Deep Learning Intrusion Detection and Mitigation of DoS Attacks

Mohammed Nasereddin*, Mert Nakıp†, and Erol Gelenbe‡
*†‡*Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (IITIS-PAN)*, Gliwice, Poland
‡*King's College London*, London, United Kingdom
‡*Lab. I3S Université Côte d'Azur*, Nice, France
{*mnasereddin, †mnakip, ‡seg}@iitis.pl

***Abstract*—Internet of Things (IoT) networks are highly vulnerable to common network DoS and DDoS attacks, which flood limited system resources or IoT devices, overwhelming them with large numbers of attack packets. In order to mitigate such attacks, this paper develops a lightweight yet effective Intrusion Detection and Prevention System (IDPS), that sequentially detects and mitigates the attack via a Deep Random Neural Network (DRNN) and a Drop-Idle-Repeat process. The IDPS is evaluated for UDP Floods, attacks on an experimental test-bed. The results show that UDP Flood attacks can be mitigated with the proposed IDPS, allowing the system to continue routine operations, and resume communications when the attack ends.**

***Index Terms*—Internet of Things (IoT), Intrusion Detection and Mitigation, DoS attacks, Deep Random Neural Network**

I. Introduction

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are two of the most common and damaging cyber threats to networked systems [1], which disable networked systems by flooding them with large streams of requests, causing productivity decline, financial losses, and reputational damage. In 2022, a total increase of 150% in such attacks was observed worldwide [2], with 109% more occurring at the network layer and 72% more occurring at the application layer [1]. Therefore, this paper develops an Intrusion Detection and Prevention System (IDPS) that learns only from normal traffic packets and generalizes its knowledge to differentiate attack traffic.

Section II describes the test-bed and the traffic generation we use. Section III presents the proposed IDPS, as well as the packet drop process when attacks are detected. Section III-A details the learning algorithm. Section IV evaluates the IDPS performance when Flood attacks occur, and Section V offers concluding remarks.

A. Related Work

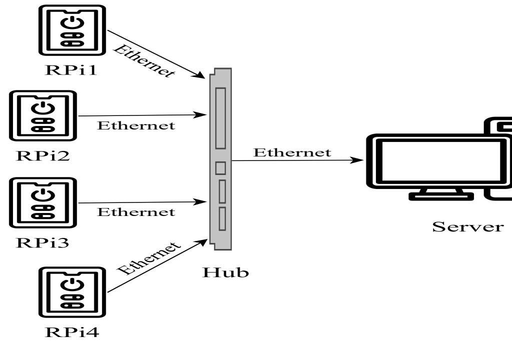
In [3], an IDPS against port-scanning and DoS attacks is developed, and an adaptive IDPS utilizing an intelligent agent for IoT networks is presented in [4]. In [5] a network-level IDPS that inspects the IoT network activity, drops malicious packets and blocks their source address is shown, while. In [6], a neural network-based IDPS to mitigate DoS attacks in mobile ad hoc networks is presented. Abdulqadder et al. [7] use deep reinforcement learning in their multi-layer IDPS to mitigate flow table overloading attacks in SDNs. In order to mitigate secure shell (SSH) brute-force and DDoS attacks in SDNs, Finally, in preliminary work [8], a performance evaluation of deep learning based attack detection on a small experimental test-bed shows promising results. While existing research mostly evaluates their methodologies using publicly available datasets, specialized test-beds tailored for cyber-physical systems, industrial control, and IoT environments have been suggested [9], in particular for industrial control systems [10], Smart Grid Substation Automation Systems (SCIDS) [11], and for Supervisory Control and Data Acquisition (SCADA) systems [12], WAN network security [13], and DoS attacks on software-defined networks and autonomous vehicle testbeds [14]. Recently [15] it was also shown that the QDTP traffic shaping policy can significantly protect an IoT server from massive Flood attacks.

II. Experimental Setup

In order to analyse the impact of DoS attacks and the performance of the IDPS developed in this paper, we build a Local Area Network (LAN) environment for experimental testing. As shown in Figure 1, this environment presently consists of four IoT devices and a Server; however, it can easily be expanded to include an arbitrary number of connected devices with multiple sources of traffic and attacks. Three of the IoT devices (RPi1, RPi2, and RPi3) generate and transmit normal – benign – IP packet traffic, while the last device (RPi4) generates a combination of benign and malicious traffic.

The test-bed comprises four Raspberry Pi 4 Model B Rev 1.2 units that emulate IoT devices. They include a 1.5 GHz ARM Cortex-A72 quad-core processor, 2GB LPDDR4 - 3200 SDRAM, and run Raspbian GNU/Linux 11 (bullseye). An Intel Core i7 - 8705G-powered robust Server with 16GB of RAM, eight 3.10 GHz cores, and a 500GB hard drive, operating on Linux 5.15.0-60-generic 66-Ubuntu SMP, assumes the pivotal role of detecting attacks and storing incoming packets. All devices are interconnected through a central Ethernet hub, as shown in Figure 1, and communicate via UDP. Attacks are generated from a public MHDDoS repository [16], containing 56 different types of DoS attacks against the IP transport and application layer.RPi1, RPi2, and RPi3 generate normal traffic, and RPi4 generates normal and Flood attack traffic.

This research was supported by the European Commission's H2020 DOSS Project under Grant Agreement No. 101120270.



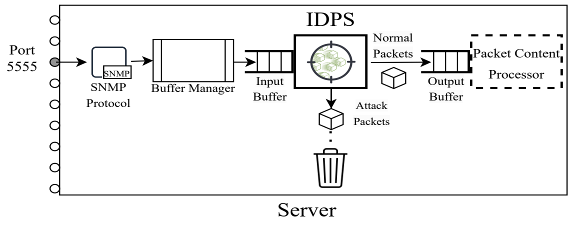


Fig. 1. The LAN test-bed (above) communicates using Ethernet, with four Raspberry Pis as sources for both normal and attack traffic, and an Intel 8-Core Processor as the Server. The description of the Server that supports the IDPS is shown below.

III. Intrusion Detection and Prevention System

The IDPS is resident on the Server, and is composed of the Intrusion Detection (ID) algorithm and the mitigation algorithm based on dropping packets. It receives externally arriving packets via the “Buffer Manager”, which queues the packets at the entrance of the IDPS. The ID classifies each packet as “benign” or “malicious” using the ADRNN from the traffic metrics of batches of M successive packets. Let t_i be the instant when packet i is transmitted from its source, and b_i be its length in bytes. The three metrics ID uses are:

1) The total size of the last I packets up to and including the latest packet i, and the
2) The average inter-transmission time of those packets,

$$x_i^1 = \sum_{j=0}^{I-1} b_{(i-j)}, \quad \text{and} \quad x_i^2 = \frac{1}{I} \sum_{j=0}^{I-1} \big[t_{(i-j)} - t_{(i-j-1)}\big]. \tag{1}$$

3) The total number of packets transmitted in the last T seconds up to the transmission of packet i:

$$x_i^3 = \big|\{j : (t_i - T) \leq t_j < t_i\}\big|. \tag{2}$$

Each metric is then normalized via min-max scaling on the whole dataset $\mathcal{D}_{\text{train}}$

$$x_i^m \leftarrow \min\Big[\frac{x_i^m - \min_{j\in\mathcal{D}_{\text{train}}} x_j^m}{\max_{j\in\mathcal{D}_{\text{train}}} x_j^m - \min_{j\in\mathcal{D}_{\text{train}}} x_j^m}, 1\Big], \quad \forall m. \tag{3}$$

The ID learning algorithm we use is described in [17]. It uses the auto-associative Deep Random Neural Network (ADRNN), an instance of the G-Network [18] that learns only from normal traffic. For each packet i, the ID computes the decision $y_i \in \{0, 1\}$ based on the input vector $x^i = [x_i^1, x_i^2, x_i^3]$, and the AADRNN weights that are learned from normal traffic, so that the ID outputs $\hat{x}_i = [\hat{x}_i^1, \hat{x}_i^2, \hat{x}_i^3]$. The decision making module computes the decision variable y_i (attack or non-attack) from the difference between x^i and $\hat{x}_i$.

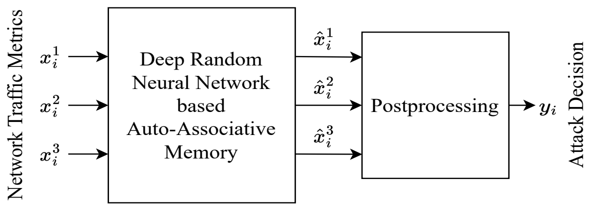


Fig. 2. The ID algorithm is composed of the Auto-Associative Deep Random Neural Network (AADRNN) and the decision making module. It computes the decision variable y_i from the traffic metrics $[x_i^1, x_i^2, x_i^3]$.

A. Learning Algorithm for Attack Detection

The AADRNN is a Deep Random Neural Network neuronal network model [19] whose auto-associative learning provides accurate attack detection in large scale test cases [20]. It is organized in L feed-forward layers, each comprised of N_l clusters of n_l identical neurons, $l \in \{1, \ldots, L\}$. The weight matrices W_l connect the clusters of layers l to $l + 1$, and the weights are learned to create an auto-associative memory. The forward pass of the AADRNN is:

$$\begin{aligned} \hat{x}_i^l &= \zeta([\hat{x}_i^{l-1}, 1]\, W_{l-1}),\ 1 \leq l \leq L, \\ \hat{x}_i &= [\hat{x}_i^{L-1}, 1] W_{L-1}, \end{aligned} \tag{4}$$

where $\hat{x}_i^l$ is the output of layer l for packet i, $\hat{x}_i^0 = x_i$, while $[\hat{x}_i^l, 1]$ indicates that 1 is concatenated to the output of each layer l as a multiplier of the bias, and $\zeta(\lambda)$ is the DRNN specific neuron activation function [19].

If n_l is large we can simplify the activation function to:

$$\zeta(\lambda) = \frac{[r(1-p) - p\lambda^+]\Big[1 \pm \sqrt{1 - \frac{4p(\lambda+\lambda^-)[\lambda^+ - r - \lambda - \lambda^-]}{r(1-p) - p\lambda^+}}\,\Big]}{2p(\lambda + \lambda^-)}, \tag{5}$$

where r is the total firing rate of each neuron, λ^+ and λ^- are external excitatory and inhibitory spike rates arriving at the given cell, and p is the probability that any other neuron in the network fires when a given neuron fires, representing the soma-to-soma interactions.

The binary decision variable y_i is then: using the threshold $1 > \gamma > 0$:

$$y_i = \begin{cases} 1, & \text{if } \frac{1}{3}\sum_{m=1}^{3}\big|x_i^m - \hat{x}_i^m\big| \geq \gamma \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The learning algorithm for the IDPS uses a three-layer AADRNN (i.e. $L = 3$), where each layer contains 3 clusters, whose weights are learned using only normal (benign) traffic. At system initialization, the first 500 traffic packets received by the Server are selected to be normal traffic, and during this cold-start period, the connection weight matrices W_l between each layer l and $l+1$ are calculated using the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [21]:

$$W_l = \arg \min_{W:W\geq 0} \Big[\; ||W||_{L_1} \\ + ||[adj(\zeta(\hat{X}^{\text{train}}_{l-1} W_R)), \mathbf{1}_{(|\mathcal{D}_{\text{train}}|\times 1)}]W - \hat{X}^{\text{train}}_{l-1}||^2_{L_2} \; \Big], \\ where \; \hat{X}^{\text{train}}_{l} = \{\hat{x}^l_i\}_{i\in\mathcal{D}_{\text{train}}},$$

and $\hat{X}^{\text{train}}_{l}$ is the matrix of outputs of layer l resulting from the training dataset $\mathcal{D}_{\text{train}}$ (i.e. the first 500 normal packets), and $\mathbf{1}_{(|\mathcal{D}_{\text{train}}|\times 1)}$ is a column vector of ones, of length $|\mathcal{D}_{\text{train}}|$.

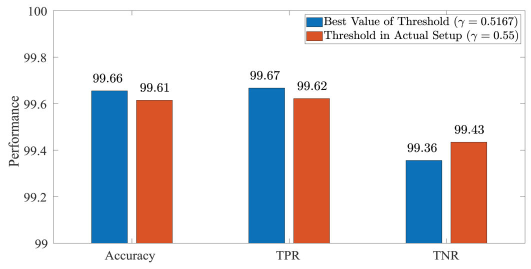


Fig. 3. The performance of AADRNN with a parameter setting of $\gamma = 0.55$ is assessed and compared to the optimal value of $\gamma = 0.5167$, with respect to Accuracy, True Positive Rate (TPR), and True Negative Rate (TNR) in the experimental scenario where RPi4 launches a UDP Flood attack lasting for 10 seconds.

B. Mitigation

Based on the binary intrusion decision y_i provided by the ID algorithm for packet i, the IDPS Mitigation and Server protection system prevents the Server from receiving attack packets and being overwhelmed by the overload created by the attack. If an intrusion is detected by the ID, all the packets in the Server input buffer are simply dropped, and the Server shuts down its communication with other devices, and runs its own task processing for some time interval, so that its critical operations, and its data, remain secure.

Algorithm 1 The IDPS

```
 1: $p \leftarrow 0$
 2: **while** $|B_{in}| > 0$ **do**
 3:     $p \leftarrow p + 1$
 4:     $x_p \leftarrow$ **CalcMetrics**$( B_{in}, \; p )$
 5:     $y_p \leftarrow$ **ID**$( x_p )$
 6:     **if** $( p \geq M )$ **and** $( \sum_{j=(p-M+1)}^{p} y_p \geq \frac{M}{2} )$ **then**
 7:         $B_{in} \leftarrow ( \, )$
 8:         **back-off**$( \tau )$
 9:         $p \leftarrow 0$
10:     **end if**
11: **end while**
```

The pseudo-code of the IDPS is given in Algorithm 1, where it is assumed that the externally arriving packets are received into the buffer – denoted by B_{in} – which is updated by the Buffer Manager in parallel to the operations of the IDPS. As shown in this algorithm (between Lines 2-11), IDPS analyzes each packet that is added to B_{in}. For each packet indexed by p in B_{in}, first, the intrusion decision is obtained: in Line 4, the traffic metrics are calculated via the **CalcMetrics** function which operates through (1) - (3). In Line 5, the intrusion decision is computed by the trained ID. Then, between Lines 6-10, if the majority of the most recent M packets are classified as an attack, the drop process starts. In order to eliminate the long processing of the enormous number of packets sent by the attacker, on Line 7, B_{in} is emptied. Subsequently, in line 8, the packet processing system of the Server, including the Buffer Manager and the IDPS, **backs-off** for τ seconds, and the Buffer Manager does not accept any new packet and drops all incoming packets. At the end of τ secs, the ID continues with new incoming packets.

M which is chosen so that the ID decisions are accurate, and we observe from Figure 4 that the number of false positive decisions is minimized when $M = 40$.

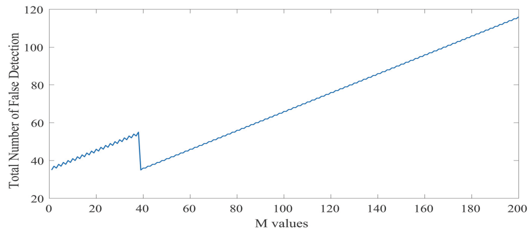


Fig. 4. Total number of packets classified as an attack with false decision for different values of M during a UDP Flood attack that lasts 10 seconds

The average inter-transmission time T_n for any normal packet is greater than the execution time per packet T_e of the ID algorithm. On the other hand, when an attack occurs, the average packet inter-transmission time T_m of malicious traffic will typically be smaller than T_e.

The second critical parameter τ is the back off period after a drop decision, and should satisfy $\tau > T_d$. Otherwise, if the attack continues during the decision process between two successive back-off periods, a large amount of malicious traffic may flood the input queue and cause system slowdown (or even shutdown). Accordingly, we write:

$$\tau = \frac{1}{\beta} T_d, \tag{7}$$

where we empirically set $\beta = 0.2$ in our experiments. As $M = 40$, $T_n = 0.25 \; s$, and $T_e = 3.2 \; ms$, the resulting $\tau = 25.32 \; s$.

IV. PERFORMANCE EVALUATION

Our experiments provided the results summarized in Figure 5 for a 60-second UDP Flood attack. They reveal that the proposed IDPS successfully mitigates the attack resulting in a short packet queue. Although the UDP Flood attack transmits more than $400,000$ packets, the IDPS mitigates the attack after 912 packets are received, and prevents the Server from becoming paralysed. Figure 5 (top) shows that the IDPS takes successive decisions fast enough that the input queue length reaches up to 39 packets between two decisions, and Figure 5 (middle) shows that these decisions keep packet delay under 1 second for the packets that are processed after the 60 sec. attack ends. Figure 5 also reveals that the IDPS has made a false positive decision just after the attack ends, resulting in an extra $\tau = 25.32\ s$ spent in the idle state, and causing additional packet loss. Figure 5 (bottom), shows that the cumulative packet loss during a total of three idle states reaches only up to 297 packets.

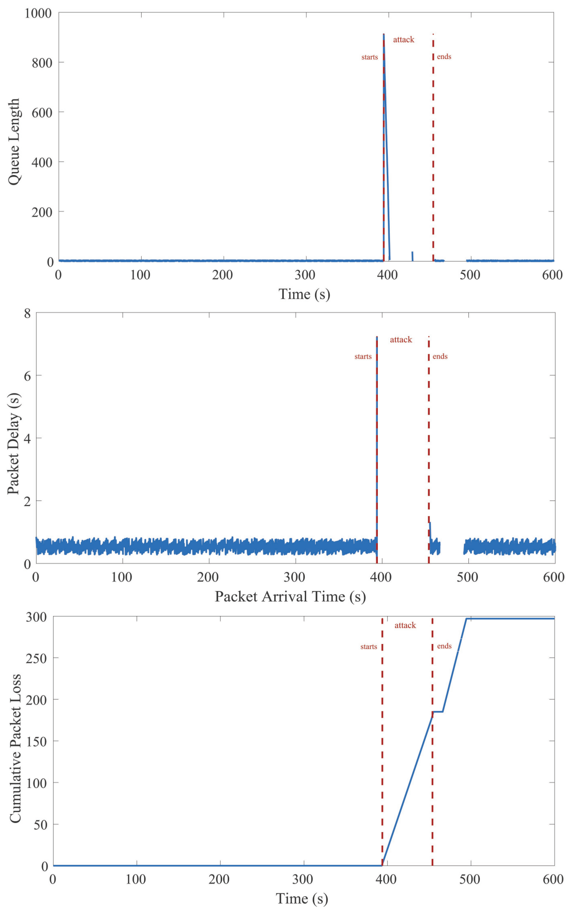


Fig. 5. Queue length (top), packet delay (middle), and packet loss (bottom), measurements on the Server utilizing the proposed IDPS during the 60-second UDP Flood attack; the decision to perform mitigation via the DIR process results in short packet queue length, avoiding Server paralysis

V. Conclusions

DoS and DDoS Floods are common attacks against IoT networks, that overwhelm system resources, causing delayed, lost, or inaccurate data and service interruptions. In this paper, we have described an IDPS that detects and mitigates such attacks and secures the overall IoT system. IDPS is comprised of a Deep Random Neural Network (DRNN) [19] together with an algorithm that drops all the packets in the system's input buffer if an intrusion is detected by DRNN among the most recent M packets, where M is set to 40. Experiments on a system test-bed show that IDPS successfully detects and mitigates Flood attacks.

References

[1] ExtraHop, Denial of Service Attack: Definition, Examples, and Prevention, accessed: 2023-03-09 (January 2022).
URL https://www.extrahop.com/resources/attacks/dos/

[2] S. Staff, Organizations fought an average of 29.3 attacks daily in late 2022 (Feb 2023).
URL https://www.securitymagazine.com/articles/98958-organizations-fought-an-average-of-293-attacks-daily-in-late-2022

[3] C. Birkinshaw, E. Rouka, V. G. Vassilakis, Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks, Journal of Network and Computer Applications 136 (2019) 71–85.

[4] S. T. Bakhsh, S. Alghamdi, R. A. Alsemmeari, S. R. Hassan, An adaptive intrusion detection and prevention system for internet of things, International Journal of Distributed Sensor Networks 15 (11) (2019) 1550147719888109.

[5] A. Kumar, K. Abhishek, M. R. Ghalib, A. Shankar, X. Cheng, Intrusion detection and prevention system for an iot environment, Digital Communications and Networks 8 (4) (2022) 540–551.

[6] M. Islabudeen, M. Kavitha Devi, A smart approach for intrusion detection and prevention system in mobile ad hoc networks against security attacks, Wireless Personal Communications 112 (2020) 193–224.

[7] I. H. Abdulqadder, S. Zhou, D. Zou, I. T. Aziz, S. M. A. Akber, Multi-layered intrusion detection and prevention in the sdn/nfv enabled cloud of 5g networks using ai-based defense mechanisms, Computer Networks 179 (2020) 107364.

[8] M. Nasereddin, M. Nakıp, E. Gelenbe, Measurement based evaluation and mitigation of flood attacks on a lan test-bed, in: 2023 IEEE 48th Conference on Local Computer Networks (LCN), IEEE, 2023, pp. 1–4.

[9] O. A. Waraga, M. Bettayeb, Q. Nasir, M. A. Talib, Design and implementation of automated iot security testbed, Computers & Security 88 (2020) 101648.

[10] M. Kaouk, F.-X. Morgand, J.-M. Flaus, A testbed for cybersecurity assessment of industrial and iot-based control systems, in: Lambda Mu 2018-21è Congrès de Maîtrise des Risques et Sûreté de Fonctionnement, 2018.

[11] M. Annor-Asante, B. Pranggono, Development of smart grid testbed with low-cost hardware and software for cybersecurity research and education, Wireless Personal Communications 101 (2018) 1357–1377.

[12] B. Reutimann, I. Ray, Simulating measurement attacks in a scada system testbed, in: Critical Infrastructure Protection XV: 15th IFIP WG 11.10 International Conference, ICCIP 2021, Virtual Event, March 15–16, 2021, Revised Selected Papers 15, Springer, 2022, pp. 135–153.

[13] S.-U. Park, S.-M. Hwang, Test bed construction for apt attack detection, International Journal of Control and Automation 11 (4) (2018) 175–186.

[14] P. V. Sontakke, N. B. Chopade, Impact and analysis of denial-of-service attack on an autonomous vehicle test bed setup, in: Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2021, Springer, 2022, pp. 221–236.

[15] E. Gelenbe, M. Nasereddin, Protecting iot servers against flood attacks with the quasi deterministic transmission policy (best paper award, ieee trustcom 2023), in: IEEE 22nd International Conference on Trust, Security and Privacy Computing and Communications (TrustCom), Exeter, UK, 2024. doi:10.1109/TrustCom60117.
URL https://arxiv.org/pdf/2306.11007.pdf

[16] MHDDoS - DDoS Attack Script With 56 Methods, Online, accessed: 2023-02-22 (May 2022).
URL https://github.com/MatrixTM/MHDDoS

[17] M. Nakıp, E. Gelenbe, Mirai botnet attack detection with auto-associative dense random neural network, in: 2021 IEEE Global Comm. Conference (GLOBECOM), IEEE, 2021, pp. 01–06.

[18] E. Gelenbe, G-networks with instantaneous customer movement, Journal of Applied Probability 30 (3) (1993) 742–748.

[19] E. Gelenbe, Y. Yin, Deep learning with random neural networks, in: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 1633–1638.

[20] E. Gelenbe, M. Nakıp, Traffic based sequential learning during botnet attacks to identify compromised iot devices, IEEE Access 10 (2022) 126536–126549.

[21] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM journal on imaging sciences 2 (1) (2009) 183–202.