Measurement Based Evaluation and Mitigation of Flood Attacks on a LAN Test-Bed

Mohammed Nasereddin, Mert Nakıp and Erol Gelenbe *Fellow IEEE*
Institute of Theoretical and Applied Informatics
IITIS-PAN, Polish Acad. Sci., 44-100 Gliwice, PL

***Abstract*—The IoT's vulnerability to network attacks has motivated the design of intrusion detection schemes (IDS) using Machine Learning (ML), with a low computational cost for online detection but intensive offline learning. Such IDS can have high attack detection accuracy and are easily installed on servers that communicate with IoT devices. However, they are seldom evaluated in realistic operational conditions where IDS processing may be held up by the system overload created by attacks. Thus we first present an experimental study of UDP Flood Attacks on a Local Area Network Test-Bed, where the first line of defence is an accurate IDS using an Auto-Associative Dense Random Neural Network. The experiments reveal that during severe attacks, the packet and protocol management software overloads the multi-core server, and paralyses IDS detection. We therefore propose and experimentally evaluate an IDS design where decisions are made from a very small number of incoming packets, so that attacking traffic is dropped within milli-seconds after an attack begins and the paralysing effect of congestion is avoided.**

***Index Terms*—Internet of Things, Local Area Networks, Cybersecurity, UDP Flood Attacks, Intrusion Detection and Mitigation**

I. Introduction

The risk of cyber threats, which may do considerable damage to businesses, has increased with the growing dependence on networked technologies. Denial of service (DoS) attacks, which can disable a target system or network by flooding it with a huge stream of requests, are among the most common and destructive forms of cyberattacks which cause reputational damage, and financial and productivity losses to organizations. Thus the year 2022 saw a significant increase in Distributed DoS (DDoS) attacks, with a jump of 150% worldwide [1], indicating a higher number, complexity, volume, duration, power, and frequency of such attacks. On average, organizations faced 29.3 attacks per day during Q4 2022, or 3.5 times higher than in 2021, while the largest reported DDoS attack started in September 2017, but was only disclosed in 2020. It targeted Google, with spoofed packets sent to 180,000 web servers which then responded to Google, attaining total bitrates of 2.54 Tera-bits per second [2].

However, DoS attacks also target the IoT and industrial control systems, as well as vital infrastructure, such as power grids and transportation systems [3], [4]. Among the different types of DoS and DDoS attacks, SYN attacks [5] overwhelm the victim by creating repeated requests for the opening of a connection and overloading the victim's processing capacity, and its energy if it is battery operated, while Botnet attacks can be devastating [6] since they spread by using victims as attackers [7]–[9].

UDP Flood attacks [10] are simple and "popular" since they readily overwhelm the target network with a large number of forged-source address UDP packets, causing it to crash or become unresponsive. Often launched with a small number of compromised systems, they direct a high volume of traffic at the targets, resulting in a denial of service for normal users. When networks have limited capabilities such as sensor networks, UDP Flood attacks cause delayed or lost data and inaccurate or incomplete readings [11], and UDP's connectionless behavior [12] will cause even closed ports to respond by sending back an ICMP message that creates overhead for the victim.

A. Aims of this Paper

While there is abundant literature on attack detection methods, most evaluations of these methods are conducted under ideal conditions on a general purpose computer where the attack traffic is treated as data. Such a setting cannot represent the actual arrival process of attack traffic, the backlog that forms in front of the attack detector after the traffic enters the port that it is attacking, the possible effects of an avalanche of attack traffic that causes the overflow of input buffers and legitimate traffic to be dropped, or the effect of delayed decisions concerning the packets that are malicious and those which are legitimate.

As a consequence, in this paper, we use a practical cost-effective test-bed for network attack detection evaluation, which incorporates transmitting devices and a network port placed at a server where traffic is received and attack detection takes place. The purpose is to compare the "ideal" evaluation results concerning attack detection algorithms, with the actual overall system performance in a Local Area Network (LAN) environment. In this context, we can measure the precision of the IDS itself, but also its delay in providing decisions due to congestion during a UDP Flood Attack. The test-bed allows us to study remedial actions to drop attacking packets and protect the bandwidth and buffer needs of benign traffic.

In this paper, we therefore use the LAN test-bed for evaluating an attack detection technique by conducting a systematic study of the performance of a recent machine learning based Intrusion Detection System (IDS) [13] that uses the Auto-Associative Dense Random Neural Network (AA-DenseRNN).

The rest of the paper is organized as follows: Section II reviews the recent related works. Section III describes the experimental setup and devices used, and Section IV presents the AADRNN-based attack detection algorithm and its performance under ideal conditions compared to real-world experiments. Section V presents the system's behaviour when exposed to UDP Flood attacks through different scenarios and the improvements achieved by an attack mitigation algorithm. Finally, Section VI concludes the paper and outlines directions for future work.

II. Related Work

Because it allows for the simulation of real-world network conditions in controlled and reproducible environments, developing and using a reliable test-bed to evaluate DoS attack detectors was recommended in early work [14], but was not frequently used.

Several researchers have developed test-beds for cyber-physical systems, industrial control systems (ICSs), and IoT environments [15]. In [16], a semi-physical test-bed for ICSs was proposed, while in [17], a low-cost Smart Grid test-bed for SCIDS systems using Arduino microcontrollers, XBee radio modules, Suricata and Snort intrusion detection and prevention systems (IDPSs), Bonesi botnet simulator, and Winlog Lite was evaluated for using TCP flood attacks. In [18], a real-time test-bed for cyber-physical systems was implemented, whereas in [19], the performance of an attack-resilient control system for Automatic Generation Control (AGC) in power systems was evaluated. In [20] the performance of an attack-resilient control system for wind farm SCIDS systems (WFSS) was studied using a test-bed with SYN flood attacks, and in [21]–[23], SCADA systems are examined.

In other contexts, in [24], they proposed a test-bed using six NetFlow tools for collecting, analyzing, and displaying data with HTTP-GET flood attacks on a WAN network. In [25], the impact of current datasets on IoT systems and developed a real-time data collection platform for DNS amplification attacks in IoT was investigated, and [26] addressed the problem of DoS attacks on software-defined networks (SDN), and [27] conducted experiments analyzing DoS attacks on an autonomous vehicle test-bed.

The KDD99 dataset and its improved edition, NSL-KDD, are widely used in network security research because of the vast collection of network traffic records they include. They are still frequently used as a benchmark dataset for evaluating the effectiveness of DoS attack detection. However, one notable shortcoming is that they were generated in a simulated environment, which may not adequately reflect the complexities and nuances of real network traffic. Many other examples of datasets are used for the same purposes (e.g., UNSW-NB15, CICDS2017, and Bot-IoT dataset) [25].

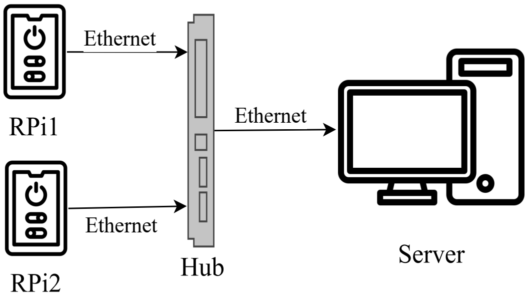

Fig. 1. Testing Environment using Ethernet for communications, with Raspberry Pi machines acting as forwarders of normal and attack traffic, and an Intel 8-Core Processer used as a server to process incoming packet traffic and detect attacks.

Recent work develops datasets that better reflect current threats, so this paper uses the MHDDoS repository [28] that performs real-world DoS attacks with 56 different modern constantly updated methods.

III. Experimental Setup

Practically all published work on cyberattack detection techniques publish statistical results based on testing in a pure software environment, which smooths over the realities of the network and device hardware, or the side effects of attack traffic on the receiver devices or network ports, such as the creation of large queues of packets. Such ideal environments can obtain purely statistical evaluations regarding the accuracy of the algorithms being used, but cannot apprehend the huge processing backlogs that such attacks often cause, which impede attack detection from being carried out in a timely fashion which is needed to take mitigating measures, and which also can cause the loss and delay of legitimate traffic due to the large packet backlogs.

Thus, in this work, we attempt to address these issues by establishing a physical test environment to evaluate LAN network attack detector software and algorithms in more realistic conditions. This environment, which can be expanded to include an arbitrary number of linked devices with multiple sources of traffic and attacks, presently consists of three scalable devices. Two traffic-generating devices, one that transmits normal benign IP packet traffic while the other sends a combination of benign and malicious traffic. These devices are embodied by two Raspberry Pi 4 Model B Rev 1.2 machines (RPi1 and RPi2) as transmitters. They each have a 1.5GHz ARM Cortex-A72 quad-core processor and 2GB $LPDDR4 - 3200$ SDRAM and run the latest version of Raspbian GNU/Linux 11 (bullseye), a Debian-based operating system optimized for the Raspberry Pi hardware. A server with an Intel Core $i7-8705G$ processor acts as the receiver of the packet traffic and is responsible for detecting the attack and for storing the arriving packets. It has 16GB of RAM and a 500GB hard drive. It runs Linux $5.15.0-60-$ generic $66-$Ubuntu SMP, an Ubuntu-based operating system with eight cores, each running at 3.10GHz. The traffic is carried over Ethernet connections between all devices interconnected via a hub, as shown in Figure 1.

The specifications of the Raspberry Pi devices and the computer were carefully chosen to ensure that the devices are capable of effectively transmitting and receiving packets of data through the Ethernet connection. These devices communicate using the UDP protocol due to its simplicity and low overhead. In contrast to TCP, UDP operates without establishing a connection before transmitting data and without providing any ACK or error recovery mechanisms and is a fast and efficient protocol for real-time applications [29].

IV. The IDS and its Ideal Performance

For completeness, we first detail the Attack Detection Algorithm used in this paper and its performance. It is a version of the IDS developed in [13] based on the Deep Random Neural Network (DRNN) [30] with Auto-Associative Learning (AADRNN).

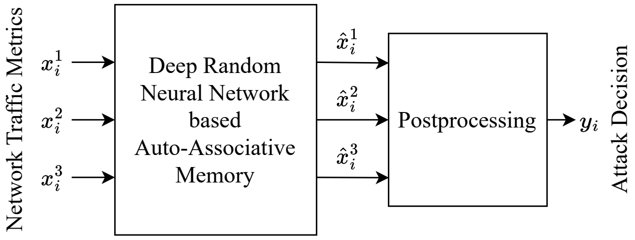

Fig. 2. The structure of the IDS system that computes the decision variable y_i from the network traffic metrics $[x_i^1, x_i^2, x_i^3]$ with the DRNN based Auto-Associative Random Neural Network (AADRNN) and the postprocessing module.

Figure 2 shows how the AADRNN algorithm computes the decision $y_i \in \{0, 1\}$ using three metrics calculated from network traffic. To perform this operation, the attack detection scheme is comprised of the AADRN followed by a postprocessing module. The algorithm is an anomaly-based intrusion detector, and only **learns with normal traffic** with measured metrics $x^i = [x_i^1, x_i^2, x_i^3]$ from successive sets of packets. The AADRNN learns to predict the metrics that are expected to be measured from traffic in the absence of an intrusion, namely $\hat{x}_i = [\hat{x}_i^1, \hat{x}_i^2, \hat{x}_i^3]$. When a measurement x^i is entered into the AADRNN, it outputs the response $\hat{x}_i$, and the difference between the input and the output is used to compute the decision variable y_i (attack or non-attack).

The AADRNN is built using the DRNN neuronal model [30], an extension of the Random Neural Network [31], which incorporates soma-to-soma triggering between neurons, as well as the commonly used excitatory and inhibitory spikes. It uses auto-associative learning as the attack detection technique in [32], and provides accurate detection in significant test cases [13], [33]–[35]. The DRNN is organized in $l \in \{1, \ldots, L\}$ feed-forward layers, each comprised of N_l clusters, each cluster having n_l identical neurons. Weight matrices W_l connect the clusters of layer l to those of layer $l+1$, and the weights are learned to create an auto-associative memory. For the input vector x_i, the forward pass of the AADRNN is:

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

where $\hat{x}_i^l$ is the output of layer l for packet i, $\hat{x}_i^0 = x_i$, and $[\hat{x}_i^l, 1]$ indicates that 1 is concatenated to the output of each layer l as a multiplier of the bias, and $\zeta(\lambda)$ is the neuron activation function [30]. If the n_l is large we can simplify the transfer function to:

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

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. In our experiments, we have set the values of these parameters as follows: $r = 0.001$, $\lambda^+ = \lambda^- = 0.1$, and $p = 0.05$.

The weights W_l between layers are only learned for normal or "benign" traffic using the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [36]:

$$\begin{aligned} & W_l = \\ & \underset{\{W : W \geq 0\}}{\arg\min} [\; ||adj(\zeta(\hat{X}_{l-1}^{\text{train}}\, W_R))W - \hat{X}_{l-1}^{\text{train}}||_{L_2}^2 + ||W||_{L_1} \;] \end{aligned} \tag{3}$$

where $\hat{X}_l^{\text{train}}$ is the matrix of outputs of layer l resulting from data from the training dataset $\mathcal{D}_{\text{train}}$:

$$\hat{X}_l^{\text{train}} = \{\hat{x}_i^l\}_{i \in \mathcal{D}_{\text{train}}} \tag{4}$$

In the experiments reported in this paper, AADRNN learning is carried out with a small dataset consisting of the first 500 packets received by the server. Thus the time until 500 packets are received can be viewed as the "cold-start", and we ensure that only benign packets are received during this time. The duration of the cold-start depends on the ongoing packet arrival rate, varying between 25 seconds and as long as 9 minutes.

A. Traffic Metrics and Decision Making

We use traffic metrics from recent work [13] that aim to capture the signatures of DDoS attacks, especially Mirai Botnet attacks. In [34] these metrics were extended to identify several different DoD and DDoS attacks, as well as Botnets, and it is the latter approach that we use in this work. If t_i is the instant when packet i is transmitted and b_i be its length in bytes. The first metric is the total size of the last I packets observed by IDS up to and including packet i, while the second one is the average inter-transmission time of the last I packets observed by IDS up to and including packet i:

$$x_i^1 = \sum_{j=0}^{I-1} b_{(i-j)},\ x_i^2 = \frac{1}{I}\sum_{j=0}^{I-1}\big[t_{(i-j)} - t_{(i-j-1)}\big]. \quad (5)$$

The third metric is the total number of packets transmitted in the last T seconds up to the transmission of packet i:

$$x_i^3 = \big|\{j : (t_i - T) \leq t_j < t_i\}\big|. \quad (6)$$

Each metric is normalized via min-max scaling using the training dataset $\mathcal{D}_{\text{train}}$ as

$$x_i^m \leftarrow \min\Big[\frac{x_i^m - \min_{j\in\mathcal{D}_{\text{train}}} x_j^m}{\max_{j\in\mathcal{D}_{\text{train}}} x_j^m - \min_{j\in\mathcal{D}_{\text{train}}} x_j^m}, 1\Big] \quad (7)$$

From the output $\hat{x}_i$ of the AADRN with input x_i, the binary decision variable y_i is obtained using the threshold $1 > \gamma > 0$:

$$y_i = \begin{cases} 1, & \text{if } \frac{1}{3}\sum_{m=1}^{3}\big|x_i^m - \hat{x}_i^m\big| \geq \gamma \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

B. *Real-Time Detection Performance of AADRNN*

An experiment was run for a UDP Flood attack that lasts for 10 seconds and the IDS identified $2,343$ benign and $153,657$ malicious packets that were received over 17 minutes. The resulting performance of AADRNN deployed on the LAN test-bed is summarized in Figure 3, reporting the Accuracy, TPR, and TNR for the experiment, which lasts approximately 17 minutes, where RPi2 starts a UDP Flood attack randomly, which lasts for 10 seconds. The results show that AADRNN achieves high performance both when a predefined value of threshold $\gamma = 0.3$ is used in real-time testing and when the best value of threshold $\gamma = 0.3787$ is used. The experimental results show that AADRNN yields around 99.7% Accuracy and TPR, while its TNR is 98.48%. Thus the *ideal performance of IDS* under the best threshold selection is, as expected, only slightly higher. The results were not significantly different when the attack lasted 60 seconds between those for threshold $\gamma = 0.3$ compared to the best threshold value $\gamma = 0.2176$: Accuracy and TPR were 99.89%, and TNR was of 96.31%. We also observed that the AADRNN also raised an alarm just after the attack traffic from the compromised device RPi2 stopped, as shown in Figure 4.

V. System Behaviour with Normal and Attack Traffic

We now analyze the behaviour of the system operating within the server in our experimental setup. As this system is shown in Figure 5, the server receives the traffic packets from linked devices on port 5555, which are then passed to the buffer manager by a network protocol and queued to be analyzed by the AADRNN-based IDS. Based on the decisions of IDS, a batch of 10 packets is classified as normal or attack traffic. The packets in this batch are classified as normal only if the IDS detects the majority of them as normal traffic. Packets classified as normal traffic are forwarded to the packet content processor (representing the rest of the operations performed by the server); otherwise, they are dropped. In this way, we aim to ensure the security and accessibility of the server.

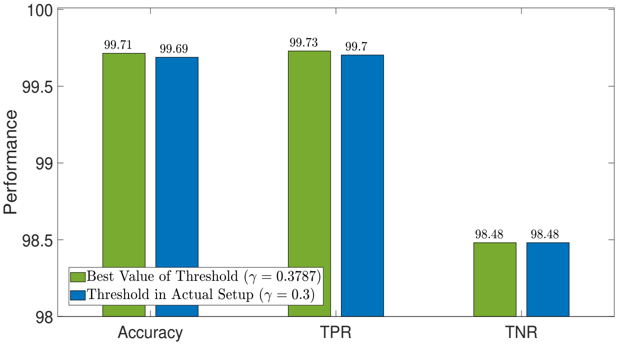

Fig. 3. The performance of AADRNN with $\gamma = 0.3$, and compared with the best value of $\gamma = 0.3787$, is evaluated with respect to Accuracy, TPR, and TNR for the experiment where RPi2 starts a UDP Flood attack lasting 10 seconds

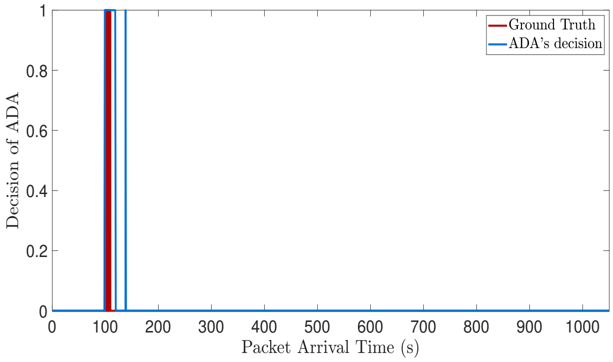

Fig. 4. The AADRNN binary decisions using $\gamma = 0.3$ for the experiment where RPi2 starts a UDP Flood attack, which lasts for 10 seconds

A. *Traffic Generation*

During normal operation, when there is no attack, each of the RPi1 and RPi2 devices continuously generates normal IP packet traffic containing the device's CPU temperature and transmits it to the server every 1 second using the UDP protocol.

The attack traffic generator exploits the public repository MHDDoS [28], which contains 56 methods for generating different types of DoS attacks that can be directed toward the transport and application layers of the OSI model. Using this repository, the user may provide the type of attack, target IP address, proxy, number of threads to use, attack duration, requests pre-connection (RPC), and debug mode, configured to ensure that the network bandwidth is flooded with a large number of packets that delay or stop communication between devices.

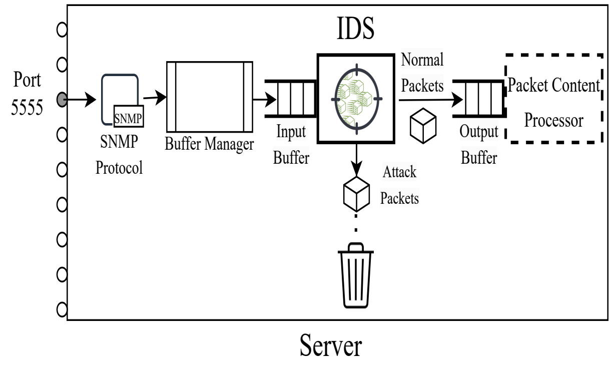

Fig. 5. Schematic system organization of the server that supports the IDS based attack detection and mitigation capability. Mitigation is based on triggering packet drop decisions for all packets that enter the IDS Input Buffer (in this figure) as soon as the IDS has detected a majority of attack packets among the most recent M packets. After the Input Buffer has emptied, the IDS will resume its testing for incoming packets. In our experiments we have taken $M = 20$.

We tested the script, and it showed effective performance in generating aggressive, high-impact attack traffic.

During our experiments, RPi1 generates only normal traffic, while the compromised device RPi2 generates both normal and attack traffic via random sampling. In particular, every 1 second, it initiates a UDP Flood attack with a probability of 0.10 or sends one normal traffic packet with a probability of 0.90. As we perform two different experiments to analyze the changes in the behaviour of the system, the during of the initiated attack is first set to 10 seconds, then to 60 seconds.

B. Experiment I : The UDP Flood Attack Lasts 10 Seconds

In Figure 6, we display an example of a UDP Flood attack effect on the server, where the RPi2 device starts targeting the server with attack traffic at the 99th second to disrupt normal traffic on the network. The figure shows that an intense flow of attack packets arrive in 10 second interval with 1032 byte packets, while under normal operating conditions flow rates are on average of two small packets per second.

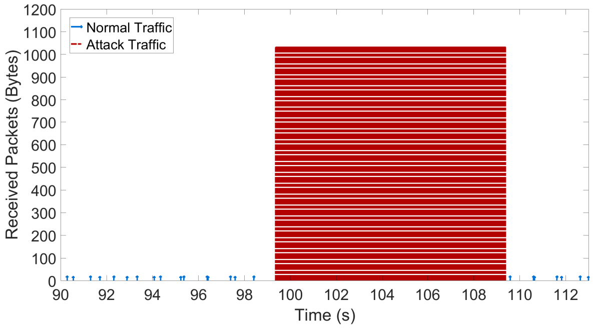

Fig. 6. The difference between the form of the normal and attack traffic on the server when it is targeted by a UDP Flood attack.

To examine the UDP Flood effect on the server, we also conducted several experiments by increasing the duration of the attack in the subsequent experiments. Figure 7 shows the resulting packet queue length at the server, and displays the sharp rise in the number of packets waiting to be analyzed (for attack detection) in front of the IDS, and also the gradual decreases of the queue length once the attack ceases.

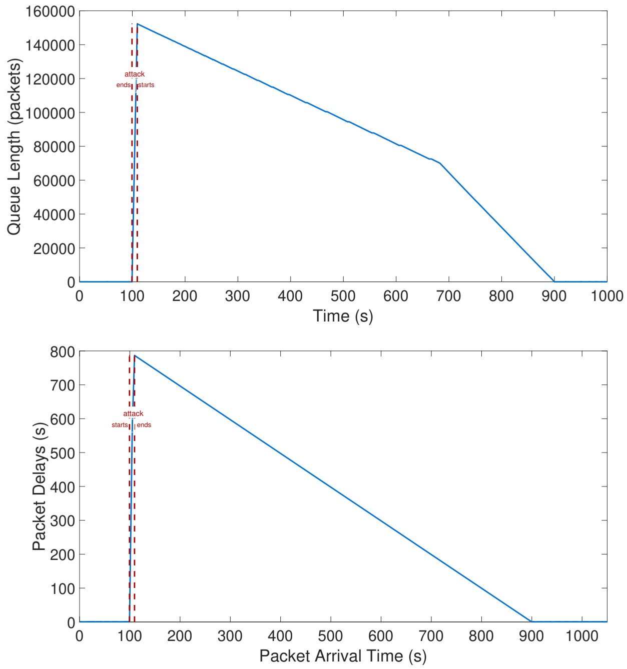

Fig. 7. The top figure shows the queue length infront of the IDS in an attack whose duration is10 seconds, and the vertical red dashed lines show the active duration of the attack originating in the compromised device RPi2. The bottom figure plots the packet delay before the packet is processed by the IDS.

C. Experiment II : The UDP Flood Attack Lasts 60 Seconds

Figure 8 shows the effect of the attack when it lasts for 60 seconds on the packet processing rate (y-axis in packets/sec) of the server. We observe that the server is intermittently paralyzed as the attack continues, so its processing rate drops intermittently to zero.

From Figures 7 and 8, we see that although the attack lasts only 10 seconds in the first experiment, it floods the packet queue such that the IDS completes the analysis of the accumulated packets over a very long 15 minute period, and it can take some 5.85 hours when the attack lasts for 60 seconds as in the second experiment. When the duration of the attack is increased to 60 seconds, the IDS becomes intermittently "paralyzed" since the server's four cores are all committed to handling the incoming, and is unable to process packets as shown in Figure 9. After observing the attacker's and server's behavior, we concluded that these severe attack symptoms occur if the attack itself lasts for 60 seconds, as

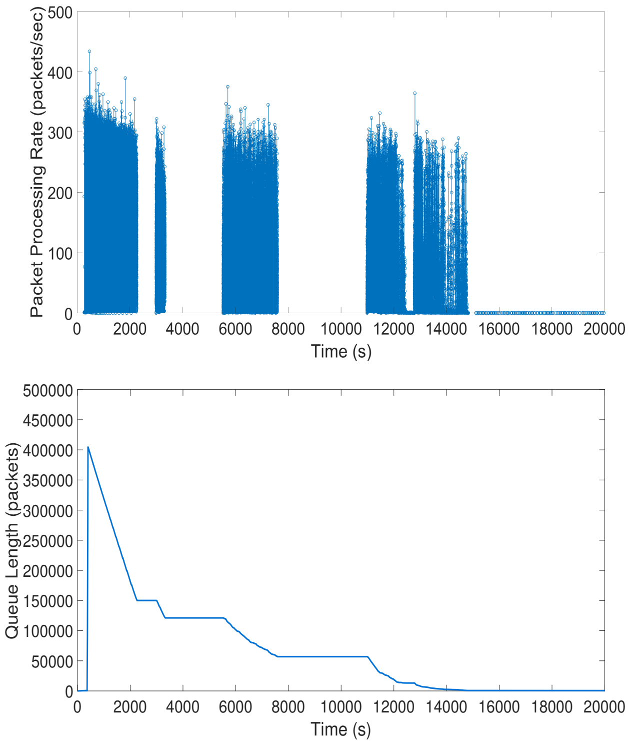

Fig. 8. At the top, the effect of a 60 second UDP Flood attack on the IDS traffic processing rate in packets per second, is shown when the attack duration is 60 seconds. The corresponding packet queue length infront of the IDS is shown at the bottom.

the server receives approximately $408,500$ packets of which $407,796$ are attack packets during this period. Thus in the absence of any mitigation action as per the IDS's decision, the effect of the attack on the server can last much longer than the activity of the attacker.

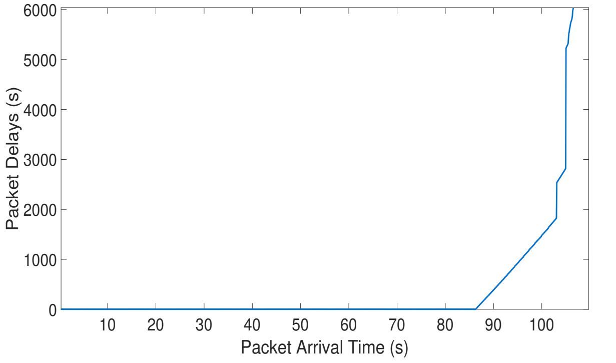

Fig. 9. Packet delay after the packet has been processed by the IDS when the attack duration is 60 seconds.

D. System Behaviour for Experiments I and II with Attack Mitigation

We now present measurements of the system behaviour when mitigation action is taken based on the decision of IDS. Recall that in order to mitigate the impact of an attack, if the IDS detects the majority (more than 10) of the 20 latest packets as attacks, the input buffer is emptied and all incoming packets within the next 30 second window are dropped. This is repeated at the end of the 30 second window.

Figure 10 displays the queue length in the input buffer when the attack mitigation is performed against the UDP Flood attack, which lasts 10 seconds. It is seen that the queue length increases until the IDS processes 20 packets and decides to empty the buffer; the mitigation decision is made just after the attack starts and IDS then waits for a predefined period (in this case 30 seconds) and we observe that the 10 second long attack is mitigated successfully.

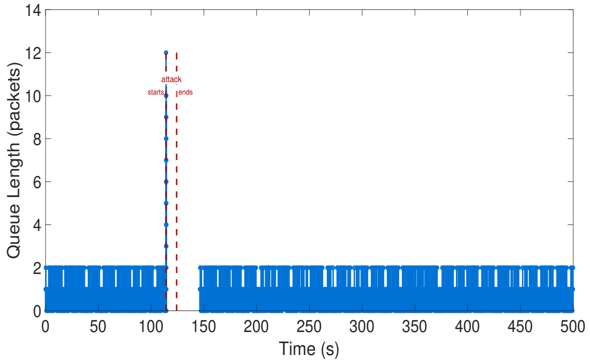

Fig. 10. During the 10 second attack, the decision to drop packets results in subsequent very short packet queue length, avoiding server and IDS paralysis.

Figure 11 displays the queue length when the attack lasting 60 seconds is mitigated: the buffer length increases up to 22 packets, which is small compared to the results without mitigation in Figure 8. During the attack, the mitigation decision was taken twice, and the IDS was not paralyzed. Another mitigation decision occurs between 162 and 192 seconds after an IDS detection event.

VI. Conclusions

IDS are very useful to detect and evaluate network attacks, but are often evaluated under ideal off-line conditions, when the effect of the attack itself is not felt on the server which is used to evaluate the accuracy or quality of an IDS.

Thus in this paper we have installed an AADRNN based IDS on a server which receives traffic via Ethernet from devices in a LAN network test-bed. Realistic UDP Flood attack packets have been installed one one of the network devices, and experiments were run where the Flood attack was directed at the server. During a short 10 second attack, it was observed that the IDS was able to accurately detect the attack, but that

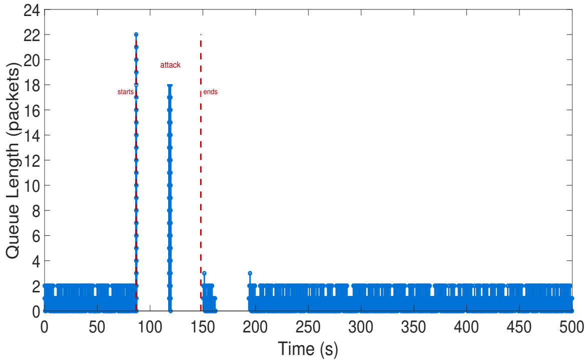

Fig. 11. The figure shows that during the attack's 60 seconds, mitigation decision occurs twice. Another mitigation decision following detection between 162 and 192 seconds.

long packet queues accumulated at the server. During longer 60 second attacks we observed that the IDS soon became unable to carry out attack detection because of the congestion, while the server became intermittently paralyzed due to the server's overload caused by the attack.

This led us to design a fast mitigation technique, which takes a decision very rapidly based on a small number of 20 successive packets. If an attack is detected then all incoming packets are dropped. The traffic is allowed to re-enter the serever's port after some time, and the IDS again takes a mitigation decision based on the first 20 consecutive packets and the procedure is repeated. We saw that this approach avoided UDP Flood attack based congestion at the server and also allowed the IDS to operate effectively.

In addition to experimentally showing that the installation of an IDS at a server is not sufficient to protect it against the consequences of an attack, and that a highly accurate IDS is by itself no guarantee that an attack will be inneffective, this work shows the value of evaluating an IDS in the context of a real test-bed.

Future work will study optimum mitigation policies that examine several mutually dependent aspects, such as the amount and duration of traffic that needs to be blocked or dropped when an attack is first detected, the frequency with which the IDS should sample and analyze the incoming traffic, and the manner in which blocking and loss of valid (benign) traffic can be minimized when attacking traffic is being blocked or dropped.

References

[1] S. Staff, "Organizations fought an average of 29.3 attacks daily in late 2022," Feb 2023. [Online]. Available: https://www.securitymagazine.com/articles/98958-organizations-fought-an-average-of-293-attacks-daily-in-late-2022

[2] Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/

[3] L. Rajesh and P. Satyanarayana, "Detecting flooding attacks in communication protocol of industrial control systems," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 396–401, 2020.

[4] Y. Al-Hadhrami and F. K. Hussain, "Ddos attacks in iot networks: a comprehensive systematic literature review," *World Wide Web*, vol. 24, no. 3, pp. 971–1001, 2021.

[5] S. Evmorfos, et al, "Neural network architectures for the detection of SYN flood attacks in IoT systems," in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2020, pp. 1–4.

[6] N. Statt, "How an army of vulnerable gadgets took down the web today," October 2016. [Online]. Available: https://www.theverge.com/2016/10/21/13362354/dyn-dns-ddos-attack-cause-outage-status-explained

[7] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, "The impact of DoS attacks on resource-constrained IoT Devices: A study on the Mirai attack," *arXiv preprint arXiv:2104.09041*, 2021.

[8] H. Sinanović and S. Mrdovic, "Analysis of Mirai malicious software," in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2017, pp. 1–5.

[9] M. Antonakakis, et al., "Understanding the Mirai Botnet," in *Proceedings of the 26th USENIX Security Symposium*, 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[10] Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/

[11] A. D. Wood and J. A. Stankovic, "A taxonomy for denial-of-service attacks in wireless sensor networks," *Handbook of sensor networks: compact wireless and wired sensing systems*, vol. 739, pp. 763–780, 2004.

[12] J. Mirkovic, et al., "Accurately measuring denial of service in simulation and testbed experiments," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 2, pp. 81–95, 2008.

[13] M. Nakip and E. Gelenbe, "MIRAI Botnet attack detection with Auto-Associative Dense Random Neural Network," in *IEEE Global Communications Conference (Globecom)*, 2021, pp. 1–6.

[14] J. Mirkovic, S. Fahmy, P. Reiher, and R. K. Thomas, "How to test dos defenses," in *2009 cybersecurity applications & technology conference for homeland security*. IEEE, 2009, pp. 103–117.

[15] O. A. Waraga, M. Bettayeb, Q. Nasir, and M. A. Talib, "Design and implementation of automated iot security testbed," *Computers & security*, vol. 88, p. 101648, 2020.

[16] M. Kaouk, F.-X. Morgand, and 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.

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

[18] C. B. Vellaithurai, S. S. Biswas, and A. K. Srivastava, "Development and application of a real-time test bed for cyber-physical system," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2192–2203, 2015.

[19] A. Ashok, et al., "Testbed-based performance evaluation of attack resilient control for agc," in *2016 Resilience Week (RWS)*. IEEE, 2016, pp. 125–129.

[20] V. K. Singh, R. Sharma, and M. Govindarasu, "Testbed-based performance evaluation of attack resilient control for wind farm scada system," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2020, pp. 1–5.

[21] A. Ghaleb, S. Zhioua, and A. Almulhem, "Scada-sst: a scada security testbed," in *2016 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 2016, pp. 1–6.

[22] A. Tesfahun and D. L. Bhaskari, "A scada testbed for investigating cyber security vulnerabilities in critical infrastructures," *Automatic Control and Computer Sciences*, vol. 50, pp. 54–62, 2016.

[23] B. Reutimann and 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.

[24] S.-U. Park and S.-M. Hwang, “Test bed construction for apt attack detection,” *International Journal of Control and Automation*, vol. 11, no. 4, pp. 175–186, 2018.

[25] R. Arthi and S. Krishnaveni, “Design and development of iot testbed with ddos attack for cyber security research,” in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. IEEE, 2021, pp. 586–590.

[26] A. P. Wright and N. Ghani, “A testbed for the evaluation of denial of service attacks in software-defined networks,” in *2019 SoutheastCon*. IEEE, 2019, pp. 1–6.

[27] P. V. Sontakke and 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.

[28] “MHDDoS - DDoS Attack Script With 56 Methods,” Online, May 2022, accessed: 2023-02-22. [Online]. Available: https://github.com/MatrixTM/MHDDoS

[29] S. Kumar and S. Rai, “Survey on transport layer protocols: Tcp & udp,” *International Journal of Computer Applications*, vol. 46, no. 7, pp. 20–25, 2012.

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

[31] E. Gelenbe, “Random neural networks with negative and positive signals and product form solution,” *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.

[32] O. Brun, Y. Yin, and E. Gelenbe, “Deep learning with dense random neural network for detecting attacks against iot-connected home environments,” *Procedia Computer Science*, vol. 134, pp. 458–463, 2018.

[33] M. Nakip and E. Gelenbe, “Botnet attack detection with incremental online learning,” in *Security in Computer and Information Sciences: Second International Symposium, EuroCybersec 2021, Nice, France, October 25–26, 2021, Revised Selected Papers*. Springer, 2022, pp. 51–60.

[34] E. Gelenbe and M. Nakıp, “Traffic based sequential learning during botnet attacks to identify compromised iot devices,” *IEEE Access*, vol. 10, pp. 126 536–126 549, 2022.

[35] E. Gelenbe and M. Nakıp, “G-networks can detect different types of cyberattacks,” in *MASCOTS’22: 30th International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, IEEE Computer Society*, pp. 1–6.

[36] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.