2017 ACM/IEEE Symposium on Architectures for Networking and Communications

Detecting Remote Access Trojans through External Control at Area Network Borders

Shuang Wu
State Key Laboratory of MEAC
No.62 Science RD, High-technol.
Zone, Zhengzhou, China
8615838308803
wushuang10@tsinghua.org.cn

Shengli Liu, Wei Lin, Xing Zhao, Shi Chen
State Key Laboratory of MEAC
No.62 Science RD, High-technol.
Zone, Zhengzhou, China
{liu05.meac, buzztiger, gzhjzx, ccccsober}@gmail.com

ABSTRACT

The Remote Access Trojan (RAT), whose exposure often lags far behind its widespread infection, plays a part in the growing number of cyber-attacks. In terms of intrusion detection, signature-based methods still occupy the dominant position together with anomaly-based methods that are deployed to be complementary. The anomaly-based methods are efficient and resource saving; however, anomaly-based RAT detection mainly utilizes machine learning on collective features without considering packet sequences. In addition, imbalanced data has a negative impact on machine learning classification. Accordingly, we analyzed packet sequences from various applications and found a general correlation between the packet direction sequences and the external control behaviors of RATs.

This research presents a novel framework for tracking external control at the borders of area networks. We extract packet payload-size sequences, inter-arrival time sequences and packet direction sequences of IP flows. Then, RAT network behavior is exposed in the packet direction sequences of flow slices, which are generated by the inter-arrival time sequences. Naïve Bayes is utilized for classification, and a frequent sequence mining algorithm is implemented to eliminate noise. After deploying the device, we detected all the RAT sample sessions. Our method achieves a false positive rate of less than 0.6% on real-world campus network data, which demonstrates its efficacy.

CCS Concepts

• **Security and privacy → Malware and its mitigation;** *Intrusion detection systems; Network security;* Firewalls
• **Networks** → Network measurement

Keywords

Traffic data analysis; Remote Access Trojan; External control

1. INTRODUCTION

Trojans have a history spanning several decades that includes variants such as backdoor, rootkit, dropper, key logger, virus, ransomware, etc. In contrast to early Trojans, which were largely benign, modern Trojans can cause enormous societal damage nowadays, solely to fulfil an attacker's intent. Since it is very difficult to detect all Trojans using only signatures and the specific characteristics of the IDS, in this paper, we focus on only one type of Trojan: the Remote Access Trojan, commonly called RAT, because it is capable of data exfiltration, typically under human control from outside an area network. This scenario often occurs under beginning or ongoing attack conditions.

In recent years, APT attacks have attracted significant attention. An APT is an attack that can evade most network security defenses and can hide in infected targets for a very long time. Eric *et al.* [1] constructed a kill chain model to provide a better understanding of APT attacks and defenses. This model consists of seven stages—reconnaissance, weaponization, delivery, exploitation, installation, command and control (C&C) and actions on objectives. RATs play roles in issuing commands at the sixth stage and receiving data at the final stage.

Trojans have existed for many years. The privilege policies of Unix-like operating systems and the open source features of Linux make it easier to change some important system files to generate a rootkit [2]; therefore, Linux Trojans have a history nearly as long as Linux. The first RAT in Windows was the back orifice [3] discovered in 1998. This work focuses mainly on Windows RATs.

Overall, there are five methods for detecting malicious code, and they can be grouped into two large categories: host-based methods and network-based methods, as shown in Table 1.

Host-based methods include both signature detection and behavior detection. Host-based behavior detection utilizes sandbox or driver hooks to detect sensitive behaviors such as registry editing or port hiding when a malicious process is first loaded. However, the resource consumption of this method could cause users to shut down the detection tool to avoid performance degradation. Moreover, host-based signature detection can detect only old samples because attacks are polymorphic.

Table 1. Five methods for detecting RATs.

Categories	Subclass	Applied to RATs Detection
Host-based	Signature	Static, detect binary
	Behavior	Sandbox, dynamic detect
Network-based	Signature	Static, detect application layer
	Anomaly	Statistics of traffic flow
	Protocol	Nonstandard protocol

Network-based methods include signature detection, anomaly detection and protocol detection. Network-based signature detection, namely misuse detection, detects malicious traffic packets through known signatures and DPI technology. Since Snort [4] was proposed, it has become the base component of many large projects such as Einstein 2. The protocol method, used generally for HTTP or HTTPS, detects the protocol state that does not conform to a standard. Anomaly detection has been shown to have good effects on botnet network detection, with the development of big data and neural networks in recent years [5; 6]; however, it is still rarely deployed in industry, especially for RAT detection. Due to the limited quantity and quality of RAT samples, we develop a system that can usually detect hidden RAT communications by means of recognizing data exfiltration typically under external control. The system can efficiently recognize all the tested RATs and can find active RATs in campus network traffic samples with a false positive rate below 0.6%.

The contributions of our work are as follows:

1. We propose a general external control and data exfiltration detection method and demonstrate its effectiveness in human-controlled RAT session detection.

2. Traditional machine learning methods often have special integrity requirements for traffic flows or must capture a flow from the beginning to classify it correctly. Because our method detects humanly controlled RAT characteristics, which are not totally dependent on training, it works on continuous traffic fragments and does not require complete start or end flags.

3. Our proposed method can efficiently find RATs from real traces with a low false negative rate.

The remainder of this paper is structured as follows. Section 2 summarizes the related work on traffic classification and RAT session detection. Section 3 presents the background for RAT session analysis and RAT features. Section 4 outlines our approach, including details on algorithms, datasets, experiments and evaluations of results. Section 5 discusses the false alarms and evasion, while Section 6 concludes and discusses future work.

2. RELATED WORK

Network Intrusion Detection Systems (NIDSs) have existed for more than 30 years. Although misuse detection and anomaly detection have both been studied for many years, a striking imbalance exists in terms of actual deployments; signature-based methods are more widely used than anomaly detection. One reason is that signature-based methods have low latency. Sommer *et al.* [7] noted that operational deployment focused more on traffic statistical profiles than on the generality that the papers envision, which means that researchers should consider more actual situations in their research regardless of which methods are being used. Despite this imbalance, because anomaly detection can function even with unknown and encrypted traffic, which has been a mainstream evolutionary trend of malware [8], we chose this method to analyze and detect RATs.

First, we introduce some typical traffic classification methods that utilize machine learning. Then, we provide an overview of hybrid methods that can be applied to detecting RATs.

2.1 Machine Learning Methods

Malicious traffic detection methods based on machine learning all arise from one key point—that malware traffic can be strictly classified by analyzing some of its flow features. Often, these features belong to a subset called Moore discriminators [9]. These methods are more concerned with good performance than with feature interpretations [10], although most of the work has focused on feature selection.

Lim *et al.* [11] used the k-means algorithm to divide traffic flows into clusters, which were then tagged by different letters. During the training process, flow letters from the same malware were organized into a sequence ordered by their arrival times. Then, a sequence alignment algorithm was applied to classify different types and find malicious flows. The results showed that new types could be appropriately classified into malware families and their variants. The authors also warned that their experiments were preliminary, that the number of malware samples needed expanding and that their classification algorithms needed refining.

Lakhina *et al.* [12] used entropy as a measure to describe the dispersal and concentration degree of IP numbers and port usage over time. They reported that analysis of isolated entropy points over time could reveal when attacks occurred. Their study collected flow data from two backbone networks: Abilene with 11 PoPs and Géant with 22 PoPs. Thus, they acquired 121 OD flows from Abilene and 484 from Géant. Multiway subspace and k-means were leveraged to automatically detect anomalies from the flow data.

Bernaille *et al.* [13] found that the sizes of the first five packets of TCP connections were sufficient to distinguish different applications. The author used unsupervised methods for clustering TCP flows and developed DPI tools to match applications with the flow clusters. Notably, the authors reported that a large majority of short flows in the traffic might cause bad effects, not only to his work but also to previous classification techniques. Experiments showed that subdominant applications, such as POP3, could be classified correctly only after considering additional server port information.

Similarly, Wang [14] transformed the first 1024 bytes of the application layer payloads to image pixels used as input for a deep neural network. This approach performed well in feature learning and protocol classification. At the same time, the method could even find unknown protocols and anomalous protocols. However, at the Black Hat conference in 2015, the author declared that the system was not good at recognizing completely encrypted payloads.

Louvieris *et al.* [15] simulated a small office environment and divided attacks into four classes through using their attack effects; for example, the Attack 1 class represented a single source with a single request. First, 20 features of the TCP/IP headers were leveraged to perform k-means clustering. Then, evaluated subsets of the feature set were generated separately using a wrapper-based approach, naïve Bayes and the Kruskal-Wallis H test. These three subsets were applied to the C4.5 decision tree to classify attack packets. The results showed that the selected features can provide a high degree of accuracy; however, the tested attacks were defined by the researchers themselves.

Above we have mentioned some classic works on traffic classification. The majority of these works notably combined the machine learning methods with the sequences, such as the cluster flow sequences [11], the packet payload-size sequences [13], and the byte sequences of packet payloads [14]. The paper [15] worked on feature selection, while the paper [12] detected isolated points over time. Though these methods have not detected RAT sessions, we can still infer that sequences are vital to RAT session detection, which will be discussed subsequently.

2.2 Hybrid Methods for RATs

Anomaly-based malicious detection methods have not been widely deployed in real industry environments. Additionally, research into anomaly-based detection of RAT sessions is rare and limited to samples, even when a RAT session has obvious features that distinguish it from normal traffic data. Machine learning is still the mainstream method for identifying RAT sessions, although some other special features of RATs are typically considered to improve the results.

Jiang *et al.* [16] found that there fewer packets occurred at the early stages of RAT sessions. Normal applications tend to perform configuration steps initially to prepare for their upcoming work, while RATs tend to restrict their communications initially to remain inconspicuous. The author extracted 7 features from the TCP headers of packets at the early session stage to perform training and classification. Five supervised methods were applied and their performances were evaluated. The C4.5 decision tree and random forest models achieved better results than the other tested machine learning methods. Adachi *et al.* [17], following Jiang's work, associated the sessions with processes to extract 2 additional features for machine learning and achieved a better result in every host.

Li *et al.* [18] reported that existing network-based methods (such as Snort) are not suitable for detecting RATs among vast amounts of network traffic. Therefore, they proposed a system called Manto. They extracted four features not only from the flow-level but also from the IP-level traffic features. Then, they utilized k-means methods to perform clustering and, finally, proposed maximum likelihood estimation to tag different clusters as normal or abnormal. The Manto system achieved a good performance on real-world traces. However, the author emphasized its disadvantages in real-time detection.

Pu *et al.* [19] comprehensively analyzed RAT network behaviors and designed a multi-logic linear system to detect RATs. They used a keep-alive detector and a master-slave connection detector in parallel for first-layer detection. Additionally, they leveraged a mistake detector applied to judge whether the download/upload rate was normal or abnormal as second-layer detection. This method uses some outlying RAT features that are not comprehensive and cannot represent all RATs.

Yamada *et al.* [20] questioned Manto's methods and instead proposed improved SMB-based methods to detect RATs that had penetrated an Intranet. However, the intranet penetration process used in the experiments was created by the researchers; they achieved no real-world results.

Vectra Network [21] developed a system to detect external human control of traffic flow sessions. A flow session is divided into rapid-exchange instances and dormancy periods. The start times, the end times and the lengths of the dormancy periods can determine whether a session is software-driven or human-driven and how many humans or robots are involved in the session. The time sequence methods of this patented approach motivated our work.

Compared with the papers [11-15], the papers [16-20] on RAT detection have not considered the sequences. Overall, only the paper [18] conducted tests on real–world traces, while it could not detect RATs in real time. The patent [21] first introduced the packet sequences into detecting external control; however, the method ignored heartbeats in the sequences. Comprehensively, we provide a framework to analyze packet sequences using naïve Bayes and filter the heartbeat patterns through frequent sequence mining. Also, the real-world results show that our method has capability of immediately detecting RATs through external control.

Table 2. User common behaviors in campus traffic flows

User behavior	Remarks	Protocol	Examples
Resource down/upload	Data transport	TCP	FTP, Thunder
Multimedia	Online game, video, and more	UDP, TCP	MMS, RTSP
Website visit	Ajax	TCP, UDP	HTTP, HTTPS, DNS
Cloud storage	Sync	TCP	Dropbox, SkyDrive
Remote access	Connect to special server for work	TCP	TeamViewer, SSH
P2P-like interaction	Peer-to-peer	UDP, TCP	QICQ, Skype, BitTorrent
Mail	Mail servers, transfer agents	TCP	POP3, SMTP

3. BACKGROUND

3.1 Trojan and Normal Applications

Traffic flow data includes everything that flows through the network. We intend to understand the Internet and provide a tough guard at the campus network perimeter; therefore, a comprehensive model of internet use is essential. Traffic flows are divided by their behavior patterns and effects into the following classes: resource download, multimedia, website visit, personal cloud storage, remote access, P2P-like interactions, and mail, as shown in Table 2.

The taxonomy is sufficiently broad that it can contain all the scenarios completely, but there are also overlaps between some classes. Regardless, it reveals common behaviors of campus net users. People seldom configure a server for outside clients behind a NAT. If we ignore the servers in the DMZ, we will find that most of the connections are controlled by humans inside the campus, and some instant messages or P2P downloads could show equivalent human control from both sides. Therefore, we can infer that when a session between an internal host and the Internet shows signs that human control occurs completely from the outside-in, the internal host is likely compromised.

Data flows are intended for data transfer. UDP is widely used in instant messaging and quick transmissions, where a strict packet sequence is not required, while TCP is still the primary method for intact file transmission. A major feature of RATs is that file exfiltration also relies on TCP. Therefore, we focus mainly on the TCP protocol. Moreover, because protocols not based on the standard TCP/IP protocol stack can usually be easily detected by other methods, we focus only on the standard TCP protocol.

Regardless of the protocol, in most real-world situations, an IP flow represents a complete session between a server and a client over a period of time, especially those of large web applications with multiple IPs for CDN or load balancing. Some small websites may share the same public IP among different domain names. Usually, users seldom request large traffic volumes for

Table 3. RAT session features

Trojan	Connection	Heartbeat	Multi-TCP stream	Outside first push
DarkstRat	Short	No	No	Yes
SpyNet	Long	Bidirectional	No	Yes
Synrat	Short	No	No	No
Darkmoon	Short	No	No	No
Byshell	Long	Unidirectional	No	Yes
Cmdrat	Long	Bidirectional	Yes	Yes
Pcshare	Long	Unidirectional	Yes	No
Nuclear	Long	Unidirectional	Yes	No
Xdoor	Long	Unidirectional	Yes	No
Gh0st	Long	Unidirectional	Yes	No

these websites, and when one is compromised, the others are as well. Therefore, we consider the traffic data from these websites as a single IP flow. We analyze all traffic data in the form of IP flows.

Considering all of the above, we manually inspected the IP flow sessions of 10 publicly available RATs based on their TCP traffic, as shown in Table 3.

In Table 3, the connection column denotes whether the sessions are long or short connections, in which either the TCP keep-alive or an application heartbeat is used to maintain the connection over longer periods of time. RATs prefer to design their own heartbeats rather than use the default TCP keep-alive mechanism. The multi-TCP stream column represents whether a new TCP connection is initiated from the compromised host for detailed interactions when a new command arrives. There are also some applications that strictly distinguish the upstream and downstream traffic of different TCP flows, such as TeamViewer. The TCP sessions in an IP flow can be complicated. The column labeled outside first push shows a scenario in which the host outside the area network first pushes the packets with a PSH flag after an internal initiation of a three-way handshake. This type is rare in most normal application sessions, but occurs in an incomplete human-controlled protocol—SMTP.

Some old or small applications do not have heartbeats to keep them alive. The internal host reissues a three-way handshake after receiving packets with an RST flag. In this scenario, we choose the longest TCP sessions to represents the full IP flow. In other scenarios, we consider all the TCP flows in an IP flow as a whole. Then we imitate patent [21] by cutting the IP flows into flow slices based on the packet arrival time intervals and analyze the packet direction sequence statistics from each flow slice.

Before this, however, we need to filter out the controlling packets that perform no data transmission. This process will be illustrated in subsections 3.2 and 3.3.

3.2 TCP Control Packets and Keep-Alive

Based on the standard TCP specification, a considerable portion of the traffic involves control mechanisms. These packets have SYN, FIN, or RST flags or only an ACK flag and are used to connect, close, and reset connections or acknowledge packet arrivals. In these packets, the size of the application layer body is always zero because they are generated by TCP. TCP keep-alive is also a TCP default setting used by some applications that carries a default one-byte payload of "0x00". We remove these two types of packets from the IP flows. Then, we can process only those IP flows generated by the applications.

Currently, in our research, we filter out all the TCP control packets and TCP keep-alive packets by removing packets whose payload size is 0 or 1. Subsequently, we obtained a packet payload size vs. timeline 2D plot, as shown in Figure 1.

Some repetitive values or sequences, called packet patterns, exist in the packet inter-arrival times and payload sizes as can be seen in the Byshell and TeamViewer traffic flow data in Figure 1. These patterns reflect the application heartbeats, which will be presented in subsection 3.3. Additionally, if we leave the heartbeats aside temporarily, a common phenomenon will appear

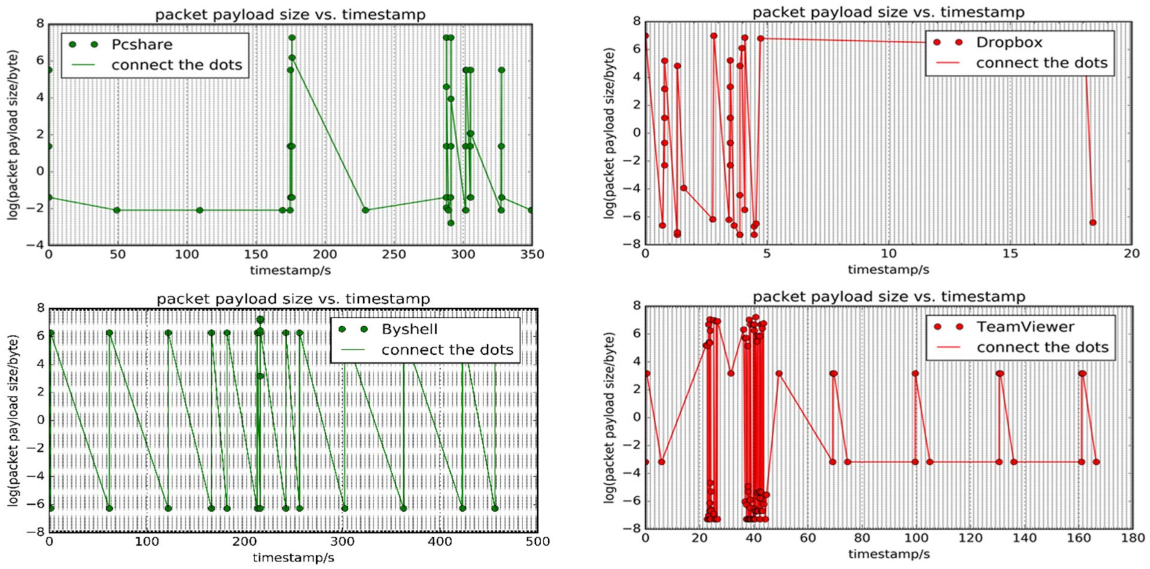


Figure 1. Packet payload-size sequence of time. Images colored green on the left depict traffic from two RATs—Byshell and Pcshare, and images colored red on the right depict traffic from two normal applications—TeamViewer and Dropbox. A log process is applied to the packet payload sizes and the out-to-in packet sizes are defined as negative numbers on the y-axis. Note that the zero-size payloads (control packets) are ignored.

that the direction of the first packet after a dormant interval differs between the RATs and normal applications. This aspect will be discussed in subsection 3.4.

3.3 Application Heartbeats

When we cut the IP flow into flow slices using the time intervals, the beginning of a slice depends mainly on two aspects: one is the heartbeat interval, and the other is the time at which random user commands arrive. For example, in the TeamViewer data shown in Figure 1, the intervals and payload sizes of the heartbeats tend to be fixed: they follow a repeating pattern that often contains fewer packets than TCP handshakes (i.e., three packets).

Based on the conclusions above, we searched for frequent sequences in the packet size sequences of the flow slices that are less than 4 items, and we removed those that formed the same interval pattern of packets in each flow slice only once. Applications, especially RAT requests, can also follow patterns similar to heartbeats, and the method mentioned above is efficient at avoiding erroneously cutting them.

Then, the packet direction sequence of a flow slice shows whether the slice is controlled from an external or internal source. All the flow slices of a flow denote whether the flow is normal or controlled externally through a RAT.

3.4 Packet Intervals, Directions and Sizes

Packet arrivals have been a research topic for many years. In 2004, Uhlig *et al.* [24] concluded that arrivals below second-level intervals were memoryless, while second-level intervals were correlated, and minute-to-hour-level intervals showed self-similarity.

In our research, this finding means that we do not need to consider intervals below the second-level or above the hour-level. Therefore, we cut the IP flows into flow slices through the arrival time intervals and ensure that the arrival interval of two adjoining flow slices is above the second-level.

Most of the time, one flow slice comprises one interaction. We accounted for the direction sequence distribution frequency in both benign and RAT flow slices and utilized naïve Bayes and statistics to classify unknown IP flows.

The occasions could be different for servers that automatically delay dispatching parts of a resource and balance visits to improve the user experience, especially those that serve webpages based on HTTP, which can be delayed 2~10 seconds because of AJAX. Therefore, we accounted for the flow slices only when the uploaded data was larger than the downloaded data in a flow slice and regard these as potential RAT flow slices.

4. METHODOLOGY AND EXPERIMENTS

4.1 Traffic Data Description

During experiments, we manually generated RAT traffic data by executing 24 publicly available RAT programs based on TCP and handling some file transmission operations. Because there is no standard evaluation for RAT traffic detection, previous literature mixes benign traffic flow data with RAT traffic flow data to calculate false positive and false negative rates. This strategy causes two problems. The first is that we cannot ensure a scientific balance between benign data and RAT data. The second is that we could not discern from the benign background traffic how benign it is overall. In our methods, we regard the false negative rate of benign data to be zero, and we calculate only the false positive rate of the test set. Then, we utilize part of the DARPA 1998 traffic to simply verify our approach. The results show that 17 of the total 18 TCP attacks trigger an alert; the non-triggering exception is a dictionary attack. Next, we tested our methods on other real-world traces to determine some useful information. The data that we use is summarized in Table 4 and will be introduced in subsections 4.1.1, 4.1.2, and 4.1.3.

Table 4. Data description

Apply	Data	Time	Real-traces	IP flows
evaluation	Benign data	1week	lab	43321
	RAT data	—	×	24
verification	DARPA 1998	1day	×	1781
Real-traces	WITS 2007	0.5day	campus	340806
	WITS 2011	1day	campus	1200257

4.1.1 Benign traffic and RAT traffic

Similar to other methods, we also generated traffic flows to evaluate our method. All the traffic was familiar regardless of whether it was benign or RAT. Flow slice direction sequence statistics of this traffic will be used to classify the flows.

4.1.1.1 Benign traffic flow

It is unrealistic to capture all the traffic of all the applications separately in a normal environment, because most of the servers are so large that we cannot rebuild. Even the Microsoft Message Analyzer cannot capture all the packets by checking the process names, because some of the packets are returned directly to the kernel instead of to the process. Thus, we captured network traffic at the border of our lab gateway for 1 week continually to generate benign traffic. The captured data contains 43321 IP flows and reaches 1960 MB. Each packet are truncated to 95 bytes. Similar to in other studies in the literature, this traffic is regarded as benign flows and includes traffic data generated by common applications such as Dropbox, TeamViewer, Slack, Secure Shell, and others. We regard an IP flow as an application session.

4.1.1.2 RAT traffic flow

RATs often have paired controllers and controlled sides. We deployed the RAT-controlled side in a virtual environment inside the campus network border and executed the controller program on a VPS outside the campus network. Then, we performed some file transmission operations to generate traffic data. The IP flow of a real RAT is also regarded as an application. In addition to the RATs listed in Table 3, we used 14 other RATs to generate the RAT traffic data: Dute, Fbi, Forshare, HackLegend, HongheiRat, Ruser, DoubleEleven, RatFlooder, YuanxiaoFestival, Renwoxing, XimenRat, YiLanguage, Star, and Remoteghost.

4.1.2 DARPA 1998 training set

The DARPA 1999 data is inadequate for use in this study because it contains inside attacker traffic data. In our methods, inside attackers are no different from normal network users inside the LAN network.

Therefore, we used the training data from Thursday of the sixth week from the DARPA 1998 dataset, which includes 36 attacks. The data was not generated from real traces; it was generated in a controlled environment. The first SYN packet direction of most of the IP flows was from out-to-in, which is not typical of real network situations.

4.1.3 *Some other real-world traces*

To complete our methods, we performed research on real traffic that was captured on the border of the campus or enterprises. On the Internet, some existing traffic data datasets for IDS research exist. Because our work required some of the TCP and IP field information, a packet-level storage format is adequate.

4.1.3.1 *WITS data*

The Waikato Internet Traffic Storage, called WITS, is a contiguous packet header trace captured at the border of the University of Waikato network. The IPs are anonymized by a prefix-preserving anonymization method: the Crypto-Pan AES encryption, and the encryption key is weekly updated. The packet records are truncated to four bytes after the end of the transport header. We obtained the files *20070609-000000-0.gz* of Waikato V and *20110407-000000-0.gz* of Waikato VIII from their site to conduct our analysis.

4.2 Extracting and Processing

4.2.1 *Quick judgments*

Usually, the RAT-controlled side automatically connects to the controller. When a connection cannot be established, SYN packets will be sent continuously. On other occasions, RST and FIN packets will be sent continuously, forming a flooding attack. At the quick judgment stage, we calculated the proportions of control packets (defined in subsection 3.2) among all the packets sent in the same direction. The notation $N_{ctl-pkts}$ describes the number of the control packets sent in a specific direction, while $N_{ctl-direction-pkts}$ describes the total number of packets sent in the same direction. When the IP flow satisfies Formula (1), we judge that IP flow to be a malicious flooding attack and do not consider it in our subsequent research:

$$\frac{N_{ctl-pkts}}{N_{ctl-direction-pkts}} > PR_{ctl}. \qquad (1)$$

We set PR_{ctl} to 0.8. A complete TCP flow, which contains the connection initiation and termination, has at least 7 control packets with 3 packets in one direction and 4 packets in another direction. If this flow transfers actual data in both directions, the Formula (1) will never be satisfied.

4.2.2 *Flow slicing*

As described in subsection 3.1, before slicing the IP flow, we must first remove the packets that satisfying the following condition (2):

$$Length_{pkt-payload} = 0 \; or \; 1, \qquad (2)$$

where $Length_{pkt-payload}$ describes the size of a packet payload. Then, we obtain a five tuple of <*flow-id, timestamp, length, is-upload, seq-number*> for every packet in an IP flow.

Flow-id, a tag of a flow, is calculated using the source and destination IP. We define the internal IP of a flow as the source IP.

Timestamp denotes the time relative to the time of the first IP flow packet.

Length is the payload size of a packet. We define the source-to-destination (or upload, or in-to-out) packet size as a positive number, while those in the opposite direction are negative numbers.

Is-upload denotes whether the packet direction is out-to-in. If so, the value is 0; otherwise, it is 1.

ALGORITHM 1: Flow slicing Algorithm

```
input: A list, flow[]
output: Nested list, flow_slice[]
1  max_t = max(interval) in flow
2  last_i = 0
3  for i in range(1,len(flow)):
4    if $timestamp_i - timestamp_{i-1} > t_{hr}$:
5      if $seq\_number_i$ is not repeated:
6        if $timestamp_i - timestamp_{i-1} > t_{ht}$:
7          flow_slice.append(flow[last_i:i])
8          last_i = i
9        else if $timestamp_i - timestamp_{last\_i} > max\_t$:
10         flow_slice.append(flow[last_i:i])
11         last_i = i
12       else if $i - last\_i > 1$:
13         $\Delta time = \frac{SCALE \times (timestamp_{i-1} - timestamp_{last\_i})}{i-1-last\_i}$
14         if $timestamp_i - timestamp_{i-1} > \Delta time$:
15           flow_slice.append(flow[last_i:i])
16           last_i = i
17         End
18       End
19     End
20   End
21 End
```

Seq-number is the TCP sequence number used for transmission control. We use it to eliminate some of the retransmission packets when the network performance is poor.

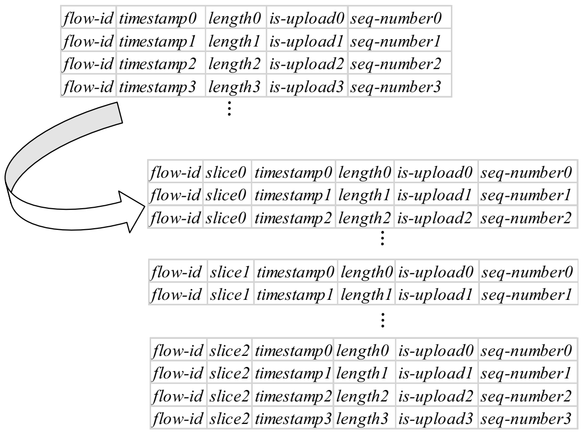
Currently, we need to cut an IP flow into several flow slices through packet arrivals. According to subsections 3.3 and 3.4, the time interval between two adjoining flow slices should be at second-to-minute-level. Therefore, during the slicing phase, a human reaction time threshold—t_{hr}, is set to 0.8s to recognize two adjoining packets, which possibly belong to different flow slices. Then, there are 3 alternative conditions. First, the interval is beyond a mortal think-time threshold—t_{ht}, which is empirically set to 3s. Second, the newest flow slice duration is longer than the maximum interval of the flow. Third, the interval is far longer than the average interval of the newest flow slice, and the scale threshold—SCALE, is set equal to 10. Any of the three conditions can be a determining condition to generate flow slices.

The pseudo-code of our flow slicing process is shown in Algorithm 1. Thus, we divide the packet sequences of an IP flow into several flow slices, as shown in Figure 2. After this operation, we obtain a six-tuple consisting of <*flow-id, slice-id, timestamp, length, is-upload, seq-number*>.

Because the IP flow we analyze might not contain the complete beginning interaction of the first flow slice, we ignore the first slice in the subsequent stage.

4.2.3 *Filtering heartbeats*

Looking back at Figure 1, we can find that on most occasions, the slices obtained using the procedure in section 4.2.2 mainly represent human interactions or the heartbeat patterns of applications.



flow-id	*timestamp0*	*length0*	*is-upload0*	*seq-number0*
flow-id	*timestamp1*	*length1*	*is-upload1*	*seq-number1*
flow-id	*timestamp2*	*length2*	*is-upload2*	*seq-number2*
flow-id	*timestamp3*	*length3*	*is-upload3*	*seq-number3*

⋮

flow-id	*slice0*	*timestamp0*	*length0*	*is-upload0*	*seq-number0*
flow-id	*slice0*	*timestamp1*	*length1*	*is-upload1*	*seq-number1*
flow-id	*slice0*	*timestamp2*	*length2*	*is-upload2*	*seq-number2*

⋮

flow-id	*slice1*	*timestamp0*	*length0*	*is-upload0*	*seq-number0*
flow-id	*slice1*	*timestamp1*	*length1*	*is-upload1*	*seq-number1*

⋮

flow-id	*slice2*	*timestamp0*	*length0*	*is-upload0*	*seq-number0*
flow-id	*slice2*	*timestamp1*	*length1*	*is-upload1*	*seq-number1*
flow-id	*slice2*	*timestamp2*	*length2*	*is-upload2*	*seq-number2*
flow-id	*slice2*	*timestamp3*	*length3*	*is-upload3*	*seq-number3*

Figure 2. Flow slicing diagram.

Because the heartbeats are generally used to keep a connection alive rather than to transmit data, we can filter them and preserve only the pure application data interaction behaviors of IP flows.

To do this, we apply a method called frequent sequence mining to the packet lengths of the flow slices.

To minimize the time complexity of the calculation, we assume that the number of packets involved in one heartbeat will never be larger than the number required for a three-way handshake. Therefore, when a frequent sequence contains 4 or more items, we do not filter it.

Immediately after a frequent 1-sequence is identified, we can easily determine whether the following condition is satisfied:

$$N_{1\_item} \geq \frac{4N_{slice}}{\lceil N_{slice} \times support \rceil}, \tag{3}$$

where N_{1_item} is the number of frequent 1-items, N_{slice} is the number of flow slices obtained above, and *support* is the frequency threshold of the frequent sequences shown in the slices. *Support* should be determined by the composition of actual data packets and heartbeats in the flows. A high frequency threshold could cause missed detection, while a low threshold could cause false detection and could significantly increase compute complexity. In our experiments, the frequency threshold was set to 0.65 empirically. Finally, we take only the maximum-length frequent sequences as the heartbeats. For each slice, we count and remove the same frequent sequence only once. In the calculation, we also apply the restriction that in one heartbeat, the two serial packets must be in different directions.

4.2.4 Direction statistics

4.2.4.1 Naïve Bayes

Now that we have obtained the actual data transmission flow slices of IP flows, we define the packet direction sequence of the flow slices as event *A*. The two frequencies of event *A* — $P(A|S_{RAT})$ and $P(A|S_{benign})$, calculated separately from RAT traffic and benign traffic, are used to estimate the prior probability shown in Table 5. We count only the first 3 bytes of a flow slice.

We consider an unknown flow slice with event *A* to be an external control slice when the following condition is satisfied:

$$P(S_{RAT}|A) > P(S_{benign}|A), \tag{4}$$

Table 5. Frequencies of event *A* in the flow slices

Event *A*	**RAT**	**benign**
'0'	0.059	0.155
'1'	0.119	0.119
'00'	0.030	0.042
'11'	---	0.033
'01'	0.230	0.022
'10'	---	0.195
'000'	0.007	0.038
'010'	0.185	0.011
'011'	0.170	0.024
'111'	0.104	0.036
'110'	---	0.108
'100'	---	0.158
'101'	---	0.051
'001'	0.096	0.007

which equals

$$\frac{P(A|S_{RAT})P(S_{RAT})}{P(A)} > \frac{P(A|S_{benign})P(S_{benign})}{P(A)}, \tag{5}$$

which equals

$$\frac{P(A|S_{RAT})}{P(A|S_{benign})} > \frac{P(S_{benign})}{P(S_{RAT})}. \tag{6}$$

According to Formula (6), we look back at Table 5. When considering the unknown flow slices, we ignore '1' and '00' because their frequencies are similar in opposite traffic directions. We also ignore '0' and '11' for symmetry. Subsequently, we can tag '01','011','010','001'and '111', which show exactly higher prior probability in RATs than in benign traffic, as RAT slices. In the same way, we tag '10','100','101','110' and '000' as benign slices. Note that this approach is symmetrical. That means the potential generality without being affected by the imbalance of the training data.

4.2.4.2 Directions and data sizes

The method used in subsection 4.2.4.1 describes external control of a flow slice. By tagging RAT slices with 1 and benign slices with 0, we obtain a 0-1 vector ***slice_control*** of an IP flow. As described in subsection 3.4, the data exfiltration feature is also used to improve the accuracy.

We accumulate the *length* values of the flow slice payloads tagged with ***slice_data***. The out-to-in packet payload sizes are defined as negatives, while the in-to-out packet payload are defined as positive numbers. When the result is greater than 0, the data flow is in-to-out, and we will tag the slice with 1; otherwise, we tag it with 0. Thus, a 0-1 vector of ***slice_data*** is generated that describes the data flow direction corresponding to ***slice_control***. We define the formulas as follows:

$$\boldsymbol{Behavior_{RAT}} = \boldsymbol{slice\_control} .* \boldsymbol{slice\_data}, \tag{7}$$

$$ratio_{RAT} = \frac{sum(\boldsymbol{Behavior_{RAT}})}{len(\boldsymbol{Behavior_{RAT}})}. \tag{8}$$

We define the $ratio_{RAT}$ threshold as 0.785, which equals the total frequency of '01','011','010','001' and '111' in our RAT slices. When the $ratio_{RAT}$ of an unknown flow is beyond the threshold, it is a RAT session.

Table 6. Threshold index

Index	Threshold	How to set
4.2.1 Formula(1)	PR_{ctl}=0.8	A complete TCP flow contains at least 3 control packets in one direction and 4 in the other direction. N_{inictl} denotes the initial control packet number. N_{addctl} denotes the additional control packets during the data transmission. $N_{adddata}$ denotes the actual data packet number. If $N_{adddata} \leq 1$, there will exist $PR_{ctl} = \frac{N_{inictl}+N_{addctl}}{N_{inictl}+N_{addctl}+N_{adddata}} \geq \frac{4+N_{addctl}}{4+1+N_{addctl}} \geq \frac{4}{5} = 0.8$.
4.2.2 Algorithm1	$t_{hr} = 0.8$s	Human reaction time, empirically set.
	$t_{ht} = 3s$	Human think-time, empirically set.
	SCALE=10	Interval dilatation coefficient, empirically set.
4.2.3 Formula(3)	$support = 0.65$	*Support* should be determined by the composition of actual data packets and heartbeats in the flows. A high threshold could cause missed detection, while a low threshold could cause false detection and could significantly increase compute complexity. We set it after several experiments.
4.2.4 Formula(8)	$ratio_{RAT} = 0.785$	We define the $ratio_{RAT}$ threshold as 0.785, which equals the total frequency of '01','011','010','001', and '111' in RAT slices shown in Table 5. This threshold can be adjusted depending on the actual requirements.

We have introduced a detailed procedure of our proposed method. Now we provide guidance on how to set all the thresholds appeared in the process in Table 6.

4.3 Results and Evaluation

We performed three experiments on the three datasets introduced in section 4.1.

4.3.1 Evaluation criterion

When dealing with classification on an imbalanced dataset, we had better use the true positive rate and false positive rate to evaluate the performance. Since all the datasets include only a few RAT sessions, one missed detection could cause an extreme low true positive rate. Thus, we focus on the false positive rate. Also, the number of true alerts among the number of actual RATs will be referred. Besides, given the nature of finding RATs in real-word traces, it seems impossible to estimate the true negative rate. Because our approach achieves almost full detection of known RATs, we tacitly approve that the false negative rate of unknown data equals zero.

We define the false positive rate, which shows the percentage of negative instances misclassified, as

$$FPR = \frac{FP}{FP + TN}, \tag{9}$$

where FP denotes the number of false positive results, while TN denotes the number of true negative results.

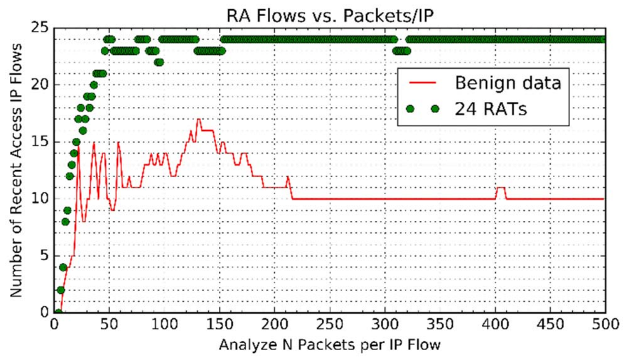


Figure 3. The number of malicious flow alerts in benign data & RATs.

4.3.2 Benign data and RAT data

We use the data captured from lab gateway as benign data, and we executed the 24 RATs to capture 24 RAT IP flows.

In the data collection stage, some IP flows last a long time. However, because our methods involve time slicing, a few slices are sufficient to determine the features of an IP flow that occurs over a long time. The computational burden increases when calculating frequent sequences in a large number of flow slices; thus, we must determine the number of packets from an IP flow to use for analysis that are sufficient to represent the entire flow.

In our work, we scanned packets 0–500 of every flow to analyze the IP flow features. The results are shown in Figure 3.

Figure 3 shows that as the packets grow, the results tend to be stable, especially when the number of packets is above 200. For the RATs, this approach results in almost full detection, while the false alert of benign data involve only 10 flows.

In the benign data, $FP_{benign} = 7$ represents the number of false positive alerts, and because we ignore an unknown FN_{benign} of benign data, we can calculate $FP_{benign} + TN_{benign} \approx ALLflow_{benign} - TP_{benign} = 43321$, where FPR_{benign} represents the false positive ratio. In the benign data, FPR_{benign} equals 0.023%.

To further illustrate the number of packets in an IP flow, we illustrate the packet number distribution in different IP flows in Figure 4.

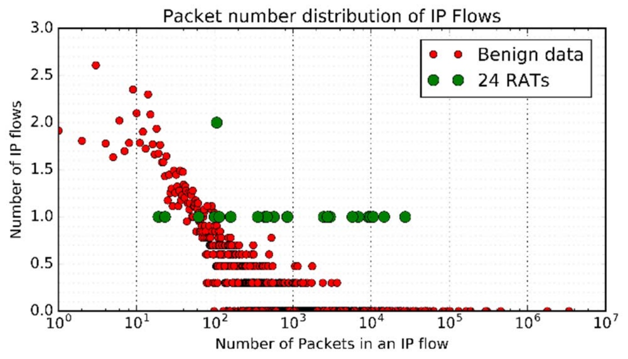


Figure 4. Packet number distribution of IP flows. A log operation is performed on the y-axis values of the smaller red dots, while y-axis of the bigger green dots is labeled such.

Table 7. Traffic data of DARPA 1998 sixth Thursday

Prot.	Attack	Count	Details	Alert
UDP (4)	Land	2	Dos, UDP packet with the same source and destination	---
	Teardrop	2	Dos, Mis-fragmented UDP packets	---
ICMP (14)	Ipsweep	5	port sweep or ping on multiple host	---
	Smurf	2	Dos, icmp echo reply flood.	---
	Satan	2	Network probing tool	---
	Pod	5	Dos, ping of death	---
TCP (18)	Eject	7	Eject buffer overflow, user->root	√
	Perlmagic	2	Perl attack, root shell	√
	Ffb	2	Ffbconfig buffer overflow, root shell	√
	Phf	1	CGI, execute arbitrary command	√
	Dict	1	Guess passwords	×
	Neptune	2	Dos, SYN flood	√
	Portsweep	3	Sweep many ports of a host to determine services	√

The number of packets below 250 in the benign data is 34,772—almost 80.3% of the total data. Thus, we can infer that analyzing 250 packets from an IP flow is sufficient to represent the entire flow over a period of time. As insurance, when analyzing real-word traces, we use first 500 packets of an IP flow.

4.3.3 Performance metrics from DARPA 1998

The sixth week's Thursday traffic data contained 36 attacks, mainly using the UDP, ICMP and TCP protocols, as shown in Table 7. In our experiments, we did not analyze the UDP and ICMP packets, but this aspect may be addressed using our approach in the future.

There are 7 types of TCP attacks with 18 IP flows on which we can perform experiments. The last column of Table 7 shows the detection results, and the last 17 of these trigger our alert. Neptune and Portsweep trigger the flooding-detection alert, while Eject, Perlmagic, Ffb and Phf involve exploits, privilege escalation or command execution.

Dict attacks do not trigger our alert. Although the ***slice_control*** of a Dict attack contains many 1's, its ***slice_data*** contains a large number of 0's because the response packet size of the password guessing attack is smaller than the input request size; therefore, the $\text{ratio}_{\text{RAT}}$ of this attack is below the threshold.

Furthermore, there are 12 IP flow alerts other than the 17 TCP attacks. One http flow and 2 ftp flows show typical external control to the intranet hosts, and the servers are inside the internal network. The other 9 IP flows are all SMTP protocols in which the mail servers and client have few interactions at first; then, following a dormant period, the servers outside push a payload "250…", which is a special feature similar to RATs. According to Equation (9), we calculate that $FPR_{DARPA} = \frac{9}{1781-20-1} = 0.51\%$.

In the traffic data, we also found that most connections are initiated from out-to-in. To adapt to this feature, we did not limit the first SYN packet direction, which must be from in-to-out in this case. However, with regard to post-penetration tools, RATs tend to connect from in-to-out. Strictly speaking, the DARPA data does not contain any RAT data.

It appears that our methods do not fit the demands of the DARPA 1998 attack classification. Although it cannot cover all types of attacks, our approach still shows the potential to reveal many types of attacks in a simple and tough manner.

4.3.4 Other campus data

We also conducted our experiments on two other traces from WITS, as shown in Table 8. Since there are many out-to-in SYN initial packets in WITS data, on this occasion we only consider the IP flows with initial SYN from in-to-out, labeled initiated from inside (IFI) column in the table.

We manually inspected the alerts and labeled the flows in a manner similar to RATs as TP; in other words, they maintain a heartbeat and sometimes respond instantly to a sudden out-to-in packet. Note that the flooding-attacks, described in flood column, are obtained before the IFI judgement. And we calculate the FPR of IFI data according to Equation (9) to evaluate the classification performance.

Notably, flows in different size ranges have different features. We inspect the captured data, whose payloads are truncated to four bytes. The small-size range (<10KB) contains the most flows and we ignore it in calculation because the flows are too short to become RAT sessions. The most RAT alarms come from the middle-size range (10KB-1MB). The big-size range (1MB-10MB) and the huge-size range (>10M) have significantly fewer flows, which reversely lead to a higher false positive rate. Besides, although the false positive is low, the total number of false alarms is high in terms of actual deployment. There are still some false positive alerts arising out of SMTP.

Table 8. WITS data

Flow-size range	WITS 2007						WITS 2011					
	Flows	Flood	Initiated from inside	Alerts	TP	FPR	Flows	Flood	Initiated from inside	Alerts	TP	FPR
<10K	313460	---	---	---	---	---	1122561	---	---	---	---	--
10K-1M	26731	193	11007	19	1	0.164%	75358	1461	29046	28	1	0.093%
1M-10M	549	0	369	4	2	0.542%	2014	10	942	5	0	0.531%
>10M	66	0	0	0	---	---	324	1	141	0	---	---
ALL	340806	193	11376	23	3	0.176%	1200257	1472	30129	33	1	0.106%

Table 9. Suspicious RATs

	IPs	Ports	Time	Packets	Size	Heartbeat	Up-down sequences of flow slices
1	195.231.212.63	51638, 51723, 51844	2007-06-09 13:20:48 2007-06-09 16:30:49	365	65 KB	[-14, 4]	['01', '01', '0111', '00111111', '001', '01111', '001101110100110110110110101100100000101010101 01001010']
	83.63.3.134	34363					
2	195.231.121.238	1037, 1045, … 4973	2007-06-09 08:00:34 2007-06-09 19:59:53	45 K	43 MB	Not obvious	['1', '1', '1', '01', '01', '0111111111111110011111111111111111111101111111111 1111111111111111111111111', '0', '0', '00', '00', '1', '01', '0', '0', '0', '0', '1', '01', '01']
	32.17.118.87	26339					
3	195.231.121.238	1694, 1949, 2706, 3938	2007-06-09 08:12:18 2007-06-09 15:12:25	36 K	34 MB	Not obvious	['1', '01', '0', '01', '0111111111111111111111111', '011111111111110111111111111011111111111101111 1111110111111111111100111111111111101111111011 1111111111101111111111111111111101111111101111 11111111111111111110111111111111101111']
	72.82.139.24	54798					
4	2.65.133.152	33708	2011-04-08 03:00:13 2011-04-08 03:08:34	6801	5 MB	Not obvious	['010101011010101110111111001111101101110110 110110001100000111111100000111110011101101 01 110100010011111100101001100111011101010110 11 1010010000011101100100 1111', '0111111110011100010101110000011 11', '0111011111100100110001011', '011011000011', '01']
	119.120.183.169	5999					

After manually inspecting all the alerts, we suspect that the three RAT sessions in WITS 2007 data and one RAT session in WITS 2011 data are likely to be RATs, with an in-to-out initial SYN, data exfiltration and almost all slices under humanly external control. RAT sessions of this traffic have never been referred in previous research. Because of the Crypto-Pan AES encryption and the truncated payloads, we cannot get more information about the data, except for the ports, time duration (Beijing time), packet number, data size, and heartbeat of the flows. These details are showed in Table 9. For convenience, we still use the IPs as our result shows and we give details about the up-down sequences of the flows. The 195.231.*.* and 2.65.'*'.'*' networks are internal addresses, and their corresponding ports are listed in the ports column. Thus, our proposed methods demonstrate practicability in real-world traces.

All of these experiments show our capability of tracking humanly external control at the area network borders. When dealing with the RATs, we should also discuss why SMTP causes so many false positive alerts. Also, some specifically elaborate RATs may evade our detection. We will discuss them subsequently.

5. FALSE ALARMS AND EVASION

After analyzing the false negative alerts, we find that the SMTP protocol flows lead to the majority of them. Not all the SMTP flows trigger the alerts, but only the servers, which create a ten-second-level delay between responding the client command "mail" with an ACK packet and sending the payload "250…" to inform task completion, can cause the false alarms. These events can be filtered through detecting round number intervals of the delays in the future by our methods.

But what we concern about is, in the case that the servers are designed intentionally to delay dispatching when the scale of network increases nowadays, whether there are other normal protocols sharing the same features with SMTP. SMTP flows behave really like RATs that the servers push first packet immediately after a connection initiation, the upload data is larger than the download, and there may be a delay in the servers.

On the other hand, since our novel methods have not been targeted, it is easy for us to detect malicious external control. The external control is an inevitable feature for RATs, if the malware designers cannot notice it before our research. That means the evasion may happen when the attackers make the RAT controlled-sides intentionally randomly send redundant packets, which are responded automatically with some reverse direction packets. This evasion causes a lot consumption and the attackers need to design the flags to tag what packets need automatic responses, which could be easily detected by other methods. To our methods, the further research on redundant and automatic packets should be implemented.

6. CONCLUSIONS

The RATs have a vital role in cyber-attacks, while the detection methods are still lack. As the mainstream trend of traffic anomaly detection, the machine learning methods with traffic sequence analysis, which have not been used in other RAT detection studies, motivate our work on RAT detection. We proposed a framework to detect RATs at area network borders, using time slicing algorithm to cut the IP flow into flow slices, frequent sequence mining to filter heartbeat and naïve Bayes to classify the slices. Then, we performed tests on a week of our lab continuous traffic data and two types of internet traffic storage. The experiments on the datasets show our proposed methods are excellent in tracking external control and data exfiltration through analyzing only a few packets of the flows. Since the external control stands for a beginning or ongoing attack, our work meets actual demands for security. Our methods have worked well on real-world traces with a false positive rate of less than 0.6%, which is the best known result in existing research.

This approach, in which we use a simple but accurate logic, provides an entirely new perspective for auditing networks. The framework is lightweight, resource saving and effective on known RATs; however, it also exhibits shortfalls of recognizing automatic behaviors, which could cause false alarms from a server delay dispatching or missed detection of an automatic RAT.

The next stage of our research will involve constructing a model of automatic behaviors that can address both SMTP and other similar protocols with a delay dispatching. Then as previously stated, we will study on other protocols besides TCP to research whether this method works on them or how to work on them.

7. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No.61271252).

8. REFERENCES

[1] Hutchins, E. M., Cloppert, M. J. and Amin, R. M. 2011. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research* 1 (2011), 80.

[2] Al-Saadoon, G., Al-Bayatti, H. M. and others. 2011. A Comparison of Trojan Virus Behavior in Linux and Windows Operating Systems. *arXiv preprint arXiv:1105.1234* (2011).

[3] Wuest, C. 2004. Advanced communication techniques of remote access trojan horses on Windows operating system GSEC Practical v1. 4b (option 1). (2004).

[4] Roesch, M. and others. 1999. Snort: Lightweight Intrusion Detection for Networks. In *LISA*. 229–238.

[5] Feily, M., Shahrestani, A. and Ramadass, S. 2009. A Survey of Botnet and Botnet Detection. In IEEE, 268–273. DOI= https://doi.org/10.1109/SECURWARE.2009.48.

[6] Singh, K., Guntuku, S. C., Thakur, A. and Hota, C. 2014. Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests. *Information Sciences* 278 (September 2014), 488–497. DOI= https://doi.org/10.1016/j.ins.2014.03.066.

[7] Sommer, R. and Paxson, V. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*. IEEE, 305–316.

[8] Villeneuve, N. and Bennett, J. 2012. Detecting apt activity with network traffic analysis. *Trend Micro Incorporated Research Paper* (2012).

[9] Moore, A., Zuev, D. and Crogan, M. 2005. *Discriminators for use in flow-based classification*. Queen Mary and Westfield College, Department of Computer Science.

[10] Lim, Y., Kim, H., Jeong, J., Kim, C., Kwon, T. T. and Choi, Y. 2010. Internet traffic classification demystified: on the sources of the discriminative power. In *Proceedings of the 6th International Conference*. ACM, 9.

[11] Lim, H., Yamaguchi, Y., Shimada, H. and Takakura, H. 2015. Malware classification method based on sequence of traffic flow. In *2015 International Conference on Information Systems Security and Privacy (ICISSP)*. IEEE, 1–8.

[12] Lakhina, A., Crovella, M. and Diot, C. 2005. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*. ACM, 217–228.

[13] Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A. and Salamatian, K. 2006. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review* 36, 2 (2006), 23–26.

[14] Wang, Z. 2015. *The Applications of Deep Learning on Traffic Identification*. BlackHat USA (2015).

[15] Louvieris, P., Clewley, N. and Liu, X. 2013. Effects-based feature identification for network intrusion detection. *Neurocomputing* 121 (2013), 265–273.

[16] Jiang, D. and Omote, K. 2015. An Approach to Detect Remote Access Trojan in the Early Stage of Communication. In IEEE, 706–713. DOI= https://doi.org/10.1109/AINA.2015.257.

[17] Adachi, D. and Omote, K. 2016. A Host-Based Detection Method of Remote Access Trojan in the Early Stage. In *International Conference on Information Security Practice and Experience*. Springer, 110–121.

[18] Li, S., Yun, X., Zhang, Y., Xiao, J. and Wang, Y. 2012. A General Framework of Trojan Communication Detection Based on Network Traces. In IEEE, 49–58. DOI= https://doi.org/10.1109/NAS.2012.10.

[19] Pu, Y., Chen, X., Cui, X., Shi, J., Guo, L. and Qi, C. 2013. Data Stolen Trojan Detection based on Network Behaviors. *Procedia Computer Science* 17 (2013), 828–835. DOI= https://doi.org/10.1016/j.procs.2013.05.106.

[20] Yamada, M., Morinaga, M., Unno, Y., Torii, S. and Takenaka, M. 2015. RAT-based malicious activities detection on enterprise internal networks. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 321–325.

[21] Beauchesne, N. and Prenger, R. J. 2015. *Method and system for detecting external control of compromised hosts*. Google Patents.

[22] From Poisson Processes to Self-Similarity: a Survey of Network Traffic Models.

[23] Paxson, V. and Floyd, S. 1995. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking (ToN)* 3, 3 (1995), 226–244.

[24] Uhlig, S. 2004. Non-stationarity and high-order scaling in TCP flow arrivals: a methodological analysis. *ACM SIGCOMM Computer Communication Review* 34, 2 (2004), 9–24.

[25] Barbosa, R. R. R., Sadre, R., Pras, A. and Meent, R. 2010. Simpleweb/university of twente traffic traces data repository. (2010).