IoT Network Attack Detection and Mitigation

Erol Gelenbe, Piotr Fröhlich
and Mateusz Nowak
IITIS-PAN, Polish Academy of Sciences
ul. Baltycka 5, PL Gliwice 44100, Poland

Stavros Papadopoulos, Aikaterini Protogerou,
Anastasios Drosou and Dimitrios Tzovaras
ITI-CERTH, GR 57001 Thermi
Thessaloniki, Greece

***Abstract*—Cyberattacks on the Internet of Things (IoT) can cause major economic and physical damage, and disrupt production lines, manufacturing processes, supply chains, impact the physical safety of vehicles, and damage the health of human beings. Thus we describe and evaluate a distributed and robust attack detection and mitigation system for network environments where communicating decision agents use Graph Neural Networks to provide attack alerts. We also present an attack mitigation system that uses a Reinforcement Learning driven Software Defined Network to process the alerts generated by the attack detection sysem, together with Quality of Service measurements, so as to re-route sensitive traffic away from compromised network paths using. Experimental results illustrate both the detection and re-routing scheme.**

***Index Terms*—IoT Security, Graph Neural Nets, Cognitive Packet Network, Random Neural Networks, Software Defined Networks**

I. Introduction

The IoT [1] has the potential to improve the critical processes that are at the heart of our socio-economic systems [2], [3]. However, it creates raises risks that go way beyond the individual technologies such as the Internet, wireless networks and machine to machine systems [4], [5]. In addition to risks related to system malfunctions [6], quality of service (QoS) failures, and excessive energy consumption, the theft and tampering of data, conventional network attacks and attacks that deplete the energy of autonomous sensors and actuators also need to be considered [7]–[13]. Since IoT devices can carry out real-time measurements and controls much faster than human reaction times, we must design IoT networks that *both detect and mitigate* security risks automatically and adaptively, while preserving Quality of Service (QoS), and energy efficiency [6], [14]. Thus we propose an autonomic [15] scheme offering (a) distributed attack detection based on deep learning (DL) and graph neural networks to achieve high detection probabilities with low false alarm rates [16], [17], and (b) mitigation that exploits network Self-Awareness [18], [19] centered on Software Defined Networks [20] to achieve secure QoS based routing of traffic flows using machine learning and adaptivity [21], [22].

Thus Section II discusses a multi-agent system (MAS) for network attack detection, and summarizes its performance. The overall system architecture for attack detection and mitigation is presented in Section III. The node attack detection probability estimated by MAS is used to compute safer paths in the network using reinforcement learning as described in Sections III-A and III-B. Experimental results are presented in Section III-C, and Section IV presents conclusions and future work.

II. Distributed attack detection

IoT systems are distributed have a heterogeneous structure which is an additional challenge for real-time anomaly detection [23], [24]. Thus the distributed MAS for detecting attacks monitors the network traffic in a distributed manner, and outputs to the novel routing system described in Section III, to mitigats attacks with a SDN based routing engine. The MAS's mutually communicating multiple agents can improve its robustness by incorporating redundancy in the detection algorithm [25]. The MAS also offers scalability, since its modularity allows new agents to be added if the IoT network grows, and agents exchange information [26] in a structure inspired by Graph Neural Networks [16], [17].

The structure of the IoT network is reflected by the graph $G(V,E)$, where V corresponds to the set of nodes of the IoT network, and $E \subset V \times V$ is a set of edges which represent the nodes which communicate (directly or indirectly) with each other through the IoT network. The nodes can represent sensors or actuators, edge nodes, servers or routers in the IoT network. We associate a real-valued feature vector $x_i \in \mathbb{R}^{N_V}$ to each $i \in V$, where N_V is its length. Similarly we associate the feature N_E-vector of real numbers $e_{ij} \in \mathbb{R}^{N_E}$ with each edge $(i,j) \in E$. An example of the features for the nodes and edges is given in Table I. Measurements that collect the feature vector parameters are taken in the IoT network during successive time slots $[(t-1)T, tT$ where T is the slot length and t is the slot index. The slots are long enough to provide representative data, but short enough to reflect time variations in the system. Thus all feature vectors are also associated with individual slots and successive values. Thus $x_i^{t,k}$ the $k-th$ successive value of x_i within the $t-th$ slot, while $e_{ij}^{t,k}$ is the $k-th$ successive value of e_{ij} in the $t-th$ slot. We will denote by e_{ij}^t and X_i^t, respectively, the feature vector values at the *end* of the $t-th$ slot, while e_{ij}^0 and X_i^0 are their values when the measurement system starts to operate and the first slot begins. The MAS uses four Deep Neural Networks (DNNs):

- The EDNN (edge DNN) which undertakes the update:
$$e_{ij}^{k+1,t} \leftarrow EDNN(x_j^{k,t}, x_i^{k,t}, e_{ij}^{k,t}) \ . \tag{1}$$
EDNN uses an edge's current features, and the features of the two nodes at its edges, to update its features.

- The NDNN (node DNN) which undertakes the update:
$$x_j^{k+1,t} \leftarrow NDNN(\hat{e}_j^{t-1}, \hat{x}_j^{k,t}) \ , \tag{2}$$
and updates a given node's features using the average value of the features of the nodes with which it communicates and of the related edges, with:
$$\hat{e}_j^{t-1} = \sum_{i \ s.t. \ (i,j) \in E} \frac{e_{ij}^{t-1}}{m_j} \ , \hat{x}_i^{k,t} = \sum_{i \ s.t. \ (i,j) \in E} \frac{x_i^{k,t}}{m_j}, \tag{3}$$
where $m_j = |\{i \ s.t. \ (i,j) \in E\}|$ is the number of neighbours of $j \in V$, and $s.t.$ stands for "such that".
- The third DNN, CLN builds $p_{Ni}^{k,t}$, the probability that node i is compromised, only using its own feature vector:
$$p_{Ni}^{k,t} \leftarrow CLN(x_i^{k,t}). \tag{4}$$
Finally the fourth DNN, CLNEI builds $p_{Nij}^{k,t}$ the probability that i determines that its neighbour j is compromised:
$$p_{Nij}^{k,t} \leftarrow CLNEI(e_{ji}^{k,t}, e_{ij}^{k,t}, x_j^{k,t}). \tag{5}$$

These four networks constitute the node agent, and are duplicated in each node, and can be trained off-line. They operate in each node separately and asynchronously. Starting from feature vectors from data gathered during the previous time slot, they update the decision probabilities and communicate their updated feature values and decison probabilities to their neighbours. These computations are shown schematically in Figures 1 and 2.

TABLE I
AN EXAMPLE OF THE NETWORK FEATURES USED FOR ANOMALY DETECTION. IN THIS EXAMPLE THE NUMBER OF FEATURES IS $N_V = 5$ (NODE FEATURES) AND $N_E = 3$ (EDGE FEATURES).

#	Target	Feature description
1	edge/node	Average number of packets sent
2	node	Average number of packets received
3	edge/node	Average number of bytes sent
4	node	Average number of bytes received
5	edge/node	Average connection duration

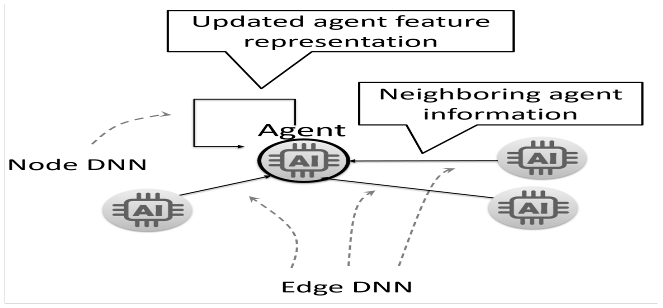
Updated agent feature representation
Neighboring agent information
Agent
Node DNN
Edge DNN
AI

Fig. 1. The figure shows Edge Deep Neural Networks (EDNN) that inputs the previous edge feature vector and adjacent node feature vectors, and updates the its dege features, and a Node DNN (NDNN) that aggregates the information from updated edge features, and updates the features of the node.

The mutiple iterations of these operations represented by the integer k, allow nodes and edges to update and exchange

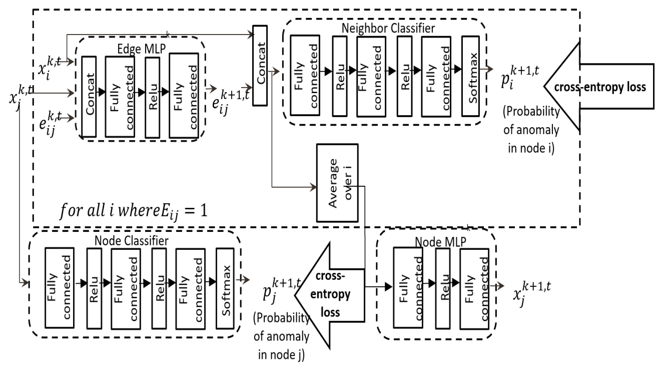

Fig. 2. The figure shows the operation of the Edge and Node Deep Neural Networks (EDNN and NDNN) in the anomaly detection system that use Deep Neural Networks implemented with Multi-Layer Perceptrons. The EDNN takes the feature vectors of neighboring nodes and updates the edge feature vector, while the NDNN uses the adjacent updated edge feature values and updates the feature values of the specific node. Edge features predict anomalies in the neighborhood, while node features detect anomalies in the specific node.

information multiple times within each time interval as shown in Figure 2. The entire network of nodes is trained in a supervised manner using the back-propagation algorithm with cross-entropy as the cost function for classification.

In the system that we have described, each node agent can perform anomaly detection not only on itself, but also with regard to its neighbouring nodes. This redundancy improves the algorithm's robustness in cases where some agents may fail, since the agents at neighbouring nodes may still detect anomalies that occur at their neighbours. Finally, to combine the overlapping decisions of different agents into a single decision for each node in the IoT network, we use a simple aggregation method, where a node is considered anomalous if at least one agent has reported it as being anomalous. More sophisticated aggregation schemes can be considered in future work.

To evaluate the proposed approach, we have uses a simulated infiltration attack, where the attacker tries to infiltrate the network by scanning a range of IP addresses in order to run services, and performs a dictionary attack in order to find vulnerable IoT devices. The resulting Receiver Operating Characteristic (ROC) is shown in Figure 3. The overall results are summarized in Table II. The metrics used forthe evaluation are the Area Under the Curve (AUC) score, the detection accuracy, the utilized Bandwidth, and the Power consumption [27]. For the first two metrics, which measure detection efficiency, the proposed approach outperforms all other methods we have tested for anomaly detection, achieving Area Under the Curve (AUC) score and accuracy of 0.99, compared to 0.97 for the second (random forest) and third (decision tree) classifiers. With respect to the last two metrics, i.e. Bandwidth and Power consumption, the proposed decentralized approach greatly reduces the bandwidth required for monitoring, which in turn reduces the power that is consumed. However, the execution time and the power consumed at each node will determine the energy consumed by our approach, so that a slower low power approach may consume more energy than a very fast method that uses higher power.

TABLE II
EVALUATION OF PROPOSED APPROACH AND COMPARISON WITH OTHER METHODS. THE PROPOSED APPROACH OUTPERFORMS OTHER METHODS THAT WERE TESTED FOR ANOMALY DETECTION, NAMELY, RANDOM FOREST, SUPPORT VECTOR MACHINE (SVM), DECISION TREE, AND AUTOENCODER.

Method	AUC	Accuracy	Bandwidth	Power
Proposed	**0.99**	**0.99**	**452.26 bits/s**	**353 W**
Random Forest	0.96	0.96	33985.7 bits/s	353.01 W
SVM	0.98	0.98	33985.7 bits/s	353.01 W
Decision Tree	0.97	0.97	33985.7 bits/s	353.01 W
Autoencoder	0.69	0.70	33985.7 bits/s	353.01 W

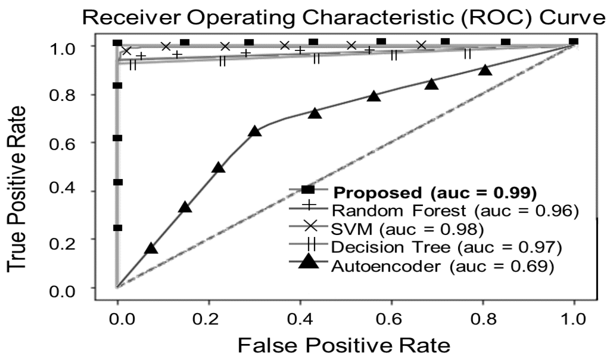

Fig. 3. Receiver Operating Curve curve of detection scheme for the simulated infiltration attack scenario, using several attack detection mechanisms. The approach we propose has the highest Area Under the Curve (AUC) score.

III. SYSTEM ARCHITECTURE AND ROUTING ENGINE

The Architecture of the SerIoT system is shown in Figure 4, with interconnected smart forwarding engines (SFE) that are connected to sfixed or mobile IoT devices, IoT gateways, and to Cloud Servers which may also be Fog servers. SFEs may be connected to Honeypots (H) whose role is to attract and interpret attacks. Specific sofware at IoT devices and gateways may be used to detect attacks [28], but here we us the decisions provided by the distributed attack detector of Section II. QoS, Energy and Security are monitored and forwarded to “smart controllers and routing engines” (SRE) which operate as OpenFlow SDN Controllers to choose paths and download the to the SFEs [29], [30].

The SRE uses the Cognitive Packet Network (CPN) routing algorithm [31], implemented with the Random Neural Network (RNN) and Reinforcement Learning [32] which has also attracted interest from industry [33]. It extends a standard SDN network using the SRE, with SFEs which are extensions of SDN forwarders, and the Monitoring and Anomaly Detection (MAD) module which detects potential threats from data collected by SerCPN, Active Honeypots that attract attacks, deflect them to safe IP locations, and inform the SRE, and local attack detectors at nodes and gateways [28].

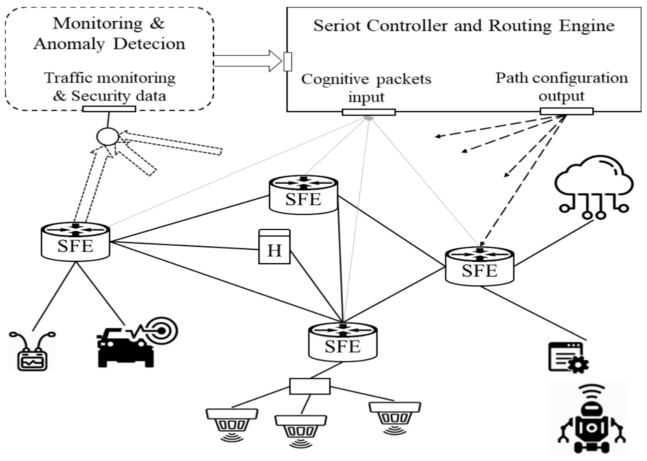
Monitoring & Anomaly Detecion
Traffic monitoring & Security data
Seriot Controller and Routing Engine
Cognitive packets input
Path configuration output
SFE
SFE
H
SFE
SFE

Fig. 4. Overview of the Secure IoT network System Architecture.

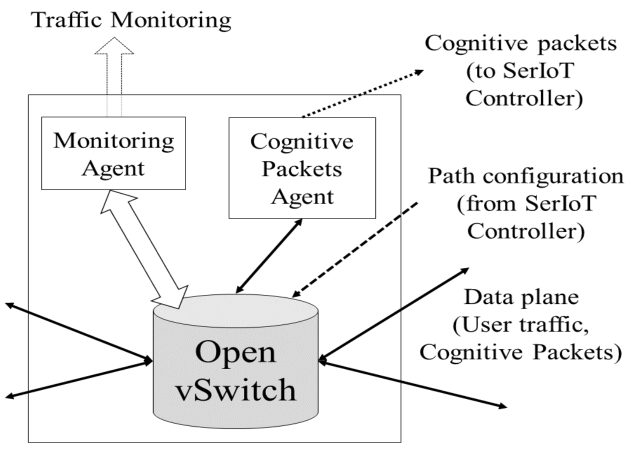
Traffic Monitoring
Cognitive packets (to SerIoT Controller)
Monitoring Agent
Cognitive Packets Agent
Path configuration (from SerIoT Controller)
Data plane (User traffic, Cognitive Packets)
Open vSwitch

Fig. 5. SFE: the SerIoT Forwarding Element.

Each SFE, shown schematically in Figure 5, switches SDN flows according to the OpenFlow protocol. In addition to payload traffic, SFEs also forward smart packets (SPs) which gather security, QoS and energy usage data from the SFEs, IoT devices and gateways. Each SFE has a Cognitive Packet Agent (CPA) that unpacks the SP, adds its own data to the list stored inside, packs it again and forwards it to the SFE. SPs travel over paths, carrying information provided bythe SFEs on the path. When a CPA recognizes that a SP has attained the end of its path, it encapsulates it and forwards it to the corresponding SRE, where its data is unloaded into the local Network State Database (NetStatDB). SFEs can also forward data that is monitored, such as packet counters or byte counters) to the MAD at the SRE.

Each SRE is based on ONOS [34] and its software implemented as an ONOS application with the three main modules shown in Figure 6. The heart of the system is the Cognitive Routing Module (CRM) that implements decision taken by a RNN [35] with Reinforcement Learning for path selection based on QoS, security or energy consumption in the network. The MAD detects attacks at nodes using MAS of Section II.

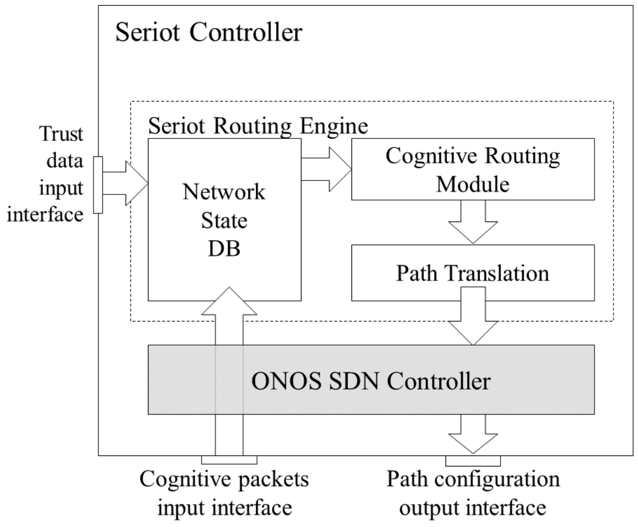

Fig. 6. SRE: The SerIoT Controller and Routing Engine.

Other attack detection methods will also be considered in futire work [28].

The SRE selects paths based on a Goal Function $G(f,P)$ which has non-negative real values and which must be minimized, where f denotes the packet flow to or from an IoT device or end-user software, and P denotes a path travelled by and the df, and the MAS in Section II provides the probability p_i that node i is under attack. For some SFE or network node i, the *Trust Level* $T(f,i)$ is non-negative number that is high when i is not deemed secure enough to convey the flow secure f. Also, $S(f,i)$ is defined as the sensitivity of f to attacks at node i.

A. Linking $T(f,i)$ to the p_i from MAS

Let $A>0$ be a large positive constant used so that $T(f,i)$ may take values comparable to QoS values such as the delay of links, and p_i is the probability that an attack is detected at node i by MAS. Then $T(f,i)=A.(1-p_i)$ is the security level of f related to node i. Let $S(f,i)$ be the sensitivity of f to the security of e. The *Insecurity Factor* $I(f,i)$ is then used to "separate" e and f:

$$I(f,i)=[S(f,i)-T(f,i)]^+, \tag{6}$$

where use the notation $[X]^+=X$ if $X>0$, and $[X]^+=0$ if $X<0$. If we take $S(f,i)=A$, then $I(f,i)=A.p_d(i)$. and we see that as p_i increases, the "security cost" incurred by f as it travels through i increases. the "Insecurity Factor" that relates flows to paths, is:

$$I(f,P)=\sum_{i\in P} I(f,i). \tag{7}$$

When less attention is paid to security, we may take the smaller value $I(f,P)=\max_{i\in P} I(f,i)$.

Let $L(f,p)$ be the packet loss ratio, and $D(f,P)$ be the forwarding delay for a packet of f on path P, while J_i is the energy consumption per packet at node i. The packet retransmissions due to packet losses [31], [36] result in:

$$Q(f,P)=\frac{D(f,P)}{1-L(f,P)},\ hence \tag{8}$$

$$G(f,P)=\begin{cases} I(f,P), & I(f,P)\geq\theta, \\ Q(f,P), & if I(f,P)<\theta, \end{cases} \tag{9}$$

where $\theta\ \geq 0$ is a security threshold that can be chosen based on the importance of security considerations for this system. $G(f,i)$ or $G(f,P)$ are quantities to be minimized, but Reinforcement Leaning (RL) requires a "reward" $R(f,i)$ that should be maximized, where:

$$R(f,i)=\frac{1}{G(f,i)},\ R(f,P)=\frac{1}{G(f,P)}. \tag{10}$$

B. Reinforcement Learning

The metrics that feed into the quantity $R(f,i)$ arecollected via measurements, except the ones that are initially fixed, namely θ, $S(f,i)$ and the parameters such as α, β, γ describing the relative importance of different factors. Therefore the RL based routing scheme to improve network security, QoS and energy consumption, collects at each node i the quantity $G(f,i)$ and hence $R(f,i)$, at successive arrivals of a SP packet to an SDN controller. The SP will collect bring back the relevant data for $R(f,i)$ concerning each node i that the SP has visited, to the SDN router that exploits the RL algorithm to compute a "next hop" for SPs. Let the integer l refer to the $l-th$ value of the reward $R_l(e,f)$ computed by the SDN router for the node i and flow f. The RL algorithm will first compute the quantity:

$$T_l=aT_{l-1}+(1-a)R_l,\ 0\leq a<1, \tag{11}$$

that describes the historical behaviour of the reward, and tells how well the network has been doing. The RL algorithm will then compute a set of RNN [35] weights as follows.

For an N node RNN, where N is the number of outgoing links for node i, we associate with each outgoin link i a neuron whose state is represented by the "excitation probability" q_i of the RNN. The RNN weights are real numbers $W^+_{ij},\ W^-_{ij}\ \geq 0$ for $i,\ j\in\{1,\ ...\ ,N\}$. From RNN theory [35] we know that:

$$q_j=\frac{\lambda^+_j+\sum_{l=1}^N q_l W^+_{lj}}{r_j+\lambda^-_j+\sum_{l=1}^N q_l W^-_{lj}}, \tag{12}$$

where $r_j\ =\ \sum_{l=1}^N[W^+_{jl}+W^-_{jl}]$ is the "total firing rate" of the neuron j. $\lambda^+_j,\ \lambda^-_j$ are, respectively, the arrival rate of excitatory and inhibitory spikes to neuron j from outside the neuron i, which are set so that when all connection weights are equal, then all neurons in the network have an excitation probability of $q_j=0.5$.

Let k be the index of the neuron for which, after the $v-1$-th update of the RNN we have $q_k=max\{q_1,\ ...\ q_N\}$. Also save the current value $r_j\leftarrow\sum_{l=1}^N[W^+_{jl}+W^-_{jl}]$. Note that the node from which a SP entered the node where the next-hop decision is being taken will not be used as the next-hop, so that the decision at a given node will select one outgoing link among $N-1$. The RNN's weights are updated as follows:

$$If\ R_l \geq T_{l-1}:\ \forall\ j \neq k,\ j \neq i(Previous), i \neq k,$$
$$W^+_{ik} \leftarrow W^+_{ik} + R_l,\ W^-_{ij} \leftarrow W^-_{ij} + \frac{R_l}{N-2}, \qquad (13)$$
$$If\ R_l < T_{l-1}:\ \forall\ j \neq k,\ j \neq i(Previous), i \neq k,$$
$$W^-_{ik} \leftarrow W^-_{ik} + R_l,\ W^+_{ij} \leftarrow W^+_{ij} + \frac{R_l}{N-2}, \qquad (14)$$

where we divide by $N-2$ since we are excluding $i(Previous)$ from which the SP initially arrived, also not increasing the inhibitory weights of the winner node when $R_l \geq T_{l-1}$, nor increasing the excitatory weights of the loser node when $R_l < T_{l-1}$. We then also renormalize the weights as follows:

$$r^*_j \leftarrow \sum_{l=1}^{N} [W^+_{lj} + W^-_{lj}], \qquad (15)$$
$$W^+_{lj} \leftarrow W^+_{lj} \frac{r_j}{r^*_j},\ W^-_{jl} \leftarrow W^-_{jl} \frac{r_j}{r^*_j}\ . \qquad (16)$$

Finally we calculate all the q_j from equation (12), to select the new output link for flow f at node i by selecting the new output link k^* with $q_{k^*} = max\{q_j,\ j \neq i,\ 1 \leq j \leq N\}$.

C. *Experimental Results*

Experiments were run on a network with several with SFEs composed of Linux boxes with ARMv8 processors (1,4 GHz, 4 GBit Ethernet, 2.4GHz and 5GHz 802.11b/g/n/ac WiFi interface). They were configured to use Ethernet as the data plane interface shown in Figure 7, and WiFi for management, monitoring and for communications with the SRE. In the figure $s1,\ ...\ ,\ s7$ denote SFEs, $h1,\ ...\ ,\ h4$ are IoT devices each with a 633MHz MIPS processor, 100Mb/s Ethernet port, and 2.4Mhz WiFi connection used as a management port, the SRE is a workstation connected by WiFi to the test-bed, and the MAD is installed on a separate workstation connected by Ethernet port to the SRE, and by WiFi to the test-bed. The type of experiments we run are represented by the measured event trace shown in Figure 8.

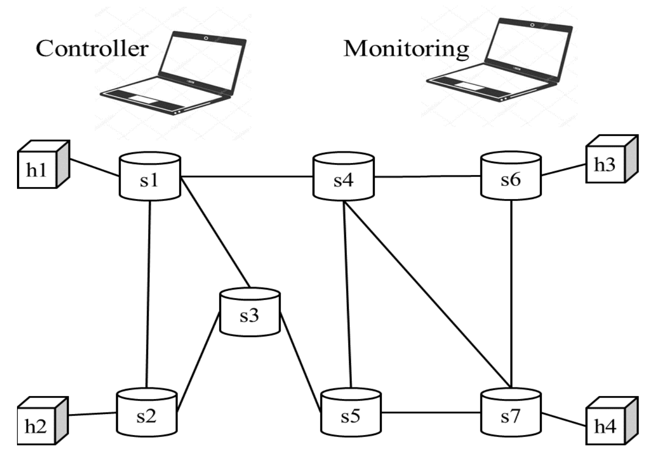

Fig. 7. Topology of the test-bed.

In our experiments, every distinct pair of IoT devices in $\{h1,\ ...\ ,h4\}$ forward 20 packets/sec or roughly $20-40$ Kb/sec with 12 ongoing connections so that each packet rate is compatible with IoT connections monitoring temperature or water flow in pipes, etc. SPs are generated by every edge SFE at 10 packets/sec. SRE management traffic includes OpenFlow commands, link and topology discovery packets, and traffic statistics. Management packet traffic through SFEs measured using the Wireshark packet analyzer was four to five times higher than SP traffic.

The experiments illustrate the system's aptitude to be Self-Aware and adapt, and we measure the SRE's reaction time to abrupt changes in the security conditions expressed by the trust level for connections, and track changes to parameters R_l and T_{l-1} given in (11) for the Reinforcement Learning Algorithm's successive steps l. The SRE was programmed to change network paths every 5 seconds, so that the experimental results we present are limited by this constraint that has been placed to avoid frequent changes that may increase system overheads. The effect of changing the trust $TF(.,.)$ is shown in Figure 8. The quantity that is plotted is the proportion of the time it takes the SRE to respond to a large increase of 100 in the value of $TF(f,i)$ for a node i on the path that is currently used. We see that the reaction tie is on average around 1 second, waitha maximum value around 2 seconds.

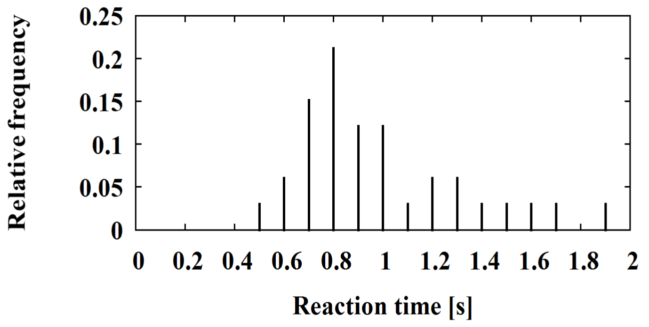

Fig. 8. Histogram of routing engine reaction time to a sudden substantial change in the Trust Metric $TF(.,.)$.

IV. Conclusions and Future Work

In this paper we have described a system that detects node attacks in an IoT network using a deep learning based Mulltiple Agent System, and exploits attack detection in order to automatically mitigate the attacks by re-routing sensitive traffic using Reinforcement Learning, while also taking into consideration the QoS of different network paths. We have also provided a preliminary evaluations of the performance of both the attack detection and mitigation system. In future research, additional measurements, fine tuning of parameters, and experiments will be conducted to better evaluate the interaction of QoS and security in complex adaptive IoT networks. Using methods from diffusion processes [37], [38], we will investigate the transients due to SDN based frequent route updates, in response to potential attacks and changes in QoS. We will also test locally operating anomaly and attack detection software at nodes to reduce computation times for anomaly detection, and improve the response of the system to QoS and security changes, while possibly reducing the accuracy offered by the proposed network-wide anomaly detection scheme.

Acknowledgement

We thank the EC H2020-IOT-2017 Program for its support of the *SerIoT Research and Innovation Action* via Grant Agreement No. 780139.

References

[1] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[2] H. E. Melcherts, "The internet of everything and beyond," *Human Bond Communication: The Holy Grail of Holistic Communication and Immersive Experience*, p. 173, 2017.

[3] H. Elhammouti, E. Sabir, M. Benjillali, L. Echabbi, and H. Tembine, "Self-organized connected objects: Rethinking qos provisioning for iot services," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 41–47, 2017.

[4] R. Ratasuk et al., "Recent advancements in M2M communications in 4G networks and evolution towards 5G," in *Proc. 18th IEEE International Conference Intelligence in Next Generation Networks (ICIN)*, Paris, France, 2015, pp. 52–57.

[5] E. Gelenbe, "Preface: Current research on cybersecurity in europe," in *Proceedings of the 2018 ISCIS Security Workshop: Recent Cybersecurity Research in Europe*, E. Gelenbe, P. Campegiani, T. Czachorski, S. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds., vol. 821. Lecture Notes CCIS, Springer Verlag, 2018.

[6] E. Gelenbe, J. Domanska, P. Frohlich, M. Nowak, and S. Nowak, "Self-aware networks that optimize security, qos and energy," *Proceedings of the IEEE, accepted for publication*, vol. 108, no. 7, 2020.

[7] M. Pirretti et al., "The sleep deprivation attack in sensor networks: Analysis and methods of defense," *International Journal of Distributed Sensor Networks*, vol. 2, no. 3, pp. 267–287, 2006.

[8] X. Lu, M. Spear, K. Levitt, N. S. Matloff, and S. F. Wu, "A synchronization attack and defense in energy-efficient listen-sleep slotted mac protocols," in *Emerging Security Information, Systems and Technologies, 2008. SECURWARE'08. Second International Conference on*. IEEE, 2008, pp. 403–411.

[9] E. Gelenbe and Y. M. Kadioglu, "Energy life-time of wireless nodes with and without energy harvesting under network attacks," in *Proceedings of the 2018 ISCIS Security Workshop: Recent Cybersecurity Research in Europe*, E. Gelenbe, P. Campegiani, T. Czachorski, S. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds., vol. 821. Lecture Notes CCIS, Springer Verlag, 2018.

[10] K. Kalkan and S. Zeadally, "Securing internet of things (iot) with software defined networking (sdn)," *IEEE Communications Magazine*, November 2017.

[11] A. Collen et al., "Ghost: Safeguarding home iot environments with personalised real-time risk control," in *Recent Cybersecurity Research in Europe: Proceedings of the 2018 ISCIS Security Workshop, Imperial College London*, vol. Lecture Notes CCIS 821. Springer Verlag, 2018.

[12] D. He, S. Chan, Y. Qiao, and N. Guizani, "Imminent communication security for smart communities," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 99–103, Jan 2018.

[13] E. Gelenbe, J. Domanska, T. Czachórski, A. Drosou, and D. Tzovaras, "Security for internet of things: The seriot project," in *2018 International Symposium on Networks, Computers and Communications, ISNCC 2018, Rome, Italy, June 19-21, 2018*. IEEEXplore, 2018, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ISNCC.2018.8531004

[14] G. Baldini, P. Fröhlich, E. Gelenbe, Hernandez-Ramos, J. Luis, M. Nowak, S. Nowak, S. Papadopoulos, A. Drosou, and D. Tzovaras, "Iot network risk assessment and mitigation: The seriot approach," 2020.

[15] S. Dobson et al., "A survey of autonomic communications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 2, pp. 223–259, 2006.

[16] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *arXiv preprint arXiv:1812.04202*, 2018.

[17] P. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," *arXiv*, 2018. [Online]. Available: https://arxiv.org/pdf/1806.01261.pdf

[18] E. Gelenbe, P. Liu, and J. Lainé, "Genetic algorithms for route discovery," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 6, pp. 1247–1254, 2006.

[19] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-things-based smart cities: Recent advances and challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16–24, 2017.

[20] F. Francois and E. Gelenbe, "Towards a cognitive routing engine for software defined networks," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[21] C. Tsarouchis et al., "A policy-based management architecture for active and programmable networks," vol. 17, no. 3, pp. 22–28.

[22] A. Galis, S. Denazis, C. Brou, and C. Klein, *Programmable Networks for IP Service Deployment*. Artech House Inc., 2004.

[23] G. Oke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with bayesian classifiers and the random neural network," in *2007 IEEE International Fuzzy Systems Conference*. IEEE, 2007, pp. 1–6.

[24] F. Khodadadi, A. V. Dastjerdi, and R. Buyya, "Internet of things: an overview," in *Internet of Things*. Elsevier, 2016, pp. 3–27.

[25] J. Qin, Q. Ma, Y. Shi, and L. Wang, "Recent advances in consensus of multi-agent systems: A brief survey," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4972–4983, 2016.

[26] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[27] A. Vishwanath, K. Hinton, R. W. Ayre, and R. S. Tucker, "Modeling energy consumption in high-capacity routers and switches," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 8, pp. 1524–1532, 2014.

[28] 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.

[29] "Openflow switch specification, version 1.3.5," *Open Networking Foundation*, March 2015.

[30] K. Phemius and M. Bouet, "Monitoring latency with openflow," in *Network and Service Management (CNSM), 2013 9th International Conference on*, October 2013, p. 122–125.

[31] E. Gelenbe, "Steps towards self-aware networks," *Communications of the ACM*, vol. 52, no. 7, pp. 66–75, July 2009.

[32] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 33–45, 2018.

[33] S. Salam, "Weight initialization for random neural network reinforcement learning (appl. no. 15/718,901)," *United States Patent Application US2019/097912AI*, March 2019.

[34] P. Berde et al., "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620744

[35] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[36] E. Gelenbe and T. Mahmoodi, "Energy-aware routing in the cognitive packet network," *Energy*, pp. 7–12, 2011.

[37] X. Mang and E. Gelenbe, "Call admission control in atm using the diffusion model," in *Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference 3*, 1996, pp. 1700–1704.

[38] T. Czachórski, K. Grochla, and F. Pekergin, "Diffusion approximation model for the distribution of packet travel time at sensor networks," in *Wireless Systems and Mobility in Next Generation Internet, 4th International Workshop of the EuroNGI/EuroFGI Network of Excellence*, L. Cerdà-Alabern, Ed., 2008, pp. 10–25.