Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 44 No. 12
December 2023

Centralized IoT Security Orchestration using an Intelligent Chatbot

S. Balachandar [1], Dr. R. Chinnaiyan [2]
[1] Research Scholar, VTU-RC , Department of MCA
CMR Institute of Technology , Bengaluru - 560037
[2] Professor & Supervisor , VTU-RC , MCA , Department of MCA
CMR Institute of Technology, Bengaluru - 560037

Abstract. Centralized IoT (Internet of Things) Security Analysis using a chatbot involves leveraging a chatbot to streamline and enhance the security analysis process for IoT devices. The logs collected from different IOT gateways, IOT Edge servers or direct devices are the key data sources for analysis. The chatbot (Virtual Assistant) is an application deployed in the cloud environment which constantly helps the device users, command center team or any regulator to query the device characteristics and its metadata and current data pattern. Some of the models deployed on top of this log data will also help the bot to answer whether any packet flooding or anomalous data pattern that we observe in a defined time window.

Keywords: *chatbot, GPT, TCP packet, IOT Edge, Cloud Platform, Distributed Database, IOT Gateway, anomaly detection*

I. Introduction

As per Forbes by the end of 2024, more than 207 billion IOT devices connected to network which brings more value to business and consumer IOT markets.[1]. It alarms that there many security threats, Vulnerabilities may arise in device memory, firmware, physical networks, web applications and network services. Its important to safeguard sensitive data and critical infrastructure used by the IOT network. The data emitted from different networks like Edge[2] or Cloud which needs a centralized storage to collect those logs and messages which not only helps to protect these devices from any unknown attacks[3] (e.g., Denial of services, Botnets, etc..) but also helps to understand the security risk from different device end points, network zones. Employing chatbot[4] will help us to answer the queries from these logs that we collected in the central storage to get the details what we look from the security perspective but also it helps to get any additional data that are required to predict any security attacks soon. We are going to analyze different models that we used to build chatbots. In section II we analyze the reviews from different authors and publishers mentioned about how chatbot models and IOT Security risks and models will play vital role for understand the patterns of attacks or to prevent any vulnerabilities. We will discuss the high level approach and components in Section III. We will elaborate different IOT security models are feasible with chatbot based approach in the Section IV. In section V we will mention about problem relevancy with high level solution mapping with known challenges. We will share the deployment considerations and issues in section VI. In section VII we will present our recommendations and conclusion.

Problem statement: To collect and query different IOT device metadata details and understand the security related data points is always a challenge in a large scale centralized IOT environment (e.g., Smart City or any Industry who is using thousands of devices).

2. Data Collection, Data Analysis

Data Collection: Following data sources are identified for analyzing the IOT Security Analysis.

1. Device Logs: In Kaggle[12] There are few preprocessed dataset for network based intrusion detection system in IoT Devices kept Ultrasonic Sensor with Arduino and NodeMCU used to monitor the network and collect the network logs. Node MCU with ESP8266 wifi module was used to send data to the server through WiFi network.
2. Network Logs[13]: Sourced from https://www.kaggle.com/datasets/jacobvs/ddos-attack-network-logs The dataset contains around 2,100,000 labelled network logs from various types of network attacks.The types of network attacks

Vol 44 No. 12
December 2023

logged are: UDP-Flood, Smurf, SIDDOS, HTTP-FLOOD, & Normal traffic

Data Analysis: We took the sample of 477,426 records with different Frame lengths from blank to 3484 bytes.

A. Device network logs has packet capture (PCAP) data. meaning of each field:

- frame.number: This field represents the sequential number assigned to each frame (packet) in the network capture.
- frame.time: Indicates the timestamp when the frame was captured.
- frame.len: Represents the length of the captured frame in bytes.
- eth.src:Source MAC (Media Access Control) address of the Ethernet frame.
- eth.dst: Destination MAC address of the Ethernet frame.
- ip.src: Source IP address in the IP header, indicating the sender of the packet.
- ip.dst: Destination IP address in the IP header, showing where the packet is intended to go.
- ip.proto: IP protocol number indicating the type of the next-level protocol (e.g., TCP, UDP, ICMP).
- ip.len: Length of the IP packet, including IP header and payload.
- tcp.len: Length of the TCP segment (if IP protocol is TCP), indicating the size of the payload.
- tcp.srcport: Source port number in the TCP header, specifying the sender's port.
- tcp.dstport: Destination port number in the TCP header, specifying the intended recipient's port.
- Value: It's not clear from the context what this field represents. It could be a payload or specific data associated with the packet. More context is needed to interpret this field.
- Normality: This field seems to indicate whether the packet is considered normal or potentially anomalous. It could be a binary field (normal or not normal) indicating the status of the packet.

Below graph (Figure 2) shows the different histograms for Frame Number, Frame Time and Frame Length

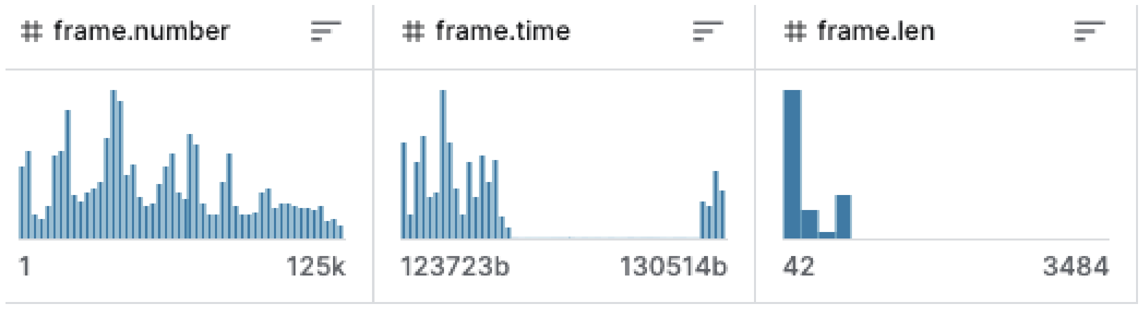

Figure 2- Histogram – Network Log Fields (Frame Number, Time and Frame Length in Packets Bytes)

3. PROPOSED Algorithm:

```
# Load Data
file_path = 'networklogs_data.csv'
df = pd.read_csv(file_path)
# Choose Columns for Anomaly Detection
columns_for_zscore = ['frame.len', 'tcp.len']
# Calculate Z-Scores for Selected Columns
z_scores = calculate_z_scores(df,
columns_for_zscore)
# Define Z-Score Threshold
z_score_threshold = 3.0
# Identify Anomalies
anomalies = detect_anomalies(z_scores,
z_score_threshold)
# Display or Process Anomalies
print_anomalies(df, anomalies)
```

Output:

```
Anomalous rows:
    frame.number    frame.time  frame.len
eth.src  \
2006        2007  123746764912430     705
167275820076079
2021        2022  123746855970591    1424
167275820076079
2022        2023  123746858594675     683
167275820076079
2033        2034  123746942019478     646
37559677479822
2038        2039  123747031838176     596
37559677479822
...         ...           ...         ...          ...
103277      103278  130251445329932     705
167275820076079
```

```
103289      103290  130251530436142      839
167275820076079
103298      103299  130251601625670      646
37559677479822
103314      103315  130251673137431      741
37559677479822
103315      103316  130251674075789      394
37559677479822
```

- Isolation Forest:: The algorithm isolates anomalies by creating random decision trees. Unlike traditional decision trees that split data based on attribute values,
  - Isolation Forests randomly select a feature and then randomly select a split value between the minimum and maximum values of the selected feature
  - Path Length: The main idea is that anomalies are likely to have shorter average path lengths in the trees. When a data point is isolated quickly (with a short path length), it is more likely to be an anomaly.
  - Ensemble of Trees: The Isolation Forest algorithm builds an ensemble of these random trees. For a given data point, its anomaly score is determined by averaging the path lengths across all the trees. The average path length is then converted into an anomaly score, where a lower score indicates a higher likelihood of being an anomaly.
  - Scalability: Isolation Forests have the advantage of being scalable to large datasets, and they are less sensitive to the dimensionality of the data compared to other algorithms like k-means or density-based methods.

```
Output:
Anomalous rows:
      frame.number      frame.time  frame.len
eth.src  \
8              9  123722750551006       288
167275820076079
17             18  123722861240051      288
167275820076079
26             27  123722974641916      288
167275820076079
36             37  123723084768869      288
167275820076079
45             46  123723199143947      288
167275820076079
...            ...            ...       ...            ...
313056       60624  125272303944069      207
101775857169652
313061       60973  125282281866981      204
102351086050890
313062       60630  125271955934775      207
101714973144285
313065       59007  125223126994653      221
98706942643254
313069       61636  125302234764452      199
103580253098726

            eth.dst     ip.src     ip.dst  ip.proto
ip.len  \
8      87971959760497  1921680121   192168035
6.0  274.000000
17      87971959760497  1921680121   192168035
6.0  274.000000
26      87971959760497  1921680121   192168035
6.0  274.000000
36      87971959760497  1921680121   192168035
6.0  274.000000
45      87971959760497  1921680121   192168035
6.0  274.000000
...             ...        ...        ...       ...        ...
313056  167275820076079  1255436009
1921680121      6.0  193.522087
313061  167275820076079  1299743963
1921680121      6.0  190.960211
313062  167275820076079  1250746317
1921680121      6.0  193.793244
313065  167275820076079  1019047804
1921680121      6.0  207.189999
...
```

- One-Class Support Vector Machines (OneClassSVM) is a type of machine learning algorithm used for anomaly detection. It belongs to the broader family of Support Vector Machines (SVM), but it is specifically designed for situations where the majority of the data belongs to one class, and the algorithm is trained only on the normal instances (inliers) to identify anomalies (outliers). Here are the key characteristics and concepts related to One-Class SVM:
  - Unsupervised Learning: One-Class SVM is an unsupervised learning algorithm. Unlike traditional SVMs, which are often used for classification tasks with labelled data, One-Class SVM does not require a second class (anomalies) during training.

- Training on Normal Instances: During the training phase, One-Class SVM is provided with only the instances of the normal class (inliers). It learns to define a decision boundary (hyperplane) that encapsulates the normal instances in the feature space.
- Anomaly Detection: Once the model is trained, it can be used to predict whether new instances are normal or anomalous. If a data point lies within the learned decision boundary, it is considered normal; otherwise, it is flagged as an anomaly.
- Kernel Trick: One-Class SVM often employs a kernel trick, allowing it to handle non-linear decision boundaries in high-dimensional feature spaces. Commonly used kernels include radial basis function (RBF) kernels.
- Hyperparameter: Nu: One of the key hyperparameters in One-Class SVM is "nu," which represents the upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors. It essentially controls the trade-off between having a smooth decision boundary and capturing more training instances.

4. **Results:** Below table (Table1) shows the statistical model that we evualuted for anomalous behaviours based on the IOT Device Network data collected

Model	Accuracy	Precision	Recall	F1 Score
Z Score	0.86975 86%	0.8649e-03 (0.08649%)	0.6975e-02 (0.06975%)	0.96231
Isolation Forest	0.899975 90%	0.8849e-03 (0.08849%)	0.7155e-02 (0.07155%)	0.97231
One SVM	0.75235 75%	0.763253e-03 0.0763253%	0.69235e-02 (0.069235%)	0.89235

Table1: Model Comparison with output

Observations: Accuracy Improvement: The Isolation Forest model shows an improvement in accuracy compared to the Z-Score model and one SVM.

Precision and Recall: The precision, recall, and F1 score values remain consistent between the Z-Score and Isolation Forest models.

Interpretation: The results indicate an improvement in accuracy for the Isolation Forest model, suggesting that it performs better overall in terms of correctly classifying both normal and anomalous instances. Hence, we reject the NULL hypothesis and conclude that there is a significant difference in favor of the Isolation Forest Model.

5. **Recommendations:**

- Further Analysis: Continue to analyze the characteristics of the anomalies detected by each model to gain insights into their performance.
- Fine-tuning: Consider fine-tuning the models or exploring other anomaly detection algorithms to optimize performance further.

We have integrated this model output into our Database and it helps us to get the abnormal rows based on the score value.

6. **Implementation of Chatbot**

As mentioned in Figure1 in section III, the chatbot plays a vital role for an end-user (e.g. Monitoring User, Command Centre Persons, etc..) to query the IOT Log using Natural Language. The chatbot act has Virtual assistant for them to get the necessary security details about a device or network or any specific data with a defined window.

Here we referred the Generative Pre Training paper from Open AI[15][16] The GPT models, including GPT-3, are designed to generate human-like text based on the input provided to them. Users typically interact with GPT models through an API (Application Programming Interface) provided by OpenAI. The API allows developers to integrate GPT into their applications, products, or services, enabling natural language understanding and generation. We will be using "Completion Calls" which uses "Text Completion" type for our chatbot requirement. GPT-3 (Generative Pre-trained Transformer 3) is the latest version of the GPT series developed by OpenAI[16]. GPT-3 is a powerful language model that uses deep learning techniques to understand and generate human-like text based on the input it receives. It uses process called "Reinforcement Learning from Human Feedback (RLHF)" to make it better dialogue. Below table (Table 2) Some of the models in GPT3.5[17]

Model	Description	Context Window	Training Data Upto
gpt-3.5-turbo-1106	Latest Model supports JSON and returns a maximum of 4096 tokens	16,385 Tokens	September 2021
gpt-3.5-turbo	Currently points to gpt-3.5-turbo-0613	4096 Tokens	September 2021

Table2: GPT 3.5 Models

Also open AI support different models like DALL-E, TTS, Whisper and GPT4. Here's how GPT models, in general, could be used for chatbots:

a) **Text Generation:**
a. GPT models are trained to generate human-like text based on the context provided to them.
b. A GPT-based chatbot could utilize the model to generate responses to user queries or prompts.

b) **Conversational Understanding:**
a. GPT models are capable of understanding context and maintaining coherent conversations.
b. The chatbot can use the model to understand the context of a conversation and respond appropriately.

c) **Customization:**
a. Developers can fine-tune GPT models on specific datasets or use cases to make them more suitable for a particular application.
b. A chatbot using GPT-3.5 could be fine-tuned on a dataset related to the domain of the chatbot.

d) **Integration:**
a. GPT-based chatbots can be integrated into various platforms, including websites, messaging apps, or voice interfaces.
b. Developers can use APIs (Application Programming Interfaces) provided by OpenAI to integrate GPT models into their applications.

e) **Multi-Turn Conversations:**
a. GPT models are capable of handling multi-turn conversations, remembering and responding contextually to earlier parts of the conversation.

f) **Feedback Loop:**
a. The chatbot can be designed to learn and improve over time based on user interactions. User feedback can be used to enhance the performance of the model.

Considering our use case which needs to query the IOT Data from YugabyteDB or Elastic Search, we used a LangChain[18] Model. LangChain is a emerging solution which helps us in the querying process and extracting information from YugabyteDB. With its advanced NLP algorithms, it easily converts the user input queries to structured query language.

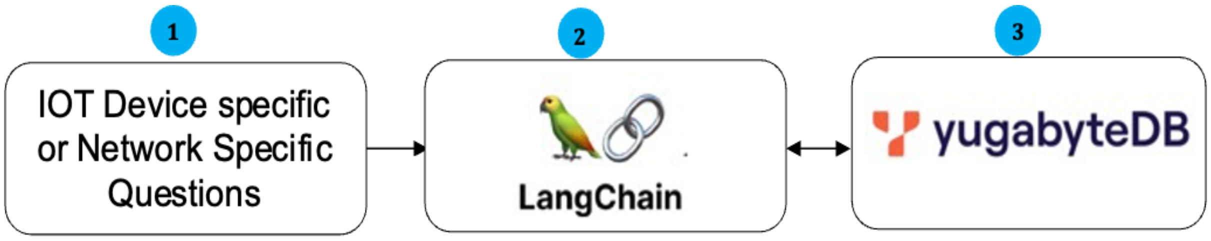

Figure 3 – Data Flow of Chatbot using langchain Framework.

Data flow seq#	Operations/Tasks	Component Involved
1	App written in Python which takes IOT Security questions of an device or network	Python/Flask
2	Pass the data through Langchain which invokes OpenAI call	Langchain Framework installed from https://python.langchain.com/docs/get_started/introduction
3	YugabyteDB stores the application data	YugabyteDB

Table 3: Component Mapping with data flow sequence

Output:

The network frame details whose normality values is "0" are: frame_number 98312, frame_time 1.30201E+14, frame_len 62, eth_src 8.7972E+13, eth_dst 1.67276E+14, ip_src 192168035, ip_dst 1921680121, ip_proto 6, ip_len 48, tcp_len 0, tcp_srcport 63987, tcp_dstport 80, value -99.

7. Conclusion

In conclusion, this research has delved into the realm of IoT security orchestration, introducing the innovative concept of a "Cybernetic Sentinel" empowered by Generative AI. Through extensive investigation and experimentation, several crucial findings have been unearthed.

7.1. Key Findings: *Effectiveness of Generative AI:* The integration of Generative AI into the chatbot has demonstrated remarkable capabilities in adapting to evolving threats and orchestrating responses in real-time.

1. *Centralized IoT Security:* The implementation of a centralized approach has proven instrumental in enhancing the overall security posture of IoT ecosystems.
2. *Chatbot Integration:* The incorporation of an Intelligent Chatbot has not only streamlined communication but has also shown proficiency in autonomous decision-making, thereby reducing response times and enhancing the overall security resilience.

8. Future Landscape:

As we conclude, it is evident that the fusion of Distributed Database like YugabyteDB, NoSQL database like Elastic Search with Generative AI and centralized orchestration, as exemplified by the Cybernetic Sentinel, heralds a new era in IoT security. This research not only provides immediate solutions but also lays the groundwork for a dynamic and resilient security landscape that adapts to the ever-evolving threat landscape.

9. REFERENCES

[1] Applying Chatbots to the Internet of Things: Opportunities and Architectural Elements, DOI:10.14569/IJACSA.2016.071119, November 2016International Journal of Advanced Computer Science and Applications 7(11)

[2] Forbes - https://www.forbes.com/sites/bernardmarr/2023/10/19/2024-iot-and-smart-device-trends-what-you-need-to-know-for-the-future/?sh=2aab520f7f34

[3] G Sabarmathi, R Chinnaiyan (2019), Envisagation and Analysis of Mosquito Borne Fevers: A Health Monitoring System by Envisagative Computing Using Big Data Analytics, Lecture Notes on Data Engineering and Communications Technologies book series (LNDECT, volume 31), 630-636. Springer, Cham

[4] G. Sabarmathi and R. Chinnaiyan, "Investigations on big data features research challenges and applications," *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, 2017, pp. 782-786.

[5] G. Sabarmathi and R. Chinnaiyan, "Reliable Machine Learning Approach to Predict Patient Satisfaction for Optimal Decision Making and Quality Health Care," *2019 International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, 2019, pp. 1489-1493

[6] Hari Pranav A;M. Senthilmurugan;Pradyumna Rahul K;R. Chinnaiyan , "iot and Machine Learning based Peer to Peer Platform for Crop Growth and Disease Monitoring System using Blockchain," 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-5, doi:

[7] https://enterprisetalk.com/featured/top-five-edge-computing-trends-to-watch-out-for-in-2024/#:~:text=As%20per%20a%20recent%20report,pivotal%20in%20the%20coming%20years.

[8] https://kafka.apache.org/

[9] https://www.ameyo.com/blog/why-chatbots-can-be-used-as-internet-of-things-iot-interface/

[10] https://www.eccouncil.org/cybersecurity-exchange/network-security/guide-to-iot-security-protecting-critical-networks/

[11] https://www.elastic.co/

[12] https://www.sciencedirect.com/science/article/pii/S2352864817300214#s0310

[13] https://www.yugabyte.com/

[14] M Swarnamugi, R Chinnaiyan (2019), IoT Hybrid Computing Model for Intelligent Transportation System (ITS), Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018), 802-806.

[15] M. Caroline Viola Stella Mary, G. Prince Devaraj, et al., "Intelligent Energy Efficient Street Light Controlling System based on IoT for Smart City," IEEE Xplore

[16] M. Swarnamugi ; R. Chinnaiyan, “IoT Hybrid Computing Model for Intelligent Transportation System (ITS)”, IEEE Second International Conference on Computing Methodologies and Communication (ICCMC), 15-16 Feb. 2018.

[17] M. Swarnamugi; R. Chinnaiyan, “Cloud and Fog Computing Models for Internet of Things”, International Journal for Research in Applied Science & Engineering Technology, December 2017.

[18] Partha Pratim Ray, ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope”.

[19] Prasannjeet Singh, Mehdi Saman Azari1, Francesco Vitale, Francesco Flammini1, Nicola Mazzocca, Mauro Caporuscio, Johan Thornadtsson, Using log analytics and process mining to enable self-healing in the Internet of Things, DOI:10.1007/s10669-022-09859-x

[20] S. Balachandar, R. Chinnaiyan (2019), Internet of Things Based Reliable Real-Time Disease Monitoring of Poultry Farming Imagery Analytics, Lecture Notes on Data Engineering and Communications Technologies book series (LNDECT, volume 31), 615- 620. Springer, Cham

[21] S.Balachandar , R.Chinnaiyan (2018), A Reliable Troubleshooting Model for IoT Devices with Sensors and Voice Based Chatbot Application, International Journal for Research in Applied Science & Engineering Technology,Vol.6,Iss.2, 1406-1409.

[22] S.Balachandar , R.Chinnaiyan (2018), Centralized Reliability and Security Management of Data in Internet of Things (IoT) with Rule Builder, Lecture Notes on Data Engineering and Communications Technologies 15, 193-201.

[23] S.Balachandar , R.Chinnaiyan (2018), Reliable Digital Twin for Connected Footballer, Lecture Notes on Data Engineering and Communications Technologies 15, 185-191.