AI/ML Anomaly Detection Analytics for Wireless Internet of Things Systems

Stephanie Polczynski, Samuel Fox, Erik Brasile, NSA
Dr. Robert Simon, George Mason University

[Photo credit: iStock.com/NatalyaBurova]

Around the world, critical infrastructures increasingly incorporate Internet-of-Things (IoT) devices and networks. Since, by definition, most IoT devices are connected to national if not global networks, the explosive growth of IoT systems introduces a vast new set of potential vulnerabilities and exploitable cyber threat vectors. IoT cyberattack classes range from injecting false or misleading sensor data and distributed denial-of-service attacks to long-term stealthy advanced persistent threats (APTs).

This article describes our efforts to provide tools to detect potentially compromised IoT devices based on behavioral analysis of their local wireless [e.g., Bluetooth, Bluetooth Low Energy (BLE), ZigBee, etc.] communications. We have integrated these tools into a project called the Wireless IoT Monitoring System (WIMS). We adopt a classification-based machine learning (ML) approach, whereby device-based IoT packet exchanges are encoded into a set of images. Our ML models are trained to recognize well-behaving devices and identify anomalous traffic through these generated images as an indication of a potentially compromised device or system.

The wireless IoT protocol landscape consists of dozens of over-the-air protocols based on both open-source and proprietary standards. While many types of attacks against older protocols such as Bluetooth have been identified, attacks against newer protocols such as long-range wide area network (LoRaWAN) and narrowband (NB)-IoT are less understood and the threat space and exploit scenarios are not as well-defined. To address this discrepancy, we leverage transfer learning (TL) techniques. Using TL, we can train an anomaly detection classifier for a well-known class of attacks, say against Bluetooth, and reuse that knowledge to shorten the training time and improve the accuracy of anomaly detection for a newer protocol, such as LoRaWAN.

We first provide a brief background into the wireless IoT threatscape, and then discuss our IoT monitoring system. Next, we describe how we transform IoT packet captures into picture images representing encoded time series data from the network traffic and use ML on those images to identify anomalous activity. Thereafter, we explain how we use TL to rapidly train image classifiers on new protocols and attack classes. Lastly, we provide some experimental results and offer conclusions.

Threatscape for wireless IoT

There has been limited work on using ML to monitor wireless IoT traffic and detect suspicious behavior. Current solutions primarily focus on wired or 802.11 wireless IoT devices and focus on the communication patterns between an IoT device and the host server. For example, one might monitor the encrypted web communications of a smart home hub/gateway from the router to the manufacturer's cloud server. These solutions would examine features like the number of times the device usually communicates with the server, average data transmission size, frequency of communications, etc. However, these solutions are not examining the local wireless communications of the IoT devices. IoT end devices often use protocols like Bluetooth, ZigBee, Z-Wave, Thread, NB-IoT, LoRaWAN, and now 5G to communicate with the managing device, hub, gateway, or cloud server.

Monitoring traffic over these protocols requires special equipment that is often designed for use by radio-frequency (RF) experts and is not tailored for use by the average network analyst and certainly not designed to be used for long-term monitoring and detection. Further, IoT devices typically lack traditional security mechanisms such as firewalls, antivirus programs, system logs, etc. that can be used to prevent, detect, and analyze potential malicious activity. Because of this, we sought to develop a wireless IoT monitoring system that performs anomaly detection using what is often the only source of information available from wireless IoT devices—the packets they transmit to communicate between other IoT end devices and IoT gateways.

To fully grasp the IoT problem set, it is useful to understand the scale of the IoT space from the number of protocols available and in use by technology developers, to the complex networks spanning multiple protocols. The number of IoT protocols is a relatively difficult number to pin down. A quick Internet search of "How many IoT protocols are there?" yields results of anywhere from the top six IoT protocols to 26 IoT protocols [1]. Most of these articles also acknowledge that their lists are "extensive—but not exhaustive." As wireless technologies improve, new protocols are constantly being developed to cater to the specific needs of mobile networks. The Fifth Generation of Mobile Telephony (5G) standard, for example, was "functionally frozen in June 2018 and fully specified by September 2019." 5G promises support for a "massive Internet of Things" integrating "the operational aspects that apply to the wide range of IoT devices and services anticipated in the 5G timeframe" [2]. On the other hand, some IoT sensor devices have been in use for decades and will continue to be so for many more years, without the opportunity for updates. Therefore, IoT-enabled devices often have support for not only new technologies, but also legacy protocols.

In addition to the number individual protocols available for use, many modern IoT networks make use of several IoT protocols to provide better quality of service (QoS) to their users. Amazon Sidewalk, for example, uses (at a minimum) BLE [2.4 gigahertz (GHz) industrial, scientific, and medical (ISM) band] for short-range communications and LoRa [900 megahertz (MHz) ISM band] for long range [3]. Many devices support multiple methods of communication and automatically switch between the protocols based on any number of metrics determined by the device manufacturer. Each protocol a device supports is another communication link and therefore potential attack vector for malicious actors.

How does an organization protect itself when incorporating devices with IoT interfaces into its operations? Simply not using devices with IoT capabilities is likely the safest option; however, this option is growing increasingly difficult and impractical. It is not uncommon for device manufacturers to include one or more wireless protocol interfaces. There are many reasons to do so from a manufacturer's perspective to include, QoS (as discussed previously), redundant links for firmware updates, mesh network support, device metrics, information collection, etc.

Wireless IoT monitoring options

Before developing analytics to perform behavioral analysis of wireless IoT traffic, a process first needed to be developed to collect the traffic from the IoT devices. There are two primary means of doing this currently. The first is using commercially available IoT protocol analyzers/sniffers, which automatically detect and demodulate supported IoT signals and output the observed traffic in an easily parseable, often packetized format for easy analysis of the data. The second method is using software-defined radios (SDRs) which provide a generic radio front end that captures raw samples of the RF spectrum. These devices allow for adjustable tuning and instantaneous bandwidth to cover nearly any portion of the RF spectrum; however, they require specialized expertise to operate to obtain the IoT data.

IoT protocol analyzers

Commercially available protocol analyzers/packet sniffers are available to monitor the most common wireless IoT protocols to include Bluetooth Classic [basic rate/enhanced data rate (BR/EDR)], BLE, Wi-Fi, and 802.15.4-based protocols such as Zigbee and Thread [4, 5, 6]. Many implementations are hardware-based solutions that cater to specific frequency bands or specific protocols. Higher-end wireless sniffers can often detect packets from several different protocols, perform simultaneous collection on each of them, while also providing the raw RF data stream. Example solutions include the Spanalytics PANalyzer, the Ellisys Vanguard, and Frontline X500, all of which target the 2.4 GHz ISM band.

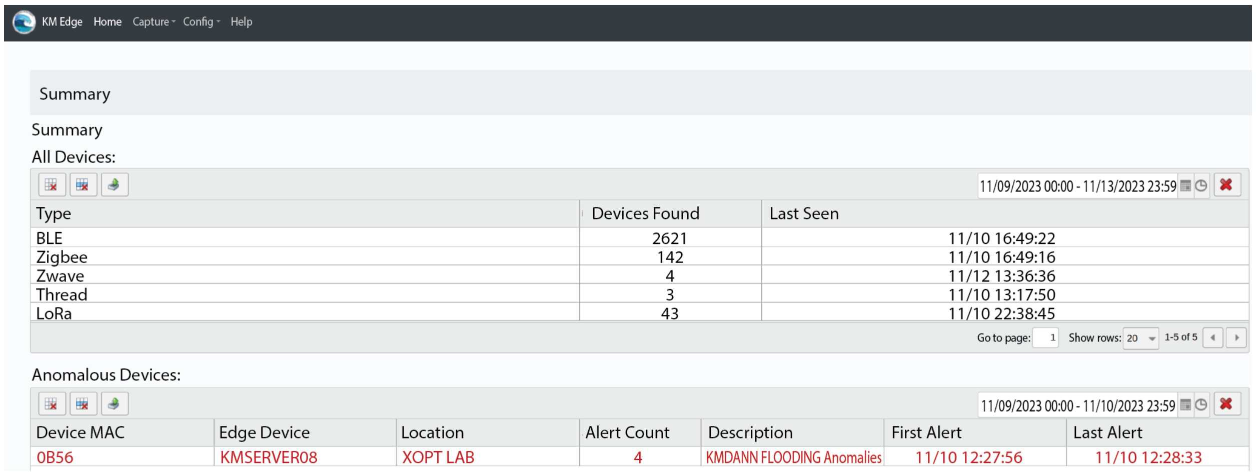
The difficulty of dealing with hardware-based RF solutions is that in order to adapt to new protocols or protocols in a different frequency band, additional hardware or at a minimum new firmware/drivers must be pushed to all devices. Adding new hardware introduces many considerations into a system's design to include physical footprint, complexity (e.g., potentially multiple manufacturers/interfaces), power consumption, additional cost, maintenance, etc.

These commercial-off-the-shelf (COTS) packet sniffers are a great option for targeting the most common protocols being used. With that being said, because of the vast number of protocols in the IoT space, quite a few are not supported or require additional hardware to cover. Each of the manufacturers previously listed provide additional hardware expansions to provide support for additional popular protocols in the 900 MHz ISM band such as LoRa and Z-Wave. This however still leaves quite a few protocols unsupported and again raises the issues of adding additional hardware to the system architecture.

Software-defined radios

To attempt to fill in the gaps and solve some of the hardware scalability issues, software-defined radios (SDRs) provide flexibility and adaptability. SDRs provide a "generic" radio front end that captures raw samples of the RF spectrum. These devices allow for adjustable tuning and instantaneous bandwidth to cover nearly any portion of the RF spectrum needed. Many provide onboard field-programmable gate arrays (FPGAs) for use as signal processors and provide high-speed interfaces such as universal serial bus (USB), Ethernet, peripheral component interconnect express (PCIe), etc. for offloading to a separate processor. These separate processors can make use of modern processor units [central processing units (CPUs), graphics processing units (GPUs), etc.] to process wideband captures of the spectrum in real time. Because the SDR front end is relatively generic, the system used to process the RF sample stream can be upgraded to meet the needs of the signal processing toolchain. Additionally, the architecture of SDR-based systems makes processing multiple protocols in parallel relatively simple.

Because the radio front-end interface is abstracted from the processing unit, developing signal packet sniffers becomes extremely modular and software based. Signal processing toolchains can be written in any language supported by the processing unit which is likely running a modern operating system and can support almost any compiler. GNU Radio, for example, is a popular open-source SDR development framework commonly used by hobbyists and researchers to implement signal processing toolchains.



Summary

Summary

All Devices:

11/09/2023 00:00 - 11/13/2023 23:59

Type	Devices Found	Last Seen
BLE	2621	11/10 16:49:22
Zigbee	142	11/10 16:49:16
Zwave	4	11/12 13:36:36
Thread	3	11/10 13:17:50
LoRa	43	11/10 22:38:45

Go to page: 1 Show rows: 20 1-5 of 5

Anomalous Devices:

11/09/2023 00:00 - 11/10/2023 23:59

Device MAC	Edge Device	Location	Alert Count	Description	First Alert	Last Alert
0B56	KMSERVER08	XOPT LAB	4	KMDANN FLOODING Anomalies	11/10 12:27:56	11/10 12:28:33

FIGURE 1. This Wireless IoT Monitoring System (WIMS) dashboard provides an overview of which IoT protocols and devices were observed in the coverage area. It also displays alerts for devices with detected anomalous traffic.

GNU Radio is built on C/C++ and Python and greatly simplifies the work required to develop an SDR protocol sniffer implementation. This provides an additional benefit due to the availability of open-source implementations of nearly every IoT protocol. Adding support for a protocol could be as simple as installing an open-source GNU Radio module.

To further increase modularity, a management framework could be used to abstract the front-end radios from any signal processing units. A management framework would control distributed antennas as well as multiple SDRs monitoring multiple portions of the spectrum in different locations simultaneously. This management framework would be able to control not only the portions of the spectrum being monitored but also control the protocols being processed. An SDR solution is a major improvement over an IoT protocol analyzer since it is extremely modular and adaptable to the specific requirements of an organization.

Distributed monitoring architecture

For the initial effort, we utilized commercial IoT protocol analyzers/sniffers due to their ease of use and support for the most common wireless IoT protocols (e.g., Bluetooth, BLE, ZigBee, etc.). However, as we continue to add support for additional wireless IoT protocols to the system which do not have commercially available sniffing tools, we are also adding SDR support to enable the processing of further protocols.

To enable the use of artificial intelligence (AI)/ML anomaly detection analytics for wireless IoT traffic, we designed a distributed architecture and the supporting software to monitor the wireless IoT traffic over an area. The area to be monitored could be a room, building, campus, etc. if sufficient IoT traffic sniffers were placed throughout the area to ensure adequate coverage to collect the wireless IoT traffic. The sniffers need to be distributed optimally through the space due to low power and low-transmission range of most wireless IoT protocols. These distributed sniffers will send all observed wireless IoT traffic to a central server for further processing. This system is called the Wireless IoT Monitoring System (WIMS).

The central WIMS server provides two primary functions:

1. **Situational awareness into which wireless IoT protocols and devices are observed in the coverage area.** This includes a count of how many devices of each protocol were observed over a set time period, the addresses of the devices, and the sniffer(s) that detected the device to aid in geolocation. Additionally, if the end device uses a resolvable public media access control (MAC) address—the system will provide as much identifying information on the device as possible.
2. **ML-based anomaly detection analytics on the observed IoT traffic.** If a potential anomaly is flagged, it will be displayed on the alert dashboard, can be clicked-into for further information, and alerts can optionally be forwarded to other security tools or to designated security officials.

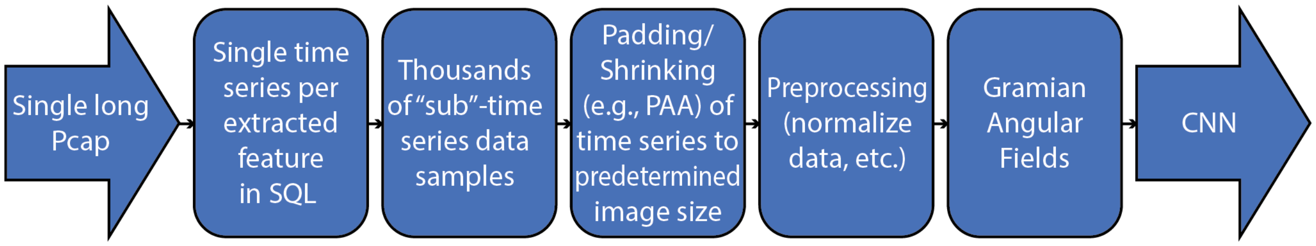

Single long Pcap
Single time series per extracted feature in SQL
Thousands of "sub"-time series data samples
Padding/ Shrinking (e.g., PAA) of time series to predetermined image size
Preprocessing (normalize data, etc.)
Gramian Angular Fields
CNN

FIGURE 2. In this ML processing pipeline, the IoT data is ingested in PCAP form, converted into a time series for each feature, and then pre-processed prior to being transformed into a Gramian angular field (GAF) picture and inputted into a CNN for anomaly detection.

Anomaly detection approach

Wireless behavioral monitoring systems such as WIMS must ingest and process large numbers of IoT packet data streams. Sophisticated IoT cyberattacks may only be detectable by understanding the statistical relationships both within each packet and between packets in each data stream. Given the sheer number of packets and complex interrelationships between packets inside of data streams, current ML approaches are particularly attractive since they automatically learn to classify individual datasets into discrete classes.

IoT data streams are a type of time series information. These time series form data types that can be viewed as one-dimensional or two-dimensional grids. Convolutional neural networks (CNNs) do extremely well at classifying grids. For instance, CNNs are known to excel at image classification, where each image is processed as a two-dimensional grid of pixel values. By treating IoT data streams as time series grids representing packet data, WIMS takes advantage of existing computer vision and image processing ML capabilities.

The fundamental ML approach used is to turn packet captures from a particular IoT protocol into a set of images which are then analyzed by a CNN. WIMS uses a Gramian angular field (GAF), described below, to convert the time series into a two-dimensional grid which can be treated as an image.[a] The advantage of this approach is that it preserves temporal dependencies among packet events, provided the events are properly encoded. The CNN is trained to recognize anomalous versus normal traffic in a process akin to image classification. Once a stable model is developed, that model is used for TL to a new protocol.

After suitably accurate results for correct classification (i.e., normal versus anomalous) are achieved with this method, the next goal is to transfer the well-trained CNN for anomaly detection on other protocols. When building this capability, we first sought to demonstrate, with nearly 100 percent accuracy, the ability to detect a Bluetooth response flooding attack using a CNN image-classification algorithm, and then use TL to substantially reduce the training time for anomaly detection of a response to a flooding attack in an 802.15.4 network such as ZigBee.

As traffic is collected from the IoT traffic sniffers and forwarded to the central server, the data is stored in a MySQL database in the form of packet capture (PCAP) files. The first steps of the ML processing pipeline are to turn each individual PCAP into an image which can be processed by a CNN. This pipeline is shown below and is explained in detail in the following section.

Image production

The first step in transforming packet captures into images is to extract time series data from the database from each feature of interest. Example features of interest are packet interarrival time, packet size, and protocol message type. Specifically, each value in the time series comes from one packet. For the database used in the WIMS, this corresponds to one row. The time series is split into fixed size chunks corresponding to a certain duration of elapsed real time. The current default is one second. This produces a subseries for each feature.

When processing data extracted from a database (e.g., training, validation, testing) or a dictionary (i.e., real-time prediction), the values of each feature must be parsed in an appropriate fashion. The parser

a. It should be noted that the images produced by WIMS cannot be readily interpreted by human analysts.

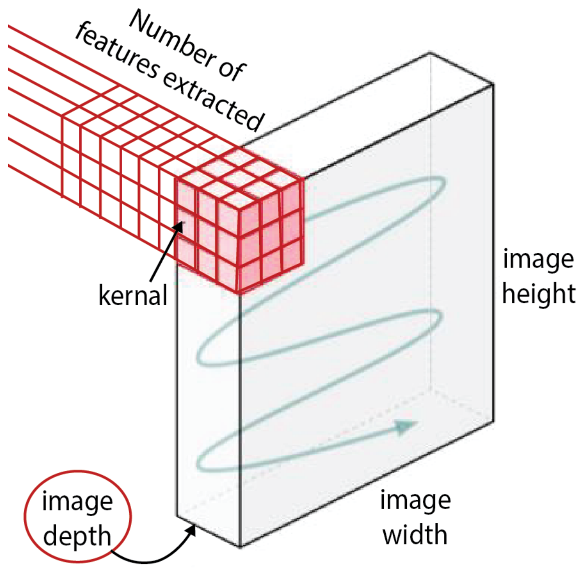


FIGURE 3. In building this Gramian angular field (GAF) representation, the value of each feature in the time series is converted into a polar coordinate system, and a two-dimensional grid (i.e., the image) is produced where each entry is the trigonometric sum between each pair of points.

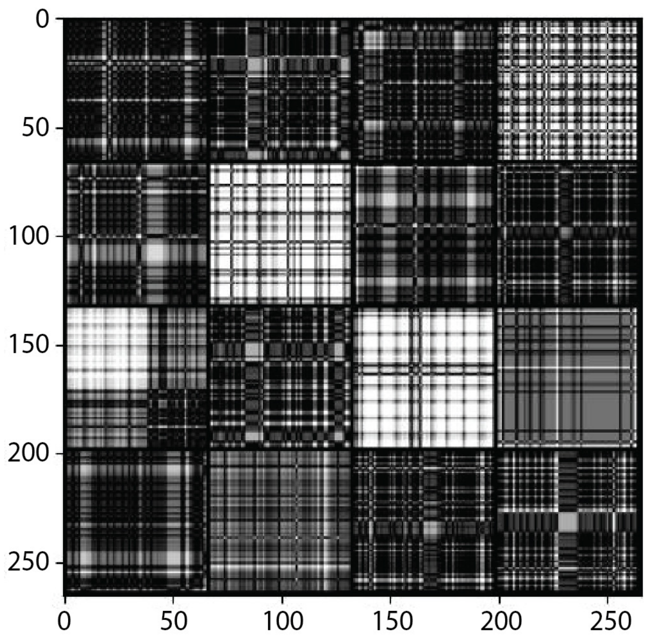


FIGURE 4. This sample Bluetooth GAF representation contains 16 separate subsequences from a Bluetooth flooding attack. The initial input was packet interarrival time.

replaces any NULL values from the database with a chosen pad value for the feature, then converts the data extracted from the database with values of the appropriate data type. The rules for pad values are as follows:

1. It cannot be a value which could legitimately arise from a true observation; for example, don't pick a positive value for packet length, for example.
2. It makes sense in the context of its respective feature.
3. It can be represented by the chosen data type for this feature; for instance, don't pick a negative pad value when using an unsigned integer like np.uint32.

At this point a Gramian angular field is produced for each subseries. This is done by rescaling all values to fall within the interval [0,1]. These values are then put into a polar coordinate system by encoding the value as the angular cosine and the time stamp as the radius. A square matrix is then used where each entry is the trigonometric sum between each pair of points. This enables the identification of temporal correlations between different time intervals. A square Gramian matrix is produced which is then transformed into an red-green-blue (RGB) image.

A key part of the data processing pipeline is to use stackable GAF representations, each corresponding to the same time slice for each feature to form a tensor of arbitrary dimensionality. This allows for an N-channel image that allows a network traffic representation of arbitrary complexity, providing maximum flexibility for feature selection. This is shown in the figure 3.

Figure 4 shows an example image, consisting of 16 samples, from a training dataset used to identify a Bluetooth flooding attack. The image shown here is for interarrival time. The advantage of this approach is that we can make full use of existing ML techniques for image classification for anomaly detection. We train our CNN to function as a one-class classifier. It learns to recognize "normal" images that correspond to non-anomalous traffic and to reject non-normal images. These rejected images correspond to anomalous traffic.

Convolutional neural network and transfer learning

The WIMS treats anomaly detection as a binary classification problem. The approach is to produce

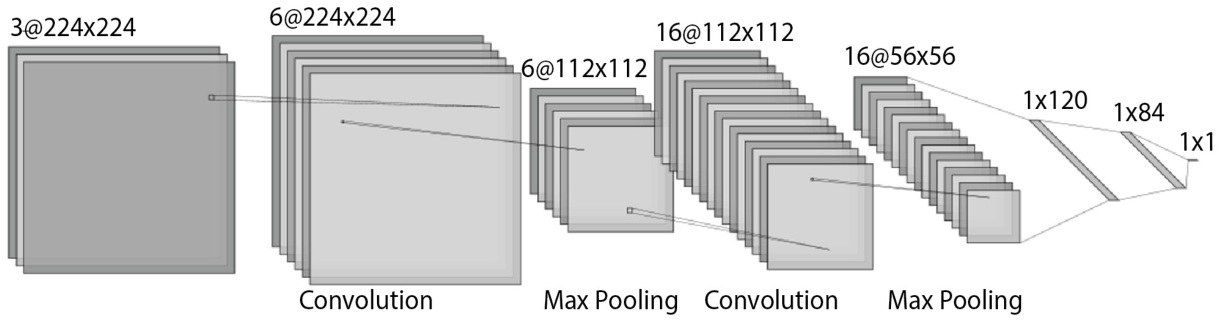


FIGURE 5. In this WIMS CNN, the CNN uses two convolutional layers with a relatively low number of output channels, three fully connected dense layers with a relatively low number of parameters, max polling layers, and ReLU activation.

images using the technique just described of both normal behaving traffic and anomalous traffic. The CNN is then trained to predict whether a newly presented sample falls in the normal class of images it has been trained on. All samples not classified as normal are considered anomalous and can generate an alert.

The WIMS uses a relatively small CNN. It is shown in figure 5. It uses two convolutional layers with a relatively low number of output channels, three fully connected dense layers with a relatively low number of parameters, max polling layers, and rectified linear (ReLU) activation. Taken together there are approximately six million trainable parameters. Considerable effort was put into fine tuning the CNN values to produce highly accurate results.

Let TP represent the true positive rate, TF represent the true negative rate, FP represent the false positive rate, and FN represent the false negative rate. We evaluated the performance of our approach using F1 and the Matthews correlation coefficient (MCC) metrics. An F1 score is the harmonic mean of precision and recall. The F1 score can be written as

$$F1 = \frac{2\text{*TP}}{(2 * \text{TP}) + \text{FP} + \text{FN}}.$$

For binary classifiers, MCC shows the correlation between predictions and actual observations. A value of 1.0 shows perfect prediction, 0 is essentially random, while -1.0 represents completely incorrect predictions. The MCC can be calculated directly from the 2x2 confusion matrix by calculating the following:

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}.$$

In our work, once routine F1 and MCC scores of 0.98 or above were obtained, it was possible to start the process of TL. The goal was to take a CNN trained for Bluetooth anomaly detection and transfer learned values to a new CNN, in this case anomaly detection for an 802.15.4 network.

The primary method used entails "freezing" the first layers of the pre-trained network and using these layers as a "feature extractor," but plugging in and training entirely new, "unfrozen," randomly initialized weight layers to replace the last layers of the network, which will perform classification based on those extracted features. This method assumes that the important features to be extracted from an input to perform an accurate classification of that input are the same for both the base and target tasks, but that the correct classification based on those extracted features may differ between the base and target tasks.

We implemented the following eight different variants:

1. A special placeholder value that denotes training from scratch (no TL, random weight initialization, all layers unfrozen)
2. Unfreeze only the last fully connected layer
3. Unfreeze all (3) fully connected layers
4. Unfreeze the later convolutional layers (the second one) and all fully connected layers
5. Unfreeze all (2) layers (convolutional and fully connected) in the entire network
6. Scenario 1, but reset the unfrozen layers
7. Scenario 2, but reset the unfrozen layers
8. Scenario 3, but reset the unfrozen layers

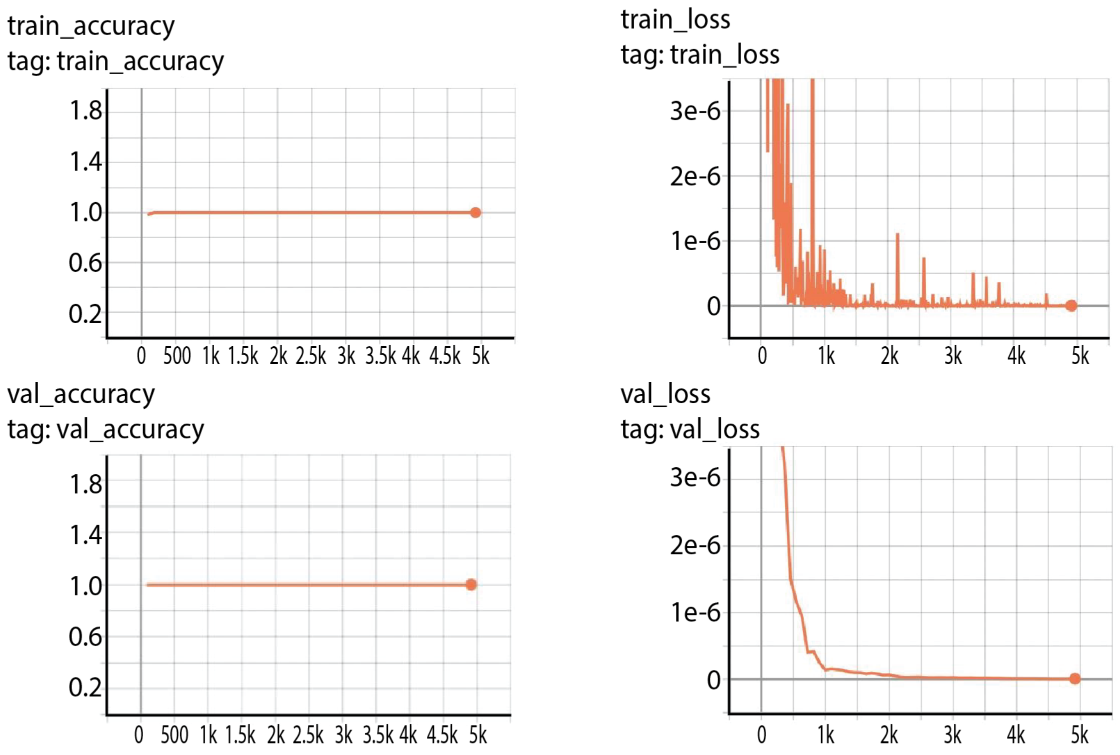


FIGURE 6. For these Bluetooth anomaly detection model results, the Braktooth anomaly [8] was used to create a flooding attack on a Bluetooth speaker. Graphs are depicted for the results of training accuracy, value accuracy, training loss, and value loss.

The WIMS software includes tunable per-layer learning rates so that unfrozen layers can learn at different rates depending upon whether their weights have been reinitialized prior to TL. For this application, we want larger learning rates for training from scratch and smaller learning rates for transfer. This allows the system to learn quickly from scratch and fine-tune existing knowledge carefully during transfer. The system also supports different L2 regularization penalties used by the adaptive moment estimation (ADAM) optimizer [7]. ADAM is a computationally efficient gradient-based optimization algorithm commonly used to fine-tune weights in neural networks. WIMS also uses different learning rate scheduling policies based on the layer type—“base” meaning unfrozen and reset (or learning from scratch in scenario 1), or “transfer” meaning unfrozen but not reset.

The system uses k-fold cross validation when training models to generate checkpoints, and when testing those models or using them together to vote and make a collective prediction. The k-fold cross validation randomly splits the training dataset into 10 equally sized subsets. Ten networks are then trained where each network uses a different 1/10 of the data for validation and the other 9/10 for training. When testing and making predictions, the 10 model checkpoints work together to vote and determine if a given sample is anomalous or non-anomalous. They each get an equal vote, and the average of their confidences that a given sample is anomalous is used to make a collective prediction, with a threshold of 0.5 (>0.5 means anomalous, <= 0.5 means non-anomalous). This was experimentally determined to be critical for achieving accurate results.

When generating models using TL, several questions arise when training models to detect a given anomaly: Which protocol should we start with (i.e., train from scratch on)? Which existing checkpoints should we use to train new models using TL? In which order should we train on protocols—can we use a previous protocol’s TL checkpoint as a base for further TL on a new protocol?

To address these challenges, we developed a tree-based algorithm for exploring the optimal scenario. Each tree has a single protocol at its root; we train from scratch on that protocol. The tree is traverse

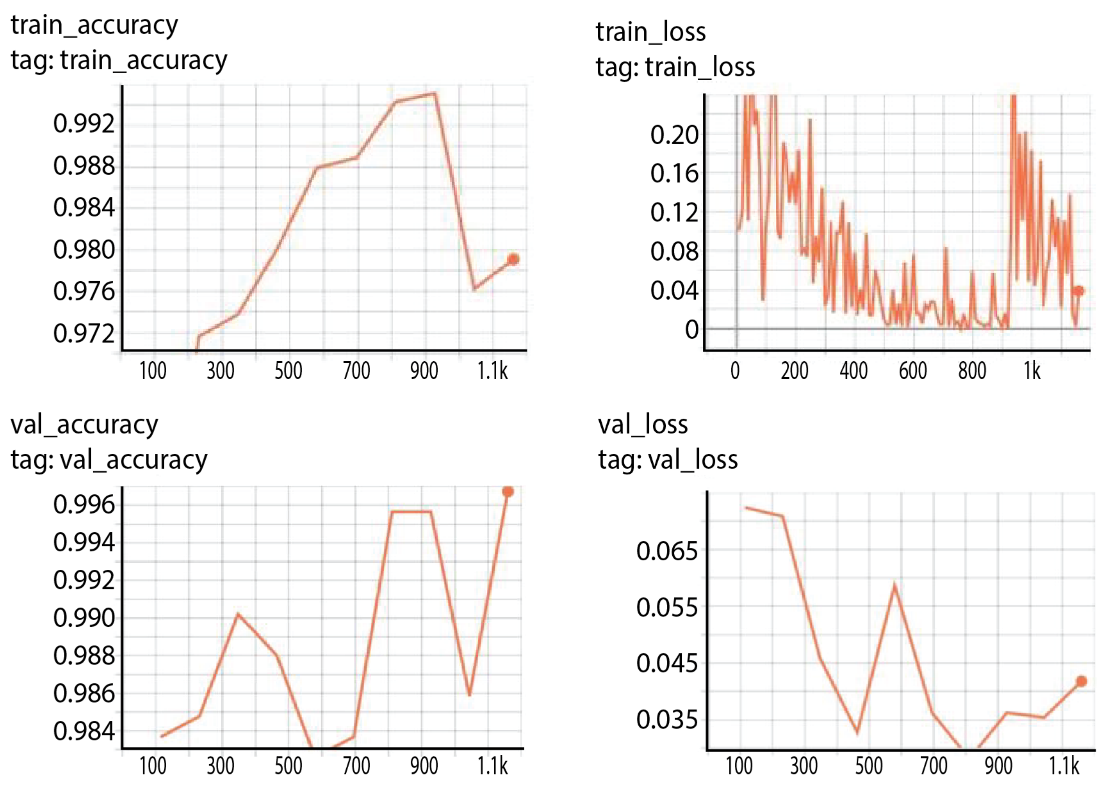


FIGURE 7. In our results, TL was used to apply Bluetooth flooding anomaly models to an 802.15.4 protocol. Graphs are depicted for the results of training accuracy, value accuracy, training loss, and value loss.

TL from a parent node to generate its child node(s). All methods of TL are compared, combinations of frozen, unfrozen, and "reset" layers. At that point, the algorithm selects the best F1 score to occupy a tree node going forward before moving to the node's children. Each tree contains all possible TL checkpoints starting from a base protocol without any cycles—an ancestor in the tree is never revisited as a child. Multiple trees can be produced to see which protocol makes for the best base.

Evaluation

We tested the above classification and prediction algorithms and then used the TL techniques described above and experimented under a variety of conditions. For the Bluetooth protocol, we used a Bluetooth speaker and the Braktooth anomaly [8]. This causes a response-flooding result—the speaker is disconnected from Bluetooth and then reboots. The three features selected for the stackable CNN were packet interarrival time, packet size, and packet type. The results are shown in figure 6. After fine tuning training parameters, we were able to achieve 1.0 accuracy. The results also show training and testing loss.

We then performed TL to the 802.15.4 protocol. We ran tests using a wireless personal area network (WPAN) consisting of an 802.15.4 hub and a thermostat, and we implemented a flooding response. Using combinations of the eight TL scenarios, we were interested in the two following figures of merit: 1) can we achieve high levels of accuracy, and 2) can we reduce the training time?

Figure 7 shows the results for one of these scenarios. As can be seen, after only nine training epochs, we obtained prediction accuracy of 0.97.

By observing training and target loss it is also possible to observe periods of model instability during gradient update epochs. To better understand this issue, we developed and implemented numerical function fitting software for IoT data for 12 different probability distributions, including log-gamma, parteo, and exponential. They revealed fundamental structural differences between Bluetooth and WPAN for the features we selected.

From an algorithmic perspective, the project achieved its objectives of demonstrating TL for multiple wireless protocols. Enabling factors that contributed to the success of the TL include the development of stackable GAF image production techniques, a deeper understanding of how to fine tune CNN parameters for prediction, the use of the eight different scenarios for TL, and the tree-based approach for TL protocol selection. Future work could explore different types of IoT-based attacks such as man-in-the-middle and the use of adversarial ML techniques to enhance our understanding of poorly understood protocols. We also expect that the probability distribution fitting tool will be extremely useful in understanding new datasets.

Conclusion

This paper described WIMS, a distributed system used to monitor deployed IoT networks at the enterprise level. It uses COTS IoT protocol analyzers/sniffers and SDRs to collect wireless network traffic. On a device-by-device basis, packet captures are turned into images. A CNN is trained to recognize normal versus anomalous images. Upon detection of an anomalous image, alerts are generated.

One of the notable aspects of WIMS is the use of TL. This approach reduces the time to train a CNN on new protocols and new attack vectors. Our current plan is to continue to classify new IoT protocols as they come online, as well as new types of IoT cyber threats.

References

[1] Link Labs. "The complete list of wireless IoT network protocols." 2016 Feb 8. Available at: https://www.link-labs.com/blog/complete-list-iot-network-protocols.

[2] 3GPP. "5G system overview." 2022 Aug 8. Available at: https://www.3gpp.org/technologies/5g-system-overview.

[3] Amazon.com Inc. "What is Amazon Sidewalk." *Developer Guide: AWS IoT Core.* [Accessed online 2023.] Available at: https://docs.aws.amazon.com/iot/latest/developerguide/amazon-sidewalk-overview.html.

[4] Spanalytics, LLC. "PANalyzer." [Accessed online 2023.] Available at: https://spanalytics.com/product/panalyzr/.

[5] Ellisys. "Ellisys Bluetooth Vanguard advanced all-in-one Bluetooth analysis system." [Accessed online 2023.] Available at: https://www.ellisys.com/products/bv1/.

[6] Teledyne Lecroy. "Frontline X500 wireless procotol analyzer." 2023. Available at: https://cdn.teledynelecroy.com/files/pdf/frontline-x500-datasheet.pdf.

[7] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." ArXiv preprint; arXiv:1412.6980 (2014). Available at: https://arxiv.org/abs/1412.6980.

[8] Garbelini ME, Chattopadhyay S, Bedi V, Sun S, Kurniawan E. "BRAKTOOTH: Causing Havoc on Bluetooth Link Manager." Asset Research Group. Available at: https://asset-group.github.io/disclosures/braktooth/.