IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, November 2022
Impact Factor: 6.252

Autonomous IoT System Security Capability: Pushing IoT Security to the Edge

Mundhe Prachi N.[1] and Dr. Rokade M. D.[2]
Department of Computer Engineering[1,2]
Sharadchandra Pawar College of Engineering, Dumbarwadi, Otur, Maharashtra, India

Abstract: *Complex systems of IoT devices (SIoTD) are systems that have a single purpose but consist of multiple IoT devices. These systems are becoming ubiquitous, have complex security requirements, and face a diverse and ever-changing array of cyber threats. Privacy and bandwidth issues will prevent all data from these systems from being sent to a central location, so these systems cannot completely rely on a centralized cloud service for their security. The security of these systems must be provided locally and autonomously. In this paper, we describe the capability to address this problem, explain the system specifications, present our work to enable SIoTD, and show initial results of a novel edge-based machine learning application to build this capability.*

Keywords: IoT, Machine Learning, Assured Autonomy, Edge, Security

I. INTRODUCTION

Complex Internet of Things Device (SIoTD) systems are becoming more common as IoT devices proliferate in our society (eg our homes, smart cars, smart offices, smart buildings, smart hospitals and smart cities). These systems often have complex security requirements and face a diverse and ever-changing array of cyber threats. Devices may come and go from these systems at any time, and some systems may only be intermittently connected to central processing nodes (clouds). The impending increase in data transfer speeds (5G) and ever-changing network requirements will further complicate the network defender's job of maintaining situational awareness and ensuring security for these systems.

In addition to these issues, many SIoTD networks are segmented, meaning that data from all devices may not be sent back to one location. So these systems cannot rely on a centralized cloud service for their security. Rather, the protection of these systems must take place locally and in an autonomous or semi-autonomous manner.

To get the full picture of SIoTD and to meet security and related analytics requirements, processing will need to occur close to the point of data generation. This area is known as the edge.

- There are three data-centric use cases related to SIoTD that drive IoT situational awareness and system security to the edge:
- IoT applications will generate so much localized data, making it impractical to send all data back to a central location.
- System owner concerns about data privacy will limit the storage of sensitive data in the public cloud. System owners will want to be in control of their personal, valuable data.
- Many IoT use cases have strict timing requirements that cannot tolerate large latencies based on round-trip timing from the point of data generation to the cloud and back. These cases require on-site data processing and information generation.

This paper describes ongoing work to explore new techniques to build the capability to provide real-time, edge-oriented, automated situational awareness and assurance to IoT networks.

Our contributions are as follows: (1) we identify a gap in SIoTD security and propose solutions to fill this gap, (2) we experimentally detail the machine learning (ML) pipeline and the development of the corresponding model, and (3) we present the results of porting assurance algorithms from server to a small size, weight and power (SWaP) device that runs autonomously at the edge located at the point of data generation.

The remainder of the paper is organized as follows. Part II provides an overview of the types of automation that govern this capability. Section III discusses related works. Part IV outlines the IoT security architecture. Part V describes

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, November 2022
Impact Factor: 6.252

element selection, engineering and generation activities. Section VI introduces the IoT data pipeline and related design. Section VII presents an overview of model selection and training. Section VIII describes the results. Finally, Section IX presents conclusions and discusses future work.

II. TYPES OF AUTOMATION

AI initiatives can be broadly categorized by their goals [2]. Two of the most common are:

- Automation of Processes
  - Automating digital and physical tasks using "robotic (physical or virtual via computer code)" process automation technologies.
- Cognitive insight
  - Using algorithms to detect patterns in huge volumes of data and interpret their meaning

To meet the needs of this system, we will use elements of process automation and cognitive insights. This capability will operate in a semi-autonomous manner, acting as an agent or virtual robot, and consists of a collection of related hardware, frameworks, algorithms, and peripherals. This capability will essentially be an autonomous implementation of CHASM for SIoTD [3].

III. RELATED WORK

A literature survey was conducted to gather a list of relevant features and techniques that could be useful for fingerprinting/profiling IoT devices.

Meidan et al. [4] extracted features from TCP sessions and trained a multi-class Random Forest classifier to identify whitelisted IoT devices and discriminate between known and unknown devices.

Shahid et al. [5] presented an ML-based approach to identify IoT devices by analyzing the streams of packets sent and received with an overall accuracy of up to 99.9% in their tests and using random forest classifiers.

Truong et al. [6] used a recurrent neural network to identify and classify IoT devices in network traffic.

Buczak et al. [7] used Random Forest augmented with the statistical moment of mean, variance, skewness, and kurtosis to develop an ensemble algorithm to identify the presence of hidden DNS tunnels in network traffic.

Sivanathan et al. [8] used Random Forest in a smart campus environment equipped with various IoT devices for 3 weeks to distinguish IoT devices from non-IoT devices and classify IoT devices with 95% accuracy.

Bai et al. [9] used a LSTM-CNN cascade model to automatically identify the semantic type of devices to achieve automatic device classification in the network traffic flows of IoT devices.

Bezawada et al. [10] extracted a wide range of features from packet headers to create device fingerprint vectors and used Gradient Boosting to classify IoT devices.

The features selected for implementation are listed in Table I.

IV. IOT SECURITY ARCHITECTURE

Our goal in this endeavor is to train models that can autonomously identify IoT device types. We want these models to perform well both in a controlled lab environment and in the real world. Therefore, it is essential that we train the models on datasets that include a large number of IoT devices with a large amount of traffic from each one. It is believed that when classifying devices (i.e. guessing which category a device belongs to), it would also be best to have many different devices in each category so that the model does not over-fit the behavior of a few devices.

An IoTtestbed containing 56 IoT devices of various types was built. IoT devices were installed in a network demilitarized zone (DMZ), also referred to as an isolated perimeter network, that had access to the Internet. This is because the devices were untrusted but needed internet access to function properly. We used the span port on the switch to send all network traffic to the Jetson Nano, which ran tcpdump and archived the data in PCAP format. This data was then transferred to a local local "cloud" server outside the DMZ to be used as training data for the models

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, November 2022
Impact Factor: 6.252

4.1 Feature Selection, Engineering and Generation

When determining what features to extract from the network packet data, it is important to remember that many IoT devices already use encryption methods such as TLS to secure their traffic against eavesdropping, and that more devices will use this type of security in the future. It would therefore be naïve to assume that features extracted from unencrypted packet payload data will remain useful in the future. Rather, the features selected should be those that can be easily extracted from both encrypted and unencrypted traffic. For example, features extracted from packet headers such as packet length, IP lifetime, TCP window size, and payload entropy can be extracted from packets regardless of whether the content is encrypted or not..

Additionally, any features used for IoT discovery can also be useful for IoT profiling, threat detection, and anomaly detection (future work).

4.2 Data Pipeline Design

Successful edge deployment involves more than simply pushing a trained model to the edge device to be executed. The feature extraction code and other data processing logic that is part of the associated analysis pipeline must also be ported to run at the edge. This is because these processing components transform raw sensor data into a form that is ready for ML processing. Therefore, any feature extraction or data processing code that is part of the analytics pipeline must be designed to either be massively parallelized in a central data center for training purposes, or pushed to the edge to run on a less capable computer. platform. If the code can run in the same form on both types, that's another productivity bonus platform without modification, simplifying the deployment process.

With this in mind, the feature extraction and processing pipeline for this system was designed to be portable from the ground up (Fig. 2), so deploying the entire analysis processing pipeline to the edge was easy. We implemented feature extraction in C++ on top of the open source library libtins [12], which provides efficient methods for collecting and analyzing network packet data. The upper software layers that aggregate packet data by device and transform the data into ML-ready vectors were written in Python using a combination of scikit-learn, pandas, and custom Python code. Keras and TensorFlow were used for model training and inference.

These software modules have been combined into a reusable Python package called iot-ai, which allows the user to create modular data pipelines for downstream ML development. Since model training is a batch operation and model execution on a real network is ideally a streaming operation, the iot-ai library was designed to allow streaming. Batch operations are handled as streaming by streaming individual lines of static files over a channel. This reduces system complexity by unifying two very similar processing paths and code bases into one. With this design, no code conversion is needed when porting the feature extraction pipeline from batch mode in the cloud (for training) to streaming mode at the edge (for live inference).

The nodes in the pipeline have flexible interfaces and can be combined in different ways to facilitate the many operations typically involved in training an ML model on network data. Furthermore, individual pipeline nodes can be parallelized to take advantage of multi-core CPUs when available.

In addition, the libtins library provides the ability to read packets from a PCAP file or to read packets directly from the network interface, meaning that our pipeline can easily be converted from handling static data to running on live data.

The result is a modular pipeline design that can be easily reused for data collection at the edge, training in the cloud, model execution in the cloud, and model execution at the edge.

V. SELECTION OF MODELS AND TRAINING

The chosen model architecture was a recurrent neural network consisting of a single LSTM layer of 128 units, followed by a softmax output of width 10 (one for each device category). The LSTM layer was fed with fixed-length input sequences of 20 feature vectors, one vector per network packet.

The model was trained on PCAP data that was collected from an IoT test device over a period of 3 months (Fig. 3). This included data from 56 different IoT devices and amounted to 94 GB of PCAP (about 262 million packets). This dataset was then split into 80% for training, 19% for validation during training, and 1% hold for final model performance evaluation.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, November 2022
Impact Factor: 6.252

Traffic from each IoT device was grouped by MAC address and arranged in chronological order. The data from each device was then transformed into sequences of 20 packets. The values in these sequences were then normalized, encoded, and labeled by device category (eg, this sequence of 20 packets is from a camera, this next sequence of 20 packets is from a thermostat, etc.).

VI. RESULTS

The initial ML model results were different for our original data set. However, the change observed was in speed of execution, not accuracy.

A. Model Execution and Results (On the Server)

The model was successfully trained on a single Nvidia Tesla V100 GPU in less than 1 hour and achieved 93% accuracy on each of the training, validation, and maintenance datasets. Detailed results of the model's performance on the stamina set for each category are shown in Table II.

B. At the time of writing, the volume of traffic generated by IoT devices in the test environment is relatively small (20 MB every 10 minutes). With a parallelization level of 3×, the end-to-end data pipeline including the trained model can process the aforementioned 10 minutes of network data in 47 seconds on a large VM server with 30 Intel® Xeon® processors at 2.50 GHz and 512 GB of RAM. In this configuration, the process pipeline consumed only six CPUs and 3 GB of RAM.

C. Port to Low-SWaP Devices

D. A Raspberry Pi 3B with a quad-core, 1.2 GHz Broadcom, 64-bit ARMv7 CPU and 1 GB of RAM was obtained to implement the edge model. Tensorflow2 was installed, followed by the iot-ai library and its dependencies (libtins, libpcap, etc.). This device was then connected to the same network as the IoT device, and a span port was set up on the network switch to forward all network traffic from each device to the Raspberry Pi (Fig. 4).

E. Results of Model at Edge

On the edge platform, the data channel was able to handle 10 minutes of traffic on the test device in 6 minutes. While this is encouraging and indicates that the edge device is currently able to keep up with a live stream of test data, this data volume is still somewhat small.

The execution speed of the pipeline at the edge could probably be improved by running the model on an ML-focused edge device such as the NvidiaJetson TX2, or by porting the model to TensorFlow Lite, which Google claims will soon offer support for fixed sequence length LSTM networks [13].

VII. CONCLUSIONS AND FUTURE WORK

These results indicate that it is possible to run IoT device discovery, classification and verification models at the edge of low SWaP devices; however, we feel there are areas of improvement that can be explored to increase system performance.

7.1 Tuning the Model

We did not perform extensive hyperparameter tuning, actively train individual IoT devices to collect traffic during all phases of device operation, or normalize the number of training examples from each device category. This could be explored in future work.

7.2 Complex Network Topologies

The network topology used in this effort was relatively straightforward; all devices were connected to the same router and network traffic was captured on that router. In future work, we intend to apply the concepts and systems developed in this effort to more complex network topologies.

7.3 Model Training at the Edge

We believe that, over time, advances in low-SWaP devices will enable model training at the edge to occur concurrently with model execution. We will explore this in future work

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, November 2022
Impact Factor: 6.252

7.4 Analytics Files

The IoT environment is complex and evolving; new devices appear all the time. Therefore, training a single model for IoT Discovery would be problematic, as there would be a constant need to retrain that model as new types of devices are (often) produced.

Two other challenges to the single-model approach are the inherent trade-offs between generating results quickly and observing the device long enough to make good predictions, and the fact that individual models can be biased, fragile, or error-prone in certain situations.

Combining multiple models into a file can help in this regard. If some models are noisy or inaccurate due to the rapidly changing environment of the device, they can be balanced by others. For example, if four models say the device is a camera and one says something else, it's probably a camera.

Using an ensemble approach would also allow some models to focus on fast but less accurate predictions and others on much slower but more accurate predictions. The user would thus receive a notification practically immediately and the result would be refined over time. Allowing the user to browse and see results from individual models would provide an additional level of explanation in these situations.

Other findings of this work include the following:

a. The same analytics that were designed for IoT device discovery can also be used for IoT device authentication and threat detection.

Example 1: All file analysts agree that device X is a camera, but suddenly half of them now claim it is a thermostat.

i. The device may have started behaving differently (may indicate a potential compromise)

i. OR it means a problem with the models

Example 2: None of the file parsers match on device X.

This may be a new type of device that we have not seen before and further investigation will be required

ii. We believe these results indicate that a distributed, edge-based, semi-autonomous capability could be developed and deployed in complex, dynamic SIoTD environments, and the associated model may be able to maintain good performance even when the model is presented with data that is not trained. We will explore this in future work.

ACKNOWLEDGMENTS

The authors would like to thank Tracy Herriotts, Lien Duong, Christine Piatkova, and Barbara Johnson of JHU/APL for providing data science, cyber analysis expertise, and documentation assistance during the development of this paper.

REFERENCES

[1]. Real-Life Use Cases for Edge Computing - IEEE Innovation at Work.(n.d.). Retrieved from https://innovationatwork.ieee.org/real-life-edge-computing-use-cases/,accessed 05-Feb-2020

[2]. Monika D. Rokade, Dr. Yogesh kumar Sharma, ”Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic.”IOSR Journal of Engineering (IOSR JEN),ISSN (e): 2250-3021, ISSN (p): 2278-8719

[3]. Monika D. Rokade, Dr.Yogesh kumar Sharma” MLIDS: A Machine Learning Approach for Intrusion Detection for Real Time Network Dataset”, 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), IEEE

[4]. Monika D. Rokade, Dr. Yogesh Kumar Sharma. (2020). Identification of Malicious Activity for Network Packet using Deep Learning. International Journal of Advanced Science and Technology, 29(9s), 2324 - 2331.

[5]. Sunil S. Khatal, Dr. Yogesh kumar Sharma, “Health Care Patient Monitoring using IoT and Machine Learning.”, IOSR Journal of Engineering (IOSR JEN), ISSN (e): 2250-3021, ISSN (p): 2278-8719

[6]. Sunil S. Khatal, Dr.Yogesh kumar Sharma, “Data Hiding In Audio-Video Using Anti Forensics Technique For Authentication ", IJSRDV4I50349, Volume : 4, Issue : 5

[7]. Sunil S. Khatal Dr. Yogesh Kumar Sharma. (2020). Analyzing the role of Heart Disease Prediction System using IoT and Machine Learning. International Journal of Advanced Science and Technology, 29(9s), 2340 - 2346.