An Approach to Detect Remote Access Trojan in the Early Stage of Communication

Dan Jiang, Kazumasa Omote
School of Information Science,
Japan Advanced Institute of Science and Technology
Ishikawa, Japan 923-1292
Email: {jiangdan, omote}@jaist.ac.jp

***Abstract*—As data leakage accidents occur every year, the security of confidential information is becoming increasingly important. Remote Access Trojans (RAT), a kind of spyware, are used to invade the PC of a victim through targeted attacks. After the intrusion, the attacker can monitor and control the victim's PC remotely, to wait for an opportunity to steal the confidential information. Since it is hard to prevent the intrusion of RATs completely, preventing confidential information being leaked back to the attacker is the main issue. Various existing approaches introduce different network behaviors of RAT to construct detection systems. Unfortunately, two challenges remain: one is to detect RAT sessions as early as possible, the other is to remain a high accuracy to detect RAT sessions, while there exist normal applications whose traffic behave similarly to RATs. In this paper, we propose a novel approach to detect RAT sessions in the early stage of communication. To differentiate network behaviors between normal applications and RAT, we extract the features from the traffic of a short period of time at the beginning. Afterward, we use machine learning techniques to train the detection model, then evaluate it by K-Fold cross-validation. The results show that our approach is able to detect RAT sessions with a high accuracy. In particular, our approach achieves over 96% accuracy together with the FNR of 10% by Random Forest algorithm, which means that our approach is valid to detect RAT sessions in the early stage of communication.**

I. Introduction

With the development of network technology in recent years, the problem of data leakage has become more and more severe. In order to steal confidential information from enterprise or government network, attackers often use targeted attacks to attain their goal. There are three main intrusion ways of targeted attack: E-mail, USB and Drive-by-download. Remote Access Trojan (RAT) is used to steal confidential information from the target organization. This kind of Trojan allows the attacker to control the victim's PC remotely and secretly [1]. Attacker can not only remotely download or upload files, but also surveil the keylogger or the system screen. RATs can automatically send connection requests to the attacker's Command and Control (C&C) server from the Intranet, in order to build an encrypted tunnel to connect two sides through the firewall. Therefore, some of the existing policies like port filtering and Deep Packet Inspection (DPI) [2], [3], [14] are not enough to prevent data leakage problems caused by targeted attacks [3]. The Port filtering policy is limited since RAT can use port 80 or 443 as in some normal sessions. DPI is also limited since it is incompetent for an encrypted payload.

As it is difficult to prevent the intrusion of RATs completely, detecting illegal RAT sessions after its intrusion is necessary. Li et al. [3] propose a detection system to detect Trojans by their original network behaviors, which achieves an accuracy of more than 90%. This system extracts network features from a session that begins from a SYN packet in the TCP three-way handshake and ends with a FIN/RST packet. However, RAT often maintains its TCP tunnel for a long time hence confidential information may already leak before the detection. Meanwhile, there is also another problem: how to distinguish RATs sessions from legitimate sessions. RATs need to steal confidential information and send it back to the attacker, which means that the outbound traffic is much greater than the inbound [3]. In the recent past, some popular cloud services and P2P applications had similarly great outbound traffic. Accordingly, it became difficult to divide them while detecting. Liang et al. [5] hybridize host-based and network-based techniques to raise the True Positive Rate above 97%. Nevertheless, it is necessary to install the detection system in all terminals, so it might be inconvenient to manage the detection system in a real organization.

In this paper, we define an *early stage* of the communication, and collect network features from the packets of this period to detect RAT sessions by machine learning techniques. Session's early stage is a preparing period of two communication sides such as handshaking and negotiating. Collecting network behavior features in the early stage accomplishes two goals: detecting as early as possible and dividing RAT sessions and legitimate sessions more clearly. To distinguish between RAT sessions and legitimate sessions, we pay attention to their inherent behavior features. More specifically, the duration of RAT's early stages is usually shorter than that of legitimate sessions. RATs usually confine their communications to low-and-slow network traffic in order to hide themselves inconspicuously. Meanwhile, legitimate sessions do not need to hide their network behaviors, which means that legitimate sessions have far more traffic than RAT sessions.

Three phases compose our method: the feature extraction phase, the training phase, and the detection phase. Seven network features are collected in the first phase. During the learning phase, we label these collected data normal or abnormal, to let machine learning algorithms analyze their network behaviors. Lastly, the detection model, which is generated in the second phase, is used to detect whether a real session is normal or not. For the implementation, we collect 10 kinds of RAT and 10 kinds of normal application to obtain

175 session samples. We then use K-Fold cross validation to evaluate the detection model. Principally, our approach is able to distinguish RAT and normal sessions with greater than 96% accuracy. The advantage of our approach is that it can execute the detection system speedily, while ensuring the convenience of management, and also applicable to detect unknown RATs when the TCP connections are used.

The remainder of the paper is structured as follows: In Section 2, we discuss some related works of Trojan detection. The background information on RAT and machine learning algorithms are presented in Section 3. Section 4 gives a detailed description of our proposed method. In Section 5, we present the implement process and result. Section 6 discusses our idea in more detail. Finally, Section 7 conclude the paper.

II. Related Work

Numerous approaches propose network-based detection methods in different ways. The common used network behavior features are packet number, data size, and the duration of the session. Li et al. [3] introduce some original network behavior features of RATs to train the detection system. For instance, the average packet interval time of RAT sessions is longer than that of legitimate sessions since the attackers need analyzing time to decide the next command. Although the detection system of [3] could attain a accuracy of around 90%, preventing data leakage might be difficult. The reason is that information from a SYN packet to a FIN/RST packet is applied to extract network behavior features. Unfortunately, RATs usually maintain the sessions unless their processes are interrupted, hence confidential information might be leaked before detection. Borders et al. [4] propose an approach which focuses on the outbound HTTP traffic. Outbound traffic is useful for detection; recently, cloud and P2P services also have similar outbound behaviors as RATs. Therefore, it might not be enough to detect RAT sessions only depending on the outbound traffic information. The system proposed by Yamada et al. [6] detects illegal activities of spying in the Intranet; the limitation of their system is that it only corresponds with Trojans which conduct spying activities with other PCs in the Intranet. Thus, it might be difficult to detect Trojans which do not communicate with other PCs in the Intranet. Yamauchi et al. [7] introduce the standard deviation of access interval time to detect C&C traffic in Bot-nets. Their idea is that it is common to use a robot program as the attacker in a Bot-net, because the access frequency of the attacker is less random than that of humans. However, it is still difficult to distinguish RAT sessions from legitimate sessions because of RAT attacker trends more possibility to be a human. There are also some papers about the detection method which combine Host-based and Network-based detection technique [5], [8], [9]. Though they attain high accuracy together with low false positive rate, the major limitation is the necessity of installing applications in all terminals in order to collect host-based information. This might be not suitable for a large organization.

Furthermore, Vahid et al. [10] introduce a new behavior feature which is the number of times that PSH flags (push flag) were set in the session to classify normal applications. Even so, like RAT sessions, cloud and P2P service sessions set numerous PSH flags during their real-time communication. We could not clearly distinguish RAT sessions from legitimate sessions by this feature. Qu et al. [11] use the first few packets through the communication pattern of those protocols to classify different normal protocols. However, whether it is valid for classifying RATs is still unknown.

III. Preliminary

A. Remote Access Trojan (RAT)

RATs are Trojan which can allow the attacker monitor and control a victim's PC remotely through networks. RATs usually consist of client-side and server-side. Contrary to the common knowledge, the server-side of a RAT is the victim. From RAT's operation screen (Figure 1), we confirm that a RAT can not only download any file from a victim's PC, but can also monitor the keylogger and system screen. There are two patterns of RATs communication. One of them begins from an attacker's client-side. The client-side is outside of target's network, so the traffic can not pass through the firewall easily. The other begins from a victim's server-side. In this pattern, traffic can even pass through a proxy server.

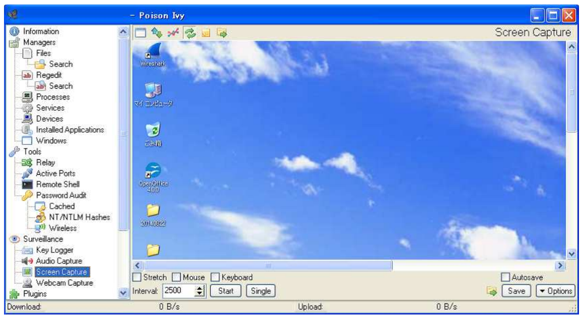


Fig. 1: Part of the RAT's Operation Screen

B. Machine Learning Algorithm

Machine learning is an intelligence technique which collects and analyzes information from enormous quantities of data. Machine learning algorithms constantly adjust themselves along with the change of data to process data automatically [12]. We choose 5 supervised algorithms, Decision Tree (DT), Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), and Random Forest (RF) to train the detection model in our implementation.

1) DT: DT has a tree structure. It divides branches by choosing the best property for classification from the input features [13]. Finally, it will generate a non-linear decision boundary to classify data. There are 3 methods to calculate the best property: ID3, C4.5, and CART (classification and regression tree). CART calculates the *gini index* for each property using the following formula:

$$Gini\ Index = 1 - \sum_{c} p^2$$

c represents all the classes, and p represents the probability of assignment to each class.

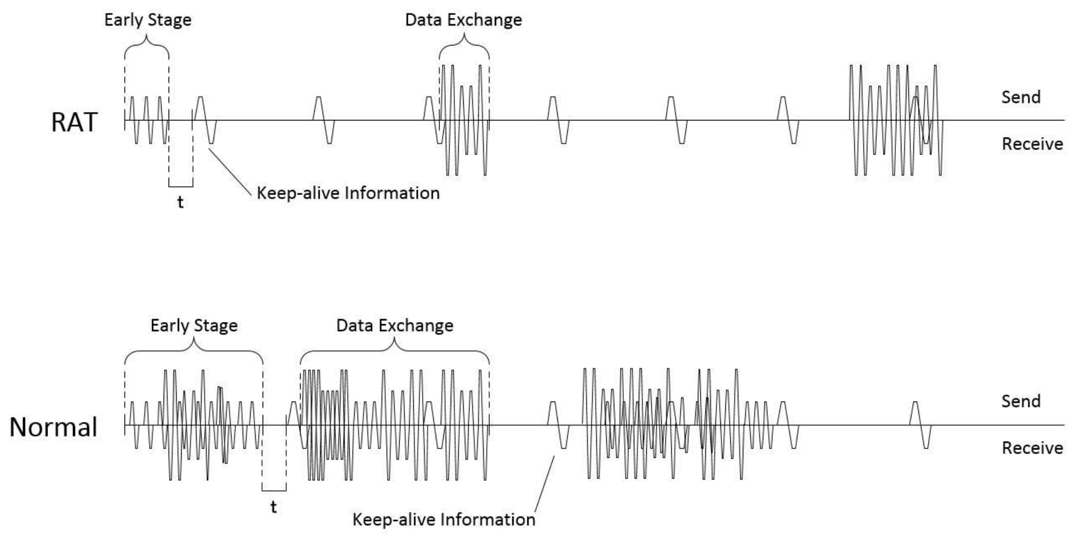


Fig. 2: Image of the Data Size in the Network Traffic of the Early Stage

2) SVM: SVM needs to determine a separating hyperplane which can let the closest training data point have the maximum margin[12]. The margin is the distance between the separating hyperplane and the nearest training data point of any class. In order to lower the generalization error, the margin should be as large as possible.

3) NB: NB is a simple and effective classifier which can process a large number of datasets quickly [13]. The method to choose the class is calculating the conditional probability as follows:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

4) KNN: KNN classifies data into a class according to the number of its neighbors. More specifically, the label of a data is determined by a majority vote of its nearest K class labels [12]. The distance of two vectors, p and q, is calculated in euclidean distance:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

5) RF: RF is a bagging method of DT [12]. By randomly creating numerous sub-decision trees, the estimating results of RF are more stable and accurate than single classification decision trees.

IV. Our Approach

In this section, we describe the design of our approach. Our approach collects packet information in the early stage and then, by means of machine learning algorithms, trains the detection model according to sample data network behaviors. After learning the relationship between network behaviors and their class labels, our approach detect RAT sessions from real communication sessions. Before describing the details, we first provide a formal definition of early stage.

Definition 1 (**Early Stage**): *The Early Stage of a session is a packet list which satisfies conditions as follows:*

- *Begins from the SYN packet of TCP 3-way handshake.*
- *Each packet interval time is less than the threshold t second(s).*

Packets in the early stage represent the difference between RAT session and legitimate session. Normal applications traffic is usually greater than that of RATs during the communication early stage. In other words, RATs always trend to behave secretly in order to hide themselves in the Intranet as long as possible. The reason is that, large amount of traffic is more easy to be discovered. Comparing with RATs, normal applications do not need to hide their network behavior. Moreover, normal applications leverage multi-thread technique in pursuit of a high communicate speed. Figure 2 is the image of traffic data size in both directions during the communication early stage.

We divide the whole process into three main phases: the feature extraction phase, the training phase, and the detection phase. The feature extraction phase collects network features from sessions in the early stage, then calculates the feature vectors. During the training phase, each vector is marked as Normal/Abnormal in order to train the detection model by machine learning algorithms. Finally, a new session will be classified into a Normal or Abnormal class based on the detection model in the detection phase. The following describes theses three phases in detail.

A. Feature Extraction Phase

In this phase, we choose 5 network features: PacNum, OutByte, OutPac, InByte, InPac based on existing works [3]-[11], and collect them from TCP packets in the early stage of a session at first. Then, in order to gain more information from these features, we calculate 2 more features: O/Ipac and OB/OP depend on them (Table I). After preparing all 7

TABLE I: Extract Network Features

Feature	Description
PacNum	packet number
OutByte	outbound data size
OutPac	outbound packet number
InByte	inbound data size
InPac	inbound packet number
O/Ipac	rate of OutPac/InPac
OB/OP	rate of OutByte/OutPac

features, we generate a network feature vector of this session. Figure 3 shows the complete process in this phase. First, feature variables are initialized to 0 or NULL. Second, after one packet is read by the process, there are two ways to continue the process according to the interval time. If it is shorter than the threshold *t* second(s), No.1∼5 feature variables in figure 3 increase recurrently as addition assignments. If it is longer than *t* second(s), No.6∼7 feature variables in figure 3 are calculated. Finally, 7 features are gathered to generate a network feature vector for the session. This process can be executed for all sessions at once in batches.

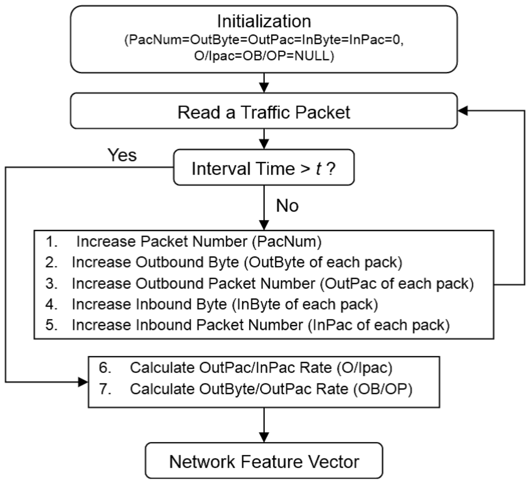

Initialization
(PacNum=OutByte=OutPac=InByte=InPac=0,
O/Ipac=OB/OP=NULL)
Read a Traffic Packet
Yes
Interval Time > t ?
No
1. Increase Packet Number (PacNum)
2. Increase Outbound Byte (OutByte of each pack)
3. Increase Outbound Packet Number (OutPac of each pack)
4. Increase Inbound Byte (InByte of each pack)
5. Increase Inbound Packet Number (InPac of each pack)
6. Calculate OutPac/InPac Rate (O/Ipac)
7. Calculate OutByte/OutPac Rate (OB/OP)
Network Feature Vector

Fig. 3: Process of Feature Extraction

B. Training Phase

During the training phase, the feature vectors generated in the aforementioned phase are learned by machine learning algorithms, to obtain the detection model (Figure 4). At first, we add normal/abnormal labels to every session's vector. The normal class is expressed by a label '0' as legitimate sessions of normal applications. Similarly, Abnormal class is labeled in '1' as RAT sessions.

Labeling sessions is for training a detection model by a supervised machine learning algorithm, which used to do the classification. Algorithm can learn the features and decide an identification method automatically. The final output of this phase is a detection model.

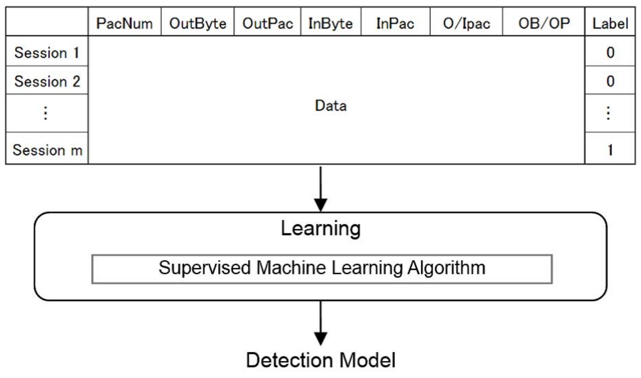


Fig. 4: Process of Training

C. Detection Phase

In the detection phase, the detection model predicts the label of a new session (Figure 5). The input is a feature vector which has no label information. If the output is '0', it means that this session is legitimate. Otherwise, this session is established by a RAT process.

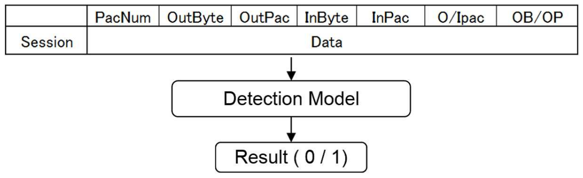


Fig. 5: Process of Detection

V. Evaluation

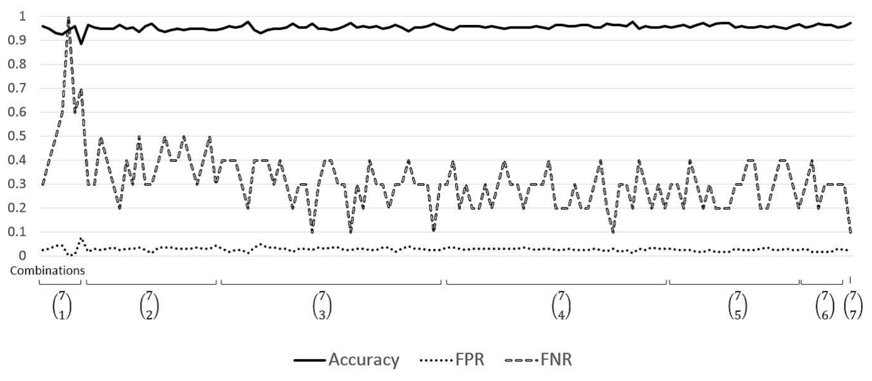
In this section, we evaluate the performance of our proposed approach, to verify its availability. The K-Fold cross validation technique is used to calculate accuracy, False Positive Rate (FPR), False Negative Rate (FNR) of the detection model. All phases are implemented by Python.

A. Experimental Data

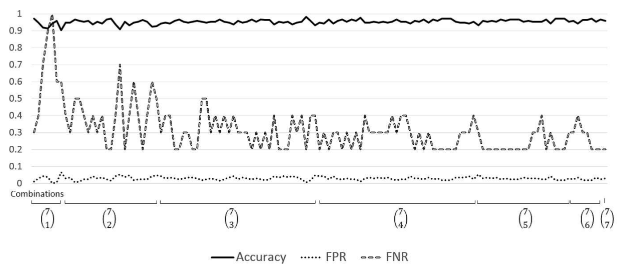
For the experimentation, we collect 10 types of RATs and 10 types of normal applications from the Internet. Their session data is applied to training the detection model. In order to confirm whether the network behavior features in the early stage can make the differences between RAT sessions and legitimate sessions clearly, we chose some cloud and P2P applications, which behave similarly with RATs. Table II shows the normal applications we chose.

B. Procedure

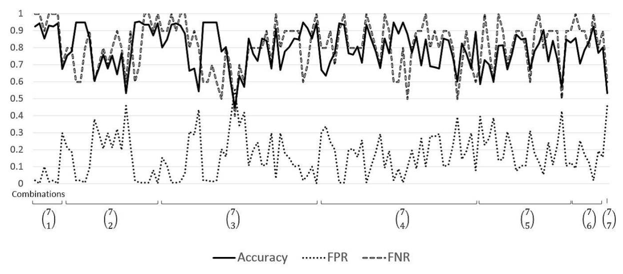
1) Pre-process: RAT samples are used in the real-world, which only can be executed in a virtual environment. On the other hand, traces of the normal application samples are collected on our laboratory network. First, we use Wireshark, a traffic monitor, to capture all these samples' traces. Then,



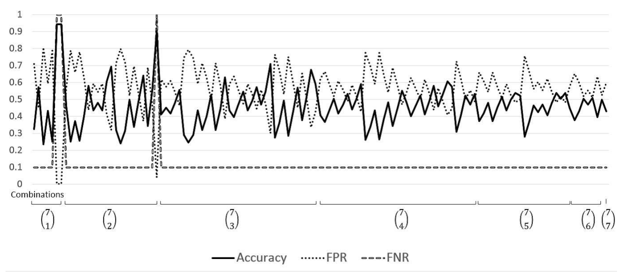
(a) RF



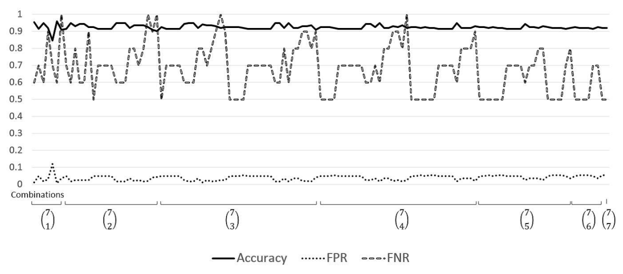
(b) DT



(c) SVM



(d) NB



(e) KNN

Fig. 6: Results of All Combinations

we divide the traces into sessions. In this paper, we define connections in a IP-pair communication as one session. Next, we pick out the invalid sessions and delete them. As an exception, because our purpose is to prevent confidential data being leaked out of the Intranet, the sessions in which source IP address and destination IP address are in a same LAN network, are excluded from the experimental object. Furthermore, we also pick out the one-direction sessions from our experimental object. The reason is that if a RAT tries to establish a tunnel communication, there must exist two directions packets during TCP handshaking. Finally, 175 sessions are extracted in total, 10 are RAT sessions, the rest are legitimate sessions.

2) Feature Extraction: During this phase, we assume that the threshold t is 1 second in our experiment, to verify the result. We take a Gh0st session (a kind of RAT) and a Dropbox session (a kind of normal application) as an example to show their feature vectors.

Name	PacNum	OutByte	InByte	OutPac	InPac	O/Ipac	OB/OP
Gh0st	6	355	138	4	2	2	88.75
Dropbox	47	17588	5732	27	20	1.35	651.4

TABLE II: 10 Types of Normal Application

Name	Description
Dropbox	cloud service
Skype	instant messenger
Chrome	web browser
YahooM!	instant messenger
Firefox	web browser
BitComet	P2P download tool
Teamviewer	P2P remote management tool
TorBrowser	anonymous web browser
BitTorrent	P2P download tool
PPTV	P2P video sharing tool

3) Detection Model Training: There is a machine learning library in Python named scikit-learn. We use 5 famous supervised machine learning algorithms in the scikit-learn. They are Decision Tree (DT), Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), and Random Forest (RF).

4) Cross Validation: Instead of detecting new sessions, we use the K-Fold Cross Validation (CV) technique to evaluate the detection model. K-Fold CV has 3 steps:

1) Divide the sample data into K parts without overlapping;
2) Use $K-1$ parts for training, and the rest one part for testing;
3) Loop step 2 in K times to generate evaluation parameters.

During the experimentation, we set K=10 to test one RAT session sample at one time. The evaluation parameters accuracy, FPR, and FNR are calculated basing on the following formula:

$$Accuracy = \frac{TrueDetectionNumber}{TotalNumber}$$

$$FPR = \frac{FalseDetectionNumber of '1'}{LegitimateSampleNumber}$$

$$FNR = \frac{FalseDetectionNumber of '0'}{RAT\,SampleNumber}$$

In order to detect any doubtful session in the Intranet, a low FNR is more important than a low FPR in our evaluation.

5) Result: Due to the 7 features, we train the detection model of each machine learning algorithm by all feature combinations. After analyzing all feature combinations' results, we found that 5 features and the ratio features perform better in RAT detection. More specifically, the combination "PacNum + OutByte + InByte + OutPac + InPac + O/Ipac + OB/OP" shows a comparatively good result. Along with the valid features, we also verified the valid machine learning algorithms for detection are RF and DT. The detailed evaluation parameters are summarized in Table III. For this combination, SVM gives the worst result by low accuracy and high FNR. Although NB gives a low FNR, the FPR is much higher. The result of KNN is also unsatisfied because of the high FNR. Comparatively, DT and RF are more suitable as their accuracy higher than 96%, together with FNR less than 20%.

TABLE III: The Performance Result of the Detection Model

Item	Accuracy	FPR	FNR
RF	0.971	0.023	0.100
DT	0.960	0.030	0.200
SVM	0.533	0.458	0.600
NB	0.430	0.597	0.100
KNN	0.919	0.054	0.500

In order to verify results of all 127 combinations for 5 algorithms, we output them as Figure 6. The horizontal axis represents the combinations as 1 element to 7 elements from left to right. The vertical axis represents the percentage of the evaluation parameters. The solid line indicates the accuracy, the square dot line indicates the FPR, and the dash line indicates the FNR. Several observations can be obtained from Figure 6. First, Both RF and DT perform well when more features are used for training. Second, SVM and NB are not suitable for detecting RAT sessions because of the high FNR or FPR. Third, in spite of KNN's accuracy, its high FNR leads it to be an unsuitable algorithm for detecting.

VI. Discussions

A. The Early Stage

In order to confirm whether features in the early stage can represent network behaviors, we compared the outbound data size (OutByte) between RAT sessions and legitimate sessions as Table IV. As normal applications usually have several sessions during their communications, the average values of legitimate sessions OutByte are shown in Table IV. Comparing the data size in the early stage, the difference between RAT sessions and legitimate sessions is clear. However, the data size of Bandook, Nuclear, Turkojan, Chrome, Firefox, and BitComet sessions in first one hundred packets are mixed together in a small range. This is because RATs behave secretly, so the early stage of RAT sessions usually end before one hundred packets. Beyond the early stage, RAT sessions will also have little traffic while receiving a monitoring command. On the contrary, there are over half of legitimate sessions have greater traffic over the early stage because their early stages are longer than one hundred packets. That is why the data size features are easily mixed together in first hundred packets. Here, we also give an real example to show the difference in the early stage (Figure 7), where we use Cerberus session as the RAT sample and one Skype session as the normal application sample. In this case, the Cerberus session's early stage only lasts 1.16 seconds but the Skype session lasts more than 3.53 seconds. The next packet after the early stage in the Cerberus session occurs 3 seconds later.

B. The Features

Reviewing the existing approaches, a session is suspected as abnormal when it has a larger outbound traffic than inbound. Traditional normal applications usually have greater inbound traffic rather than outbound traffic: for instance, web browsers or ftp services. However, P2P services such as Teamviewer or cloud services like Dropbox have a greater outbound traffic, similar to RATs. Table V shows the comparison of 7 network

TABLE IV: The Difference of Early Stage and First Hundred Packets (OutByte)

RAT	Early Stage / Fist Hundred Packets	Legitimate	Early Stage / Fist Hundred Packets (average)
Bandook	303/2319	Dropbox	9148/10161
Bozok	308/4472	Skype	1626/1776
Cerberus	516/3055	Chrome	6458/2498
Nuclear	343/2420	YahooMessenger	18954/9888
Poison Ivy	1368/3612	Firefox	3387/2504
Turkojan	294/2977	BitComet	65827/2677
Gh0st	355/3175	Teamviewer	5232/5623
Netbus	129/74696	TorBrowser	90435/31593
OptixPro	221/1390	BitTorrent	1094/4271
ProRat	130/4583	PPTV	63471/3700

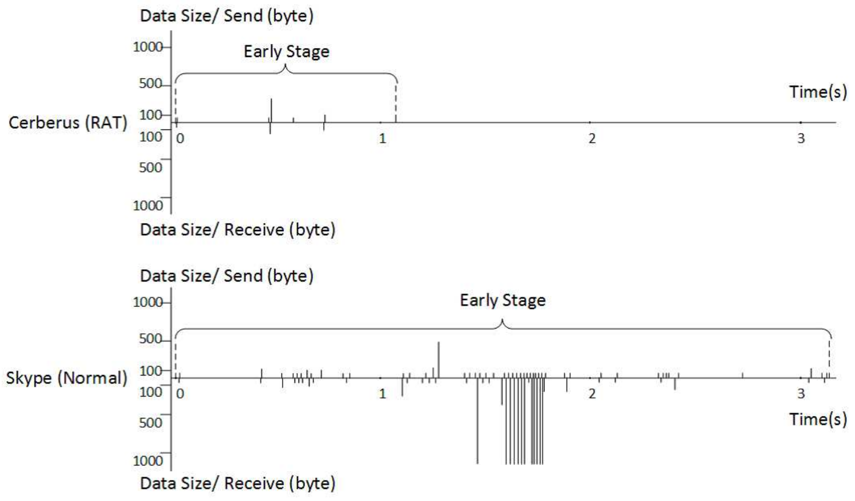


Fig. 7: An Example of Real Traffic Data Size in the Early Stage

features used in our approach. These data show that, during the early stage of a communication, RATs usually hide themselves in the Intranet as long as possible.

TABLE V: Compare the Features in the Early Stage

Feature	Type	Trend
PacNum	N	78% are more than**10pack**
	R	20% are more than**10pack**
OutByte	N	93% are more than **500byte**
	R	20% are more than **500byte**
OutPac	N	52% are more than **10pack**
	R	10% are more than **10pack**
InByte	N	71% are more than **500byte**
	R	10% are more than **500byte**
InPac	N	38% are more than **10pack**
	R	10% are more than **10pack**
O/Ipac	N	99% are more than **1**
	R	60% are more than **1**
OB/OP	N	68% are more than **100byte/pack**
	R	30% are more than **100byte/pack**

* N: Normal Application , R: RAT

VII. Conclusion

In this paper, we presented the idea of detecting RATs in the early stage of communication and implemented the process of detection. Our proposed approach relies on the network behavior features which are extracted from the TCP header. Hence, our approach could execute quickly, even possible to detect an unknown RAT while it communicates through TCP protocol. The inherent feature of RATs is that they behave as secretly as possible; they usually have less traffic than normal applications during the preparing period (the early stage) of communication. DT and RF detection models of show better results than other machine learning algorithms in detecting RAT sessions, due to their accuracy of greater than 96% together with their FNR of less than 20%. Future work will include increasing the number of RAT samples in order to further improve the accuracy and reduce the FNR of the predictions.

References

[1] TrendMicro. (2009) *Data Stealing Malware Focus Report* [Online]. Available: http://www.trendmicro.com

[2] ISACA. (2010) *Data Leak Prevention* [Online]. Avaliable: http://www.isaca.org

[3] S. Li, X. Yun, Y. Zhang, J. Xiao, and Y. Wang, "A General Framework of Trojan Communication Detection Based on Network Traces", in *IEEE Seventh International Conference on Networking, Architecture, and Storage*, 2012, pp. 49-58.

[4] K. Borders and A. Prakash, "Web tap: detecting covert web traffic", in *11th ACM conference on Computer and communications security*, 2004, pp. 110-120.

[5] Y. Liang, G. Peng, H. Zhang, and Y. Wang, "An Unknown Trojan Detection Method Based on Software Network Behavior", *Wuhan University Journal of Natural Sciences*, Vol. 18, No. 5, pp.369-376, Mar. 2013.

[6] M. Yamada, M. Morinaga, Y. Unno, S. Torii, and M. Takenaka, "A Detection Method against Espionage Activities of Targeted Attack on The Internal Network", The 31st Symposium on Cryptography and Information Security, 2014.

[7] Y. Yamauchi, J. kawamoto, R. Hori, and K. Sakurai, "Extracting C&C Traffic by Session Classification Using Machine Learning", The 31st Symposium on Cryptography and Information Security, 2014.

[8] Y. Zeng, X. Hu, and K. G. Shin, "Detection of Botnets Using Combined Host- and Network-Level Information", in *IEEE/IFIP International Conference on Dependable Systems & Networks*, 2010, pp. 291-300.

[9] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. Steen, F. C. Freiling, and N. Pohlmann, "Sandnet: network traffic analysis of malicious software", in *the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, 2011, pp. 78-88.

[10] F. A. Vahid, and A. N. Zincir-Heywood, "Investigating Application Behavior in Network Traffic Traces", in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2013, pp. 72-79.

[11] B. Qu, Z. Zhang, L. Guo, and D. Meng, "On accuracy of early traffic classification", in *IEEE Seventh International Conference on Networking, Architecture, and Storage*, 2012, pp. 348-354.

[12] Peter Harrington, "Machine Learning in Action", Manning, 2012, Shelter Island, NY, pp. 28-178.

[13] Kuldeep Singh, and Sunil Agrawal, "Comparative Analysis of Five Machine Learning Algorithms for IP Traffic Classification", in *International Conference on Emerging Trends in Networks and Computer Communications*, 2011, pp. 33-38.

[14] T. Kocak and I. Kaya, "Low-power bloom filter architecture for deep packet inspection", *Communications Letters, IEEE*, Vol. 10, No.3, 2006, pp. 210-212.