Filtering malicious flows for IoT devices in Software-Defined Networks

Woraphop Limpisuk  
*Department of Computer Engineering  
Chulalongkorn University*Bangkok, Thailand  
woraphop.l@student.chula.ac.thKunwadee Sripanidkulchai  
*Department of Computer Engineering  
Chulalongkorn University*Bangkok, Thailand  
kunwadee@cp.eng.chula.ac.th

*Abstract*— Nowadays, many IoT devices are popularly used on household and various places, but due to IoT devices have multiple security vulnerability, causing propagation of malware that targeted to IoT devices and make these devices under controlled for network attacking, so we developed malicious flows filtering system from malicious software, by utilizing machine learning to creating classifier for filtering flows in software-defined network, which could protect various network attacking, saving cost, scalable, and easily upgrade to preventing newer attacks in future .

Keywords— Software defined networking, Internet of things, Machine learning

# Introduction

รูปภาพประกอบด้วย ข้อความ, แผนที่

คำอธิบายที่สร้างโดยอัตโนมัติNowadays, IoT devices are popular on many sections, in household, commercial, and industrial. For increasing convenient and performance. However, IoT systems are insecure for various aspect, according to OWASP IoT Top 10 2018 [1] there are several insecurities, including but not limited to unsecured or hard-coded passwords, Insecure interfaces, lack of system updates, outdated systems, Insufficient privacy, and lack of physical protection. A notable attack involved with IoT devices is Mirai malware, this malware takes over control of devices and executed DDoS attack around 1 Tbps [2] on Dyn, a DNS provider.

To address the problems, there should be detecting and preventing system in the network, a SDN approach is good for fixing problems on traditional approach about scalability, and machine learning is used for increasing detection accuracy.

# Background and Motivation

To preventing attack in network, the firewall or IDS must be used. Each kind of them have different properties and performance for each kind of attacks. Firewall is easy to setup and config, but not flexible for distributed attack. Intrusion detection system (IDS) is better for detecting attack that firewall cannot prevent, there are many kinds of open sourced IDS like Snort, Suricata. However, rule based IDS (e.g. Snort) have a problem about false alarming on default setting, according to studies[3], some kind of manipulated packets could match all of snort rules and degraded overall detection system[4] , so Machine learning is needed to suppressing false alarms[5][6], also, the signature rules need to be regularly update for newer novel attacks.

Another kind of IDS is anomaly flow-based that could fix the problems from signature-based IDS, by anomaly detection, it has advantage about performance, better detection rate, could detect novel attacks. [7], although it also generates false alarms, it can be suppressed with hybrid approach of machine learning. [8]

# System Design

## System archiecture

The proposed system consists of SDN controller. SDN switch, and IoT devices, every equipment are inside private networks as show in Fig. 1 The SDN switch is preconfigured to mirror every packets to SDN controller for generating flow with Argus[9], then detection module will parse Argus flow into flow features, generate additional features and classify flow, when malicious flow is detected, the detection module will edit Access control list of SDN controller entries in order to block attack flow from source IP address as show in Fig. 2.

Fig. 1. Network diagram.

รูปภาพประกอบด้วย ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ

Fig. 2. Internal diagram of SDN controller.

## Faucet SDN Controller

The Faucet SDN controller is configurated by reading configuration file (Faucet.yaml) that describe number of ports and VLANs for each SDN switches, each of ports can be further define for mirroring, tagging VLANs, apply ACLs, etc. In our system, the ACLs file is separated from main configuration file and each time that attack is detected, the system will edit ACLs file and reload SDN controller to apply new configuration. Then the flow table of SDN switch is updated according to the new configuration.

## Testbed

The proposed Testbed are consisting of Linksys WRT54GL with OpenWRT firmware and CPqD OpenFlow 1.3 Software switch module [10] , The SDN controller is Faucet [11], since controller and switch configuration are rely on YAML configuration files, The file is preconfigured to mirror packets to specific interface and will be modified by detection system. The Faucet controller is run on Raspberry Pi with controller interface and Mirroring interface.

According to progress examination, due to firmware problem on OpenWRT router that make OpenFlow switch failed to forward any packets, an alternate testbed is developed. The system will simulate in Graphical Network Simulator-3 or GNS3 with OpenVSwitch as SDN switch and Ubuntu 18.10 Virtual machine as SDN controller, except IoT devices that cannot be simulated. Instead, IoT device that hosted HTTP server is Raspberry Pi connected to simulated network by tunneling Cloud node on GNS3 to physical interface. As shown in Fig. 3

## Dataset

The dataset for training classification model is BoT-IoT Dataset [12], created by UNSW Canberra cyber. This dataset contains Argus [9] flow of normal IoT traffic and simulated botnet attacks traffic. The attack traffic contains DDoS, DoS, OS and Service scan, Keylogging and Data exfiltration attacks, we selected DoS and Service scanning attack for this system. The provided labeled traffic is unbalanced and need to be normalized for training. This dataset also provided 10 best features according to statistical analysis of data by calculating and finding features with minimal correlation coefficient and maximum joint entropy [12].

1. Number of selected bot-iot Attack data

|  |  |  |
| --- | --- | --- |
| **Attack type** | **Normal Traffic** | **Attack traffic** |
| Service scanning data | 2,733 | 1,443,364 |
| TCP DoS data | 608 | 12,315,997 |

1. 10 Best features of Bot-iot Dataset

|  |  |
| --- | --- |
| **Feature Name** | **Meaning** |
| state | Transaction state |
| seq | Argus sequence number |
| mean | Average duration of aggregated records |
| stddev | Standard deviation of aggregated records |
| min | Minimum duration of aggregated records |
| max | Maximum duration of aggregated records |
| srate | Source-to-destination packets per second |
| drate | Destination-to-source packets per second |
| N IN Conn P SrcIP | Number of inbound connections per source IP |
| N IN Conn P DstIP | Number of inbound connections per destination IP |

# Evaluation

## Model Training & Evaluation

The detection model was trained by using feedforward neural network and random forest techniques, we calculated prediction time and accuracy of each techniques, the accuracy is not different for each technique, but neural network classification time is faster, therefore, we select neural network for machine learning approach. The neural network was comprised of 10 neurons for input layer, and 8 hidden layers comprised of 30,40,40,60,80,80,90 neurons for hidden layers and 1 neuron for output layers. Due to unbalanced of labeled data for Bot-IoT dataset, the data was balanced with random oversampling, and normalize with Min-Max transformation, then split 75% of data for training and 25% for testing. After training by these parameters. Each of model is tested for accuracy, precision, and recall, along with confusion matrix.

1. Accuracy, Precision, and recall of tested Service scan and tcp DoS models

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Service Scan** | **TCP DoS** |
| Accuracy | 0.9936 | 1.0000 |
| Precision | 0.9991 | 1.0000 |
| Recall | 0.9880 | 1.0000 |

1. Confusion matrix of tested Service scan Model

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predict | |
| Normal | Attack |
| Actual | Normal | 366388 | 303 |
| Attack | 4362 | 360629 |

1. Confusion matrix of TCP DoS scan Model

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predict | |
| Normal | Attack |
| Actual | Normal | 3078094 | 0 |
| Attack | 101 | 3079804 |

## Real system Evaluation

To evaluate real attacking condition, we started HTTP server on Raspberry Pi and use webterm Docker image [13] as HTTP client for simulating regular IoT traffic, then perform each kind of attacks 5 times and summation the results, and calculated accuracy, precision and recall with summed results. Service scanning attack was performed by Nmap [15] to scan every possible open ports, and DoS attack was performed by hping3 by flooding TCP packets on port 8080, the result accuracy on Service scan model have high accuracy as shown in table VI and VII, but the DoS detection model has low accuracy due to time-dependent feature like N\_IN\_Conn\_P\_SrcIP and N\_IN\_Conn\_P\_DstIP does not generated very well on our testbed, as shown in table VIII. Also, “stddev” feature is not generated correctly on live traffic capture, as it always report as zero, and “seq” feature is monotonically increasing sequence that not related to traffic and not contributing the prediction result, suspected that causing false positive prediction in long run.

1. Accuracy, Precision, and recall of Real traffic Service scan and tcp DoS results

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Service Scan** | **TCP DoS** |
| Accuracy | 0.9541 | 0.4759 |
| Precision | 0.9643 | 0.8721 |
| Recall | 0.9677 | 0.4312 |

1. Confusion matrix of Service scan Model

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predict | |
| Normal | Attack |
| Actual | Normal | 249 | 20 |
| Attack | 18 | 540 |

1. Confusion matrix of TCP DoS scan Model

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predict | |
| Normal | Attack |
| Actual | Normal | 221 | 98 |
| Attack | 881 | 668 |

After training model without 2 features (seq,stddev) and testing with same previous method, both model have slightly lower false positive as shown in table X and accuracy, precision, recall are slightly increasing as shown in table IX. However, the TCP DoS still have considerable amount of false negative as shown in table XI.

1. Accuracy, Precision, and recall of Real traffic Service scan and tcp DoS results without  
    stddev and seq features

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Service Scan** | **TCP DoS** |
| Accuracy | 0.9747 | 0.4194 |
| Precision | 0.9810 | 0.9678 |
| Recall | 0.9872 | 0.3639 |

1. Confusion matrix of Service scan Model  
    Without stddev and seq features

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predict | |
| Normal | Attack |
| Actual | Normal | 265 | 21 |
| Attack | 14 | 1082 |

1. Confusion matrix of TCP DoS Model  
   Without stddev and seq features

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predict | |
| Normal | Attack |
| Actual | Normal | 130 | 15 |
| Attack | 78 | 450 |

## Mitigation time

To measure mitigation time, which is the time that first packet arrive IoT device to the time that last attack packet arrive before get blocked by our system while attack still ongoing, we captured packet flow with Wireshark then calculate recovery time by subtract arrival time of last attack packet and arrival time of first attack packet, we measured 5 times and average the results as shown in table XII.

1. Average of mitigation time

|  |  |
| --- | --- |
|  | Average of Mitigation time (s) |
| Service scanning | 6.5309 |
| TCP DoS | 8.7105 |

# Implications

Our SDN system can filer malicious flow for some extent. But there are many limitations about system such as slow recovery time due to how Faucet controller reload for reading

config files, this could make attack that use short duration time (ex. Commonly use port scanning) are not able to be prevented on time, also the trained model is not reliable due to problem about generating features, some features that original dataset provider suggested that statistically best are not practical to use on real traffic, after eliminate these features does slightly improve model accuracy. The TCP DoS detection model is likely overfitted, so the better dataset and machine learning techniques is needed in order to reduce false prediction. Also, the more robust tools for generating flows and better SDN controller is needed for real-time protection.

# Conclusions

In this project, we developed a network intrusion prevention system by utilizing advantage of SDN for scalability and utilize machine learning for detection accuracy, although this system is more effective than traditional Firewall method or existed signature based IDS method, the system still need to be improved in order to increase accuracy and reliability of system.

##### References

1. https://www.owasp.org/index.php/OWASP\_Internet\_of\_Things\_Project
2. njccic (December 28, 2016). "Mirai Botnet". The New Jersey Cybersecurity and Communications Integration Cell (NJCCIC). Retrieved 28 December 2016.
3. S. Axelsson, “The Base-rate Fallacy and the Difficulty of Intrusion Detection,” ACM Trans. on Information and System Security, 3 (3), 186-205 (August 2000).
4. Patton, Samuel & Yurcik, William & Doss, David. (2001). An Achilles' Heel in Signature-Based IDS: Squealing False Positives in SNORT.
5. Law, Kwok & Kwok, Lam. (2004). IDS false alarm filtering using KNN classifier. 3325. 114-121. 10.1007/978-3-540-31815-6\_10.
6. Shin, MS & Kim, EH & Ryu, Keun Ho. (2004). False alarm classification model for network-based intrusion detection system. 259-265.
7. Mishra, Preeti & Varadharajan, Vijay & Tupakula, Uday & Pilli, Emmanuel. (2018). A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. IEEE Communications Surveys & Tutorials. PP. 1-1. 10.1109/COMST.2018.2847722
8. Goeschel, Kathleen. (2016). Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis. 1-6. 10.1109/SECON.2016.7506774. Ddd
9. Argus (audit record generation and utilization system). https://openargus.org/
10. OpenFlow 1. 3 for OpenWRT, [online] Available: https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow-1.3-for-OpenWRT.
11. faucet 2018. Faucet. (2018). https://faucet.nz/
12. Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, Benjamin Turnbull, “Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset”, https://arxiv.org/abs/1811.00701, 2018.55
13. https://hub.docker.com/r/ehlers/webterm/
14. G. F. Lyon, Nmap network scanning: The official Nmap project guide to network discovery and security scanning, Insecure, 2009. 55
15. hping. http://www.hping.org