# A Botnet Detection System Based on Machine-Learning using Flow-Based Features

Chien-Hau Hung, Hung-Min Sun
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 30013
e-mails: dars2106@gmail.com, hmsun@cs.nthu.edu.tw

*Abstract*—**Botnets have always been a formidable cyber security threat. Internet of Things (IoT) has become an important technique and the number of internet-connected smart devices has been increasing by more than 15% every year. It is for this reason that botnets are growing rapidly. Although the antivirus on Personal Computer (PC) has being applied for a long time, the threats from the botnets still cannot be eliminated. Smart devices and IOT are still in their initial stages, hence there are uncertainties about the security issues. In the foreseeable future, more devices will become victims of botnets. In this paper, we propose a system for detecting potential botnets by analyzing their flows on the Internet. The system classifies similar flow traffic into groups, and then extracts the behavior patterns of each group for machine learning. The system not only can analyze P2P botnets, but also extracts the patterns to application layer and can analyze botnets using HTTP protocols.**

***Keywords- botnet; machine learning; feature selection; J48.***

## I. INTRODUCTION

Victims of botnets, along with smart devices, have grown substantially in number. According to IoT Online Store [1], there are 22.9 billion devices around the globe being connected to the Internet and being used for multiple purposes. The number of smart devices is estimated to be more than 50 billion by 2020. However, smart devices, such as PC, smart phones and other devices are not as safe as we think. They could be infected by malicious software without any abnormal symptoms until they are needed to act as bots. The bots are controlled by a botmaster through Command and Control (C&C) channels using different kinds of communication protocols.

Over the last decade, a lot of research has been done on the detection of different bot families. Most of the research is based on machine learning, of which the performance mainly depends on the features selected for the classifier. Therefore, selecting proper features for the classification model is important. However, there is a trade-off between achieving high detection accuracy and spending huge computation time on constructing a large classification model. On one hand, using all features to build a classification model leads to a significant overhead. On the other hand, using improper or too few features may cause the accuracy rate to decrease.

*Motivation.* As the number of botnet attacks has been increasing [2], it is very difficult to find devices without any vulnerability, not to mention the fact that common users do not patch their devices on time. Hence, there is a need for a botnet detection system to verify if the botnets are within the devices.

*Our Contribution***.** We develop a system that can classify the botnets' flow by using machine learning. Users can input the pcap file and automatically generate the report. For the classification model of each botnet family, we select the appropriate sets of features.

In this paper, Section II describes different kinds of botnets and the machine learning technique we apply to detect botnets. In Section III, we discuss several ways to detect botnet. In Sections IV and V, we explain our framework and implementation in detail. Section VI shows the evaluation of our work. Conclusions are given in the last section.

## II. BACKGROUND

### A. Botnet

The word botnet is a combination of the words robot and network. A botnet consists of a botmaster, bots and usually a C&C server. Botnets can be used to perform various kinds of attacks, e.g., to launch a Distributed Denial of Service Attack (DDoS), to steal data, to send spam, and to function as a backdoor.

**Client-Server Botnet.** Botnets were originally constructed and operated using a Client-Server model. The infected clients connect to an infected server awaiting commands from the botmaster. Once the botmaster sends commands to the infected server, each client retrieves and executes those commands and reports back their results of actions to the infected server.

**P2P Botnet.** The problem with client-server botnets is the single point of failure. Therefore, to avoid this issue, new botnets fully operate over Peer-to-Peer (P2P) networks, where each peer acts simultaneously both as a client and as a server. Despite this kind of structure, which operates without a centralized point that makes it hard to be blocked by IP address, botnets are still blockable by ports. Therefore, a combination of HTTP and P2P botnet is used, called HTTP2P botnet, which uses HTTP as the communication protocol and often employs port 80, a method that makes it impossible to be blocked by ports.

**Internet of Things with Botnet.** With the booming of IoT and promotion of IPv6, there is an increase in the number of Internet-Connected devices. In [3], Y.Feng gives botnets a lots of potential bots to infect, which could result in large scale botnets. Also, the P2P network topology is more sophisticated which makes it easier for botmasters to hide and larger attacks might happen.

### *B. Machine Learning*

There are three major classifications of machine learning algorithm: supervised, unsupervised and semi-supervised. Waikato Environment for Knowledge Analysis (WEKA) [6], which was developed by University of Wakato in New Zealand, is a Java language tool with a collection of machine learning algorithms for data mining.

**Flow-Based.** Open Systems Interconnection (OSI) model [4] defined network architecture into seven layers, with different protocols. The network layer uses protocols like Internet Protocol (IP) [4], Internet Control Message Protocol (ICMP) [4], Internet Group Management Protocol (IGMP) [4] and Internetwork Packet Exchange (IPX) [4]. The transport layer's protocols are TCP, UDP, and svice port addressing. A flow is a network connection with five properties: Source IP, Destination IP, Source Port, Destination Port and Protocol.

As a classifier, we use the J48 Decision Tree [5]. In classification, a classifier uses features to build up a model and classifies the inputs into groups in accordance with their respective features. Among the various classifiers, the decision tree classifier [5], which as the name implies is built in a tree shape, serves as one of the major algorithms. Features of the input data go through the nodes, which are rules of the tree, when fitted, until they reach the leaves of the tree, where the classification is done. While being an open source, J48 is implemented by JAVA based on C4.5 decision tree algorithm.

**C4.5 Algorithm** [6]. We employ the C4.5 algorithm, developed by Ross Quinlan, to generate a decision tree for the purpose of classification. It became popular after being ranked No.1 in a paper entitled "Top 10 Algorithms in Data Mining" published by Springer Lecture Notes in Computer Science (LNCS) in 2008 [1]. C4.5, which is written based on the Iterative Dichotomiser 3 (ID3) algorithm, uses a training data sets to build a decision tree in a similar way as ID3, except C4.5 utilizes the concept of gain ratio to overcome the problem of biased information entropy. The attribute of the maximum gain ratio is selected as the node to split the tree. The detailed process is explained below.

**Feature Selection.** The purpose of feature selection is to reduce the number of features. This reduces the training time of the learning model and helps with over-fitting problems. We use Consistency Subset Evaluation as our features selection algorithm. It is a greedy algorithm, which will try 77*(total number of features)$^5$, each time randomly choosing a subset from the total feature set, then calculating the inconsistency. Finally, result will be one set of features with the smallest inconsistency [8].

## III. RELATED WORK

The authors of [9] utilize the flow-based features of existing botnets and select 21 features from 16 botnet families for machine learning. The outcome of the average detection rate is 75%. Q. Yan et al. [10] present a system named PeerClean that detects P2P botnets in real time only by using features extracted from higher layer of OSI network model in the C&C network flow traffic. T. Cai and F. Zou [11] present some features of HTTP Botnet and design a new method for detection. In [12], F. V. Alejandre detects botnets with machine learning algorithms and use genetic method to select features of botnets.

## IV. SYSTEM ARCHITECTURE AND DESIGN

### *A. Goals*

We propose a system to analyze and predict botnets' behavior by using machine learning tools: WEKA. Furthermore, we hope to provide a system that is feasible, expansible and user-friendly, easy to implement for system managers.

### *B. System Framework*

Figure 1 shows the overall system flow, which consists of five parts, namely, "Input Pcap file", "Preprocess", "Machine learning", "Trained Model" and "Report".

1. Record the network flow and save it into pcap format, in this step we can use some tools (e.g., wireshark) to record packet over the network interface. Then input it to the preprocess program.
2. The preprocess program accepts the pcap file then process it into features described in section 4.3. The outcome result of each pcap file will be two arff format files. Then input them into machine learning tools (e.g., WEKA).
3. Supply training sets and test sets and use selected features to build a well performed model.
4. Input a Well-trained model to predict the files from process (B).
5. The output prediction from process (D) along with other model's result then form a prediction report.

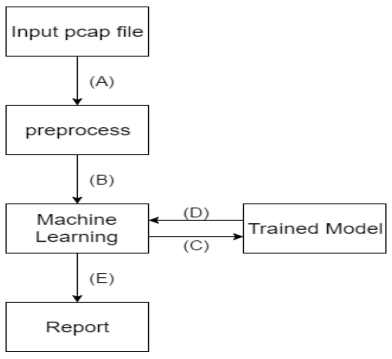


Figure. 1. System Flow

### *C. Features*

There are many flow-based features, from which our system selects some. In this section, we will describe some features output from our system.

- Source and Destination port: This is a feature that is often used in detection of botnet traffic, especially port 80 for HTTP and port 443 for HTTPS. These two ports are often used by normal users. Therefore, it is likely that HTTP-based botnets hide their traffic inside.
- Protocol: these features are often used in classifying different traffic. It would look strange when some seldom used protocols appeared in traffic. For example, botnets using UDP protocol will stand out in the network traffic because normal users do not use UDP protocol to communicate. Instead, they often use HTTP or HTTPS, which are using TCP in the transport layer.
- Duration: It means the total time of the connection from beginning to the end. It has been used a lot in detecting potential botnets. It may vary depending on different kind of botnets, but certain types of botnets are known to be chatty. This feature may be useful for some types of botnets.
- First packet length (FPS): First packet length is the length of first packet transferred in the connection, in some situations is similar to duration feature. First packet transferred in the flow reveals some characteristics, which may be useful in detecting specific botnets.
- Flow size features
  - Total number of bytes (TBT) is the total number of bytes transferred and received in the flow. This feature is used to get similarities out of botnet traffic, such as fixed length commands.
  - Average payload packet length (APL) is the average payload of all packet in the flow. This feature is used to get similarities out of botnet traffic.
  - Total number of packets with the same length over the total number of packets (DPL) is the number of same size packets in the flow. This feature is used to get similarities out of botnet traffic.
  - Standard deviation of payload packet length (PV) is the standard deviation of every packets' payload in the flow. This feature is used to get similarities out of botnet traffic.

The reason for using these features is based on the assumption that the traffic generated by bots is more uniform than traffic generated by normal users. For instance, if botnets use fixed length commands, using these features to detect them is more feasible.

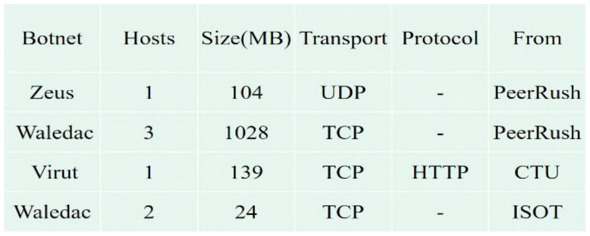
- Ratio between the number of incoming packets over the number of outgoing packets (IOPR): Many studies suggest that there should be some difference between the input and output traffic for different kind of protocols. Although there is no evidence indicating any relation between the feature and botnet behavior, this feature still gives ratio between incoming and outgoing traffic for detection of some potential botnet behavior.
- Packet exchange: There is an assumption that botmaster needs to manage all bots, so to keep their connections alive communicating with each bot is necessary. Number of packets exchanged might be useful to identify this behavior.
- Reconnect: In botnet detection, it is common to analyze communication and other features with accurate captured flows. Hence, a simple strategy to prevent detection is by randomly reconnecting as an established connection. Therefore, this feature can be controlled by setting up a specific time window to detect reconnection.
- Number and percentage of small/null packets exchanged: It is widely known that botnets use small packets to maintain communication between bots and C&C servers. However, recent researches haven't seen botnets using null packets. Despite of that, small or null packets were tested in recent researches.
- The features below are used to get similarities in botnets' traffic.
  - average bits-per-second (BS) is the average transferred and received bits per second.
  - average packets-per-second(PPS) is the average transferred and received packets per second.
  - average inter arrival time of packets (AIT) is the average inter arrival time of packets received.
- HTTP method: It is a part of header from HTTP protocol, which indicates the desired action to be performed for a given resource. Botnets use HTTP method [13] to disguise their communication flows among other HTTP flows. However, botnets might still be detectable if a combination of HTTP method and other features is employed.
- HTTP request: HTTP request is spited into three parts: total added weight, length weight and average weight. These three features together can be used to detect potential botnets if botnets use fixed length commands.

## V. IMPLEMENTATION

### *A. Data Set*

To build and test the detect module, we prepared 3 families of botnet data set: Zeus, Waledac and Virut. The size of each data set is 104MB, 1024MB and 138.77MB. Figure 2 depicts the properties of these three families. PeerRush [14] published their journal and data about detecting P2P botnet. Czech technical university [15] made their records of botnet behavior pcap file public. The fourth botnet data is the testing data from ISOT. This data set is only used to test our model for comparison.

To simulate real world traffic, we record three users' behavior and collect over 120GB data. There are many kinds of different behavior, like using different browsers for various activities. Types of application also vary, e.g., games, communication, SSH and so on.



| Botnet | Hosts | Size(MB) | Transport | Protocol | From |
|---|---|---|---|---|---|
| Zeus | 1 | 104 | UDP | - | PeerRush |
| Waledac | 3 | 1028 | TCP | - | PeerRush |
| Virut | 1 | 139 | TCP | HTTP | CTU |
| Waledac | 2 | 24 | TCP | - | ISOT |

Figure. 2. Botnet data set properties

## B. *Preprocess*

**Parsing packets.** Firstly, we read every packet from the stored pcap file, then we separate each layer, from data link layer to transport layer, and we also took a look at the application layer if the bot uses HTTP protocol. After parsing step, we retrieve twelve properties from packets.

**Construct Flow and Conversation.** Secondly, we use the flow based analysis. However, we believe flow based features are not enough to distinguish bots from normal users, hence we set a threshold of 2000 seconds to split conversation. After that, we calculate more properties from previous step and get two lists at the end: list of flow and list of conversation. The content blocks of both lists contain nineteen properties.

**Calculate Features for Machine Learning.** Finally, we employ the thirty-one properties from the above two steps to build up features for machine learning tools [16]. Lots of computation is involved in this step, and numpy library was very helpful. For the testing purpose, we add some noise to payload, inter arrival time and features related to these two features to make sure the model performs well.

- source port
- destination port
- protocol
- total number of packets exchanged
- number of null packets exchanged
- number of small packets exchanged
- percentage of small packets exchanged
- ratio between the number of incoming packets over the number of outgoing packets
- number of re-connection
- flow duration
- length of the first packet
- total number of bytes
- average payload
- average payload sent
- average payload received
- total number of packets with the same length over the total number of packets
- standard deviation of payload packet length
- average bits-per-second
- average inter arrival time of packets
- average inter arrival time of packets sent
- average inter arrival time of packets received
- average packets-per-second
- median inter arrival time of packets
- median inter arrival time of packets sent
- median inter arrival time of packets received
- variance packet size
- variance packet size sent
- variance packet size received
- max packet size
- HTTP method
- HTTP uniform resource locator (URL) total weight
- HTTP uniform resource locator (URL) length
- HTTP uniform resource locator (URL) average weight

**Noise Algorithm.** To verify if our model is robust enough, we added some noise on the features. The add noise algorithm [10] formula is shown in figure 3, in which X represents the feature to which noise is going to be added. 'Var' is the integer randomly chosen in the range of -1 to 1. Noise stands for the scale of noise going to be added. In this case, we adopt the noise at half to one-third the scale of the original data. Noise will be added in payload, inter arrival time and features related to these two, e.g., median inter arrival time of packets, average inter arrival time of packets, average payload and other similar features.

$$X' = X + Var*Noise*X$$

X = feature need to add noise

Var = random [-1,1]

Noise = random [25,33] / 100

Figure. 3. Add noise algorithm

## VI. EVALUATION

### A. *Experimental Design*

**Purpose.** Our basic purpose is to analyze the behavior of each family of botnets. That is, when botnets infect any Internet-connected devices, they should be detected through recording every packet over the Internet. Our system can further identify which device might have been infected.

**Experimental Process.** This process aims to find normal data affection to the model we built. Firstly, reduce the instances of normal data to about 50% to make the characteristics of infected data more obvious. Secondly, increase the percentage of normal data over infected data to make it closer to the real situation. Finally, we add different level noise into testing data sets in payload and inter arrival time and some other related features to evaluate the model.

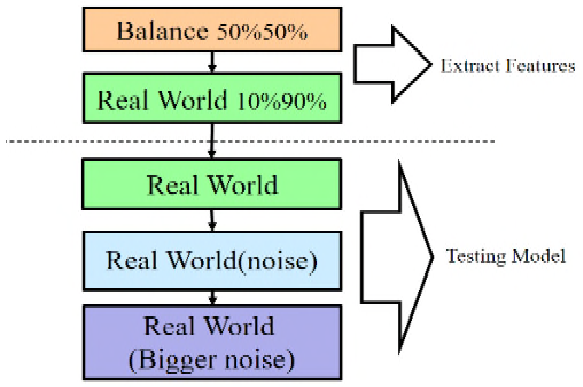


Figure. 4. Work flow of experiment process

## B. Evaluation

Figure 5 shows the data sets for each step. To avoid obvious characteristics, we remove destination and source ports before selecting features to build J48 tree.

| | Balance | | Real World | | | | Real World (noise) | | Real World (bigger noise) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Feature selection | | Feature selection + Train | | Test | | Test | | Test | |
| | Infected | Healthy | Infected | Healthy | Infected | Healthy | Infected | Healthy | Infected | Healthy |
| Zeus | 6,020 | 6,906 | 6,020 | 42,761 | 6,098 | 45,619 | 6,098 | 45,619 | 6,098 | 45,619 |
| Virut | 1,717 | 1,732 | 1,717 | 29,196 | 186 | 29,593 | 186 | 29,593 | 186 | 29,593 |
| Waledac | 20,437 | 21,323 | 8,276 | 59,301 | 9,957 | 71,135 | 9,957 | 71,135 | 9,957 | 71,135 |

Figure. 5. Data set of each process

Figure 6 shows the feature selecting result while evaluating Balance data set, we choose features scoring over 8 points as the feature set. As the results show, normal data and virut using HTTP protocol, so the HTTP feature of virut got high score. Figure 7 shows the feature selecting result while evaluating Real World data set. We also choose features scoring over 8 points as the feature set. As shown by the result, with the increase of normal data, the HTTP feature of every botnet family got higher score.

| | protocol | PX | NNP | PSP | IOPR | Reconnect | Duration | FPS | TBT | avg_payload | avg_payload_sent | avg_payload_received | DPL | PV | BS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Zeus | 4 | 10 | 0 | 10 | 5 | 0 | 6 | 9 | 10 | 0 | 0 | 0 | 0 | 0 | 1 |
| Virut | 5 | 10 | 0 | 7 | 9 | 10 | 0 | 1 | 9 | 10 | 0 | 10 | 0 | 3 | 0 |
| Waledac | 10 | 6 | 0 | 10 | 2 | 10 | 9 | 0 | 10 | 3 | 8 | 10 | 0 | 0 | 0 |
| | **avg_iat** | **avg_iat_sent** | **avg_iat_received** | **PPS** | **median_iat** | **median_iat_sent** | **median_iat_received** | **var_pkt_size** | **var_pkt_size_sent** | **var_pkt_size_received** | **max_pkt_size** | **HTTP_method** | **URL_total_weight** | **URL_len** | **URL_average_weight** |
| Zeus | 10 | 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 6 |
| Virut | 0 | 0 | 0 | 0 | 10 | 6 | 2 | 3 | 0 | 0 | 10 | 7 | 2 | 9 | 8 |
| Waledac | 0 | 0 | 2 | 0 | 5 | 10 | 0 | 2 | 0 | 0 | 8 | 0 | 0 | 0 | 4 |

Figure. 6. Balance feature selection

| | protocol | PX | NNP | PSP | IOPR | Reconnect | Duration | FPS | TBT | avg_payload | avg_payload_sent | avg_payload_received | DPL | PV | BS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Zeus | 1 | 10 | 0 | 10 | 6 | 0 | 10 | 9 | 10 | 0 | 0 | 0 | 0 | 0 | 1 |
| Virut | 2 | 10 | 0 | 4 | 9 | 10 | 0 | 1 | 10 | 7 | 0 | 10 | 0 | 2 | 0 |
| Waledac | 10 | 5 | 0 | 10 | 3 | 10 | 9 | 0 | 10 | 3 | 8 | 6 | 0 | 1 | 0 |
| | **avg_iat** | **avg_iat_sent** | **avg_iat_received** | **PPS** | **median_iat** | **median_iat_sent** | **median_iat_received** | **var_pkt_size** | **var_pkt_size_sent** | **var_pkt_size_received** | **max_pkt_size** | **HTTP_method** | **URL_total_weight** | **URL_len** | **URL_average_weight** |
| Zeus | 10 | 8 | 5 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 4 | 1 | 10 | 5 | 9 |
| Virut | 0 | 0 | 0 | 0 | 10 | 10 | 4 | 5 | 0 | 0 | 10 | 9 | 3 | 10 | 9 |
| Waledac | 0 | 0 | 2 | 0 | 5 | 9 | 0 | 0 | 0 | 0 | 10 | 0 | 9 | 3 | 10 |

Figure. 7. Real world feature selection

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Zeus** | PX | PSP | Duration | FPS | TBT | avg_iat | avg_iat_sent | URL_total_weight | URL_average_weight | | | |
| **Virut** | PX | IOPR | Reconnect | TBT | avg_payload | avg_payload_received | median_iat | median_iat_sent | max_pkt_size | HTTP_method | URL_len | URL_average_weight |
| **Waledac** | protocol | PSP | Reconnect | Duration | TBT | avg_payload_sent | avg_payload_received | median_iat_sent | max_pkt_size | URL_total_weight | URL_average_weight | |

Figure. 8. Feature selection for each family

Figure 8 shows the three models and the features they are built upon to perform well in the simulation.

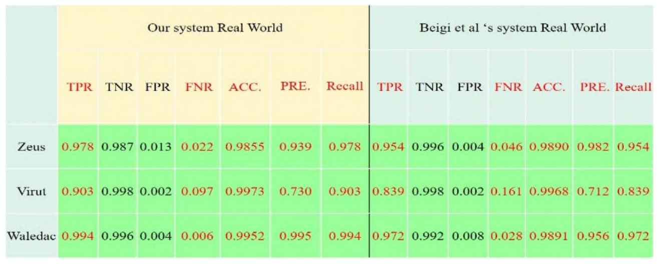
| | Real World | | | | | | |
|---|---|---|---|---|---|---|---|
| | TPR | TNR | FPR | FNR | Accuracy | Precision | Recall |
| Zeus | 0.978 | 0.987 | 0.013 | 0.022 | 0.9855 | 0.939 | 0.978 |
| Virut | 0.903 | 0.998 | 0.002 | 0.097 | 0.9973 | 0.730 | 0.903 |
| Waledac | 0.994 | 0.996 | 0.004 | 0.006 | 0.9952 | 0.995 | 0.994 |
| | Real World (noise 25~33%) | | | | | | |
| Zeus | 0.971 | 0.987 | 0.013 | 0.029 | 0.9843 | 0.939 | 0.971 |
| Virut | 0.892 | 0.998 | 0.002 | 0.108 | 0.9972 | 0.728 | 0.892 |
| Waledac | 0.960 | 0.996 | 0.004 | 0.040 | 0.9804 | 0.994 | 0.960 |
| | Real World (bigger noise 33~50%) | | | | | | |
| Zeus | 0.963 | 0.987 | 0.013 | 0.037 | 0.9829 | 0.938 | 0.963 |
| Virut | 0.872 | 0.998 | 0.002 | 0.128 | 0.9972 | 0.726 | 0.872 |
| Waledac | 0.945 | 0.996 | 0.004 | 0.055 | 0.9735 | 0.994 | 0.945 |

Figure. 9. Result of each botnet family with different noise level

While evaluating each model we built, we add noise in test data set. First, testing with pure data Zeus and Waledac perform well, although Virut only performs 90% but it is still a good model. Second, we add 25% to 33% noise in test data set. Adding noise causes the TPR of our model to decrease, but overall the result is acceptable. Finally, we add a bigger noise range from 33% to 50% in the test data set. As we can see from Figure 9, the TPR of our model decrease a bit, but it is still a good result. We think it is because of using a higher level features.

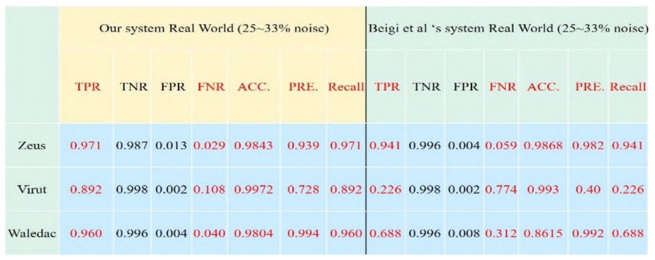
## C. Comparison

We implement the system in [9] and use the same data set that we test on our system. Figure 10 shows the tested result of our work and the system in [9].



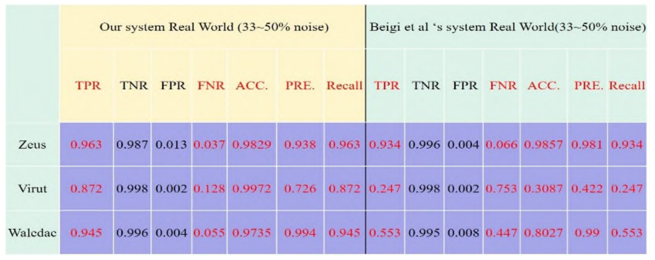
| | Our system Real World | | | | | | | Beigi et al 's system Real World | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPR | TNR | FPR | FNR | ACC. | PRE. | Recall | TPR | TNR | FPR | FNR | ACC. | PRE. | Recall |
| Zeus | 0.978 | 0.987 | 0.013 | 0.022 | 0.9855 | 0.939 | 0.978 | 0.954 | 0.996 | 0.004 | 0.046 | 0.9890 | 0.982 | 0.954 |
| Virut | 0.903 | 0.998 | 0.002 | 0.097 | 0.9973 | 0.730 | 0.903 | 0.839 | 0.998 | 0.002 | 0.161 | 0.9968 | 0.712 | 0.839 |
| Waledac | 0.994 | 0.996 | 0.004 | 0.006 | 0.9952 | 0.995 | 0.994 | 0.972 | 0.992 | 0.008 | 0.028 | 0.9891 | 0.956 | 0.972 |

Figure. 10. Comparison without adding noise



| | Our system Real World (25~33% noise) | | | | | | | Beigi et al 's system Real World (25~33% noise) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPR | TNR | FPR | FNR | ACC. | PRE. | Recall | TPR | TNR | FPR | FNR | ACC. | PRE. | Recall |
| Zeus | 0.971 | 0.987 | 0.013 | 0.029 | 0.9843 | 0.939 | 0.971 | 0.941 | 0.996 | 0.004 | 0.059 | 0.9868 | 0.982 | 0.941 |
| Virut | 0.892 | 0.998 | 0.002 | 0.108 | 0.9972 | 0.728 | 0.892 | 0.226 | 0.998 | 0.002 | 0.774 | 0.993 | 0.40 | 0.226 |
| Waledac | 0.960 | 0.996 | 0.004 | 0.040 | 0.9804 | 0.994 | 0.960 | 0.688 | 0.996 | 0.008 | 0.312 | 0.8615 | 0.992 | 0.688 |

Figure. 11. Comparison after adding 25% to 33% noise



| | Our system Real World (33~50% noise) | | | | | | | Beigi et al 's system Real World(33~50% noise) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPR | TNR | FPR | FNR | ACC. | PRE. | Recall | TPR | TNR | FPR | FNR | ACC. | PRE. | Recall |
| Zeus | 0.963 | 0.987 | 0.013 | 0.037 | 0.9829 | 0.938 | 0.963 | 0.934 | 0.996 | 0.004 | 0.066 | 0.9857 | 0.981 | 0.934 |
| Virut | 0.872 | 0.998 | 0.002 | 0.128 | 0.9972 | 0.726 | 0.872 | 0.247 | 0.998 | 0.002 | 0.753 | 0.3087 | 0.422 | 0.247 |
| Waledac | 0.945 | 0.996 | 0.004 | 0.055 | 0.9735 | 0.994 | 0.945 | 0.553 | 0.995 | 0.008 | 0.447 | 0.8027 | 0.99 | 0.553 |

Figure. 12. Comparison after adding 33% to 50% noise

As shown in Figures 11 and Figure 12, our model can resist more noise compared to the system in [9]. Under this context, we add features to higher layer like HTTP features and it indeed helps in separating healthy data from infected data since normal users use HTTP and HTTPS more often nowadays.

## VII. CONCLUSION

We propose a system for detecting potential infected bots by using machine learning and flow based detecting techniques. As the result shows, our model can clearly recognize normal users from all packets. On top of that, we retrieve features from data link layer to app layer. Although botnets do not necessarily employ HTTP, HTTP features, at least it could help to learn normal users' behavior and thus improve our accuracy rate to higher than the average accuracy rate in the paper "Towards Effective Feature Selection in Machine Learning-Based Botnet Detection Approaches" [9]. System managers can easily use our system by simply recording the network flow into pcap format, and our system will process it into machine learning format and output the results in a report.

## ACKNOWLEDGMENT

This research was supported in part by the Ministry of Science and Technology, Taiwan, under the Grant MOST 107-2218-E-007-030.

## REFERENCES

[1] "IoT Online Store's report of IoT device number," http://www.iotonlinestore.com/. [Dec., 2016]

[2] "Highest botnet flow increasing by year," http://www.ithome.com.tw/news/111220. [Jan., 2017]

[3] Y. Feng, "How to fight against botnets in IoT," http://staff.cs.kyushu-u.ac.jp/data/event/2016/02/160107_Yaokai_Feng.pdf. [Feb., 2016]

[4] "ISO-OSI-layer-model-tcpip-model," http://programmerhelp404.blogspot.tw/2014/01/iso-osi-layer-model-tcpip-model.html. [Jan., 2014]

[5] "WEKA classifiers trees j48," http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html.

[6] " C4.5 algorithm," https://en.wikipedia.org/wiki/C4.5_algorithm. [Jul., 2018]

[7] X. Wu et al., "Top 10 algorithms in data mining," Knowledge and information systems, vol. 14, no. 1, pp. 1–37, 2008.

[8] H. Liu, R. Setiono, "A probabilistic approach to feature selection-a filter solution," in ICML, vol. 96, 1996, pp. 319–327.

[9] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in Communications and Network Security (CNS), 2014 IEEE Conference on. IEEE, 2014, pp. 247–255.

[10] Q. Yan, Y. Zheng, T. Jiang, W. Lou, and Y. T. Hou, "Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis," in Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015, pp. 316–324.

[11] T. Cai and F. Zou, "Detecting http botnet with clustering network traffic," in School of Information Security Engineering Shanghai Jiao Tong University, 2012, pp. 1–6.

[12] F. V. Alejandre, N. C. Cortés, and E. A. Anaya, "Feature selection to detect botnets using machine learning algorithms," in Electronics, Communications and Computers (CONIELE- COMP), 2017 International Conference on. IEEE, 2017, pp. 1–7.

[13] "Hypertext transfer protocol," https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Jul., 2018]

[14] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: mining for unwanted p2p traffic," in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2013, pp. 62–82.

[15] C. R. CTU University, "The ctu-13 dataset. a labeled dataset with botnet, normal and background traffic," 2013.

[16] E. Alparslan, A. Karahoca, and D. Karahoca, "Botnet detection: Enhancing analysis by using data mining techniques," in Advances in Data Mining Knowledge Discovery and Applications. InTech, 2012.