IoT-NetSec: Policy-based IoT Network Security using OpenFlow

Mehdi Nobakht*† Craig Russell† Wen Hu* Aruna Seneviratne‡
*School of Computer Science and Engineering, UNSW Sydney, Australia
*School of Electrical Engineering and Telecommunications, UNSW Sydney, Australia
†Data61 | CSIRO, Sydney, Australia

***Abstract*—The increasingly widespread adoption of the Internet of Things (IoT) has resulted in concerns about IoT security. Recently, there have been proposals to leverage software-defined networking (SDN) to augment IoT device security with network-level measurements. We argue that existing general-purpose security solutions using SDN are impractical for supporting today's home and corporate networks due to the high volume and rates of network traffic, differences in characteristics of IoT systems and computer networks, and limited resources in underlying network switches. To this end we propose IoT-NetSec, a framework that enables policy-based and fine-grained traffic monitoring of the network segments that include only IoT devices. We describe a prototype implementation and its integration with an SDN controller. The prototype implementation and simulations with three network service attacks (port scanning, SYN DoS Flooding and smurf DDoS) demonstrate IoT-NetSec feasibility in a network of real IoT devices.**

I. Introduction

Innovation, efficiency and cost-saving are main motivations in the development of emerging IoT solutions, and surprisingly security is most often last. This is evident by numerous security breaches that have already been reported for IoT solutions [1], [2], [3]. Attack vectors and surfaces in IoT environments are often different compared to traditional computer networks. These differences lie in specific characteristics of IoT systems such as resource-constrained and limited-function devices that most often communicate wirelessly with IoT gateways between such devices and the Internet. Unfortunately, running an extra security software on IoT devices or even patching vulnerabilities by updating their firmwares are not viable solutions [4].

Many security issues stem from ignoring best-security practices which makes IoT systems vulnerable to attacks. Among others, weak passwords, lack of encryption and backdoor accounts are common IoT device attack vectors [5]. These vectors can be exploited to perform various kinds of attacks ranging from device level attacks to network service attacks. Unauthorised remote access to IoT devices in an attempt to seek profit by gaining control of it is a type of device level attacks that has been shown in many reports [6]. Examples of network service attacks include network reconnaissance and heavy hitters such as Distribute Denial of Service (DDoS). A recent example of DDoS attack that occurred in 2016 caused large-scale attacks that used telnet and a weak password vector to compromise IoT devices to be used as a large botnet [1].

There have been many works suggesting to implement security mechanism into entire IoT echosystem of devices, networks, and applications during design and development phases. However, this solution cannot address vulnerabilities in legacy and otherwise unsupported IoT systems which most often unpatchable [4]. Moreover, new attack surfaces make even current robust and secure IoT systems vulnerable in future due to their inability to satisfy security requirement given to their limited computation and power resources. To address these limitations, there is a growing body of research work proposing to implement additional security measures at network-level [7], [8].

Software-defined Networking (SDN), thanks to its programmability and its ability to give visibility into the network, makes it possible to improve network security properties. In the past few years, there have been research work proposing to leverage SDN to orchestrate security measures at a larger scale based on the central view of the network [9], [10]. These prior works either assumed that the network is only consist of IoT devices or do not scale well. However, today's home and enterprise networks most often include a mixture of networked applications, e.g., web browsers and data-hungry apps, and devices such as PCs, tablets, mobile phones and IoT devices.

We argue that it is impractical to incorporate a general-purpose security solution for today's networks using SDN. First, high data volume and rate of today's home and enterprise network traffic imposes a significant burden on security applications on top of SDN controllers [8]. Furthermore, the traffic characteristics of IoT systems such as intermittent connectivity, data usage pattern, and most often low data rates make it difficult to incorporate a general-purpose security solution applicable to both IoT systems and conventional computer network systems. Moreover, required resources for traffic measurement tasks in underlying network switches, e.g., TCAM (Ternary Content-Addressable Memory) counters, are fundamentally limited for power and cost reasons [11].

In the light of these challenges, we suggest to perform network-level security monitoring only for a particular network segment which includes IoT devices; let well-matured security measures for networked computers to be responsible for security threats through conventional computer network systems. This significantly reduces overhead on security applications over SDN controllers, and additionally valuable and limited resources in hardware switches, e.g. TCAM counters, can thus be used more effectively. Beyond that, due to the nature of myriad IoT devices of all kinds and heterogenous architecture of IoT systems, we propose to use a vigilance policy-based security approach that should be created and updated for each class of device. Away form network characteristics of an IoT device, usage pattern and context can also be used to learn

temporal and spatial usage data that can be used in constructing a more flexible and dynamic security policy that adapts to a user's situation.

In this paper, we discuss the design of a security system for IoT networks, called IoT-NetSec. Users of IoT-NetSec can register IoT devices of interest into the system at an SDN controller and attach pertinent security policies to the registered things. IoT-NetSec uses security policies to allow a few remote servers on the Internet which maintained by device manufacturers and legitimate device users to communicate with devices. Additionally, it monitors merely network traffic of IoT devices to detect data traffic rate that exceeds the expected threshold specified in their security policies. IoT-NetSec enforces pre-determined security requirements specified in policies and raises an alarm once traffic rate changes significantly.

IoT-NetSec leverages one important observation that supports its users to construct security policies. That is IoT devices often perform a limited functions because of resource constraints of embedded devices, which makes their network traffic rate pattern predictable. To represent traffic rate patten, we use *traffic volume* as a feature to identify flows exceeding a given threshold. IoT-NetSec first installs associated flow entries into SDN switches and then continuously measures statistics associated with flow entries enabled by OpenFlow capability to count packet and bytes. Before that, the threshold of permitted traffic volume must be determined. This can be done either using device makers' specification or training a model on the data set of normal activities of IoT devices.

II. IoT-NetSec

A. Overview

The IoT-NetSec framework is a software tool running on an SDN controller. It is designed to monitor the traffic of intended IoT devices across the network to ensure that ***i)*** only approved communications are allowed and everything else is denied, and ***ii)*** network attacks (e.g. heavy hitter) are not occurring against IoT devices, and ***iii)*** compromised IoT devices are not carrying out such attack against victim servers on the Internet. Users of IoT-NetSec register IoT devices of interest to the system and attach related device policies to them. Figure 1 shows an overview of the IoT-NetSec workflow and illustrate both user and SDN network interface to IoT-NetSec. A device policy specifies threshold values of incoming and outgoing flow rates for the device, and additionally may specify a flow filter to prevent inappropriate communications with the device. IoT-NetSec blocks access attempts to an IoT device that violates its policy and periodically extracts ongoing flow statistics associated with the device and raises an alarm once the traffic volume exceeds a pre-determined threshold. An IoT-NetSec user can be a Security as a Service (SaaS) provider [12], or a network operator.

The users of IoT-NetSec are required to have a good understanding of expected device behaviour to turn it into the device policy. Network security rules in the device policy defined by packet header fields which specified using five tuples including *source IP address* (SrcIP), *destination IP address* (DstIP), *protocol*, *source port* (SrcPort) and *destination port* (DstPort). For example, a security rule in the device policy

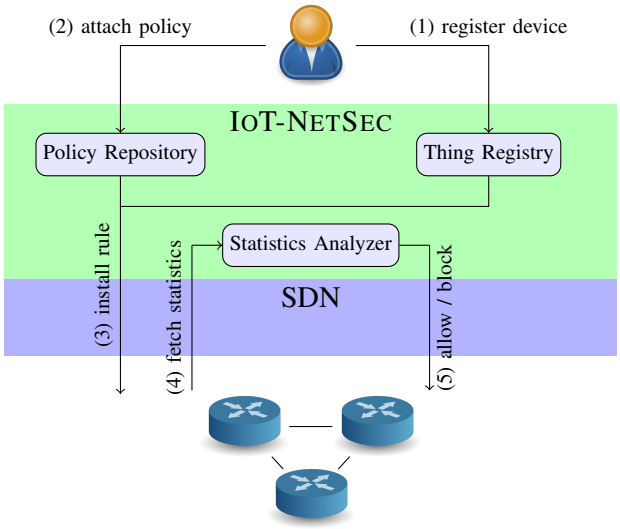


Fig. 1: IoT-NetSec Workflow.

of an IP-based baby monitor allows the device communicates only with its legitimate server with known IP address and port and denies any other requests to establish a TCP connections to/from the baby monitor.

B. Feature Identification

As mentioned above in earlier section, a good understanding of expected device behaviour helps to identify right features. Basically, network features should represent connections' critical aspects. In case of TCP connections, time-based traffic features of TCP connections examine connection behaviour within a time frame and calculate statistics related to protocol behaviour or service.

IoT-NetSec aims to detect network attacks and particularly three common types of network attacks including port scanning, DoS, and DDoS. Basically, features should be chosen so that are relevant to each attack. For example, consider TCP SYN flood attack where an attacker sends repeated SYN packets to the victim device. This will increase the number of packets received at the victim device. Thus, if a TCP SYN flood attack is occurring against a target device, the attack then would be identified by the traffic rate destined to the device; packets with the device IP address as destination address (DstIP) would exceeds the given threshold value specified in the device policy.

C. Architecture

IoT-NetSec's framework is composed of the following components:

1) ***Device Policy Repository****:* This component contains policy documents which are associated for all known IoT devices. A device policy is a JSON document including security policies for the class of devices. It includes device name and type, and a set of network security flow rules to allow an instance of the device to communicate only with its pre-determined controllers and servers, and also to specify threshold values for the device network features. The network

security flow rules are created by users of IoT-NetSec using the device usage description and by training its normal network behaviour.

*2) **IoT Device Registry**:* Using this component, a user of IoT-NetSec registers an IoT device into the system and attaches a related device policy to it. During registration, the user specifies either the IP address of the device or the port number of the SDN switch connected to the device. Attaching a device policy to a device provides necessary information about the device to monitor its network traffic.

*3) **Security Flow-rule Installer**:* This component parses the security rule set in the attached device policy, creates related security flow-rules for traffic monitoring purpose, and installs flow entries in SDN switches. There are two types of monitoring flow entries which IoT-NetSec may create: routing and non-routing flow entries. Routing flow entries are related to TCP/IP protocol stack Level 3 routing, e.g., IPv4 and IPv6. While non-routing flow entries are used to monitor non-routing field like TCP SYN, TCP SYN-ACK and TCP FIN. IoT-NetSec uses routing flow entires to define flow filters to prevent illegitimate access to an IoT device. For example individual flows from an IoT remote controller with specific SrcIP to destination IoT end-point device. Non-routing field instead are used for heavy hitter detection like port scan and TCP SYN flood attacks[1].

Before installing flow entries, IoT-NetSec first ensures whether there are already relevant flow entry rules at switches in the network. In case, there is no related flow entry rule with the same granularity required by the security rule set in the policy, new flow entry rules will be created by this component to provide fine-grained measurements of time-varying traffic at all switches in the network.

*4) **Statistics Collector**:* The component is responsible for collecting packet and byte counts associated with monitoring flow entries in SDN switches in the IoT network. These measurements of packet and byte counts provide fine-grained visibility into network traffic and are considered as network statistical features to be used by the analysis algorithms.

*5) **Statistics Analyzer**:* This component takes counter statistics about flows and matches them against the pre-determined thresholds values defined in the security rule set within the device policy. IoT-NetSec analyses these counter statistics at multiple spatial and temporal scale to achieve a better accuracy. To this end, a small memory has been considered in order to update counter statistics over damped window. In a damped window model, a higher weight is given to recent data than those in the past; the weight of older counter values decrease over time [13].

D. implementation

We have implemented a complete prototype of IoT-NetSec as a software tool to perform measurement tasks. In our prototype, we use Faucet SDN [14][15] which is a compact open source OpenFlow controller, mainly designed for enterprise and campus networks. Faucet has two main OpenFlow controller components; Faucet itself, and Gauge. Faucet controls all forwarding and switch state, and exposes its internal state. Gauge also has an OpenFlow connection to the switch and monitors ports and flow states. Gauge, however, does not ever modify the switch's state, so that switch monitoring functions can be upgraded, restarted, without impacting forwarding.

In order to store timely measurement of time-varying traffic in the IoT network under test, we use Prometheus [16] which is an open-source time-series database. Prometheus fundamentally is a full monitoring system that includes built-in and active scraping, storing, querying, graphing, and alerting based on time series data. Figure 2 shows our implementation interfaces both with Faucet SDN controller and Prometheus time-series data base. The OpenFlow switches and a set of IoT devices also depicted in the figure.

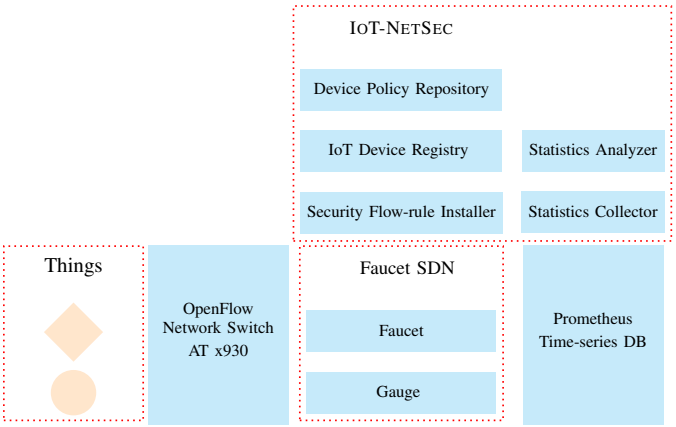


Fig. 2: An illustration of IoT-NetSec's prototype.

III. Evaluation

We use our implemented prototype of IoT-NetSec to evaluate our approach.

A. Experimental Setup

In order to evaluate the IoT-NetSec prototype, we used a modern hardware OpenFlow-enabled switch (Allied Telesis switch x930 series [17]). In our experiments, our IoT-NetSec implementation runs on a dual core 2.4 Ghz Intel Xeon processor connected to the switch through a 1Gbps shared link with ping delay of 0.5 ms.

1) Parameter Settings: We use one second scraping interval for Prometheus; every one second the network measurement tasks are performed and new samples are appended to the corresponding time-series. Additionally, we set the window size that updates counter statistics to three. In this way the most recent counter measurement receive the higher weight than older measurement.

2) IoT Devices: We use a network of smart-home IoT devices to evaluate IoT-NetSec prototype. We have selected three devices among the common smart-home IoT niches. These devices are ***i*)** *Philips Hue connected bulb* and ***ii*)** *Belkin WeMo switch and Motion sensor*. In order to provide some background, we briefly describe them.

[1]From OpenFlow version 1.5, TCP flags matching has been added which allows to match all TCP flags such as SYN, ACK and FIN.

The *Philips Hue connected bulb* is a smart lighting system that can be controlled wirelessly. It is a system of 4 components including ***i***) a series of Hue light bulbs, ***ii***) an App as a way of controlling the lights within the home network, ***iii***) a web-based control panel providing controlling the lights from anywhere in the Internet, and ***iv***) an Ethernet enabled bridge for communication between the App or web portal and the light bulbs. The wired connection of the bridge to the home router provide the communication to the outside world and wireless *Zigbee* protocol is a means of communication between the bridge and the light bulbs. In its architecture model, the bridge acts as server which accepts commands from its users via HTTP protocol.

The *Belking WeMo Switch and Motion Sensor* is a kit of two smart devices that connect to the Internet via WiFi provided interface. WeMo switch is a power socket allows its users to control electrical home appliances power wirelessly and WeMo Motion Sensor enables turning devices on or off as soon as movement is detected. Users can control connected appliances using provided App and can see any activity or the status of appliances. Furthermore, users can configure the kit by inserting rules via the App.

Smart-home IoT devices in our network are connected to the switch ports using a direct wired connection (Philips Hue) and WiFi radio connections (Belking WeMo switch). These devices are connected to the Internet via WAN port of the switch.

3) Attack Simulations: In order to generate attack traffic, we used a tool called *Scapy* [18] to simulate real attacks against our smart home IoT devices installed in our lab. Scapy is a powerful Python library for the purpose of packet manipulation. It is used in this study to initiate flooding and probation attacks. Attacks are launched from a machine in the our home network and connected to one of Open vSwtich (OVS) [19] software switch.

In order to launch TCP SYN Flood attacks, the attack script generates SYN packets with our target IoT devices as destination IP addresses and ports, the attacker machine as source IP address, and a random set of source IP ports. For the spoof-based TCP SYN Flood attacks, the script first identify the active IoT devices within the network and then use their IP addresses to spoof source IP address of attack packets. Thus, the victim IoT device will respond connection requests to other devices in the network rather than the attacker machine.

For Portscan traffic, we constructed packets with source IP address of the attacker machine and random source ports, destination IP address of target IoT device and a set of destination ports. The attacker script sends a TCP SYN packet on each of destination ports and depends on SYN-ACK or RST response infers open ports of target IoT device. Finally, for DDoS attack we launched smurf scenario. For this, the attacker script broadcasts ICMP packets in the network with spoofed source address as IP address of a victim IoT device.

B. Dataset

In order to evaluate the capability of IoT-NetSec in detecting network attacks, several experiments (See Table I) were conducted in a realistic IoT network in our lab. The network consists of three types IoT devices as listed in Section III-A and three PCs. One PC in the network is used to launch the attacks and two others were used for IP spoofing traffic in host pool. We interacted with IoT devices in our experimental network and issued various range of commands to generate legitimate traffic. Next, we launched various network attacks from a machine in our network during training phase. Legitimate traffic and various attacks traffic are mixed in the training dataset.

The legitimate traffic consists of different protocols including TCP, UDP and ICMP. For attack traffic, *Scapy* tool was used to launch network attacks. For training purpose, both legitimate and attack traffic captured and labeled accordingly. We generated 104,253 attack packets and collected 231,133 legitimate packets during the training phase and used `tcpdump` tool to capture traffic packet traces at the interface between the host and the SDN switch. Table I shows the breakdown of the dataset used in our experiment during training and testing phases.

C. Classification

Aiming to detect network attacks against IoT devices in our network, we sought for a predictive model, i.e., a classifier, capable of distinguishing between legitimate access and attack or intrusion. We used the two captured traffic traces (legitimate and attack traffic). Each instance in the training set contains one class label (legitimate or attack) and features. The number of features depends on the window size of damped window mode. For example, in TCP SYN Flood attack, there are four features: the scraping interval that used to fetch count data from time series database and three packet counts associated with IP address of intended device since we set the window size to three (See Section III-A1). The packet counts data includes the number of packets toward the device in the last statistic collection, the second last statistic collection, and the third last statistic collection.

The output of a binary classification is a value of 0 or 1, where the classifier computes the probability that a sample belongs to what class which is a value between 0 and 1. We used *Support Vector Machines* (SVMs) [20], which are a robust technique for non-linear data classification. SVMs use maximum margin kernel to nonlinearly map samples into a higher dimensional space. We considered Radial Bias Function (RBF) kernel (a.k.a Gaussian) that performs well in practice. In using RBF kernel, there are two parameters (C and γ) that must be determined before the learning process. We used a 10-fold cross validation to identify the best parameters and incorporated the LIBSVM library to obtain the optimised classification model.

D. Evaluation Metrics

A classifier performance typically depends on the probability threshold that discriminates between two classes. Thus, setting a threshold has a great effect on the performance of the classifier. In this work, we use negative class to represent benign traffic and positive class to indicate attack traffic.

The detection performance of a classifier is measured using *confusion matrix* which includes True Positives (TP), True Negatives (TN), False Positive (FP), and False Negatives (FN).

TABLE I: Network Attack Datasets for Training and Testing

Traffic Type	Attack Name	# *Training Packets*	*Training time (min.)*	# *Testing Packets*	*Test time (min.)*
Legitimate	Normal traffic	111,131	49	231,872	146
Denial of Service Attack	SYN DoS - Direct	5,064	32	9,342	61
Reconnaissance	Port Scans	18,743	29	39,239	64
Denial of Service Attack	DDoS - Smurf	21,729	28	41,279	58

Although confusion matrix captures all the information about a classifier performance but is not a scalar. Thus, we measure the detection performance of IoT-NetSec prototype using *True Positive Rate* (TPR) metric.

TPR also known as *positive recall* or *sensitivity* indicates the fraction of all malicious packets that IoT-NetSec correctly detected as attack and is computed as:

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

We measure the performance of the prototype using sensitivity metric when the classifier threshold is selected so that the fraction of benign traffic that accidentally classified as malicious is very low (i.e. 0.004) and zero. Such thresholds are set by *False Positive Rate* which is computed as

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

E. Detection Performance

Figure 3a shows TPR of IoT-NetSec prototype for different networking attacks on the network of three IoT devices when the threshold is selected so that FPR is zero. In machine learning detection, it is important that false alarm rates should be minimum to save analysis time to investigate each alarm. To compare the affect of false alarms on detection performance, we performed another experiment and set the classification threshold so that FPR to be very low. The first step of very low FPR in our experiment was 0.004. Similarly, Figure 3b shows TPR of IoT-NetSec prototype when FPR is 0.004.

Note that in our experiment, the widow size was set to three as stated in Section III-A1. Thus, there were four features in our classification algorithm as mentioned in Section III-C. We computed TPR by monitoring traffic for signs of three types of network attacks including SYN DoS Flooding , port scanning, and DDoS. TPR is calculated for each individual attack separately. The measurements were made once the attacks targeting each individual IoT device in our network.

The low FPR demonstrates that our detection algorithm is less likely to label a benign traffic as an attack one. Finally, note that these performance achieved with two parameter settings. There is a trade-off between the detection performance and two parameters including the *window size* that updates counter statistics associated with flow entries and *scraping interval* that time series data base queries the switch. Better detection performance can be achieved by higher window size and lower scraping interval. However, this come at the cost of a larger memory size and computation power required by IoT-NetSec. The users of IoT-NetSec have this flexibility to adjust these parameters according to the requirements of security goals and available resources.

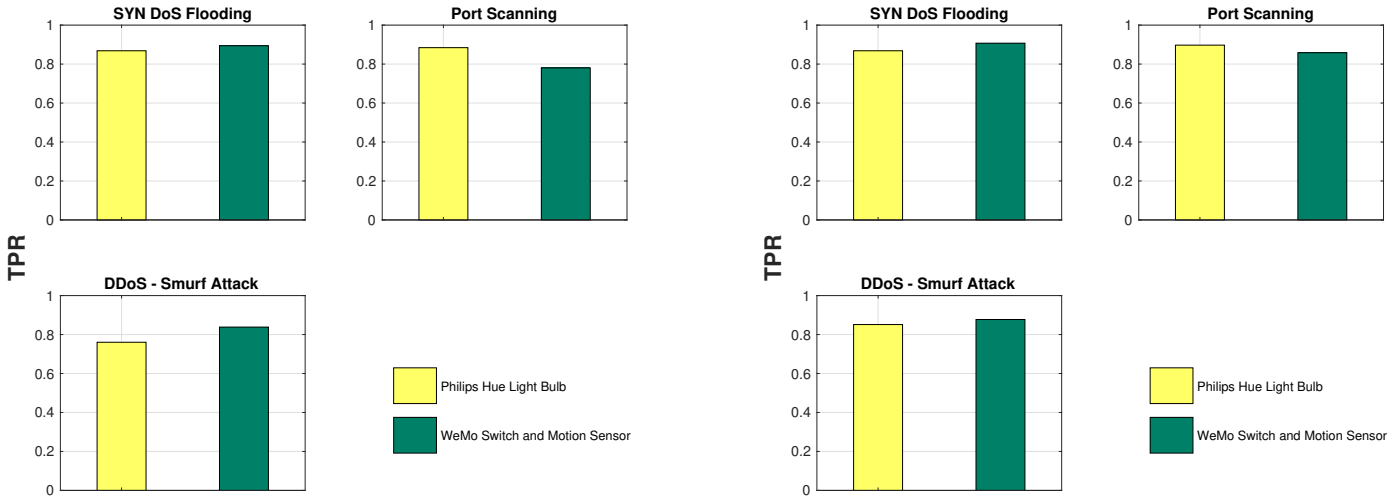
IV. Related work

The SDN architecture with its rich functionalities in traffic monitoring, optimising, and management enables applications on SDN controllers to perform security tasks using dynamic software programs. There is a growing body of research work proposing SDN to meet security challenges in IoT environments. Bull et al. [21] proposed the use of an SDN gateway as a distributed means of monitoring the traffic originating from and directed to IoT based devices. However, their approach is not a viable solution for most of IoT systems as IoT gateways are not typically programable. Shif et al. [22] proposed a SD-VPN architecture to address security and scalability issues of IoT systems using traffic separation provided by VPN and network service chaining based on SDN. Flauzac et al. [23] addressed security policy dissemination across multiple SDN domains using SDN-based IoT architecture. On device availability for IoT, Lee et al. [24] assessed the impact of DoS attacks on IoT gateways through simulation. However, no solution is proposed to mitigate such attacks.

These prior works on using SDN to address security issues, either assumed that the network is only consist of IoT devices or proposed a general-purpose security application to both IoT systems and conventional computer network systems. The work in this paper differs from that above work in two different aspects. First, it performs network-level security monitoring only for a particular network segment which includes only IoT devices and leave the responsibility of security monitoring for networked computers to well-matured security applications for conventional computer network systems. This will reduce a significant load on security applications on SDN controller. Second, it uses a vigilance dynamic policy-based security approach that should be created and updated for each class of devices. In this way, our proposal adapts to the traffic characteristics of IoT devices and also enables security applications to learn temporal and spatial context to provide more accurate detection.

V. Conclusion

IoT promises great benefits and opportunities for individuals and businesses. However, there exists an increasing concerns about its security and privacy. We have presented IoT-NetSec to prevent illegitimate access to IoT devices and protect IoT network to be compromised or to be target of network service attacks such as port scanning, DoS, and DDoS. We described IoT-NetSec's design and its prototype



(a) True Positive Rates at FPR=0

(b) True Positive Rates at FPR=0.004

Fig. 3: Detection Performance.

implementation using the Faucet SDN controller. The feasibility of the prototype demonstrated through conducting extensive experiments in a real smart home IoT network.

REFERENCES

[1] (2016) Hacked Cameras, DVRs Powered Today's Massive Internet Outage. Kerbs on Security. [Online]. Available: https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/

[2] (2017) BrickerBot Author Claims He Bricked Two Million Devices. Bleeping Computer. [Online]. Available: https://www.bleepingcomputer.com/news/security/brickerbot-author-claims-he-bricked-two-million-devices/

[3] M. Nobakht, Y. Sui, A. Seneviratne, and W. Hu, "Permission Analysis of Health and Fitness Apps in IoT Programming Frameworks," in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications*, ser. IEEE TrustCom '18, Aug 2018, pp. 533–538.

[4] (2014) The Internet of Things is wildly insecure and unpatchable. Wired. [Online]. Available: https://www.wired.com/2014/01/theres-no-good-way-to-patch-the-internet-of-things-and-thats-a-huge#-problem/

[5] (2018) IoT Attack Surface Area. OWASP. [Online]. Available: {https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#IoT_Attack_Surface_Areas_Project}

[6] (2016) Search engine lets users find live video of sleeping babies. The Guardian. [Online]. Available: https://www.theguardian.com/technology/2016/jan/25/search-engine-lets-users-find-live-video-of-sleeping-babies

[7] N. Feamster, "Outsourcing Home Network Security," in *Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks*, ser. HomeNets '10, 2010, pp. 37–42.

[8] A. Gupta, R. Birkner, M. Canini, N. Feamster, C. Mac-Stoker, and W. Willinger, "Network Monitoring As a Streaming Analytics Problem," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16, 2016, pp. 106–112.

[9] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *35th IEEE Conference on Local Computer Networks*, ser. LCN, Oct 2010, pp. 408–415.

[10] C. Gonzalez, S. M. Charfadine, O. Flauzac, and F. Nolot, in *International Multidisciplinary Conference on Computer and Energy Science*, ser. SpliTech.

[11] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Dream: Dynamic resource allocation for software-defined measurement," in *ACM Conference of the Special Interest Group on Data Communication*, ser. SIGCOMM '14, 2014, pp. 419–430.

[12] M. Nobakht, V. Sivaraman, and R. Boreli, "A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using OpenFlow," in *11th International Conference on Availability, Reliability and Security*, ser. ARES '16, Aug 2016, pp. 147–156.

[13] N. Jiang and L. Gruenwald, "Research issues in data stream association rule mining," *the Special Interest Group on Management of Data*, vol. 35, no. 1, pp. 14–19, Mar. 2006.

[14] (2018) Faucet Homepage. [Online]. Available: https://faucet.nz/

[15] J. Bailey and S. Stuart, "Faucet: Deploying SDN in the Enterprise," *Queue*, vol. 14, no. 5, pp. 30:54–30:68, Oct. 2016.

[16] (2018) Prometheus Homepage. [Online]. Available: https://prometheus.io/

[17] (2018) Allied Telesis x930 switch. [Online]. Available: https://www.alliedtelesis.com/sites/default/files/ati-x930series-ds.pdf

[18] (2015) Scapy Webpage. [Online]. Available: http://www.secdev.org/projects/scapy/

[19] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending Networking into the Virtualization Layer," in *8th ACM Workshop on Hot Topics in Networks*, ser. HotNets, October 2009.

[20] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92, 1992, pp. 144–152.

[21] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson, "Flow Based Security for IoT Devices Using an SDN Gateway," in *4th IEEE International Conference on Future Internet of Things and Cloud*, ser. FiCloud, Aug 2016, pp. 157–163.

[22] L. Shif, F. Wang, and C. Lung, "Improvement of security and scalability for IoT network using SD-VPN," in *IEEE/IFIP Network Operations and Management Symposium*, ser. NOMS, April 2018, pp. 1–5.

[23] O. Flauzac, C. Gonzalez, A. Hachani, and F. Nolot, "SDN Based Architecture for IoT and Improvement of the Security," in *29th IEEE International Conference on Advanced Information Networking and Applications Workshops*, March 2015, pp. 688–693.

[24] Y. Lee, W. Lee, G. Shin, and K. Kim, "Assessing the Impact of DoS Attacks on IoT Gateway," in *Advanced Multimedia and Ubiquitous Engineering*. Springer Singapore, 2017, pp. 252–257.