PAPER Special Section on Cryptography and Information Security

A RAT Detection Method Based on Network Behavior of the Communication's Early Stage*

Dan JIANG[†a)], *Nonmember and* **Kazumasa OMOTE**[†], *Member*

SUMMARY Remote Access Trojans (RAT) is a spyware which can steal the confidential information from a target organization. The detection of RATs becomes more and more difficult because of targeted attacks, since the victim usually cannot realize that he/she is being attacked. After RAT's intrusion, the attacker can monitor and control the victim's PC remotely, to wait for an opportunity to steal the confidential information. As this situation, the main issue we face now is how to prevent confidential information being leaked back to the attacker. Although there are many existing approaches about RAT detection, there still remain two challenges: to detect RAT sessions as early as possible, and to distinguish them from the normal applications with a high accuracy. In this paper, we propose a novel approach to detect RAT sessions by their network behavior during the early stage of communication. The early stage is defined as a short period of time at communication's beginning; it also can be seen as the preparation period of the communication. We extract network behavior features from this period, to differentiate RAT sessions and normal sessions. For the implementation and evaluation, we use machine learning techniques with 5 algorithms and K-Fold cross-validation. As the results, our approach could detect RAT sessions in the communication's early stage with the accuracy over 96% together with the FNR of 10% by Random Forest algorithm.

key words: *Remote Access Trojan (RAT), machine learning, network detection*

1. Introduction

As data leakage accidents occur every year, the security of confidential information is becoming increasingly important. Remote Access Trojan (RAT) can steal the confidential information, which invades the victim's PC through targeted attacks. There are three main intrusion ways of targeted attack: E-mail, USB and Drive-by-download. Through the targeted E-mail attack, the attacker often pretend to be a legal organization, as well as the content of the E-mail is also related with victim's business. This kind of E-mail is completely different with simple spam-mail. It is hard to distinguish by machine, even by human. Therefore, some of the existing policies like port filtering and Deep Packet Inspection (DPI) [1], [5] are not enough to prevent data leakage problems caused by targeted attacks [7]. 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.

Manuscript received March 23, 2015.
Manuscript revised June 23, 2015.

[†]The authors are with the Faculty of School of Information Science, Japan Advanced Institute of Science and Technology, Nomishi, 923-1292 Japan.

*The preliminary version of this paper was presented at AINA'15 [4].

a) E-mail: jiangdan@jaist.ac.jp

DOI: 10.1587/transfun.E99.A.145

There exist two detection approaches as host- and network-based detection. Host-based approaches run on individual hosts/devices on the network, but network-based approaches monitor traffic between all devices on the network [2]. For example, FireEye [3] is a network-based countermeasure against targeted attacks. However, there are two challenges we still need to face: to detect RAT sessions as early as possible, and distinguish them with similar normal sessions accurately. Li et al. [7] propose a detection system named MANTO, to detect RATs by their original network behavior, which achieves an accuracy of more than 90%. MANTO needs to extract network features from a completely session, but RAT often maintains its TCP tunnel for a long time hence the confidential information might already leak before the detection. Qu et al. [9] propose an idea that uses the first few packets to classify different normal sessions. Although it can handle the process at an early stage, RAT detection is not the object of this research.

In this paper, we set four goals to achieve: 1. network-based, 2. less cost, 3. early detection, and 4. clearly distinction. The reasons are as follows: 1. Network-based approach is for ensuring the convenience of management, as we do not need to install detection applications to each terminal, also do not need to adjust applications while there is any change about the organization. 2. To keep the cost not so much, we just extract network behavior features from the first 58 bytes of each packet. 3. Early detection can let us have enough time to mitigate RATs' malicious behaviors. 4. Because of some normal applications, such as P2P service, behave similarly as RAT sessions on their network communications, it is necessary to divide RAT sessions and normal sessions more clearly. Our method defines an *early stage* of the communication, which is a preparing period of two communication sides such as handshaking and negotiating, to detect RAT sessions from the features of these period. RATs usually confine their communications to low-and-slow network traffic in order to hide themselves inconspicuously. Meanwhile, normal sessions do not need to hide their network behavior, which means that normal 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 behavior. Lastly, the detection model, which is generated in the second phase, is used to detect whether

Table 1 Summary of related researches.

Related Work	Approach	Limitation
LYZXW'12 [7]	detect RATs by network features such as the rate of Outbound / Inbound traffic and the difference of the packet interval time	might be difficult to detect early
LPZW'13 [6]	detect RATs by network & host features such as upload connection rate, concurrent connection rate, process connections number	might be complex for management
YKKS'14 [13]	detects illegal activities of RAT by the SMB & HTTP traffic	detection for a specific RAT
MHJF'12 [8]	detect C&C traffic by network features like TTL (Time To Live)	not for RAT detection
YMUTT'14 [12]	detect C&C traffic by the character that C&C traffic usually less random than normal traffic	
VH'13 [11]	classify normal traffic by network features such as PSH flag numbers	
QZGM'12 [9]	classify normal traffic through the first few packets transmission pattern	

a real session is normal or not. For the implementation, we collect 10 types of RATs and 10 types of normal applications 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 Sect. 2, we discuss some related works of RAT detection. Section 3 gives a detailed description of our proposed method. In Sect. 4, we evaluate the performance of our proposed method. Sect. 5 discusses our idea in more detail. Finally, Sect. 6 concludes the paper.

2. Related Work

There are some network-based RAT detection methods [7], [13]. The common used network behavior features are packet number, data size, and the duration of the session. Li et al. [7] 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 normal sessions since the attackers need analyzing time to decide the next command. Although the detection system [7] 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. The system proposed by Yamada et al. [13] detects illegal activities of spying in the Intranet; the limitation of their system is that it only corresponds with RATs which conduct spying activities with other PCs in the Intranet. Thus, it might be difficult to detect RATs which do not communicate with other PCs in the Intranet.

For a Bot-nets detection, Mathews et al. [8] uses TTL (Time To Live) information as a new feature to detect C&C traffic in Bot-nets. However, whether this feature is valid for classifying RATs is still unknown. Yamauchi et al. [12] 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, since the access frequency of the attacker is less random than that of humans. However, it is still difficult to distinguish RAT sessions from normal 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 [6], [14]. Although 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 the detailed host-based information. This might be not suitable for a large organization.

There are some classification methods of normal traffic. Vahid et al. [11] 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, P2P service sessions set numerous PSH flags during their real-time communication from our observation. Thus, we could not clearly distinguish RAT sessions from normal sessions by this feature. Qu et al. [9] 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.

We summarized these related researches as Table 1.

3. 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 behavior. After learning the relationship between network behav-

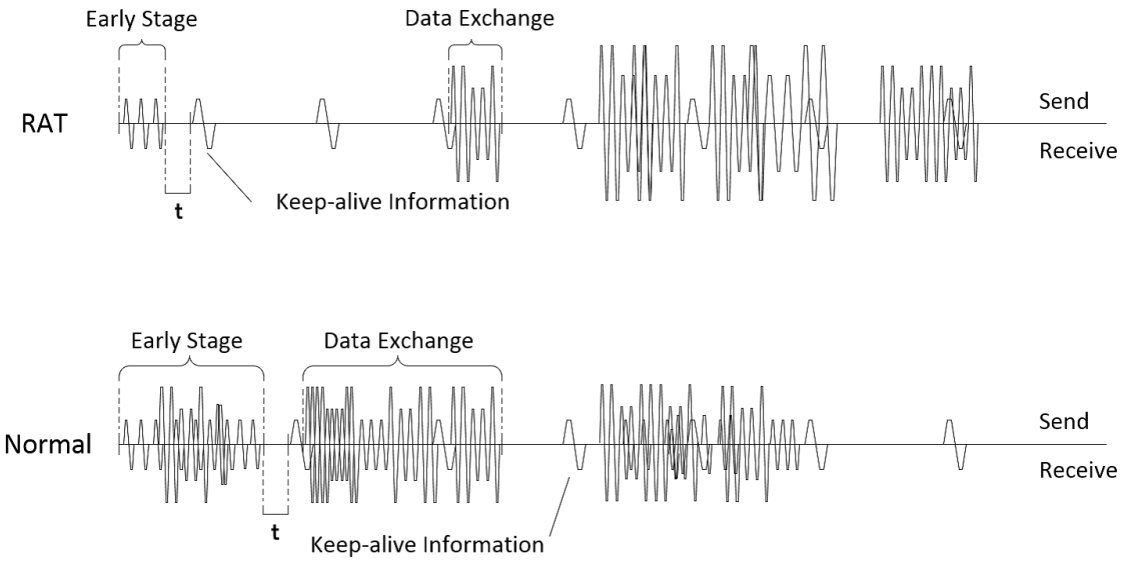


Fig. 1 Image of the data size in the network traffic of the early stage.

ior 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 normal 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 easier to be discovered. Comparing with RATs, normal applications do not need to hide their network behavior. Moreover, normal applications leverage multi-thread techniques in pursuit of a high communicate speed. Figure 1 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, and 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.

3.1 Feature Extraction Phase

At first we choose 5 network features: PacNum, OutByte, OutPac, InByte, InPac based on existing works [6]–[14]. In order to gain more information from these features, we employ 2 more features: O/Ipac and OB/OP described in Table 2. From the preliminary experiment using several features, we found that O/Ipac and OB/OP had the effective ones distinguishing RAT sessions from normal sessions. We collect them from the first 58 bytes of TCP packets in the early stage of a session. After preparing all seven features, we generate a network feature vector of this session.

Table 2 Extracted 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

Figure 2 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, No.1~5 feature variables increase recurrently as addition assignments. After that, there are two ways to continue the process according to the interval time. If it is shorter than the threshold t second(s), then the process reads the next packet. If it is longer than t second(s), No.6~7 feature variables are calculated. Finally, seven features are gathered to generate a network feature vector for the session. This process can be executed for all sessions at once in batches.

3.2 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 (Fig. 3). At first,

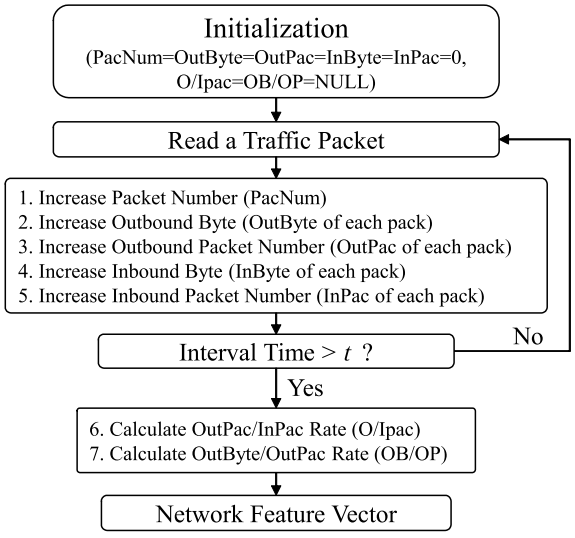


Fig. 2 Process of feature extraction.

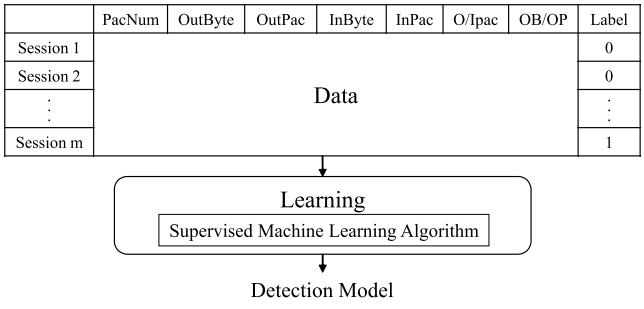


Fig. 3 Process of training.

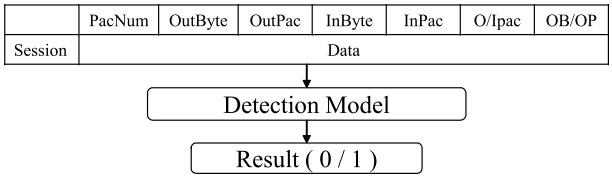


Fig. 4 Process of detection.

we add normal/abnormal labels to every session's vector. The normal class is expressed by a label '0' as normal 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.

3.3 Detection Phase

In the detection phase, the detection model predicts the label of a new session (Fig. 4). The input is a feature vector which has no label information. If the output is '0', it means that this session is normal. Otherwise, this session is established by a RAT process.

Table 3 10 types of RATs and 10 types of normal applications.

RAT or Normal Application	Push or Pull	Keep -alive	Encryption
RAT			
Bandook	Push	Yes	Yes
Bozok	Push	Yes	Yes
Cerberus	Push	Yes	Yes
Gh0st	Push	Yes	Yes
Netbus	Push	No	No
Nuclear	Push	Yes	Yes
OptixPro	Push	No	No
Poison Ivy	Push	Yes	Yes
ProRat	Push	No	No
Turkojan	Push	Yes	Yes
Normal Application			
BitComet (P2P download tool)	Push	Yes	No
BitTorrent (P2P download tool)	Push	Yes	No
Chrome (web browser)	Pull	Yes	No
Dropbox (cloud service)	Push	Yes	Yes
Firefox (web browser)	Pull	Yes	Yes
PPTV (P2P video sharing tool)	Push	Yes	No
Skype (instant messenger)	Push	Yes	Yes
Teamviewer (P2P remote management tool)	Push	Yes	Yes
TorBrowser (anonymous web browser)	Push	Yes	Yes
YahooM! (instant messenger)	Push	Yes	Yes

4. 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.

4.1 Experimental Data

For the experimentation, we collected 10 types of RATs and 10 types of normal applications from the Internet described in Table 3, in which we summarize these RATs and normal applications from the viewpoints of "push or pull", keep-alive and encryption. Especially, as for a normal application, we selected HTTP, HTTPS, P2P and CHAT applications which contribute a significant portion of the total network traffic in the same way as [7]. Furthermore, we took into consideration the viewpoint of push-type communication similar to RAT (RATs and their similar normal applications receive the data from servers through push-type communication), and employed many normal applications with push-type communication including a cloud service. We confirm whether the network behavior features in the early stage can make the differences between RAT and normal sessions clearly. Note that we define "session" as connections in a IP-pair communication like [7]. Their session data is applied to train the detection model.

Table 4 An example of feature vector.

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

4.2 Procedure

4.2.1 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, we divide the traces into sessions. Next, we pick out the invalid sessions and delete them. As an exception, since 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 the 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 RAT sessions and 165 normal sessions.

4.2.2 Feature Extraction

During this phase, we assume that the threshold t is 1 second in our experiment, to verify the result. From the preliminary experiment using $t = 0.5$, 1 and 2, we found that the result by $t = 1$ was the best, hence we used $t = 1$ in this experiment. 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 in Table 4.

4.2.3 Detection Model Training

In order to train the detection model, we use the supervised machine learning algorithms in the scikit-learn which is a machine learning library in Python. We can use Decision Tree (DT), Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), and Random Forest (RF) as a common supervised machine learning algorithm in this library.

4.2.4 Cross Validation

Instead of detecting new sessions, we use the K-Fold Cross Validation (CV) technique to evaluate the detection model. The advantage using K-Fold CV is to evaluate the detection of RATs which are not used for training. Accordingly, this evaluation considers the detection of unknown RATs. Since our experimental results by K-Fold CV are relatively-good, we can guess that unknown RATs are distinguishable from normal applications such as HTTPS, HTTPS, P2P and CHAT. 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:

$$Accuracy = \frac{True\ Detection\ Number}{Total\ Number}$$

$$FPR = \frac{False\ Detection\ Number\ of\ `1'}{Legitimate\ Sample\ Number}$$

$$FNR = \frac{False\ Detection\ Number\ of\ `0'}{RAT\ Sample\ Number}$$

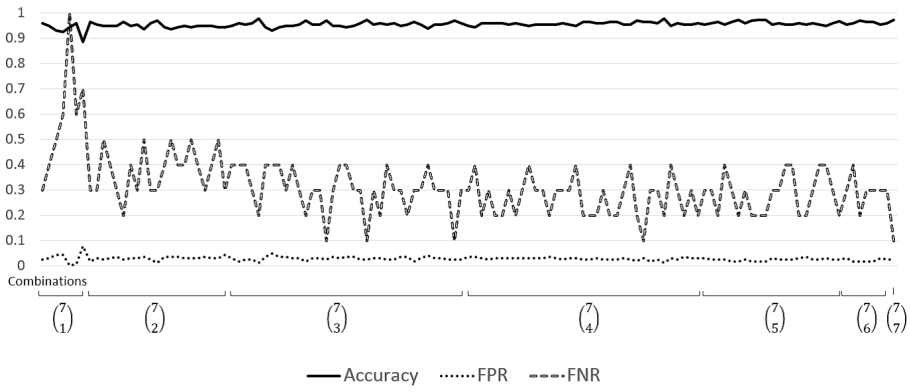
In order to detect any doubtful session in the Intranet, a low FNR is more important than a low FPR in our evaluation.

4.2.5 Result

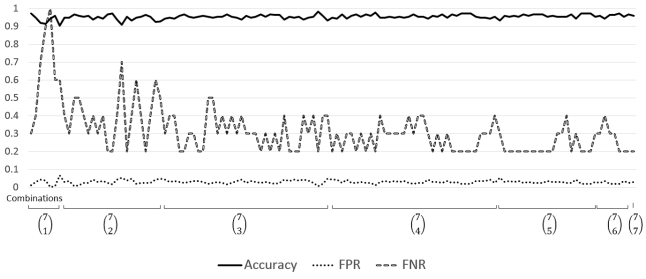
Due to the seven features, we train the detection model of each machine learning algorithm by all feature combinations. In order to verify results of all 127 combinations for 5 algorithms, we output them as Fig. 5. The horizontal axis represents the combinations as 1 element to 7 elements from left to 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 Fig. 5. 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.

We summarized three feature combination patterns that perform better in RAT detection in Table 5. More specifically, three combinations are "OutByte + OutPac + OB/OP" (3-dimensional), "PacNum + OutByte + OutPac + InPac + OB/OP" (5-dimensional), and "PacNum + OutByte + InByte + OutPac + InPac + O/Ipac + OB/OP" (7-dimensional). For this combination, SVM gives the worst result by low accuracy and high FNR. Although NB gives a higher FPR, the FNR is lower. 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%.

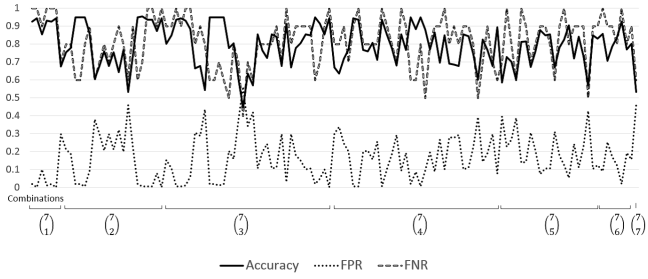
SVM and NB have a smooth and continuous boundary for discrimination, but the tree-based algorithms like DT and RF do not have such a smooth boundary. A smooth boundary has disadvantage; SVM and NB with a smooth boundary may cause false classification near a boundary if dataset has



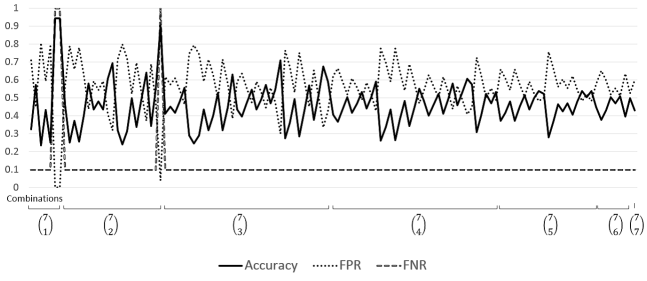
(a) RF



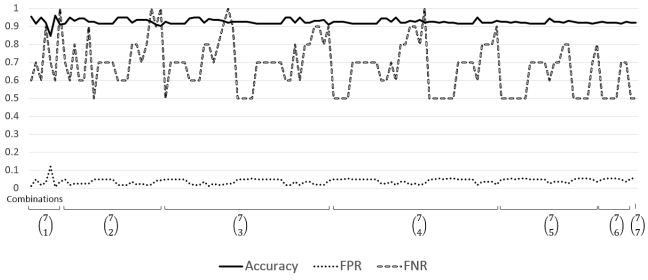
(b) DT



(c) SVM



(d) NB



(e) KNN

Fig. 5 Experimental results of all feature combinations.

Table 5 Performance result of detection model.

Algorithm	Accuracy	FPR	FNR	Feature						
DT	0.954	0.030	0.300	OutByte	OutPac	OB/OP	-	-	-	-
RF	0.954	0.030	0.300							
SVM	0.724	0.244	0.800							
KNN	0.914	0.048	0.700							
NB	0.573	0.446	0.100							
DT	0.966	0.024	0.200	OutByte	OutPac	OB/OP	PacNum	InPac	-	-
RF	0.972	0.018	0.200							
SVM	0.760	0.206	0.800							
KNN	0.914	0.048	0.700							
NB	0.499	0.525	0.100							
DT	0.960	0.030	0.200	OutByte	OutPac	OB/OP	PacNum	InPac	InByte	O/Ipac
RF	0.971	0.023	0.100							
SVM	0.533	0.458	0.600							
KNN	0.919	0.054	0.100							
NB	0.430	0.597	0.100							

a distinct border. In this case, the tree-based algorithms are generally better suited for such dataset. Therefore, we guess that our seven features have a property distinguishing RAT from normal applications more clearly, and that it follows that DT and RF are effective for detecting RATs.

5. Discussions

5.1 The Early Stage

In order to confirm whether features in the early stage can represent network behavior, we compared the outbound data size (OutByte) and the number of packets (PacNum) between RAT sessions and normal sessions in Table 6. As normal applications usually have several sessions during their communications, we use the average values of normal sessions; OutByte and PacNum. Comparing the data size in the early stage, the difference between RAT and normal sessions is clear. However, the data size of Bandook, Nuclear, Turkojan, Chrome, Firefox, and BitComet sessions in the first 100 packets are mixed together in a small range. This is because RATs behave secretly, so the early stage of RAT sessions usually ends before 100 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 normal sessions have greater traffic over the early stage since their early stages are longer than 100 packets. That is why the data size features are easily mixed together in the first 100 packets. Additionally, as for the number of packets in the early stage, the difference between RAT and normal sessions is also clear.

Here, we also give a real example to show the difference in the early stage (Fig. 6), 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.

Table 6 The difference of early stage and first 100 packets.

RAT or Normal Application	Early Stage / First 100 Packets (OutByte)	PacNum in the Early Stage
Bandook	303 / 2319	5
Bozok	308 / 4472	5
Cerberus	516 / 3055	10
Gh0st	355 / 3175	6
Netbus	129 / 74696	5
Nuclear	343 / 2420	7
OptixPro	221 / 1390	7
Poison Ivy	1368 / 3612	33
ProRat	130 / 4583	5
Turkojan	294 / 2977	7
BitComet	65827 / 2677	1895
BitTorrent	1094 / 4271	11
Chrome	6458 / 2498	78
Dropbox	9148 / 10161	52
Firefox	3387 / 2504	64
PPTV	63471 / 3700	2099
Skype	1626 / 1776	31
Teamviewer	5232 / 5623	41
TorBrowser	90435 / 31593	705
YahooM!	18954 / 9888	74

5.2 The Features

Table 7 shows the comparison of seven network 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.

From the table we can also know that outbound features such as OutByte, OutPac, and OB/OP are valid in our detection method, in order to make sure which one is the most valid feature, we visualized the result of decision tree as Fig. 7. In this figure, X[0]=OutByte, X[1]=OutPac, X[2]=OB/OP. As the algorithm of decision tree, during the classification, the first selected feature contains more entropy than other features. The result shows that based on our method, OutByte is the most valid feature to detect RAT sessions.

5.3 FNR

FNR is somewhat high in our proposed method. More pre-

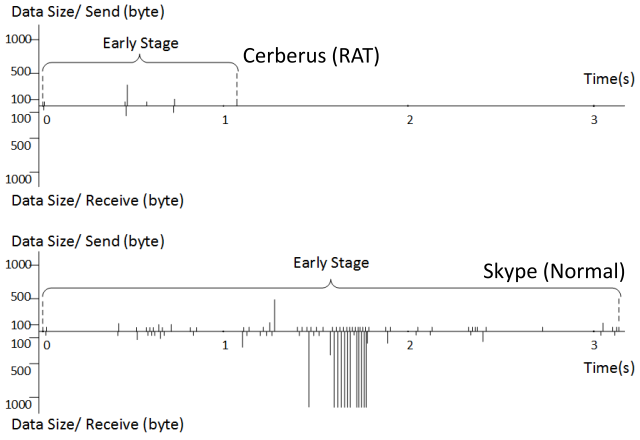


Fig. 6 An example of real traffic data size in the early stage.

Table 7 Compare the features in the early stage.

Feature	Type	Trend
PacNum	N	78% are more than**10packs**
	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

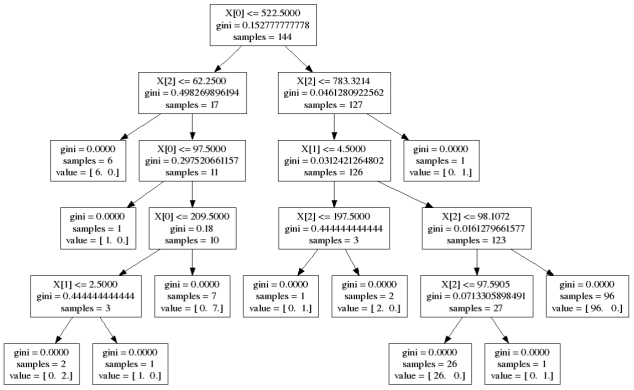


Fig. 7 DT's generation tree.

cisely, our method using RF was not able to detect two kinds of RATs (Poison Ivy and Cerberus) well. We guess that our method cannot detect RATs which is similar to the network behavior of normal applications. For example, the amount of OutByte and the number of packets (PacNum) of these two RATs are larger than the other RATs in Table 6.

5.4 Evasion

If RAT behaves like normal communication in the early stage, it may be able to evade the detection by our method. This is a limitation of our approach. However, this customized RAT does not have the inherent feature that a RAT tries to hide its own trace of communication. This is a disadvantage for RATs whose trace or evidence increases. This may make detection by the other approaches easy. RATs behave as secretly as possible so that it may not be found by a network administrator or the user. In order to confirm it, we summarize the characteristic behavior of 10 kinds of RATs and 10 kinds of normal applications in the early stage in Table 6. We can easily see the tendency for RATs to reduce OutByte and PacNum as much as possible in the early stage from this table. We guess that RATs do not try to behave like normal communication but tend to hide its own trace or evidence of communication in the early stage.

5.5 Toward Real-Time Processing

We aim at a real-time RAT detection. Here, we discuss the possibility of real-time processing from two aspects; (1) the number of packets in the early stage and (2) the processing time of machine learning algorithms. Our method uses only the first 58 bytes of a packet. Also, it uses only the packets of early stage of a session. So, if the number of packets is small, processing becomes efficient and it is effective in real time. Thus, we derived the number of packets of RAT and normal application on average in the early stage in Table 6. As a result, we have only to process 5∼2099 packet headers on average for each session as described in Table 6. Therefore, we guess that a real-time operation could be possible enough for our method to process.

Table 8 The processing time of one session detection (average).

Algorithm	n-dimensional vector		
	n=3	n=5	n=7
DT	4.60ms	4.32ms	4.58ms
RF	390ms	397ms	471ms
SVM	9.00ms	8.90ms	9.30ms
KNN	33.6ms	34.8ms	35.4ms
NB	16.6ms	16.2ms	16.9ms

We measured the processing time during the detection phase for ensuring machine learning algorithms will not cost so much time while detecting. The 3, 5 and 7 features combinations in Table 5 are used in the implementation. Table 8 shows the results of all 5 algorithms' processing time during detection. DT and SVM processes fast that less than 10 milliseconds while RF gives the processing time much longer than others. In the rough estimation using the results of Table 8, as long as the new session is generated less than once on the network during hundreds of milliseconds, a real-time operation using 7-dimensional vector could be possible enough.

Note: Because of the different implementation data, we did not compare the detection result of our approach and approaches of related work.

6. 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] ISACA, "Data leak prevention," http://www.isaca.org, 2010.
[2] N. Das and T. Sarkar, "Survey on host and network based intrusion detection system," Int. J. Advanced Networking and Applications, vol.6, no.2, pp.2266–2269, 2014.
[3] FireEye, "Network Security NX Series," https://www.fireeye.com/products/nx-network-security-products.html
[4] D. Jiang and K. Omote, "An approach to detect remote access Trojan in the early stage of communication," 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, pp.706–713, 2015.
[5] T. Kocak and I. Kaya, "Low-power bloom filter architecture for deep packet inspection," IEEE Commun. Lett., vol.10, no.3, pp.210–212,

2006.
[6] Y. Liang, G. Peng, H. Zhang, and Y. Wang, "An unknown Trojan detection method based on software network behavior," Wuhan Univ. J. Nat. Sci., vol.18, no.5, pp.369–376, 2013.
[7] S. Li, X. Yun, Y. Zhang, J. Xiao, and Y. Wang, "A general framework of Trojan communication detection based on network traces," 2012 IEEE Seventh International Conference on Networking, Architecture, and Storage, pp.49–58, 2012.
[8] M. Mathews, P. Halvorsen, A. Joshi, and T. Finin, "A collaborative approach to situational awareness for CyberSecurity," Proc. 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2012.
[9] B. Qu, Z. Zhang, L. Guo, and D. Meng, "On accuracy of early traffic classification," 2012 IEEE Seventh International Conference on Networking, Architecture, and Storage, pp.348–354, 2012.
[10] C. Rossow, C.J. Dietrich, H. Bos, L. Cavallaro, M. van Steen, F.C. Freiling, and N. Pohlmann, "Sandnet: Network traffic analysis of malicious software," Proc. First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS'11, pp.78–88, 2011.
[11] V.A. Foroushani and A.N. Zincir-Heywood, "Investigating application behavior in network traffic traces," 2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), pp.72–79, 2013.
[12] Y. Yamauchi, J. Kawamoto, R. Hori and K. Sakurai, "Extracting C&C traffic by session classification using machine learning," 31st Symposium on Cryptography and Information Security, 2014 (in Japanease).
[13] M. Yamada, M. Morinaga, Y. Unno, S. Torii and M. Takenaka, "A detection method against espionage activities of targeted attack on the internal network," 31st Symposium on Cryptography and Information Security, 2014 (in Japanease).
[14] Y. Zeng, X. Hu, and K.G. Shin, "Detection of botnets using combined host- and network-level information," 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN), pp.291–300, 2010.

Dan Jiang received her M.S. degree in information science from Japan Advanced Institute of Science and Technology (JAIST) in 2015. Her research interests include machine learning and network security, specially for preventing the data leakage caused by targeted attacks.

Kazumasa Omote received his Ph.D. degrees in information science from Japan Advanced Institute of Science and Technology (JAIST) in 2002. He worked at Fujitsu Laboratories, LTD in 2002–2008, and was engaged in research and development for network security. He has been a research assistant professor at Japan Advanced Institute of Science and Technology (JAIST) since 2008, and has been an associate professor at JAIST since 2011. His research interests include applied cryptography and network security. He is a member of IEICE and IPSJ.