PRATD: A Phased Remote Access Trojan Detection Method with Double-Sided Features

Chun Guo 1,*,†, Zihua Song 1,2,†, Yuan Ping 3,*, Guowei Shen 1, Yuhei Cui 1 and Chaohui Jiang 1

1 Guizhou Provincial Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; sc_zhsong@163.com (Z.S.); gwshen@gzu.edu.cn (G.S.); yhcui@gzu.edu.cn (Y.C.); jiangchaohui@126.com (C.J.)
2 College of Cybersecurity, Sichuan University, Chengdu 610065, China
3 School of Information Engineering, Xuchang University, Xuchang 461000, China; pyuan.lhn@xcu.edu.cn
* Correspondence: cguo@gzu.edu.cn
† These authors contributed equally to this work.

Received: 19 October 2020; Accepted: 9 November 2020; Published: 11 November 2020

Abstract: Remote Access Trojan (RAT) is one of the most terrible security threats that organizations face today. At present, two major RAT detection methods are host-based and network-based detection methods. To complement one another’s strengths, this article proposes a phased RATs detection method by combining double-side features (PRATD). In PRATD, both host-side and network-side features are combined to build detection models, which is conducive to distinguishing the RATs from benign programs because that the RATs not only generate traffic on the network but also leave traces on the host at run time. Besides, PRATD trains two different detection models for the two runtime states of RATs for improving the True Positive Rate (TPR). The experiments on the network and host records collected from five kinds of benign programs and 20 famous RATs show that PRATD can effectively detect RATs, it can achieve a TPR as high as 93.609% with a False Positive Rate (FPR) as low as 0.407% for the known RATs, a TPR 81.928% and FPR 0.185% for the unknown RATs, which suggests it is a competitive candidate for RAT detection.

Keywords: Remote Access Trojan; malware detection; feature extraction; network-based detection; host-based detection

1. Introduction

In recent years, Advanced Persistent Threat (APT) [1] has become the most serious cyber attack. It steals confidential information or undermines the information system from a particular organization or company. As a kind of high-latency, high-hidden, high-harm malware, Trojan plays an indispensable role in the APT attacks. A Trojan called Remote Access Trojan (RAT) that is often used by APT attackers, which can give them interactive access to a victim’s computer and steal confidential data [2,3]. RAT is often embedded in the system by email attachment, USB memory, file bundle, or in combination with Zero-day vulnerability penetration. Not only is it hard for ordinary users to detect implanted RAT, but also it is difficult for administrators to find such malware.

Given the fearful security threat of RAT, a growing number of research focused on proposing an efficient RAT detection method to alleviate this damage. Existing RAT detection methods can be divided into two categories [4]: host-based detection methods and network-based detection methods. The host-based method that employs static analysis is largely based on syntactic signatures or semantic features. However, the method based on static analysis alone might not be sufficient to identify RAT [5]. The host-based detection method that employs dynamic analysis is a good complement to static analysis method, but this type of methods have difficulty in detecting RAT in its keep-alive
state. Network-based detection methods usually identify RATs by checking the payload of network traffic, analyzing the statistical characteristics of network traffic, or analyzing the behaviour of network traffic. For the method of analyzing the payload of network traffic needs to establish a characteristic storehouse of the RATs’ network traffic in advance and thus this kind of methods cannot detect the unknown RAT. In addition, for the detection methods of analyzing the communication behaviour of RATs, they can detect some unknown RATs due to the fact that they pay attention to the different communication behaviours between the RATs and benign programs. However, unfortunately, many benign programs such as P2P software have some similar characteristics as RAT in the aspect of network communication.

To complement one another’s strengths, the hybrid method combining host-based and network-based detections is feasible. Based on this idea, we propose a phased RATs detection method with double-sides features (PRATD). In PRATD, network traffic and host operation records of a program are separately extracted at the same time, and then the extracted double-sides (host-side and network-side) features are combined into a feature vector. Besides, a detection model for each the two different running states of RAT, i.e., keep-alive and command & control states, are trained respectively. In particular, PRATD neither collects the packet’s payload transmitted by the program nor monitors the concrete application data used by the program, i.e., it does not threaten the privacy of the user.

The main contributions of this article can be summarized as follows:

1. We combine both the host-side and network-side features to detect RATs. Since RAT performs operations on the network-side or host-side throughout its running process, this article extracts some features from the network-side and host-side to cover RAT’s network and host behaviours, which is conducive to distinguishing the RATs from benign programs.

2. We propose a phased method named PRATD for detecting RAT based on the double-side features and the two major states of RAT. RAT in the different stages has different behaviour characteristics, so we train the detection model for each of the two major states (i.e., keep-alive and command & control) to improve the detection performance for RAT.

3. We implement a prototype system and evaluate PRATD on different kinds of benign programs and RATs. The experimental results show that the proposed PRATD is able to obtain a good detection performance, for example, the obtained True Positive Rate (TPR) and False Positive Rate (FPR) of PRATD for known RATs is 93.609% and 0.407% respectively when AdaBoost is used to train its two detection models.

The article is organized as follows. Section 2 summarized the related works. In Section 3, we analyzed the runtime status of the RATs. Section 4 described the used features of PRATD and introduced the details of the method. In Section 5, we gave a description of the experimental data and the experimental results. Section 6 discussed our results. Finally, Section 7 concluded our article.

2. Related Works

As a kind of malware with great harmful, RAT has received the researchers’ extensive concern. Farinholt et al. [6] and Rezaeirad et al. [3] tries to reveal the operators and procedures of RAT. However, they only concentrate on analysing one or two RAT families and do not propose any detection method. To detect Trojan, many detection methods have been proposed in the past few decades. Based on the objects to be monitored, these methods can be divided into two categories, i.e., host-based and network-based methods. For the host-based method, two types of technologies are currently used, namely, static and dynamic analysis technologies. Static technology analysis of the executable program before this program is executed. The most widely used static technology is the traditional signature-based technique [7], but it can easily be evaded by obfuscation techniques [8]. Instead of using only the signatures, some researchers detect Trojan by employing the opcode sequences of the executable program. In work [9], opcodes and API function names are used to classify malware. Some methods based on malware images and deep leaning [10,11] can be also classified to the
static analysis methods. They convert malware binaries into images and then use deep learning for malware classification. In general, static detection method has a fast detection speed, but it is difficult to analyze the malware that uses code obfuscation techniques (e.g., UPX, PEX, ASPack and FSG) [12]. As for dynamic analysis technology, it analyzes the behaviours of a program during it is running. In recent years, the information collected during the program is running, such as API call sequence [13], system logs [8], CPU usage [14], process behaviours [15] and system provenance graph [16], is used in some dynamic analysis methods. For example, Yang et al. [8] propose an instrumentation-free RAT forensic system that can reconstruct RAT attack by using the system logs on Windows platform. However, most dynamic detection methods can achieve high TPR only when RAT has performed a lot of operational behaviours.

A growing number of people pay more attention to the network-based RAT detection method. The most widely used detection technique is the Deep Packet Inspection (DPI) [17], it distinguishes between benign programs and malware by analyzing whether the payload of each network packet has sensitive information. This technique can effectively identify abnormal traffic and has been deployed to many Network Intrusion Detection Systems (NIDS). However, this traditional technique not only needs to maintain the detection rules continuously but also has to detect the content of payload [18], so it often has a high False Negative Rate (FNR) and threatens user’s privacy. To overcome the limitation of DPI technique, many researchers focus on the communication characteristics of RATs and detect RATs based on different network features [4,19–21]. In [19], the authors pay attention to the early stage of RAT communication and propose a method to detect RAT by analyzing the behaviours of RAT in the early stage. Xie et al. [21] extract features from packet-level and flow-level to form two feature vectors, and then build a hybrid structure neural network model based on deep learning to detect HTTP-based Trojan. To improve the detection result, Pallaprolu et al. [22] detect RAT based on the voting result of three different classifiers. It can be found that the research on network-based RATs detection is focused on analyzing the statistical characteristics of network traffic of RAT. However, some benign software has similar network behaviour as RATs, especially P2P software. Therefore, the network-based RAT detection methods usually have the problem of a high FPR.

Each of host-based and network-based methods has a distinct analysis object, and each has distinct advantages and disadvantages. Therefore, a method that integrates host and network data can benefit from both of their strengths. In the domain of bot/botnet detection, a few methods that detect bot/botnet based on multi-source were proposed recently. Zeng et al. [23] proposed a botnet detection method that combines host-level and network-level information for the first time. This method detects the behaviours of bots on the host-side and network-side respectively. When its network-side analyzer detects an abnormality, this method triggers the correlation engine to correlate the detection result of the host-side and network-side, then obtains the final detection result. This method aims at the problem of botnet-detection and needs to be based on the assumption that it is the existence of similarities among bots, so this method is not good for RAT detection. Shin et al. [24] proposed a host-network cooperated framework for bot malware detection. This method correlates information from the different host-level and network-level aspects and performs heavy monitoring only when necessary, one of its core modules mainly assumes that bots will use DNS to contact their master, thus this method is also not suitable for RAT detection. Kalpika and Vasudevan [25] proposed a system that consists of folder monitoring, network traffic monitoring and API hook monitoring for detecting Zeus bot. This system will trigger an alert for the presence of Zeus bot if the three conditions regarding specific folder, network traffic and API hooks are all satisfied. Ahmed A.awad et al. [26] introduced a machine learning-based framework for detecting compromised hosts and networks that are infected by the RAT-Bots. This method relies heavily on the host agent because its network agent starts to run until it receives the alarm sent by the host agent. However, the host agent is difficult to obtain a very high TPR because modern RATs use various concealing technology. Different from the above methods, the proposed PRATD focus on RAT detection and can simultaneously analyze the
behaviours of RAT on host-side and network-side in the two states of keep-alive and command & control, thus contributing to achieve high detection accuracy.

3. RAT Runtime State Analysis

Generally, RATs are based on the C/S architecture and thus RAT consists of a client and a server. The client is controlled by the attacker and the server is implanted into the victim computer. The attacker uses the client to control the server to implement remote control for the victim host. In the early days, the client of a RAT will actively connect to its server, but this kind of RAT is easy to be detected because many security devices strictly check incoming traffic. For concealment, the RATs that use the server to actively connect to the client are widely used recently. As is shown in Figure 1, the whole communication of RAT includes three runtimes states: connection establishment, keep-alive and command & control. In the three states, connection establishment is the first state and only last a short time, while keep-alive and command & control states appear alternately and repeatedly [19]. In particular, this article focuses more on the keep-alive and command & control states because they take up most of the entire runtime of RAT. Next, we will introduce the details of the two states.

![Whole communication of RAT](image)

3.1. Keep-Alive State

The keep-alive state has the longest period of the runtime of RAT, and it is also the state with the smallest change. When the server of RAT does not receive a command request within a certain amount of time, the server will actively send a keep-alive request to the client to inform the hacker that it is online. On the one hand, since the size of the keep-alive request is small, the network packet of a RAT that carries the keep-alive request often contains only a few bytes and the most same content. On the other hand, the server of RAT in the keep-alive state would perform few actions on the host, since it does not execute substantial operations without receiving a command request from the client. Therefore, from the perspective of the detection, the size and content of the network packet, and the number of host operation records can be used to identify the behaviours of RAT in its keep-alive state. However, some benign programs such as instant messaging software also have the same kind of communication that keeps the connection alive, so it is difficult to distinguish between these benign programs and RATs by using only the network-side features in this state. Considering the mentioned conditions, it is rational to add host-side features to detect the behaviours of RAT in its keep-alive state more efficiently.

3.2. Command & Control State

Command & control is the state in which the client and server of RAT interact most frequently. After the client and server successfully establish a connection, the hacker controls the client to send
a command request to the server as needed. After receiving the requested command, the server of RAT will complete the requested task quickly and return the result to the client. Firstly, since the commands for stealing victim’s data are often used by an attacker, the server of RAT usually send the corresponding data to the client in its command & control state, which results in the RAT’s connections in this state often have large amounts of upload data or packets. Secondly, since there will be a think time after the hacker receives the result of the request command, the next request command will be sent until the hacker makes a decision, which leads to a relatively long interval time between a request and the next request in this state. Thirdly, some host behaviours and network traffic will be generated on the victim host because the server needs to use system resources to complete the request command and send the result to the client, which generates many host and network records in this state. Therefore, from the perspective of the detection, it is important to pay attention to the network traffic and host behaviours related to the executions of RAT’s commends.

As mentioned above, there are some interactions between the client and server of RAT in the keep-alive and command & control states, and as a result, the network traffic and host operation records regarding to the actions of RAT are generated correspondingly. Therefore, both the network traffic and host operation records are useful for detecting RAT. Besides, it can be found that the network traffic and operational records in the keep-alive and command & control states of a RAT are different. In the keep-alive state, the amounts of network traffic and operation records generated by RAT during different periods are usually the same, while they are often different in the command & control state. Based on these observations, it is reasonable to detect RATs by using the double-side features and in each of the keep-alive and command & control states. Follow this line of thought, we proposed PRATD and give the details in Section 4.

4. PRATD: A Phased RATs Detection Method with Double-Sides Features

To describe the proposed PRATD, we first introduce its architecture. Then, the features used in the PRATD are described. Finally, we explain the details of the model training and detection.

4.1. Architecture

The intuition of PRATD is that we believe that a RAT not only transmits information on the network but also uses host resource during it is running. Besides, as mentioned above, RAT has two main runtime states, i.e., keep-alive and command & control, and we find that RAT has the different host or network behaviours when they are in these two states. Therefore, we combine the host-side and network-side features and train two detection models separately for these two states. As shown in Figure 2, the proposed PRATD can be divided into three parts: Feature extraction phase, Model training phase, Detection phase. In the feature extraction phase, the features of the host-side and network-side are firstly extracted separately, and then these features are combined into one feature vector. In the model training phase, the feature vectors are labelled and be used as two training sets to build two detection models for the keep-alive and command & control states of RAT by using machine learning algorithms. Then, the feature vectors of the test set will be identified by the built detection models to examine whether they belong to the behaviours of the RAT. More details will be described next.

4.2. Feature Extraction

A communication session represents all information in a communication interaction between two parties. It is usually determined by a 5-tuple (source IP, source port, destination IP, destination port, protocol). However, communications initiated to the same destination address (including the destination IP and destination port) over a period of time are often initiated by the same process. Therefore, in this article, we define a session as all network traffic that has the same destination address in a time-window, which is set to five minutes. Specifically, a session consists of four elements: source IP, destination IP, destination port, and protocol. To obtain a good detection performance,
both host-side and network-side features are used in this article to detect RATs. In this research, seven network session features and 10 host process features are respectively extracted from network and host.

**Figure 2. Architecture of the PRATD.**

4.2.1. Network-Side Features Extraction

To analyze the network behaviour of RAT, we mainly monitor network layer and transport layer protocols. Given that the most widely used transport layer protocols are TCP and UDP, this article focuses solely on the sessions using TCP or UDP. Then, we extract features from the network sessions. Compared to most benign programs, RATs may have less communication traffic during communication in order to maintain the characteristics of concealment. To test this idea and investigate the different network behaviours of benign programs and RATs, we used five benign programs and three RATs. These five benign programs belong to five different types of applications including browsers, instant messaging software, video software, collaborative office software, and download software. The three RATs are darkcomet, njrat and vantom, which are well-known and widely used. We collected network traffic of the eight programs separately in a pure Windows 7 environment and compared them from four different perspectives by calculating the mean of five time-windows. Specifically, the RATs exhibited the following network characteristics during communication: (1) The RATs generally communicate with the target host with fewer source ports than benign programs in order to maintain characteristics of concealment. As shown in Figure 3a, the numbers of source port usages of the three RATs are usually less than that of the five benign programs. (2) For the servers of the RATs, their received packets are used to transmit command and control information, so the size of these packets is usually small. Figure 3b shows the ratios of small download packets to all download packets of the eight processes, and it can be seen that the ratio of RAT is usually greater than that of benign programs. (3) After receiving the result of a request command, the hacker often needs a period of think time to decide its next move, thus there is usually a longer interval between a request and the next request than many benign programs. Figure 3c exhibits the average intervals of communication interaction of the eight processes. (4) For the servers of the RATs, the size of upload-side data is generally larger than the download-side data. The ratios of the upload data to the download data of the eight processes are given by Figure 3d, and it can be observed that the ratio of RAT is usually greater than 1.0. Please note that in Figure 3a a few benign programs exhibit some similar network characteristics with RATs.
Figure 3. Some characteristics of several benign programs and RATs on network side.

Based on the above network characteristics of the RATs, we have extracted seven network features: $N_{src}$, $N_{min}$, $I_{avg}$, $R_m$, $R_p$, $R_l$, and $protocol$. Detailed description is as follows:

- $N_{src}$: The number of source ports used in a session, also known as the number of sub-connections.
- $N_{min}$: The number of small download packets in a session. This article regards a packet as a small packet when the size of the packet is less than a threshold (it is set to 70 bytes in this article).
- $I_{avg}$: The average interaction time of multiple interactions in a session.
- $R_m$: The ratio of small download packets and all download packets in a session. $R_m = \frac{N_{min}}{N_{all}}$, $N_{all}$ represents the number of all packets in a session.
- $R_p$: The ratio of upload and download packets in a session. $R_p = \frac{N_{upac}}{N_{dpac}}$, $N_{dpac}$ represents the number of all upload packets, $N_{upac}$ represents the number of all download packets.
- $R_l$: The ratio of the size of upload data to that of download data in a session. $R_l = \frac{L_{up}}{L_{down}}$, $L_{up}$ represents the size of all upload data, $L_{down}$ represents the size of all download data.
- $protocol$: The used transport layer protocol.

4.2.2. Host-Side Features Extraction

From the observations in Section 4.2.1, it can be found that using only network-side features to detect RATs may lead to some mistakes. Intuitively, RATs not only generate network traffic but also leave traces on the host when they are running, because they need system resources to complete its functions. In contrast to most benign programs, RATs usually use less security-critical system resources for most of the time because they need to maintain their own concealment. Therefore, it is helpful for distinguishing between benign programs and RATs by adding host-side features.
Base on the analysis of different programs, we believe that file, network, registry and process belong to security-critical system resources. Similarly, we compared several host operations of the same eight programs as Section 4.2.1 in a pure Windows 7 environment. The number of file creation records, the number of network connection creation, the number of registry operation (including creating registry record, deleting registry record, and modifying registry record) and the number of process operation (including process accessing and process creating) of these eight programs in a time-window (the value is the average of five time-windows) are shown in Figure 4a–d, respectively. As shown in these figures, there are big differences between RATs and benign programs in their operations of file, network, registry and process.

![Figure 4](image)

**Figure 4.** Some characteristics of several benign programs and RATs on host side.

The host-side features are extracted based on each process. Through observation and analysis, it is found that the content of network transmission changes little and the usage of security-critical system resources is relatively fixed when RATs are in the keep-alive state, and the transported information is varied greatly when the RATs in command & control state. Considering that there are different system resource usages in the different states of RAT, we have extracted 10 host-sides features. These features can be divided into two categories: one is the number of resource usage of a process in a time-window and the other is the proportion of resources used by the process in the current time windows. A detailed description of the host-side features is shown in Table 1.

As shown in Table 1, the 10 host-side features are related to the host operations of a program. For example, the number of network connection creations indicates the number of network connections created by a process in the current time-window, and the proportion of network connections represents the ratio of the number of network connection creations of a process to the total number of network connections created by all processes of an operational system in the current time-window. Others are similar to the features described above.
Table 1. Detail of host-side features.

| No. | Description | Total | Detailed |
|-----|-------------|-------|----------|
| 1   | Number of security-critical system resources used | 5     | The number of network connection creation, file creation, process access, process creation, registry operation in a time-window. |
| 2   | Proportion of security-critical system resources used | 5     | The proportion of network connection creation, file creation, process access, process creation, registry operation in a time-window. |

4.2.3. Features Combination

The extraction of the network-side and host-side features are performed simultaneously. Then, we need to combine the double-sides features to construct a combined feature vector which will be used as the input for the next phase. We monitor the record of TCP/UDP network connection for each process and correlate host-side and network-side features by finding the process ID of the session. PRATD determines a process ID corresponding to a session by querying the network connection record that is closest to the current session time. The details are shown in Algorithm 1.

Algorithm 1 Double-sides features combination

**Input:** A collection $N$ of network-side feature vectors composed of $n$ vectors, $\{I_1, I_2, \ldots, I_n\}$, a collection $H$ of host-side feature vectors composed of $m$ vectors, $\{K_1, K_2, \ldots, K_m\}$, the collection $D$ of records of each process creating network connection in host composed of $r$ records, $\{d_1, d_2, \ldots, d_r\}$.

**Output:** The collection $M$ of double-sides feature vectors composed of $s$ vectors, $\{V_1, V_2, \ldots, V_s\}$.

1: $M = \emptyset$;
2: Sort $d_k (k = 1, \ldots, r)$ in ascending order of the generated times;
3: for each feature vector $I_i \in N$ do:
4:    $s_i \leftarrow$ the 4-tuple of $I_i$;
5:    $t_i \leftarrow$ the timestamp of $I_i$;
6:    $c \leftarrow$ the ID of the record $d_k$ of network connection creation in $D$ whose timestamp is closest to $t_i$
7:    while $c > 0$ do:
8:       if $d_c == s_i$ then: // $d_c$ is the 4-tuple of $c$
9:          $p_i = p_c$; break; // $p_i$ represents the process ID of $I_i$, $p_c$ indicates the process ID of the record $d_k$ in $D$
10:      else: $c = c - 1$;
11:     end if
12:    end while
13: for each feature vector $K_j \in H$ do:
14:    if process ID of $K_j == p_i$ then:
15:       $K_i = K_j$; break // $K_i$ represents the host-side features related to $I_i$
16:     end if
17: end for
18: $V_i = I_i \cup K_i$;
19: $V_i$ appends in $M$;
20: end for
21: return $M$

As shown in Algorithm 1, line 2 sorts the records $d_k (k = 1, \ldots, r)$ of each process creating network connection in ascending order of the generated times, where $r$ is the number of records. If the quick-sort algorithm [27] is employed, the time complexity of the sort is $O(r \log r)$. The main loop (Lines 3–20) looks for a feature vector from the collection $H$ of host-side feature vectors associated with every feature vector in the collection $N$ of network-side feature vectors. The first inner loop (Lines 7–12) searches the process ID that creates the session connection, the time complexity for this loop is $O(c)$, where $c$ is the
ID of the record whose timestamp is closest to the network-side feature vector. The second inner loop (Lines 13–17) finds the corresponding network-side feature vector related to every host-side feature vector based on the found process ID, the time complexity of this loop is \(O(m)\) (\(m\) is the number of host-side feature vectors). Then, the host-side features are concatenated to the network-side features in Line 18. Overall, the time complexity of Algorithm 1 is \(O(nm)\).

4.3. Model Training and Detection

Model training: In this phase, two RAT detection models will be trained. To obtain two training sets, a controllable real environment is built and the network traffic and host behaviours of benign programs and RATs are collected during they are running. As mentioned in Section 4.2, each feature vector in these two training set contains both network-side and host-side features as well as a class label (RAT or benign program). Besides, owing to these two detections models being used for the different two runtime states of RAT, it is necessary to collect the network traffic and host behaviours of RAT in its keep-alive state and command & control state respectively, and then give a tag (keep-alive state or command & control state) to the corresponding feature vector. Please note that these tags are only used in the training phase to distinguish different states. Therefore, two training sets will be obtained in this phase. One consists of the feature vectors belonging to benign programs and RAT (keep-alive state), another includes the feature vectors belonging to benign programs and RAT (command & control state). Based on the two training set and machine learning algorithms, two RAT detection models are built, and then they will be used in the detection phase.

Detection: In this phase, the detection results for each unlabelled sample of the test set will be given. Specifically, each feature vector in the test data will be used as input and then will be simultaneously examined by the built two RAT detection models, which will give its detection result respectively. After the two detection results are obtained, we considered an unlabelled instance from the test set as a RAT if one of the two detection models regards it as RAT, because its feature vector is similar with the feature vectors belonging to at least one mainly states of RAT. The details are shown in Algorithm 2. In Lines 2–3, the feature vector is simultaneously detected by model A and model B, and get the respective detection result. The time complexities of the training Model A and Model B and detection depend on the classification algorithms that are used. Lines 4–8 give the final detection result of a feature vector by combining the respective detection result. For \(n\) feature vectors, the time complexity for judging their final detection results is \(O(n)\).

Algorithm 2 Merge phased results to give final detection result

| Input: | A collection \(M\) of double-sides feature vectors composed of \(n\) vectors, \(\{V_1, V_2, \ldots, V_n\}\), the detection model \(ModelA\) for the command & control state of RAT, the detection model \(ModelB\) for the keep-alive state of RAT. |
| Output: | The collection \(R\) of all detection results, \(\{r_1, r_2, \ldots, r_n\}\), \(r\) is the predicted result (benign or RAT). |

1: for each feature vector \(V_i \in M\) do:
2:    Model A detects \(V_i\) and give a result \(r_a\) (benign or RAT);
3:    Model B detects \(V_i\) and give a result \(r_b\) (benign or RAT);
4:    if \(r_a = RAT\) or \(r_b = RAT\) then:
5:        \(r_i = RAT\);
6:    else:
7:        \(r_i = benign\);
8:    end if
9:    \(r_i\) appends in \(R\);
10: end for
11: return \(R\)
5. Evaluation

In this section, we describe the experimental environment and show our real-world data sets collected from both benign programs and RATs. The detection results of the proposed PRATD for the binary classes (benign and RAT) on the two test sets are also given. Besides, different classification algorithms are used in PRATD to test its performance. Our experiments were conducted using Windows 7 and 10 with Intel CPU I7-8565U. The experimental software environment is python 3.6 and the machine learning package uses scikit-learn 0.19.1.

5.1. Experiment Environment

To evaluate the proposed PRATD, we collected data on several machines for seven days. To prevent the RATs from harming normal hosts, we have established some controlled environments, which are Windows 7 X64 or Windows 10 X64 running on VMware and can access the Internet. We install each individual server of RAT in one controlled environment to ensure no cross-infection between different RATs. The client of each RAT is placed on a cloud server with a public IP address. To extract the host-features, we use System Monitor [28] to record the operation of process. While listening to the operations in the controllable system, the network traffic is captured by using the Wireshark installed in the VMware host. In particular, we only focus on the TCP and UDP protocols and filter the multicast protocol. Moreover, we also filter the traffic of NetBIOS service which ports are 137 and 138. In the process of data collection, the data that is collected during the periods of RAT’S command interactions would be given a tag “command & control state” and the rest is given a tag “keep-alive state”. After collecting and preprocessing the raw data, we store the data with a structured form into the MYSQL database.

5.2. Experimental Data Sets

The work of this article is mainly to detect whether a program with network communication is RAT. Software with network communication used by normal users can usually be divided into five categories: browsers, instant messaging software, video software, collaborative office software, and download manager software. Therefore, we select one popular benign program belonging to every five categories and 20 well-known RATs for the experiment. The details are listed in Table 2.

| Type | Quantity | Program Description |
|------|----------|---------------------|
| Benign | 5 | Internet Explorer, Tencent QQ, Iqiyi, YoudaoNote, Xunlei Thunder |
| RAT | 20 | darkcomet, njrat, Bozok, vantom, xtreme, spynet, comet, greame, DarksRat, Pain, daleth, xena, pshare, cmdr, pandora, NoveLite, MLRAT, mq5, Proton, Quasar |

Experimental data include training and test sets. Both of them contain benign and RAT samples. For the experiment, we collected data from 20 VMware hosts. Each VMware host installs the five benign programs and one of the 20 RATs, and the RAT in each VMware host is different. During the data collection, we use each benign program routinely and execute different commands of RATs. The training data is collected from 12 VMware hosts with Windows 7 X64. For test data, we collected records from 20 VMware hosts with different desktops and divided them into two test sets. The data newly collected from the 12 VMware hosts with Windows 7 X64 used for training set collection are regarded as the known test set, the data collected from the eight newly added VMware hosts (three based on Windows 7 X64, five based on Windows 10 X64) are used as the unknown test set.
Please note that the several new VMware hosts with the different desktops that we have no data collected for training are intentionally added to demonstrate that the training models of PRATD are not limited to specific machines or RATs. In particular, the data from each VMware host is collected for about two hours. Therefore, about 64 h of data are used to build the training and test sets. After feature extraction and double-sides features combination, the training set contains 28,730 benign records and 251 RAT records. The known RAT test set includes 25,305 benign records and 266 RAT records, and the unknown RAT test set includes 30,318 benign records and 166 RAT records. The distributions of these data sets are in line with the real situation that this is a low ratio of RAT’s conversations. The details of the data sets are shown in Table 3.

|                | Training Set | Testing Set (Known) | Testing Set (Unknown) |
|----------------|--------------|---------------------|-----------------------|
| Trojan record  | 28,730       | 25,305              | 30,318                |
| Trojan record  | 251          | 266                 | 166                   |
| Total          | 28,981       | 25,571              | 30,484                |

5.3. Evaluation Metrics

We employ several widely used evaluation metrics, including TPR, FNR, TNR and FPR to evaluate the detection performance of PRATD. TPR is the proportion of the correctly detected RAT Samples. Similarly, FNR is the proportion of RAT samples misclassified as benign, TNR is the proportion of benign samples are correctly classified, and FPR is the proportion of benign samples misclassified as RATs.

For better illustration, we also use the confusion matrix to illustrate the four possible outcomes of a binary classifier. They can be calculated using the confusion matrix shown in Table 4, and are defined as follows:

| Actual Value | Predict Value | Trojan | Benign |
|--------------|---------------|--------|--------|
| Trojan       | True Positive (TP) | False Negative (FN) |
| Benign       | False Positive (FP) | True Negative (TN) |

\[
TPR = \frac{TP}{TP + FN} \quad (1)
\]
\[
FNR = \frac{FN}{TP + FN} \quad (2)
\]
\[
TNR = \frac{TN}{FP + TN} \quad (3)
\]
\[
FPR = \frac{FP}{FP + TN} \quad (4)
\]

5.4. Evaluation Results

5.4.1. Different Classification Algorithms Combination Results

Two RAT detection models are included in PRATD and used for the different runtime states of RAT. To achieve a good detection performance, we use TPR, FNR, and FPR as the metrics to evaluate the detection results of each combination of two classification algorithms and find an optimal combination as the two detection models used for PRATD. In this experiment, six well-known classification algorithms including k-Nearest Neighbor (KNN) [29], Logistic Regression (LR) [30], Support Vector Machine (SVM) [31], Decision Tree (DT) [32], Naive Bayes (NB) [33] and AdaBoost [34] are selected, since they are widely used in the domain of malware detection and get good results.
We use the known test set to test the TPR, FNR, TNR, FPR of different combinations of the six algorithms and select the combination that obtains the best result. The results of different combinations are shown in Tables 5–8.

**Table 5.** The TPR (%) achieved by different combinations of Model A and Model B.

| ModelA | KNN     | LR     | DT     | SVM    | NB     | AdaBoost |
|--------|---------|--------|--------|--------|--------|----------|
| KNN    | 65.414  | 54.511 | 86.466 | 66.917 | 86.466 | 81.955   |
| LR     | 55.639  | 24.060 | 77.444 | 59.774 | 78.571 | 72.932   |
| DT     | 79.323  | 75.940 | 92.481 | 83.083 | 91.729 | 89.474   |
| SVM    | 58.647  | 27.068 | 77.820 | 59.774 | 79.699 | 73.684   |
| NB     | 93.609  | 87.594 | 99.248 | 90.602 | 96.992 | 96.617   |
| AdaBoost | 91.353 | 86.842 | 99.248 | 90.602 | 96.992 | 96.617   |

**Table 6.** The FNR (%) achieved by different combinations of Model A and Model B.

| ModelA | KNN     | LR     | DT     | SVM    | NB     | AdaBoost |
|--------|---------|--------|--------|--------|--------|----------|
| KNN    | 34.586  | 45.489 | 13.534 | 33.083 | 13.534 | 18.045   |
| LR     | 44.361  | 75.940 | 22.556 | 40.226 | 21.429 | 27.068   |
| DT     | 20.677  | 24.060 | 7.519  | 16.917 | 8.271  | 10.526   |
| SVM    | 41.353  | 72.932 | 22.180 | 40.226 | 20.301 | 26.316   |
| NB     | 6.391   | 12.406 | 0.752  | 9.398  | 3.008  | 3.383    |
| AdaBoost | 8.647  | 13.158 | 6.015  | 10.150 | 5.639  | 6.391    |

**Table 7.** The TNR (%) achieved by different combinations of Model A and Model B.

| ModelA | KNN     | LR     | DT     | SVM    | NB     | AdaBoost |
|--------|---------|--------|--------|--------|--------|----------|
| KNN    | 99.447  | 99.771 | 97.977 | 99.609 | 95.289 | 99.360   |
| LR     | 99.494  | 99.850 | 98.664 | 99.676 | 95.345 | 99.399   |
| DT     | 99.451  | 99.763 | 98.747 | 99.605 | 95.301 | 99.368   |
| SVM    | 99.506  | 99.854 | 98.609 | 99.688 | 95.357 | 99.411   |
| NB     | 90.476  | 90.646 | 89.544 | 90.630 | 89.555 | 90.559   |
| AdaBoost | 99.395 | 99.715 | 98.261 | 99.553 | 95.337 | 99.593   |

**Table 8.** The FPR (%) achieved by different combinations of Model A and Model B.

| ModelA | KNN     | LR     | DT     | SVM    | NB     | AdaBoost |
|--------|---------|--------|--------|--------|--------|----------|
| KNN    | 0.553   | 0.229  | 2.023  | 0.391  | 4.711  | 0.640    |
| LR     | 0.506   | 0.150  | 1.336  | 0.324  | 4.655  | 0.601    |
| DT     | 0.549   | 0.237  | 1.253  | 0.395  | 4.699  | 0.632    |
| SVM    | 0.494   | 0.146  | 1.391  | 0.312  | 4.643  | 0.589    |
| NB     | 9.524   | 9.354  | 10.456 | 9.370  | 10.445 | 9.441    |
| AdaBoost | 0.605  | 0.285  | 1.739  | 0.447  | 4.663  | 0.407    |

From Tables 5–8, it can be seen that there is a large difference in the detection results of the various combinations of the different classification algorithms. Tables 5 and 6 exhibit the TPR and FNR of various combinations of classification algorithms. It can be seen that when NB and DT are respectively used to build the Model A and Model B, the TPR of PRATD reaches the best (99.248%), and the corresponding FNR is also the best (0.752%). Tables 7 and 8 shows the TNR and FPR of the various combinations of classification algorithms and we can see that when SVM and LR are respectively used to build the Model A and Model B, the TNR of PRATD is the best (99.854%), and the corresponding
FPR is 0.146%. Taken together the obtained values of TPR, FNR, TNR and FPR, PRATD obtains a good result when its Model A and Model B all use AdaBoost. Therefore, AdaBoost is selected to build the two detection models of PRATD in the following sections.

5.4.2. Known RAT Detection

The detection performance of PRATD for the RATs which appear in the training phase is presented in this subsection. The experimental data is the known test set which has been introduced in Section 5.2. To accomplish a better observation, we tested six different methods separately, including host-based method (using the 10 host-side features and AdaBoost algorithm, hereinafter abbreviated to H), network-based method (using the seven network-side features and AdaBoost algorithm, hereinafter abbreviated to N), the method based the double-side feature (using the double-sides features and AdaBoost algorithm, hereinafter abbreviated to H+N), phased host-based method (using the 10 host-side features and two detection models, hereinafter abbreviated to PH), phased network-based method (using the seven network-side features and two detection models, hereinafter abbreviated to PN) and the proposed PRATD. Besides, a RAT detection method based on seven network features [19] and a RAT detection method based on the voted result of three different classifiers [22] are also used to compare with these methods. The experiment results of the eight different methods on the known test set are shown in Figure 5.

![Detection Results](image)

**Figure 5.** The detection results obtained by the different methods on the known test set.

Figure 5 shows the detection results of eight different detection methods for the known test set. As shown in Figure 5a, the TPRs of those eight methods is different. On the one hand, the TPRs of the methods based the double-side features (i.e., H+N and PRATD) are better than the methods using only one-side features (i.e., H or N, PH or PN). On the other hand, the TPRs of the phased methods (PH, PN, and PRATD) are better than the corresponding methods without dividing states (i.e., H, N, and H+N) respectively. Among these eight methods, PRATD obtains the best TPR (93.609%), and the TPRs of the other seven methods are all less than 91%. Correspondingly, the FNR of PRATD is 6.391%, while the FNRs of the other seven methods are all higher than 9%. Meanwhile, Figure 5c shows that the
5.4.3. Unknown RAT Detection

The detection performance of PRATD for the RATs which did not appear in the training phase is given in this subsection. The experimental data is the unknown test set which is introduced in Section 5.2. We also compared the detection results of PRATD and the same seven different methods described in Section 5.4.2.

The results of eight different detection methods for detecting the unknown RATs are given in Figure 6. As shown in Figure 6, we can also find that compared to the methods using only the one-side features (i.e., H, N, PH, PN), the methods based on the double-side features (i.e., H+N, PRATD) can obtain better TPRs and FNRs in detecting the unknown RATs. Meanwhile, the TPR can be increased and the FNR can be decreased when the multiple detection models are used. Figure 6a,b show that the TNRs and FPRs of N, H+N, PN and PRATD are better than the other four methods. Besides, Figure 6c,d show that the TNRs are all higher than 99.8%, and corresponding FPRs are all less than 0.2%.

![Figure 6](image)

**Figure 6.** The detection results obtained by the different methods on the unknown test set.

5.4.4. Detection Efficiency

In this subsection, we compared the detection times of PRATD and the same seven methods described in Section 5.4.2. The known test set is used here by measuring the detection times of the eight methods. Each method run 10 times and the average time was used as its detection time. The detection times of eight methods are shown in Table 9.

| Method | H  | N  | H+N | PH | PN | Method [19] | Method [22] | PRATD |
|--------|----|----|-----|----|----|-------------|-------------|-------|
| Time cost (s) | 0.233 | 0.267 | 0.292 | 0.686 | 0.703 | 0.022 | >1000 | 0.740 |
As shown in Table 9, the detection time of PRATD is only a little more than PH and PN, which suggests that there is no major increase in detection time when the double-sides features are used. It also can be observed that methods with multiple detection models (PH, PN and PRATD) need much more detection time than the methods with single detection model (H, N, H + N), and the detection time of PRATD is 0.740 second which can meet the need of fast detection. Method [19] has the shortest detection time, but its obtained TPR, FNR, TNR and FPR on the two test sets are not good. Method [22] needs much more detection time because it examines each packet in its detection phase. Overall, considering the obtained accuracy and detection time, it can be found that PRATD obtains highly competitive results compared with the other seven methods for detecting the known and unknown RATs, which suggests it is a competitive candidate for RAT detection.

6. Discussion

6.1. Host-Side and Network-Side Detections

As a kind of malware with the intention of stealing user’s privacy, executing commands without being noticed is a major design concern for every RAT. To detect RAT with high accuracy, it is necessary to study and analyze every behaviour of the RAT. As mentioned in [35], most existing RATs have the functions of the killing process, editing registry, searching file, executing program, uploading files, sustaining connection and so on. By analyzing these functions from the perspective of detection, it can be found that the behaviours of killing process, editing registry and searching file can be detected by using appropriate host-side features because the execution of these functions left more traces on the host than on the network. In contrast to the previous behaviours, the behaviours of sustaining connection and uploading file are better detected by using appropriate network-side features because they would generate a long time or a large amount of network traffic on the network, while the records of these behaviours on the host are less. Therefore, from the perspective of detection, it is still difficult to accurately identify all the behaviours of RAT when the detection is solely on the network side or host side. In this article, several features are extracted from the host and network sides respectively and are all used for detection. The purpose is to combine the detection capabilities of both the host and network sides to improve the detection performance for the behaviours of RAT. The experiment results shown in Section 5 reflect that the TPRs of the methods based on the double-side features are better than the methods using only one-side features.

6.2. Detection in Different Runtime States

RAT is typically under human control [4]. When the initial connection establishment state is finished, the emergences of keep-alive and command & control states are irregular. In the whole runtime of RAT, the keep-alive state has the longest period. Please note that the RAT in keep-alive state does not execute the substantial operations that are often done in the command & control state. To improve the TPR in the keep-alive state, PRATD collects the host behaviours and network traffic in these two states respectively and trains two detection models for each of the states. The experiment results shown in Section 5 reflect that the TPR for RAT can be improved by using two detection models for the two different states.

6.3. Evasion

The host-sides features used in this article are extracted by the records collected by the tool of System Monitor. If a RAT adopts some advanced technologies that can evade the monitoring of System Monitor, RAT will be able to evade detection by PRATD, because the process of feature combination of PRATD cannot be finished in this case. However, a program leaves not host operation records but generates a lot of network traffic should be made suspect. In that case, the other host monitoring techniques need to be used in PRATD for further detection for the RAT.
7. Conclusions

RAT has long been a threat to organization and personal computers. In this article, we propose a RAT detection method named PRATD. The first core of PRATD lies in its each detection model is separately trained for each of the two runtime states of RAT. Based on it, PRATD can detect RATs in their runtime states of keep-alive and command & control. The second core of PRATD is that it combines the features extracted from network-side and host-side, thus the detection capacities of RATs on network and host are combined together. We conduct the experiment by using five kinds of benign programs and 20 famous RATs. The experimental results show that PRATD obtains better detection results than host-based and network-based methods, which suggests it is the benefit to enhance the accuracy for detecting RAT by using the double-sides features and building detection models for the different RAT’s runtime states.

As for future work, we will test PRATD in practical scenarios. Moreover, the work of extracting better host-side features and further improve the TPR of RAT detection method also need to be addressed.

Author Contributions: Conceptualization, C.G.; methodology, C.G. and Z.S.; software, Z.S.; validation, Z.S. and G.S.; formal analysis, C.G. and Y.P.; investigation, C.G., Z.S., Y.P., Y.C. and C.J.; data curation, Z.S.; writing—original draft preparation, Z.S.; writing—review and editing, C.G., Y.P., G.S., and Y.C.; supervision, C.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China under Grant 61802081, the Science and Technology Foundation of Guizhou Province nos. [2020]1Y268 and [2018]3001, the Program for Science & Technology Innovation Talents in Universities of He’nan Province under Grant No. 18HASTIT022, Key Technologies R & D Program of He’nan Province under Grant No. 192102210295.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Milajerdi, S.M.; Gjomemo, R.; Eshete, B.; Sekar, R.; Venkatakrishnan, V.N. HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 1137–1152, [CrossRef]
2. Valeros, V.; Garcia, S. Growth and Commoditization of Remote Access Trojans. In Proceedings of the 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), Genoa, Italy, 7–11 September 2020; pp. 454–462, [CrossRef]
3. Rezaeirad, M.; Farinholt, B.; Dharmdasani, H.; Pearce, P.; Levchenko, K.; McCoy, D. SchröDinger’s RAT: Profiling the Stakeholders in the Remote Access Trojan Ecosystem. In Proceedings of the 27th USENIX Conference on Security Symposium (SEC’18), Baltimore, MD, USA, 15–17 August 2018; pp. 1043–1060.
4. Wu, S.; Liu, S.; Lin, W.; Zhao, X.; Chen, S. Detecting remote access trojans through external control at area network borders. In Proceedings of the Symposium on Architectures for Networking and Communications Systems (ANCS), Beijing, China, 18–19 May 2017; pp. 131–141.
5. Moser, A.; Kruegel, C.; Kirda, E. Limits of Static Analysis for Malware Detection. In Proceedings of the Twenty-Third Annual Computer Security Applications Conference (ACSAC), Miami Beach, FL, USA, 10–14 December 2007.
6. Farinholt, B.; Rezaeirad, M.; Pearce, P.; Dharmdasani, H.; Yin, H.; Le Blond, S.; McCoy, D.; Levchenko, K. To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 770–787, [CrossRef]
7. Signatures of RATs. Available online: http://bit.ly/2QnEaIC (accessed on 28 June 2020).
8. Yang, R.; Chen, X.; Xu, H.; Cheng, Y.; Xiong, C.; Ruan, L.; Kavousi, M.; Li, Z.; Xu, L.; Chen, Y. RATScope: Recording and Reconstructing Missing RAT Semantic Behaviors for Forensic Analysis on Windows. IEEE Trans. Depend. Secur. 2020, [CrossRef]
9. Kang, J.; Jang, S.; Li, S.; Jeong, Y.S.; Sung, Y. Long short-term memory-based Malware classification method for information security. Comput. Electr. Eng. 2019, 77, 366–375. [CrossRef]
10. Yuan, B.; Wang, J.; Liu, D.; Guo, W.; Wu, P.; Bao, X. Byte-level malware classification based on markov images and deep learning. *Comput. Secur.* 2020, 92, 101740.1–101740.12. [CrossRef]

11. Vasan, D.; Alazab, M.; Wassan, S.; Safaei, B.; Zheng, Q. Image-Based malware classification using ensemble of CNN architectures (IMCEC). *Comput. Secur.* 2020, 92, 101748. [CrossRef]

12. Pendleton, M.; García-Lebrón, R.; Cho, J.H.; Xu, S. A Survey on Security Metrics. *ACM Comput. Surv.* 2016, 49, 1–35. [CrossRef]

13. Zhang, H.; Zhang, W.; Lv, Z.; Sangaiah, A.K.; Huang, T.; Chilamkurti, N. MALDC: A depth detection method for malware based on behavior chains. *World Wide Web* 2020, 23, 991–1010. [CrossRef]

14. Bridges, R.; Hernández Jiménez, J.; Nichols, J.; Goseva-Popstojanova, K.; Prowell, S. Towards Malware Detection via CPU Power Consumption: Data Collection Design and Analytics. In the Proceedings of the 2018 17th IEEE International Conference on Trust, Security And Privacy in Computing And Communications/12th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1680–1684, [CrossRef]

15. Adachi, D.; Omote, K. A host-based detection method of remote access trojan in the early stage. In Proceedings of the International Conference on Information Security Practice and Experience (ISPEC), Zhangjiajie, China, 16–18 November 2016; Springer: Cham, Switzerland, 2016; pp. 110–121.

16. Wang, Q.; Hassan, W.U.; Li, D.; Jee, K.; Yu, X.; Zou, K.; Rhee, J.; Chen, Z.; Cheng, W.; Gunter, C.A.; et al. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis. In the Proceedings of the 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, CA, USA, 23–26 February 2020.

17. Markel, S.; Iñaki, G.; Mikel, I.; Urko, Z. Deep packet inspection for intelligent intrusion detection in software-defined industrial networks: A proof of concept. *Log. J. IGPL* 2020, 28, 461–472.

18. Kim, J.; Kim, J.; Kim, H.; Shim, M.; Choi, E. CNN-Based Network Intrusion Detection against Denial-of-Service Attacks. *Electronics* 2020, 9, 916. [CrossRef]

19. Jiang, D.; Omote, K. An approach to detect remote access Trojan in the early stage of communication. In Proceedings of the IEEE 29th International Conference on Advanced Information Networking and Applications (AINA), Gwangju, Korea, 24–27 March 2015; pp. 706–713.

20. Zhu, H.; Wu, Z.; Tian, J.; Tian, Z.; Qiao, H.; Li, X.; Chen, S. A Network Behavior Analysis Method to Detect Reverse Remote Access Trojan. In Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 November 2018; pp. 1007–1010, [CrossRef]

21. Xie, J.; Li, S.; Yun, X.; Zhang, Y.; Chang, P. HSTF-Model: An HTTP-based Trojan Detection Model via the Hierarchical Spatio-Temporal Features of Traffics. *Comput. Secur.* 2020, 96, 101923. [CrossRef]

22. Pallaprolu, S.C.; Namayanja, J.M.; Janeja, V.P.; Adithya, C.T.S. Label propagation in big data to detect remote access Trojans. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 3539–3547, [CrossRef]

23. Zeng, Y.; Hu, X.; Shin, K.G. Detection of botnets using combined host-and network-level information. In Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Chicago, IL, USA, 28 June–1 July 2010; pp. 291–300.

24. Shin, S.; Xu, Z.; Gu, E. EFFORT: A new host–network cooperated framework for efficient and effective bot malware detection. *Comput. Netw.* 2013, 57, 2628–2642. [CrossRef]

25. Kalpika, R.; Vasudevan, A.R. Detection of Zeus Bot Based on Host and Network Activities. In *Security in Computing and Communications*; Springer: Singapore, 2017; pp. 54–64.

26. Shafee, A.; Sayed, S.; Salem, S. Collaborative Framework for Early Detection of RAT-Bots Attacks. *IEEE Access* 2019, 7, 71780–71790.

27. Xiang, W. Analysis of the Time Complexity of Quick Sort Algorithm. In Proceedings of the 2011 International Conference on Information Management, Innovation Management and Industrial Engineering, Shenzhen, China, 26–27 November 2011; Volume 1, pp. 408–410, [CrossRef]

28. Sysmon. Available online: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon (accessed on 28 June 2020).

29. Baldini, G.; Geniatiakis, D. A Performance Evaluation on Distance Measures in KNN for Mobile Malware Detection. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 23–26 April 2019; pp. 193–198, [CrossRef]
30. Singh, H.; Bijalwan, A. Botnet Detection Using Logistic Regression Technique. *Int. J. Inf. Secur.* **2017**, *15*, 306–313.

31. Wadkar, M.; Di Troia, F.; Stamp, M. Detecting malware evolution using support vector machines. *Expert Syst. Appl.* **2020**, *143*, 113022. [CrossRef]

32. Utku, A.; Doğru, I.A.; Akcayol, M.A. Decision tree based android malware detection system. In Proceedings of the 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2–5 May 2018; pp. 1–4, [CrossRef]

33. Alkhateeb, E.M.; Stamp, M. A Dynamic Heuristic Method for Detecting Packed Malware Using Naive Bayes. In Proceedings of the 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, UAE, 19–21 November 2019; pp. 1–6, [CrossRef]

34. Zhang, X.Y.; Hou, Z.; Zhu, X.; Wu, G.; Wang, S. Robust malware detection with Dual-Lane AdaBoost. In Proceedings of the IEEE INFOCOM 2016—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, USA, 10–14 April 2016.

35. Yamada, M.; Morinaga, M.; Unno, Y.; Torii, S.; Takenaka, M. RAT-based malicious activities detection on enterprise internal networks. In Proceedings of the International Conference for Internet Technology & Secured Transactions (ICITST), London, UK, 14–16 December 2015; pp. 321–325.

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).