# A Denial of Service Detector based on Bayesian Classifiers and the Random Neural Network

Gülay Öke* and Georgios Loukas

*Electrical and Electronic Engineering, Imperial College London, UK*

## Abstract

In spite of extensive research in defence against Denial of Service (DoS), such attacks remain a predominant threat in today's networks. Due to the simplicity of the concept and the availability of the relevant attack tools, launching a DoS attack is relatively easy, while defending a network resource against it is disproportionately difficult. The first step of any comprehensive protection scheme against DoS is the detection of its existence, ideally long before the destructive traffic build-up. In this paper we propose a generic approach for DoS detection which uses multiple Bayesian classifiers and random neural networks (RNN). Our method is based on measuring various instantaneous and statistical variables describing the incoming network traffic, acquiring a likelihood estimation and fusing the information gathered from the individual input features using likelihood averaging and different architectures of RNNs. We present and compare seven different implementations of it and evaluate our experimental results obtained in a large networking testbed.

### Keywords

Denial of Service, Random Neural Networks, Network Security, Intrusion Detection, Bayesian Classifiers

## 1 Introduction

During the last decade Denial of Service attacks (DoS) have evolved from simple acts of nuisance to a predominant network security threat with repercussions including significant financial losses [39], endangerment of human life [40] and compromising of national security [41]. In the majority of DoS attacks the attacker acquires control of a large number of hosts, which are unaware that their machines are compromised, and orders them to simultaneously target a victim network node or set of nodes. The goal of a DoS attack is to deny service to legitimate users of the victim system by overwhelming its network or processing resources.

The extreme diversity of DoS attack types, targets and motives has produced similarly diverse protection proposals from the network security research community. In the most general sense, one can identify three stages that should comprise a complete DoS defence system:

- **Detection**. Usually a system running on the victim machine identifies in real-time the existence of an attack and triggers the initiation of the next two stages. The detection phase consists of looking either for anomalies in the incoming traffic or for signature characteristics of known attack types.
- **Classification**. Then, the victim or the rest of the nodes of the same network should monitor the incoming traffic and attempt to distinguish between normal traffic (sent by legitimate users) and attack traffic (sent by nodes controlled by the attacker).
- **Response**. Common approaches for the response phase include dropping the traffic that was identified as attack traffic during the classification phase, and redirecting it to a trap where it can be analysed.

This three-phase paradigm does not mean that detection, classification and response have to be separate entities or that they have to be performed in strict sequence. For example, detection and classification may be achieved by observing the same traffic characteristics. Over the years several smart methods

*g.oke@imperial.ac.uk, Imperial College, Exhibition Road, SW7 2BT, London, UK

have been proposed to address each of these stages, but most become obsolete to an extent after a while, as the attack types evolve to counter the latest defence trends.

In this paper we concentrate on the first of the three stages. To provide a network with an effective system of protection against DoS attacks, one must first employ a method to detect such an attack. This would not be needed in the case of an ideal response architecture with proactive qualities that would render a DoS attack impossible, but such a system has not been built to date, and proactive solutions are usually too expensive resource-wise to operate in the absence of an attack. A detection mechanism should monitor the traffic continuously and signal any developing attacks in the network, which should then trigger a response mechanism aiming to protect the network resources and maintain a satisfactory level of quality of service for the legitimate users. The success of a detection mechanism is determined by a number of factors, including its probability of correct detection of the attack, missed detection, and false alarm in the absence of an attack. Also, it should consume minimal resources and reach detection decisions quickly in real-time so that the classification and response mechanisms are initiated before the attack builds up.

Since DoS detection is often considered as a pattern recognition problem where the aim is to observe and analyse the incoming traffic, various machine learning techniques have been proposed. For example, the authors of [29] have designed the Statistical Pre-Processor and Unsupervised Neural Net based Intrusion Detector (SPUNNID), in which the statistical pre-processor is used to extract features from packets, and the feature vector is changed to numerical form and fed to an unsupervised Adaptive Resonance Theory net (ART). Neural networks for DoS detection are also used in [15], which follows a data mining approach. In [28], a scheme is presented which comprises a collector of the appropriate data fields from the incoming packets, a feature estimator that evaluates the frequencies for the encoded data, and a radial basis function neural network detector to characterise the incoming traffic as normal or DoS. In [30], an Adaptive Neuro-Fuzzy Inference System is used together with a Fuzzy C-Means Clustering Algorithm to detect DoS attacks, and in [33], a fuzzy classifier is decribed, where cross-correlation functions between incoming and outgoing traffic are used as inputs. In [16] another three computational intelligence techniques for DoS detection are compared, namely support vector machines, multivariate adaptive regression splines and linear genetic programs, for each one of which the same authors present feature ranking algorithms in [17]. The problem of selecting the most useful input features has also been addressed in [26], where a genetic algorithm is used.

An important section of DoS detection research is directed towards observing and analysing statistical properties and the energy content of normal and attack traffic. Normal Internet traffic is known to be long-range dependent (LRD) and self-similar, but in the case of a DoS attack there are usually important deviations for these properties [34]. For example, in [19] the incoming traffic is characterised as normal or DoS based solely on its autocorrelation function. In [20], the self-similarity property of Internet traffic is used to identify DoS attacks. The authors use the packet number or packet size as the input feature and evaluate the Hurst parameter $H$ by statistical techniques. In their approach, the variance of $H$ in consecutive time intervals is calculated and if there is doubling of the variance, it is decided that a DoS attack is in progress. In [12] the entropy is computed as a measure of randomness, and the chi-square statistic as a measure of statistical significance and estimate of confidence, to detect the existence of an attack. Based on the observation that highly correlated packets are used in Dos attacks, while legitimate traffic is in some sense random, in [9] Kolmogorov complexity metrics, a measure of correlation in the strings forming the packets, are used for DoS detection. This technique is later improved in [32], where a different Kolmogorov metric estimate, which measures the correlation between the first and second halves of the strings is proposed and the fluctuations of the Kolmogorov complexity differentials are evaluated. Another paper which suggests the use of signal processing techniques is [23] where single and multi-source DoS attacks are classified using the information of packet header content, ramp-up behaviour and spectral content.

The Cumulative Sum (CUSUM) algorithm which is a change point detection algorithm applicable for detecting sharp changes in variables, has been used widely in DoS detection. It was first proposed for DoS detection in [10], where TCP flags of the incoming traffic were used as input. In [22], the performance of the CUSUM as a SYN-flood DoS detector is evaluated in terms of detection probability, false alarm ratio, and detection delay.

Since the energy distribution of normal traffic is known to be relatively stationary, while an attack usually results in changes in the energy distribution variance, in [31] wavelets are used for computing the variations in the energy distribution in the incoming traffic. In another study that exploits energy content [18], flat energy bursts in the traffic are determined with the continuous wavelet transform.

Here, we attempt to bridge these two general directions of DoS detection, machine learning and information gathered with statistical methods. We have built a system which uses several statistical features deemed in the literature as most significant for a DoS attack, and combines the individual decisions in a machine learning fashion. We present and compare seven different implementations of it, which combine multiple Bayesian classifiers and the random neural network (RNN). Bayesian classifiers have been used before for DoS detection [13], but applied only on the rate of appearance of specific flags in the packets' headers, and in [36], where hypothesis testing was used on the spectral analysis of bitrate to detect only one very specific type of attack. In our work we present a more general approach which aggregates likelihood estimation of heterogeneous statistical features and combine them in a neural network structure.

The random neural network (RNN) introduced by Gelenbe [2] is an alternative neural network model based on the spiking behaviour of the biological neuron instead of the classical approaches which assume analog transmission of signals. In this paper, we exploit the capability of the RNN to model the excitatory and inhibitory interactions among its inputs for the case of malicious incoming traffic in a network. In fact, the experiments we conduct also exploit a RNN-based networking protocol, the Cognitive Packet Network (CPN) [24], which is an autonomic Quality of Service (QoS)-driven routing protocol. In CPN each flow specifies the QoS metric that it wishes to optimise, and data payload is carried by source routed "dumb packets" (DPs), while "smart packets" (SPs) and "acknowledgment packets" (ACKs) gather and carry control information which is used for decision making. In CPN, each flow specifies its QoS requirements in the form of a QoS "goal". SPs associated with each flow constantly explore the network and obtain routing decisions from network routers based on observed relevant QoS information. At each CPN node, the SPs use a local reinforcement learning algorithm based on measurements collected by previous SPs and ACKs, to elicit a decision from the node for the next hop to travel to. Here, CPN uses the RNN because it has a unique solution to its internal state for any set of "weight" and input variables. At each node there is a separate RNN for each QoS class and destination, and for a given QoS class, a specific neuron of the corresponding RNN is associated with a specific output link of the node. When a SP reaches the destination node of the flow, an ACK packet is generated and returned to the source according to the opposite path traversed by the SP. When the ACK reaches the source, the the reverse of the route that it used is stored for subsequent payload of dumb packets (DPs) which will be source-routed to the destination. The performance of the CPN routing protocol has been thoroughly investigated in [7, 8, 25]. In our experiments we use the CPN to ensure that the traffic arrives to their destination quickly using the optimal routes.

The rest of this paper is structured as follows. In section 2 we give a brief summary of the Bayesian decision theory as applied to our two-class pattern classification problem of distinguishing between normal and DoS traffic. We continue in section 3 with a detailed description of our choice of input features, and in section 4 with offline statistical information gathering. Section 5 presents the core of the detection technique, including the decision taking process and the methods for the fusion of the information gathered from the individual features. Then, section 6 contains the performance evaluation of our detection mechanism in a networking testbed and we conclude in sections 7 and 8 with a discussion on flash-crowds and our final remarks.

## 2 Multiple Bayesian Classifiers in DoS Detection

As mentioned earlier, the task of DoS detection can be formulated as a pattern classification problem, where the observed traffic is to be classified as normal or attack traffic. The Bayesian Decision theory is a basic approach used in pattern recognition problems. It assumes the availability of probabilistic descriptions of the underlying features of a problem and aims to find a decision rule which would minimise the risks encountered by the decision taking process [6]. For a two-category classification problem, let us assume we can measure an observation value $x$ for a certain feature, and we have to decide whether the observed data point falls into the normal ($w_N$) or DoS ($w_D$) category. We can proceed as follows according to Bayesian decision theory:

$$\text{decide } w_D \text{ if } P(w_D|x) > P(w_N|x); \text{ otherwise } w_N$$

where $P(w_N|x)$ and $P(w_D|x)$ are the posterior probabilities. For evidence $f(x)$ of the specific feature, and prior probabilities $P(w_N)$ and $P(w_D)$, the joint probabilities according to Bayes rule become:

$$f(w_N, x) = P(w_N|x)f(x) = f(x|w_N)P(w_N)$$
$$f(w_D, x) = P(w_D|x)f(x) = f(x|w_D)P(w_D)$$

Since the evidence $f(x)$ is only a scale factor and can be neglected in computations, we get:

$$\text{decide } w_D \text{ if } f(x|w_D)P(w_D) > f(x|w_N)P(w_N); \text{ otherwise } w_N$$

Now we can compute the average risk $R$, incurred by the costs $c_{ij}$, the cost of deciding in favour of class $i$, represented by subspace $r_i$, when $j$ is the actual class of the data point. For our two-category classification problem, the average risk is evaluated as:

$$R = c_{DD}P(w_D)\int_{r_D} f(x|w_D)dx + c_{NN}P(w_N)\int_{r_N} f(x|w_N)dx$$
$$+c_{ND}P(w_D)\int_{r_N} f(x|w_D)dx + c_{DN}P(w_N)\int_{r_D} f(x|w_N)dx$$

Above, since the first two terms represent correct decisions, we assume $c_{DD} < c_{ND}$ and $c_{NN} < c_{DN}$. Also, $r = r_D + r_N$ and $\int_r f(x|w_D)dx = \int_r f(x|w_N)dx = 1$. So, the average risk is derived as:

$$R = c_{DD}P(w_D) + c_{NN}P(w_N) +$$
$$\int_{r_D} [P(w_N)(c_{DN} - c_{NN})f(x|w_N) - P(w_D)(c_{ND} - c_{DD})f(x|w_D)]dx$$

Since the first two terms are fixed, to minimize $R$, we must ensure that $P(w_D)(c_{ND} - c_{DD})f(x|w_D) > P(w_N)(c_{DN} - c_{NN})f(x|w_N)$, which is equivalent to:

$$\frac{f(x|w_D)}{f(x|w_N)} > \frac{P(w_N)(c_{DN} - c_{NN})}{P(w_D)(c_{ND} - c_{DD})}$$

Hence, the practical utilisation of Bayesian classifiers in the two-category classification problem entails evaluating the likelihood ratio $\Lambda(x) = \frac{f(x|w_D)}{f(x|w_N)}$ and comparing it with a threshold $T = \frac{P(w_N)(c_{DN}-c_{NN})}{P(w_D)(c_{ND}-c_{DD})}$, and $x$ is assigned to category $w_D$ if $\Lambda(x) > T$, or to $w_N$ otherwise.

In our DoS detection mechanism, we monitor the incoming traffic to measure various features for decision taking and we utilise multiple Bayesian classifiers to take individual decisions for each of the input features. The collected information is then combined in a fusion phase to yield an overall decision about the traffic. We have used a likelihood averaging method and implementations of two different architectures (feedforward and recurrent) of the RNN to compare. In the following sections we present our approach, including the selection of the input features, offline statistical information gathering and information fusion for the final decision taking.

## 3 Selecting the Input Features

For any pattern classification problem, the selection of useful and information bearing input features constitutes a significant part of the solution. In our scheme we have used the features described below for a generic description of the monitored traffic. Our motivation behind this choice was to try to capture both the instantaneous behaviour and the longer-term statistical properties of the traffic, and also to employ input features which were easily measurable or calculable without high computational cost. Since the goal of the attacker is to deny or degrade the service for legitimate users by overwhelming either the processing or the networking resources of a victim network, a DoS detection mechanism should not further aggravate this condition with considerable overhead. Being able to measure them quickly is also a factor, since the faster detection decisions are taken the easier it is for the classification and response mechanisms to counter the attack.

- ***Bitrate***. An unexpectedly high rate of incoming traffic is by far the most conspicuous indicator of a flooding DoS attack. Similar measurements, such as the number of packets per flow are often used in detection mechanisms [15].

- ***Increase in Bitrate***. Another obvious characteristic of DoS attacks is the sudden and sustained rate of increase of the bitrate of the incoming traffic. For example, flooding attacks start with a long period of increasing bitrate, while in pulsing attacks there are consecutive periods of increasing and decreasing bitrate.

- ***Entropy***. If the data has a probabilistic description, e.g. in terms of probability distribution functions, the entropy will be inherently related to the randomness or uncertainty of information in the data. It has been reported in the technical literature that the entropy contained in normal internet traffic and traffic under DoS attack differ significantly [12]. In our work, we compute the entropy of the value of the incoming bitrate at the nodes we monitor according to [1]:

$$E = -\sum_{i} f_i log f_i$$

where $f_i$ are the histogram values obtained for the bitrate, as explained in Section 5. This is expected to yield a higher value when the probability distribution expands over a wider range of values, indicating an increase in uncertainty.

- ***Hurst Parameter.*** Another statistical attribute which exhibits different behaviour for normal and attack traffic is the self-similarity. It has been studied in detail in [34] that the self-similarity properties of normal and attack traffic are distinctively different. Hurst parameter is an indicator of the self similarity of traffic and can be used in DoS detection. For example, Xiang et al. [20] use the variations of the Hurst parameter of the number and the size of packets to detect attacks. In our approach we compute the actual value of the Hurst parameter for the incoming bitrate, for which we have used the (R/S) analysis, as described in [21]. If $x$ is the bitrate of the incoming traffic, $n$ is the observation time, and $N$ is the total number of observation points, then $(R/S)$ is given by:

$$(R/S)_N = \frac{\max\limits_{1\le n\le N} \sum_{n=1}^{N}(x-\bar{x}) - \min\limits_{1\le n\le N} \sum_{n=1}^{N}(x-\bar{x})}{\sqrt{\frac{\sum_{n=1}^{N}(x-\bar{x})^2}{N}}}$$

The Hurst parameter and $(R/S)_N$ are related by $(R/S)_N = cN^H$, which for $c = 1$ becomes $H = log_N((R/S)_N)$.

- ***Delay.*** A natural consequence of high bitrate and building up of congestion is the increase in the packet delays. Still, to our knowledge it has not been used before as an attack indicator. For the fastest and least invasive way to detect changes in the delays, the node we monitor sends constantly packets at a very low rate to all its direct neighbours. By measuring the average round trip time (RTT) for the acknowledgments to return, we have a clear indication of the congestion near the node.

- ***Delay Rate.*** As with bitrate, depending on the type of the attack and for its whole duration, the packet delays are expected to undergo significant changes. We are not aware of existing work using the change of the delay as a detection feature, but we consider it a natural next step.

## 4 Offline Statistical Information Gathering

The probabilistic description of the network is acquired in the statistical information gathering phase which mainly consists of two steps. First, the probability density function (pdf) values are obtained for both normal and attack traffic and then the likelihood ratios are calculated based on the pdfs. At each victim candidate of the network, the incoming traffic is analysed offline to collect this statistical information. Estimates of probability density functions for both normal and attack traffic are computed for each of the input features described in Section 3. The pdfs are denoted by $f_{feature}(x|w_N)$ and $f_{feature}(x|w_D)$, where *feature* is replaced by bitrate, increase in bitrate (bit acceleration), entropy, Hurst parameter, delay and delay rate respectively, $x$ is the measured value of the feature from the available traffic data, $w_N$ denotes the normal traffic and $w_D$ the DoS traffic. We have used the histogram method to compute the estimates of the probability density functions. With this method the range of observable values for a variable is divided into a number of intervals and for each interval, we compute the ratio of the number of data points that fall into it to the total number of data points available [6].

In the second step, the probability density function estimates obtained above for each input and for both traffic types are used to compute the likelihood ratios $l_{feature}$ of each feature: $l_{feature} = \frac{f_{feature}(x|w_D)}{f_{feature}(x|w_N)}$. These likelihood ratios are later used in real-time by the decision taking mechanism (Section 5). Likelihood ratios, actual values and quantised actual values (histogram category values) of the features are used also in the training of RNNs.

## 5 Detection decision

The statistical information collected about the network off-line is utilised during the decision taking process, which comprises two steps. In the first step, decision for each feature is given individually, and the individual decisions are then combined in an information fusion step to yield a final outcome for the state of the traffic. The numerical values of the features are measured in real-time and a likelihood value for each feature is computed. Then, these values are aggregated in a higher-level decision taking step, which provides a compensation for possible errors, and should decrease the rate of false alarms and missed detections. As a first approach to take into account all features, we have measured an average of the individual likelihoods. Then, for a more accurate approach, we employed a feedforward (f-RNN) and a recurrent (r-RNN) architecture of the RNN, with input the individual likelihood values, histogram category values and actual values, for a total of seven different implementations of our generic detection technique.

### 5.1 Average likelihood estimation

An uncomplicated and easily applicable method to combine the individual likelihoods is to evaluate the arithmetical average of them given by:

$$l_{final} = \frac{l_{bit} + l_{acc} + l_{entr} + l_{Hurst} + l_{delay} + l_{delrate}}{total\ number\ of\ features}$$

$l_{final}$, which has a value between 0 and 1 gives an overall likelihood for the existence of a DoS attack at the victim candidate we monitor. The decision on whether the incoming traffic is normal or DoS is then taken by comparing this value to a specified threshold, which may or may not be dependent on the impact that the DoS attack is expected to have on the victim.

### 5.2 Combining statistical information with the RNN

The random neural network (RNN), proposed by Gelenbe [2] is a computational paradigm, inspired by the random spiking behaviour of the biological neurons. The RNNs are computationally efficient structures and they represent a better approximation of the true functioning of a biophysical neural network, where the signals travel as spikes rather than analog signals.The strong analogy between queuing networks and the RNN make it a powerful tool for dealing with problems where excitation and inhibition among problem inputs are prevalent. The RNN has been successfully applied in various problems, including image processing [4], pattern recognition [35], and optimisation [3].

In the RNN [2], neurons exchange positive and negative impulse signals, with unit amplitude, which represent excitation and inhibition respectively. Neurons accumulate signals as they arrive and positive signals are cancelled by negative signals. Neurons may fire if their potential is positive, to send signals either to other neurons or outside the network. The potential of neuron $i$ at time $t$, also called as the state of the neuron, is denoted by $k_i(t)$. The state $k_i(t)$ may decrease if the neuron emits a signal, an inhibitory signal is received from another neuron in the network or if an exogenous inhibitory signal is received. Similarly, $k_i(t)$ increases if an exogenous excitatory signal or an excitatory signal from another neuron in the network is received. In RNN a signal may leave neuron $i$ for neuron $j$ as a positive signal with probability $p^+(i,j)$, as a negative signal with probability $p^-(i,j)$, or may depart from the network with probability $d(i)$, where $p(i,j) = p^+(i,j) + p^-(i,j)$ and $\sum_j p(i,j) + d(i) = 1$. Positive and negative weights are computed by:

$$w^+(j,i) = r(i)p^+(i,j) \geq 0$$
$$w^-(j,i) = r(i)p^-(i,j) \geq 0$$

where $r(i)$ is a Poisson firing rate, with independent identically exponentially distributed interimpulse intervals:

$$r(i) = \sum_j w^+(i,j) + w^-(i,j)$$

The weights $w$ may be interpreted in a way analogous to the weights in artificial neural networks (ANNs), but they actually represent excitatory and inhibitory signal emission rates.

The steady state probability that the neuron $i$ is excited is defined as:

$$q_i = lim_{t\to\infty} Pr[k_i(t) > 0] \tag{1}$$

It is computed to be $q_i = \frac{N(i)}{D(i)}$, where

$$N(i) = \sum_j q_j w^+(j,i) + \Lambda(i)$$

$$D(i) = r(i) + \sum_j q_j w^-(j,i) + \lambda(i)$$

with $\Lambda(i)$ and $\lambda(i)$ denoting the rates of exogenous excitatory and inhibitory signal inputs into neuron $i$, respectively.

RNNs can be designed in both feedforward and recurrent architectures (f-RNN and r-RNN). In our work we have implemented both architectures for the fusion of inputs to compare the results. The structure of the f-RNN we have designed is shown in Fig. 1.

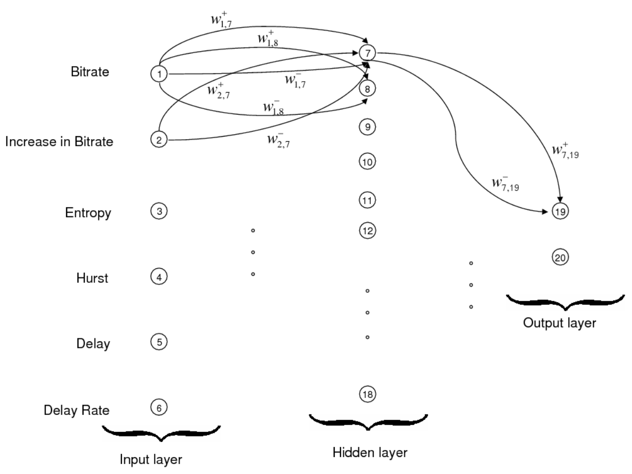

Figure 1: Random Neural Network in the feedforward architecture used in the experiments

It consists of an input layer of six nodes, a hidden layer with twelve nodes and an output layer with two nodes. Each output node stands for a decision; attack or not. The final decision about the traffic is determined by the ratio of the two output nodes. The r-RNN structure we have designed is depicted in Fig. 2. It consists of two layers, an input layer with twelve nodes and an output layer with two nodes. In the input layer, there are two nodes for each input variable; one for the excitatory signals and one for the inhibitory signals. Each node sends excitatory signals to nodes of the same type and inhibitory to the rest. At the output layer, one node sums up the excitatory and a second the inhibitory signals. Just as in the feedforward case, the decision is given by computing the ratio of the two output nodes.

For the implementation of the RNNs, we have used software developed in [42]. For both f-RRN and r-RNN we have used three different types of input:

- **Likelihood ratios**. The likelihood ratios $l_{feature}$ for the six input features are obtained by measuring their real-time values and resorting to the likelihood ratio values stored during the offline statistical information gathering phase described in section 4.

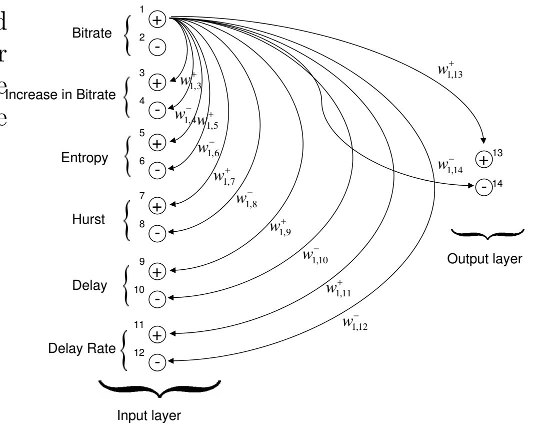

Figure 2: Random Neural Network in the recurrent architecture used in the experiments

- **Histogram categories**. These refer to the histogram intervals used to estimate the pdfs described in section 4 and they essentially quantise the available values.

- **Actual values**. For the sake of comparison we have also investigated the case in which the actual values of the input features are directly fed in the two types of RNN.

## 6 Experimental Evaluation

We have implemented and evaluated our detection mechanism on a networking testbed in our laboratory, which consists of 46 nodes connected with 100 MBits/sec links according to the topology of Fig. 3. Instead of following a random topology, we chose to recreate a representative academic one, which is the SwitchLAN [1] backbone network topology. We chose a specific node to play the role of the victim while the rest of the nodes send traffic to it according to a variety of datasets that we tried. In our experiments, we utilised traffic traces of DoS attacks designed both

[1]The SwitchLAN network provides service in Switzerland to all universities, two federal institutes of technology and the major research institutes.

in our laboratory and by other academic sources: (i) normal traffic, (ii) attack traffic of our design slowly increasing, (iii) attack traffic downloaded from [38] representing rapid flood, and (iv) attack traffic downloaded from the same source representing a pulsing attack, with traffic rates reciprocating between very low and very high values, which makes it challenging to detect.

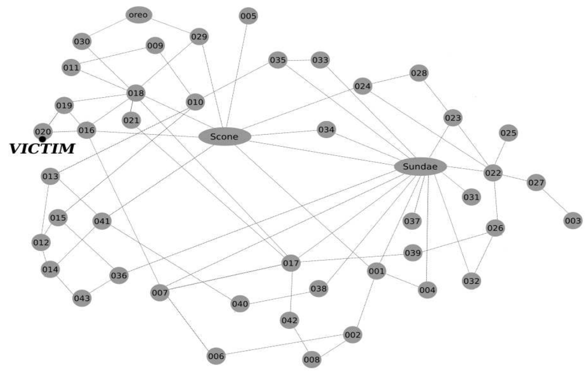

Figure 3: The network topology used in the experiments

We have used two types of RNN architecture, f-RNN and r-RNN, with three types of input for each feature: likelihood ratios, quantised histogram category values and actual values. Together with the method of simply averaging the likelihoods (for various values of decision threshold), we have tested a total of seven variations of the detection mechanism on our four traffic datasets. Each experiment lasts 120 sec. In the attack cases, to illustrate the difference in the traffic and to see graphically the operation of the detection mechanism, we start with normal traffic on which we superimpose attack traffic for the time period between 50s and 100s. The last 20s the network returns to its normal operation, as the attack sources stop sending traffic to the victim. We used a sampling time of 2 sec.

Table 1 summarises the performance results of the detection mechanism in terms of average correct detection, false alarm rates and detection delays, while figures 4-7 show the real-time detection decisions as time progresses. The $y$ axis in these graphs is in logarithmic scale with the detection metric being the ratio of the two output RNN nodes as described in section 5. The decision threshold over which an attack is signalled is the RNN output ratio of 1. The closer the detection ratio is to 1 the less certain the mechanism is of its detection decision.

The results of Fig. 4 show that for normal traffic the RNNs signal correctly the absence of attack throughout the duration of the experiment, although not with the same degree of certainty, apart from the case of f-RNN with actual values which incorrectly signals an attack. Figures 5-7 correspond to the three attack datasets that we used. All three implementations of the RNN detect the attacks quickly and have minimal missed detections and false alarms. The r-RNN with actual values had high detection rates, no false alarms and the lowest detection delay for all datasets, but also provided the lowest degree of certainty for its decisions by yielding values close to 1. The use of the quantised values of histogram categories appear to improve the correct detection rate even further, but at the expense of a few false alarms, while the likelihood value methods for both f-RNN and r-RNN performed at about the same level in both respects, but took a little longer to detect the pulsing attack (dataset3). We have also include the results for the average likelihood method, which performs adequately for all datasets, but detects the attacks with longer delay (up to 12s for dataset3).

We have also plotted some representative ROC curves for the seven variations, based on one of the datasets (Fig. 8). The ROC curves are graphs of false positives versus true positives, to which we have referred in this paper as false alarms and correct detections respectively. In an ideal system, the ROC curve should rise immediately from (0,0) to (0,1) and continue to (1,1). We see this behaviour in all seven variations, with a slight advantage of the r-RNN with histogram categories and f-RNN with likelihood values.

Looking at the overall performance of our detection mechanism, it emerges that apart from the f-RNN with actual values implementation, all methods are quite powerful detectors of attack traffic. In general, the r-RNN implementations are observed to be more accurate and slightly faster.

## 7 Discussion on flash-crowds

The flash crowd is a situation in which there is an unusually sharp increase in the number of legitimate visitors to a website due to some significant event. Although all traffic may be legitimate, the consequences are very similar to those of DoS attacks, such as network outages and dramatically reduced quality of service. In this paper we have not presented any work on flash-crowds. We could, however, discuss

| | False Alarm | | | Correct detection | | | Detection Delay (s) | | |
|---|---|---|---|---|---|---|---|---|---|
| Detection implementation | Data1 | Data2 | Data3 | Data1 | Data2 | Data3 | Data1 | Data2 | Data3 |
| Avg. likelihood (threshold 0.4) | 0.11 | 0.06 | 0.03 | 0.92 | 0.96 | 0.80 | 2 | 0 | 12 |
| Avg. likelihood (threshold 0.5) | 0 | 0.03 | 0 | 0.80 | 0.88 | 0.76 | 4 | 2 | 12 |
| f-RNN likelihood values | 0.17 | 0.11 | 0.08 | 0.96 | 0.96 | 0.84 | 2 | 0 | 12 |
| f-RNN histogram categories | 0.03 | 0.11 | 0.03 | 0.92 | 1 | 0.80 | 4 | 0 | 8 |
| f-RNN actual values | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| r-RNN likelihood values | 0.06 | 0.11 | 0.03 | 0.96 | 0.96 | 0.80 | 2 | 0 | 12 |
| r-RNN histogram categories | 0.06 | 0.06 | 0.06 | 0.96 | 1 | 0.88 | 2 | 0 | 8 |
| r-RNN actual values | 0 | 0 | 0 | 0.92 | 1 | 0.84 | 2 | 0 | 6 |

Table 1: Comparison of the different detection implementations, in terms of false alarm and correct detection rates

briefly how the differentiation between "flash-crowd" and DoS attack could be incorporated in the detection mechanisms we presented.

The flash-crowd can be added as a third category in our Bayesian classification problem. Then, the learning process of the detection mechanism and the training of the RNN will need sets of three types of traffic to operate; normal, attack and flash-crowd. In addition to the six inputs, the simplest new RNN structure would need at least eighteen neurons in a hidden layer and three outputs.

A second option would be to first detect the existence of an attack in the ways presented in this paper, and then trigger a second separate mechanism, which should attempt to distinguish between the two by actively challenging the authenticity of the clients and exploiting their fundamental difference: flash crowd flows are generated by human users, while attack flows are generated by compromised computers. Thus, Reverse Turing tests have also been suggested to counter DDoS attacks against webservers [11]. The option of using our detection mechanisms to detect suspicious traffic and then try to actively distinguish between human-generated and computer-generated traffic may be more widely useful, since it does not depend on the statistical properties of the flash-crowd, which an attacker can easily mimic.

From the point of view of the users of the network, however, it is not important whether the outage they are experiencing is maliciously intended or a result of a flash-crowd. In either case, the detection mechanism should trigger a classification and response mechanism to retain the quality of service of the legitimate users. We have covered these elements of the DoS defence architecture in our previous papers [27, 37].

## 8 Conclusions

We have described the design of a generic DoS detection scheme which reaches detection decisions accurately and in a timely fashion, by employing multiple Bayesian classifiers and RNNs. We first select input features to capture both the instantaneous behaviour and the longer-term statistical properties of the traffic and in an offline information gathering step we obtain the probability density function estimates and evaluate likelihood ratios. Then, during the decision taking step, we measure the features of the incoming traffic to reach detection decisions according to each feature. These are combined into an overall detection decision using both feedforward and recurrent architectures of RNN.

There are two major weaknesses in current DoS research, namely the lack of standards of evaluation for the detection and defence methods and the scarce information on modern types of attacks. Launching real attacks against real networks with real legitimate users, is impractical, and this leaves the researchers with the option of less dependable datasets, e.g. simulated or acquired from outdated traffic traces. A pragmatic solution to these problems would be to organise close cooperation of the research community with organisations which are frequently under attack, such as e-commerce and online betting websites. Accurate and up-to-date datasets will help distinguish the best defence approaches in an unbiased manner and will prompt further research and improvements in detection and defence mechanisms. Until these goals are achieved, however, researchers will have to either use the existing obsolete datasets or create their own. In this work we chose to do both, but for the reasons we explained we cannot argue on how these datasets compare against others and how realistic they are. We can, however, argue that our investigation method, using a real large networking testbed, instead of simulation, should provide a significant degree of realism. The experiments we conducted showed that our mechanism is able to detect DoS threats in a timely fashion and can even very quickly identify the end of an attack, at least for the range of attacks that we investigated.

This paper presents part of our ongoing work to develop a complete DoS defence architecture covering all three aspects mentioned in section 1. In [27, 37] we detailed our approaches on classification and response, which naturally we intend to combine with the work presented here. Also, a way to improve the detection performance of our current mechanism for each individual node employing it is by introducing cooperation between the various nodes involved in the DoS defence. This we have already achieved in our work for classification and response, and would be particularly useful for accurate detection too.

## References

[1] Shannon C.E. and Weaver W. (1963) *The Mathematical Theory of Communication.* University of Illinois Press.

[2] Gelenbe E. (1993) Learning in the Recurrent Random Neural Network. *Neural Computation*, **5**, pp. 154-164.

[3] Gelenbe E., Koubi V., and Pekergin F. (1993) Dynamical random neural network approach to the traveling salesman problem. *in: Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', IEEE Conference Proceedings*, **2**, pp. 630-635.

[4] Cramer C., Gelenbe E., and Bakircioglu H. (1996) Low bit-rate video compression with neural networks and temporal subsampling. *Proc. IEEE*, **84(6)**, pp. 1529-1543.

[5] Haykin S. (1999) *Neural Networks A Comprehensive Foundation*, Prentice-Hall Inc., pp. 143-146.

[6] Duda R.O., Hart P.E. and Stork D.G. (2001) *Pattern Classification*, John-Wiley and Sons, pp. 20-214.

[7] Gelenbe E., Lent R. and Xu Z. (2001) Measurement and performance of a cognitive packet network. *Computer Networks (Amsterdam, Netherlands: 1999)*, **37(6)**, pp. 691-701.

[8] Gelenbe E., Lent R., Montuori A. and Xu Z. (2002) Cognitive packet networks: QoS and performance. *Proc. MASCOTS 2002, Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pp. 3-9.

[9] Kulkarni A.B., Bush S.F. and Evans S.(2002) Detecting Distributed Denial-of-Service Attacks using Kolmogorov Complexity Metrics. *Technical Information Series 2001CRD176, General Electric Company.*

[10] Wang H., Zhang D. and Shin K. (2002) Detecting SYN flooding attacks. *Proceedings of IEEE INFOCOM '02*, New York, USA, pp. 1530-1539.

[11] Morein W.G., Stavrou A., Cook D.L., Keromytis A.D., Mishra V. and Rubenstein D. (2003) Using graphic Turing tests to counter automated DDoS attacks against Web servers. *Proc. 10th ACM Int'l Conference on Computer and Communications Security (CCS '03)*, pp. 8-19.

[12] Feinstein L., Schnackenberg D., Balupari R. and Kindred D. (2003) Statistical Approaches to DDoS Attack Detection and Response. *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03).*

[13] Noh S., Lee C., Choi K. and Jung G. (2003) Detecting Distributed Denial of Service (DDoS) Attacks through Inductive Learning. *Lecture Notes in Computer Science*, **2690**, pp. 286-295.

[14] Gelenbe E., Gellman M and Loukas G (2004) Defending networks against denial of service attacks. In E.M. Carapezza, editor, *Proceedings of the Conference on Optics/Photonics in Security and Defence (SPIE)*, Unmanned/Unattended Sensors and Sensor Networks, **5611**, pp. 233-243.

[15] Kim M., Na H., Chae K., Bang H. and Na J (2004) A Combined Data Mining Approach for DDoS Attack Detection. *Lecture Notes in Computer Science*, **3090**, pp. 943-950.

[16] Mukkamala S. and Sung A.H. (2004) Computational Intelligent Techniques for Detecting Denial of Service Attacks. *Lecture Notes in Artificial Intelligence*, **3029**, pp. 616-624.

[17] Sung A.H. and Mukkamala S. (2004) The Feature Selection and Intrusion Detection Problems. *Lecture Notes in Computer Science*, **3321**, pp. 468-482.

[18] Yang X., Liu Y., Zeng M. and Shi Y. (2004) A Novel DDoS Attack Detecting Algorithm Based on the Continuous Wavelet Transform. *Lecture Notes in Computer Science*, **3309**, pp. 173-181.

[19] Li M. (2004) An Approach to Reliably Identifying Signs of DDOS Flood Attacks Based on LRD Traffic Pattern Recognition. *Computers and Security*, **23**, pp. 549-558.

[20] Xiang Y., Lin Y., Lei W.L. and Huang S.J. (2004) Detecting DDOS attack based on Network Self-Similarity. *IEE Proceedings in Communication*, **151**, pp. 292-295.

[21] Cajueiro D.O. and Tabak B.M. (2004) The Hurst Exponent over Time:Testing the Assertion That Emerging Markets Are Becoming More Efficient. *Physica A*, **336**, pp. 521-537.

[22] Siris V. and Papagalou F. (2004) Application of anomaly detection algorithms for detecting SYN flooding attacks. *Proceedings of IEEE GLOBECOM '04*, pp. 2050-2054.

[23] Hussain A., Heidemann J. and Papadopoulos C.(2004) Distinguishing between single and multi-source attacks using signal processing. *Computer Networks*, **46**, pp. 479-503.

[24] Gelenbe E., Lent R. and Nunez A. (2004) Self-aware networks and QoS. *Proceedings IEEE*, **92(9)**, pp. 14781489.

[25] Gelenbe E., Gellman M., Lent R., Liu P. and Su P. (2004) Autonomous smart routing for network QoS. *International Conference on Autonomic Computing.*

[26] Gavrilis D., Tsoulos I. and Dermatas E. (2004) Feature Selection for Robust Detection of Distributed Denial-of-Service Attacks Using Genetic Algorithms. *Lecture Notes in Artificial Intelligence*, **3025**, pp. 276-281.

[27] Gelenbe E., Gellman M. and Loukas G. (2005) An autonomic approach to denial of service defence. *In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 537-541.

[28] Gavrilis D. and Dermatas E. (2005) Real-time detection of distributed denial-of-service attacks using RBF networks and statistical features. *Computer Networks*, **48**, pp. 235-245.

[29] Jalili R., Imani-Mehr F., Amini M. and Shahriari H.R. (2005) Detection of Distributed Denial of Service Attacks Using Statistical Preprocessor and Unsupervised Neural Networks. *Lecture Notes in Computer Science*, **3439**, pp. 192-203.

[30] He H.T., Luo X.N. and Liu B.L. (2005) Detecting Anomalous Network Traffic with Combined Fuzzy-Based Approaches. *Lecture Notes in Computer Science*, **3645**, pp. 433-442.

[31] Li L. and Lee G. (2005) DDoS Attack Detection and Wavelets. *Telecommunication Systems*, **28:3(4)**, pp. 435-451.

[32] Furuya T., Matsuzaki T. and Matsuura K. (2005) Detection of Unknown DoS Attacks by Kolmogorov-Complexity Fluctuation. *Lecture Notes in Computer Science*, **3822**, pp. 395-406.

[33] Wei W., Dong Y., Lu D. and Jin G. (2006) Combining Cross-Correlation and Fuzzy Classification to Detect Distributed Denial-of-Service Attacks. *Lecture Notes in Computer Science*, **3994**, pp. 57-64.

[34] Li M. (2006) Change Trend of Averaged Hurst Parameter of Traffic under DDOS Flood Attacks *Computers and Security*, **25**, pp. 213-220.

[35] Teke A. and Atalay V. (2006) Texture Classification and Retrieval Using the Random Neural Network Model. *Computational Management Science*, **3(3)**, pp. 193-205.

[36] Chen Y. and Hwang K. (2006) Collaborative detection and filtering of shrew DDoS attacks using spectral analysis. *Parallel Distrib. Comput.*, **66**, pp. 1137-1151.

[37] Gelenbe E. and Loukas G. (2007) Self-Aware Approach to Denial of Service Defence. *Computer Networks*, 51, pp.1299-1314.

[38] UCLA CSD packet traces: http://www.lasr.cs.ucla.edu/ddos/traces /public/usc/.

[39] SecurityFocus, August 2004: FBI busts alleged DDoS Mafia, http://www.securityfocus.com /news/9411.

[40] BBC, September 2001: Teenager cleared of hacking, http://news.bbc.co.uk/1/hi/england /hampshire/dorset/3197446.stm.

[41] WiredNews: Pentagon Hacker Exposed by Just. Dpt, http://www.wired.com/news/technology/0,1282,11030,00.html.

[42] H. Abdelbaki: Matlab simulator for the RNN, http://www/cs/ucf.edu/ ahossam/rnnsim.

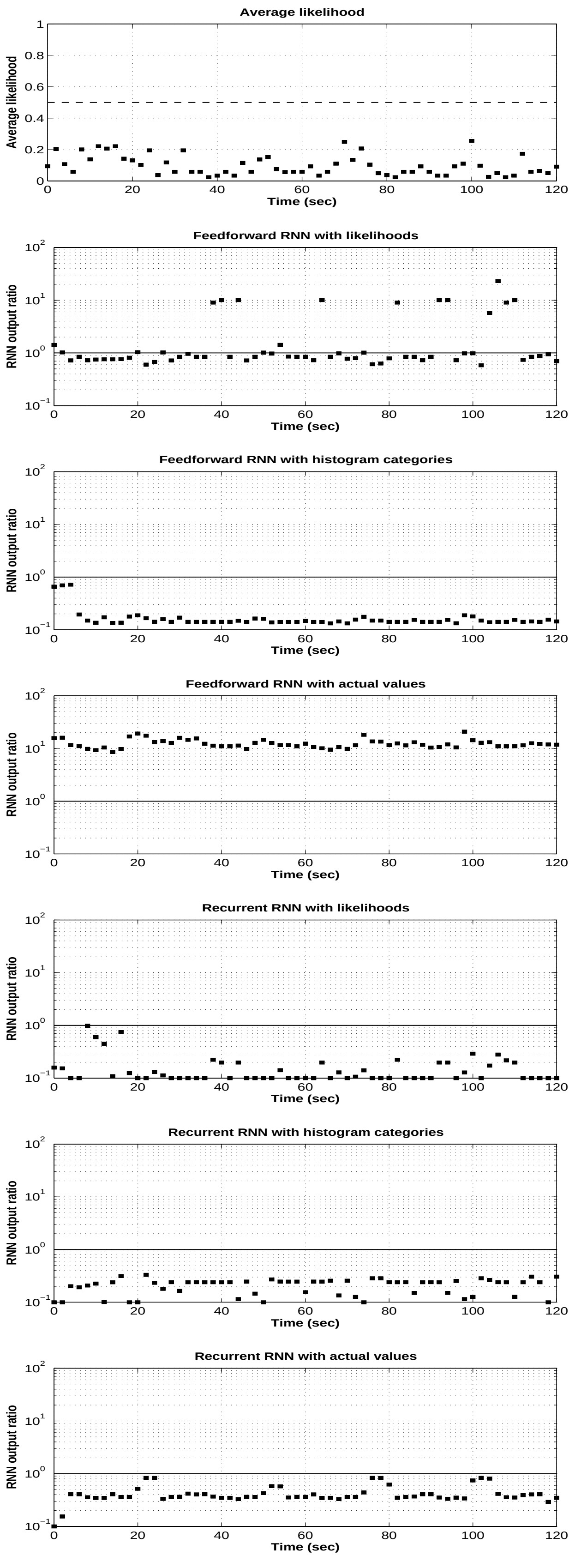

Figure 4: Detection results for Dataset0 (normal)

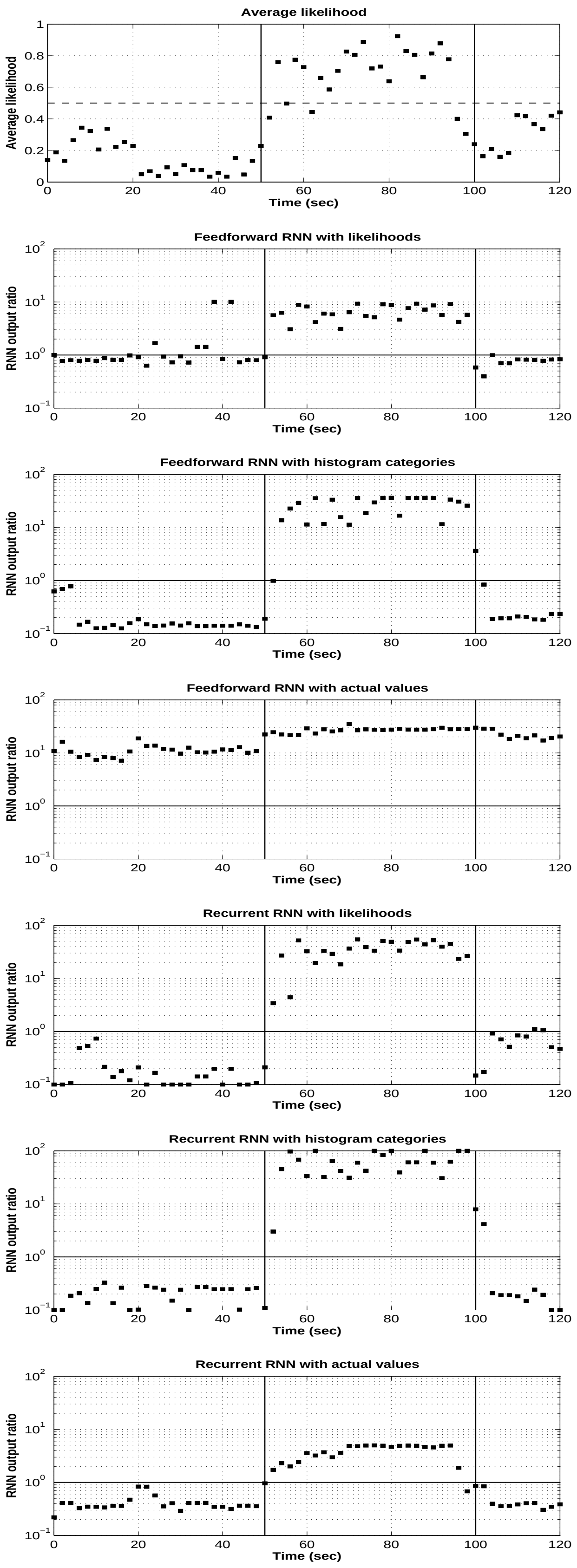

Figure 5: Detection results for Dataset1 (attack)

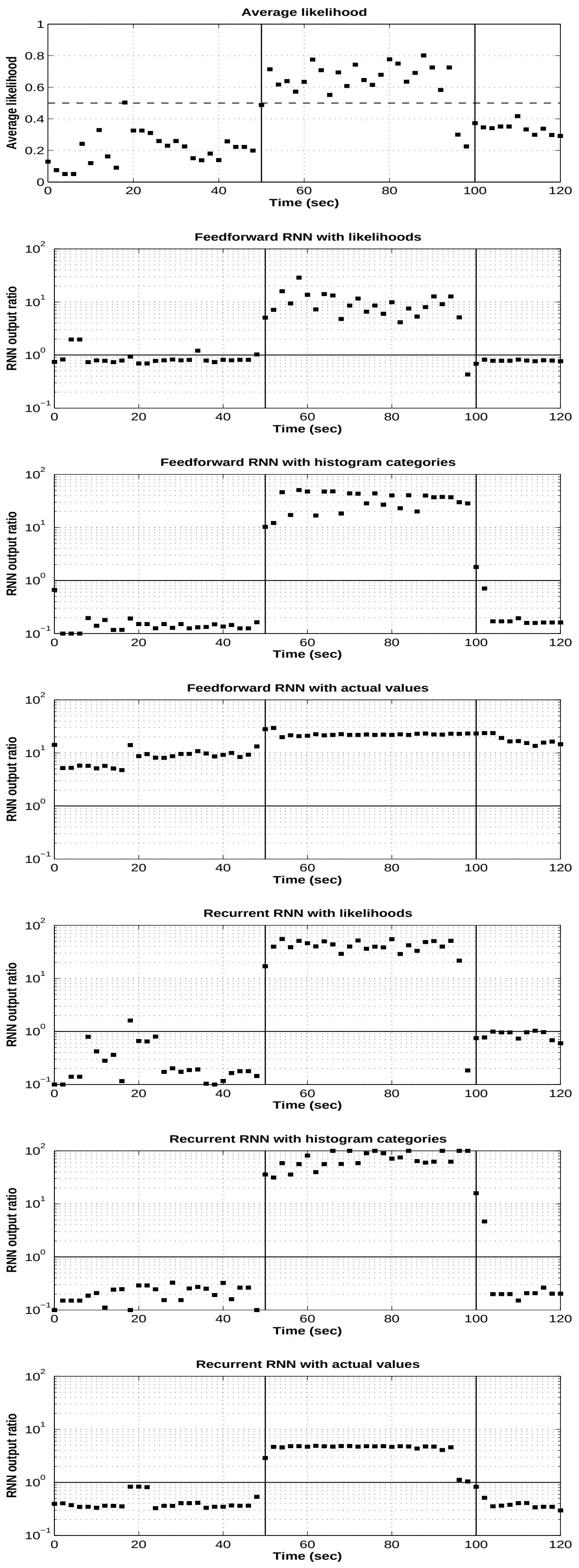

Figure 6: Detection results for Dataset2 (attack)

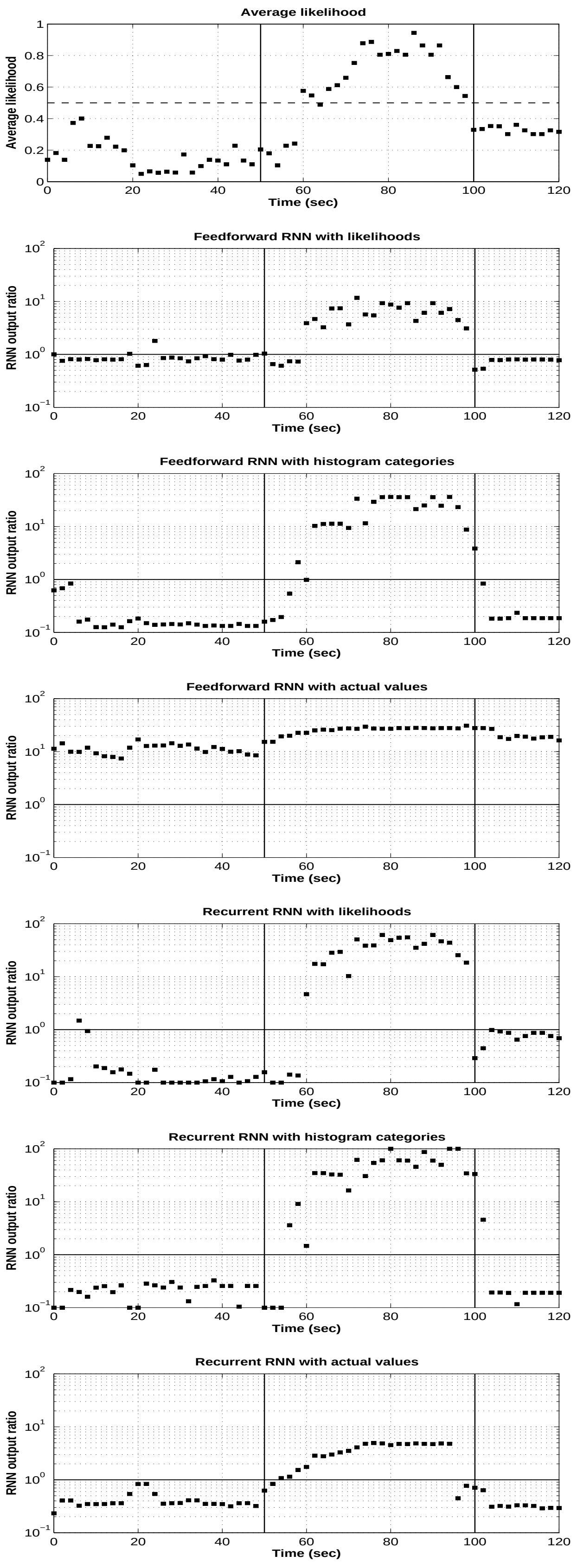

Figure 7: Detection results for Dataset3 (attack)

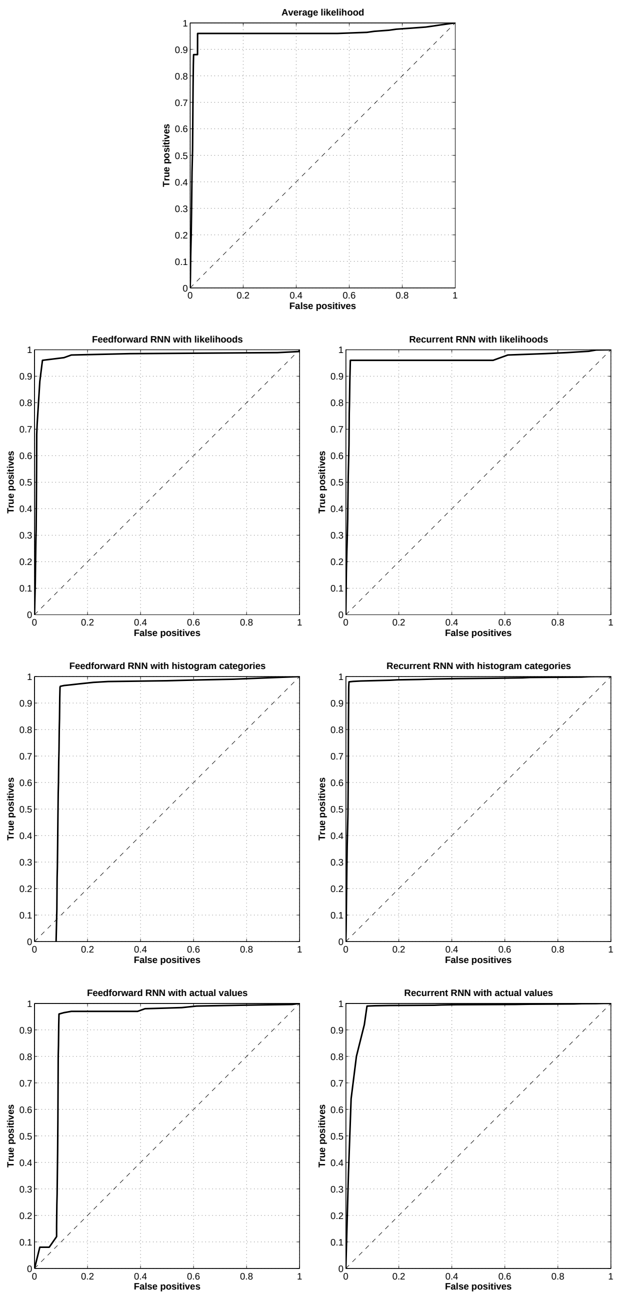

Figure 8: ROC curves for Dataset2 (attack)