Collaborative adversary nodes learning on the logs of IoT devices in an IoT network

Sandhya Aneja*, Melanie Ang Xuan En*, Nagender Aneja†
*Faculty of Integrated Technologies, Universiti Brunei Darussalam
†School of Digital Science, Universiti Brunei Darussalam,
Email: {sandhya.aneja, 16b4003, nagender.aneja}@ubd.edu.bn

***Abstract*—Artificial Intelligence (AI) development has encouraged many new research areas, including AI-enabled Internet of Things (IoT) network. AI analytics and intelligent paradigms greatly improve learning efficiency and accuracy. Applying these learning paradigms to network scenarios provides technical advantages of new networking solutions. In this paper, we propose an improved approach for IoT security from data perspective. The network traffic of IoT devices can be analyzed using AI techniques. The Adversary Learning (AdLIoTLog) model is proposed using Recurrent Neural Network (RNN) with attention mechanism on sequences of network events in the network traffic. We define network events as a sequence of the time series packets of protocols captured in the log. The distributed IoT devices can collaborate to cripple our world which is extending to Internet of Intelligence. The time series packets are converted into structured data by removing noise and adding timestamps. The resulting data set is trained by RNN and can detect the node pairs collaborating with each other. We used the BLEU score to evaluate the model performance. Our results show that the predicting performance of the AdLIoTLog model trained by our method degrades by 3-4% in the presence of attack in comparison to the scenario when the network is not under attack. AdLIoTLog can detect adversaries because when adversaries are present the model gets duped by the collaborative events and therefore predicts the next event with a biased event rather than a benign event. We conclude that AI can provision ubiquitous learning for the new generation of Internet of Things.**

***Index Terms*—deep Learning, federated learning, recurrent neural networks, gated recurrent unit, internet of things, adversary**

I. Introduction

The Internet of Things ((IoT) devices are resource-constrained low power devices collecting a large volume of data for IoT applications in healthcare, retail to transportation and manufacturing. The data collected through IoT applications is valuable and huge, therefore, the same data could be a gain for attackers and commercial competitors. Many malwares, and botnets have been observed to compromise the IoT devices by leveraging the vulnerabilities like default passwords to gain the control of IoT devices by Telnet password brute-forcing. Therefore, the attackers can change data in the network or computing system which can also affect the physical state of the device.

Attackers could manipulate the settings of implantable medical devices and may try to harm the patients and these malicious attacks can originate from anywhere in the world [1]. The authors in [2] presented a survey on Internet of things security from data perspectives. Security from data perspectives is important since massive IoT devices are sending data to the Internet for applications including healthcare. Therefore, IoT devices must ensure confidentiality and authenticity for the integrity of the data. Towards the confidentiality of IoT devices, low-cost ciphers are being developed [3], however, many lightweight ciphers are vulnerable to side-channel attacks due to their relatively simple structures [4]. The authenticity of IoT devices has been tried to achieve through swarm attestation formal security model [5]. Static attestation verifies the static binaries on the prover while swarm attestation verify the integrity of a group of provers. In this paper, we propose improved approach for IoT security from data perspective through analyzing the network traffic of IoT devices on Internet using AI analytics. This can optimize the IoT device security techniques to achieve confidentiality and authenticity.

The IoT devices generate large volume of data. While capturing data through IoT, metadata can also be captured to apply AI techniques for IoT network security. Traditional AI techniques were about centralized data. Another AI paradigm called as federated learning (FL) model is trained from distributed systems over the cloud. Here interesting observation for FL is that the learned model over distributed systems can be secured like other encrypted numbers communicated over the Internet [6, 7]. A simple/low-complexity resource allocation algorithm is proposed for a wireless network to support multiple FL groups. IoT devices may be compromised [8]. We propose in this paper to analyze network traffic logs of IoT devices distributed in a network behind the application gateways. This network traffic logged at application gateways can be used to identify compromised devices as well as collaborative adversaries.

Log analysis using the Recurrent Neural Network (RNN) method have been [9, 10] studied to predict future events. Moreover, it is found that RNN outperforms traditional machine learning, Hidden Markov Model (HMM), and Autoregressive Integrated Moving Average Model (ARIMA) methods with comparatively low values of root mean square error and the mean absolute error. An RNN model on the logs of security events from the intrusion protection systems on the web servers [9, 10] is used to predict the steps which an adversary may take to bypass the events. The set of malicious events could be identified due to the high probability of the events in the prediction of the model. Generative Adversarial Networks (GANs) [11] have been used to recreate cyber-attack data and Sweet et al. showed that GANs not only successfully learn to generate realistic alerts, but also reveal feature dependencies within alerts.

An IoT device connects to different servers e.g. DNS, NTP, TSL/SSL for services to upload the data on cloud e.g. first it communicates with DNS to get the domain name of the data server followed by TSL/SSL server for the certificate to encrypt the data and finally with the data server to upload the data using protocols TCP, UDP, HTTP, SSDP etc. The IoT devices log comprises the chronological events of the packets of these protocols. The packets include port numbers, IP addresses, sequence numbers, flags, checksum, window size and domain names. For example, domains such as example.com, example.net, and example.org are frequently requested by Amazon Echo; sub-domains of hp.com and hpeprint.com are seen in DNS queries from the HP printer [12]. Due to this complex structure of IoT log data, it is complicated to analyze logs for any information. We interface IoT log data to the seq2seq GRU RNN model with the attention decoder method to analyze the log for vulnerabilities specifically collaborative nodes in the IoT network. To the best of our knowledge, there has not been any work studying wireless networks supporting FL groups to identify the collaborative adversary nodes in the prior literature.

In ns-2 network simulator, two network scenarios were set up. The trace files log the sequences of network events of the nodes comprising of different types of protocols, packets including port numbers, IP addresses, sequence numbers, flags, checksum, window size, etc. The first scenario was a network without any adversary node while the second scenario was the network with collaborating adversary nodes which were connected through a link layer tunnel as hidden channel for adversary behavior to other nodes.

The Adversary Learning model degrades by 3-4% in the presence of attack in comparison to the scenario when the network is not under attack. A network protocol fixes the packet format in the network traffic of the devices. We observed that the Recurrent Neural Network models - LSTM, GRU etc. are learned with less execution time and better predicting for network problems in comparison to language translation, emotion detection, and fake news detection problems. The authors in [13] used a five-layer Convolution Neural Network for IoT device identification problem. A method [14, 15] is presented at the application gateway to authenticate the IoT device by analyzing the 212 features like TCP src port and TCP dst port from the packet headers of IoT devices logged in the network traffic.

The organization of the paper is as follows: Section III explains the gated recurrent neural network sequence-to-sequence model. The adversary learning system model is explained in Section IV. The Adversary Learning (AdLIoTLog) algorithm is presented in Section V. Experiment Setup, Results, and Analysis is in Section VI. Conclusions is discussed in Section VIII.

II. Our contributions

1) Collaborative adversary events detection was found effective using RNN.
2) The network simulator ns-2 trace files generated for collaborative attack dataset and further interfaced in Pytorch for AI analytics using RNN model.

III. Gated Recurrent Neural Network-Sequence-to-Sequence Model

. We take the RNN-based GRU model [16] for the problem of anomaly detection in an IoT Network defined in this paper. Assume that the IoT network log vocabulary can be expressed as input network event and predicted network event using sequences $x = x_1, x_2, \ldots, x_{|x|}$ and $y = y_1, y_2, \ldots, y_{|y|}$ respectively. The core of the GRU is composed of sequence to sequence model generating predicted network events using input network events sequence through encoder-decoder network. The encoder-decoder network consists of mainly three parts:

1) Encoder
2) Attention Context Vector
3) Decoder

Encoder: An encoder is a stack of many recurrent units where each accepts an element of the input network event sequence say $x = x_1, x_2, \ldots, x_{|x|}$, process the element and pass the state forward. The hidden states h_t are computed as in Equation 1 with the help of current input, previous state, and weights of the network. This is the final hidden state of the encoder that is represented by Equation 1.

$$h_t = f(W^{(hh)}h_{t-1} \ + \ W^{(hx)}x_t) \tag{1}$$

Attention: The context vector aims to encapsulate input network event sequence information to assist the prediction of output network event sequence by a decoder. Context vector acts as an initial hidden state for the decoder. The context vector c_p is computed as in Equation 2, 3 and 4 with the help of previous hidden state h_{t-1}, previous state s_{p-1} , and weights of the network and normalized over the source sequence in Equation 3

$$r_{tp} = v_a^T tanh(W^{(ss)}s_{p-1} + W^{(hh)}h_{t-1}) \tag{2}$$

$$\alpha_{tp} = \frac{exp(r_{tp})}{\Sigma_{t=1}^{|x|} exp(r_{tp})} \tag{3}$$

The source context vector is weighted sum of all source annotations and can be calculated in Equation 4

$$c_p = \Sigma_{t=1}^{|x|} \alpha_{tp} h_t \tag{4}$$

Decoder: A decoder is similar to the encoder as it comprises of many recurrent units cells wherein each cell predicts an element of the output network event at a time step.

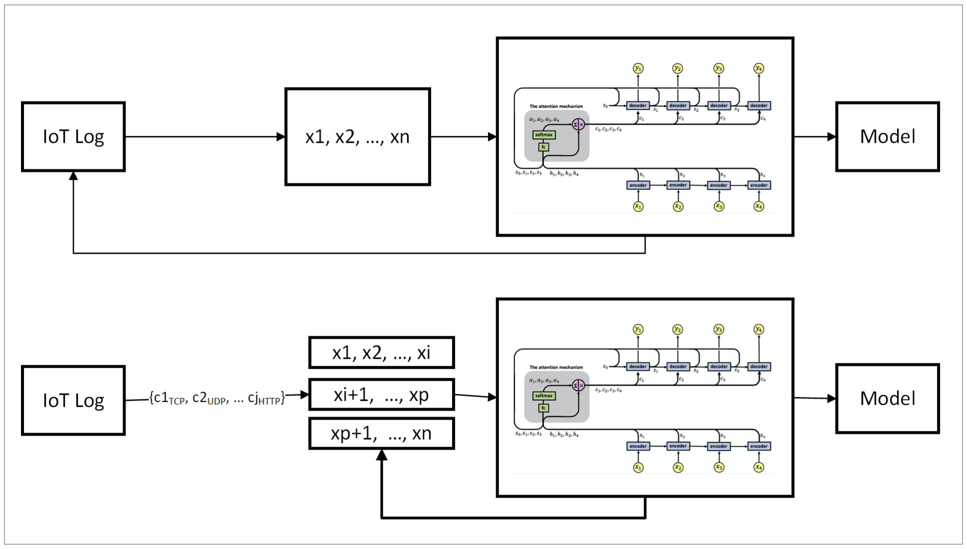

Fig. 1. AdLIoTLog model for collaborating advarsary nodes detection on IoT log

Each recurrent unit cell accepts the previous target state y_{p-1} and source context vector c_p to produce output element and next target hidden state represented by Equation 5.

$$s_p = f(W^{(ss)}.s_{p-1} + \ W^{(sy)}y_{p-1} + \ W^{(sc)}c_p) \tag{5}$$

The j^{th} target hidden state in the decoder is computed using the previous hidden state. The Probability distribution P_j over all elements of network event in target vocabulary is produced from the decoder at any time conditioned on the previous ground truth event y_{j-1}, the source context c_j, and the target hidden state s_j using Softmax Equation 7.

$$t_j = f(W^{(ss)}.s_j + \ W^{(sy)}y_{j-1} + \ W^{(sc)}c_j) \tag{6}$$

$$P_j = softmax(W^s.t_j) \tag{7}$$

Thus, sequence to sequence model can map sequences of varying lengths to each other. The simple decoder only uses the last output or the last hidden state of the encoder. This single vector or the last hidden state has the burden of encoding the entire sequence. However, attention helps the decoder network to focus on different parts of encoder's outputs for every step of the decoder's outputs. Thus, first, we compute attention weights and then these weights are multiplied by encoder output vectors to create a weighted encoded vector that contains information about the input network event sequence to help decoder to choose the right elements of the input network event sequence. The attention weights are computed by a feed-forward layer that uses the decoder's input and hidden state. We used max sentence length of 100 words to train the attention layer.

The output of a GRU is a $|y|$-dimensional tensor y which represents the probability distribution of each network event element of x over the $|y|$ classes.

The GRU is trained over by defining a loss function $L(GRU_\theta(x), y)$ minimized iteratively through backpropagation by varying parameter θ.

Figure 1 demonstrates the identified implementation of the model for an application to optimize. The main aim of our method is to feed the higher context, i.e. splitting the input text into contextual content to increase the model output probability distribution so that it matches with the probability distribution of the ground truth values. The TCP packet, UDP packet and HTTP packet were considered different contexts. This potentially can reduce the gap between training and inference by training the model to handle the situation, which will appear during test time.

IV. Adversary Learning Architecture- System model

An IoT device uses protocols e.g. UDP, TCP, HTTP, TLS, DNS, DHCP, ARP, ICMP, etc. to upload the data on the data server. The network traffic comprises events of the packets exchanged between the application gateway (AG) and the server on the cloud. This network traffic can be logged onto the application gateway (AG) for AI analytics [17].

IoT devices are vulnerable to the scenario where devices connected to a gateway can collaborate to mislead the smart decision of the IoT network. One of the scenarios is wherein IoT devices in two different LANs or locations can collaborate using a high transmission antenna to exchange data say temperature, pressure, humidity. The collaborating IoT devices can then upload the distant location data to the server with own location. Moreover, adversaries can send malicious control data in health applications where it could lead to serious issues.

IoT devices upload the data on cloud connecting to Internet through application gateway. The proposed methods for analyzing IoT devices network traffic record packets at the application gateways [17]. This paper proposes a framework, Adversary Learning (AdLIoTLog), which collects log files from the various application gateways and apply deep learning [10, 11] to detect collaborating nodes which can connect through hidden channel for adversary behavior to other nodes. The Adversary Learning model is a Sequence to Sequence Gated Recurrent Neural Network model that models over the network traffic of IoT devices shown in Figure 2.

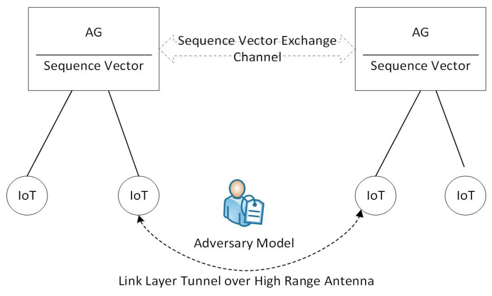
AG
Sequence Vector
Sequence Vector Exchange Channel
AG
Sequence Vector
IoT
IoT
IoT
IoT
Adversary Model
Link Layer Tunnel over High Range Antenna

Fig. 2. Deep Adversary Architecture

AdLIoTLog uses sequences of the packet events of the protocols and subsequences of the packet events logged of each protocol. We model the AdLIoTLog using data with attack and the data without attack for comparison purpose. This model can be trained without actually sharing the data [6, 7]. Initially a global model is aggregated thereafter local model updates and provides local model updates to an aggregator. An aggregator combines all local model updates (AG_1 with AG_2) and construct a new global model (AG_1 with AG_2 and $\ldots AG_n$). Edge devices query the aggregator for any adversary in the network [18].

A. Model Construction

The objective of this work is to propose a model to detect malicious nodes using Gated Recurrent Unit (GRU) Recurrent Neural Network (RNN) model. We show that AI analytics on the set of network events can identify collaborating nodes in an IoT network that can connect through hidden channels for adversary behavior to other nodes.

The GRU RNN model keeps track of dependencies among elements in the sequences and therefore in the problem of predicting sequences of network events, the GRU is set to learn the set of pairs (x, y) where x is an input sequence of events and y is the expected next sequence of network events.

Let S be a set of p malicious nodes represented by $m_1, m_2, ..m_p$. Let S_1 is the set of events of m_1 malicious nodes and S_p is the set of events of m_p malicious nodes.

The node m_1 perform l sequence of events
$e_{11} \rightarrow e_{12}, e_{12} \rightarrow e_{13} \ldots e_{1l-1} \rightarrow e_{1l}$.
The node m_p perform t sequence of events
$e_{p1} \rightarrow e_{p2}, e_{p2} \rightarrow e_{p3} \ldots e_{pt-1} \rightarrow e_{pt}$

Therefore it is required to learn a function that can be used for any given source malicious events of m_s to predict the targeted coordinated malicious events of m_t.

AdIoTLog collects IoT log over the LAN therefore AdIoTLog comprises of let $m_1, m_2, ..m_p$ nodes over one application gateway say AG_1 while $n_1, n_2, ..n_q$ nodes over another application gateway say AG_2.

AdIoTLog computes the probability of possible events in the sequence
P$((m_p : e_{p1} \rightarrow e_{p2}, e_{p2} \rightarrow e_{p3} \ldots e_{pl-1} \rightarrow e_{pl})$
$\rightarrow$
$(n_q : e_{q1} \rightarrow e_{q2}, e_{q2} \rightarrow e_{q3} \ldots e_{ql-1} \rightarrow e_{ql}))$.

Moreover, the GRU model computes the probability of entire sequence of events by applying chain rule. Therefore now if A is the set of events with the top high probabilities from IoT nodes with the same AG then the nodes are benign otherwise nodes are anomalous.

V. Algorithm

Our method AdIoTLog depicted in Algorithm 1 shows that how we trained the RNN. The algorithm also presents AdIoTLog aggregator which combines all local model updates of AG_i with AG_j. We input the network events sequences of AG_i and AG_j to the encoder and track every output and hidden state e.g when a network sequence of size 100 is input with 256 hidden sizes, it produces encoder outputs tensor of size (100, 256) and final hidden state tensor of 256 size.

The decoder is then given the first input as $\langle SOS \rangle$ token, final hidden state of the encoder, and final encoder output. The decoder can be given next input as the best guess by the decoder or the real target outputs during the training process. The concept of using target outputs as next input is called teacher forcing [19] that helps to converge the training process faster. We used the teacher forcing algorithm randomly with a probability of 0.5. However, during testing or evaluation time, the decoder is given the next input as the best guess only.

Network loss is computed based on decoder output and target tensor. Network weights for both Encoder and Decoder are optimized using Stochastic Gradient Descent (SGD) optimizer using a learning rate in range of 0.01-0.0001. We stored loss after every 100 steps to track if the network is learning.

Once the model has been trained, we test the model with random pairs of network events to compute the accuracy of the model learning. The accuracy is computed using average BLEU (Bilingual Evaluation Understudy) Score. BLEU is highly used to evaluate sequence-to-sequence models e.g. machine translation from one language to another. The predicted machine translated text is compared with one or more reference text. In this paper, we compared the predicted network event sequence with the ground truth network sequence using 1-gram (single words).

If A is the set of events with the top high probabilities from IoT nodes with same AG then the nodes are benign otherwise nodes are anomalous.

VI. Experiment Setup, Results, and Analysis

The GRU RNN model is available in PyTorch. For our experiments, we used Intel 4.7 GHz i7-8700K, 8 GB GTX 1080 with 2560 CUDA cores, and 64 GB Dual Channel DDR4 at 2400MHz to run the PyTorch library for GRU model. The various hyperparameters are explained in Table I.

```
P =TCP, UDP, HTTP, DNS, DHCP, HTTPS, NTP, ARP,
  ICMP
Input: ($Node_{m_i}$ $Node_{m_j}$) pairs using protocol P with input
    event vector ($x_1, \ldots x_{|x|}$ ) and target next event
    vector ($y_1, \ldots y_{|y|}$) logged at $AG_1$ and ($Node_{n_p}$,
    $Node_{n_q}$) using P with input event vector
    ($x_1, \ldots x_{|x|}$) and target next event vector ($y_1, \ldots y_{|y|}$)
    logged at $AG_2$ where $x, y \in$ vocabulary ($f_1, \ldots f_n$)
    at both $AG_1$ and $AG_2$
Output: AdLIoTLog
@Trigger: $AG_1$ receives labelled input event vector
    ($x_1, \ldots x_{|x|}$) and target next event vector
    ($y_1, \ldots y_{|y|}$) of node pairs ($Node_{n_p}$, $Node_{n_q}$) of
    $AG_2$. $AG_1$ has time series data
if $AG == AG_1$ then
    VocabVec, EventPairs = CreateLogVocabVec($x, y$)
    if need_train = TRUE then
        EncoderHiddenVec = InitHidden()
        foreach $x_i$ in $x_1, \ldots x_{|x|}$ do
            EmbeddedVec = CreateEmbedVec($x_i$)
            EncoderOutput, EncoderHiddenVec =
              GRUEncoder(EmbeddedVec,
              EncoderHiddenVec)
        end
        DecoderInput = SOSToken
        DecoderHidden = EncoderHiddenVec
        UseTeacherForcing = BooleanRandom(prob=0.5)
        if UseTeacherForcing then
            for $i <$ TargetLength do
                DecoderOutput, DecoderHidden,
                  DecoderAttention =
                  AttentionDecoder(DecoderInput,
                  DecoderHidden, EncoderOutput)
                Loss = NLLLoss(DecoderOutput, $y_i$)
                DecoderInput = $y_i$          /* Teacher
                  Forcing */
            end
        end
        else
            for $i <$ TargetLength do
                DecoderOutput, DecoderHidden,
                  DecoderAttention =
                  AttentionDecoder(DecoderInput,
                  DecoderHidden, EncoderOutput)
                Loss = NLLLoss(DecoderOutput, $y_i$)
                DecoderInput = MaxIndex(DecoderOutput)
                  /* No Teacher Forcing */
                break if DecoderInput is EOSToken
            end
        end
    end
    else
        TestingEventPairs = getRandomTestingEvents()
        foreach pair in TestingEventPairs do
            /* Evaluate is similar to
               Training except no teacher
               forcing & no gradient comp.  */
            PredictedNetworkEvent=
              EvaluateModel(Encoder, Decoder, pair[0])
            score= event_bleu(pair[1],
              PredictedNetworkEvent, weights=(1, 0, 0, 0))
              /* Add score to a list scores */
            if
              ($x_i$ by $Node_{m_i}$, $PredictedNetworkEvent$ by $Node_{m_j}$) &&
              ($Node_{m_i}, Node_{m_j}$) $\in AG_1$) then
                ($Node_{m_i}, Node_{m_j}$) pair is benign
            end
            else
                ($Node_{m_i}, Node_{m_j}$) pair is anomalus
            end
        end
        score = sum(scores)/ len(TestingEventPairs)
    end
end
```

Algorithm 1: AdLIoTLog Model

A. Training Data

We used ns-2 network simulator dataset to verify the proposed detection of collaborating nodes. We explain the training process that includes preparing data and training the sequence to sequence network.

1) The dataset prepared using Network Simulator: The training dataset in Network Simulator was created of 16 nodes. Figure 3 shows two scenarios (a) IoT network without collaborating nodes and (b) IoT network with collaborating nodes. Node pairs were simulated as IoT device and data server. For example, node pair (14, 2) was simulated for node 14 to upload data to node 2. In the first case, when there is no collaboration, node 14 will upload the data to node 2. In the second case, when nodes can collaborate using hidden channel, node 14 will upload the data to node 15. 8 UDP communication node pairs were used with a 1500 byte packet at the rate of 1 Mbps to generate the data. AI analytics Sequence-to-Sequence model can remember the good events and collaborative events, therefore, results in detecting the malicious events of the network.

2) Training Sequence-to-Sequence network: To interface ns-2 trace file to RNN model, we need a tensor pair. A tensor indexes IoT network log vocabulary for the input of the model. A tensor pair was prepared by including network events input tensor and network events target tensor. The logged network events were paired following the order of timestamps of network events one after the other. The input file included 12,236 network sequence pairs with 4170 unique elements that comprise different types of packets, protocols, sequence numbers, flags, etc. The combined ns-2 trace files of both setups - networks with hidden channel and network without hidden channel was input to the model. To compare the two scenarios, GRU RNN was trained without adversary node also. Figure 4 shows variation of model training (Negative Log Likelihood Loss) with moving average over 100 iterations.

Table I shows hyper parameters used in the training process.

B. Results and Analysis

We define the following performance metric based on BLEU score [20]. The BLEU score was computed by comparing the predicted network event sequence with the ground truth network sequence using 1-gram (single words). It is 100 if predicted sequence is exactly similar to ground truth sequence.

We define accuracy of the model that is based on the number of testing pairs as following:

$$\textbf{Accuracy} = \frac{sum\ of\ BLEUscore(TestingPairs)}{len(TestingPairs)}$$

Let $tp_1, tp_2, ..., tp_n$ are testing pairs and their respective bleu scores are $b_1, b_2, ..., b_n$ then accuracy of model output will be $\frac{\sum_{i=1}^{n} b_i}{n}$.

Table II shows model performance comparison of model output under collaborative-attack and non-collaborative attack scenarios. The experimental results show accuracy of 89-95% in case of collaborative attack and 91-98% in case of non-collaborative attack.

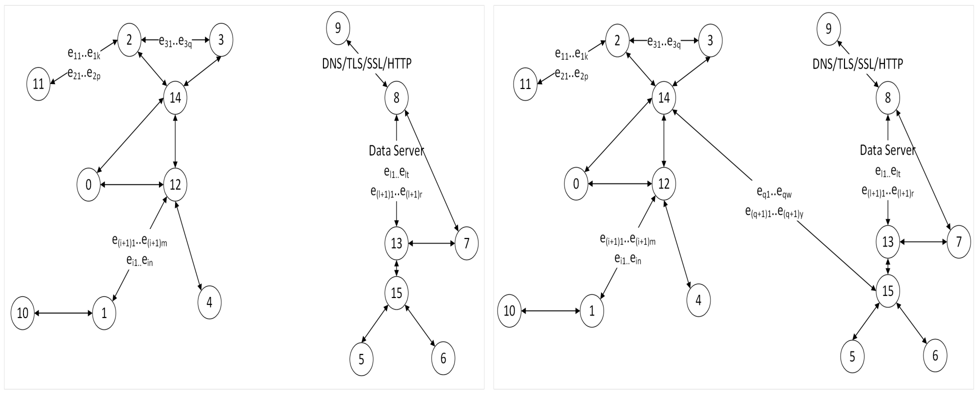

Fig. 3. (a) IoT network without collaborating nodes in ns-2 (b) IoT network with collaborating nodes in ns-2

Next, we explain the findings of the experiments based on model performance. The experiments were designed to answer the following questions:

a) What is the performance of the GRU RNN-based models when the input network events use sequences of the packet events of the protocols e.g. TCP, UDP, HTTP, etc and subsequences of the packet events e.g. sequence number, IP addresses, window scale option logged for each protocol: In simulator trace files, the values of features like sequence numbers, flags values, IP address etc. were simulated values different from the format used by TCP/IP model on a network. For Example, IP address in dotted format (e.g. 192.168.1.1) was replaced by a node number (e.g. 14). The generated sequence numbers were much easier to keep track of relatively small, predictable numbers rather than the actual numbers. Acknowledgment numbers were also not very random. We observed high accuracy of GRU RNN in predicting the network events for simulator data in comparison to dumped TCP/IP model output on a IoT network.

b) How is the subset A of x with top probabilities over $|y|$-dimensional tensor y in $|y|$ classes changed using the data under collaborative attack (shown in Figure 3a) and non attack scenario (shown in Figure 3b): When the results of both models (a) with attack data (b) without attack data were compared with each other on the set A, we observed that the accuracy of predicted network subsequences was less in the presence of attack. The collaborating nodes were connected through hidden channel, therefore the communication of adversarial node in N_1 with collaborating adversarial node in N_2 superseded in comparison to other nodes in their respective networks N_1 and N_2. The model was biased on predicting the collaborating class in the $|y|$ classes. In other words, in AdIoTLog under attack scenario, P$((m_p : e_{p1} \rightarrow e_{p2}, e_{p2} \rightarrow e_{p3} \ldots e_{pl-1} \rightarrow e_{pl}) \rightarrow$ $(n_q : e_{q1} \rightarrow e_{q2}, e_{q2} \rightarrow e_{q3} \ldots e_{ql-1} \rightarrow e_{ql}))$ was high most of the times for collaborating class. If we present Node tuple in A as (node, actual data server, pred data server) shown in Table II, for tuple (14, 2, 15), the actual node event was by 2 while model predicted node 15 rather than node 2 for source 14.

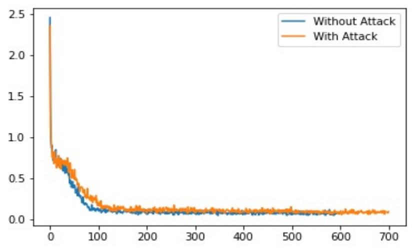

Fig. 4. Variation of model training (Negative Log Likelihood Loss Model Loss) x 100 iteration

c) How the performance of model varied when used for scenario with collaborating malicious nodes: In simulator generated log, there were sufficient instances needed for model learning. The model performance reduced in the presence of the attack. Features were much easier to keep track because of relatively small, predictable numbers rather than the actual numbers. We observed that when there is less data to train we get more iteration however when there is more data, number of iteration are less. GRU RNN can predict the event when trained on even small data with context.

Figure 4 shows variation of model training (Negative Log Likelihood Loss) with moving average over 100 iterations.

TABLE I
HYPER PARAMETERS OF MODEL

Network	No of hidden layer	No of iterations	Learning rate	Hidden Layer size	optimizer
IoT ns-2	1	70,000	0.01-0.0001	256	SGD

VII. Background

Now a days IoT platform has been in use to collect valuable data for the (i) social challenges, (ii) growth of technology, and (iii) smart applications etc. Neural networks have been found suitable for analyzing the huge volumes of data. Moreover,

TABLE II

COMPARISON OF MODEL OUTPUT UNDER COLLABORATIVE-ATTACK AND NON-COLLABORATIVE ATTACK SCENARIOS

Network	Size of set A	Node tuple in A in data without attack (node, actual data server, pred data server)	Node tuple in data with attack (node, actual data server, pred data server)	Accuracy with collaborative attack	Accuracy without collaborative attack
IoT ns-2	5	(6,15,15), (3,2,14), (6,15,15), (2,3,3), (8,9,9)	(14,2,15), (0,12,12), (2,11,14), (0,12,12), (15,5,6)	89-95%	91-98%

Neural networks - NN, CNN, RNN, and GAN are the tools for machine to learn existing hidden features in the data. IoT upload data on Internet through IoT network.

Wu et al. [9] presented a method for analyzing the bigdata collected through tools e.g. Hadoop, Mapreduce, Hive and others based on seq2seq- predictive models. Rather than input the seq vector from the bigdata components, the authors labeled the data of each component, to get the embedding vector, and subsequently input labeled vector to attention matrix. Finally the predicted vector is obtained using target information. In contrast, in our proposed method we split input vector on the basis of context of protocol which is processed by encoder and subsequently processed by decoder incorporating attention using target vector.

The model developed by Shen et al. [10] for collaborative nodes trying to use vulnerabilities of intrusion protection system also used RNN. The negative shift in model prediction was used to detect the attack, however, the attack considered was not over distributed machines, also they did not compare output of system under attack and without attack scenarios. They varied the length of input sequence to study the performance of RNN with increasing length of input sequence. Although, we observed that RNN shows better results if delimited using context rather than by arbitrarly varying the length of sequences input to the model.

The model developed by Almiani et al. [21] for intrusion detection system in an IoT network was trained on the NSL-KDD dataset for different types of attack. The authors suggested to use fog computing model for IoT network and classified different attacks using RNN. Amanullah et al. [22] studied and presented the deep learning technologies for IoT security. The growth in use of deep learning models for security shows its promising applicability on IoT log. IoT botnets are well-known attacks modeled by Meidan et al. [23] using deep autoencoder neural networks. The authors identified 23 prominent features leading to detection of the attack. Amanullah et al. [22] found deep learning approaches exhibits better performance compared to traditional machine learning models for IoT security. Deep learning technology extracts high level hidden features which helps in detection of the attack.

VIII. CONCLUSION

In this paper, we studied performance of GRU RNN model on the traffic log of an IoT network with and without adversary nodes. The adversary nodes are assumed to collaborate to cripple the data to be uploaded. We found that adversary node can be detected without considering any additional events rather it is required to log the network traffic. The log can be analyzed by AI algorithms to detect the adversary nodes in the network.

REFERENCES

[1] Vikas Hassija, Vinay Chamola, Balindam Chandra Bajpai, Sherali Zeadally, et al. Security issues in implantable medical devices: Fact or fiction? *Sustainable Cities and Society*, page 102552, 2020.

[2] Jianwei Hou, Leilei Qu, and Wenchang Shi. A survey on internet of things security from data perspectives. *Computer Networks*, 148:295–306, 2019.

[3] George Hatzivasilis, Konstantinos Fysarakis, Ioannis Papaefstathiou, and Charalampos Manifavas. A review of lightweight block ciphers. *Journal of Cryptographic Engineering*, 8(2):141–184, 2018.

[4] A Tawalbeh Lo'ai and Turki F Somani. More secure internet of things using robust encryption algorithms against side channel attacks. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–6. IEEE, 2016.

[5] Nadarajah Asokan, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, Matthias Schunter, Gene Tsudik, and Christian Wachsmann. Seda: Scalable embedded device attestation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 964–975, 2015.

[6] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *arXiv preprint arXiv:2104.07914*, 2021.

[7] Quoc-Viet Pham, Kapal Dev, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu, Thien Huynh-The, et al. Fusion of federated learning and industrial internet of things: A survey. *arXiv preprint arXiv:2101.00798*, 2021.

[8] Tung T Vu, Hien Quoc Ngo, Thomas L Marzetta, and Michail Matthaiou. How does cell-free massive mimo support multiple federated learning groups? *arXiv preprint arXiv:2107.09577*, 2021.

[9] Pin Wu, Zhihui Lu, Quan Zhou, Zhidan Lei, Xiaoqiang Li, Meikang Qiu, and Patrick CK Hung. Bigdata logs analysis based on seq2seq networks for cognitive internet of things. *Future Generation Computer Systems*, 90:477–488, 2019.

[10] Yun Shen, Enrico Mariconti, Pierre Antoine Vervier, and Gianluca Stringhini. Tiresias predicting security events through deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 592–605. ACM, 2018.

[11] Christopher Sweet, Stephen Moskal, and Shanchieh Jay Yang. On the veracity of cyber intrusion alerts synthesized by generative adversarial networks. *arXiv preprint arXiv:1908.01219*, 2019.

[12] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018.

[13] Sandhya Aneja, Nagender Aneja, and Md Shohidul Islam. Iot device fingerprint using deep learning. In *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pages 174–179. IEEE, 2018.

[14] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. Iotsense behavioral fingerprinting of iot devices. *arXiv preprint arXiv:1804.03852*, 2018.

[15] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184. IEEE, 2017.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[17] Pieter Robyns, Bram Bonné, Peter Quax, and Wim Lamotte. Noncooperative 802.11 mac layer fingerprinting and tracking of mobile devices. *Security and Communication Networks*, 2017, 2017.

[18] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021.

[19] Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[20] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[21] Muder Almiani, Alia AbuGhazleh, Amer Al-Rahayfeh, Saleh Atiewi, and Abdul Razaque. Deep recurrent neural network for iot intrusion detection system. *Simulation Modelling Practice and Theory*, page 102031, 2019.

[22] Mohamed Ahzam Amanullah, Riyaz Ahamed Ariyaluran Habeeb, Fariza Hanum Nasaruddin, Abdullah Gani, Ejaz Ahmed, Abdul Salam Mohamed Nainar, Nazihah Md Akim, and Muhammad Imran. Deep learning and big data technologies for iot security. *Computer Communications*, 151:495–517, 2020.

[23] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.