USING AN EVOLUTIONARY NEURAL NETWORK FOR WEB INTRUSION DETECTION

I. 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

Due to the complicated nature of detecting actual intrusions, most current Network Intrusion Detection Systems (NIDS) place the burden of distinguishing an actual attack from a large set of false alarms on the security analyst, resulting in a significant cognitive load. Artificial Intelligence combined with Visualization will take advantage of human perceptual abilities and expertise to amplify cognition. In this paper we will describe an Evolutionary Artificial Neural Network (EANN) used as the knowledge base for the classification of web attacks in a prototype system. The aforementioned system is a surveillance aid for the security analyst, offering him a user friendly visual tool to detect anomalies in web requests by exploring 3D graphs, to understand quickly the kind of undergoing attack by means of colours and afford him the possibility to navigate into the payload of the web request for further analysis and adequate response. The EANN system is an improvement of our original work that used a supervised multilayer Artificial Neural Network (ANN) as the web attacks classifier.

Keywords: Genetic algorithms, EANN, web attacks, visual analytics.

1. Introduction

Web sites are likely to be regularly scanned and attacked and therefore, organizations, companies and individuals are making every effort to build and maintain secure Web sites. The threat profile facing enterprise organizations has undeniably shifted from network-layer exploits to more formidable attacks against applications, primarily Web and Web services applications. NIDS assist security analysts by automatically identifying potential attacks from network activity and produce alerts describing the details of these intrusions. However, network IDS have problems, such as false positives, operational issues in high-speed environments and difficulty in detecting unknown threats.

According to a recent report published by the Common Vulnerabilities and Exposures (CVE) project [1], 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 ranked second and third today [2].

Currently, security analysts face an increasing workload as their environments expand and attacks become more and more frequent. They monitor network activity using an intrusion detection system (IDS) for evidence of actions that attempt to compromise the integrity, confidentiality or availability of a network or computer resource. They also continuously monitor output from IDS and they use that output in conjunction with other systems, network and firewall logs, to keep abreast of system activity and potential attacks.

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 Snort current version, an open source IDS available to the general public [3], there are more than 11,500 signatures for network intrusion detection,

more than 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.

According to a recent survey [4], in the intrusion detection area intelligent 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. Artificial Intelligence combined with Visualization can take advantage of human perceptual abilities and expertise to amplify cognition.

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.

In our first version of the prototype system [5] the expert system used for the web attack classification was a supervised multilayer Artificial Neural Network (ANN). Recently we replaced the ANN by a hybrid expert system, an Evolutionary Artificial Neural Network (EANN). In this paper we describe the aforementioned EANN, its design, its integration into the prototype system, its classification results, and its performance. Finally, a comparison of the two expert systems is presented to determine which classifier is the better of the two.

The rest of this paper is organized as follows: section 2 presents related work, section 3 presents briefly the visualization prototype ID system and describes in details the new development, section 4 describes the new system's performance evaluation and compares the two EANN and ANN classifiers. 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 ([6],[7],[8]).

Interesting works on the detection of web-based attacks have been published in the last few years. Statistical methods have been used in [9] 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 [10] and DFA (Deterministic Finite Automata) induction has been applied in [11] 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 [12] combine a static analysis and runtime monitoring to detect and stop illegal SQL queries. In [13] a sound, automated static analysis algorithm is developed for the detection of injections vulnerabilities, modelling string values as context free grammars and string operations as language transducers. In [14] 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 [15] 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 [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 directed 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 we used an Artificial Neural Network (ANN) for classification. ANNs 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. 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. Initially, for the prediction of the network output the "winner-takes-all" method was used, that is the output with the biggest value (rated between 0 and 1) determined the class of the web attack. Later, a threshold was used instead of the "winner-takes-all" mechanism. The best results were achieved with a threshold of 0.8.

In this 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 an 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. Prototype ID system with EANN classifier

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 into consideration in our design in addition to traditional types of web attacks (Unicode, directory traversal, buffer overflow, mail and CGI attacks). Different kinds of application insertion attempts are detected such as 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.

In total 30 fingerprints were used in the model to group all the different types of known web attacks. A description of web attacks can be found in [19] and a detailed description of web attack 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 Grossberg and Carperter's Adaptive Resonance Theory (ART1). The ART1 neural network created 15 clusters or classes. These 15 classes were finally grouped manually to 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 directory traversal (..\ or ../) fingerprints always included another type of attack (e.g. buffer overflow, command execution attempt, code injections 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*: Unix or Windows commands for code execution attempts.
2) *Insertions*: Application code injections (SQL, Perl, HTML, Javascript, Data Access).
3) *Trojan Backdoor Attempts*: Attacks triggered by virus and worms (Cod Red II, Sadmin, etc.).
4) *Mail*: Mail attacks through port 80 (formail, sendmail etc.).
5) *Buffer overflows*: Attacks corrupting the execution stack of a web application.
6) *Common Gateway Interface*: Exploitation of vulnerable CGI programs.

7) *Internet Information Server*: Attacks due to vulnerabilities of IIS.
8) *Cross Site Scripting (XSS)* attacks.
9) *Miscellaneous*: Unicode, coldfusion and malicious web request options such as PROPFIND, CONNECT, OPTIONS, SEARCH, DEBUG, PUT and TRACE.

3.2 Prototype modules

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.

3.3 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 our Institution's library 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, but 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). A short description of each module is given below so the reader can better understand the overall system structure and the new development in the knowledge base module, which will be described in detail.

3.4 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 1s fired in its vector representation and an attack belonging to a specific attack class has at least one binary representation. The outputs of the pre-processor module are two files, one with the feature vector and one with the request data. The feature vector will be the input to the classifier and the request data will be forwarded to the graph generator module. The extracted data are the most significant for the 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..%5c winnt/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 0 0 (feature vector) and
213.23.17.133 HEAD /Rpc/..%5c..%5c..%5cwinnt/ system32/cmd.exe /c+dir+c:\ (payload).

3.5 Knowledge base module

In this 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 [21]. 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.

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. 1] in which a real number corresponds to the weighted link from one neuron to another.

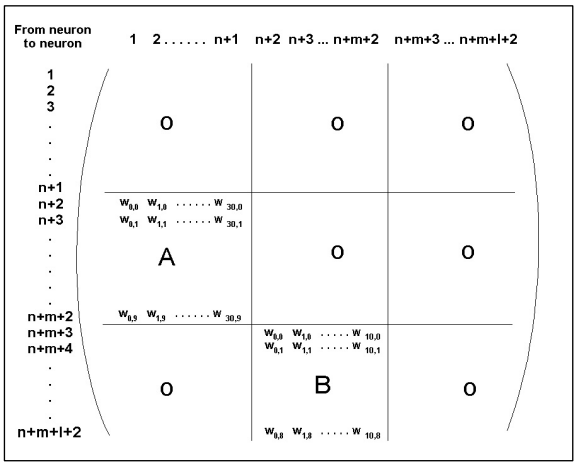

Figure 1. ANN's weight connection matrix

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 [22] 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.

The numbers of neurons are the same as in the original neural network (n=30, m=10, l=9 for input, hidden and output layer respectively). 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 following:

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

where the first part is the transposed matrix $W_{ih}[31,10]$ of weights between the input and the hidden layer (matrix A in Fig. 1, we string the rows together) and the second part concatenated is the transposed matrix $W_{ho}[11,9]$ of weights between the hidden layer and the output (matrix B in Fig. 1). 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. The algorithm converged if the minimum fitness was less than an epsilon, equal to 10^{-12} and the ratio minFit/avgFit was greater that 0.95. By setting such a severe criterion all members of the final generation became "ideal" and fit to be used for classification in the feedforward neural network, not just the best member of the population. Fig. 2 shows the evolution of the genetic algorithm. It converged 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.

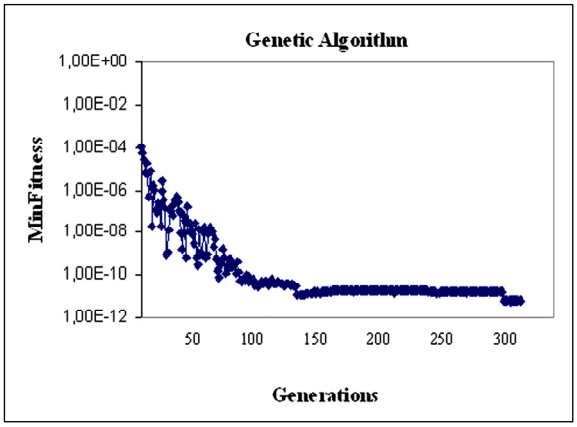

Figure 2. Genetic algorithm evolution

3.6 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 [23] 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 payload GET /hact/graphics /blackwell.gif, has as nodes the words "212.205.253.160", "GET", "hact", "graphics", "blackwell.gif" and as "directed edges" the links between these nodes from left to right: 212.205.253.160 -> GET -> hact -> graphics -> blackwell.gif. 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.

These two *dot* coloured graphs are then visualized with Tulip [24], a 3D graph visualization tool, supporting various graph algorithms and extensive features for interactive viewing and graph manipulation.

Fig. 3 shows normal and malicious web traffic and Fig. 4 only the malicious traffic for the same events. In Fig. 3 the brown graphs on the right indicate Perl injection attempts from IP 211.189.119.85, the red graphs indicate multiple command execution attempts from IP 200.24.5.98, 195.102.4.156 and other sources and the magenta graphs indicate multiple backdoor attempts (Code Red II) from IP 217.229.196.17.

In Fig. 4 we can spot additional command execution attempts from IP 213.23.17.133 and buffer overflow attacks from IP 195.77.248.102 (cyan graph). The Perl injection code can be easily read on the bottom right of the graph.

4 Comparison of EANN and ANN classifiers

The performance of the prototype IDS system has been evaluated in [25]. Figure 5 shows the performance of the EANN hybrid expert system versus the original Artificial Neural Network (ANN).

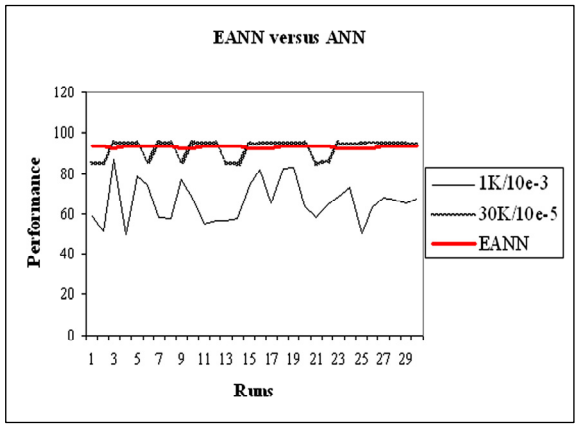

Fig 5. Comparison of EANN and ANN classifiers

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 the 30 runs (red near straight line in Fig. 5).

5. Conclusion

It is technologically impossible for any device to understand application communications or analyse application behaviour through the 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 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 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 to understand web server and web application security.

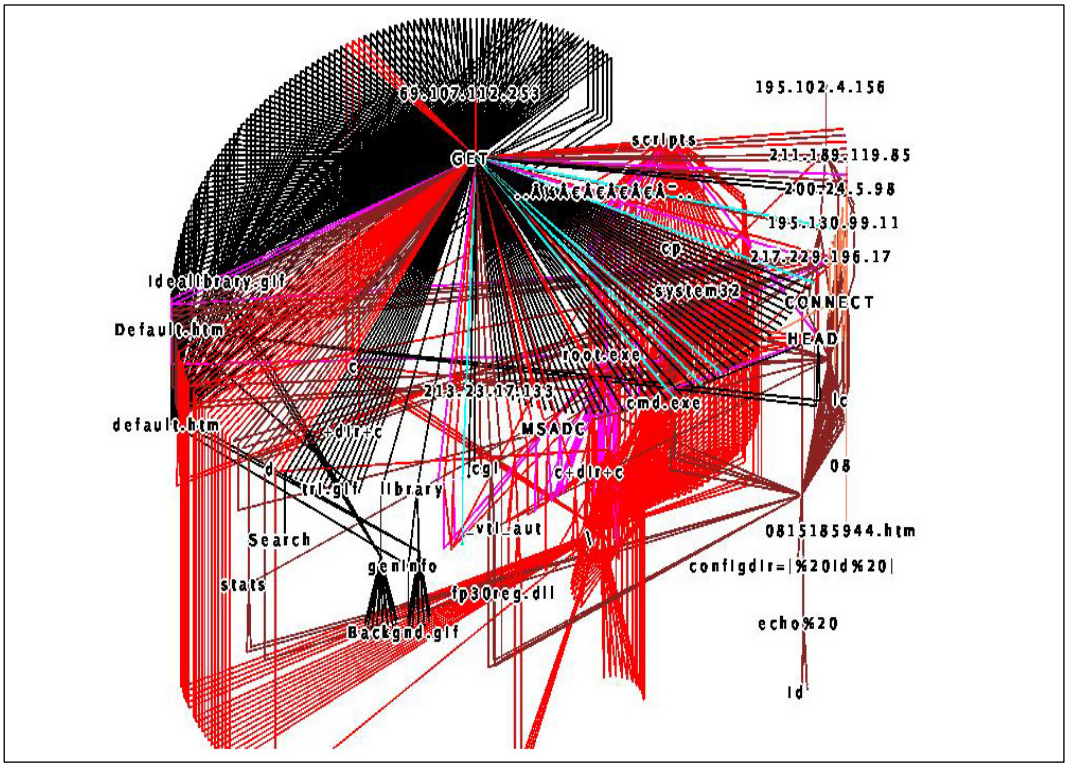

Figure 3. Normal and malicious web traffic

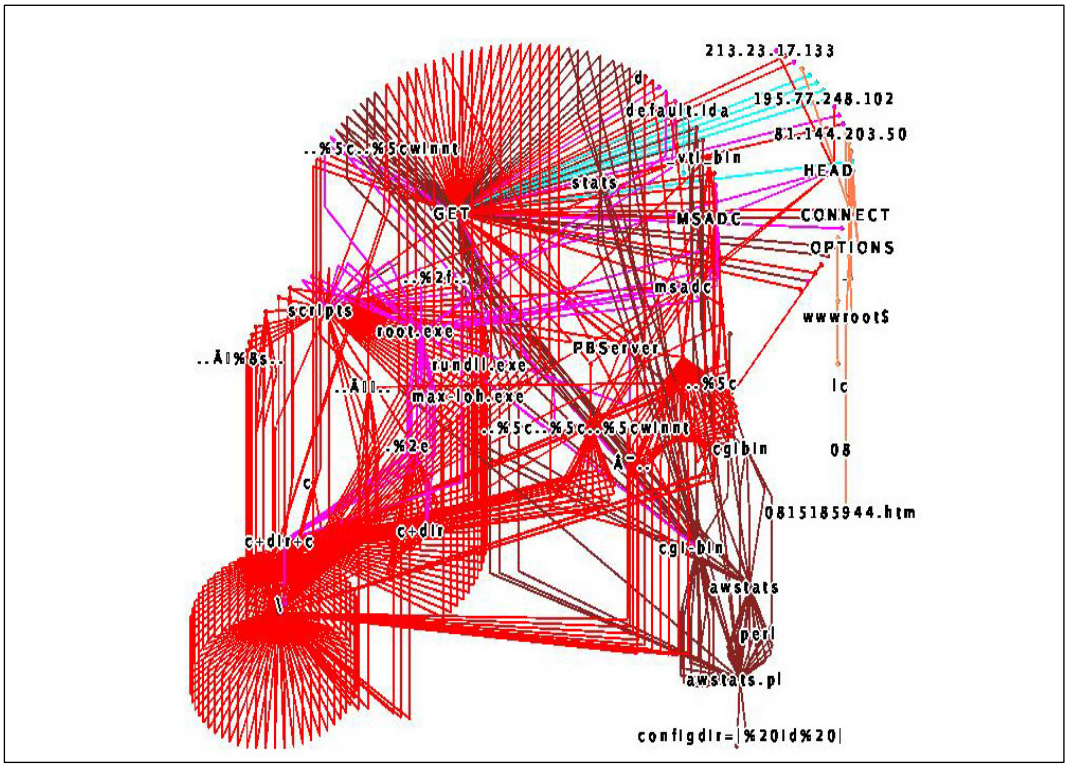

Figure 4. Malicious only web traffic

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 the new attack challenges and noisy data. This is undoubtedly the future in the ID area.

6. References

[1] CVE, Common Vulnerabilities and Exposures, The Standard for Information Security Vulnerability Names, http://www.cve.mitre.org, 2007.

[2] M. Cobb, Software security flaws begin and end with web application security, http://searchsecurity. techtarget.com, 2006.

[3] Snort software, http://www.snort.org, 2007.

[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, Apr. 2004, 1743-1746.

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

[6] 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, Nov. 2002, 265-274.

[7] W-H. Chen, S-H. Hsu, H-P. Shen, Application of SVM and ANN for intrusion detection, Computers and Operations Research, Vol.32 Issue 10, Elsevier, Oct. 2005.

[8] W. Lee, S. Stolfo, K. Mok, Adaptive Intrusion Detection: A Data Mining Approach, Artificial Intelligence Review, Vol.14 Issue 6, Kluwer Academic Publishers, Dec. 2000, 533-567.

[9] C. Kruegel, G. Vigna, W. Robertson, A multi-model approach to the detection of web-based attacks, Computer Networks, Vol. 48 Issue 5, Elsevier, Aug. 2005, 717-738.

[10] S. Cho, S. Cha, SAD: web session anomaly detection based on parameter estimation, *Computers & Security*, Vol. 23, Issue 4, June 2004, 312-319.

[11] K.L. Ingham, A. Somayaji, J. Burge, S. Forrest, Learning DFA representations of HTTP for protecting web applications, *Computer Networks*, Vol. 51, Issue 5, April 2007, 1239-1255.

[12] 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*, Nov. 2005, 174-183.

[13] 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*, June 2007, 32-41.

[14] 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,* April 2006, 330-337.

[15] 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, May 2006, 247-256.

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

[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*, Vol. 24, Issue 5, Sept-Oct. 2004, 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, Oct. 2004, 99-108.

[19] J. Chirillo, *Hack Attacks Revelead, 2nd edn* (Wiley Publishing, 2002, 485-544).

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

[21] S. Haykin, *Neural Networks, A Comprehensive Foundation, 2nd edn*, (Prentice Hall PTR, 1999, 156-208).

[22] 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, 762-767.

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

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

[25] I. Xydas, *Thesis*: "Network security policy surveillance aid using intelligent visual representation and knowledge extraction from a network operation graph", France, June 2007.