IDOTS+: Enhancing IoTDots for Smart Environments

Vijayavarman Arunasalam Harikrishnan
(va7457@rit.edu)

Sree Alekhya Teerthala
(st6626@rit.edu)

Abstract

The rise in the usage of Internet of Things (IoT) devices has introduced new difficulties for digital forensics investigations. Traditional forensic tools are often inadequate for analyzing IoT devices because of their limited memory, lack of standardization and diverse data formats. This research paper presents IDOTS+, an IoT forensics framework aimed at overcoming these obstacles. IDOTS+ utilizes the OpenHAB home automation platform to generate logs of activities within an IoT smart home environment. A machine learning model based on a Convolutional Neural Network (CNN) analyzes these logs to identify potential malicious events according to predefined rules customized for the specific smart home setup. The proposed approach achieves 89% accuracy in flagging malicious activities while allowing users to tailor the ruleset to their unique requirements.

1. Introduction

The growing popularity of IoT devices has resulted in the ease of attackers exploiting them and taking control of them. There have been numerous instances where IoT devices have fallen victim to such attacks, leading to dire consequences. The Mirai botnet attack in 2016, which used compromised IoT devices to launch massive distributed denial-of-service (DDoS) attacks, is a stark reminder of the devastation that can result from unsecured IoT devices. In the aftermath of such incidents, forensic investigation is crucial to uncover the underlying causes of such attacks and prevent them from happening again in the future. However, traditional digital forensics tools and frameworks are designed primarily for desktop and server environments and they pose significant challenges when applied to IoT devices. IoT devices often have limited computational resources and storage capacity, which makes it difficult to perform forensic analysis on the device itself. Because of this, data may need to be extracted and analyzed on external systems, which may cause data integrity issues. The IoT ecosystem lacks standardization, with different manufacturers implementing their own protocols and data formats, which can make it challenging to develop universal forensic tools and techniques that work across all IoT devices.

Given these unique challenges, it is crucial to develop specialized forensic tools, techniques, and frameworks tailored specifically for IoT devices. This research paper aims to explore the current state of IoT forensics, identify the gaps and limitations in existing approaches, and propose a solution called IDOTS+ to address the unique challenges posed by IoT devices.

Our approach involves setting up an IoT Smart home environment consisting of a Philips Smart Wi-Fi LED Model 9290023833B bulb using OpenHAB, an open-source home automation platform implemented on a Raspberry Pi 3 Fig 7. Within this environment, the smart home application will generate logs capturing various system activities and events that happen. These logs will then be extracted and processed by our custom log analyzer, which is implemented by using machine learning techniques. The log analyzer primarily identifies and flags potentially malicious instances based on some predefined rules that are set for the IoT smart home environment. Given the adaptable nature of smart homes, their configurations can vary widely depending on individual preferences, hence our model allows the users with the flexibility to customize rules and settings to suit their specific needs, thereby enhancing user-friendliness. Furthermore, our solution provides deep insights into the detected malicious incidents, pinpointing the exact instance and time of the occurrence within the IoT ecosystem. This level of detail enables investigators to swiftly understand the point of compromise and take action accordingly. By combining IoT simulation with advanced machine learning analytics, our framework strengthens forensic investigators to effectively identify and mitigate security threats in IoT

environments. The analyzer is coded in Python using the CNN model and has been trained on logs for two days and test data with logs for one day. These logs are a combination of both malicious and benign activity so that the model can generate efficient results that can cater to a real-world scenario.

Summary of contributions

- As the feature of generating logs is a part of the OpenHAB framework, there is no extra overhead regarding storage or computational power on the IoT device.
- Enabling the users to have control over what rules they set for their environment, our framework provides a more user-friendly approach that can be customized for each individual smart home environment.
- As the analyzer is a machine learning algorithm, it can be used by both smart home users and security professionals easily and pinpoint the exact time when a malicious event has occurred to fast-track the countermeasures for an attack.
- Most importantly, by using a unique and easy-to-implement machine learning model, which is a Convolution Neural Network, we could achieve an accuracy of 89% for detecting malicious activities. This novel application of the IDOTS+ framework to IoT forensics aims to address the gaps and limitations identified in previous research efforts.

Structure of the paper

The organization of the research paper is structured with a Related Work section providing a thorough examination of prior research in IoT forensics and its methodologies. Next on, we have a section giving a basic introduction to our IDOTS+ architecture and its threat model. Later, we move on to the section which consists of the thorough implementation of the IDOTS+ framework. Following that, we have a section that covers a report on the evaluation results of our model. Lastly, we have the lessons learned, limitations, and potential future directions in section 5, followed by the conclusion to complete the paper. The Appendices section contains additional screenshots for a better understanding of the framework.

2. Related Work

2.1 Digital Forensics

Digital forensics is the process of identifying, collecting, and interpreting digital data to gather evidence for legal proceedings or for investigating cyber-attacks. The need for digital forensics arises from the increasing reliance on digital devices and the potential for these devices to be involved in criminal activities, such as fraud, cybercrime, or intellectual property theft. [11]

The digital forensics lifecycle typically includes the following phases: preparation, acquisition & preservation, examining, analyzing, and reporting. The preparation phase involves the preparation of all required tools and tasks needed before the investigation can begin. The acquisition & preservation phase caters to identifying, collecting, recording and imaging all the suspected devices and the event-related data so that integrity is maintained. In the examining phase, all collected data is inserted into the investigation workstation for analysis. The analysis phase is where the digital evidence gathered during the examination phase undergoes systematic interpretation and in-depth analysis. Ultimately, the findings from the forensic investigation are compiled and presented in a comprehensive report which consists of the entire process, documenting the methodologies employed, evidence uncovered, and conclusions reached.

As shown in Fig 1, the digital forensics process involves a transformation of the collected media into data, which is then transformed into information through the analysis phase. This information is further refined and transformed into admissible evidence, suitable for presentation in court or for other internal purposes.[11]

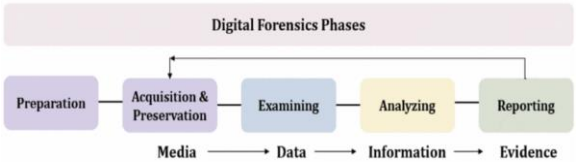

Fig 1: Phases of Digital Forensics

Digital forensics can be categorized into various types based on the nature of the evidence or the environment in which it is conducted, such as computer forensics, network forensics, mobile device forensics and cloud forensics.

2.2 IoT Forensics

IoT Forensics is a subset of Digital Forensics and its primary objective involves the forensic and lawful extraction of digital information from IoT devices [1]. The necessary evidence for IoT forensics may be gathered via a wearable device, a controlling device, a car sensor, a home automation sensor, or a cloud data store. The way these linked smart devices are configured affects the entire research and forensics process. A forensic investigation in an IoT environment typically involves multiple levels of analysis, including device-level forensics, network-level forensics, and cloud-level forensics [4]. Device-level forensics focuses on examining the internal storage, memory, and firmware of individual IoT devices to extract digital artifacts. Network-level forensics involves analyzing network traffic and communication protocols to trace the flow of data between interconnected devices. Cloud-level forensics entails investigating data stored in cloud services accessed by IoT devices [4].

For instance, consider the scenario of a smart office environment equipped with various IoT devices, including a smart thermostat responsible for regulating the office's temperature. Anomalies in the thermostat's logs, such as erratic temperature fluctuations, could signal potential cyberattacks, prompting a forensic investigation. This investigation may involve analyzing the device's firmware, memory, and network communication to uncover evidence of unauthorized access or tampering. In a typical thermostat log, you might find temperature readings, mode changes, system status updates, user interactions, network communication details, error messages, environmental conditions, and security events. So, in an ideal scenario, a forensic investigator has to go through these logs line by line to find the root cause of this potential cyber-attack. Our IDOTS+ framework uses the logs from the OpenHAB platform and feeds them into an analyzer that is trained using CNN to flag all malicious events that have occurred in the logs based on the rules that are set for the smart environment. This way, when the source of the malicious action is found within a small timeframe, it is easier to put more effort into finding a solution and implementing it.

2.3 Need for IoT Forensics

G. Surange et al. [5] in their paper discuss the importance of IoT forensics in investigating cybercrimes involving IoT devices, specifically intelligent robot vacuums. They also explore the various challenges that IoT devices pose for any forensic analysis, due to which the necessity to employ IoT forensics has arrived. IoT devices communicate and gather enormous amounts of data, including private and sensitive data. It is quite difficult to make sure that sensitive data is protected against breaches or unwanted access and also from privacy infringement. IoT devices produce data in varied data formats that are stored in different places and involve different Operating Systems, which makes it difficult to properly manage and protect this heterogeneous data in various geographical locations.

The paper by A. Ahmed et al. [6] mentions the most vital flaw in IoT devices, which is their small storage space for data, like logs and even if logs are stored, it is for a very short period. Before an attack is identified and analyzed, crucial evidence would have been lost. Storing this data in a third-party location is also a security concern and can pose a risk during data retrieval. Hence, it is critical to develop and utilize an efficient IoT Forensics Framework.

2.4 IoT Forensics Frameworks

Though the field of IoT forensics is in its nascent stages, there are many theoretical frameworks regarding its implementation. However, not many have been implemented. Furthermore, the evolving landscape of IoT technologies necessitates ongoing research and development efforts to keep pace with emerging challenges and advancements in the field by collaborating with technical experts from other domains. Some frameworks that we went through are discussed below.

2.4.1 Digital Forensics based on Federated Learning in IoT Environment

The framework discussed in the paper [7] presents a Deep Federated Learning-based Digital Forensics method for IoT networks to automate the detection and tracing of attack traces. It addresses the challenges of existing digital forensic techniques by utilizing Federated Learning (FL) with a Convolutional Neural Network (CNN) model. The method involves data processing steps and an FL-based CNN technique to detect and trace the origin of attacks in distributed IoT networks and showed an accuracy of 81%. By

preserving data privacy and sharing only learned hyperparameters with a global server, the approach aims to improve forensic examination accuracy and produce court-admissible evidence. Synchronizing data with the global server is highly essential to ensure the consistency of the model training and this is very difficult when we employ it in a real-world scenario. However, while we recognize the value of CNN for the high performance it provides when integrated with detecting the origin of attacks in IoT systems, we've made a decision in our framework to prioritize local data processing and analysis on a dedicated server. This ensures simplicity and efficiency in managing and using the framework, avoiding the complexities of frequent synchronization to a global server.

2.4.2 A Fog-Based Digital Forensics Investigation Framework for IoT Systems

The approach presented in [3] uses fog computing to protect IoT systems from cyberattacks by efficiently analyzing and storing evidence. The framework consists of six components: device monitoring manager, forensic analyzer, evidence recovery, case reporting, communication, and storage. By utilizing fog computing, FoBI can identify when an IoT device is malfunctioning, trace its history, and mine locally stored data to determine its status or send alerts. The framework also aims to mitigate cyber-attack effects by notifying other IoT devices or nodes of potential threats. Overall, FoBI is designed to provide early-stage digital forensic investigation for IoT systems. However, its integration with cloud storage poses security and privacy concerns because of the potential exposure of sensitive data to unauthorized access or breaches and sometimes even leads to denial of service. Our framework is completely local, which mitigates the risk of data leakage.

2.4.3 A blockchain-based decentralized efficient investigation framework for IoT digital forensics

The blockchain-based framework proposed in [8] enhances the forensics investigation process by recording the transmission process of IoT devices as transactions. The model includes smart contracts for evidence generation, acquisition, and report generation. The researchers simulated the model using the Ethereum private network platform, analyzing gas consumption, throughput, and CPU utilization to evaluate its scalability and effectiveness. The immutability of blockchain data poses challenges in correcting errors or updating information once recorded, which could be problematic in situations where data revisions may be necessary. IDOTS+ on the other hand, is a simple framework that is flexible and less complicated as it analyzes the logs from OpenHAB.

2.4.4 An improved digital evidence acquisition model for the Internet of Things Forensic I: A theoretical framework

The theoretical framework outlined in [10] aims to enhance digital evidence acquisition in IoT forensic investigations by implementing the LoS algorithm to streamline the acquisition process, with a focus on improving traceability and simplifying analysis. The framework emphasizes the importance of revising existing IoT forensic procedures to align with the proposed model. Additionally, it introduces the concept of a management platform to maximize the utilization of past IoT forensic cases. By prioritizing traceability improvement, overhead reduction, and management optimization, this theoretical framework offers a structured approach to overcoming the challenges associated with acquiring and analyzing digital evidence in IoT forensic scenarios.

2.4.5 IoTDots: A Digital Forensics Framework for Smart Environments

IoTDots is a model proposed in 2018 [2], which is a digital forensic framework that consists of two components: IoTDots-Modifier and IoTDots-Analyzer as shown in Fig 2. During compilation, the IoTDots-Modifier analyzes the source code of smart applications, identifies potential forensic data, and embeds tracking logs. And when the device is functional, the data is automatically logged into an IoTDots Logs Database. The IoTDots-Analyzer employs data processing and machine learning techniques to extract actionable forensic insights from device activities which were logged and stored in the database. The framework was tested in a realistic smart office environment with 22 devices which consisted of a Smart Home Hub, Smart Light, Smart Lock, Fire Alarm, Smart Monitoring System, Smart Thermostat, Motion Sensor, Light Sensor, Temperature Sensor, and Door Sensor. Additionally, the threat model used in the paper categorizes threats into regular user activities, anomalous user activities, and malicious behaviors, considering both time-dependent and time-independent actions. For the implementation of the project, the smart application

source code used was the Samsung SmartThings application open-source code. This code was run through the IoTDots-Modifier, which inserted logs and performed modifications as required. The IoTDots-analyzer in this case extracted the log data from the IoTDots Logs Database and performed the Markov Chain model for behavior detection and user activity inference. The evaluation results of the IoTDots framework indicated that it achieved over 98% accuracy in detecting user activities and over 96% accuracy in detecting user and smart app behaviors in the smart environment. The idea to analyze the logs database was a very efficient forensic methodology and could yield better performance when implemented with IoT devices, as seen in the paper [2]. Hence, our model IDOTS+ also accesses logs of IoT devices from the OpenHAB platform and analyzes the data with a CNN model.

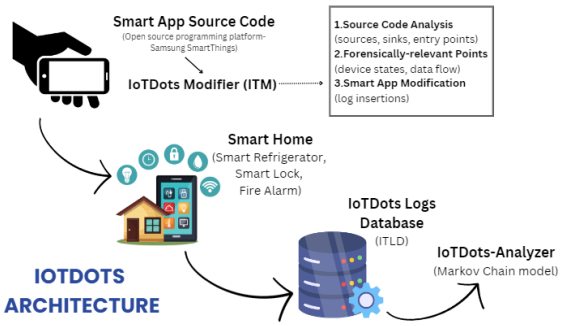

Fig 2: The IotDots Architecture [2]

3. Rules and Assumptions

Our research focuses on creating a system for logging IoT device activity and then performing forensic analysis using the IDOTS+ framework to identify potential malicious behavior. This framework involves analyzing device-generated log files with an analyzer that is implemented using the CNN model such that it can detect suspicious actions. We test the effectiveness of this approach in a smart home setup, specifically with a smart bulb Philips Smart Wi-Fi LED Model 9290023833B managed by OpenHAB on a Raspberry Pi 3. Any changes made to the smart bulb are logged on OpenHAB and then analyzed using our framework. After going through various home automation tools like Samsung SmartThings and Apple HomeKit, we decided to use OpenHAB for testing out our IDOTS+ framework on reading this paper [9] "A Comparison of Open-Source Home Automation Systems". The paper selected OpenHAB as its top 3 best tools when compared with 20 other systems with thorough use-case analysis and component feature analysis. OpenHAB is a smart home automation tool that gives users a platform to control and connect all of their smart home devices. This is done irrespective of the manufacturers of the products and the communication protocols used by the devices. With OpenHAB, users can create custom automation rules and routines to automate tasks and manage their smart home devices efficiently.

The rules that we have set for our IoT smart home are divided into 4 parts, as explained in Table 1.

Rules	Timing	Bulb On/Off	Bulb Brightness	Bulb Color
Rule 1 - Morning	8:00 am	On	25	Light blue (185,57, 100)
Rule 2 - Afternoon	12:00 pm	On	40	Warm Orange (26,68,6 6)
Rule 3 - Evening	05:00 pm	On	100	Light Purple (240,30, 66)
Rule 4 - Night	11:00 pm	Off	0	(0,0,0)

Table 1: Rules of Smart Bulb

The Fig 11 shows the rules configured in OpenHAB. Fig 12 shows the state of the Smart Bulb during each rule which can be found in Appendices section.

Throughout the day, our IoT smart home lighting system operates according to four distinct rules based on the timing, on/off status, brightness, and color of the smart bulb. In the morning, precisely at 8:00 am, the bulb switches on, emitting a light blue hue with an HSB value of (185,57,100) and a brightness level of 25, simulating natural daylight. As noon approaches at 12:00 pm, the bulb remains lit but shifts its color to a warm orange shade with an HSB value of (26,68,66) and increases its brightness to 40, providing a cozy ambiance during midday hours. By late afternoon, at 5:00 pm, the bulb changes to a vibrant light purple color with an HSB value of (240,30,66) and reaches its maximum brightness level of 100, offering ample illumination for evening activities. Finally, as night falls at 11:00 pm, the bulb dims and switches off,

marking the end of the day's lighting schedule. This is the ideal behavior of the smart bulb and deviation from this behavior is considered as malicious and will be flagged by our IDOTS+ analyzer.

Some **assumptions** we have in our project to ensure that the IDOTS+ framework is implemented efficiently are:

- Despite implementing OpenHAB and installing the smart bulb correctly, we know that both remain susceptible to cyber-attacks, as evidenced by prior research [12]. Therefore, we assume that the smart bulb is susceptible to getting compromised and exhibiting malicious behavior, which can be logged and investigated further with our analyzer.
- We assume that the logs generated by OpenHAB concerning the smart bulb's activities remain unaltered even in the event of an attack on the smart bulb itself. This assumption ensures the integrity of the data collected, allowing for accurate forensic analysis and identification of potential security breaches.

4. Design of IDOTS+

For our proposed solution Fig 3, we have implemented a smart home system centered around a smart bulb, whose functionalities are controlled through a home automation tool such as the OpenHAB platform. Within this platform, we have established rules dictating the automated and ideal behavior of the bulb. For instance, the bulb is programmed to switch on every day at 8:00 am, emitting a brightness of 25 and a light blue hue color. All activities of the bulb are meticulously logged and stored in a file on the OpenHAB platform. Upon detecting any suspicious behavior exhibited by the bulb, we promptly load the log file into our analyzer. The analyzer functions by flagging any events that deviate from the predefined rule set. This enables the fast and prompt identification of any anomalous activities and allows us to pinpoint the origin of potential attacks with ease.

To sum up, the IDOTS+ framework comprises two primary events: log generation and analysis. The former involves the systematic recording of bulb activities within the OpenHAB platform, while the latter corresponds to the utilization of our analyzer to analyze these logs for any deviations from the expected behavior.

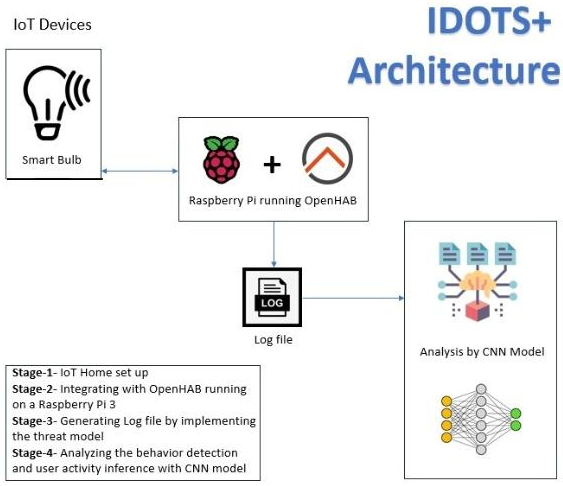

Fig 3: IDOTS+ Architecture

Generation of logs

In the OpenHAB platform, logs are generated to provide insights into system activity, including events, errors, warnings, and informational messages. The logs in OpenHAB are typically generated by various components, such as bindings, rules, and system processes. They are stored in log files located in either the userdata/log directory (for manual setups) or /var/log/openhab2 directory (for apt/deb-based setups). Two primary log files are generated by default: events.log and openhab.log.

The events.log file records events related to item state changes, while the openhab.log file contains general system logs including information about rule executions, service interactions, and errors. For our framework, we will be using the events.log file for analysis.

Log entries follow a standardized format including a timestamp, log level (e.g., DEBUG, INFO, WARN, ERROR), source of the log message (e.g., package, class), and the actual log message itself. For example, a log entry of the light switching on looks like this Fig 4.

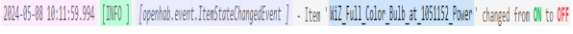

Fig 4: Log of Bulb turning Off

Here we can notice the time stamp information followed by the log level that shows that this log is an

information level. Then, it shows that an Item state changed event occurred, that is the state of the item light bulb has changed from on to off and the item we are talking about is a wiz full color bulb.

Similarly, logs are generated at a frequency of every minute. If no state change is occurring then a state update log is generated which shows that the state hasn't changed and remains the same like we can see in the Fig 5.

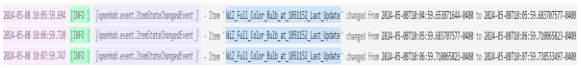

Fig 5: Periodic log update of the state of Smart Bulb

These logs can then be downloaded from the interface into an excel sheet and they will have the following columns- timestamp, info, event and item like we can see in the Fig 6.

Fig 6: Log file downloaded locally

These logs are then loaded into the analyzer for it to be flagged as malicious if a potential event has happened that is against the rule set.

IDOTS+ Analyzer

Our analyzer is a machine learning model coded in Python Programming Language that utilizes a Convolutional Neural Network (CNN) architecture to predict the malicious instances associated with log data from an OpenHAB system. The model is designed to process time-stamped events, including item state changes, item commands, and other relevant information. So, we trained our model by flagging benign instances as 0 and malicious instances as 1 in an additional column called “Flag” in the train dataset. The CNN model is defined using the Keras library in Python, and it starts with a Conv1D layer, which is a 1D convolutional layer that applies 32 filters with a kernel size of 3 to the input data. This convolutional layer is responsible for extracting local features like patterns and relationships between the event types, item names, timestamps, and their corresponding flags from the train data. The code snippet Fig 8. can be referred in the Appendices section.

5. Experimental Results

The analyzer performed well, achieving an accuracy rate of 89% when subjected to the test dataset. The analyzer was trained with a dataset composed of logs spanning two entire days, capturing the bulb's activities from morning to night, as logged by OpenHAB's events.log. To simulate real-world scenarios, deliberate deviations from the established rule set were introduced by another connected device interacting with the bulb. These deviations primarily consisted of actions where the light bulb was switched off and on and then the color was changed to “Green”, which is against the rules set.

Following the generation of the two-day log data, a new column titled "Flag" was appended to the Excel file. This column manually labeled each activity as either benign (labeled as 0) or malicious (labeled as 1). Subsequently, this labeled dataset was utilized as the training data for the analyzer, which used CNN for the training process.

Once trained, the analyzer was tested with logs from another day, comprising a mixture of both benign and malicious activities. The dataset was then fed into the analyzer for evaluation. Impressively, the analyzer accurately assigned flag values to 89% of the logs, correctly identifying benign activities as 0 and malicious activities as 1. Additionally, as the analyzing process is performed in a separate local environment, there is no performance overhead that is generated on the smart bulb. For performance metrics, we used parameters like accuracy and precision, which can be seen in the graph Fig 9. The accuracy and precision metrics chart is in Appendices Fig 10.

6. Challenges and Limitations

Challenges:

The primary challenges that we faced while doing our framework were:

- Creating the analyzer presented a significant hurdle, especially in getting the timestamps to align properly with the datatype needed for doing the CNN model. This proved to be a struggle because of the complexities involved in ensuring that the timestamps were formatted correctly to fit the model's requirements.

- Initially, we aimed to integrate both the motion sensor and the light bulb to create a more advanced smart home setup. However, this integration was unsuccessful because of

the restricted time frame we had for implementation and generating logs for both the devices and analyzing them was a very sophisticated and complex model.

Limitations:

- One major limitation of our project is the restricted dataset size. We only had data for two days for training and one day for testing. This limitation severely restricts the model's exposure to diverse scenarios and may hinder its ability to perform well in real-world situations.
- Another limitation is the presence of only a smart bulb within our ecosystem, without any other devices. This decision was made to keep the model simple and to accommodate the time constraints of implementation. However, this limitation may limit the model's applicability to broader IoT scenarios and could affect its effectiveness in addressing complex smart home environments which are found in the real world.

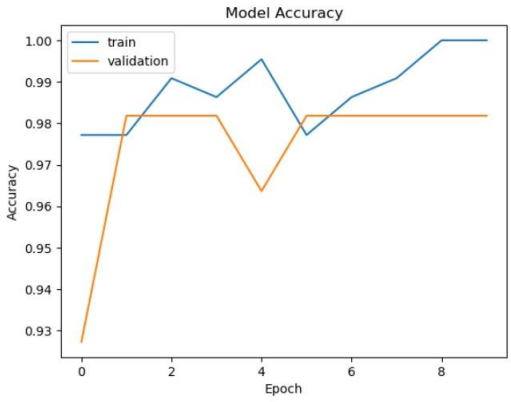

Fig 9: Performance Evaluation

7. Future Scope

To address these limitations, future efforts could focus on expanding the ecosystem to include a wider variety of devices mimicking the ones that are there in the real world. This would allow for testing and training the model with a larger and more diverse dataset. Moreover, the test and train data is small. One of the future works could be having the computer resources and environment to create an efficient analyzer such that data from weeks and months can be parsed through and analyzed easily. Additionally, integrating the model with other popular platforms such as Samsung SmartThings and Apple HomeKit and not restricting it to the OpenHAB could enhance its versatility and applicability within the IoT domain. These expansions would improve the model's effectiveness and adaptability, enabling it to handle a wider range of smart home environments and use cases.

8. Conclusion

The IDOTS+ framework presents an innovative approach to tackle the unique challenges of IoT forensic investigations. By integrating the OpenHAB platform with a machine learning-based log analyzer, IDOTS+ delivers a scalable and adaptable solution for identifying potential compromises within smart home IoT environments. A key strength of IDOTS+ lies in its ability to scrutinize device logs using a CNN model trained to spot deviations from an established set of rules. This empowers investigators to pinpoint the precise timing and nature of detected incidents with enhanced accuracy. While the current implementation focuses on a smart bulb within a smart home, future work could explore integration with additional IoT devices and platforms to further extend its applicability. Overall, IDOTS+ represents a promising advancement in IoT forensics, providing a practical and effective solution to tackle the challenges unique to this domain.

References

[1] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis and E. K. Markakis, "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues," in IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1191-1221, Secondquarter 2020, doi: 10.1109/COMST.2019.2962586.

[2] L. Babun, A.K. Sikder, A. Acar, A.S. Uluagac, IoTDots: A Digital Forensics Framework for Smart Environments(2018), doi: https://doi.org/10.48550/arXiv.1809.00745.

[3] E. Al-Masri, Y. Bai and J. Li, "A Fog-Based Digital Forensics Investigation Framework for IoT Systems," 2018 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 2018, pp. 196-201, doi: 10.1109/SmartCloud.2018.00040.

[4] G. Surange and P. Khatri, "IoT Forensics: A Review on Current Trends, Approaches and Foreseen Challenges," 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2021, pp. 909-913.

[5] H. Zhou, L. Deng, W. Xu, W. Yu, J. Dehlinger and S. Chakraborty, "Towards Internet of Things (IoT) Forensics Analysis on Intelligent Robot Vacuum Systems," 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA), Las Vegas, NV, USA, 2022, pp. 91-98, doi: 10.1109/SERA54885.2022.9806735.

[6] A. Ahmed, S. U. Din, E. Alhanaee and R. Thomas, "State-of-the-art in IoT forensic challenges," 2022 8th International Conference on Information Technology Trends (ITT), Dubai, United Arab Emirates, 2022, pp. 115-118, doi: 10.1109/ITT56123.2022.9863965.

[7] Mohammed, Hanya & Koroniotis, Nickolaos & Moustafa, Nour. (2023). Digital Forensics based on Federated Learning in IoT Environment. 92-101. 10.1145/3579375.3579387.

[8] A blockchain-based decentralized efficient investigation framework for IoT digital forensics

Ryu, J.H., Sharma, P.K., Jo, J.H. et al. A blockchain-based decentralized efficient investigation framework for IoT digital forensics. J Supercomput 75, 4372–4387 (2019). https://doi.org/10.1007/s11227-019-02779-9

[9] B. Setz, S. Graef, D. Ivanova, A. Tiessen and M. Aiello, "A Comparison of Open-Source Home Automation Systems," in IEEE Access, vol. 9, pp. 167332-167352, 2021, doi: 10.1109/ACCESS.2021.3136025.

[10] M. Harbawi and A. Varol, "An improved digital evidence acquisition model for the Internet of Things forensic I: A theoretical framework," 2017 5th International Symposium on Digital Forensic and Security (ISDFS), Tirgu Mures, Romania, 2017, pp. 1-6, doi: 10.1109/ISDFS.2017.7916508.

[11] Y. Salem, M. Owda and A. Y. Owda, "A Comprehensive Review of Digital Forensics Frameworks for Internet of Things (IoT) Devices," 2023 International Conference on Information Technology (ICIT), Amman, Jordan, 2023, pp. 89-96, doi: 10.1109/ICIT58056.2023.10226145.

[12] M. Ramljak, "Security analysis of Open Home Automation Bus system," 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2017, pp. 1245-1250, doi: 10.23919/MIPRO.2017.7973614.

Appendices

Fig 7: Raspberry Pi 3 and Smart Bulb

The figure above is the Raspberry Pi 3 device used to host the OpenHAB framework. The Philips Smart Wi-Fi LED bulb was connected to the OpenHAB application.

```
# Define CNN model
model = Sequential([
    Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

Fig 8: Code Snippet

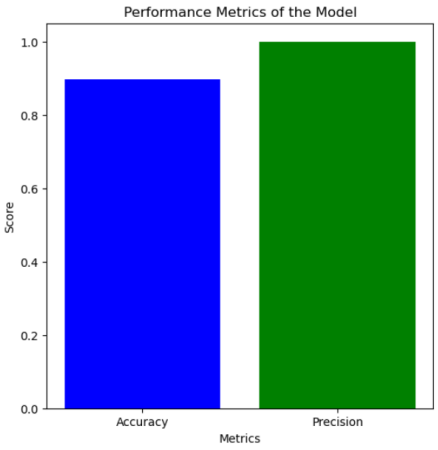

Fig 10: Accuracy and Precision Metrics

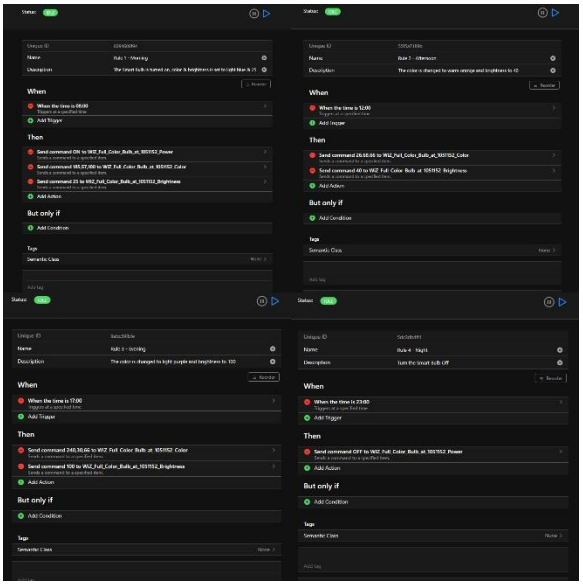

Fig 11: Rules configured in OpenHAB

Rule 1 – Morning: The Smart Bulb is turned on, color & brightness is set to light blue & 25
Rule 2 – Afternoon: The color is changed to warm orange and brightness to 40
Rule 3 – Evening: The color is changed to light purple and brightness to 100
Rule 4 – Night: Turn Off Smart Bulb

Fig 12: State of the Smart Bulb

Top Left: State of Smart Bulb in Morning
Top Right: State of Smart Bulb in Afternoon
Bottom Left: State of Smart Bulb in Evening
Bottom Right: State of Smart Bulb in Night