Fingerprinting Edge and Cloud Services in IoT

DongInn Kim
Indiana University Bloomington
Email: dikim@indiana.edu

Vafa Andalibi
Indiana University Bloomington
Email: vafandal@iu.edu

L. Jean Camp
Indiana University Bloomington
Email: ljcamp@indiana.edu

***Abstract*—Today, Internet of Things (IoT) devices, web browsers, phones, and even cars may be fingerprinted for tracking, and their connections routed through or to malicious entities. When IoT devices interact with a remote service, the integrity or authentication of that service is not guaranteed. IoT and other edge devices could be subject to man-in-the-middle (MiTM) attacks, with IoT devices attempting to connect to remote services. It is also straight-forward to use phishing or pharming to convince a user to accept a connection to a potentially malicious unfamiliar device. These risks could be mitigated by leveraging information on the edge of the network about the path to and destination of a connection. In this work we sample packets, then use packet analysis and local history to identify risky or suspicious connections. In contrast to other machine learning and big data approaches, the use of local data enables risk detection without loss of privacy.**

I. Introduction

Security is a challenge in the IoT, in home computing, and for small organizations that rely on IoT and cyber-physical systems. Due in part to limited processing capacity IoT often depends upon remote third parties for security. Recent man-in-the-middle (MiTM) vulnerabilities illustrated both attacks at nation-state scale [1] and quickly, with a MiTM in just 15 minutes [2]. These attacks illustrate the threats to the security and privacy of the endpoint relying on the cloud. Defenses against these attacks often require concentration of data for machine learning, with levels of data concentration and data exfiltration that creates its own risk. [3], [4], [5] Our goal is to secure the connection from an IoT device to identify and mitigate MiTM attacks without sacrificing users' privacy has motivated our research.

We propose to flip the current fingerprinting paradigm so that devices on the edge collaboratively fingerprint the remote services, identifying deviations from normal operations. We illustrate that this is possible by constructing a local agent built on a raspberry pi called Block-Pi. A second way in which this is innovative is that Block-Pi provides privacy by design. The privacy of the date entrusted to the system is guaranteed by localization, where the filtering and analysis is implemented between the router and the network. Rather than sending all personal data out on to the network Block-Pi polls centralized remote sources of information for blacklists and model-building, integrating local information to create a distinct model for each installation. This unique local model increases the difficulty of blackbox attacks, as an attack on one network may not function on another. It also allows for highly customized detection, e.g., homograph detection for small home businesses or individual workplaces for the work at home employee.

The underlying structure assumes that single point of access is associated with a set of IoT devices and that those devices connect to a set of remote services. The model of our proof of concept includes cloud and Edge service. The proposed agent described here identifies potentially hazardous changes in remote services after initial connection, new unfamiliar connections, and blocks known malicious connections.

We offer a proof of concept of a local agent implemented as middleware that defends local devices by identifying masquerade and malicious connections using individualized knowledge and history. Essentially, the goal is to limit the scope of global trust to that which is locally known. The local agent will not only route the traffic between the IoT devices and Edge or cloud services but also fingerprint the access to the remote services. Our specific example case is not only protecting IoT devices but also assuring the correct connection to their edge services. We believe that this is generalizable to cyber-physical systems, many of which are legacy and have not internal protection. We discuss the choice of devices further in limitations as part of discussion.

This paper targets those IoT devices which are less resource-constrained and have reliable local wireless network connectivity. The devices in our experiments are Google Home, Ring Doorbell, Belkin Power Plug, and Kangaroo Motion Sensor. These devices will be capable of providing consistent machine learning data for the local middleware agent. All of these but the Kangaroo Motion Sensor have a direct power connection, it is battery operated.

We describe our motivation in Section II. We then provide details of the proposal to support the identified contribution in Section III. Two experiments in early stage are performed to show the feasibility of the proposed system which are presented in Section IV. Finally, this is followed by the conclusions and discussion in Section VI and V.

II. Motivation

Consider an example of hijacking a Belkin Smart Power-plug. The normal services that the device provides are the ability to turn on and off the power and to send and receive the status of the power to its Edge servers in Amazon Web Services (AWS). The device is configured initially by opening up its own access point for the WeMo smartphone app in non-encrypted mode. The WeMo smartphone app allows any smartphone to connect to this access point and configure the

978-1-7281-9459-2/20/$31.00 ©2020 IEEE
DOI 10.1109/SADFE51007.2020.00011

home wireless network settings into the device. Once the initial configuration is completed, the device turns off its the access point and continues to use the home wireless network to access its Edge services. The smartphone with the WeMo app configured can control the device by communicating with the Edge services via the Internet.

The services that the Belkin Smart Powerplug connects to should be highly limited. Our goal is to characterize not only the endpoint but the steps to the connection to these limited acceptable endpoints. Our fingerprinting proof of concept currently uses Transport Layer Security (TLS) certificates (if available), Domain Name System (DNS) cache validation, history of the access, remote IP Autonomous System (AS), and reference to listing of known malicious entities (e.g., PhishTank, SpamHaus, and other blacklists). As the IoT devices access services, all the features for identifying the service are collected in the local middleware agent. The local agent will evaluate the connection request using a cross-layer comprehensive packet analysis. After an initial packet analysis is completed, the agent will have two sources of updates and information: the local home network and the global Internet. The local agent will use the information from these two sources to create a customized model to filter suspicious connections between the IoT devices and the requested sites or services.

Block-Pi builds on previous work using machine learning; for example, the work the Maglaras et al. [6] used One Class Support Vector Machine (OCSVM) as part of a Distributed Intrusion Detection System (DIDS) to detect MiTM attacks. However, this required coordinated deployment on three layers: traffic monitoring from end-user's network, Edge network, and on the cloud [7]. This requirement is very difficult to fulfill since the user does not have access to the manufacturer's network and (hopefully for privacy and security reasons) vice versa.

In addition, MiTM mitigation on Edge networks was identified as a potential strength of adoption of software-defined networking (SDN) by Li et al.[8]. The connections in SDN have flows instead of routers, and the same fingerprinting [9] approaches we discuss here could be used to evaluate the flow to the remote service. To the knowledge of authors, fingerprinting the Edge services has not been used before for preventing MiTM attacks.

We describe the possible scenarios of an TLS MiTM attack. Note that the scenario we are providing here was addressed in fine detail in previous threat analyses [10][11]. The attack is also illustrated in Figure 1.

In this scenario the attacker hijacks the transport packets between the Belkin Smart Powerplug and the Edge service in the same network or in the Internet via the WeMo smartphone app modification, injection of the forged certificate to the device firmware, and replicated Edge services in AWS. The attacker then relays the TLS handshake messages from the Belkin Smart Plug to its Edge server and vice versa by impersonating it to both sides so that the injected communication will be delivered to both sides. Finally, the attacker completes the TLS handshakes with both sides by receiving the shared symmetric key from the Smart Plug device and then sending a new symmetric key to the Edge service.

Once the MiTM attack is established the attacker can interject data, threatening the security and privacy of every service provided by the Edge. The attacker can damage the appliances by switching on and off constantly, intimidate the people whose house is compromised, or use its connectivity to direct it to DDoS. The risk is exacerbated if the home security system is powered with the compromised Smart Plug. Since the firmware of Belkin Smart Plug is shipped with the open source program, OpenWrt[1], it is possible for an attacker to inject the invalid CA certificate and then bypass the certificate validation process by modifying the firmware. The attacker can reverse engineer the WeMo smartphone app by hacking its apk file [12][13][14][15].

This paper proposes the idea of having a local agent that monitors all the incoming and outgoing traffics and provides access to recognized Edge services and issues warnings for unfamiliar services. We deployed this idea to Raspberry Pi-4 and we call it "Block-Pi" as shown at Fig 2. The assumption is that there is a vector for risk mitigation or over-ride; and our design includes an integrated user interaction for the device with the corresponding app (e.g., Google Home, Ring, WeMo, Kangaroo Motion Sensor) [2].

Our research question is to see if the collective operations of an IoT device provide adequate information for fingerprinting the requests from/to the IoT device in the home. The critical challenge of this idea is to determine whether or not the access to the Edge services is safe by analyzing multiple traffic streams simultaneously.

The innovations of this proposal are 1) the integration of devices into a single home threat model, 2) the use of multiple layers to implement a single trust measure for the connections made by those devices, and 3) the inclusion of out-of-band information such as the geography of various Autonomous Systems (AS). One component of the current implementation is prioritizing local information over global data, hence protecting the users' privacy. Using this data, we propose a packet analysis approach to determine the authenticity and integrity of the connection. The Block-Pi and Block-Pi Server's code and related data are available upon request.

III. IMPLEMENTATION

In this section, we provide details about the different components of our proposed system, Block-Pi, as shown at Fig 2, including the factors that we have considered for designing the systems, details of the implementation, as well as the system's workflow.

A. Design

One of the main assumptions considered in designing the Block-Pi, is that the local network is composed of the end

[1]OpenWrt, https://openwrt.org/

[2]While the human subjects interaction is subject to current research in the larger research group, it is beyond the scope of this work.

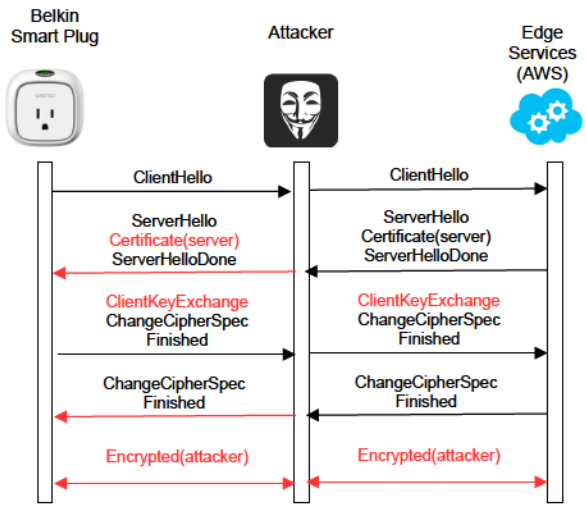

Fig. 1. TLS MiTM attack scenarios between the Belkin Smart Plug and its Edge service provider in AWS [10]. The red arrow and text represent the steps taken by the attacker.

devices that request services from Edge and cloud service providers. In this context, we consider the requesting devices as a collection of IoT devices under one administrative domain; for example, a set of devices in a home connected to one hub or a number of devices on a single LAN in a small business.

Block-Pi collects the following data about the Edge and cloud service providers by providing the routing services for the IoT devices.

- TLS Certificates (trusted / untrusted / partially trusted)
- DNS Cache server (changed / unchanged)
- Resolved IP addresses of the Edge service providers (changed / unchanged)
- AS information (changed / unchanged)
- WHOIS information (match / mismatch)
- PhishTank data (benign / phish)

The overall architecture and concept of using local information will be familiar with those who have evaluated $c_n{-}grams$. [16] Block-Pi analyzes connection data, while $c_n{-}grams$ examines the content itself. Also, Block-Pi also polls external data sources.

We will review the implementation details of each category in the next section.

B. Implementation

Block-Pi is implemented as a Java application so that it can be deployed on any platform that can support Java Virtual Machine (JVM). A daemon service runs in Block-Pi.

The TLS certificates compiled by device are analyzed using an iterative machine learning process to classify the validity of the certificate building on previous work by Zheng et al. in detecting rogue and phishing certificates [17]. The evaluation of the certificate uses 42 fields of the X.509 public key certificate as the machine-learning features. In addition, the system regularly pinged PhishTank and revocation lists.

Fig. 2. Block-Pi uses Raspberry Pi 4 to setup the external network with its Ethernet port and to host the access point with its wireless interface

The additional information obtained from revocation and phishing was used by Block-Pi. The optimized model that we chose for the TLS certificate classification is Random Forest, which provided 95% precision and a recall rate above 93% in the initial proof of concept [17].

All the devices in the home network share a DNS cache which is the first source of data not only for Berkeley Internet Name Domain (BIND) but also for site analysis. This local shared cache is used to determine if there is any unusual change in the local DNS server. If the IoT devices access a different DNS server, it would be identified as a suspicious activity. Block-Pi will notice this change and either notify the user, block the connection, or increase the weight of this feature in the future machine learning model, decision tree.

Block-Pi also tracks the IP addresses as well as the traceroute information, i.e. IP address of the routers, for each provider. If there is any anomaly, it can be a sign of a MiTM attack. The current packet analysis examines the IP address and corresponding range. Note that the benign changes in the traceroute information could be interpreted as a sign of MiTM attack. In order to minimize such false positive cases, we use the Autonomous System Number (ASN) routes and we will show how ASN routes are consistent for the normal activities. Also, such changes are not as significant as changes in the DNS server. Changes on provider's IP address affects only the related IoT devices, however, changes on DNS cache server will affect all the IoT devices in the same network, including Block-Pi.

WHOIS data is a verification factor to see if the Edge service provider has consistent data in WHOIS. Of course, the recent changes in WHOIS due to General Data Protection Regulation (GDPR) have limited the efficacy of this method, however, network information such as DNS resolver remains available. The normal re-registration of the domain does not update many fields in WHOIS. The Registrar information, including WHOIS Server, URL, Registration Expiration Date, IANA ID, Abuse Contact Email, and Abuse Contact Address,

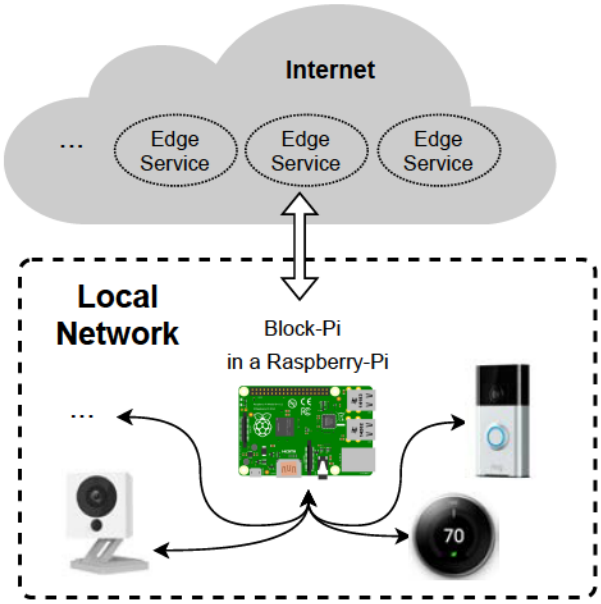

Fig. 3. Block-Pi architecture to control a small set of IoT devices

may change during the re-registration. Block-Pi keeps the history of the changes. Large changes in WHOIS information may be a sign of malicious re-registration. This information is also used during the initial connection, e.g. if the resolver is unknown or all associated information is from an AS that has never been contacted by any of the devices.

As for the PhishTank information, Block-Pi updates the local PhishTank data by synchronizing with the central PhishTank database every hour. Since the connection to PhishTank is established with the ASN routes verification, the TLS certificate, and the connection token, the data from PhishTank is securely synced to the local PhishTank database. The PhishTank data can be a reliable determinant of malicious activity if the resolved IP address of the service provider is in the PhishTank database.

C. Workflow

Block-Pi has two primary stages for fingerprinting the Edge services as described at Fig 4. Initially Block-Pi runs the packet analysis by collecting the network packets transferred between IoT devices and their Edge services. If there is any anomaly from the packet analysis step, the anomaly information is handed over to the next step, ML which is part of our future work. Otherwise, there is no need to go through the ML model.

The proposed system works based on the following essential steps: identification of a new IoT device in the network, instantiating the initial model, collecting the communication data of the device, performing the packet analysis from the collected communication data, training the model, and, periodic retraining. The initial model needs to be prepared with special care because Block-Pi does not have prior data for the

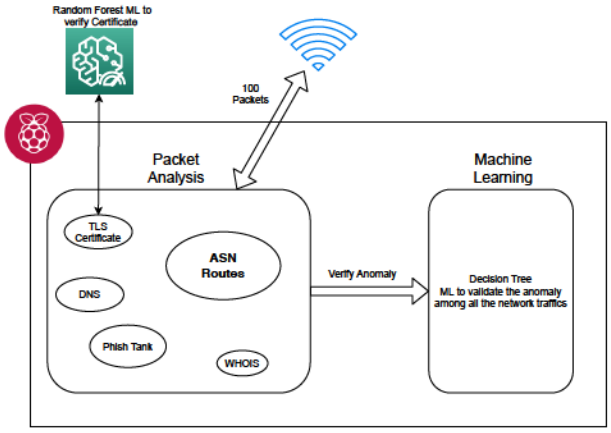

Fig. 4. Block-Pi workflow

model of the new device yet. We explain each of these steps in more details:

First a Universal Unique Identifier (UUID) is generated for each device. Block-Pi then has an identifier to associate with each connection as it compiles basic network data (internal IP address, MAC address, and destination IP address) of each connection for each device.

Second Block-Pi, functioning as a router and gateway, will then run with a general model of the network based on the available information about the local network. Since Block-Pi sets up its own local DNS server to avoid the outside DNS poisoning attacks, it will keep another local history of the resolved IP address. If the connection is encrypted the agent will analyze the validity of each certificate, classifying it as trusted, untrusted, or partially trusted. Block-Pi will save this classifying result.

Third collected communication data are applied to the packet analysis and Block-Pi finds any anomalies in the current communication by looking up the existing packet data. The ASN routes data is one of the main features to determine the anomaly from the traffic. When there is no prior ASN history of the IoT device in Block-Pi, it will refer to Block-Pi Server to retrieve the most commonly used ASN routes in the neighboring Block-Pis. If there is anomaly during the packet analysis, the ML customized data are transferred to the next step. Otherwise, Block-Pi continuously runs the packet analysis with the collected data.

Fourth over time the device continues to connect, and the machine learning model will be trained with the collected data. The preliminary training is grounded in global information, augmented by centralized information sources. Block-Pi will send a notification when there is an untrusted result, e.g., the Edge service provider is in PhishTank or when any evaluation result (the probability of malicious activity) is greater than a given threshold (50%). Otherwise, the agent forwards the request to the Edge service provider.

Newly collected data Block-Pi is combined with the previous training data, and the model will be updated with the training data every 24 hours. Standard 10-fold cross-validation

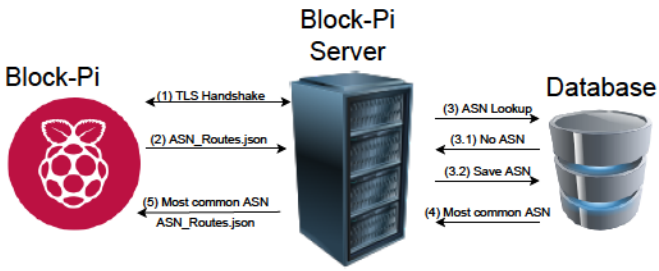

Fig. 5. Workflow between Block-Pi and Block-Pi Server

will verify the training results.

Ground truth is considered to come from a few external trusted sources (e.g., PhishTank) or explicit home owner interaction[3] After a model is newly trained, it is locally tested (recall 10-fold cross validation) using the newly collected data, yet the previous models are retained. If a local record of a trusted site is found, for example, to show up in PhishTank, the model resets to the last working version. *Absent model error*, the newly trained model will replace the previous model. The data compilation will continue, and Block-Pi will continue to detect malicious activity with the new model.

The initial packet analysis data in Block-Pi is created based on the IoT devices that we have conducted experiment on so far (e.g., Google Home, Ring Doorbell, Belkin Smart Plug, and Kangaroo Motion Sensor). However, when a new IoT device is added to the Block-Pi network for the first time, there is no way to identify the validity of the traffic due to lack of adequate information. We therefore setup a Block-Pi Server which collects only the ASN routes that each Block-Pi logs when IoT devices in their network attempt to reach their Edge services. Note that this approach is still privacy-friendly since the collected ASN route data do not include any sensitive information, such as the information about the IoT devices that Block-Pi monitors, the association of IoT devices with their services, as well as the type of Edge service that is running in the destination IP address. The collected ASN routes in Block-Pi consist of three fields: (a) IP address of Block-Pi, (b) IP address of the Edge service, and (c) ASN routes to the IP address of the Edge service.

When Block-Pi sends the ASN routes to Block-Pi Server, it only sends (b) and (c) because Block-Pi Server finds the public IP address of the Block-Pi through the TCP connection as shown in Listing 1 and Fig 5.

When a new IoT device joins a network, the Block-Pi in that network will communicate with the nearest Block-Pi Server over HTTPS. The Block-Pi will send to the Block-Pi Server the new ASN route resulted from the initiated connection of the new device. The Block-Pi Server will reply with the most common ASN route of that destination, which was collected only from the neighboring Block-Pis.

In addition, when Block-Pi detects an anomaly when comparing an ASN route with local history, it will send the new ASN route to the Block-Pi Server to see if the neighboring Block-Pis have experienced the same ASN route. Otherwise, it can be a sign of the malicious activity and it is moved to the next ML approach to verify its validity of the traffic.

Block-Pi Server will always looks for the most common ASN routes that has a source IP range that matches the range (rather than the exact IP address) of the Block-Pi, and a destination IP address that matches the exact IP address of the target edge service.

In case Block-Pi Server cannot find any ASN routes from the neighboring Block-Pis, it should return the exact ASN route between the destined edge service IP and the exact same source IP of that exact same Block-Pi.

If no close match was found either, the Block-Pi Server starts to keep the ASN route entries with the new IP addresses and returns the original JSON file to Block-Pi as described at the Fig 6. Note that the communication between Block-Pi and Block-Pi Server is not kept alive all the times. Block-Pi needs to connect to Block-Pi Server to receive the most commonly used ASN routes in the neighboring Block-Pis when a new IoT device is added to the Block-Pi network.

Security of the Block-Pi Server, including the security of the communication between the Block-Pi Server and other Block-Pis, is of significant importance because a compromised Block-Pi Server can be used for distributing malicious ASN routes as well as falsifying the fingerprint results of the IoT devices connected to the Block-Pis. Hence, the HTTPS connection between Block-Pi and Block-Pi Server is setup with the pre-configured TLS certificate to secure confidentiality and integrity of the traffic.

D. Classification

Our Block-Pi currently uses a decision tree[19] for the classification of the fingerprinting because decision tree provides competitive accuracy in classification and it is very efficient. Most importantly, we believe that decision tree fits our application for two reasons: first, it can easily distinguish and prune unwanted results, and second, it can calculate the expectation value of our experiment with a high *precision*, i.e. lower value of false positives, and *recall* value. Note that in our application false negative cases are those where Block-Pi reports the Edge service as benign while it is in fact malicious. Conversely, false positive are those where the agent reports a benign service as malicious. False positives are problematic because users will reject a system with false alerts. Conversely we may care more about false negative cases because higher false negative rate means more undetected attacks.

After each update, the model is improved by maximizing the *recall* score. If the new model achieves a worse *recall* value, it will be trained again by skipping the last day's training data. The new model should also consider maximizing the *precision* to reduce the false positive which may cause serious usability

[3]While the system interaction is a work in progress, one core design goal is to make security the easy default. Thus a home owner has to choose to make an explicit decision to take a risk after encountering risk communication and a inherent delay. Part of the design is to increase the delay just enough to require a move from fast, instinctive and emotional decision-making to slower, more deliberative, and more logical decision-making[18].

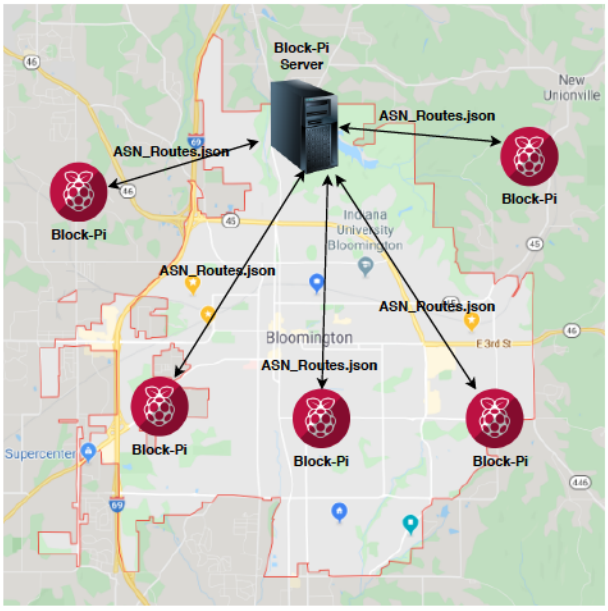

Fig. 6. Block-Pi and Block-Pi Server diagram at Bloomington, Indiana

```
{
  "35.175.5.157": {
    "156.56.83.2": "AS87",
    "134.68.3.130": "AS87",
    "149.165.254.233": "AS19782",
    "149.165.255.193": "AS19782",
    "149.165.255.210": "AS19782".
    "149.165.255.186": "AS19782",
    "64.57.28.105": "AS11537",
    "54.240.229.159": "AS165509"
  }
}
```

Listing 1. Example of the ASN routes that Block-Pi sends in JSON format

issue. The fundamental assumption of machine learning is that the distribution of training data is identical to the distribution of the future testing data. In reality, the practical machine learning data sometimes does not follow the assumption. However, the core assumption in this case is that a business or a household on one day is more similar to itself on subsequent days than it is to the entire Internet. The second assumption is that an IoT service should be similar to itself when previously contacted; than to any other IoT services at any other time. The training data will be collected to be sufficiently representative to the testing data set.

Finding the correct features is as much art as science. The features in Section III are based on previous work. We need to further find the best features for the classification. The best feature can be selected based on a function that generates the minimum impurity on each feature after partitioning. The key idea of a decision tree is to find the impurity function for the given data set.

IV. RESULT

We have conducted an early-stage packet analysis without applying any machine learning process in the following environment:

- A Linux server with two network interface cards (NIC)s is setup for Block-Pi. One NIC is for the Internet traffic and the other NIC is for the internal network. Refer to Fig 2. This server needs to host DNS and DHCP server for routing the networks of IoT devices in network address translation (NAT) environment.
- Google Home, voice-activated smart speaker: communicates with googol addresses (*.1e100.net).
- Ring Doorbell, a smart doorbell: communicates with AWS to query, store, and manage the recorded video files. There is a Ring Doorbell API for developers which can connect the ring devices without the ring app provided by ring.com. We try to use this API for the TLS MiTM attack.
- Two Belkin Smart Power Plugs: Sends the power usage data to AWS. We found that it uses the DNS connection at the boot time when a power plug is hooked to the power outlet. Belkin runs its infrastructure on Amazon EC2 Cloud and the power plugs connect *insight.lswf.net*, which is one of the AWS EC2 servers. They need to resolve the hostname with a normal DNS traffic. We can use the DNS traffic as a MiTM threat model. WEMO API is available to control the Belkin smart plugs. If an attacker makes a tcp connection with the smart plugs through the DNS poisoning, he can run the API commands to control the smart plugs.
- Two Kangaroo Motion Sensor: notifies a user with motion activities in front of the sensor. It is powered with two AA batteries and it lasts more than a year with the normal operation.
- Apple iMac and Raspberry Pi representing normal computing users.

Block-Pi runs a Java application which collects WiFi packets (IEEE 802.11 frame) from the IoT devices and eventually fingerprints the traffic as a daemon. The Java application is compiled with PCAP4J library to dissect the collected WiFi packets. Note that in our experiments, we assume that the IoT services are not compromised.

We have performed two experiments: when the four IoT devices are idle in the normal operation and when they are actively communicating with the remote services, which cause burst traffics and heavy congestion to the analysis. The first experiment ran 1000 iterations consecutively for 16.7 hours (1000 minutes) and each iteration collected 100 packets. Since the IoT devices were idle, there were no traffic bursts. Otherwise, the experiment time would be much shorter than 16.7 hours. The second experiment ran in the same condition as the first one but it took only about 1 hours. The private internal packets are uniquely identified with the similar range

of destination ports and server IP addresses as shown in table I.

The second experiment during the active operations of the IoT devices showed that the process of calculating the number of hops to corresponding server and traces of its routes is a little fluctuating because the process of analyzing the packets could not complete the current iteration before the start of the next iteration. This issue can be fixed by adjusting the number of collected packets to fit in the calculation time of the analysis. The first experiment showed that the unique private IP addresses are classified in a very short time (58 minutes) but the unique public IP addresses are setup rapidly at the beginning (up to 1 hour) and then gradually increasing after that. Especially Google Home device sends and receives its IoT services through many Google servers (*.1e100.net) and their IP addresses are continuously updated with new ones as shown in Figure 7. Even though the number of the unique public IP addresses increases, the IP addresses are in the same domain (*.1e100.net) and we can fingerprint the services with the same domain and the IoT service identification components (hops, bytes, RTT, and the trace of routers) as well as the fingerprint components described in Section III.

As mentioned prior, Block-Pi utilizes the ASN route data from Block-Pi Server whenever a new IoT device is added to the Block-Pi network. We found that the ASN routes of an IoT Block-Pi is remarkably consistent, making it a great feature to validate a given traffic. The ASN entries on an IoT device may access multiple Edge services with the exactly same service

TABLE I
LIST OF THE IP ADDRESSES FOR LOCAL IOT DEVICES AND REMOTE SERVICES

IoT Device	Dst PORTs	Server IP Addresses
10.42.0.79 (Google Home)	53 (DNS), 123 (NTP), 443 (HTTPS), 5228, 5353	8.8.8.8,10.42.0.1 74.125.132.188,224.0.0.251, 216.239.35.12,172.217.4.67, 172.217.212.94
10.42.0.111 (Ring Doorbell)	53 (DNS), 67 (BPS), 443 (HTTPS), 9998	34.237.36.230,8.8.8.8, 54.175.201.183,104.17.128.1, 18.206.166.168,34.198.218.23
10.42.0.87 (Power Plug-1)	53 (DNS), 67(BPS), 123 (NTP), 443 (HTTPS), 1900,3475, 3478,8443	10.42.0.1,192.5.41.41, 50.19.220.40,23.23.100.178, 54.225.170.208,239.255.255.250, 50.19.127.85,4.2.2.2, 23.21.86.133,54.164.188.30
10.42.0.216 (Power Plug-2)	53 (DNS), 67 (BPS),1900 123 (NTP), 443 (HTTPS)	10.42.0.1,23.23.100.178, 192.5.41.41,50.19.220.40, 239.255.255.250
10.42.0.16 (Kangaroo MSensor1)	53 (DNS), 443 (HTTPS)	10.42.0.1,34.207.44.211, 34.235.245.11,52.201.7.68, 35.171.130.145
10.42.0.17 (Kangaroo MSensor2)	53 (DNS), 443 (HTTPS)	10.42.0.1,34.207.44.211, 34.235.245.11,52.201.7.68, 52.2.187.86,54.164.145.55

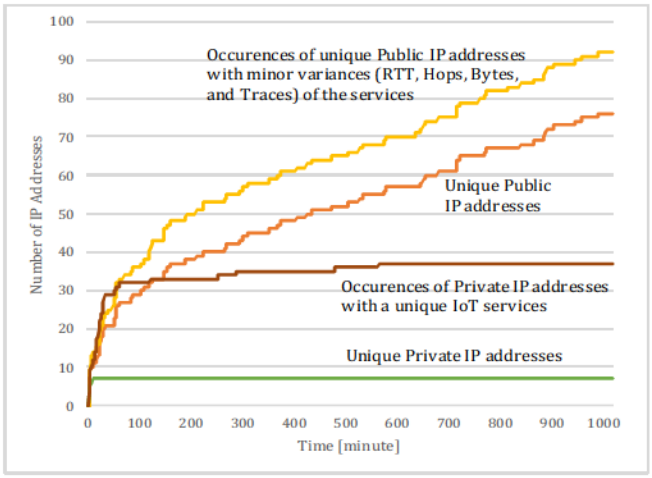

Fig. 7. Number of the IP addresses associated with the local IoT devices (Private IP) and remote services (Public IP).

port (e.g., 443 on Kangaroo Motion Sensor) because its Edge services are hosted on AWS. For example, the ASN routes for Kangaroo Motion Sensor reaching to all of its Edge services in AWS (i.e., the same port, 443 but different IP addresses) are identical in terms of the sequence of the ASN routes. The number repetitions of an ASN in the ASN route may be different because of the nature of the internet routing but the sequence of the ASNs in the route is extremely unlikely to change.

As displayed at Table II, the sequence of the ASNs does not change for 258 times accesses from Block-Pi to the Edge service provider (34.199.159.206) but the number of repetitions of an ASN is expected. In addition, we have investigated the ASN Routes from Kangaroo motion sensor in Indiana University (156.56.83.12) to its web Edge services hosted on AWS. The ASN Routes have been measured in the same location every 10 minutes 5305 times. As shown in Fig 8, the ASN routes, AS87 (Indiana), AS192782 (INDIANGIGAPOP), AS11537 (ABILENE), and AS16509 (AMAZON-02) do not change and the number of ASNs in the same route changes rarely on AS87, AS192782, and AS11537. It is possible that the destination AS (AS16509) has many discrepancy because the edge services in AWS are hosted in the cloud system and one of the hosts (e.g., 3.222.33.156, 3.224.38.120, 3.227.129.232, and 3.231.179.85) is not available at a certain time. Thus, the occurrence of AS16509 in the given ASN routes is not consistent but we can identify the network traffic by referring to the consistent ASN routes except for AS116509. The consistency of AS16509 occurrences is around 80% and the other ASNs have 99.9% consistency as calculated with the equation 1.

$$\frac{\text{Number of occurrences}}{\text{Total number of experiments}} \times 100 \qquad (1)$$

In a simple experiment, we deployed a proxy server between Block-Pi and the IoT devices and then performed an ARP poisoning so that all the traffics from the IoT devices go to

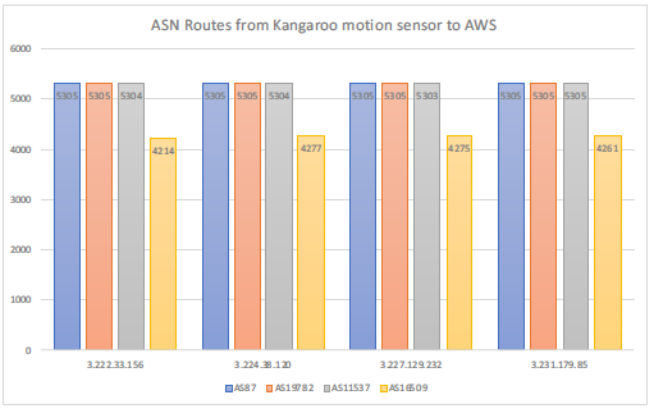

Fig. 8. Consistent ASN Routes from Kangaroo motion sensor (Indiana University, 156.56.83.12) to the same https services with the different Edge servers

TABLE II
ASN ROUTES FROM KANGAROO MOTION SENSOR TO THE EDGE SERVER (HTTPS) IN AWS.

	Indiana University		
Source IP	156.56.83.12	156.56.93.197	149.166.15.166
ASN Routes	[AS198949]	[AS198949]	[AS198949]
	[AS87]	[AS87]	[AS87]
	[AS87]	[AS87]	[AS87]
	[AS19782]	[AS19782]	[AS19782]
	[AS19782]	[AS19782]	[AS19782]
	[AS19782]	[AS19782]	[AS19782]
	[AS19782]	[AS19782]	[AS19782]
	[AS11537]	[AS11537]	[AS16509]
	[AS16509]	[AS16509]	
Destination IP	34.199.156.206		

the proxy server and then go to the internet. As demonstrated in the following table IV, a new ASN ([AS27198]) is added to the existing normal ASN route.

V. DISCUSSION

The services provided by Block-Pi should be consistently reliable and available to appropriate devices while unavailable to other devices. Access to a compromised Edge or cloud service may create serious, undetectable risks. An example of such an undetectable threat on the larger web is the large-scale hijacking of DNS[4] of a massive numbers of domains. Another remote, and often undetectable, attack includes IP hijacking. This would be particularly dangerous when a highly trusted IP address is hijacked. An IoT example would be an address for trusted updates; a historical example is Barracuda using an IP address range to enable access for its own purposes.

One point of discussion is how to add new devices to the proposed system. Block-Pi would be well informed by interacting the IETF proposed MUD or the AllJoyn trust model in identification of the legitimate initial accesses. however, should a MiTM occur at first connection, recovery is a real challenge. In any case when a new device is added to the system, a pre-evaluated model may be useful as some training period is needed as described in section III-C. If we can integrate a newly attached device with the pre-evaluated model, Block-Pi does not have to build a new preliminary model from the initial data set.

It is possible that the attacker may find a vulnerability and compromise an IoT device before our system builds a trained model when the ML model is fully implemented in the future work. The integration of the pre-evaluated model would be beneficial as long as there is no corruption. If we want to integrate pre-evaluated models, we currently are considering the two following cases. These should be subject to discussion, and other cases are of interest.

The first case assumes a separate central server, aka Block-Pi Server, coordinates the integration of new devices. The systems with Block-Pi would voluntarily work with the central server and pass relevant data to build the pre-evaluated model. All the pre-evaluated models are then collected for each IoT device and the central server can integrate and customize them for the local environment. We have experimented this approach with the ASN route data in this paper as mentioned in Section III-C. It is possible that even Block-Pi Server does

[4]https://krebsonsecurity.com/2019/02/a-deep-dive-on-the-recent-widespread-dns-hijacking-attacks/

TABLE III
ASN ROUTES FROM KANGAROO MOTION SENSOR TO THE EDGE SERVER (HTTPS) IN AWS.

	Comcast		
Source IP	68.50.16.40	68.50.16.32	68.51.126.206
ASN Routes	[AS198949]	[AS198949]	[AS198949]
	[AS7922]	[AS7922]	[AS7922]
	[AS7922]	[AS7922]	[AS7922]
	[AS7922]	[AS7922]	[AS7922]
	[AS7922]	[AS7922]	[AS7922]
	[AS7725]	[AS7725]	[AS7725]
	[AS7922]	[AS7922]	[AS33287]
	[AS7922]	[AS7922]	[AS7922]
	[AS7922]	[AS7922]	[AS7922]
	[AS7922]	[AS7922]	[AS7922]
	[AS7922]	[AS7922]	[AS16509]
	[AS16509]	[AS16509]	
	[AS16509]	[AS16509]	
Destination IP	34.199.156.206		

TABLE IV
A MITM ATTACK IS APPLIED TO THE BLOCK-PI. A NEW ASN [AS27198] IS ADDED TO THE EXISTING ASN ROUTE

Kangaro	
Normal	After MiTM
[AS198949]	[AS198949]
[AS87]	**[AS27198]**
[AS87]	[AS87]
[AS19782]	[AS87]
[AS19782]	[AS19782]
[AS19782]	[AS19782]
[AS2381]	[AS19782]
[AS11537]	[AS2381]
[AS16509]	[AS11537]
	[AS16509]

not have the pre-evaluated model, which is ASN route history of an IoT device on our current paper, e.g. in case a new IoT device is released. In this case, however, the IoT device is as unknown to Block-Pi Server as it is to an attacker. In other words, both the Block-Pi Server and attackers are not prepared for the new device. Even so, if Block-Pi is integrated with Manufacturer Usage Description (MUD) [20], which is part of our future work, Block-Pi would be able to verify the proper usage and network traffics of the new IoT devices by utilizing their MUD file.

Second, all the integration could be done via peer to peer connections building on the model of wisdom of crowds. This also assumes a certain amount of community homogeneity. Since Block-Pi does not communicate with the central system, there is no concern of exposing the personal access events to others. Block-Pi can be configured to communicate only with trusted neighboring Block-Pis. This would enable Block-Pi to share high-ranking pre-evaluated models with other Block-Pi systems. It is possible that the neighbor system has already been compromised and that the pre-evaluated model has been corrupted as well. There is no foolproof way to verify that the imported model is safe on Block-Pi.

The human interaction component is currently under active research, and recommendations are sought. The entire system fails if the recommender is subverted, so that it becomes a denial of service attack. It may also be rejected if the false positives are too frequent as discussed above.

VI. Conclusion

In this work, we have proposed a privacy-friendly method for detecting MiTM attacks targeting IoT devices. We implement this in form of a middleware called Block-Pi which is placed in the network and the traffic is passed through it. Block-Pi then runs a packet analysis based on the data and information about TLS certificates, DNS Cache server, ASN routes with resolved IP addresses, WHOIS information and PhishTank data, to fingerprint the Edge services. Unlike alternative proposals, Block-Pi guarantees the privacy of all the activities of the IoT devices in the users' network.

Having a Block-Pi that takes information in but does not broadcast to communicate with the outside world makes simultaneous or mass subversion of many classifiers difficult. It is well aligned with the cloud architecture, enabling customization for the specific IoT or SCADA devices and environment.

Acknowledgment

This research was supported by Cisco Research Award funding. This material is based upon work supported by the National Science Foundation under Grant CNS 1814518 and CNS 1565375. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. We thank our colleague Zheng Dong from Microsoft, whose insight and expertise enhanced our work; mistakes are entirely our own.

References

[1] Altaf Shaik and Ravishankar Borgaonkar. New vulnerabilities in 5g networks,, 2019.

[2] Patrick Nohe. Executing a man-in-the-middle attack in just 15 minutes. [online] Available: https://www.thesslstore.com/blog/man-in-the-middle-attack-2/ [Accessed 03/20/2020].

[3] Sandra Siby, Marc Juarez, Narseo Vallina-Rodriguez, and Carmela Troncoso. Dns privacy not so private: the traffic analysis perspective. In George Danezis, editor, *HotPETS*, Barcelona, SP, 2018. PETS 2018.

[4] Huichen Lin and Neil W Bergmann. Iot privacy and security challenges for smart home environments. *Information*, 7(3):44, 2016.

[5] Noah Apthorpe, Dillon Reisman, and Nick Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic. 2019.

[6] Leandros A Maglaras, Jianmin Jiang, and Tiago J Cruz. Combining ensemble methods and social network metrics for improving accuracy of ocsvm on intrusion detection in scada systems. *Journal of Information Security and Applications*, 30:15–26, 2016.

[7] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar. Security and privacy in fog computing: Challenges. *IEEE Access*, 5:19293–19304, 2017.

[8] Cheng Li, Zhengrui Qin, Ed Novak, and Qun Li. Securing sdn infrastructure of iot–fog networks from mitm attacks. *IEEE Internet of Things Journal*, 4(5):1156–1164, 2017.

[9] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. Iotsense: Behavioral fingerprinting of iot devices. In *researchgate*, 04 2018.

[10] Lin Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. Analyzing forged ssl certificates in the wild. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 83–97, Washington, DC, USA, 2014. IEEE Computer Society.

[11] Mishari Al Mishari, Emiliano De Cristofaro, Karim M. El Defrawy, and Gene Tsudik. Harvesting SSL certificate data to identify web-fraud. *I. J. Network Security*, 14(6):324–338, 2012.

[12] H. Liu, T. Spink, and P. Patras. Uncovering security vulnerabilities in the belkin wemo home automation ecosystem. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 894–899, March 2019.

[13] Italo Dacosta, Mustaque Ahamad, and Patrick Traynor. Trust no one else: Detecting mitm attacks against ssl/tls without third-parties. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security – ESORICS 2012*, pages 199–216, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[14] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1133–1150, Santa Clara, CA, August 2019. USENIX Association.

[15] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. A billion open interfaces for eve and mallory: Mitm, dos, and tracking attacks on ios and macos through apple wireless direct link. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 37–54, Santa Clara, CA, August 2019. USENIX Association.

[16] Patrick Dssel, Christian Gehl, Ulrich Flegel, Sven Dietrich, and Michael Meier. Detecting zero-day attacks using context-aware anomaly detection at the application-layer. *International Journal of Information Security*, 16:475–490, 07 2016.

[17] Zheng Dong, Apu Kapadia, Jim Blythe, and L Jean Camp. Beyond the lock icon: real-time detection of phishing websites using public key certificates. In *2015 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–12, Barcelona, Spain, 2015. IEEE.

[18] Daniel Kahneman and Patrick Egan. *Thinking, fast and slow*, volume 1. Farrar, Straus and Giroux New York, 2011.

[19] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[20] E Lear, R Droms, and D Romascanu. Rfc 8520: Manufacturer usage description specification. In *Internet Engineering Task Force, March*, 2019.