the sum, the fitter the chromosome. We used crossover and mutation as genetic operators. The crossover and mutation probabilities were 0.8 and 0.05 respectively. We started with a mutation probability of 0.02, but we finally used 0.05 as it accelerated the evolution of the GA.
The used algorithm of the EANN system can be described in a pseudo-code as following:

1) Randomly generate an initial population of chromosomes (population size 30) with weights in the range of [-0.5, 0.5].
2) Train the network for 1000 epochs using the BP algorithm. Calculate the fitness function for all individuals.
3) Select a pair of chromosomes for mating with a probability proportional to their fitness (roulette-wheel selection).
4) Create a pair of offspring chromosomes by applying the genetic operators crossover (multi-point crossover) and mutation.
5) Place the created offspring chromosomes in the new population.
6) Repeat step 4 until the size of the new population becomes equal to the size of the initial population and then replace the parent chromosome population with the new (offspring) population.
7) Go to step 2 and repeat the process until the algorithm converges or a specified number of generations has been reached (we used a maximum of 1000 generations).
8) Use the weights of the best member (ideal) of the last generation for the feedforward only operation of the ANN (classification).

For each generation we calculated the minimum (minFit) , the average (avgFit) and the maximum fitness (maxFit) of the population. In the belief that the algorithm should converge if the minimum fitness were less than an epsilon, equal to 10^{-12} and the ratio minFit/avgFit were greater that 0.95. In this way, by setting such a severe criterion all members of the final generation would become "ideal" and fit to be used for classification in the feedforward neural network, not just the best member of the population. The algorithm did indeed converge after 305 generations giving a minimum fitness of 6.61e-12 and 30 ideal members, a set of 30 best optimized weights for the operation of the ANN.

3.4 Graph generator module

The predicted attack by the EANN is then used to create a coloured directed graph in *dot* form of the well known GraphViz [25] package, using the corresponding *DOT* language. This language describes four kinds of objects: graphs, nodes, edges and labels and has a large number of attributes that affect the graph drawing. The payload of a web request is cut in nodes and the directed edges are the links between these nodes from left to right. Therefore, a web request from an IP source 212.205.253.160 with paylod GET /Hact/Graphics/Springer.gif, has as nodes the words "212.205.253.160", "GET", "Hact", "Graphics", "Springer.gif" and as "directed edges" the links between these nodes from left to right:

212.205.253.160 → GET → Hact → Graphics → Springer.gif

When each web request with its IP source address and the requested data is visualized in a 3D graph the security analyst can navigate into the graph for quick interpretation and evaluation in case of a malicious attempt. Timestamps were not added to the graph as graphs are displayed in real time and the objective here was to keep the display as simple as possible.
There are two graphs generated with the GraphViz package. One graph contains real time traffic, e.g. both normal and possible malicious traffic and the other does not contain normal but only the possible malicious traffic. Normal traffic is visualized in black and malicious traffic in 9 different colours, one for each attack class, such as red (Commands), brown (Insertions), magenta (Backdoor attempts), green (Mail), cyan (Buffer overflows), gold (CGI), blue (IIS), yellow (XSS) and coral (Miscellaneous). This

visual separation was necessary because normal traffic overloads the display and the security analyst cannot interpret quickly the malicious attempts. When visualizing both normal and malicious traffic the security analyst spends more time navigating through the graph trying to eliminate normal traffic by zooming into the coloured part of the display, than he would if he had only a coloured graph to contend with.
The malicious traffic colour display in both graphs is the result of the visual analytics process whereby web traffic is first analysed (i.e., the most important data is chosen, Intrusion Detection analysis is done using Artificial Intelligence) and then displayed. By employing intelligent means in the analysis process the visual representation of the traffic allows the security analyst to gain insight into the intrusion problem quickly and will be of invaluable help in the decision making process as he will be able to rapidly extract knowledge from the graph.
These two *dot* coloured graphs are then visualized with Tulip [26], a 3D graph visualization tool, supporting various graph algorithms and extensive features for interactive viewing and graph manipulation. Fig. 5 shows normal and malicious web traffic and Fig. 6 only the malicious traffic for the same events. In Fig. 5 the brown graphs on the left indicate Perl injection attempts, the cyan graphs buffer overflows attempts, the red graphs indicate multiple command execution attempts from IPs 69.107.112.253, 203.163.130.94 and other sources and the magenta graphs indicate multiple backdoor attempts (Code Red II) from IP 203.163.130.94. In Fig. 6 we can spot additional backdoor attempts from IPs 213.23.17.133 and 129.115.207.6 and buffer overflow attacks from IP 195.130.70.133 (cyan graph). The Perl injection code can be easily read on the bottom right of the graph.

3.5 Statistical analysis module

The system performance was tested using real data, captured with the *tcpdump* utility in June and November 2005 and web logs of 2005 and 2006. In the statistical analysis module of the system a confusion matrix is calculated to display the classification results of a network. The confusion matrix is defined by labelling the desired classification in rows and the predicted classifications in columns. For each exemplar, a 1 is added to the cell entry defined by (desired classification, predicted classification). Since we want the predicted classification to be the same as the desired classification, the ideal situation is to have all the exemplars end on the diagonal cells of the matrix. Table 2 shows such a confusion matrix for test1 (web logs 2005).
In addition, for each test a 2x2 table is calculated containing, on the first row the Hits (attacks present or True Positives) and the False Alarms (or False Positives) and on the second row the Misses (attacks present but not detected or False Negatives) and the Correct Rejections (normal traffic or True Negatives). Results are presented in Table 3 in this form. All tests have been run for various values of a detection threshold to show how changing the detection threshold affects detections versus false alarms. If the threshold is set too high then the system will miss too many detections and conversely, if the threshold is set too low there will be too many false alarms. For the tests we have used threshold values rating from 0.3 to 1.0 with a step of 0.1. The best results using the BNN were obtained with a threshold value of 0.8 giving maximum detections of 95% and a minimum of false alarms. Using the EANN we obtained almost the same results for a threshold rating between 0.3 and 0.9, due to the stable performance (93.50%) of the hybrid expert system. Table 3 summarizes the results with various testing data sets.

Table 2. Confusion matrix for test1 (EANN with threshold 0.7)

	CMD	**INS**	**TBA**	**MAI**	**BOV**	**CGI**	**IIS**	**XSS**	**MIS**	**NRM**
CMD	**17469**	**241**	0	0	0	0	0	**9**	0	0
INS	0	**5**	0	0	0	0	0	0	0	0
TBA	0	**0**	**312**	0	0	0	0	0	0	0
MAI	0	0	0	**3**	0	0	0	0	0	0
BOV	0	0	0	0	**421**	0	0	0	0	0
CGI	0	0	0	0	0	**7**	0	0	0	0
IIS	0	0	0	0	0	0	**95**	0	0	0
XSS	0	**5**	0	0	0	0	0	**0**	0	0
MIS	0	0	0	0	0	0	0	0	**173**	0
NRM	0	0	0	0	0	0	0	0	0	**130780**

Hits:	**18485**	**False Alarms:**	**255**	
Missed:	**25**	**Normal traffic:**	**130780**	**Total events: 149545**

Table 3. Performance evaluation tests (EANN)

Test Data **Positives Negatives**	**Logs 2005** **149545 events**	**Logs 2005** **149456 events**	**Logs 2006** **149450 events**	**Logs 2006** **149503 events**	**Logs 2006** **149749 events**	**online data (Oct. 06) 49372 events**	**online data (Nov. 06) 22022 events**
TP FP **FN TN**	18485 255 25 130780	9136 12 582 139726	7575 2 56 141817	10176 0 62 139265	3631 0 63 146055	9 0 0 49363	34 22 13 21953

4. Results and Discussion

4.1 System Performance Evaluation - ROC curve

The performance of the prototype IDS system has been evaluated in [27]. In a two-class problem there are two possible types of error that may be made in the decision process. If class ω_1 is termed the positive class and class ω_2 the negative class, then ε_1 is referred to as the *false negative rate (or error of Type I)*, the proportion of positive samples incorrectly assigned to the negative class and ε_2 is the *false positive rate (or error of Type II)*, the proportion of negative samples classed as positive. The Neyman-Pearson decision rule is to minimize the error ε_1 subject to ε_2 being equal to a constant, a, say. Using different terminology, the Neyman-Pearson decision rule is to maximize the detection probability P_D (P_D=1-ε_1), while not allowing the false alarm probability (P_F) to exceed a certain value.
Using the Neyman-Pearson decision rule we calculated the detection probabilities (P_D) and therefore the missed probabilities ($1-P_D$) for different accepted false alarm rates (P_F). Table 4 displays these results and Fig. 3 displays the Receiving Operating Characteristic curve (ROC) of the prototype system based on the results of Table 4.
From the ROC curve one can verify visually, that with a P_F of around 15% a maximum detection (P_D) of about 92% is achieved (upper left point of the curve). This is the best trade-off between the false alarm rate and the detection rate of the developed prototype system.

Table 4. False alarm, Detection and Missed probabilities of the prototype system

$P_F = \alpha$	P_D	$1-P_D$
0.05	0.8053	0.1947
0.10	0.8829	0.1171
0.15	0.9247	0.0753
0.20	0.9337	0.0663
0.25	0.9426	0.0574
0.30	0.9515	0.0485
0.35	0.9584	0.4160
0.40	0.9599	0.0401
0.45	0.9614	0.0386
0.50	0.9629	0.0371
0.55	0.9644	0.0356
0.60	0.9659	0.0341
0.65	0.9662	0.0338
0.70	0.9666	0.0334
0.75	0.9666	0.0334
0.80	0.9668	0.0332
0.85	0.9670	0.0330
0.90	0.9672	0.0328
0.95	0.9674	0.0326

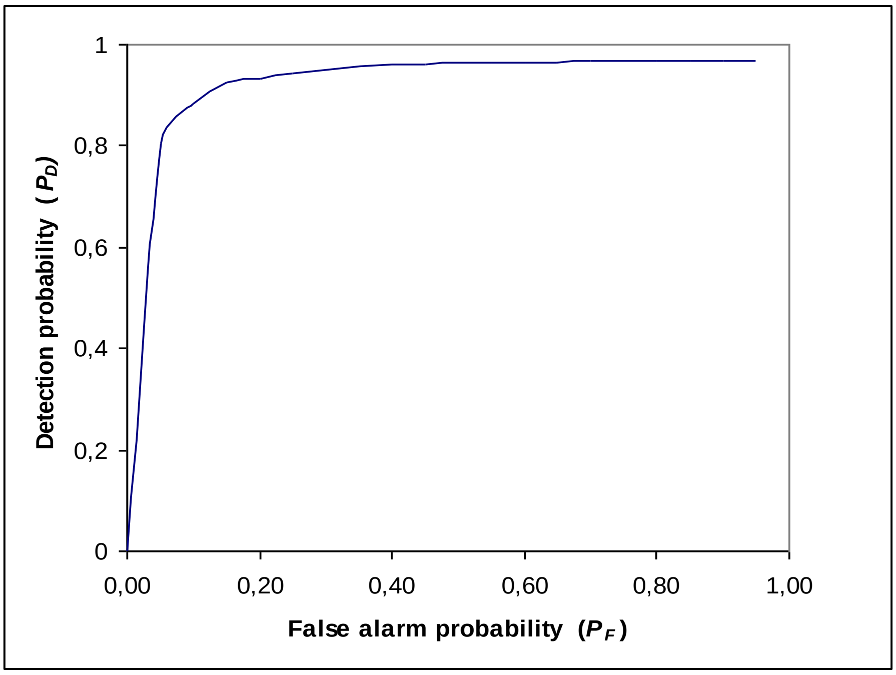

Figure 3. Receiving Operating Characteristic (ROC) curve of the prototype system

4.2 Comparison of EANN and ANN classifiers

Fig. 4 shows the performance of the EANN hybrid expert system versus the original Artificial Neural Network (ANN).

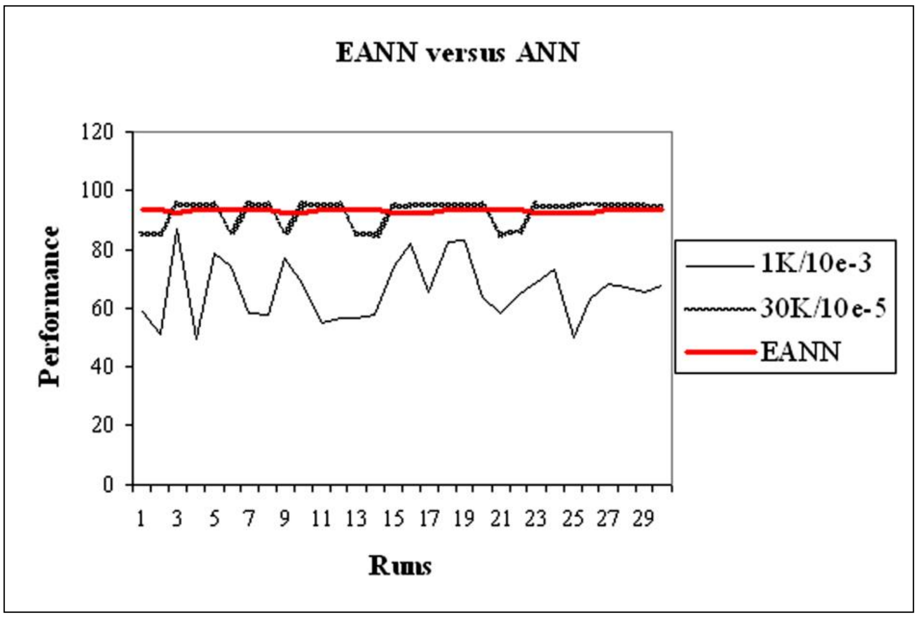

Figure 4. Comparison of EANN and ANN classifiers

The near straight line indicates the stable performance (93.50%) of the EANN. Initial training was done with only 1000 epochs and a SSE limit of 10^{-3}. The other two lines show the performance of the simple ANN using the BP algorithm. We can distinguish the stochastic behaviour of the ANN's performance. Using 1000 epochs and a SSE limit of 10^{-3} the ANN system performance rated between 50-87%, giving an average of 66.15% for 30 runs. Using 30,000 epochs and a SSE limit of 10^{-5} the ANN system performance rated between 85-94% giving an average of 92.52% for 30 runs. In the first version of the prototype system we used the latter combination, which had the drawback of a slow training cycle. Using the hybrid expert system with the GA approach for the weight optimization and test data different from the training set, a stable neural network performance of about 93.50% was achieved for all 30 runs (red near straight line in Fig. 4).

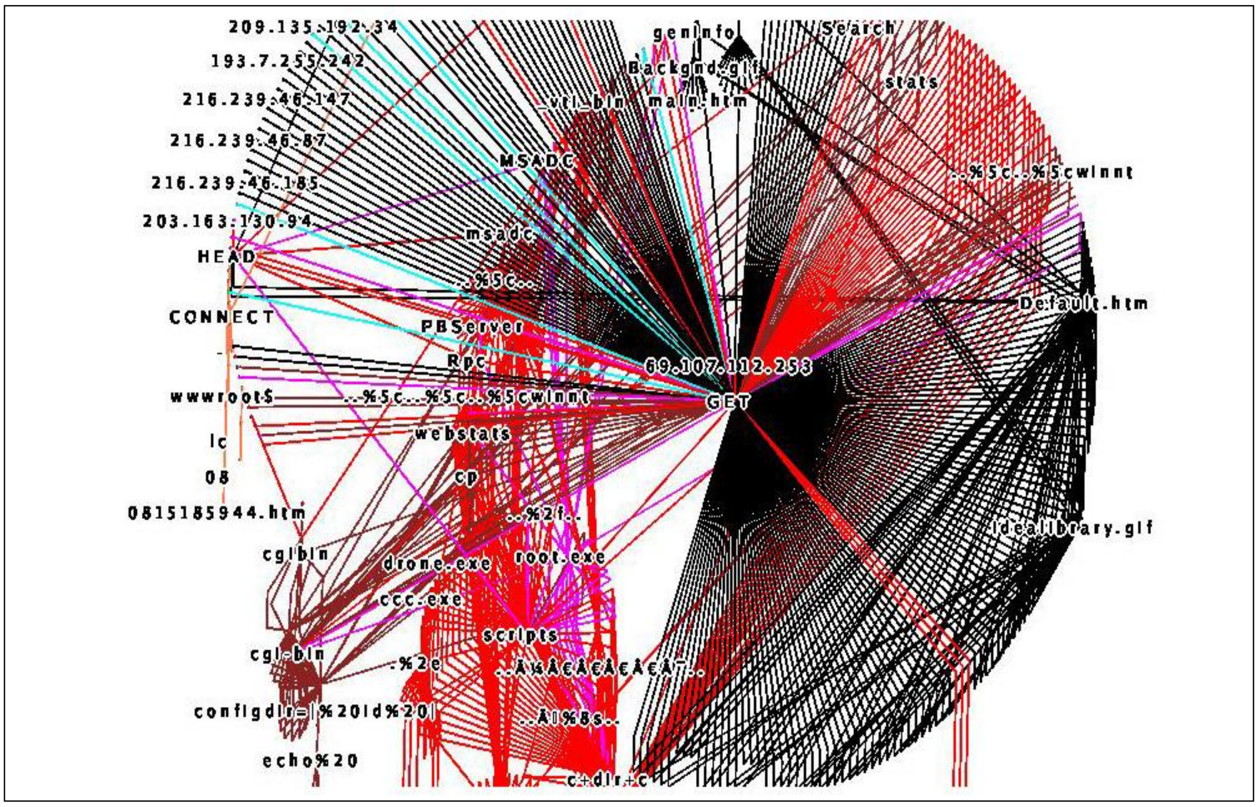

Figure 5. Normal and malicious web traffic

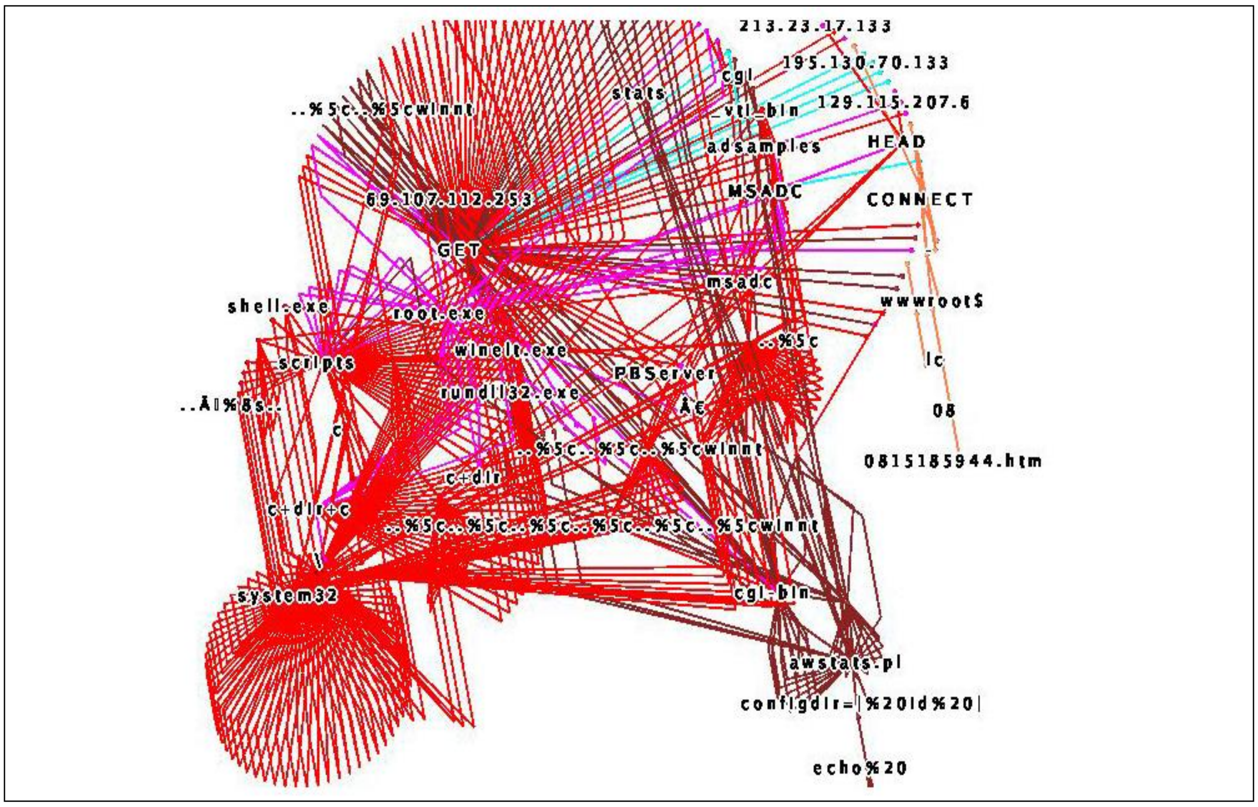

Figure 6. Malicious only web traffic

5. Conclusion

It is technologically impossible for any device to understand application communications or analyse application behaviour through deep inspection of IP packets, either individually or reassembled into their original sequence. Network firewalls and Intrusion Detection Systems (IDS) are useful for validating the format of application header information to ensure standards compliance. In addition, network-level security devices may detect a small number of known, easily identifiable attacks by looking for pre-programmed patterns (i.e. attack signatures) in an HTTP stream.

Unfortunately, without any awareness of the HTML data payload or session context, devices that rely exclusively on the inspection of IP packets will fail to detect the vast majority of application-layer exploits. For example, IP packet inspection will not detect a hacker who has maliciously modified parameters in a URL (Universal Resource Locator) request.

Network data analysis is a very important but a time consuming task for any administrator. A significant amount of time is devoted to sifting through text-only log files and messages generated by networks tools in order to secure networks. Artificial intelligence and visualization offer a powerful means of analysis that can help the security analyst uncover hacker trends or strategies that are likely to be missed with other non-visual methods.
With our work we have contributed the following to artificial intelligence and network security:

- Use of an Evolutionary Neural Network as a knowledge base for rapid classification of web attacks. The stable performance of the EANN establishes it as a better classifier for web intrusion than a simple neural network.
- The application of automatic analysis methods before the interactive visual representation offers an intelligent visualization of web traffic that enables rapid perception and detection of unauthorized traffic.
- A surveillance aid for the security analyst.
- A visualization prototype system ideal for educational purposes and in understanding web server and web application security.

This project has demonstrated that artificial intelligence considerably reduces the time required for data analysis and at the same time provides insights which might otherwise be missed during textual analysis. The web traffic surveillance could be expanded to other basic but popular internet services, such as email or DNS.
Combining traditional or novel analytical methods with visual presentation techniques can generate a very robust approach to network security. Artificial intelligence and visual analytics can be incorporated in ID systems to produce more powerful security systems capable of dealing with new attack challenges and noisy data. This is undoubtedly the future in the ID area.

References

[1] Snort software, http://www.snort.org, 2008.

[2] CVE, Common Vulnerabilities and Exposures, The Standard for Information Security Vulnerability Names, http://www.cve.mitre.org, 2008.

[3] M. Cobb, Software security flaws begin and end with web application security, http://searchsecurity. techtarget.com, 2008.

[4] A. Komlodi, J.R. Goodall, and W.G. Lutters, An Information Visualization Framework for Intrusion Detection. *CHI '04 extended abstracts on Human factors in computing systems*, ACM press, Vienna, Austria, 2004, p. 1743-1746.

[5] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, Specification-based Anomaly Detection: A new Approach for Detecting Network Intrusions, *Proceedings of the 9th ACM conference on computer and communications security*, ACM Press, Washington, DC, 2002, p. 265-274.

[6] W-H. Chen, S-H. Hsu, H-P. Shen, Application of SVM and ANN for intrusion detection, *Computers and Operations Research*, *32*(10), Elsevier, 2005, p. 2617-2634.

[7] W. Lee, S. Stolfo, K. Mok, Adaptive Intrusion Detection: A Data Mining Approach, *Artificial Intelligence Review*, *14*(6), Kluwer Academic Publishers, 2000, p. 533-567.

[8] C. Kruegel, G. Vigna, W. Robertson, A multi-model approach to the detection of web-based attacks, *Computer Networks*, *48*(5), Elsevier, 2005, p. 717-738.

[9] S. Cho, S. Cha, SAD: web session anomaly detection based on parameter estimation, *Computers & Security*, *23*(4), 2004, p. 312-319.

[10] K.L. Ingham, A. Somayaji, J. Burge, S. Forrest, Learning DFA representations of HTTP for protecting web applications, *Computer Networks*, *51*(5), 2007, p. 1239-1255.

[11] W.G.J. Halford, A. Orso, AMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks, *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering ASE '05*, Long Beach, CA, 2005, p. 174-183.

[12] G. Wassermann, Z. Su, Sound and Precise Analysis of Web Applications for Injection Vulnerabilities, *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation PLDI '07*, San Diego, CA, 2007, p. 32-41.

[13] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic, Noxes: A Client-Side Solution for Mitigating Cross-Site Scripting Attacks, *Proceedings of the 2006 ACM Symposium on Applied Computing SAC' 06,* Dijon, France, 2006, p. 330-337.

[14] S. Kals, E. Kirda, C. Kruegel, N. Jovanovic, SecuBat: A Web Vulnerability Scanner, *Proceedings of the 15th International Conference on World Wide Web '06*, ACM Press, Edinburgh, Scotland, 2006, p. 247-256.

[15] D. A. Keim D, F. Mansmann,J. Schneidewind, T. Schreck, Monitoring Network traffic with Radial Analyzer, *2006 Symposium On Visual Analytics*, Baltimore, MD, 2006, p. 123-128.

[16] S-T. Teoh, S. Ranjan, A. Nucci, C-N. Chuan, BGP Eye: A New Visualization Tool for Real-time Detection and Analysis of BGP Anomalies, *Proceedings of the 3rd International Workshop on Visualization for Computer Security VizSEC '06*, Alexandria, Virginia, 2006, p. 81-90.

[17] S.T. Teoh, K-L. Ma, S-F. Wu, T.J. Jankun-Kelly, Detecting Flaws and Intruders with Visual Data Analysis, *Computer Graphics and Applications, IEEE, 24*(5), 2004, p. 27-35.

[18] S. Axelsson, Combining a Bayesian Classifier with Visualisation: Understanding the IDS, *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, ACM Press, Washington, DC, 2004, p. 99-108.

[19] J. Chirillo, Carol A. Long (Ed.), *Hack attacks revelead, 2nd edn* Indianapolis: Wiley Publishing, 2002, p. 485-544.

[20] Fingerprinting Port 80 Attacks, A look into web server and web application attack signatures, admin@cgisecurity.com, 2002.

[21] G. Carpenter, and S. Grossberg, A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine, *Computer Vision, Graphics and Image Processing 37*, 1987, p. 54-115.

[22] I. Xydas, G. Miaoulis, P-F. Bonnefoi, D. Plemenos, D. Ghazanfarpour, 3D Graph Visualisation of Web Normal and Malicious Traffic, *Information Visualization IV06*, London, UK, 2006, p. 621-629.

[23] S. Haykin, *Neural networks, a comprehensive foundation, 2nd edn* (Upper Saddle River, NJ: Prentice Hall, 1999).

[24] D. Montana, and L. Davis, Training feedforward neural networks using genetic algorithms. *Proceedings of 11th International Joint Conference Artificial Intelligence*, San Mateo, CA, Morgan Kaufmann, 1989, p. 762-767.

[25] GraphViz software, http://www.graphviz.org

[26] Tulip software, http://www.tulip-software.org

[27] I. Xydas, *Network security policy surveillance aid using intelligent visual representation and knowledge extraction from a network operation graph*, doctoral diss., University of Limoges, France, 2007.

Biographies

Ioannis XYDAS

Dr. *Ioannis XYDAS* graduated in Mathematics from the University of Athens (Greece), in Computer Science MSc. from the University of Geneva (Switzerland) in 1980 and in Computer Science Phd. from the University of Limoges (France) in 2007. He worked as software engineer at the European Organization for Nuclear Research (CERN) in Geneva, as a computer analyst in the Greek Parliament, as a database specialist in multinational company (DEC) and as technical manager in a networking company. He is technical manager of the Network Operation Centre at the Technological Educational Institution (TEI) of Athens and since 1996 has been a part time professor in networking in the Department of Computer Science, TEI of Athens. He is member of the Network Technical Committee of TEI of Athens and his research area is Network Security, Artificial Intelligence and Visualization.

Georgios MIAOULIS

Prof. *Georgios MIAOULIS* gained an Engineering degree from the Aristotle University of Thessaloniki, Greece, the MSc degrees in Informatics and Organisation from CNAM, Paris, France and in Quantitative Methods from EHESS Paris, France. He gained a PhD degree in Medical Informatics from the University of Crete, Greece and a PhD degree in Informatics from the University of Limoges, France. He has worked in the Construction Industry and in the Public Administration sector as Informatics Specialist. Since 1987 he has been a member of educational staff and from 1998 a full Professor in the Department of Informatics, TEI of Athens. His research interests include Intelligent Information Systems, Image-based and Visualisation Systems in various application fields (medical, education, engineering etc).

Pierre-François BONNEFOI

Pierre-François BONNEFOI is Assistant Professor at the XLIM Laboratory (UMR CNRS 6172) of the University of Limoges (France). He obtained his Ph.D at the University of Limoges (France) in June 1999 and in January 2004 joined the Information Security team. Since December 2005, Pierre-François has been working in the research field of ad hoc networks with the use of embedded secure hardware devices such as smartcard in the MADNESS project (Mobile Ad Hoc Network with Embedded Secure System). He also has an interest in the use of graphical visualization and artificial intelligence techniques to further his knowledge of security information about evolving state of a network. His favourite research interests are security, networks and artificial intelligence.

Dimitri PLEMENOS

Dimitri PLEMENOS is an Emeritus Professor at the XLIM laboratory of the University of Limoges (France). His main research area is intelligent techniques in Computer Graphics, including Declarative Modelling, Intelligent Rendering and Intelligent Virtual World Exploration. He is author or co-author of several papers and member of the IPC of many international conferences and journals. He is organiser and general chair of the 3IA International Conference on Computer Graphics and Artificial Intelligence.

Djamchid GHAZANFARPOUR

Djamchid GHAZANFARPOUR has been a Professor at University of Limoges, France, since 1993 and head of *Realistic Image Synthesis* research team. Professor GHAZANFARPOUR received his BS from *Tehran Polytechnic*, Iran, in 1980, his MS from *INPG* in Grenoble, France, in 1982, his Doctor-Engineering degree from *ENSMSE* in Saint-Etienne, France, in 1985, his PhD from *ENSMSE* and *Saint-Etienne University*, France, in1990, and his Research Supervision Diploma from *Strasbourg University*, France, in 1992. His main research interests include texture synthesis, surface details modelling, natural phenomenon synthesis and animation, antialiasing and real time rendering.