TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 18/18 pp432–442
DOI: 10.26599/TST.2021.9010041
Volume 27, Number 2, April 2022

Effective Algorithms to Detect Stepping-Stone Intrusion by Removing Outliers of Packet RTTs

Lixin Wang*, Jianhua Yang, Michael Workman, and Pengjun Wan

Abstract: An effective method to detect stepping-stone intrusion (SSI) is to estimate the length of a connection chain. This type of detection method is referred to as a network-based detection approach. Existing network-based SSI detection methods are either ineffective in the context of the Internet because of the presence of outliers in the packet round-trip times (RTTs) or inefficient, as many packets must be captured and processed. Because of the high fluctuation caused by the intermediate routers on the Internet, it is unavoidable that the RTTs of the captured packets contain outlier values. In this paper, we first propose an efficient algorithm to eliminate most of the possible RTT outliers of the packets captured in the Internet environment. We then develop an efficient SSI detection algorithm by mining network traffic using an improved version of k-Means clustering. Our proposed detection algorithm for SSI is accurate, effective, and efficient in the context of the Internet. Well-designed network experiments are conducted in the Internet environment to verify the effectiveness, correctness, and efficiency of our proposed algorithms. Our experiments show that the effective rate of our proposed SSI detection algorithm is higher than 85.7% in the context of the Internet.

Key words: network security; intrusion detection; stepping-stone intrusion; round-trip time; k-Means clustering; connection chain

1 Introduction

Presently, most intruders tend to launch attacks on a remote target system through compromised hosts to avoid being detected by security professionals[1–6]. These compromised hosts are called stepping-stones. With a Stepping-Stone Intrusion (SSI), an attacker uses a chain of stepping-stone hosts in the context of the Internet as relaying machines and remotely logs into these stepping-stones using command-line tools, such as SSH. The intruder sends attacking packets to a remote target system by relaying via the intermediate stepping-stones in the connection chain until the attacking commands arrive at the potential victim target.

It is well-known that TCP sessions are independent of one another in a TCP connection between a source and a destination, even if these sessions are relayed and belong to the same TCP connection. This nature of the TCP protocol makes detecting the attacker's original geographical location very difficult if SSI is used for an attack, as the intruder is hidden behind a long interactive connection chain due to the existence of the intermediate stepping-stones between the attacker's machine and the potential victim host. With SSI, the possibility for a potential victim host to obtain information about the origin of the attack is very small.

With SSI, attackers usually create a connection chain similar to that of Fig. 1 in which the intruder uses command-line tools, such as SSH, to log in to the stepping-stones and then launch an attack. In Fig. 1, Host 0 is assumed to be the attacker's host that is used

• Lixin Wang, Jianhua Yang, and Michael Workman are with TSYS School of Computer Science, Columbus State University, Columbus, GA 31907, USA. E-mails: {wang_lixin, yang_jianhua, workman_michael} @columbusstate.edu.

• Pengjun Wan is with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA. E-mail: wan@cs.iit.edu.

∗ To whom correspondence should be addressed.

Manuscript received: 2021-04-22; revised: 2021-05-08; accepted: 2021-05-16

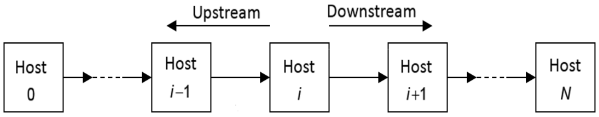


Fig. 1 A sample connection chain.

to log in to stepping-stone hosts Host 1, Host 2, ..., Host $i-1$, Host i, Host $i+1$, ..., and Host $N-1$. Host N represents the remote target system that is under attack. To perform SSI detection, any stepping-stone host (other than the attacker's host and the target system) can serve as the sensor machine where a detection program, such as TCPDump, is installed and used to capture the packets. In Fig. 1, Host i is assumed to be the detection sensor machine with a detection program running. The connection from Host $i-1$ to Host i is referred to as an incoming connection to Host i, and the connection from Host i to Host $i+1$ is referred to as an outgoing connection from Host i. The part of the connection chain from the attacker's Host 0 to the sensor Host i is called the downstream chain, and the part of the connection chain from the sensor Host i to the target system Host N is called the upstream chain. If at least one relayed pair is between all the incoming connections and all the outgoing connections, then it is most likely that Host i is being used as a stepping-stone, and the session is being manipulated by hackers. The purpose of SSI detection is to determine whether the detecting sensor Host i is being used as a stepping-stone host by a hacker.

With host-based SSI detection, the outgoing connections are compared with the incoming connections of the same machine, and then the packets are analyzed to check whether a relayed pair in the outgoing and incoming connections can be found. This type of detection method for SSI includes the known approaches presented in Refs. [1–8]. The main disadvantage of this type of detection approach is that it only focuses on a single host, and thus can easily introduce high false-positive errors because it is well-known that stepping-stones are actually used to access remote servers by some legal applications. Client browsers and Web servers are typical examples of this type of application to access remote servers. Another disadvantage of host-based SSI detection is that it is ineffective when the sessions are manipulated by intruders using certain hacking techniques, such as meaningless chaff-packet permutation and/or time-jittering of changing the timestamps of attacking packets.

To overcome the disadvantages of host-based SSI detection, network-based detection approaches were proposed for SSI[9–20]. The key idea of this type of detection method is to estimate the number of connections from the intruder's host to the target host (as shown in Fig. 1), which is referred to as the length of a (whole) connection chain. If a connection chain contains three or more connections, then the user attempts to gain access to a remote target host via three or most stepping-stones. The more hosts involved in an interactive session to access a remote server, the slower the data communication process. Legal applications are well-known for never using three or more intermediate hosts to access a remote server. Therefore, if a target host is accessed via three or more stepping-stones from another host, then the possibility that the session is manipulated by a malicious hacker and the target system is under attacks is very high.

Clearly, false-negative errors could be generated by using the network-based detection methods. Estimating the length of the upstream connection is referred to as upstream detection. Similarly, estimating the length of the downstream connection chain is referred to as downstream detection. Upstream detection is performed with more difficulty and complexity due to the loose coupling of Send and Echo packets sent from the intruder's host. The Send packets captured at the sensor machine from the upstream connection do not have any correlation with the Echo packets sent from the attacker host. Therefore, the length of an upstream connection is much more difficult to be estimated precisely, which has been a long-standing open problem for SSI detection. If no other hosts are between the sensor machine and the target host, the length of the entire chain is almost identical to that of the upstream connection. In this case, all malicious intrusions cannot be detected because of the false-negative errors caused by the network-based detection approaches. As the length of an upstream connection is at least one, as long as the length of the downstream connection is at least two, the length of the entire connection chain will be at least three. Therefore, if the downstream connection contains two connections, then the target host is probably under attacks, and the session is being manipulated by intruders. Most network-based detection algorithms in the literature only considered the downstream connection. For simplicity, the length of a connection chain means that the downstream and the upstream connection parts are always ignored in this paper, as done in most existing network-based detection algorithms.

However, existing network-based detection methods for SSI are either ineffective in the Internet environment because of the existence of outlier values in the RTTs obtained from the captured TCP packets, or they are inefficient, as many packets must be captured and processed for these detection approaches. Because of the high fluctuation caused by the intermediate routers in the Internet environment, RTTs of the captured packets with outlier values are unavoidable. In this paper, we first propose an efficient algorithm to eliminate most of the possible outliers of the RTTs of the packets captured in the Internet environment. We then develop an efficient detection algorithm for SSI by mining network traffic using an improved version of *k*-Means clustering. Our proposed detection algorithm for SSI is accurate, effective, and efficient in the context of the Internet. Many well-designed network experiments in the Internet environment are conducted to verify the effectiveness, correctness, and efficiency of our proposed detection algorithm. Our network experiments show that the effective rate of our proposed detection algorithm for SSI is higher than 85.7% in the Internet environment.

Next, we give a literature review on SSI detection that focuses on various network-based detection approaches, which are more relevant to the work we will present in this paper. The first proposed detection algorithm via a network-based approach was presented in Ref. [16] by Yung in 2002. The key idea used in Ref. [16] was to calculate the RTT of a Send packet and try to match this Send with its corresponding ACKnowledgement (ACK) packet transmitted from the next adjacent host in the connection chain. The algorithm in Ref. [16] slightly reduced the false-positive errors. However, this algorithm for SSI detection generates high false-negative errors due to the use of the ACK packet from the next adjacent host, and the actual Echo packet is not used and analyzed. The problem of the work in Ref. [16] is that the connection chain was not appropriately set up. As a result, capturing the actual Echo packets from the connection chain created in Ref. [16] is impossible.

To overcome the problem caused in the work in Ref. [16], a step-function detection method was proposed to estimate the length of a connection chain in Ref. [17] by Yang and Huang in 2004. Unlike the detection method developed in Ref. [16], the step-function approach developed in Ref. [17] reduced the false-positive and false-negative errors in the context of a local area network (LAN). The connection chain was properly established in Ref. [17] so that the corresponding Echo packet of a Send packet can be captured and analyzed. In Ref. [17], the step-function approach is used to calculate the packet RTTs by matching a Send packet with its corresponding Echo packet. The drawback of the detection method proposed in Ref. [17] is that it only works well in LANs and is ineffective in the context of the Internet. The conservative and greedy packet matching method for SSI detection proposed by the authors in Ref. [18] worked well in the Internet environment, but very few TCP packets can be matched using this detection approach. As a result, the SSI detection method proposed in Ref. [18] is ineffective and cannot be used to protect practical computer networks that are exposed to the Internet.

To the best of our knowledge, the clustering and partitioning data mining approach presented in Ref. [19] by Yang and Huang is one of the best-existing connection-chain-based SSI detection approaches. In Ref. [19], the packet RTTs are calculated using the maximum-minimum distance (MMD) clustering algorithm, and all the possible packets go through during the process of packet matching, unlike the previously known methods of matching the Send and Echo packets, through which Echo and Send packets are compared one at a time. On the basis of this approach, the number of clusters produced by the MMD algorithm determines the number of connections in the chain. Additionally, the MMD approach proposed in Ref. [19] reduced the false-positive and false-negative errors significantly. However, this detection approach for SSI requires that many TCP packets must be captured and processed. As a result, the MMD approach proposed in Ref. [19] is inefficient in terms of packet capturing and the processing time of the detection algorithm.

To overcome the drawback of the MMD data mining approach proposed in Ref. [19], a detection algorithm for SSI using the *k*-Means clustering approach was proposed in one of our prior works[20]. This detection algorithm based on *k*-Means clustering does not require many packets to be captured and processed, and thus it is efficient. Packet RTTs cluster near several levels, either in LANs or in the Internet environment[17, 19]. Usually, the *k*-Means data mining approach is used to put elements of a dataset into groups of related observations without prior knowledge of those relationships. As long as most of the RTT outliers can be removed from the real RTTs in the input file, the detection algorithm based on *k*-Means clustering proposed in Ref. [20] works fine in the LAN environment.

Some recent significant research work related to network security merit acknowledgment. In Ref. [21], a voting-based deep learning framework was proposed to detect abnormal network behavior by exploiting any type of deep learning structure. When several models created by different aspects of data and various deep learning structures are considered, this framework provides the ability to aggregate the best models to produce more accurate and robust results. Reference [22] proposed a new approach for intrusion detection by applying a big databased deep learning system. The authors of Ref. [22] proposed a big databased hierarchical deep learning system that uses behavioral features and content features to understand network traffic characteristics and information stored in the data payload to enhance the performance of machine learningbased intrusion detection systems. Intrusion detection systems suffer from numerous vulnerabilities during the analysis and classification of data activities. To overcome this problem, Ref. [23] developed a new analysis method and implemented a relevant system to monitor the circulated traffic. Reference [23] modeled and validated a heterogeneous traffic classifier that can categorize collected events within networks. This model is based on a proposed machine learning algorithm that comprises an input layer, a hidden layer, and an output layer. Reference [24] investigated how to launch an inference attack exploiting social networks with a mixture of non-sensitive attributes and social relationships. In Ref. [24], this problem is mapped into a collective classification problem. In the collective inference model proposed in Ref. [24], an attacker uses user profiles and social relationships collectively to predict sensitive information of related victims in a released social network dataset. Reference [25] proposed an approach for detecting fake news over online social media via domain reputations and content understanding. The authors of Ref. [25] characterized hundreds of popular fake and real news measured by shares, reactions, and comments on Facebook from two perspectives: domain reputations and content understanding. Reference [26] developed an approach to uploading data in smart cyber-physical systems, in which energy conservation and privacy preservation are considered. Reference [27] proposed a privacy-preserved data sharing structure for the industrial Internet of Things, in which several competing clients could exist in distinct stages of the system. Reference [28] developed a mechanism for trading range counting results. Under differential privacy, this mechanism used a sampling method to produce rough counting results that are theoretically verified to achieve unbiasedness, bounded variance, and a privacy guarantee. Reference [29] proposed mechanisms that enable a comprehensive detection and reconstruction of attacks. The contributions of Ref. [29] are toward a better overall detection accuracy at two stages of the intrusion detection process. First, security monitoring is enhanced to produce high-quality monitoring data and to leverage it for an accurate reporting of alerts. Second, new alert correlation mechanisms identify relations among the alerts and summarize the reconstructed attacks. Reference [30] developed a comprehensive framework to simulate realistic stepping-stone behavior that includes effective evasion tools and released a large dataset, which we use to evaluate detection rates for eight state-of-the-art methods in the literature.

All notations used in this paper are listed in Table 1 for easy referencing.

The remainder of this paper is organized as follows. In Section 2, preliminary knowledge needed to propose the detection algorithm is presented. An efficient algorithm is developed in Section 3 to remove the outliers in the packets RTTs captured in the Internet environment. In Section 4, we propose an effective algorithm for SSI detection by mining network traffic using an improved version of the k-Means clustering method. Performance analysis of the proposed detection algorithm is provided in Section 5. Finally, the paper is summarized, and future research directions are given in Section 6.

2 Preliminary

In this section, we introduce some basic concepts in computer networks that are required to design our detection algorithm for SSI and the rationale for using packet RTTs to estimate the length of a connection chain.

2.1 Definitions of send/echo packets

Let us use Fig. 1 to define Send and Echo packets. Host

Table 1 All notations used in this paper.

Notation	Meaning
X	a random variable
μ	mean of a random variable
σ	standard derivation
k	the number of clusters
c_i	the center of the i-th clusters, where $i = 1, 2, \dots, k$
C	the set of k centers $\{c_1, c_2, \dots, c_k\}$
Dataset-j	a dataset of RTT values, where $j = 1, 2,$ or 3

i is the detecting sensor. In the incoming connection of Host i, a Send packet is defined as a TCP packet received at Host i and sent from Host $i-1$, with the TCP.Flag.PSH bit set; an Echo packet is defined as a TCP packet received at Host $i-1$ and sent from Host i, with the TCP.Flag.PSH bit set. In the outgoing connection from Host i, a Send packet is defined as a TCP packet received at Host $i+1$ and sent from Host i, with the TCP.Flag.PSH bit set; an Echo packet is defined as a TCP packet received at Host i and sent from Host $i+1$, with the TCP.Flag.PSH bit set.

Now, let us use an example to explain which Send packet and Echo packet are *a matched pair*. When a user types a command on a command line in a Linux system, such as "ls", it might be sent to the server in one or two packets. Suppose that the command "ls" is sent to the remote server in two separate Send packets: "l" and "s". When "l" is typed on the user's command line, the packet will be sent to the server-side. Once this Send packet is echoed, an Echo packet is sent back to the user's host, and the letter "l" will be shown on the terminal of the user's host. These Send and Echo packets are called a matched pair. For the letter "s", a matched pair can be similarly obtained: a Send "s" and an Echo "s". Using the timestamps of a matched pair of Send and Echo packets, their packet RTT can be easily computed. The length of the connection time of an interactive TCP session is represented by the RTT of a matched pair. The RTT computed from the matched pair of the Send and Echo packets of "l" is different from the RTT computed from the matched pair of the Send and Echo packets of "s"; these two numbers are very close to each other because these two RTTs stand for the length of the same connection in different periods. A Send packet could be echoed by one or more Echo packets. Additionally, an Echo packet could echo one or more Send packets.

2.2 Distribution of packet RTTs for a connection chain

A packet RTT of a TCP connection is the sum of four delays, including the processing delay, queuing delay, transmission delay, and propagation delay of the underlying connection. For connection-chain-based SSI detection, the length of a connection chain is estimated using the packet RTTs. The RTTs obtained from matched Send and Echo pairs can be used to represent the network traffic. Yang and Huang[19] showed that the length of a connection chain is equal to the number of clusters that are produced using the RTTs obtained from the connection chain.

Paxson and Floyd[31] found that packet RTTs obtained from a connection chain obey the Poisson distribution. This finding can be used to match TCP packets and further estimate the number of connections in an extended connection chain. Figure 2 shows a typical experiment in which packet RTTs obey the Poisson distribution, where the Y axis represents the probability of the occurrence of each RTT, and the X-axis represents the values of the RTTs in microseconds. The values of the RTTs shown in Fig. 2 were from the packets collected from a connection chain composed of four connections. With this experiment, most values of the RTTs are near the mean $\mu = 138\,500$ (microsecond) of all the RTTs, with at least 95% of the RTT values between 137 000 (microsecond) and 141 000 (microsecond) (refer to Fig. 2).

Assume that a random variable $\mathcal{X}$ follows the Poisson distribution, and its mean and standard deviation are represented by μ and σ, respectively. Then, we have

$$|\mathcal{X} - \mu| \leqslant 2\sigma \tag{1}$$

On the basis of this inequality, most values of the random variable $\mathcal{X}$ must be near its mean value μ. The difference between $\mathcal{X}$ and its mean μ is upper bounded by 2σ. According to our above discussion, the packet RTTs obtained from a connection chain follow the Poisson distribution. In other words, most values of the packet RTTs obtained from a connection chain of fixed length must be near its mean within a circle of radius 2σ. Therefore, the RTTs corresponding to a connection chain of fixed length belong to the same cluster, with the mean μ being the cluster center. Thus, if a detection algorithm can be developed to obtain the number of these data clusters, then we can easily obtain the matched packets

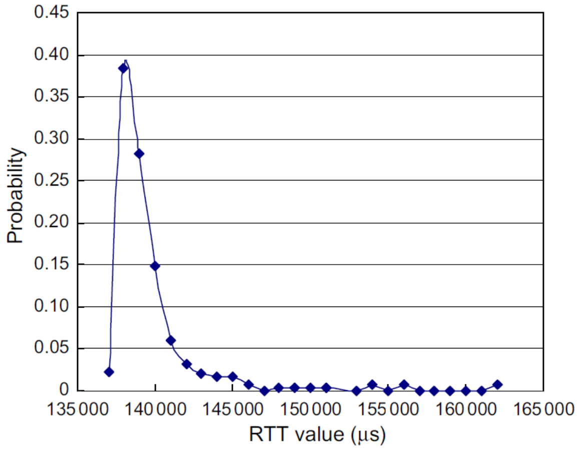


Fig. 2 Distribution of packet RTTs for a connection chain.

and estimate the length of the connection chain.

Many legitimate applications, such as Web applications, are well-known for usually using other servers as stepping-stones to access a remote server. However, most such legitimate applications never use three or more hosts as stepping-stones to access a remote server. Obviously, the more hosts used to access a remote server, the slower the network traffic. If there is no intention of hiding malicious activities, then accessing a remote server via three or more stepping-stones is unnecessary, as it could make the performance of the remote accessing unacceptable. Therefore, a reasonable assumption is that a host who uses three or more stepping-stones to access a remote server is highly suspect as a malicious intruder. This assumption is popularly used in almost all network-based approaches for SSI detection in the literature.

3 An Efficient Algorithm for Removing RTT Outliers

In this section, we present an efficient algorithm for removing the outliers in the packet RTTs captured in the Internet environment. Because of the high fluctuation caused by the intermediate routers on the Internet, outlier values are unavoidable in the RTTs of the captured packets. Most of the existing network-based detection methods for SSI are ineffective because of the presence of outliers in the packet RTTs captured in the context of the Internet. The input of this algorithm is a TXT file with two columns including packet timestamps and the packet type (either Send or Echo) obtained from the packets captured in the Internet environment. The output of this algorithm is a TXT file that contains the packet RTTs with most of the RTT outlier values removed. The key idea of this algorithm is that before an RTT value is added to the file of RTT values, we need to check whether it is a possible RTT outlier by determining whether it is tenfold larger than the current average of all the RTT values that have been added to the file. If it is, then it is likely to be an outlier RTT value. The algorithm for removing RTT outliers is described in Algorithm 1.

Algorithm 1 An efficient algorithm for removing the outliers of packet RTTs

```
Input: a TXT file with two columns (including packet
    timestamps and the packet type) obtained from the packets
    captured in the Internet environment
Output: a TXT file output.txt containing the packet RTTs with
    most of the RTT outliers removed
1: for each of the first five Echo packets, compute the time
   difference between the Echo packet and its immediate prior
   Send packet; write this time difference to the file output.txt
   for each of these five Echo packets
2: compute the average of all the RTTs in the file output.txt and
   store the value in the variable average
3: for the 6th Echo packet, compute the time differences between
   this Echo packet and its prior five Send packets; assume these
   Send packets are represented by Send1, Send2, Send3, Send4,
   and Send5, respectively
   (a) let timediff1 denote the difference between this Echo
   packet and Send1. Write timediff1 to the file output.txt if
   timediff1 is less than $10 \times average$; recompute the average
   of all the RTTs in output.txt and then update the value of
   average
   /* if the time difference is larger than or equal to tenfold the
   average, then the RTT value is most likely an outlier and will
   not be written into the output file. Here, the tenfold criterion
   is based on observation*/
   (b) let timediff2 denote the difference between this Echo
   packet and Send2. Perform the same process for timediff2 as
   done in the above Step 3(a)
   (c) let timediff3 denote the difference between this Echo
   packet and Send3. Perform the same process for timediff3 as
   done in the above Step 3(a)
   (d) let timediff4 denote the difference between this Echo
   packet and Send4. Perform the same process for timediff4 as
   done in the above Step 3(a)
   (e) let timediff5 denote the difference between this Echo
   packet and Send5. Perform the same process for timediff5 as
   done in the above Step 3(a)
4: repeat Step 3 for each of the remaining Echo packets in the
   input file until all the Echo packets are processed
```

4 An Effective Detection Algorithm for SSI

In this section, we present an effective algorithm to detect SSI by mining network traffic using an improved version of the k-Means clustering approach. Our network experiment shows that when $k = 2$ (i.e., all the RTTs are partitioned into exactly two clusters), our proposed detection algorithm for SSI works effectively, with an effective rate higher than 85.7% in the Internet environment. In this paper, the k-Means clustering approach is only applied when $k = 2$. The k-Means clustering approach is neither needed nor used in our proposed algorithm for other values of k, as it does not work effectively when $k > 2$ in the context of the Internet. The input of our proposed detection algorithm for SSI in this section is three datasets that were captured from distinct connection chains with two, three, or four connections. Our proposed detection algorithm can effectively output whether a possible intrusion is underway in the underlying network.

For the general k-Means clustering method, the main idea is to find the appropriate k centers, one for each of the k clusters. The k centers need to be calculated and then updated in a smart way to minimize the final standard derivation calculated based on the updated k centers in the last iteration of the algorithm. Refer to Ref. [32] regarding the k-Means algorithm and its implementation. A k-Means clustering algorithm is an appropriate option for clustering because of the unique characteristics of TCP packets. Legal applications are well-known for never using three or more intermediate hosts to access a remote server. Therefore, if a target host is accessed via three or more stepping-stones from another host, then it is highly possible that the session is being manipulated by a malicious hacker, and the target system is a potential victim host.

Now, we explain how to use the 2-Means clustering approach to detect SSI effectively. Consider three RTT datasets obtained from connection chains of length two, three, or four in the context of the Internet. We run the 2-Means clustering algorithm on these three RTT datasets. Intuitively, the output generated by the dataset collected from the connection chain of length two would produce the smallest standard derivation. On the basis of this intuition, for these three datasets, we may conclude that the dataset that produced the output with the smallest standard derivation was obtained from the connection chain of length two.

Let $P = \{p_1, p_2, \dots, p_n\}$ be a set of n data points. Assume that $C = \{c_1, c_2, \dots, c_k\}$ is the set of k centers that were found by the k-Means clustering algorithm. We partition P into k disjoint subsets $p_1, p_2, \dots, p_k$, where the data points in p_j are associated with their nearest center c_j in C for each $1 \leqslant j \leqslant k$ by associating each point in P with the nearest center in C. For each $1 \leqslant j \leqslant k$, let $p_j = \{p_{j_1}, p_{j_2}, \dots, p_{j_{|p_j|}}\}$. The standard derivation σ of the given dataset P based on k-Means clustering is defined as the square root of its variance as follows:

$$\sigma = \sqrt{\frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{|Y_j|}\|y_{j_i} - c_j\|^2} \tag{2}$$

Let dataset-1, dataset-2, and dataset-3 denote the three RTT datasets obtained from three connection chains of distinct lengths of two, three, or four, respectively. Our proposed algorithm for SSI detection using the 2-Means clustering method can accurately determine which connection chain(s) are sessions possibly manipulated by malicious hackers. This detection algorithm for SSI is described in Algorithm 2.

Next, we explain Algorithm 2, which is used to detect possible SSI. In Step 1, the 2-Means clustering algorithm is called on dataset-1. The procedure is as follows.

Initially, we randomly select two points in the set P as the two centers for the clustering algorithm that executes the steps below:

(1) To generate the two clusters, each point in P is scanned and assigned to the cluster center whose distance from the cluster center is the minimum between both cluster centers;

(2) Calculate the standard derivation σ_{curr} using the current partition and cluster centers according to Eq. (2);

(3) For each cluster, recompute the new center, which is the average of the data points in the cluster;

(4) Use the new cluster centers to reproduce the two clusters following the same procedure described in the above Step (1);

(5) Recompute the standard derivation σ_{new} using the updated partition and cluster centers according to Eq. (2);

(6) If $\sigma_{\text{new}} \geqslant \sigma_{\text{curr}}$, then Exit; otherwise, repeat Step (3).

When the 2-Means clustering algorithm exits, the standard derivation is minimized using the cluster centers and partition obtained in the last round of the clustering algorithm. Thus, at the end of the algorithm execution, the standard derivation can no longer be reduced by changing the centers of each cluster. According to

Algorithm 2 An effective algorithm for SSI detection

```
Input: dataset-1, dataset-2, and dataset-3
Output: The connection chain(s) that are sessions possibly
    manipulated by malicious hackers
call the 2-Means clustering algorithm on dataset-1. Assume$\sigma_1$
    represents the standard derivation outputted based on Eq. (2)
    using the two clusters obtained at the end of the 2-Means
    clustering algorithm execution
call the 2-Means clustering algorithm on dataset-2. Assume$\sigma_2$
    represents the standard derivation outputted based on Eq. (2)
    using the two clusters obtained at the end of the 2-Means
    clustering algorithm execution
call the 2-Means clustering algorithm on dataset-3. Assume
    $\sigma_3$ represents the standard derivation outputted based on
    Eq. (2) using the two clusters obtained at the end of the
    2-Means clustering algorithm execution
if $\sigma_1 = \min\{\sigma_1, \sigma_2, \sigma_3\}$, dataset-2 and dataset-3 are sessions
    possibly manipulated by malicious hackers
if $\sigma_2 = \min\{\sigma_1, \sigma_2, \sigma_3\}$, dataset-1 and dataset-3 are sessions
    possibly manipulated by malicious hackers
if $\sigma_3 = \min\{\sigma_1, \sigma_2, \sigma_3\}$, dataset-1 and dataset-2 are sessions
    possibly manipulated by malicious hackers
```

Eq. (2), the value of σ_1 is the standard derivation obtained using the two clusters obtained in the last iteration of the 2-Means clustering algorithm.

Similarly, the 2-Means clustering algorithm is called on dataset-2 and dataset-3, respectively, at Steps 2 and 3 of the algorithm. Then, the standard derivations σ_2 and σ_3 are in turn calculated.

If σ_1 is the minimum among the three values σ_1, σ_2, and σ_3, then dataset-2 and dataset-3 are sessions possibly manipulated by malicious hackers. Similarly, if σ_2 is the minimum, then dataset-1 and dataset-3 are sessions possibly manipulated by malicious hackers. If σ_3 is the minimum, then dataset-1 and dataset-2 are sessions possibly manipulated by malicious hackers. Therefore, this algorithm can effectively determine which connection chain(s) are sessions possibly manipulated by hackers.

5 Network Experiments and Performance Analysis of the Proposed Algorithm

In this section, we conduct network experiments in the Internet environment and analyze the performance of our proposed algorithms for SSI detection. The window size we used to generate the RTT datasets in the experiments is three for all the captured packets. That is, for each Echo packet, we compute the timestamp differences between this Echo packet and up to three previous Send packets. We will verify the effectiveness and correctness of our proposed detection algorithm for SSI through well-designed network experiments. In total, we collected seven datasets, each of which contains three captured files obtained from connection chains of lengths two, three, or four. Assume that we do not know the length of the connection chain where each of these datasets was collected.

Our network experiments are conducted in the Internet environment using the virtual machine servers we created on our Amazon AWS account. To set up the network experiments, we identified four AWS virtual servers located in different parts of the world. Our local MacBook Pro running Ubuntu with IP address: 192.168.1.29 served as the intruder's host (Fig. 3). Then, we used SSH to gain remote access to our Mac Mini Ubuntu desktop as the sensor machine (the first stepping-stone $S1$ in Fig. 3) with IP address 192.168.1.40. The first AWS Ubuntu server, located in Virginia, USA, with IP address 34.239.106.90, served as the second stepping-stone $S2$. The second AWS Ubuntu server, located in Ireland with IP address 34.243.200.211, served as the third stepping-stone $S3$. The next AWS Ubuntu server, located in Germany with IP address 18.194.239.70, served as the fourth stepping-stone $S4$. Finally, the fourth AWS Ubuntu server, located in Mumbai, India, with IP address 13.233.125.64, served as the victim's host (Fig. 3).

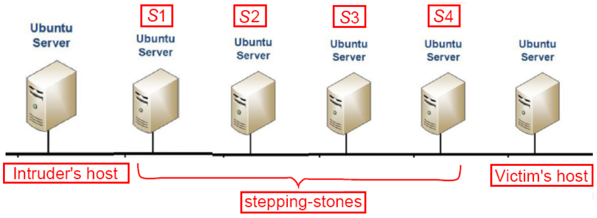


Fig. 3 Connection chain of length 4 (from the sensor host $S1$ to the victim's host).

We set up the connection chain starting from our MacBook Pro running Ubuntu as the intruder's host and our Mac Mini Ubuntu desktop as the sensor host $S1$ from which all the packets are captured using TCPDump. We used SSH to gain remote access to the second stepping-stone $S2$. Once we finished with $S2$, we then used SSH to gain remote access to the third stepping-stone $S3$. This gave us a total of three hosts and two connections in the downstream chain starting from the sensor host $S1$. Once we finished with $S3$, we used SSH to log into the fourth stepping-stone $S4$. This gave us a total of four hosts and three connections in the downstream chain. Finally, we used SSH to log in to the victim's host located in Mumbai, India. This gave us a total of five hosts and completed the connection chain with four connections in the downstream chain from the sensor host $S1$ to the victim host. After the connection chains of lengths two, three, and four were created, we then used the TCPDump command to capture packets on the sensor host $S1$ from each of these connection chains.

See Fig. 3 for the connection chain of four connections. For the connection chain of two connections, we monitored and captured its TCP Send and Echo packets, and then computed the RTTs for the connection chain. The RTT datasets were generated using a window of size three. We found that most of its RTTs are actually near the mean of these RTTs of the connection chain. Theoretically, this result is true according to the distribution of the packet RTTs of a connection chain we discussed in Section 2. Then, we perform the same procedures for the connection chains of three and four connections, respectively. For the connection chain of length three, most of its RTTs are indeed near the mean of the RTTs of the connection chain. The same is true for the connection chain with length four. Let dataset-1, dataset-2, and dataset-3 represent the datasets of packet RTTs generated from the captured TCP Send and echo packets collected from the connection chains of lengths 2, 3, and 4, respectively, using a window of size three.

Then, we run the 2-Means clustering algorithm (i.e., all the elements in the dataset are divided into exactly two clusters) on dataset-1, dataset-2, and dataset-3. For each input dataset, this 2-Means clustering algorithm outputs the standard derivation calculated using the two clusters obtained at the last iteration of the 2-Means clustering based on Eq. (2). The standard derivations outputted by this clustering algorithm are listed in Table 2. Based on the data shown in Table 2, the RTT dataset we collected from the connection chain of length two achieves the smallest standard derivation on six of the seven datasets we captured (only the output generated on dataset-2 is incorrect). This result means that our proposed detection algorithm for SSI generates correct output on six of the seven datasets. Therefore, this network experiment shows that the effective rate of our proposed detection algorithm for SSI is $6/7 \approx 85.7\%$ in the Internet environment.

6 Conclusion and Future Research Directions

In this paper, we first developed an efficient algorithm to remove most of the outliers in the packet RTTs obtained from packets captured in the Internet environment. Because of the accumulated data communication delay at the intermediate routers on the Internet, the RTTs obtained from the captured TCP packets usually have outlier values in the context of the Internet. This result makes most of the known network-based detection methods for SSI ineffective in the Internet environment. We then proposed an effective SSI detection algorithm by mining network traffic using an improved version of the k-Means clustering algorithm. With most of the RTT outliers removed, both proposed algorithms work together and make our detection method for SSI developed in this paper more effective and efficient than the existing network-based detection approaches in the literature. We also conducted several well-designed network experiments in the context of the Internet to verify the effectiveness, correctness, and efficiency of the proposed SSI detection algorithm.

Table 2 Standard derivations outputted by the 2-Means clustering algorithm (k = 2) on the seven captured datasets.

Dataset	Standard derivation		
	2 connections	3 connections	4 connections
Dataset-1	3184.3	11 785.8	17 431.3
Dataset-2	**20 645.9**	**2146.2**	**1542.9**
Dataset-3	2138.0	16 407.4	5326.3
Dataset-4	20 744.7	27 048.5	74 598.9
Dataset-5	22 709.1	26 717.7	163 099.7
Dataset-6	25 121.4	26 071.2	85 997.0
Dataset-7	23 549.5	25 468.2	84 584.2

Note: Each dataset contains three captured files of RTTs collected from downstream connection chains with 2, 3, or 4 connections, respectively. The RTT dataset we collected from the connection chain of length 2 achieves the smallest standard derivation on six of the seven datasets we captured (only the output generated on dataset-2 is incorrect and in bold face).

For future research in this direction, we plan to improve our SSI detection algorithms so that they will resist session manipulation by intruders using hacking tools, such as time-jittering and/or meaningless chaff perturbation, in the Internet environment. The work presented in this paper may also be extended by improving the proposed algorithm to remove the packet RTT outliers so that the effective rate of the detection algorithm for SSI can be increased in the context of the Internet.

Acknowledgment

This work was supported by the the National Centers of Academic Excellence in Cybersecurity (NCAE-C) Grant (No. H98230-20-1-0293) at the National Security Agency with Columbus State University, Georgia, USA.

References

[1] A. Blum, D. Song, and S. Venkataraman, Detection of interactive stepping-stones: Algorithms and confidence bounds, in *Proceedings of International Symposium on Recent Advance in Intrusion Detection* (*RAID*), Sophia Antipolis, France, 2004, pp. 20–35.

[2] B. Mathew, UNIX security: Threats and solutions, in *Proc. of Invited Talk Given at the 1995 System Administration, Networking, and Security Conference*, Washington, DC, USA, 1995, pp. 6–36 .

[3] Y. Chen and S. Wang, A novel network flow watermark embedding model for efficient detection of stepping-stone intrusion based on entropy, in *Proceedings of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government*, Las Vegas, NV, USA, 2016.

[4] Y. Zhang and V. Paxson, Detecting stepping-stones, in *Proc. of the 9th USENIX Security Symposium*, Denver, CO, USA,

2000, pp. 67–81.

[5] D. Bhattacherjee, Stepping-stone detection for tracing attack sources in software-defined networks, in *Degree Project in Electrical Engineering*, Stockholm, Sweden, 2016.

[6] D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, Multiscale stepping-stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay, in *Proc. of the 5th International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science*, Zurich, Switzerland, 2002.

[7] X. Wang and D. Reeves, Robust correlation of encrypted attack traffic through stepping-stones by flow watermarking, *IEEE Transactions on Depend-able and Secure Computing*, vol. 8, no. 3, pp. 434–449, 2011.

[8] S. Staniford-Chen, and L. T. Heberlein, Holding intruders accountable on the internet, in *Proc. of IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1995, pp. 39–49.

[9] W. Ding, M. J. Hausknecht, S.-H. S. Huang, and Z. Riggle, Detecting stepping-stone intruders with long connection chains, in *Proc. of the Fifth International Conference on Information Assurance and Security*, Zurich, Switzerland, 2009.

[10] J. Yang, B. Lee, S. S.-H. Huang, Monitoring network traffic to detect stepping-stone intrusion, in *Proc. of the 22nd IEEE International Conference on Advanced Information Networking and Applications* (*AINA 2008*), Okinawa, Japan, 2008, pp. 56–61.

[11] S. S.-H Huang, H. Zhang, and M. Phay, Detecting stepping-stone intruders by identifying crossover packets in ssh connections, in *Proc. of the 30th IEEE International Conference on Advanced Information Networking and Applications*, Fukuoka, Japan, 2016, pp. 1043–1050.

[12] S. S.-H. Huang, R. Lychev, and J. Yang, Stepping-stone detection via request-response traffic analysis, in *Proc. of the 4th IEEE International Conference on Automatic and Trusted Computing*, Hong Kong, China, 2007, pp. 276–285.

[13] K. Yoda, H. Etoh, Finding connection chain for tracing intruders, in *Proc. of the 6th European Symposium on Research in Computer Security*, Toulouse, France, 2000, pp. 31–42.

[14] J. Yang, S.-H. S. Huang, and M. D. Wan, A clustering-partitioning algorithm to find TCP packet round-trip time for intrusion detection, in *Proc. of the 20th International Conference on Advanced Information Networking and Applications-Volume 1* (*AINA'06*), Vienna, Austria, 2006, pp. 231–236.

[15] L. Wang and J. Yang, A research survey in stepping-stone intrusion detection, *EURASIP Journal on Wireless Communications and Networking*, vol. 276, pp. 1–15, 2018.

[16] K. H. Yung, Detecting long connecting chains of interactive terminal sessions, in *Proc. of International Symposium on Recent Advance in Intrusion Detection* (*RAID*), Zurich, Switzerland, 2002, pp. 1–16.

[17] J. Yang and S.-H. S. Huang, A real-time algorithm to detect long connection chains of interactive terminal sessions, in *Proceedings of the 3rd ACM International Conference on Information Security* (*Infosecu'04*), Shanghai, China, 2004, pp. 198–203.

[18] J. Yang and S.-H. S. Huang, Matching TCP packets and its application to the detection of long connection chains, in *Proceedings of the 19th IEEE International Conference on Advanced Information Networking and Applications* (*AINA'05*), Taipei, China, 2005, pp. 1005–1010.

[19] J. Yang and S. S.-H. Huang, Mining TCP/IP packets to detect stepping-stone intrusion, *Journal of Computers and Security*, vol. 26, nos. 7&8, pp. 479–484, 2007.

[20] L. Wang, J. Yang, M. Mccormick, P.-J. Wan, and X. Xu, Detect stepping-stone intrusion by mining network traffic using *k*-means clustering, in *Proc. of* 2020 *IEEE 39th International Performance Computing and Communications Conference* (IPCCC), Austin, TX, USA, 2020, pp. 1–8, doi:10.1109/IPCCC50635.2020.9391521.

[21] M. H. Haghighat and J. Li, Intrusion detection system using voting-based neural network, *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 484–495, 2021.

[22] W. Zhong, N. Yu, and C. Ai, Applying big data based deep learning system to intrusion detection, *Big Data Mining and Analytics*, vol. 3, no. 3, 181–195, 2020.

[23] A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection, *Big Data Mining and Analytics*, vol. 4, no. 1, 18–24, 2021.

[24] Z. Cai, Z. He, X. Guan, and Y. Li, Collective data-sanitization for preventing sensitive information inference attacks in social networks, *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2016.

[25] K. Xu, F. Wang, H. Wang, and B. Yang, Detecting fake news over online social media via domain reputations and content understanding, *Tsinghua Science and Technology*, vol. 25, no. 1, pp. 20–27, 2019.

[26] Z. Cai and X. Zheng, A private and efficient mechanism for data uploading in smart cyber-physical systems, *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2018.

[27] X. Zheng and Z. Cai, Privacy-preserved data sharing towards multiple parties in industrial IoTs, *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[28] Z. Cai and Z. He, Trading private range counting over big IoT data, in *Proc. of the 39th IEEE International Conference on Distributed Computing Systems* (*ICDCS*), IEEE, Dallas, TX, USA, 2019, pp. 144–153.

[29] S. Haas, Security monitoring and alert correlation for network intrusion detection. PhD dissertation, Staatsund Universitätsbibliothek Hamburg Carl von Ossietzky, Hamburg, Germany, 2020.

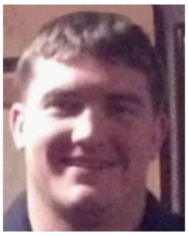
[30] H. Clausen, S. G. Michael, and D. Aspinall, Evading stepping-stone detection with enough chaff, in *Network and System Security*, 2020, pp. 431–446.

[31] V. Paxson, S. Floyd, Wide-area traffic: The failure of Poisson modeling, *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[32] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, An efficient *k*-means clustering algorithm: Analysis and implementation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

Lixin Wang received the PhD degree in computer science from Illinois Institute of Technology, Chicago, IL, USA, in 2010, the MS degree in computer science from University of Houston, Clear Lake, TX, USA, in 2004, the MS degree in applied math from University of Houston, Houston, TX, USA, in 1999, the MS degree in math from Fudan University, Shanghai, China, in 1991. He is an associate professor of computer science with TSYS School of Computer Science in Columbus State University, Columbus, GA, USA. His research interests include network security, wireless networking, and algorithm design and analysis.

Jianhua Yang received the PhD degree in computer science from University of Houston, TX, USA in 2006, and the MS and BS degrees in electronic engineering from Shandong University, Jinan, Shandong, China, in 1990 and 1987, respectively. He is currently working at TSYS School of Computer Science, Columbus State University (CSU), Columbus, GA, USA, as a full professor. He has published more than 60 peer-reviewed journal papers and conference proceedings on cybersecurity. He has been awarded by the National Security Agency (NSA), the National Science Foundation (NSF), and the Department of Defense (DoD) of the United States with amount of more than half-million dollars. His current research interest is computer network and information security.



Michael Workman received the BS degree in computer science-cybersecurity track from Columbus State University, Columbus, GA, USA in 2021. He is currently working for Aflac Inc. as an IT apprentice and a major incident manager. His research interests include intrusion detection and network security.

Pengjun Wan received the BS degree from Tsinghua University, Beijing, China in 1990, the MS degree from the Chinese Academy of Sciences, Beijing, China in 1993, and the PhD degree from the University of Minnesota, Minneapolis, MN, USA, in 1997. He is a fellow of IEEE, and a professor of computer science with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA. His research interests include wireless networks and algorithm design and analysis.