IDENTIFICATION AND CLASSIFICATION OF IoT DEVICES IN VARIOUS APPLICATIONS USING TRAFFIC CHARACTERISTICS BY NETWORK LEVEL

BREEZY S[1], Dr.P.KAVITHA[2]

[1]Student, Department of Computer Science and Engineering, P.S.R. Engineering College, Sivakasi, India.
[2]Professor, Department of Computer Science and Engineering, P.S.R. Engineering College, Sivakasi, India.

---***---

Abstract-*The Internet of Things (IoT) is being hailed as the next wave revolutionizing our society, and smart homes, enterprises, and cities are increasingly being equipped with a plethora of IoT devices. Yet, operators of such smart environments may not even be fully aware of their IoT assets, let alone whether each IoT device is functioning properly safe from cyber-attacks.*

First, I instrument a smart environment with 28 different IoT devices spanning cameras, lights, plugs, motion sensors, appliances and health-monitors. I collect and synthesize traffic traces from this infrastructure for a period of 6 months, a subset of which I release as open data for the community to use. Second, I present insights into the underlying network traffic characteristics using statistical attributes such as activity cycles, port numbers, signalling patterns and cipher suites. Third, I develop a multi-stage machine learning based classification algorithm and demonstrate its ability to identify specific IoT devices with over 99% accuracy based on their network activity.

Finally, I discuss the trade-offs between cost, speed, and performance involved in deploying the classification framework in real-time. My study paves the way for operators of smart environments to monitor their IoT assets for presence, functionality, and cyber-security without requiring any specialized devices or protocols.

Key Words- IoT devices, Preprocess, Feature Extraction**,** SVM Classifier**.**

1. INTRODUCTION

The number of devices connecting to the Internet is ballooning, ushering in the era of the "Internet of Things" (IoT). IoT refers to the tens of billions of low cost devices that communicate with each other and with remote servers on the Internet autonomously. It comprises everyday objects such as lights, cameras, motion sensors, door locks, thermostats, power switches and household appliances, with shipments projected to reach nearly 20 billion by 2030. Thousands of IoT devices are expected to find their way in homes, enterprises, campuses and cities of the near future, engendering "smart" environments benefiting our society and our lives. One would expect that devices can be identified by their MAC address and DHCP negotiation. However, this faces several challenges: (a) IoT device manufacturers typically use NICs supplied by third-party vendors, and hence the Organizationally Unique Identifier (OUI) prefix of the MAC address may not convey any information about the IoT device; (b) MAC addresses can be spoofed by malicious devices; (c) many IoT devices do not set the Host Name option in their DHCP requests; indeed we found that about half the IoT devices we studied do not reveal their hostnames; (d) even when the IoT device exposes its hostname it may not always be meaningful. For these reasons, relying on DHCP infrastructure is not a viable solution to correctly identify devices at scale [1].

This paper, we address the above problem by developing a robust framework that classifies each IoT device separately in addition to one class of non-IoT devices with high accuracy using statistical attributes derived from network traffic characteristics. Qualitatively, most IoT devices are expected to send short bursts of data sporadically. Quantitatively, our preliminary work in was one of the first attempts to study how much traffic IoT devices send in a burst and how long they idle between activities. We also evaluated how much signalling they perform (e.g. domain lookups using DNS or time synchronization using NTP) in comparison to the data traffic they generate[2]. This paper significantly expands on our prior work by employing a more comprehensive set of attributes on trace data captured over a much longer duration (of 6 months) from a test-bed comprising 28 different IoT devices. There is no doubt that it is becoming increasingly important to understand the nature of IoT traffic. Doing so help unnecessary multicast/broadcast traffic, reducing the impact they have on other applications. It also enables operators of smart cities and enterprises to dimension their networks for appropriate performance levels in terms of reliability, loss, and latency needed by environmental, health, or safety applications. However, the most compelling reason for characterizing IoT traffic is to detect and mitigate cyber security attacks[3].

It is widely known that IoT devices are by their nature and design easy to in filtrate. New stories are emerging of how IoT devices have been compromised and used to launch large-scale attacks. The large heterogeneity in IoT devices has led researchers to propose network-level security mechanisms that analyze traffics patterns to identify attacks, success of these approaches relies on a good understanding of what "normal" IoT traffic profile looks like. Our primary focus in this work is to establish a machine learning framework based on various network traffic characteristics to identify and classify the default (i.e. baseline) behaviour of IoT devices on a network. Such a framework can potentially be used in the future to detect anomalous [4].

2. SYSTEM MODELING AND DESIGN

2.1 Modules

2.1.1 Preprocess

Preprocess, is a program that processes its input data to produce output like a compiler. We identify key statistical attributes such as port numbers, Domain Names and cipher suites, and use them to give insights into the underlying network traffic characteristics. For each device, if a port is used more frequently then it is shown by a larger font-size in the respective word cloud. It can be seen that IoT devices each uniquely communicate with a handful of server ports whereas non-IoT devices use a much wider range of services. DNS is a common application used by almost all networked devices. Domains such as example.com, example.net, and example.org are frequently requested by Amazon Echo, sub-domains of hp.com and hpeprint.com are seen in DNS queries from the HP printer. In order to initiate the TLS connection and negotiate the security algorithms with servers, devices start handshaking by sending a Client Hello packet with a list of cipher suites that they can support, in the order of their preference. The cipher suites that Amazon Echo offers to two different Amazon servers. Each cipher suite (i.e. 4-digit code) can take one of 380 possible values and represents algorithms for key exchange, bulk encryption and message authentication code (MAC). For example, the cipher 002f negotiated by an Amazon server uses RSA, AES 128 CBC, and SHA protocols for key exchange, bulk encryption and message authentication, respectively.

2.1.2 Feature Extraction

In Feature Extraction the IoT devices have the properties of their traffic flows. We define two key attributes at a per-flow level to characterize IoT devices based on their network activity: flow volume (i.e. sum total of download and upload bytes), flow duration (i.e. time between the first and the last packet in a flow). The flow volume for the Belkin motion sensor (depicted by green bars) is slightly higher; over 35% of flows transfer between [2800, 3800] bytes. For the Amazon Echo (depicted by blue bars), over 95% of flows transfer less than 1000 bytes. Though we present the flow volume histogram for only a few devices, most of our IoT devices exhibit a similar predictable pattern. The flow duration 40% of flows for Amazon Echo, while duration of 60 seconds is seen for the LiFX light bulb and Belkin motion sensor with a probability of 50% and21% respectively.

2.1.3 SVM Classifier

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. We note that three of our attributes namely "set of domain names", "set of remote port numbers" and "set of cipher suites" are nominal (i.e. are not treated as numeric values) and multi-valued (for example, {"53":3, "123":1, "443":2} represents a set of remote port numbers with three occurrences of port number 53, two occurrences of port 123, and one occurrence of port number 443). Our remaining attributes including flow volume/duration and DNS/NTP intervals contain single quantitative and continuous values

.

2.1.4 Performance Analysis

In Performance Analysis we compare the existing system with proposed system.

SVM classifier mechanism:

Support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. The key concept of SVMs, which were originally first developed for binary classification problems, is the use of hyper planes to define decision boundaries separating between data points of different classes. SVMs are able to handle both simple, linear, classification tasks, as well as more complex, i.e. nonlinear, classification problems. Both separable and non separable problems are handled by SVMs in the linear and nonlinear case. The idea behind SVMs is to map the original data points from the input space to a high-dimensional, or even infinite-dimensional, feature space such that the classification problem becomes simpler in the feature space.

Consider a training data set $\{x_i, y_i\}_{i=1}^N$, with $x_i \in \mathbb{R}^d$ being the input vectors and $y_i \in \{-1, +1\}$ the class labels. SVMs map the d-dimensional input vector x from the input space to the d_h-dimensional feature space using a (non)linear function $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d_h}$. The separating hyper plane in the feature space is then defined as $w^T\varphi(x) + b = 0$ with b ∈ R and w an unknown vector with the same dimension as ϕ (x). A data point x is assigned to the first class if f (x) = sign (w^Tϕ (x) + b) equals +1 or to the second class if f (x) equals−1. The SVM classifier starts from the following formulations

$$w^T\varphi(x_i) + b \geq +1 \quad \text{for } y_i = +1,$$
$$w^T\varphi(x_i) + b \leq -1 \quad \text{for } y_i = -1,$$

Equivalent to

$$y_i(w^T\varphi(x_i) + b) \geq 1, \quad i = 1, \ldots, N.$$

The classifier is written as,

$$f(x) = \text{sign}(w^T\varphi(x) + b).$$

However, in most real-life applications data of both classes are overlapping, which makes a perfect linear separation impossible. Therefore, a restricted number of misclassifications should be tolerated around the margin. The resulting optimization problem for SVMs, where the violation of the constraints is penalized, is written as

$$\min_{w,\xi,b} \mathcal{J}_1(w, \xi) = \frac{1}{2}w^Tw + C\sum_{i=1}^N \xi_i,$$

Such that

$$y_i(w^T\varphi(x_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, N,$$
$$\xi_i \geq 0, \quad i = 1, \ldots, N,$$

Where C is a positive regularization constant.

We develop the algorithm by following the procedure of derivation of the standard SVM for the linearly non separable case.

Consider the training samples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \quad \mathbf{x}_i \in \mathbf{X}, \quad y_i \in \{-1, 1\}, \quad i = 1,$$

These two classes cannot be separated without error by a hyper plane since there are misclassified patterns or measurement noises in the training samples. In this case, there is no decision function, such that, the inequalities hold true.

$$y_i f(\mathbf{x}_i) \geq +1, \quad i = 1, 2, \ldots l$$

An SVM is a general algorithm based on guaranteed risk bounds of statistical learning theory, i.e., the so-called structural risk minimization principle. It is a learning machine capable of implementing a set of functions that approximate best the supervisor's response with an expected risk bounded by the sum of the empirical risk and the Vapnik-Chervonenkis (VC) confidence. The latter is a bound on the generalization ability of the learning machine that depends on the so-called VC dimension of the set of functions implemented by the machine. SVMs can be used to solve pattern recognition, regression estimation and density estimation problems. SVMs have found numerous applications such as in optical character recognition object detection, face verification, text categorization, engine knock detection, bioinformatics, and database marketing and so on.

SVM is basically a two-class classifier based on the idea of "large margin" and "mapping data into a higher dimensional space". The principle of SVM is to make minimize the structure risk, in the high dimensional feature space, find an optimal discriminate hyper plane with low VC dimension to make the distance between the two classes' data have large margin. When the feature space is not linear dividable, SVM maps the data into high dimensional feature space with non-linear mapping, and finds the optimal classification hyper plane in high dimensional feature space.

Based on the principle of configuration risk minimization, suppose in inner product space F exists two kinds discriminable samples $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in R^n$, $y_i \in \{-1, +1\}$, $i = 1, 2, \cdots, n$. -1 and +1 denote two kinds; the optimal classification hyper plane can be expressed as:

$$\{\mathbf{x} \in F : (\mathbf{w} \cdot \mathbf{x}) + b = 0\}$$

Where w is support vector, b is translation vector. In order to make classification hyper plane and w one-to- one correspondence, we standardize it and let the distance of the sample which is nearest to hyper plane is $1/\|\boldsymbol{w}\|$. So hyper plane after standardization satisfies:

$$\min_{i=1,2,\cdots,n} |(\boldsymbol{w} \cdot \boldsymbol{x}_i) + b| = 1$$

Solving the optimal classification hyper plane can be transformed into quadratic optimization problem:

$$\min \quad \Phi(\mathbf{w}) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w})$$
$$s.t. \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0$$

The optimal hyper plane is discussed on the condition that samples can be classified linearly, if cannot, we will use slack variables $\xi_i \geq 0$ and penalty factor C to resolve generalized optimal classification hyper plane (to classify samples farthest an d make the largest classify margin at the same time):

$$\min \quad \Phi(\mathbf{w}) = \frac{1}{2}(\mathbf{w}\cdot\mathbf{w}) + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \begin{cases} y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0 \\ \xi_i \geq 0 \end{cases}$$

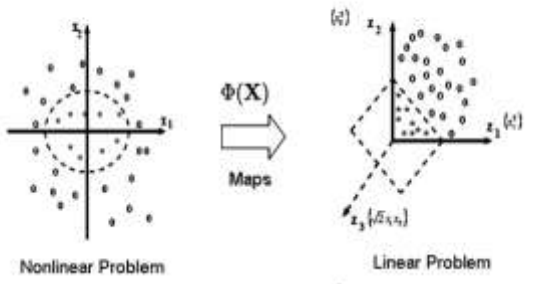
Where $i = 1, 2, \cdots, n$, C is a certain constant, it is the control of the punishment of samples which are classified mistakenly. It is a compromise between the proportion of false classified samples and algorithm complexity.

The SVM is a useful statistic machine learning technique that has been successfully applied in the pattern recognition area. If the data are linearly non separable but nonlinearly separable, the nonlinear support vector classifier will be applied. The basic idea is to transform input vectors into a high-dimensional feature space using a nonlinear transformation ϕ, and then to do a linear separation in feature space.

To construct a nonlinear support vector classifier, the inner product (x , y) is replaced by a kernel function K(x , y):

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l}\alpha_i y_i K(x_i, x) + b\right).$$

The SVM has two layers. During the learning process, the first layer selects the basis $K(x_i, x),\ i = 1, 2, \ldots, N,$ from the given set of bases defined by the kernel; the second layer constructs a linear function in this space. This is completely equivalent to constructing the optimal hyper plane in the corresponding feature space. The SVM algorithm can construct a variety of learning machines by use of different kernel functions. Three kinds of kernel functions are usually used. They are as follows:



$$\mathbf{Z} = \Phi(\mathbf{X}) = \{x_1^2, x_2^2, \sqrt{2}x_1x_2\}$$

Fig -1: linear and non linear problem

1. Polynomial kernel of degree d

$$K(X, Y) = (\langle X, Y\rangle + 1)_q.$$

2. Radial basis function with Gaussian kernel of width C > 0:

$$K(X, Y) = \exp\left(\frac{-|X - Y|^2}{c}\right).$$

3. Neural networks with tan h activation function:

$$K(X, Y) = \tan h(K\langle X, Y\rangle + \mu),$$

where the parameters K and μ are the gain and shift.

3. LANGUAGE STUDY: MATLAB

The MATLAB language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution. With the MATLAB language, you can program and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. In many cases, MATLAB eliminates the need for 'for' loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.

At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, object-oriented programming (OOP), and debugging features.

MATLAB lets you execute commands or groups of commands one at a time, without compiling and linking, enabling you to quickly iterate to the optimal solution. For fast execution of heavy matrix and vector computations, MATLAB uses processor-optimized libraries.

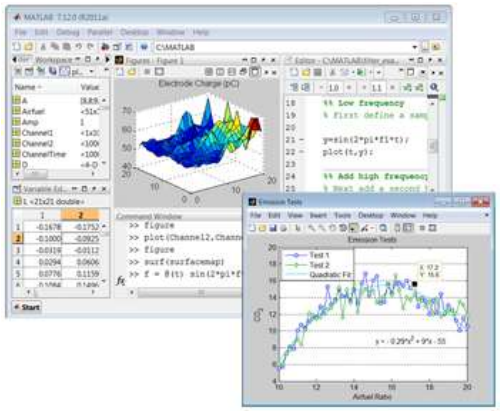


Fig -2: Compilation technology

3.1 MATLAB tools

Development Tools

MATLAB includes development tools that help you implement your algorithm efficiently. These include the following:

MATLAB Editor - Provides standard editing and debugging features, such as setting breakpoints and single stepping

Code Analyzer - Checks your code for problems and recommends modifications to maximize performance and maintainability

MATLAB Profiler - Records the time spent executing each line of code

Directory Reports - Scan all the files in a directory and report on code efficiency, file differences, file dependencies, and code coverage.

Designing Graphical User Interfaces

You can use the interactive tool GUIDE (Graphical User Interface Development Environment) to layout, design, and edit user interfaces. GUIDE lets you include list boxes, pull-down menus, push buttons, radio buttons, and sliders, as well as MATLAB plots and ActiveX controls. Alternatively, you can create GUIs programmatically using MATLAB functions.

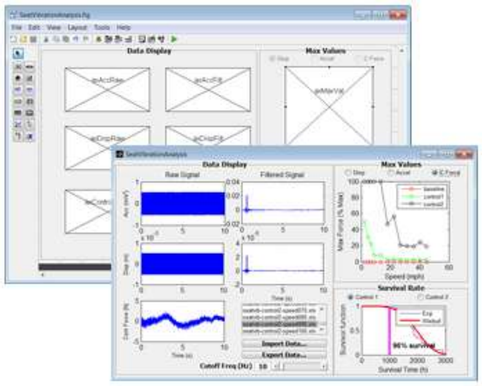


Fig -3: GUID

4. IMPLEMENTATION DETAILS AND RESULTS

4.1 Preprocess values

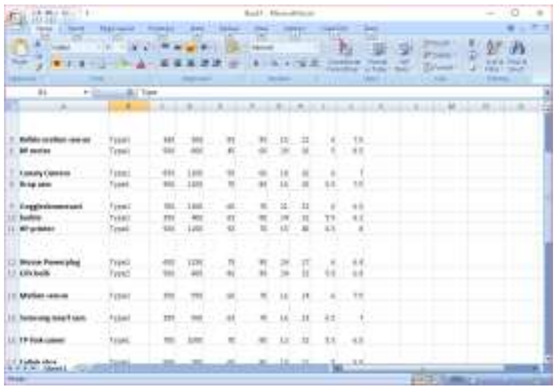


Fig -4: Preprocess values

4.2 Feature extraction

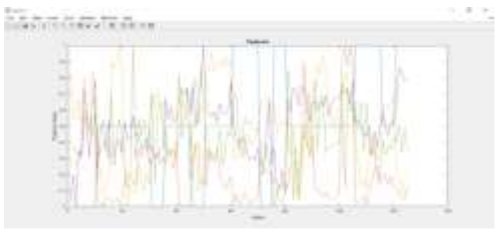


Fig -5: Feature extraction

Each colour indicates each feature the features are flow volume and flow duration the flow volume for the Belkin motion sensor (depicted by green bars) is slightly higher. Flow duration indicates the pink colour.

4.3 Training result in confusion matrix

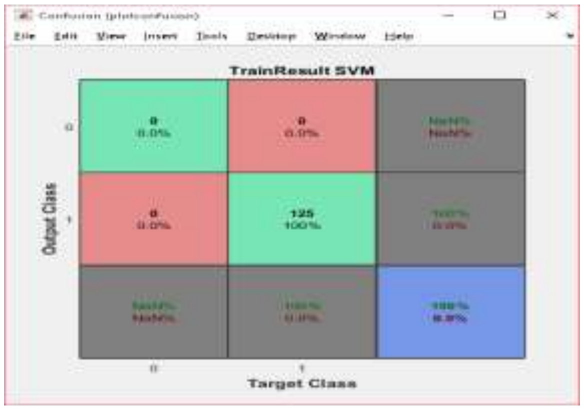


Fig -6: Training results in confusion matrix

The above figure shows that the Support Vector Machine Learning Algorithm that shows the result of both target class results and output class result and finally its gives the train result of SVM.

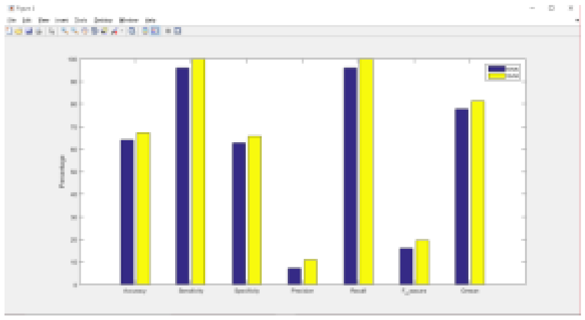


Fig -7: Performance Evaluation for Testing Data

In performance Analysis compares the Existing system with Proposed System. In Existing System use the K-Nearest Neighbor classification and the percentage of accuracy, sensitivity, specificity, precision, recall, $F_m$measure, Gmean is low and in proposed system use the Support Vector Machine Learning Algorithm and the percentage of accuracy, sensitivity, specificity, precision, recall, $F_m$measure, Gmean is High.

5. CONCLUSION

Despite the proliferation of IoT devices in smart homes, enterprises, campuses, and cities around the world, operators of such environments lack visibility into what IoT devices are connected to their networks, what their traffic characteristics are, and whether the devices are functioning appropriately free from security compromises. This work is the first to systematically characterize and classify IoT devices at run-time. We instrumented a smart environment with 28 unique IoT devices and collected traffic traces continuously over 26 weeks. We then statistically characterized the traffic in terms of activity cycles, signalling patterns, communication protocols and cipher suites. We developed a multi-stage machine learning based classification framework that uniquely identifies IoT devices with over 99% accuracy. Finally,

We evaluated our method and compare it with Existing System use the K-Nearest Neighbour and in proposed system use the Support Vector Machine Learning Algorithm and performance is high. the real-time operational cost, speed, and accuracy trade-offs of our classification method. This paper shows that IoT devices can be identified with high accuracy based on their network behaviour, and sets the stage for future work in detecting misbehaviours resulting from security breaches in tech smart environment.

REFERENCES

[1] I. Spectrum. (Last accessed July 2017.) Popular Internet of Things forecast of 50 billion devices by 2020 is outdated. https://goo.gl/ 6wSUkk.

[2] Cisco, "Cisco 2017 Midyear Cyber security Report," Tech. Rep., 2017.

[3] A. Schiffer. (2017) How a fish tank helped hack a casino. https: //goo.gl/SAHxCX.

[4] Ms. Smith. (2017) University attacked by its own vending machines, smart light bulbs & 5,000 IoT devices. https://goo.gl/ cdNJnE.

[5] S. Alexander and R. Droms, "DHCP Options and BOOTP Vendor Extensions," Internet Requests for Comments, RFC Editor, RFC 2132, March 1997. [Online]. Available: https: //tools.ietf.org/rfc/rfc2132.txt

[6] A. Sivanathan et al., "Characterizing and Classifying IoT Traffic in Smart Cities and Campuses," in Proc. IEEE Infocom Workshop on Smart Cities and Urban Computing, Atlanta, USA, May 2017.

[7] S. Notra et al., "An Experimental Study of Security and Privacy Risks with Emerging Household Appliances," in Proc. M2MSec, Oct 2014.

[8] F. Loi et al., "Systematically Evaluating Security and Privacy for Consumer IoT Devices," in Proc. ACMCCS workshop on IoT Security and Privacy (IoT S&P), Texas, USA, Nov 2017.

[9] I. Andrea et al., "Internet of Things: Security vulnerabilities and challenges," in 2015 IEEE Symposium on Computers and Communication (ISCC), July 2015.

[10] K. Moskvitch, "Securing IoT: In your Smart Home and your Connected Enterprise," Engineering Technology, vol. 12, April 2017.