Neurocomputing 172 (2016) 385–393

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Detection of known and unknown DDoS attacks using Artificial Neural Networks

Alan Saied, Richard E. Overill, Tomasz Radzik

*Department of Informatics, King's College London, Strand, WC2R 2LS, UK*

ARTICLE INFO

*Article history:*
Received 13 September 2014
Received in revised form
2 April 2015
Accepted 11 April 2015
Available online 8 August 2015

*Keywords:*
DDoS attacks
DDoS detectors
Genuine and DDoS patterns

ABSTRACT

The key objective of a Distributed Denial of Service (DDoS) attack is to compile multiple systems across the Internet with infected zombies/agents and form botnets of networks. Such zombies are designed to attack a particular target or network with different types of packets. The infected systems are remotely controlled either by an attacker or by self-installed Trojans (e.g. roj/Flood-IM) that are programmed to launch packet floods. Within this context, the purpose of this paper is to detect and mitigate known and unknown DDoS attacks in real time environments. We have chosen an Artificial Neural Network (ANN) algorithm to detect DDoS attacks based on specific characteristic features (patterns) that separate DDoS attack traffic from genuine traffic.



## 1. Introduction and background

The key objective of a Distributed Denial of Service Attack (DDoS) attack is to compile multiple systems across the Internet with infected zombies/agents [1] and form networks of botnets. Such zombies are designed to attack a particular target or network with different types of packets. The infected systems are remotely controlled either by an attacker or by self-installed Trojans (e.g. roj/Flood-IM) [2] that are programmed to launch packet floods. The authors of [3] have explained different DDoS architectural structures used by DDoS engineers (i.e. hackers) to launch successful attacks. DDoS attacks are serious security issues that cost organisations and individuals a great deal of time, money and reputation, yet they do not usually result in the compromise of either credentials or data loss. They can damage one or a group of devices and their resources. A DDoS attack slows or halts communications between devices as well as the victim machine itself. It introduces loss of Internet services like email, online applications or programme performance.

Within this context, the purpose of this study is to detect and mitigate known and unknown DDoS attacks in real time environments. In our context, DDoS attacks that are not detected by existing available detection solutions are called unknown (zero-day) attacks. We have chosen an Artificial Neural Network (ANN) algorithm [4] to detect DDoS attacks based on specific characteristic features (patterns) that separate DDoS attack traffic from genuine traffic. We have intensively trained the algorithm with real life cases and DDoS attacking scenarios (patterns) that are produced using the existing popular DDoS tools. We observed that the more we trained the algorithm with up-to-date patterns (latest known DDoS attacks), the further we increased the chances of detecting known and unknown DDoS attacks, while considering that over training and repetitive patterns are avoided. This is because the ANN algorithm learns from scenarios and detects zero-day patterns that are similar to what it was trained with. In its yearly reports (2010–2014), Prolexic, the world´s largest DDoS mitigation service, has stated that TCP, UDP and ICMP are the most used protocols to launch DDoS attacks [5].

Our primary aim is to combine detection of known and unknown DDoS attacks followed by a defence mechanism that prevents forged packets from reaching the victim, but allows genuine packets to pass through. Furthermore, we aim to observe ANN's behaviour towards unknown DDoS detection when trained with old and up-to-date datasets[1]. The objectives of our work can be summarised by the following points:

- Detect known and unknown DDoS attacks (high and low rates) in real time as opposed to only detect known attacks.
- Identify high volume of genuine traffic as genuine without being dropped.
- Prevent DDoS attacking (forged) packets from reaching the target while allowing genuine packets to get through.

*E-mail addresses:* alan.saied@kcl.ac.uk (A. Saied), richard.overill@kcl.ac.uk (R.E. Overill), tomasz.radzik@kcl.ac.uk (T. Radzik).

[1] Old datasets are datasets with old known DDoS attack patterns while up-to-date datasets are datasets with latest known DDoS attack patterns.

http://dx.doi.org/10.1016/j.neucom.2015.04.101
0925-2312/

- Train, deploy and test the solution in a physical environment as opposed to simulators.
- Reduce the strength of the attack before it reaches the victim as opposed to nearby detection systems.
- Evaluate our approach using both old and up-to-date datasets with related work, based on accuracy, sensitivity, specificity and precision.

This paper is organised as follows: Section 2 critically reviews approaches related to this work. Section 3 explains the theoretical framework and architectural design of our approach. Section 4 explains the testing process. Section 5 evaluates our solution in comparison with other related work. Section 6, we identify the limitations and further research questions related to our approach and the final Section 7, we present our summary and conclusions.

## 2. Related work

Various methodologies and techniques for reducing the effects of DDoS attacks in different network environments have been proposed and evaluated. Jie-Hao and Ming [27] have used ANN to detect DDoS attacks where they compared the detection outcome with decision tree, ANN, entropy and Bayesian. The authors identified users' requests or demands to a specific resource and their communicative data. Then samples of such requests are sent to the detection systems to be judged for abnormalities. Also, Liu and Gu have used Learning Vector Quantisation (LVQ) neural networks to detect attacks [6]. This is a supervised version of quantisation, which can be used for pattern recognition, multi-class classification and data compression tasks. The datasets used in the experiments were converted into numerical form and given as inputs to the neural network. Akilandeswari and Shalinie [7] have introduced a Probabilistic Neural Network Based Attack Traffic Classification to detect different DDoS attacks. However, the authors focus on separating Flash Crowd Event from Denial of Service Attacks. As part of their research, they have used Bayes decision rule for Bayes inferences coupled with Radial Basis Function Neural Network (RBFNN) for classifying DDoS attack traffic and normal traffic. Siaterlis and Maglaris [8] have experimented with single sets of network characteristics to detect attacks. They use Multi-Layer Perceptron (MLP) as a data fusion algorithm where the inputs are metrics coming from different passive measurements that are available in a network and then coupled this with traffic that was generated by the researchers themselves. Gupta, Joshi and Misra [9] have used a neural network to detect the number of zombies that have been involved in DDoS attacks. The objective of their work is to identify the relationship between the zombies and in sample entropy. The process workload is based on prediction using a feed-forward neural network.

Another line of research is to use infrastructure for detecting and mitigating DDoS attacks. Badishi, Keidar and Yachin [10] have used authentication and cryptographic approaches to protect network services from DDoS attacks. A similar approach has been introduced by Shi, Stoica and Anderson [11], but a puzzling mechanism is used instead to detect DDoS attacks before reaching the target. Hwang and Ku [12] have developed a distributed mechanism to combat DDoS. Their architecture, called Distributed Change-point Detection (DCD), is designed to reduce DDoS attacks. They adopt the non-parametric CUSUM (Cumulative Sum) algorithm to describe any changes in the network traffic.

Some previous research has focused on the source of the attack for the purpose of detection. The authors of [13–15] have used packet-marking mechanisms and entropy to identify the source of the packet considering that each packet is marked on each router that it passes through.

Some of these approaches use ANN or infrastructure to detect known DDoS attacks while others focus on the source of the attack (traceability). However, these researchers have not covered unknown (zero-day) high and/or low rate DDoS attack detection in their approaches. Detecting unknown DDoS attacks is one of the vital objectives that distinguish our work from theirs.

## 3. Theoretical framework and architectural design

The strength of the attack can be minimised if multiple effective DDoS detectors are deployed across the network. These detectors analyse the network for abnormalities and prevent them from reaching the victim when detected. It is important to allow genuine traffic flows to pass through the detectors and reach their destinations. Thus, it is vital for the detection process to be accurate and tested against all possible existing use-cases and patterns. Due to ease of implementation, practicality and online documentations of TCP, UDP and ICMP protocols [16], most DDoS designers manipulate such protocols to launch their attacks as explained in the Prolexic yearly reports [5]. Our detection mechanism is based on a supervised ANN (Feed-forward, Error Back-Propagation with a Sigmoid activation function [4]) where its accuracy primarily relies on how well the algorithm is trained with relevant datasets.

The patterns used for training purposes are instances of packet headers, which include source addresses, ID and sequence numbers coupled with source destination port numbers. Based on our experiments and analysis, most installed zombies use their built-in libraries as opposed to operating system libraries to generate packets. This is to assist the attacker manipulate and forge the packets during the attack and introduce effectiveness. Therefore, one can study the characteristic features of genuine packets that are generated by genuine applications and compare them with forged packets that are generated by the DDoS attacking tools and present them as input variables to train the ANN. Selecting the patterns to be our inputs began by building new network infrastructures in corporate and isolated environments with different types of DDoS attacks launched at different levels (high and low rates). The results were carefully studied, compared and cross-matched with genuine traffic to verify the characteristic patterns that separate genuine from attack traffic. This part of the process required intensive understanding of how different protocols exchange and communicate with each other. The datasets are organised and structured to accommodate genuine and attack patterns in a qualified format that Java Neural Network Simulator (JNNS) [17] accepts. A total of 80% of each dataset is used to train the algorithm and 20% to validate the learning process. In our work we did not use other online datasets in order to train our approach as we wanted to learn about the behaviour of the DDoS and the genuine traffic. Furthermore, offline datasets cannot provide such real time of DDoS behaviour. However, one can train our approach using other available online datasets and compare the outcome with our outcome to identify strengths and weaknesses. Before training the algorithm using JNNS, the input values are normalised to maximise the performance in sensitive applications like ours where accurate detection is vital. If the input values are not normalised and applied directly, then large values may lead to suppressing the influence of smaller values [18]. Jayalakshmi, Santhakumaran, Zhang and Sun have also explained the positive effect of normalisation on ANN performance and training process [19,20].

A typical ANN consists of input, hidden and output layers where the patterns are fed to the learning algorithm via the input nodes. The input values represent the characteristic patterns (variables) that separate attacks from genuine traffic. We selected

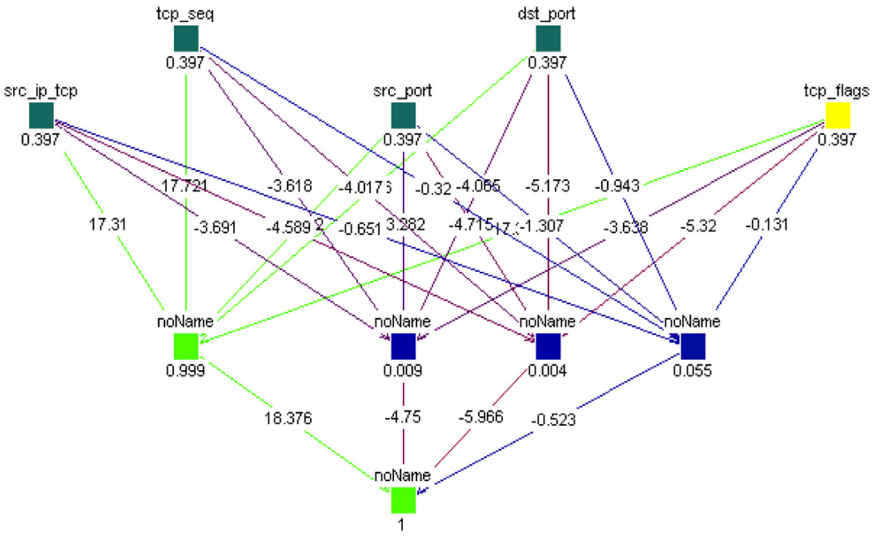


**Fig. 1.** ANN TCP topological structure.

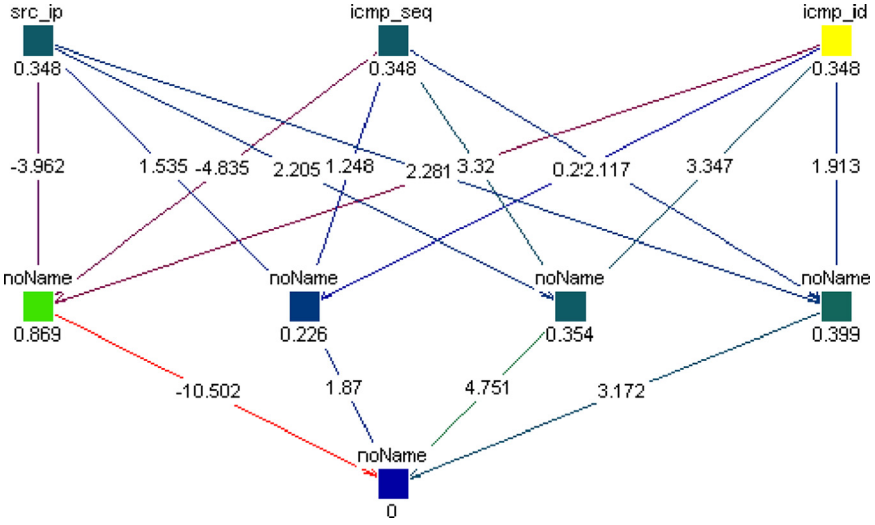


**Fig. 2.** ANN ICMP topological structure.

three topological ANN structures, each with three layers (input, hidden and output layers). The number of nodes in each topological structure is different – for example, the ANN ICMP topological structure consists of three inputs and four hidden nodes, the ANN TCP topological structure consists of five input and four hidden nodes, and the ANN UDP topological structure consists of four input and three hidden nodes. Hidden nodes deal with the computation process with respect to input and output nodes.

The output layer consists of one node to represent 1 (attack) or 0 (genuine traffic). Figs. 1, 2 and 3 respectively represent TCP, ICMP and UDP ANN topological structures. Choosing a relevant learning algorithm or the number of hidden nodes and activation function was based on initial experiments where Sigmoid activation function and Back-Propagation learning provided most accurate results. The comparison was made between QuickProp, Back-Propagation, Backprop Weight Decay, Backprop through Time, while Sigmoid, Elliott, SoftMax, BAM were used as activation functions [4,18,19]. Our experiments showed that Back-Propagation learning coupled with a Sigmoid activation function and the chosen topological structures in Figs. 1–3 can produce a 98% detection accuracy. This is by far, the highest detection accuracy in comparison with other related learning algorithms and activation functions. The detection accuracy is calculated as

$$\text{Accuracy} = (\text{Truepositives} + \text{True Negatives})/(\text{True positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}) \times 100$$

In Fig. 1, the input layer of TCP topological structure consists of five nodes that accommodate TCP sequence, TCP flags, TCP source and destination port numbers and source IP addresses. Fig. 2 represents an ICMP topological design where the input nodes are source IP addresses, ICMP sequence number and ICMP-ID. While Fig. 3 shows the UDP topological structure where the input values are source IP addresses, packet length, and UDP source and destination port.

The numbers between the nodes represent the weight that is used by Back-Propagation to adjust and learn by example (patterns). The more new examples are provided, the better it would be in identifying unknown attacks considering that repetitive patterns are avoided as they produce biased or inaccurate results. The algorithm does this by changing the network weights between the layers until the closest desired output is obtained (0 or 1). The nodes between the layers are connected via Feed-Forward links.

When forged packets are detected by our solution, its defence mechanism is activated to drop the forged packets while allowing

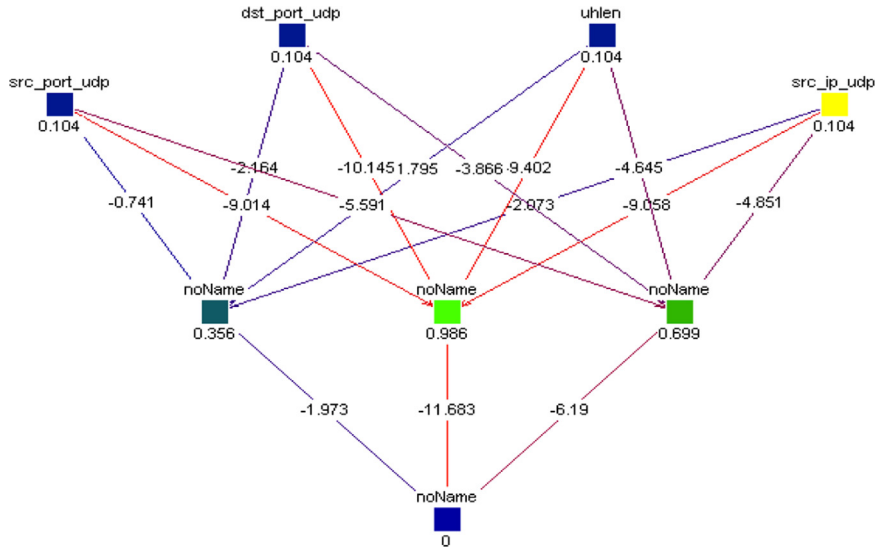


**Fig. 3.** ANN UDP topological structure.

genuine packets to pass through. Blocked packets are unblocked as soon as the system flags the traffic as normal. However, genuine traffic flowing to the target machine will not be disturbed as it flagged to be genuine by our proposed solution. Furthermore, the detectors communicate with each other via encrypted messages to provide assistance when required. Such exchange of information between the detectors helps security officers identify abnormal behaviour and further deploy countermeasures if required.

Our solution is designed to continuously monitor the network for abnormal behaviour by retrieving packets from the network and analysing their header information using the trained ANN. However, retrieving a large number of packets in a busy network requires high processing rates and is expensive. Therefore, we have introduced individual packet thresholds for each protocol. If the number of packets in a given network is greater than a specific threshold per protocol, the retrieved packets are subjected to investigation. Choosing the right threshold per protocol was based on real time experiments by calculating the number of packets per unit time (using IPTraf [21]) in different network environments, where the threshold values are configurable. Once the packets are separated and prepared for examination, our proposed solution pipes the patterns (variables) into ANN codes to decide upon the legitimacy of the retrieved traffic (packets). Our design is illustrated in Fig. 4, where each network is installed with one DDoS detector that communicates with others via encrypted messages.

Fig. 4 can be summarised as follows

1. DDoS detectors are installed on different networks.
2. Each detector registers the IP address of all neighbouring DDoS detectors to inform and send encrypted message when DDoS attacks are detected.
3. Detectors continuously monitor their networks for abnormalities.
4. Abnormalities are flagged when the number of passing packets is greater than a predefined threshold for each protocol.
5. If the number of packets is greater than the threshold, then:
   a. The organiser sorts the packets accordingly.
   b. The IP identifier identifies the victim's IP addresses.
   c. The ANN calculator calculates retrieved patterns and prepares them for the ANN engine
   d. The trained ANN engine takes the patterns as inputs and produces one output (1=attack or 0=normal).
   e. Before activating the defence system, the steps under point 5 above are repeated twice more, producing a total of 3 outputs.
6. The defence system receives the outputs from the detection component and:
   a. If the outputs are (0,0,0), then no action is required by the defence system, as the traffic is clean.
   b. If the outputs are (1,1,1), (1,1,0), (1,0,1) or (0,1,1), it activates defence and stops the attack while allowing genuine traffic to pass through.
   c. If the outputs are (1,0,0), (0,1,0) or (0,0,1), the solution repeats point 5. If the outputs of the new retrieved traffic are:
      I. (1,1,1), (1,1,0), (1,0,1) or (0,1,1), activate the defence system.
      II. (1,0,0), (0,1,0) or (0,0,1), deploy mitigation as this is considered to be a low rate attack.
      III. (0,0,0), no actions are required.
   d. However, if the output is none of the above, then the system generates value 2. This means the traffic is unidentified (not used in training) by the ANN. At this point, the solution checks its local record to learn if the same traffic is received and detected by neighbouring DDoS detectors. If the received traffic from the neighbouring detectors is 1 or 0 then the algorithm is out-dated since its detection was 2 while other detectors identified the traffic as 1 or 0. This means the algorithm on the local DDoS detector needs to be retrained (off-line) with both existing and new patterns. Otherwise no action is taken.
7. The knowledge sharing component sends encrypted messages containing the type of the attack; destination IP address and protocol involved to all registered neighbouring DDoS detectors. Such information is also composed by the email element and sent to the security offices to take countermeasures if required or for the purpose of logistics or forensics.

The output of the detection process is 2 when the algorithm is trained only with old datasets and thus lacks new patterns. The ANN has the ability to detect unknown patterns if the attack is similar to what the algorithm was trained with. However, our experiments show that the algorithm fails to detect unknown patterns if trained with old datasets only. We identified that the ANN trained with up-to-date

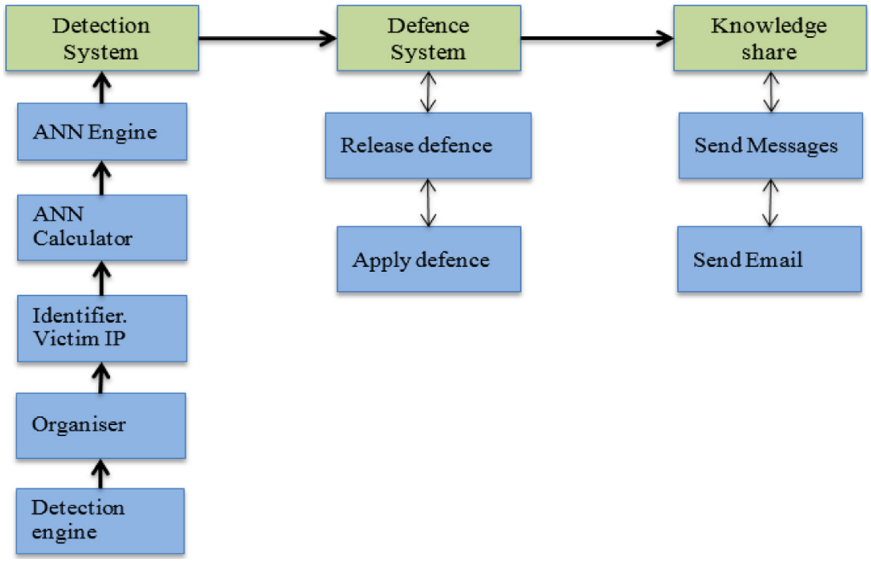


**Fig. 4.** Detection, defence and cooperative mechanism.

patterns can detect all known and most unknown attacks while the ANN trained with old patterns fails to do so. At this point the ANN of the DDoS detector that was previously trained with old datasets and fails to detect some known DDoS attacks must be retrained. The training process must be done offline as this is a supervised process and new patterns must be introduced in addition to the existing ones. Therefore, sharing knowledge between the detectors can provide extra assistance to make further decisions when the ANN is not up-to-date.

Meanwhile, each detector sends a composed email to the security officer with a complete report of all detected DDoS attacks. One may compare this approach with having one common central server to collect all the attacks and sending all as one email. However, if the central point is down, then information cannot be sent to the security officer. Consequently no extra countermeasures are deployed when required. Each DDoS detector is designed to function as a standalone component or distributed detector that sends encrypted messages to many registered detectors located in different networks. The solution is not restricted to a limited number of detectors to send or receive encrypted messages. Therefore, if one DDoS detector is not functional, other detectors still receive and send messages making the overall solution resilient and immune to individual DDoS detector or system failure.

The aims, design and implementation processes of our work can be summarised as follows:

1. The purpose of this study is to detect and mitigate known and unknown DDoS attacks before they reach the victim.
2. We selected DDoS attacks due to the deficiencies in existing approaches in comparison with other security domains, to detect known and unknown DDoS attacks and the ability of DDoS attackers to crash or overload a destination.
3. We only selected TCP, UDP and ICMP DDoS attacks due to their popularity among DDoS attackers, but other types of DDoS attacks also warrant research.
4. To define an approach that detects known and unknown DDoS attacks we:
   a. Studied and learned how DDoS attackers built their approaches through testing available DDoS attack methodologies.
   b. Reviewed related academic and industrial DDoS detection mechanisms where applicable.
5. We learned about the existing approaches and analysed them according to environments, algorithms, methodologies, and detection accuracy.
6. We built physical environments to test and analyse forged packets generated by the DDoS methodologies and genuine packets that are generated from genuine applications.
7. From point 6, we learned that DDoS designers use their own custom code as opposed to operating system resources to generate packets. This allows DDoS attackers to have better control over the packet type, which makes it more effective.
8. To extend our knowledge, we began by learning and investigating background information and fundamental concepts of all related technologies that are involved in DDoS attacks.
9. From points 7 and 8, we identified that DDoS attackers change specific protocol header patterns to confuse the detection system or indeed the destination. Such an approach assists DDoS attacks to look genuine and, hence, bypass detection systems. Therefore, we focused on protocol headers to detect DDoS attacks.
10. From points 6 to 9, we selected specific patterns that separate genuine traffic from DDoS attacks.
11. We employed an ANN to detect known and unknown DDoS attacks based on patterns identified from point 10. ANN was selected due to its ability to detect known and unknown patterns that are similar to those it was trained with.
12. Most traditional approaches use volume limitation and signature based detection systems to control their traffic. In signature based detection systems, an administrator is required to include rules and signatures (database) to detect old and known attacks. Our approach uses the ANN algorithm to detect known and unknown DDoS attacks. Therefore, no administration is required and any unknown (zero-day) attacks that are similar to the DDoS attacks and used to train ANN are detected. A volume limitation approach is used when the volume is higher than a certain threshold. This normally results in both genuine and DDoS traffic volumes being dropped. Our approach, drops DDoS attacks based on the detection components output.
13. The ANN needs to be trained with different datasets that represent patterns mentioned in point 10.
14. To train the ANN, one can either use existing old datasets or generate an up-to-date database that contains the most recent patterns. We selected up-to-date datasets that cover old and

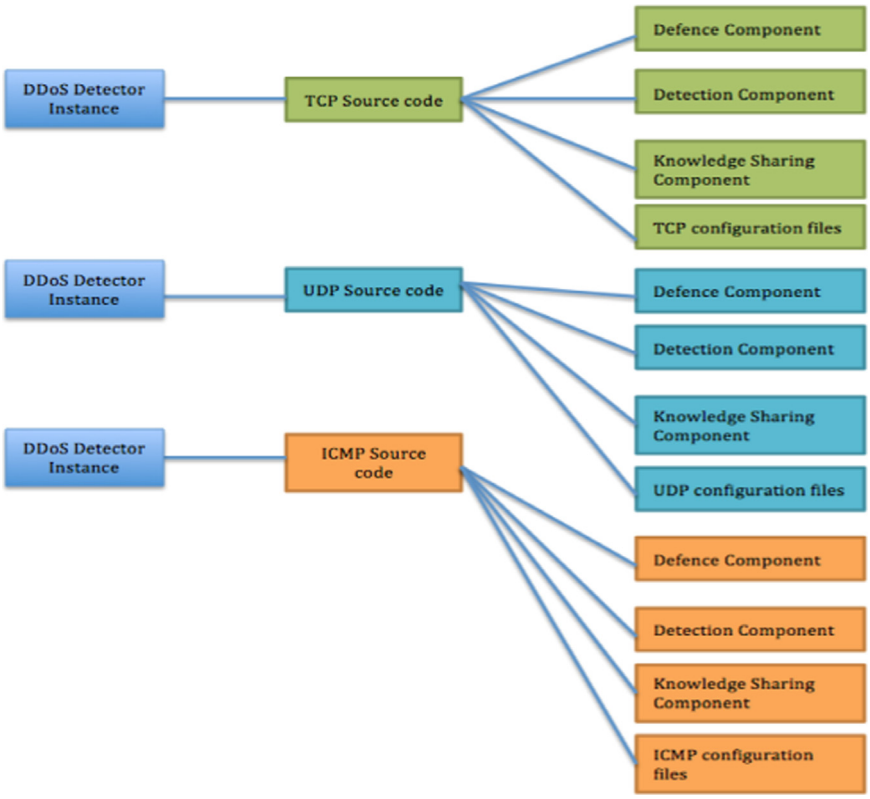


**Fig. 5.** Representations of three DDoS detector instances.

new patterns. However, we trained ANN with old patterns just to identify the ANN´s response when detecting known and unknown DDoS attacks.

15. To generate datasets, we built realistic corporate safe environments where we launched different TCP, UDP and ICMP DDoS attacks coupled with genuine traffic that was generated by genuine applications.
16. The datasets were prepared in the format that JNNS accepts to train our ANN algorithm (off-line). We selected 80% of the datasets to train the learning algorithm and 20% to verify the training process.
17. The process of design and implementation of our solution can be summarised in the following points:
    a. Retrieve packets from the network based on thresholds.
    b. Organise and calculate the packets for the ANN engine to verify the legitimacy of the retrieved packets.
    c. The output is either 1=attack, 0=normal or 2=unidentified traffic.
    d. If the output is 1, the defence component activates and drops the forged packets, but if the output is 0, then no action is required. However, if the output is 2, the defence component relies on the output that it receives from other DDoS detectors.
    e. Destination and type of the attack are distributed to all other DDoS detectors for the purpose of awareness.
    f. The proposed system loops back and retrieves packets from the network for further monitoring. The defence component removes restrictions if the traffic to the same destination is verified as genuine by the ANN engine. In all cases, genuine traffic is untouched and unobstructed.
18. To rigorously and realistically test our solution, we used physical corporate environments to evaluate our approach against known and unknown DDoS attacks. We used detection accuracy, sensitivity and specificity to evaluate our approach vis-à-vis other approaches.
19. To implement our solution mechanism, we developed our detection system as plugins and integrated it with Snort-AI [22]. Snort-AI is based on the Snort signature Intrusion Detection System project [23] – one of the present authors (AS) is an active contributor to Snort-AI – by providing plugins and other integration processes. The output of the detection system is coupled with the destination IP address to instruct iptables [24] to mitigate forged packets while allowing genuine traffic to pass through. We also used the RSA public key encryption mechanism to encrypt the messages over the TCP connections between the detectors where each detector acts as both a sender and a receiver.

The solution needs to be functional when parts of its code are subject to technical issues or maintenance. For example, if the TCP detection code requires an update, ICMP and UDP detection codes should be able to function without any downtime. In addition, our solution must be organised in terms of logging traffic for the purpose of debugging if required. One can either embed all three TCP, UDP and ICMP detection, mitigation and knowledge sharing codes in one application or separate them as instances. Programming has shown that embedding all three source codes into one requires more time to implement and more testing, while separating and creating instances of TCP, UDP and ICMP yield the following benefits:

- Turn off any instance when needed without affecting other instances.
- Separate the detection mechanisms according to protocols (better control).
- A crashed instance due to technical issues will not affect the other instances.
- Easily debug to identify technical issues/problems.
- Avoid a single point of failure (protocol based).
- Separate crashes according to protocols (if applicable).

**Table 1**
Test results that reflects the requirements and the objectives.

| Requirements | Testing approach | Expected results | Results | Comments |
|---|---|---|---|---|
| Unknown (zero-day) and known detection when high and low rate DDoS attacks are launched. | Integration | All modules of detection component are integrated together. | **Passed** All modules are integrated to detect DDoS attacks. | The integrated modules such as organiser, victim IP-identifier, calculator and ANN engine. |
| | Functional | Detect normal traffic and known/ unknown DDoS attacks. | **Passed** 0 – Normal 1 – Attack 2 – Unidentified | Attacks are launched at low and high rates. |
| | System | Our DDoS detector is capable of detecting **ALL** DDoS attacks. | **Failed** | Not **ALL** DDoS attacks are detected. See conclusion. |
| Detect and separate genuine traffic that is look-a-like attack traffic (e.g. a high load of genuine traffic towards a particular website). | Functional | Detect genuine traffic that looks like a DDoS attacks. | **Passed** 0 – Normal | N/A |
| | System | Our DDoS detector is capable of detecting **ALL** genuine traffic. | **Passed** 0 – Normal | N/A |
| | Load | Detects high and low load genuine traffic. | **Passed** 0 – Normal | N/A |
| Ability to detect, cope with and defend against DDoS attacks if the DDoS detectors themselves are under DDoS attack (avoid crashing). | Load/system | Detect DDoS attacks while our DDoS detector itself is under attack. | **Passed** 1 – Attack 0 – Normal 2 – Unidentified | Detect attack while both our DDoS detector system and other destinations are under DDoS attacks. |
| Mitigate against DDoS attacks when detected and allow genuine traffic to pass through. | Integration | All modules of defence component are integrated together with detection component. | **Passed** | Detection and defence component are integrated together. |
| | Functional-1 | Regardless of type or strength of the attack. Victim must be defended from DDoS attacks. | **Passed** | N/A |
| | Funcational-2 | Output of detection component determines the activation of the defence component against any type of DDoS attack. | **Passed** | 0 – No action is required by the defence.<br>1. Take action.<br>2. Use the information from other DDoS detectors to take action. |
| | System | A DDoS detector detects and mitigates DDoS attacks. | **Passed** | N/A |
| Communication between the DDoS detectors on different networks via encrypted connections. Also send an email to the Security Officer for extra countermeasures if required. | Integration | All components and elements are integrated. | **Passed** | N/A |
| | Functional-1 | Send encrypted messages to other DDoS detectors. | **Passed** | N/A |
| | Functional-2 | Send an email to Security Officer. | **Passed** | N/A |
| | System | Each DDoS detector, detects and mitigates DDoS attacks and shares the information with other DDoS detectors. | **Passed** | All modules & components of a DDoS detector are involved in this test. |
| Minimise the strength of DDoS attacks before they reach their destination | System | Strength of the attack is minimised after mitigation. | **Passed** | N/A |
| Detect high and low rate DDoS attacks. | Functioinal-1 | Detect high rate DDoS attacks. | **Passed** | NA |
| | Functional-2 | Detect low rate DDoS attacks. | **Passed** | NA |

Based on the above points, our approach provides scalability, resilience and avoids single points of failure (see Fig. 5).

Although the approach shown in Fig. 5 is scalable, controllable and avoids a single point of failure, it requires more physical resources to operate. We, prioritise avoiding a single point of failure over physical resources as this can be resolved by upgrading the hardware specifications of the machine (CPU and RAM).

## 4. Testing process

In order to verify whether the implemented solution meets the functional, performance, design and implementation requirements we have conducted different tests which reflect the objectives and the requirements of this work. For this purpose, we have used Integration, Functional, System and Load tests, as shown in Table 1:

## 5. Quantitative evaluation and contribution

We have evaluated our solution based on accuracy, precision, sensitivity (ability to identify positive results) and specificity (ability to identify negative results). The tests were conducted in a controlled, isolated network environment where genuine and DDoS attack flows (high and low rates) are deployed. During the experiments, known and unknown DDoS attacks were launched each with 20–120 zombies forming a total of 1160 known and unknown DDoS

**Table 2**
Comparison between our approach and other work.

| Approach | Accuracy (%) | Sensitivity (%) | Specificity (%) | Precision (%) |
|---|---|---|---|---|
| Our approach | 98 | 96 | 100 | 100 |
| Snort | 93 | 90 | 97 | 96 |
| PNN [7] | 92 (P1); 97 (P2) | NA | NA | NA |
| BP [6] | 90 (BP) | NA | NA | NA |
| Chi-square [25] | 94 | 92 | NA | NA |
| K-PCA-PSO-SVM [26] | NA | 96 | NA | NA |

attacks. The zombies were installed and attacked from virtual platforms on VMware boxes where the boxes were connected to physical devices (victim) via virtual routers. Then, the DDoS detectors/Snorts were deployed between the virtual routers and the victim where they analysed the traffic for abnormalities. The results are presented in Table 2 and compared with a signature-based solution (Snort) and four other approaches [6,7,25,26] for which quantitative evaluations are reported.

Afterwards, we trained our solution with old and up-to-date datasets and deployed different DDoS attacks (known/unknown). The results of the experiment are recorded in Table 3.

As a result, the following can be identified as our main contributions:

- Based on our test experiments, on average our approach provides a significantly higher rate of detection, accuracy, sensitivity and specificity compared to the alternative approaches, as shown in Table 2.
- As shown in Table 3, our approach responded less well when trained with old datasets (92% detection accuracy), but when trained with up-to-date datasets, the solution produced a 98% detection accuracy. This means that the more up-to-date attack patterns we use to train the ANN, the better the solution responds in detecting DDoS attacks.
- To further analyse our contribution, we launched 580 known and 580 unknown DDoS attacks. On average our approach detected 95% of unknown DDoS attacks (552 attacks detected) and 100% of known DDoS attacks. This means that our solution failed to detect approximately 5% of unknown DDoS attacks. This is to be expected because, as explained earlier, the ANN only detects attacks that are similar to those it was trained with, and 5% of the attacks were completely different.

We extended our testing and evaluation process by mixing genuine and forged patterns when we launched the DDoS attacks. We then added extra patterns that the trained ANN was not trained with. At this point, our solution detected the DDoS attacks, because our solution is based on existing trained patterns and additional new patterns do not change the nature of the detection process as long as the DDoS attack contains one or more patterns that ANN was trained with. Furthermore, in our experiments we did not experience our solution to flag genuine traffic as DDoS attack and therefore we have a zero measure for the False Positive Rate.

## 6. Limitations and suggestions for further research

Regarding our approach, the following points provide suggestions for further research.

*Unexpected DDoS attack*: Our solution has problems detecting DDoS attacks when the protocol headers are encrypted with any encryption algorithms. This is because our solution is not designed to analyse encrypted packets. However, an encrypted DDoS attack is not a common approach since the attack is slow and introduces high latency.

**Table 3**
Our approach with respect to old and new datasets.

| Our approach | Accuracy (%) | Sensitivity (%) | Specificity (%) | Precision (%) |
|---|---|---|---|---|
| Old datasets | 92 | 88 | 96 | 96 |
| New datasets | 98 | 96 | 100 | 100 |

*Updating the datasets and re-training*: The DDoS attack tools used to generate old datasets date back to the early years between 2000 and 2003, and they are no longer effective. However, the DDoS tools used to generate up-to-date datasets span the period between 2000 and early 2013. With old datasets (attacks that go back to 2000–2003), the detection accuracy was 92%, while with up-to-date datasets (attacks between 2000 and 2013), the detection accuracy was 98%. This means the ANN algorithm requires retraining every 5–6 years. The solution here may be the introduction of an online interactive engine that continuously searches the Internet for new DDoS attacks information. The engine would retrieve the patterns from the Internet and prepare them as datasets to continuously and automatically retrain the ANN whenever required.

*Our approach using other datasets*: One can train our approach using other available online datasets and compare the outcome with our outcome to identify its strengths and weaknesses.

*Our approach in simulated environment*: Our approach has not been tried or tested in a simulated environment. One could reproduce our work in such an environment to verify and compare the detection accuracy of our DDoS detectors in real and simulated environments. The above three points serve as suggestions for future research, to further improve our detection solution, to detect encrypted DDoS attacks using online trainable datasets.

## 7. Summary and conclusions

We have used a trained Artificial Neural Network algorithm to detect TCP; UDP and ICMP DDoS attacks based on characteristic patterns that separate genuine traffic from DDoS attacks. The ANN learning process was started by reproducing a network environment that is a mirror image of a real life environment. Different DDoS attacks were then launched while normal traffic was flowing through the network. The datasets were collected, pre-processed and prepared to train the algorithm using JNNS. The detection mechanism was then integrated with Snort-AI where it was tested against known and unknown DDoS attacks. We evaluated our solution with signature based and other related academic research, and our approach produced higher detection accuracy (98%) in comparison with other approaches shown in Table 2.

We further evaluated our approach by training the algorithm with old and up-to-date datasets (patterns), and it managed to detect known (100%) and unknown (95%) DDoS attacks that are similar to what it was trained with (up-to-date patterns). However, when trained with old data patterns only, our solution did not detect some unknown DDoS attacks. This means that improper training or old patterns can result in poor detection while a variety of DDoS cases can lead to better DDoS detection. This is due to the fact that the algorithm detects on the basis of scenarios; so more scenarios assist the ANN to understand the nature of DDoS attacks. As shown in Table 1 we have also tested our solution based on different requirements that reflects the objectives shown in Section 1. Our solution yielded such results because

- We identified the patterns that are most popular among DDoS attackers to launch an attack.

- We focused on up-to-date dataset that provides higher detection accuracy than old datasets.
- We avoided simulation and used real physical environments in learning about DDoS architectural structure.

A limitation of our solution is that it cannot handle DDoS attacks that use encrypted packet headers. Detecting encrypted DDoS attacks is an interesting and challenging topic for future research.

## References


[1] M. Reed Denial of Service attacks and mitigation techniques: Real time implementation with detailed analysis. [Online] SANS Institute InfoSec Reading Room 2011. Available from: ⟨http://www.sans.org/reading-room/whitepapers/detection⟩.

[2] Troj/Flood-IM. Backdoor DDoS Trojan. Detected by Sophas. Available from: ⟨https://secure2.sophos.com⟩.

[3] E. Alomari, B.B. Gupta, S. Karuppayah, Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art, Int. J. Comput. Appl. 49 (7) (2012) 24–32.

[4] T.M. Mitchell, Machine Learning 81–117, 128–145, 157–198, 1st ed., McGraw-Hill Science/Engineering/Math, New York (1997) 52–78, Chapters 3,4,6,7.

[5] Prolexic. Global Leader in DDoS Protection and Mitigation 2003–2014. [Online] Available from: ⟨http://www.prolexic.com⟩.

[6] J. Li; Y. Liu; L. Gu, DDoS attack detection based on neural network, in: Proceedings of the 2nd International Symposium on Aware Computing (ISAC), Tainan, 1–4 Nov. 2010, pp. 196–199.

[7] V. Akilandeswari;S.M. Shalinie, Probabilistic neural network based attack traffic classification, in: Proceedings of the Fourth International Conference on Advanced Computing (ICoAC), Chennai, 13–15 Dec. 2012, pp.1–8.

[8] C., Siaterlis; V., Maglaris, Detecting incoming and outgoing DDoS attacks at the edge using a single set of network characteristics, in: Proceedings of the 10th IEEE Symposium on Computers and Communications, (ISCC), 27–30 June 2005, pp. 469–475.

[9] B.B. Gupta, C. Joshi, M. Misra, ANN based scheme to predict number of zombies in a DDoS attack, Int. J. Netw. Secur. 13 (3) (2011) 216–225.

[10] G. Badishi; I. Keidar; O. RomanovA. Yachin, Denial of Service? Leave it to Beaver, project supported by Israeli Ministry of Science, 2006, pp. 3–14.

[11] E. Shi; I. Stoica; D. Andersen; D. Perrig, OverDoSe: A Generic DDoS Protection Service Using an Overlay Network, Technical report CMU-CS-06-114, 2006, pp. 2–12. [Online] Available from: ⟨www.cs.umd.edu/~elaine/docs/overdose.ps⟩.

[12] Y. Chen, K. Hwang, W. Ku, Collaborative detection of DDoS attacks over multiple network domains, IEEE Trans. Parallel Distrib. Syst. 18 (12) (2007) 1649–1662.

[13] B. Al-Duwairi; G. Manimaran, A novel packet marking scheme for IP traceback, in: Proceedings of the Tenth International Conference on Parallel and Distributed Systems, ICPADS, 7–9 July 2004, pp. 195–202.

[14] C. Gong, K. Sarac, A more practical approach for single-packet IP traceback using packet logging and marking, IEEE Trans. Parallel Distrib. Syst. 19 (10) (2008) 1310–1324.

[15] S. Yu, W. Zhou, R. Doss, W. Jia, Traceback of DDoS attacks using entropy variations, IEEE Trans. Parallel Distrib. Syst. 22 (3) (2011) 412–425.

[16] J. Novak, S. Northcutt, Network Intrusion Detection, 3rd ed., SAMS (2002) 8–30.

[17] Stuttgart Neural Network Simulator, University of Stuttgart (Version 4.1), 1995. Available from: http://www.nada.kth.se/~orre/snns-manual/.

[18] M. Pino, A theoretical & practical introduction to self organization using JNNS, University of Applied Sciences, Brandenburg, 2005.

[19] T. Jayalakshmi, A. Santhakumaran, Statistical normalization and back propagation for classification, Int. J. Comput. Theory Eng. 3 (1) (2011) 89–93.

[20] Q. Zhang; S. Sun, Weighted data normalization based on Eigenvalues for artificial neural network classification, in: Proceedings of the 16th International Conference on Neural Information Processing, ICONIP, 2009, pp. 349–356.

[21] J., Wallen, IPTraf (Version 3.0) "Open Source project", 2005. Available form: ⟨http://iptraf.seul.org⟩.

[22] C. Bedón; A. Saied, Snort-AI (Version 2.4.3) "Open Source project", 2009. Available from: ⟨http://snort-ai.sourceforge.net/index.php⟩.

[23] M. Roesch, Snort (Version 2.9) "Open Source Project", 1998. Available from: ⟨http://www.snort.org⟩.

[24] R. Russell, Iptables (Version 1.4.21) "Open Source project", 1998. Available from: ⟨http://ipset.netfilter.org/iptables.man.html⟩.

[25] F. Leu; C. Pai, Detecting DoS and DDoS attacks using chi-square, in: Proceedings of the Fifth International Conference on Information Assurance and Security (IAS-09), Xian, 2009, PP. 225–258.

[26] X. Xu;D. Wei; Y. Zhang, Improved detection approach for distributed denial of service attack based on SVM, in: Proceedings of the Third Pacific-Asia Conference on Circuits, Communications and Systems (PACCS), Wuhan,17–18 July 2011, pp. 1–3.

[27] C. Jie-Hao; C. Feng-Jiao, Zhang, DDoS defense system with test and neural network, in: Proceedings of the IEEE International Conference on Granular Computing (GrC), Hangzhou, China, 11–13 Aug. 2012, pp. 38–43.


**Dr. Alan Saied** received his first degree in Computer Networks from Middlesex University in London and his M.Sc. in Computing and Internet Systems from King's College London University. Alan finished his Ph.D. in 2015 at King's College London University of London. Alan's technical skills have been many and varied. His expertise covers open source & enterprise technologies, security coupled with penetration testing. Alan worked with different enterprise solutions as well as contributing to open source community.

**Dr. Richard E Overill** is a Senior Lecturer in Computer Science at King's College London, a multi-faculty School of the University of London. He has authored or co-authored some 110 publications in academic journals, international conference proceedings and invited book chapters, specializing in computational science, parallel algorithm design and performance measurement, and, since 1996, cyber security and digital forensics.

**Prof. Tomasz Radzik** received an M.Sc. in Computer Science from Warsaw University in 1985 (M.Sc. thesis: "Communication and Routing in Synchronous Parallel Machines") and a Ph.D. in Computer Science from Stanford University in 1992 (Ph.D. thesis: "Algorithms for Some Linear and Fractional Combinatorial Optimization Problems"). He was a postdoctoral research associate at Cornell University from September 1992 to July 1993 and at King's College London from August to December 1993. He has been Lecturer (1994–2001), Senior Lecturer (2002–2006) and Reader (since 2007) in the Department of Informatics (formerly Department of Computer Science) at King's College London. He is a member of the Algorithms and Bioinformatics research group.