An AI enabled system for Distributed System Characterization

Seraphin B. Calo, Dinesh Verma, Maroun Touma, Franck Le, Douglas Freimuth, Erich Nahum
IBM TJ Watson Research Center, Yorktown Heights, NY, USA
Email: {scalo, dverma, touma, fle, dmfreim, nahum}@us.ibm.com,

***Abstract*—Determining the constituent components of a distributed system and how they are interacting is in general a very difficult problem. It requires the accumulation of evidence bearing on alternative propositions and decision functions for each of the set of attributes that characterize the elements of the system and their operation. In general, the outcome depends not only on the state of the accumulated evidence but also on the cost of acquiring this evidence; and, the accuracy of the decision functions and the process for combining their outputs. In this paper we describe a characterization system that was developed for identifying IoT devices present in an IP environment based on interpretations of the network traffic that is being generated. We argue that the architecture can be applied to address many kinds of similar problems by changing the analytics and the manners in which they are interconnected.**

***Index Terms*—distributed systems, IoT, analytics, machine learning**

I. Introduction

Distributed System Characterization is the problem of discovering different entities in distributed systems, understanding their behavior and how they interact with each other. The entities in the distributed environment may be physical or virtual. A specific instance of distributed system characterization is the task of discovering IoT devices that are present in an enterprise network. Such a system would need to discover all the devices in the network, and understand their behavior to determine whether a discovered device is an IoT device or some other kind of device (e.g., PCs and servers would not usually be considered IoT devices). Some devices, such as tablets may be considered to be IoT devices or not depending upon how they are being used. Another example of distributed system characterization would be the task of understanding the classes of applications that are active in an enterprise environment. The analytic techniques needed to make such determinations depend upon the types of devices present and the available information about them.

The SCADS (Structured Collections of Analytics for Distributed Systems) system is a scalable platform that allows multiple algorithms to be deployed and combined in analysis chains, and process high speed real time traffic in a distributed manner. While our primary goal was to develop a system for Smart IoT device discovery, the system we have created is based on an architecture that incorporates many different types of analytics in a structured manner, and can be used for many different contexts. Some of the analytics in the system are based on AI techniques, but others use transactional analysis and encoded domain knowledge. The resulting characterization system examines the network traffic and uses the analytics to discover the characteristics of the devices, virtual and physical, that are part of the distributed system.

In order to do so, the SCADS system needs to collect information. This can be done with active or passive probes. Active probing would require test transactions whose outcome depended on the responses of communicating devices. These would have to be carefully constructed to elicit the desired information without interfering with the operation of the system. We opted to go for passive probing of the information available in the environment. Such information includes the traces of network traffic, and logs from well-known system services such as the domain name service [5] and dynamic host configuration protocol service [6].

The discovery of IoT devices cannot be done using traditional network management tools because the wide variety of devices from a diverse set of vendors do not support traditional network mapping and discovery protocols. However, such a discovery mechanism can significantly reduce the manual burden when enterprise assets are entered into a device registry or configuration database; or when an existing IoT solution needs to be migrated from a private network to a public cloud platform; or during a device audit. Security scans are especially critical in enterprise networks for identifying devices with known vulnerabilities. The system can also report on the existence of any banned or prohibited devices, and the use of any improper communications (e.g., not using TLS when required, contacting blacklisted sites, etc.).

The paper is organized as follows: Section II discusses related work; Section III gives a description of the architecture of the system; Section IV presents the concept of processing elements and describes the major such elements that are used in the current system implementation; Section V describes the training and deployment of machine learning models for determining the attributes of IoT devices, and their organization in analysis chains; Section VI lists some other possible use cases to which we believe the SCADS system can be applied; and, Section VII presents a few conclusions and outlines research issues for further investigation.

II. Related Work

The characterization problem is a broad problem encompassing sub-problems such as the well-studied network topology discovery problem [1], and the relatively under-explored problem of information discovery in computational grids [2]. Different approaches to discover device topology or existence of specific processes can be found in the literature ([1]- [4]). In particular, IoT devices have become a prime target for security attacks, and the problem of discovering IoT devices has gained growing attention in recent years. Miettinen et al. [12] proposed a method to fingerprint IoT device types from their network traffic, so that potentially vulnerable IoT devices can be appropriately identified and quarantined. Similarly, Sivanathan et al. [13] developed a classification method to identify IoT devices and their type from their network traffic. More recently, Guo and Heidemann [14] proposed a method that analyzes DNS traffic to detect IoT devices, and identify their type. It is based on the observations that IoT devices regularly exchange traffic with servers run by their manufacturers, and those servers tend to be distinct for each class of IoT device.

Rather than rely on any one approach to identifying and characterizing the devices on an enterprise network, our direction has been to develop and utilize a number of different approaches and combine their results. We believe that the current paper is the first one to put forward a common framework for characterization of distributed systems which can be applied to many different contexts and environments.

III. General Architecture

The SCADS system architecture follows an edge computing paradigm: AI models are created and trained in the cloud, and then sent to an observation device at the edge. The edge devices monitor network traffic and/or participate in IoT protocols and use those AI models to create profiles. Each profile is a set of attributes that capture the characteristics of the entities present in the system. As an example, when the task of the system is to create an inventory of IoT devices in the environment, the profile may consist of fields indicating the type of device, the manufacturer and model of the device, along with the characteristics of the payload being sent by the device.

The resulting system consists of two main classes of components observers and controllers, the former being located at the edge devices and the latter in the cloud. The observers provide the system observation functionality. In systems driven by real-time streaming traffic or from captured traces or probe data based upon such traffic over some time period, they are located at the edge of the network, and extract and filter the traffic data (Figure 1). Alternatively, if the system were being used in more general environments where the characterization input was not based on streaming data, the observer functionality could reside either at the edge or in the cloud depending upon the structure of the system being analyzed.

The observers run chains of analysis modules, which are divided into two types, Protocol Processing Elements (PPEs) and Cognitive Processing Elements (CPEs). PPEs are based on analytics that examine and interpret the known characteristics of the protocols used between communicating elements of the system. CPEs embody machine learning techniques based

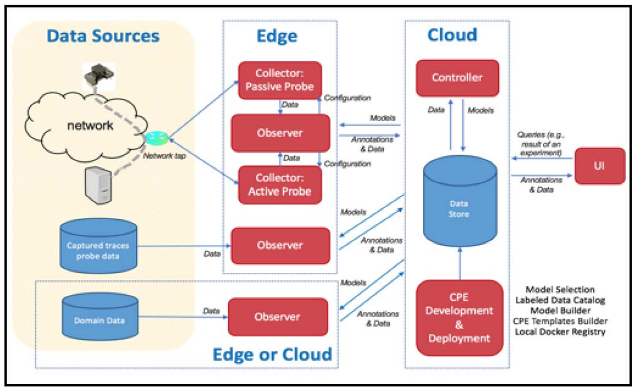


Fig. 1. SCADS Architecture

upon behaviors reflected in previously captured training data. The results of all the analyses are sent to the controllers, where they are combined and stored.

The controllers are located in the cloud and coordinate the activities of the observers. They can also perform analysis functions that may be too computationally expensive to be done at the observers. Typically, there would be one controller for many observers. More than one might be required if some supporting functionality were provided by specialized hardware, or there were very high performance or storage considerations.

SCADS system controllers also include tools that support easily producing the Machine Learning (ML) based CPEs, with functionality for engineering features, training, and deploying and scoring many common AI models (KNN, SVM, Random Forest, CNN, etc.). New AI models and classifiers would be trained in the controller, and then deployed to the associated observers.

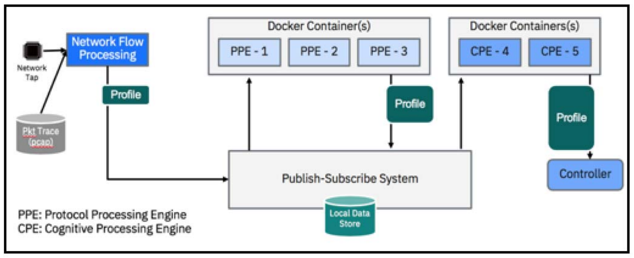


Fig. 2. Structured Collection of Analytics

The analysis components are structured in chains (Figure 2) that can be either individually selected or run in parallel against the same input stream. Each analysis component adds the results of its analysis to a profile capturing the values of the attributes that characterize the element generating the input stream. The profiles keep on getting enriched as they move through the chains, and the resulting profile at the end is sent to the controller.

The CPEs in an analysis chain determine the values of one or more of the profile attributes. One CPE can, for example, determine the type of device generating the network traffic (e.g., a video camera), while another determines the model and manufacturer (e.g., Samsung SmartCam). The controller combines the multiple results to produce a common profile characterizing the behavior of the device producing the traffic being analyzed. For different use-cases and applications of SCADS, different sets of chains would be needed.

The Composition of the outputs of the CPEs can be done in a number of different ways. The collection of classifiers can be viewed as an ensemble with all relevant classifiers contributing to the final decision on each attribute. This assumes that the classifiers are diverse and have competitive accuracies. Alternatively, the most accurate classifier for each attribute can be chosen to make the decision. This assumes that there is a way to determine the expected accuracy of each method of analysis. In systems where the ground-truth can be ascertained for a set of classified devices, this set could be used to provide a scoring set for the accuracy of the different CPEs.

The input to the chains of analysis components comes from a set of protocol parsers that extract and represent the network data. These are deployed at the observers to reduce the amount of data transferred to the cloud. The network protocol parsers focus on key events and extract the semantics behind the stream of bytes. More specifically, we rely on the open source software Bro which supports a wide range of protocols including TCP, DNS, DHCP, HTTP, etc. For each protocol, Bro creates a log which consists of key features (e.g., duration of TCP connections, number of sent bytes, number of received bytes, etc.) We extended the existing parsers to extract additional features: for example, we extended the TCP parser to record the minimum size of the sent data packets, and the minimum size of the received data packets. Also, we extended the parser to extract the first N bytes of each connection so we can apply an auto encoder to automatically discover and identify features for the machine learning components.

For the communication between the different components (e.g., network protocol parsers, PPEs, CPEs, etc.), we adopted a pub/sub communication model. More specifically, each network protocol parser publishes logs into a message bus. We use the open source distributed streaming platform Kafka for this purpose. PPEs and CPEs can then subscribe to events of interest, process them, and publish their outputs to the pub/sub platform. This design decision allows us to smoothly add and remove CPEs and PPEs. In addition, CPEs and PPEs can flexibly be combined in various configurations depending upon the needed flow of the analytics being used in particular chains.

IV. Processing Elements for IOT Discovery

The analytics for characterization are based upon insights perceived about the elements of the system and the patterns of traffic that they generate. For our IoT discovery example, these predominantly involve the structures of the various protocols used in IP communications.

Figure 3 shows the distribution of the different network protocols used by various data connections in traffic data collected from two different environments in which the SCADS system was deployed. The first environment consisted of a system which primarily had workload driven by IoT devices.

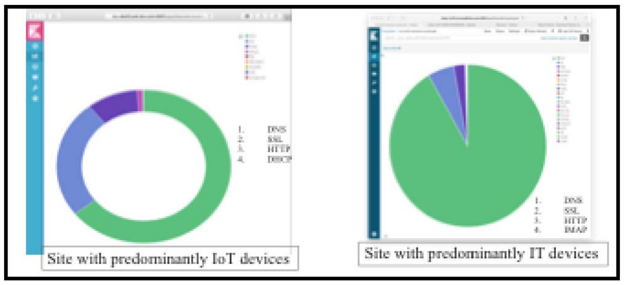


Fig. 3. Network Protocol Distribution

This environment was a specialized IoT lab established for the purposes of our project. The second environment consisted of a distributed system which had workload primarily driven by IT devices such as laptops and mobile phones which were interactively used by human beings. This environment was an enterprise network that we were able to observe for a prolonged period. While the exact distribution of the protocols in different environments are different, it is clear that DNS [5], TLS [6] and HTTP [7] are responsible for the majority of TCP connection establishments in these environments, so the analysis of these protocols provides a useful way to understand the traffic. DHCP [8] queries counted as the fourth dominant protocol for IoT lab, whereas the IMAP [9] protocol used for email access was the fourth dominant protocol for the enterprise environment.

Based upon the insights gained by analyzing traffic in different environments, a number of processing elements have been developed within for the SCADS system. Protocol Processing Elements (PPEs) exist for:

- examining protocol and address information from the packets flowing in the network;
- mapping protocol data from various sources into a common data abstraction; and,
- assigning each device a unique device ID

Cognitive Processing Elements (CPEs) exist for:

- determining whether a device is an IoT device;
- mining data from DNS look-ups;
- invoking Internet services like WhoIs and IP2Location for identifying characteristics of source and destination addresses;
- inspecting HTTP user agent and URI strings for device details;
- analyzing TLS certificate information for encrypted traffic; and,
- interpreting time-series characteristics of the network data streams.

These CPEs, PPEs and chains composed from them allow an enterprise to discover the different IoT devices present within their environment and the nature of their communication with servers on the Internet. Examples of the

CPEs that are currently active in the system and the insights that they incorporate are described below.

1) IoT / non-IoT: The distinction between IoT and non-IoT devices is made on the basis of the number of remote end-points with which a device communicates. IoT devices, which are performing automated functions of a specific type, e.g. monitoring temperature, listening to sounds, or capturing video or images, do not have an active human user. They tend to communicate with a small number of destinations. Non IoT devices, which usually have a human operator, have a different signature pattern, one where several destinations are contacted, and the spread is much more diverse. This distinction can be observed whether the destinations are characterized by their address, their domain names, their geographic location, or using the dominant prefix of their subnet address.

Figure 4 shows the typical distinction in the access patterns of an IoT device versus that of an IT device. IoT devices tend to access a relatively small number of remote destinations, whereas IT devices access a relatively large number of remote destinations. The access patterns of different types of devices and the threshold which demarcates an IoT device from a non-IoT device can be learnt using a machine learning model. This threshold would vary for each of the protocols used for identifying a destination (e.g., a DNS name, an IP address, an IP subnet prefix, or a geographical tag), and for each deployment environment.

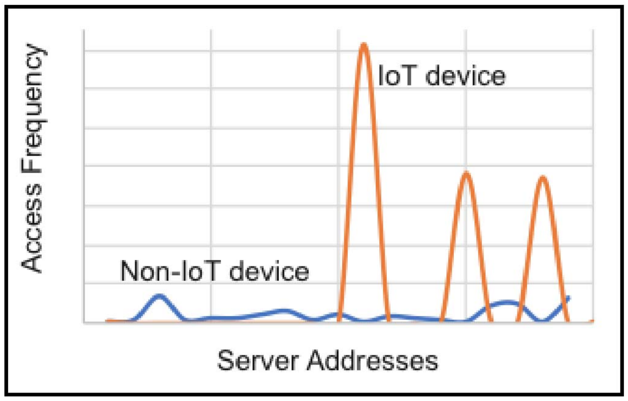
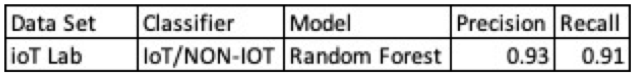


Fig. 4. IoT and non-IoT Communication Pattern

A number of CPEs have been developed to determine whether a device is an IoT device or an IT device. Some are using shallow ML models with appropriate training data. Others are using protocol information to make the determination. Figure 5 shows the performance of a Random Forest ML model for making the IoT / non-IoT determination.



Data Set	Classifier	Model	Precision	Recall
ioT Lab	IoT/NON-IOT	Random Forest	0.93	0.91

Fig. 5. Performance of an IoT / non-IoT CPE

2) Domain Name Service (DNS): DNS [5] handles queries issued by IP devices. These queries are a required part of network communication and translate a server name into an IP address. They precede the establishment of communications between the device and the server. Ways in which the DNS traffic tends to differ for IoT devices include:

- IoT devices have a much smaller spread of connectivity (number of unique destinations, domains, geographies and user agent software products) than IT devices. We are able to prove this known heuristic using an random forest model with a precision of 0.93 and a recall of 0.91.
- DNS names queried by an IoT device predominantly belong to the organization that manufactured that device
- Distribution of domain names queried by an IoT device identifies the model of the device being used
- DNS sequence patterns can identify device types across different vendors

The pattern for IoT access for server addresses shown in Figure 4 is repeated when we plot DNS queries instead of server IP addresses. Figure 6 shows the typical DNS access patterns made by two IoT devices that were observed in our environment. The domain name queries made by a Samsung Smart TV which is used primarily as a display device and the queries made by an Amazon Fire TV are shown. While several domain names are queried, mapping the domain names to the owning organization, as revealed by the Whois information shows that the dominant owner is the same as the manufacturer of the device. This provides a mechanism that can be used to identify the manufacturer of a device based on domain ownership.

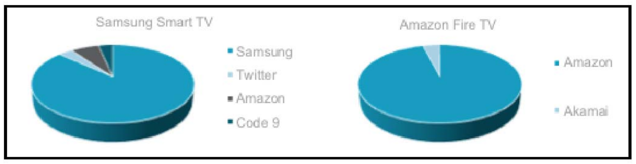


Fig. 6. DNS Query Distribution of two typical IoT Devices

Furthermore, the distribution of specific domain names among different models of the same manufacturer helps us in identifying the specific model of the device. As an illustrative example, we show the distribution of domain names made to the manufacturer site by three different devices made by the same manufacturer (netatmo) in Figure 7. While they access the same set of domain names owned by their manufacturer, the relative frequency of access is very different. This allows us to use techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) [10] to detect the specific model of a device whose manufacturer is known.

Based on these insights, two DNS based CPEs have been developed: DNS-Vendor for device manufacturer identification; and, DNS-Type for device type identification. The first uses the heuristic that the owning organization for the DNS names queried most often by the device belong to its manufacturer.

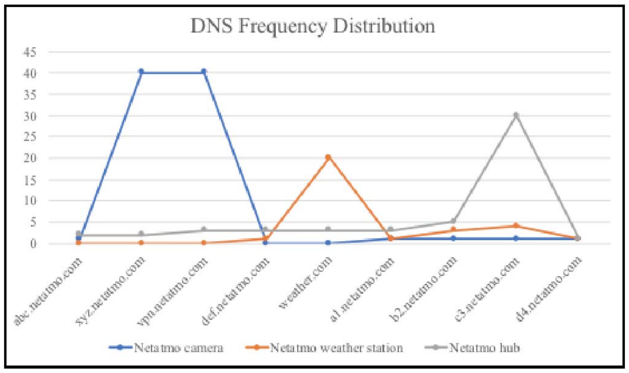


Fig. 7. DNS query patterns of different devices of same manufacturer

The second uses TF-IDF on the DNS query names to identify the device type.

The DNS CPEs also show the value of chaining in the context of our architecture. The CPE determining the manufacturer should only be invoked after a CPE has made the determination that the specific device being observed is an IoT device since the heuristic approach would not work for other type of devices. Similarly, since the original training of ML system for TF-IDF requires patterns that are specific for a manufacturer, the CPE determining the model can only be used after the manufacturer has been identified, and it has to be the third element of the chain. The same CPEs, if used in a different order in the chain, would produce results with much lower fidelity.

3) Transport Layer Security (TLS): The TLS protocol [6] aims primarily to provide privacy and data integrity between two or more communicating computer applications. Even though the payload in the TLS traffic is encrypted, there exist various ways to identify characteristics of devices that use TLS for communication. Some of the approaches that we have created CPEs within the SCADS system include:

- The sequence of packet size and direction in TLS have patterns that can be used to identify client devices
- Client Certificates used in a TLS handshake provide information about the manufacturer of the device
- Cipher-suites proposed for TLS communication yield information about the nature of the originating device, including information about the application stack used on the device

The TLS CPE uses the set of client proposed cipher suites plus the accepted cipher suite, the set of session destination endpoints (addresses and/or host names), and volumetric information for the TLS record session (after negotiation) to identify the device type.

Figure 8 shows the performance of this CPE in the lab environment with predominantly IoT devices that was discussed earlier. The CPE was trained on part of the network traffic in the lab and used to detect devices on traffic on a date that was not part of the training data set. The set of cipher-suites that were proposed by different types were used to determine the identity of a device using encrypted communications. As the resulting F-scores show, the analysis of TLS sessions could identify the nature of communicating devise with a high degree of fidelity.

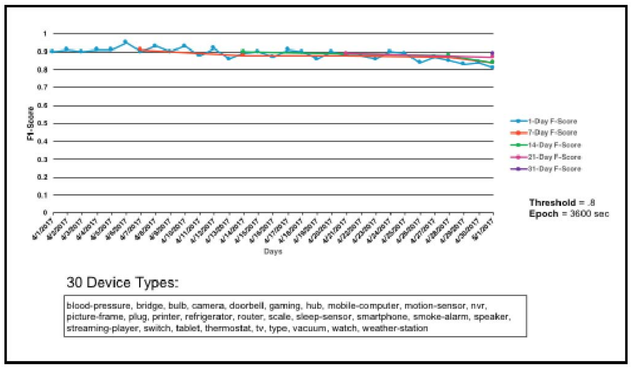


Fig. 8. Performamce of the TLS CPE in the IoT Environment

4) HyperText Transfer Protocol (HTTP): In some cases, communication among devices may happen in the clear instead of using an encryption protocol like TLS. A common protocol used for such communication is HTTP [7]. It is an application level protocol for hypermedia information systems and follows a request/response paradigm in the client/server computing model. HTTP is in wide use across the internet today. Our discovery system only requires a few transactions on HTTP to identify the device, while the same device might also predominantly use TLS and encrypted data. The HTTP header reveals detailed information about the devices communicating and the data being communicated. Some relevant fields include:

- HTTP User Agent contains information about the device type, manufacturer, model, its operating system and application
- HTTP URI field contains information identifying the device type and application on the device thus improving recall

These fields can be very revealing. By definition, the user agent field contains information about the client making the request and the software products being used. The data from these fields lend itself to device identification via regular expression matching, machine learning and more advanced AI methods. We have implemented a CPE based on the TF-IDF text analytics [10] on the HTTP User Agent and URI fields to identify the device type, manufacturer, device model, operating system and other attributes when present. These two fields can complement each other when being used to identify the type of device present on the network.

By examining the statistical characteristics of HTTP, we have found properties like IoT devices use a smaller set of user agent strings than traditional IT devices. This is not surprising since, IoT devices are typically lower functioning than traditional IT devices. We have also found our HTTP CPE to be highly precise and efficient at discovering both IoT and traditional IT devices.

Source	Large Enterprise
Number of HTTP Devices	1348
Date	April 2018
Length (days)	3

Fig. 9. HTTP Dataset Summary

Algorithm	Accuracy
HTTP Discovery	98%

Fig. 10. HTTP Results Summary

A. Evaluation

We evaluated the HTTP Discovery algorithm and describe its effectiveness below.

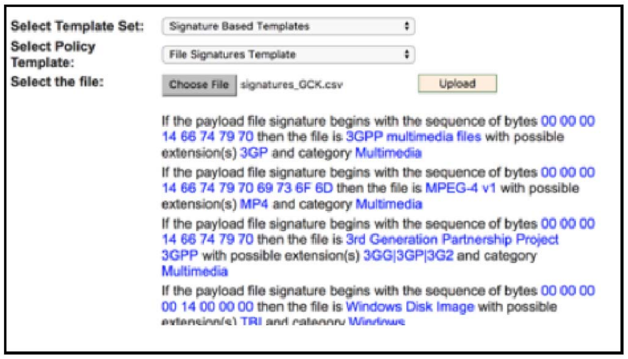
1) Methodology: We used actual HTTP data from devices deployed in a Large Enterprise environment which was captured for three days in April 2018 (Figure 9). There were 1348 devices in the ground truth that used HTTP to communicate and presented themselves for discovery by our HTTP based algorithm. We captured the live stream of data and passively monitor the network data for classification. We process HTTP events and use the user agent string and URI string over a 15 minute epoch to classify active devices.

2) Results: We had access to the ground truth for the devices that were deployed in the target large enterprise environment. The ground truth was built from the device management control system required by the IT department in the enterprise environment. Streaming data was captured, classified and scored resulting in HTTP Discovery correctly classifying 98% of the devices it classified (Figure 10).

3) Signature based CPEs: In many cases, devices have unique patterns that identify their characteristics, and these patterns can serve as signatures in the SCADS system. The SCADS system captures information about known signatures as rules and used to identify device attributes. Some observations include:

- Unique patterns defining the standard encodings of audio, video and other types of data can identify the type of payload carried in an IoT traffic data.
- Default naming conventions for device names in DHCP frequently reveal the type of a device, e.g. it may be common practice to name Apple Macbooks with a short acronym like MBP.
- Many applications are structured as finite state machines and can be identified by their states and state transitions

Many headers and payloads in IoT application traffic have unique signatures. For example, .wav encoded sounds begin with a well-defined header, and MPEG encoded video streams follow a standard syntax. A CPE has been developed to examine traffic streams to identify the patterns represented in a pre-defined set of rules. The rules to populate this CPE have been collected from sources on the web such as filesignatures.net, and a snapshot of some of these rules is shown in Figure 11. Although the conventions for names are different in different locations, a set of rules that are customized for the local environment can help to identify the type of device on the basis of the rules.



Select Template Set: Signature Based Templates

Select Policy Template: File Signatures Template

Select the file: signatures_GCK.csv

If the payload file signature begins with the sequence of bytes 00 00 00 14 66 74 79 70 then the file is 3GPP multimedia files with possible extension(s) 3GP and category Multimedia

If the payload file signature begins with the sequence of bytes 00 00 00 14 66 74 79 70 69 73 6F 6D then the file is MPEG-4 v1 with possible extension(s) MP4 and category Multimedia

If the payload file signature begins with the sequence of bytes 00 00 00 14 66 74 79 70 then the file is 3rd Generation Partnership Project 3GPP with possible extension(s) 3GG|3GP|3G2 and category Multimedia

If the payload file signature begins with the sequence of bytes 00 00 00 00 14 00 00 00 then the file is Windows Disk Image with possible extension(s) TBI and category Windows

Fig. 11. Sample Rules used to detect payload type

The names for IoT devices often would carry other details about them, e.g. identify the room or floor number where they are located. Application traffic can often be modeled as having been generated by a finite state machine (FSM), and can thus be characterized by a sequence of operations reflecting the application states and transitions. Rules can be used to identify the sequences associated with particular applications. A CPE has been developed to identify certain FSM based applications.

V. DEVELOPMENT AND DEPLOYMENT OF CPES

While the SCADS environment provides a flexible and general way to combine different types of analytics together to make inferences about the system, it needs training of the AI models for the ML based CPEs to work properly. In order to enable this operation, we have created tools for easing the development of these CPEs. Our end goal is to enable users to just upload a traffic file containing communication patterns for known types of devices, and the SCADS system uses a human-assisted approach to easily create a CPE chain which is capable of identifying devices with matching communication patterns.

1) AI Approach: AI approaches can be abstracted in terms of the 2-stage process shown in Figure 12. In the first stage of the process, a set of AI models are built using machine learning algorithms with a set of training data, by processing natural language documents, or by the encoding of human expertise. Models can be expressed in different ways, e.g., as inference rule sets, decision trees, or neural networks. Once these models are built, they can be used to make inferences from the sensor input data, and guide the operation of the system.

In the SCADS environment, the model building would be done on the controller in the cloud while the inference task would be done at the observers on the edge. This approach is bandwidth efficient and sends less data to the central location. In addition to reducing costs, and improving latency and

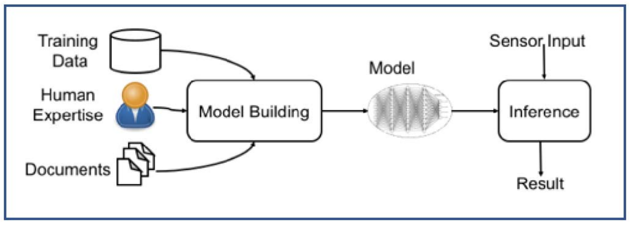


Fig. 12. Abstracted AI approach

responsiveness, this approach also has the advantage of improving the scalability of the system. The controller provides a set of services to support the analyst activities for training new models. they include a data management function for storing and retrieving labeled data, a model selection function that uses the labeled data to train multiple models simultaneously and provide a ranking based on the corresponding f-scores, and a model repository for storing the trained models ready to be deployed to the observers.

An alternative way to express the approach is that the SCADS system takes an approach of supervised learning to train its CPE chains. The packaging of the CPE and PPE chains is described next.

2) PEs, CPEs and Chains: An application may consist of many different steps, some of which may be based on AI technologies and some may not. In general, we consider an application to be composed of several processing elements (PEs), some of which we refer to as Cognitive Processing Elements (CPE) as previously discussed because they use an AI model to produce their output. An application would then consist of a flow-chart of processing elements that could contain multiple branches. A good representation for this would be produced by a system like Node-Red [11], which is a tool for building Internet of Things (IoT) applications with a focus on simplifying the wiring together of code blocks to carry out tasks.

Chains of processing elements are appropriately packaged along with any necessary supporting programs and deployed to the edge. This allows the distributed processing of data streams by sophisticated analytics, while leveraging the power of the cloud environment for development of the collections of processing elements needed, and overall management of the distributed edge environment.

This approach leverages the computational resources of the cloud to: (a) perform the complex learning operation; (b) simultaneously model, test and rank a large number of algorithms before deciding on the appropriate analytics that will be deployed to the edge; (c) use micro-services that are widely available in the cloud to engineer new features that enrich the training data set and improve the selected model accuracy; and, (d) define a verifiable process for the composition and coordinated deployment of CPE chains to the edge devices.

Templates for various CPE chains can be constructed in the cloud and associated with one or more AI models. The primary use of the CPE chain is to set up the data pipeline required for inferencing. A typical CPE chain includes five basic elements for performing data acquisition, feature engineering, aggregation, inferencing and reporting (Figure 13). CPE chains can also be augmented with user defined functions, such as user defined filters. They can also include system defined functions that are required for running the model on the edge, monitoring the performance of specific algorithms, and/or setting up triggers for reporting ML model scores or classification. CPE chains can also be designed to include multiple cognitive elements sequenced together to form a more complex cognitive program where the classification output produced by one AI model can be used as an input feature in the feature vector of a second AI model.

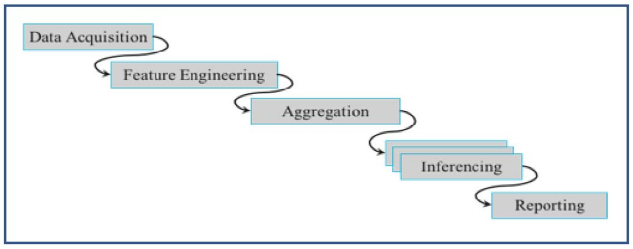


Fig. 13. Typical elements of a CPE Chain

3) Phases of Operation: The overall operation of the SCADS system can be viewed as occurring in three phases: Discover, Deploy, and Operate, as shown in Figure 14.

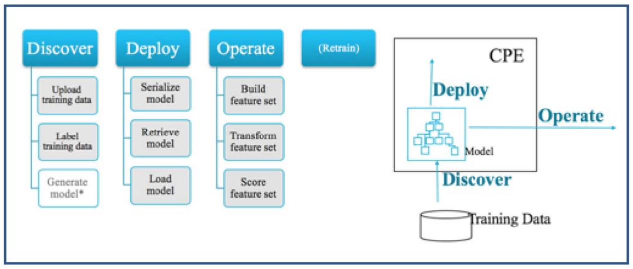


Fig. 14. Phases of AI Software Development

In the Discover phase, a user will be uploading training data, labelling it, constructing multiple machine learning models simultaneously using 80 percent of the training data and testing the resulting models with 20 percent of the data and provide a ranking of the models based on their respective f-score. There are several libraries from which such models can be obtained. The current implementation supports Tensorflow and scikit-learn open source libraries. The idea is to discover the models that are most effective for the domain of the training data. The top N such models are identified based upon their associated FScore and Loss function. The user can then choose one or more models for deployment. Automated model generation uses a combination of techniques that consists of: (a) training multiple models simultaneously (i.e., SVN, Random Forest, K-NN, etc.); (b) applying known feature engineering methods to identify potential improvement on the selected model; and, (c) using test data to compare different implementations of the same model (Python vs Spark vs SPSS).

In the Deploy phase, the chosen models are retrieved, serialized, and packaged as Docker containers that are loaded into a shared repository from which they can be accessed by the edge components. The corresponding CPE chain is implemented as a Node-Red flow with each of the nodes within the flow implementing a wrapper around the processing elements identified in the CPE chain. The Node-Red flow also includes the initial word map constructed during the training phase and includes the word vocabulary of the training set.

In the Operate Phase, a micro-service that is running on the edge device is responsible for instantiating the CPE flow and executing the associated Docker container(s) on the edge device upon request by the user. Within the edge device, the CPEs are organized into CPE groups based on feature selections and transformations that are defined during the training phase. Since CPE groups have similar feature vectors, all the pre-requisite data transformations required for running the models in the group (i.e., feature engineering, aggregation, word hash, etc.) are done once for all the ML models in the group. The models are customized to the appropriate feature set, which can be extended to include aggregations and engineered features. The latter take advantage of additional information that may be available that is known to be effective in identifying the characteristics of interest.

The Retraining phase is meant to capture information about the effectiveness of the CPEs that have been deployed so that they can be improved further. This feedback mechanism to the Cloud is meant to allow learning and continuous improvement, and can be viewed as a sub-phase of the operational phase.

VI. OTHER USE CASES

While the bulk of the discussion in the paper has been on describing the architecture and the analytics for the initial use case of detecting IoT devices, the same system can be used for many other use-cases, using a different sets of analytics and a different set of chains of CPEs and PPEs in the environment. In this section, we discuss some of those use-cases in brief.

Ease of Use for Asset Registration: In many cases, devices need to be registered in an inventory registration database for various maintenance and health-keeping functions. For active network devices, the SCADS system can be used to automate the registration process.

Device Behavior Identification: Different devices can operate in different modes. The typical tablet can be used as an embedded controller for a printer, as an entertainment device, or as a monitor in the building for security purposes. The SCADS system can be augmented to provide the information about different devices, and identify their behavior.

Device Security Audits: Many enterprises need to perform periodic audits of the devices that are located within their environment. These could be security audits to check that devices are behaving in an appropriate manner, as well as checking for existence of banned devices. The SCADS system can support a set of analytics that can ease these types of audits.

These are just some exemplars of the different types of characterization that the SCADS system is capable of doing. By adjusting the chains and the analytics elements, other use-cases for the system can also be supported.

VII. CONCLUSIONS

The SCADS system has been very effective at characterizing IoT devices within an enterprise environment. A major factor in its success has been the generality and flexibility of the architecture that it follows. It was recognized that there was no one unique approach that could keep pace with the changing IoT landscape, particularly the rapid increase in the number and types of new IoT devices and their growing sophistication and modes of interaction. Additionally, no one instance of the system could accommodate all deployment models.

It was thus imperative that the system be able to employ multiple types of analytics structured to the needs of particular environments. The ability to include both algorithmic and AI based components (PPEs and CPEs) gives SCADS a wider scope and greater effectiveness. For the Device Discovery example, it acknowledges that IoT protocols and signatures evolve in a deliberate fashion; while, AI/ML techniques are more generally adaptable to new areas of investigation, and better at dealing with more opaque situations (e.g., encrypted data). Combining the two makes the system more powerful.

The profile approach to characterization also allows the discovery problem to be decomposed into multiple smaller problems that can be addressed with their own analysis chains. The attribute values comprising the profile can be determined over time, and different methods of analysis can be used for each one. Indeed, multiple methods can be employed in discovering the value of a single attribute, both algorithmic and cognitive, thus increasing the precision of the result.

The distributed nature of the system, leveraging both edge and cloud computing, adds to its scalability and extensibility. Multiple observers can be used for complex or geographically diverse environments. Enterprise scale systems can be easily handled. At the same time, small scale versions of the characterization system can run on a laptop. SCADS can be tailored to the environment under investigation, from a home network to a widely distributed communications utility.

The architecture makes it easy to integrate different components, since both the configuration and the analysis chains are comprised of multiple, loosely linked elements. The system can be arranged to filter and reduce information flow in multiple stages, thus allowing it to handle data at scale.

We are investigating the use of the existing prototype to address other characterization problems. By changing the analytics and the manners in which they are interconnected, it can be applied to a range of different problems.

ACKNOWLEDGMENT

The SCADS system has been built with the help and support of several team members and programmers who have helped in both system development, the creation of PPEs and CPEs, curation of the data to train the CPEs, testing the overall system, and improving it iteratively. Special thanks are due to Dilip Kandlur, Steven Sawyer, Philip Nolan, Jorge Ortiz, Dimitrios Pendarakis, Erich Nahum, Catherine Crawford, Wendy Chong, Jack Beavers, and Lowell Wheeler for helping with different aspects of the overall system architecture and deployment. Different CPEs were developed by research colleagues such as Douglas Freimuth, Dean Steur, Xiping Wang, Keith Grueneberg, Pau-Chen Cheng, Ray Valdez, Ramya Raghavendra and Joshua Rosenkranz. Several other colleagues and summer interns helped us in collection of training data in different environments.

Some of the basic algorithms underpinning the logic driving the CPEs were developed in the context of program W911NF-16-3-0001 jointly funded by U.S. Army Research Labs and UK MoD. The SCADS system can be viewed as a transition of some of those algorithms in the context of a functional prototype.

REFERENCES

[1] B. Donnet and T. Friedman. ”Internet topology discovery: a survey.” IEEE Communications Surveys and Tutorials, vol 9, no. 4 (2007): 2-15.

[2] C. Schmidt and M. Parashar, Flexible information discovery in decentralized distributed systems, Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing, June 2003, (pp. 226-235).

[3] A. Ghose, G. Koliadis G and A. Chueng, Process discovery from model and text artefacts. Proc. IEEE Congress on Services, Jul 9, 2007. (pp. 167-174).

[4] M. Harchol-Balter, T. Leighton and D. Lewin, Resource discovery in distributed networks. Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing, May 1999, pp. 229-237.

[5] P. Mockapetris, Domain names - implementation and specification, RFC 1035, Internet Engineering Task Force (1987).

[6] P. Kocher, P. Karlton, A. Freier, C. Allen, T. Dierks, The Transport Layer Security (TLS) Protocol Version 1.3, IETF RFC 8446, August 2018.

[7] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, T. Berners-Lee, Hypertext Transfer Protocol HTTP/1.1, IETF RFC 2616, June 1999.

[8] R. Droms and T. Lemon, The DHCP handbook. Pearson Education, 2002.

[9] M. Crispin, ”Internet Message Access Protocol Version 4 Rev 1”, Internet RFC 3501, March 2003.

[10] J. Ramos, Using tf-idf to determine word relevance in document queries, Proceedings of Instructional Conference on Machine Learning, Dec. 2003, pp. 133-142.

[11] M. Blackstock, R. Lea, ”Toward a Distributed Data Flow Platform for the Web of Things”, Proceedings of the 5th International Workshop on Web of Things (WoT ’14), 2014.

[12] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A. R. Sadeghi, S. Tarkoma, IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT, IEEE International Conference on Distributed Computing Systems (ICDCS 2017)

[13] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, V. Sivaraman, Characterizing and Classifying IoT Traffic in Smart Cities and Campuses, IEEE Infocom Workshop on Smart Cities and Urban Computing (SmartCity 2017).

[14] H. Guo, J. Heidemann, IP-Based IoT Device Detection, Proceedings of the ACM 2018 Workshop on IoT Security and Privacy (2018).