Research Article


Cheng Huang, Jiaxuan Han, Xing Zhang, and Jiayong Liu

1 College of Cybersecurity, Sichuan University, Chengdu 610065, China
2 NSFOCUS, Beijing 100089, China

Correspondence should be addressed to Jiayong Liu; jylscu@gmail.com

Received 28 April 2019; Revised 14 July 2019; Accepted 23 August 2019; Published 22 September 2019

Copyright © 2019 Cheng Huang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Traditional security strategies are powerless when facing novel attacks in the complex network environment, such as advanced persistent threat (APT). Compared with traditional security detection strategies, the honeypot system, especially on the Internet of things research area, is intended to be attacked and automatically monitor potential attacks by analyzing network packages or log files. The researcher can extract exactly threat actor tactics, techniques, and procedures from these data and then generate more effective defense strategies. But for normal security researchers, it is an urgent topic how to improve the honeypot mechanism which could not be recognized by attackers, and silently capture their behaviors. So, they need awesome intelligent techniques to automatically check remotely whether the server runs honeypot service or not. As the rapid progress in honeypot detection using machine learning technologies, the paper proposed a new automatic identification model based on random forest algorithm with three group features: application-layer feature, network-layer feature, and other system-layer feature. The experiment datasets are collected from public known platforms and designed to prove the effectiveness of the proposed model. The experiment results showed that the presented model achieved a high area under curve (AUC) value with 0.93 (area under the receiver operating characteristic curve), which is better than other machine learning algorithms.

1. Introduction

Many traditional security detection strategies such as firewall or intrusion detection systems (IDS) have been invented to protect the system’s security, but there are still many critical issues which are reported every day [1, 2]. The situation has become more complex with the development of Internet technologies. Honeypots are defined as a new deception technology of cybersecurity defense in recent years [3], which can be used to strengthen the company’s security detection level. In other words, honeypots are used to lure attackers into interacting with them and collect the information which will be used to analyze and study the attack way of attackers [4–6].

Depending on the interaction level of honeypots, they are divided into three categories: low-interaction honeypots, medium-interaction honeypots, and high-interaction honeypots [3, 7–9]. Low-interaction honeypots and medium-interaction honeypots will not provide any physical environment for attackers. Thus, they only run a few sets of web services. Especially for low-interaction honeypots, services that are simulated through port listening could be detected easily and remotely. But for high-interaction honeypots, the services are fully deployed with native programs [10].

With the widespread use of deception technology, honeypot detection methods also need improvement, which means researchers need to modify their honeypots more realistic to cheat potential attackers instead of easily distinguished. Thus, many researchers start to focus on how to automatically detect honeypots before publishing their honeypot framework [11–14]. The authors [15] discussed several honeypot detection features by fingerprinting. Holz and Raynal [16] proposed other methods based on detecting User-mode Linux or VMWare system. With the rapid development of honeypots, the flaws of previous detection
methods are exposed gradually. The main problem is that the simulation of high-interaction honeypots is too similar to real systems, making it difficult to detect with a single feature. Therefore, other researchers start to extract other features and introduce the latest algorithms [17–19]. But there are still many shortcomings; at present, this paper explores honeypot detection technology to improve the effectiveness of deception technologies, which is helpful to promote the current development of honeypot technology. The contributions of this paper are summarized as follows:

(i) Based on the analysis of the existing honeypot detection technology, this paper proposed three group features, which can distinguish ordinary servers and honeypot servers remotely and accurately. These features could be summarized as application-layer, network-layer, and other system-layer.

(ii) The paper presented an automatic detection model using multiple features and machine learning algorithms to identify honeypots. In order to get higher accuracy, four machine learning algorithms are introduced to build separate models.

(iii) In the interest of highlighting the effectiveness of the proposed model, the paper compared receiver operating characteristic (ROC) curves of four machine learning algorithms based on 10-fold cross validation methods. According to the experiment result, the model achieved a higher detection rate with the AUC value of 0.93.

The construction of this paper is as follows. The first section introduces the issue. Section 2 is the related work which presents current research progress in honeypot detection. Section 3 is the proposed framework, and it describes the detail of our system. Section 4 is the experiment and evaluation result. Section 5 is the conclusion and future work.

2. Related Works

With the development of Internet technology, honeypots play a significant role in network security. However, the attackers could easily distinguish whether the server has deployed honeypot services or not. In response to solve such a threat, researchers need to make their honeypot service more realistic, and improve the inner mechanism and outer interface of the honeypot framework. Recently, many researchers start to focus on how to automatically detect a honeypot server [20–24].

For example, Fu et al. [18] proposed a method based on examining the link latency of honeyd honeypots. Authors said that because the stimulative accuracy of the link latency of honeypots such as honeyd is always 1 ms or 10 ms, the link latency in some virtual network is around a multiple of 1 ms or 10 ms. According to this characteristic, they designed a method based on Neyman–Pearson decision theory and achieved a high detecting rate. Wenda and Ning [25] suggested that people can determine honeypots by analyzing the network monitoring characteristic or detecting the virtual environment. Because many high-interactive honeypots are always deployed with firewalls and IDS, such features could be used to detect quickly. Defibaugh-Chavez et al. [17] proposed another method which focused on detecting network-level activities and services of honeypots. Then, Mukkamala et al. [19] proposed another method based on the SVM algorithm [26]. They added a different feature group, TCP/IP fingerprint, and designed an experiment to prove their method effective. The result proved their idea was correct. Except for these methods, Holz and Raynal [16] proposed a method based on the feedback information of honeypots. They presented a honeypot detection framework by extracting data of User-mode Linux and VMWare [27]. For UML, people could use the information of command “dmesg” in fingerprinting or view the file /etc/fstab. For VMWare, they suggested capturing the hardware and MAC address. According to IEEE Standards, a MAC address such as “00-0E-23-xx-xx-xx” is assigned for VMWare. Besides, Send-Safe Honeypot Hunter is the first commercial anti-honeypot technology [28].

Although many detection methods have been proposed, there is still a big problem with the rapid development of cloud and virtual technologies, many of them may not be as effective as they used to be. So, we studied previous researchers’ methods on honeypot detection and then proposed a novel honeypot detection framework based on machine learning technologies.

3. Framework

3.1. Architecture. In order to detect honeypots more effectively, the paper presents an automatic identification framework, which is shown in Figure 1. There are three important parts in the framework: feature extraction, label methods, and detection model. Feature extraction is primarily responsible for collecting features from the target server. In this part, all features are split into three group features: the application layer, the network layer, and the system layer. These extracted feature values will be calculated with string or integer meaning. The second important part is the label methods. As the name suggests, the purpose of this part is to collect label data which contain honeypots or normal host IP addresses. Cyberspace search engine such as Shodan, FOFA, and other public platforms are introduced to collect honeypots servers. Manual check methods are used after crawling real data from these platforms via the application programming interface (API). After obtaining the feature data and label data, both of them will be transferred to the detection method as the training data. Each data has a corresponding label data in training steps. More in detail, each data record in the training data can be described as (feature data and label data). In the detection steps, the training data will be loaded first, and then the machine learning algorithms including random forest, SVM, kNN, and Naive_Bayes will be used to train the data for building machine learning models. The best model will be selected to do parameter adjusting. The honeypot detection model will be used to predict the results.
3.2. Feature Extraction. Good features are the basis of training an excellent machine learning model. In this paper, features were divided into three groups: application-layer feature, network-layer feature, and other system-layer feature.

3.2.1. Application-Layer Feature. According to Veena’s investigation [29], HTTP, FTP, and SMTP services are three common protocols that attackers most attack. The result shows such simulated services could cheat majority of hackers. Moreover, low-interaction honeypots do not implement a complete service, which could be easily recognized by simple interaction commands [17]. For example, both HTTP-GET and HTTP-OPTIONS are provided in a real system, but only HTTP-GET method is available in the honeypot. In addition, referring to “service exercising” of Defibaugh-Chavez’s research [17] and the complexity of honeypots services proposed in [28], the application-layer feature is summarized in Table 1.

3.2.2. Network-Layer Feature. This TCP/IP fingerprints can help people recognize different devices, including honeypots. Some common TCP/IP fingerprints are TCP FIN flag, TCP initial window, and ACK value [30]. After analysis, we finally decided to use average_received_bytes, average_received_ttl, average_received_ack_flags, average_received_push_flags, average_received_syn_flags, average_received_fin_flags, and average_received_window_size as the network-layer feature. The network-layer feature is extracted as shown in Table 2. Each value of network-layer feature is computed from received packets by a special equation. For example, the average_received_ttl comes from the following equation:

$$\text{average}_{-}\text{received}_{-}\text{ttl} = \frac{\text{received}_{-}\text{ttl1} + \text{received}_{-}\text{ttl2} + \text{received}_{-}\text{ttl3}}{3}. \quad (1)$$

In this equation, the value of received_ttl1 is obtained from the packet received time from port 21 of the remote server. The value of received_ttl2 is calculated from the packet received time from port 25 of the remote server. The value of received_ttl3 is obtained from the packet returned time from port 80 of the remote server.

<table>
<thead>
<tr>
<th>No</th>
<th>Feature name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HTTP_GET</td>
</tr>
<tr>
<td>2</td>
<td>HTTP_OPTIONS</td>
</tr>
<tr>
<td>3</td>
<td>HTTP_HEAD</td>
</tr>
<tr>
<td>4</td>
<td>HTTP_TRACE</td>
</tr>
<tr>
<td>5</td>
<td>FTP_USER</td>
</tr>
<tr>
<td>6</td>
<td>FTP_PASS</td>
</tr>
<tr>
<td>7</td>
<td>FTP_MODE</td>
</tr>
<tr>
<td>8</td>
<td>FTP_RETR</td>
</tr>
<tr>
<td>9</td>
<td>SMTP_HELO</td>
</tr>
<tr>
<td>10</td>
<td>SMTP_MAIL</td>
</tr>
<tr>
<td>11</td>
<td>SMTP_DATA</td>
</tr>
<tr>
<td>12</td>
<td>SMTP_VRFY</td>
</tr>
<tr>
<td>13</td>
<td>SMTP_ETRN</td>
</tr>
</tbody>
</table>

Table 2: Network-layer feature.

<table>
<thead>
<tr>
<th>No</th>
<th>Feature name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>average_received_bytes</td>
</tr>
<tr>
<td>2</td>
<td>average_received_ttl</td>
</tr>
<tr>
<td>3</td>
<td>average_received_ack_flags</td>
</tr>
<tr>
<td>4</td>
<td>average_received_push_flags</td>
</tr>
<tr>
<td>5</td>
<td>average_received_syn_flags</td>
</tr>
<tr>
<td>6</td>
<td>average_received_fin_flags</td>
</tr>
<tr>
<td>7</td>
<td>Average_received_window_size</td>
</tr>
</tbody>
</table>

Figure 1: The framework of proposed automatic identification methods.
3.2.3. Other System-Layer Feature. Table 3 shows the features extracted in the other system-layer feature. They are port1, port2, port3, System_Fingerprint, and ICMP_ECHO_response_time.

(1) Port1, Port2, and Port3. Since low-interactive honeypots only simulate a special service, the ports open on them are different from real systems. Therefore, we chose the port number as one feature in the other system-layer feature. The meanings and values of port1, port2, and port3 are listed below:

(i) Port1 indicates whether the port 21 of target is listening. If port 21 is open, the value of port1 is 21. Otherwise, the value is −1.

(ii) Port2 indicates whether the port 25 of target is listening. If port 25 is open, the value of port1 is 25. Otherwise, the value is −2.

(iii) Port3 indicates whether the port 80 of target is listening. If port 80 is open, the value of port1 is 80. Otherwise, the value is −3.

(2) System_Fingerprint. It means the version of operating system running a honeypot, such as “Microsoft Windows Server 2016” or “Ubuntu 16.04.” According to the manual check result, it indicates that many honeypots have the same System_Fingerprint, so it is considered to be a separate feature that distinguishes honeypots from real systems.

(3) ICMP_ECHO_response_time. Mukkamala et al. [19] claim that it is common to run several virtual honeypots on a single machine. So, when handling ICMP ECHO request, honeypots are always slower than real systems. Utilizing this characteristic, ICMP_ECHO_response_time was extracted for the other system-layer feature.

3.3. Classification Model. After extracting three group features, it needs to consider choosing an effective and accurate algorithm and constructing an intelligent model. Ensemble learning is a method of accomplishing the task by structuring and combining several learners [31]. Firstly, we need to explain an important concept named hypotheses in the ensemble learning area. Hypotheses is the collection of functions which can be described as the collection of $h_1, h_2, \ldots, h_n$. As for traditional machine learning algorithms, their purpose is to find the most appropriate classifier $h_k$ for $f$ (a function describing the relationship between input $X$ and label $y$). But ensemble learning is different; all $h$ in hypotheses will be combined by a special fashion (such as voting) to decide the label of input $X$. Specifically, the process of ensemble learning can be described as two steps. One is to generate some individual learners. Another is to combine these learners by a special fashion. By combining individual learners, ensemble learning can achieve more superior generalization performance than single learning. Random forest is the most representative algorithm in ensemble learning.

The definition of random forest is “A random forest is a classifier consisting of a collection of tree-structured classifier $\{ h(X, \Theta_k), k = 1, \ldots \}$, where the $\{ \Theta_k \}$ are independent identically distributed random vectors, and each tree casts a unit vote for the most popular class at input $X$” [32]. In fact, random forest is the collection of decision trees. Each decision tree is a classifier which gives a predicted result. So there are $n$ results from different $n$ classifiers. According to the principle of random forest algorithm, the final result will be decided by the majorities of all results [33]. Random forest has the ability to give a more accurate result, and it can also handle thousands of input variables without variable deletion [34]; this is the best choice for the proposed machine learning model.

3.4. Model Evaluation. Model evaluation is a significant part of model training. The purpose of model evaluation is to assess the performance of final model. In this paper, we introduced recall, precision, F1-score (F1), ROC curve, and AUC metrics to evaluate the proposed model. Their definitions are as follows:

\[
\text{recall} = \frac{TP}{TP + FN},
\]

\[
\text{precision} = \frac{TP}{TP + FP},
\]

\[
F1 = \frac{2 \times TP}{n + TP - TN},
\]

\[
\text{AUC} = \frac{1}{2} \sum_{i=1}^{n-1} (x_{i+1} - x_i) \cdot (y_{i+1} + y_i),
\]

where $n$ is the number of samples.

True positive (TP) is the number of honeypot servers classified as honeypot servers. True negative (TN) is the number of normal servers classified as normal servers. False positive (FP) is the number of normal servers classified as honeypot servers. False negative (FN) is the number of honeypot servers classified as normal servers. Recall rate is the ratio of all samples correctly classified as honeypot servers to all honeypot servers. Precision rate is the ratio of all samples that are correctly classified as honeypot servers to all samples that are predicted to be honeypot servers. F1-score is a comprehensive measure of recall and precision.

ROC means receiver operating characteristic. It is constructed by true positive rate (TPR) and false positive rate (FPR). TPR is the ratio of all samples correctly classified as honeypot servers to all honeypot servers. FPR is the ratio of the sample incorrectly predicted to be honeypot servers to all the normal servers. In the coordinates of a
ROC curve, X-axis is FPR, and Y-axis is TPR. TPR and FPR are defined as follows:

\[
\text{TPR} = \frac{TP}{TP + FN},
\]
\[
\text{FPR} = \frac{FP}{TN + FP}.
\]

Each (FPR, TPR) is a point on a ROC curve. If a ROC curve is above the diagonal line between (0, 0) and (1, 1), it means the classification performance of the model is acceptable. If a ROC curve is below this line, there may be some mistakes in the label of the dataset. If a ROC curve is near to this line, the classification performance of the model is poor. In other words, the model could predict near to this line, the classification performance of the model is acceptable. If a ROC curve is below this line, there may be some mistakes in the label of the dataset. If a ROC curve is above the diagonal line between (0, 0) and (1, 1), it means the classification performance of the model is acceptable.

4. Experiment

In order to make the predicted result more authentic and accurate, the experiment focuses on the real honeypot servers and normal servers based on remote scanning technologies. In the experiment, scikit-learn library is used to accomplish machine learning model training. The calculation procedure ran on the computer with i7-7700HQ CPU, 16 GB of memory, and GTX 1050ti GPU.

4.1. Dataset. Shodan and Fofa are well-known cyberspace search engines which are used to search for network devices. Many known honeypot servers have been marked by these two platforms, so we can firstly get some IP addresses which have been marked as honeypots. We also randomly select many IP addresses from the Internet. Secondly, we manually check all IP addresses to determine whether they are honeypots or not. Then, we scan all IP addresses via raw socket technology to collect the feature data. Finally, we collect 2413 IP records as the experiment dataset. The number of honeypot IP addresses is 807. The number of real systems IP address is 1606. The dataset is shown in Table 4.

4.2. Experiment Design. In order to improve the accuracy of the proposed model, this paper introduced three steps in the detail experiment: dimension unification, parameter adjustment, and cross validation.

4.2.1. Dimension Unifying. In some cases, some large dimension features will affect the final result when training a model. For example, there are three records in Table 5. The dimension of feature 2 is larger than the dimension of feature 1. So, when training a model, the large dimension feature may dominate the predicted result. Therefore, the label of the 4th record may be predicted as 0. However, the truth is 1. That is why dimension unifying is needed. The paper chose popular normalization method to achieve dimension unifying. The equation of normalization is as follows:

\[
y_i = \frac{x_i - \bar{x}}{\delta},
\]
\[
\bar{x} = \frac{1}{k} \sum_{i=1}^{k} x_i,
\]
\[
\delta = \sqrt{\frac{1}{k-1} \sum_{i=1}^{k} (x_i - \bar{x})^2}.
\]

\{x_1, x_2, \ldots, x_k\} is a raw data record. Each value \(x_i\) will be transformed into another value \(y_i\) by the equation. Then, the new data record \(\{y_1, y_2, \ldots, y_k\}\) is composed of all values \(y_i\). Finally, the standard deviation of the new data record is 0, and the variance is 1. In this way, the problem that large dimension features affect the training result is solved.

4.2.2. Parameters Adjusting. Choosing the most suitable parameters for the model is a necessary part of the model training. There are two important parameters in random forest algorithm which will determine the accuracy of the model. One is \(\text{n_estimators}\). Another is \(\text{max_depth}\). \(\text{n_estimators}\) is the number of decision trees and \(\text{max_depth}\) is the maximum allowable depth for each tree. If \(\text{n_estimators}\) is too big, the result may be overfitting. However, if \(\text{n_estimators}\) is too small, the result may be underfitting. So, a suitable value of \(\text{n_estimators}\) is important for the predicted accuracy of the final model. \{100, 120, 200, 300, 500, 800, 1200\} was prepared for \(\text{n_estimators}\), and \{5, 8, 15, 25, 30, 100\} was prepared for \(\text{max_depth}\). We tested each of these two collections separately to choose the most appropriate value for these two parameters.

4.2.3. Cross Validation. We introduce 10-fold cross validation methods to avoid overfitting issues. Cross validation is a common method of model evaluation [35]. The cross validation structure of this paper is shown in Figure 2. The cross validation structure is composed of three parts. Firstly, the data \(D\) was divided into ten similar mutually exclusive subsets in the data split \(D\) was the collection: \(D_1 \cup D_2 \cup \ldots \cup D_{10}, D_i \cap D_j = \phi\). Then, the divided data

<table>
<thead>
<tr>
<th>Data type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honeypot records</td>
<td>807</td>
</tr>
<tr>
<td>Real system records</td>
<td>1606</td>
</tr>
</tbody>
</table>
could be transferred to training and testing. In the training and testing steps, nine subsets were used for training every time, and the last one was used for validation. So, after the training and testing, there were ten classifiers. Each classifier would give a predicted result. The last result was computed from these ten results.

4.2.4. Comparing Experiment. The result without comparison is not convincing. So, we designed this comparing experiment to highlight the advantages of the method in this paper. Reviewing some researches about machine learning, we chose SVM, kNN, and Naive_Bayes as the counterparts of random forest. Then, these four machine learning algorithms would be used to train four models on the same dataset, respectively. Finally, for each model, we plotted their respective ROC curves in the same coordinate system. According to these ROC curves, the performance of models is clear at a glance.

4.3. Experiment Result. This section shows the result of the experiment. Firstly, we tested the values in the two parameter collections, respectively. The effect of different values of “n_estimators” on the accuracy is shown in Figure 3. The effect of different values of “max_depth” on the accuracy is shown in Figure 4. The blue line represents the change in the training data, and the green line represents the change in the testing data.

Analyzing these two figures, when “n_estimators” was 200, the accuracy of the blue line and the green line were the highest. So, the value of “n_estimators” was decided to use 200. When “max_depth” was 30, the accuracy of blue line and green line were the highest. So, we chose 30 to be the value of “max_depth.”

Table 6 is the recall, precision, F1-score, and accuracy of the final model. Figure 5 is the ROC curves of the four algorithms. All of them were concluded after the 10-fold cross validation. In Figure 5, the blue line is the ROC curve of the random forest model, the green line is the ROC curve of the SVM model, the yellow line is the ROC curve of the kNN model, and the purple line is the ROC curve of the Naive_Bayes model. Analyzing the figure results, it can be concluded that the performance of Naive_Bayes is the worst. However, there is little difference in the AUC values between random forest, SVM, and kNN. But the performance of random forest is better. In Table 6, the recall rate is 0.82, the precision rate is 0.82, the F1-score is 0.82, and the accuracy is 0.90. The ROC curve of random forest and these indexes
prove the final model has a high detection rate and a good ability of generalization.

5. Conclusion
As an active cybersecurity defense strategy, the emergence of honeypots further reduces the attacker’s activity space. Although honeypot technology is constantly improving, attackers are constantly searching for weaknesses in honeypots to identify them. There is an old saying in China: the authorities are fascinated, bystanders clear. For honeypot researchers, in order to promote the development of honeypot technology, it is necessary to learn and discover the method of identifying honeypot remotely.

Based on the research of early honeypot and honeypot detection technology, this paper proposed a honeypot detection technology based on machine learning algorithms. The technology used machine learning technology to achieve accurate honeypot detection through extraction and collection different layer honeypot features. The experimental result proved that honeypots are still insufficient in the degree of simulating real systems, such as the integrity of the simulated service. At the same time, the experimental result provided a reference for the improvement of honeypots and promoted the development of honeypot technology.

Data Availability
The experiment data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
The authors of this work were supported by the CCF-NSFOCUS KunPeng Research Fund (2018008), the Sichuan University Postdoc Research Foundation (19XJ0002), and the Frontier Science and Technology Innovation Projects of National Key Research and Development Program (2019QY1405). The authors would like to thank Baimaohui Security Research Institute for providing some experiment data.

References


