Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6632671, 9 pages
https://doi.org/10.1155/2021/6632671

Research Article

Mining Network Traffic with the *k*-Means Clustering Algorithm for Stepping-Stone Intrusion Detection

Lixin Wang,[1] Jianhua Yang,[1] Xiaohua Xu,[2] and Peng-Jun Wan[3]

[1] *TSYS School of Computer Science, Columbus State University, Columbus, Georgia, USA*
[2] *College of Computing and Software Engineering, Kennesaw State University, Kennesaw, Georgia, USA*
[3] *Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA*

Correspondence should be addressed to Lixin Wang; wang_lixin@columbusstate.edu

Received 15 December 2020; Revised 19 January 2021; Accepted 6 February 2021; Published 3 March 2021

Academic Editor: Wenzhong Li

Intruders on the Internet usually launch network attacks through compromised hosts, called stepping stones, in order to reduce the chance of being detected. With stepping-stone intrusions, an attacker uses tools such as SSH to log in several compromised hosts remotely and create an interactive connection chain and then sends attacking packets to a target system. An effective method to detect such an intrusion is to estimate the length of a connection chain. In this paper, we develop an efficient algorithm to detect stepping-stone intrusion by mining network traffic using the *k*-means clustering. Existing approaches for connection-chain-based stepping-stone intrusion detection either are not effective or require a large number of TCP packets to be captured and processed and, thus, are not efficient. Our proposed detection algorithm can accurately determine the length of a connection chain without requiring a large number of TCP packets being captured and processed, so it is more efficient. Our proposed detection algorithm is also easier to implement than all existing approaches for stepping-stone intrusion detection. The effectiveness, correctness, and efficiency of our proposed detection algorithm are verified through well-designed network experiments.

1. Introduction

This paper is an extension of our work originally presented at the 39th IEEE International Performance Computing and Communications Conference (IEEE IPCCC 2020) [1]. Many attackers send attacking packets to remote target systems through compromised machines, for the purpose of decreasing the chance of being discovered [2–11]. The compromised machines employed by the attackers are referred to as stepping stones. In a stepping-stone intrusion (SSI), an intruder uses a chain of compromised machines on the Internet as relay hosts and remotely logs in these machines by using software tools such as SSH, rlogin, or telnet. The attacker sits in front of his local host and types attacking commands that are relayed via the stepping-stone hosts in the connection chain until the attacking packets arrive the remote target system that is under attack.

Since every TCP connection between a source node and a destination node is independent of other connections even though the connections might be relayed, accessing a remote host via several relayed TCP connections makes it very difficult to determine the attacker's actual geographical location. Because the TCP protocol has such a property, the final target machine could only see the packets from the last connection of the chain. Therefore, it is extremely hard for a target host to learn any information about the actual location of the intruder.

A benefit of launching attacks using stepping stones is that attackers could be hidden behind a long interactive connection. If a SSI could be detected within the active period of attacking, then the session could be cut off and the target system could be protected. Although some researchers worked on the back-tracing of SSI and studied the upstream detection, most researchers focused on downstream SSI detection.

Intruders using SSI could build a connection chain given in Figure 1 using software tools such as SSH to launch their attacks. In Figure 1, we assume that Host 0 is used by the

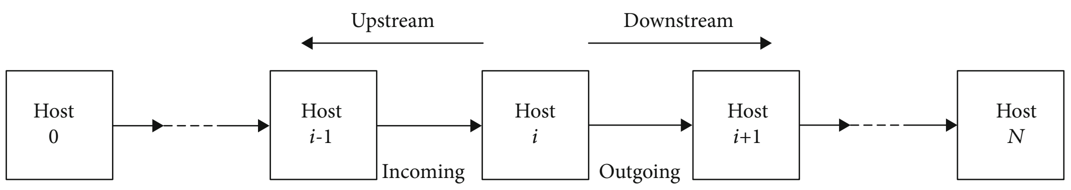

Figure 1: A sample connection chain.

attacker to launch an attack against the target Host N via compromised hosts Host 1, Host 2,..., Host $i-1$, Host i, Host $i+1$,..., and Host $N-1$. SSI detection may occur in any of these stepping stones. The detection program is assumed to reside in Host i which is referred to as a (detecting) sensor. SSI detection is to determine if the detecting sensor Host i is used as a stepping stone. The connection from Host $i-1$ to Host i is called an incoming connection to Host i, and the connection from Host i to Host $i+1$ is called an outgoing connection from Host i. If there is at least one relayed pair between all the incoming connections and all the outgoing connections, then it is highly suspicious that Host i is used as a stepping stone for intrusion.

One type of detection methods for SSI is to compare all the outgoing connections with all the incoming connections of the same host to see if there exists a relayed pair. This type of methods is referred to as *host-based* SSI detection. Quite a few approaches that can be applied to encrypted connections have been proposed since the year 1995 including the deviation-based [12] and time-based approaches [3, 4, 6, 13–18] for SSI detection. This type of methods only focuses a single host and requires only the outgoing packets leaving the host and incoming packets to the host. The main disadvantage of this type of approaches is that it usually introduces high false-positive errors as some legal applications actually use stepping stones to access remote servers. For example, when a client browser requests some resources from a Web server, the Web server may need to access a remote application server that also may need to access a remote database server. Moreover, this type of detection approaches is also vulnerable to the intruder's session manipulation that can be done by using hacking techniques such as time-jittering and/or chaff permutation.

The other type of detection methods to address the problems with the host-based methods is to estimate the number of connections from the intruder's host to the target host (as shown in Figure 1), which is called *the length of a connection chain*. If there exist three or more connections in a connection chain, it means that the user attempts to gain access to a remote target host via three or most machines. It is well-known that the more computers involved in an interactive connection to gain access to a remote server, the slower the traffic of data communication. The threshold number "three" was discovered because most legal applications seldom used three or more stepping stones to access a remote server. This type of detection methods is referred to as the *connection-chain-based or network-based* SSI detection.

However, the connection-chain-based detection methods could produce a false negative error. Let us first introduce two concepts of *upstream and downstream detections*. Estimating the length of the connection chain from the sensor (Host i) to attacker's host (Host 0) as shown in Figure 1 is referred to as *upstream detection*. Similarly, estimating the length of the connection chain form the sensor (Host i) to the target host (Host N) is referred to as *downstream detection*. The length of the entire connection chain from attacker machine to the target machine equals the length of the downstream chain plus that of the upstream one. Unfortunately, upstream detection is extremely challenging and has been a long-standing open problem. Therefore, it is very hard to estimate the length of the entire chain since we can only estimate the length of the downstream connection chain. When the detecting sensor is close to the target machine, the length of the downstream connection chain is trivial, and the upstream connection would dominate the length of the entire chain. In such a case, all malicious intrusions will escape detection due to the false negative errors.

If the downstream connection length is at least two (it means that at least two stepping-stone hosts are present between the sensor computer to the target host) plus the upstream connection length; this makes the length of the whole connection chain be three or more. Therefore, when the downstream connection length is at least two, we conclude that it is highly suspicious that the session is initiated by an attacker. Most existing detection algorithms by estimating the length of a connection chain developed by far only takes the downstream connection into consideration. In this work, the length of a connection chain always means the length of its downstream connection.

Let us give a literature review on network-based detection methods for SSI. The first detection algorithm using a network-based approach was proposed by Yung [19] in 2002. Its basic idea is to estimate the length of a connection chain by computing the RTT of a Send packet and matching it with an ACK packet sent from an adjacent node. Although the false positive error incurred by the approach proposed in [19] is reduced a little bit, this detection approach also suffers from producing false negative errors occasionally, because the method developed in [19] used the ACK packets instead of the actual echo packets. Based on the connection chain set up in [19], the actual echo packets are not available to capture.

Yang et al. [20] developed the step-function detection approach to estimate a connection chain length in a LAN. Compared to the detection algorithms proposed in [19], the step-function approach reduced both the false positive error and the false negative error. In [20], the first connection chain created contains only one connection. Then, the connection chain is extended to two connections, three connections, and so on.

The key idea of the step-function method is to obtain the RTTs by matching a Send packet with its corresponding Echo packet which is available based on the way the connections are set up. However, the approach proposed in [20] only worked when the data communications are limited within a local area network. In the context of the Internet, the same set of authors in [21] proposed the conservative and greedy packet matching algorithm for SSI detection. The weakness of such a conservative algorithm is that it can only match very few packets, which makes the detection method ineffective.

The best known connection-chain-based detection method for SSI is the clustering and partitioning data mining algorithm proposed by Yang and Huang [22]. This paper developed an algorithm for SSI detection by using a clustering and partitioning data mining approach to compute the RTTs of the packets captured from a connection chain. All of the previously known approaches of matching Send and Echo packets only compare one Echo packet with a Send packet at a time. The method proposed in [22] looked through all the possible packets to produce TCP packet matches and made the matching accurate. Using the method in [22], all the Send and Echo packets are captured from a connection chain in a time interval, and then the timestamp differences between a Send packet and those Echo packets received after that Send packet are all calculated. The method proposed in [22] made sure that the correct RTT of each Send packet is among these timestamp differences. The maximum–minimum distance clustering algorithm (MMD) was used to computer the real RTTs. The number of connections in the chain is determined based on the number of clusters generated by the MMD data mining algorithm. The results obtained from the well-designed experiments in [22] indicated that this approach can more accurately estimate the connection chain length than all of the prior detection algorithms for SSI and reduce both the false positive and false negative errors.

A weakness of the MMD clustering and partitioning approach developed in [22] is that a huge number of packets have to be captured and processed. Therefore, this algorithm is not efficient. This paper addresses these issues by introducing a novel algorithm to detect SSI via mining network traffic using the k-means clustering algorithm [23–26]. The k-means clustering algorithm is a data mining approach used to cluster observations into groups of related observations without any prior knowledge of those relationships. It is well-known that the round-trip times will cluster around several levels. As long as most of the outlier values can be removed from the real RTTs in the input file, our k-means clustering detection algorithm proposed in this work can accurately determine the length of a connection chain. Also, our proposed detection algorithm does not require a large number of packets to be captured and processed. Thus, our proposed method for SSI detection is more efficient than MMD clustering and the partitioning approach developed in [22]. The effectiveness of our innovative approach for SSI detection is verified through well-designed experiments by appropriately setting up the connection chains. The experimental result showed that our proposed algorithm can estimate the length of a connection more accurate than the known detection methods presented in the literature.

Table 1: All notations used in this paper.

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, $i = 1, 2, \cdots, k$
C	The set of k centers $\{c_1, c_2, \cdots, c_k\}$
Dataset-j	A dataset of RTT values; $j = 1, 2,$ or 3
σ_{curr}	Standard derivation based on the current cluster
σ_{new}	Standard derivation based on the updated cluster

It is worth mentioning some other work related to SSI detection and/or network security. In order to help researchers to understand how a session is chaffed and develop new tools for detecting SSI as well as resisting intruders' chaff manipulation, [27] developed a C# program to inject meaningless packets into a TCP/IP session. Reference [28] presented an adaptive network intrusion detection method using fuzzy rough set-based feature selection and pattern learning. This paper also introduced a greedy approach of global optimal Gaussian mixture model clustering method in order to extract the intrinsic structure of network instances to achieve highly discernable and stable normal and intrusion pattern libraries for the subsequent network intrusion detection. Many methods have been proposed to detect stepping stones and resist evasive behavior, but so far, no benchmark dataset exists to provide a fair comparison of detection rates. Reference [29] developed a comprehensive framework to simulate realistic stepping-stone behavior that includes effective evasion tools and release a large dataset. The detection rates of eight state-of-the-art methods are evaluated by using this framework. Reference [30] studied query-efficient adversarial attacks against automatic speech recognition systems. This paper presented a novel and effective attack on automatic speech recognition systems. Compared with prior known methods in the literatures, the approach proposed in [30] only needs limited access to the output probabilities of neural networks and achieves extremely high efficiency and success rates. Data privacy relates to how data should be handled based on its relative importance and is closely relevant to network security. Reference [31] developed an approach for uploading data in smart cyberphysical systems, in which both energy conservation and privacy preservation are taken into consideration. Reference [32] proposed a privacy-preserved data sharing structure for industrial Internet of Things, in which several competing clients could exist in distinct stages of the system. Reference [33] developed a mechanism for trading range counting results. Under differential privacy, this mechanism used a sampling method to produce rough counting results which are theoretically verified to achieve unbiasedness, bounded variance, and privacy guarantee.

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

The remaining of this paper is organized as follows. In Section 2, we provide some preliminaries needed for the detection algorithm design. In Section 3, we present our algorithm for SSI detection by mining network traffic using k-means clustering. Performance analysis of our proposed detection algorithm is given in Section 4. Finally, we summarize our paper and discuss some future research directions in Section 5.

2. Preliminaries

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

2.1. Definitions of Send/Echo Packets. Let us use Figure 1 to define Send and Echo packets. Host 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 it 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 sent back to the user's host, letter "l" will be shown on the terminal of the user's host. Such Send and Echo packets are called *a matched pair*. For the other 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 time periods. A Send packet could be echoed by one or Echo packets. Also, an Echo packet could echo one or more Send packets.

2.2. The Distribution of Packets' RTTs for a Connection Chain. A packet RTT of a TCP connection is the sum of four time delays including the processing delay, queuing delay, transmission delay, and propagation delay of the underlying connection [34]. For connection-chain-based SSI detection, the length of a connection chain is estimated by using the packet RTTs. The RTTs obtained from matched Send and Echo pairs can be used to represent the network traffic. Yang et al. [35] showed that the length of a connection chain is equal to the number of clusters that are produced by using the RTTs obtained from the connection chain.

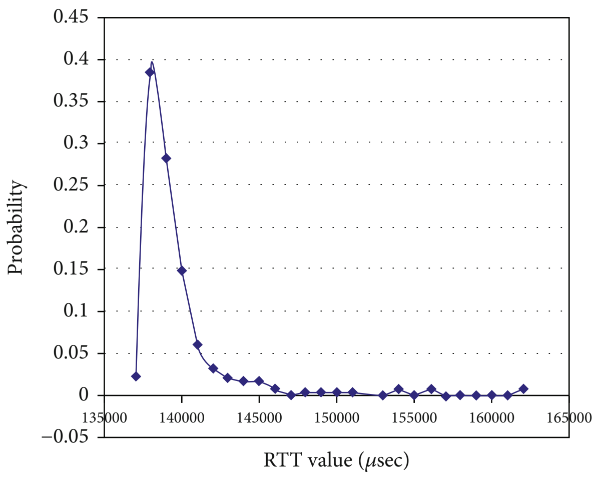

Figure 2: The distribution of packets' RTTs for a connection chain.

Paxson and Floyd [4] found the fact that packet RTTs obtained from a connection chain obey Poisson distribution. This 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 from which packet RTTs obey 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 with unit of microsecond. The values of the RTTs shown in Figure 2 were from the packets collected from a connection chain composed of four connections. With this experiment, most values of the RTTs are close to the mean $\mu = 138{,}500$ (microsecond) of all the RTTs, with at least 95% of the RTT values in the range between 137,000 (microsecond) and 141,000 (microsecond) (refer to Figure 2).

Assume that X is a random variable with mean μ and standard deviation σ. If X follows the Poisson distribution, then we have

$$|X-\mu| \le 2\sigma. \tag{1}$$

This inequality tells us that most values that X takes should be close to its mean μ. The difference between X and its mean μ is upper bounded within 2σ. That is, most RTT values of a connection chain with fixed length must be around its mean within a circle with radius 2σ. Therefore, the RTTs corresponding to a connection chain with fixed length belong to the same cluster with the mean μ as the cluster center. Consequently, the RTT values of a connection chain with distinct lengths belong to different clusters. Thus, if we can design an algorithm to obtain the number of such clusters, then we could easily obtain the matched packets and the length of a connection chain.

It is well-known that many applications such as Web applications use a host as a stepping stone to access a remote server. But for most legitimate applications, it is very seldom to use three or more hosts as stepping stones to access a remote server. The more hosts that are used to access a remote server, the slower the data communication. If there is no intention to hide malicious activities, it is unnecessary to access a remote server via three or more stepping stones as it could make the remote accessing inefficient. Therefore, it is reasonable to assume that if a host uses three or more stepping stones to access a remote server, that host is highly suspicious as a malicious intruder.

3. Stepping-Stone Intrusion Detection Using *k* -Mean Clustering

In this section, we develop an algorithm for SSI detection by mining network traffic using *k*-means clustering. Initially, we create a connection chain with only one connection from the hacker's host to the first stepping-stone host that will be used as the sensor where a detection program such as TCPDump is installed. Then, we extended the connection chain to a chain containing two connections from the sensor host to the target host. Following that, the chain is extended to contain three connections and finally four connections. The length of a connection chain means that of the downstream chain from the sensor host to the target machine. For the connection chain of two connections, its TCP Send and Echo packets are captured, and then the RTTs of the chain are calculated. Since the packets' RTTs of a connection chain follow Poison distribution, most RTTs are around the mean of the RTTs of the connection chain. For the connection chain of three connections, most of its RTTs are similarly around the mean of the RTTs of this chain. The same is true for the connection chain of four connections.

Many data mining clustering algorithms were proposed (see [22] and there references therein). Among all these known data mining clustering approaches, the *k*-means clustering method is a simple unsupervised learning algorithm that gives a solution to the well-known clustering problem [23–26]. In this paper, the *k*-means clustering *algorithm* is used in the design of our detection algorithm for SSI. With the *k*-means clustering *algorithm*, *k* is the number of clusters that is fixed and predetermined.

The key idea is to find the appropriate *k* centers, one for each cluster. These *k* centers need to calculate in a cunning approach so that the overall standard derivation calculated based on these *k* centers is minimized. It is easy to see that we need to place the *k* centers as far away as possible from each other. References [23–26] gave detailed description for the *k*-means algorithm and explanation how this algorithm can be implemented. The *k*-means clustering algorithm is a perfect choice of data clustering in this paper due to the unique properties of the dataset computed from the timestamp differences of Send and Echo packets. Such a method is one of the best approaches that can accurately estimate a connection chain length. Because most legal applications seldom employ three or more intermediate hosts to access a remote server, if a target system is accessed via three or more stepping stones, then it is most likely that the session is manipulated by a malicious hacker for SSI. Therefore, it is sufficient to only use the three values 2, 3, and 4 for the value of *k*, respectively.

Next, we describe the rationale of designing detection algorithms for SSI by using the *k*-means clustering, given three RTT datasets obtained from connection chains of length two, three, or four, respectively. But we do not know which dataset was collected from a connection chain of length two, three, or four. When $k = 2$, we run the *k*-mean clustering algorithm on these three RTT datasets. It is observed that the output generated by the dataset corresponding to the connection-chain length two would produce the smallest standard derivation. According to such an observation, 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. We have a similar observation for $k = 3$. We execute the *k*-means clustering algorithm on these three datasets. It is observed that the output generated by the dataset corresponding to the connection-chain length three would produce the smallest standard derivation among the three datasets. According to such an observation, 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 three. For $k = 4$, we also have a similar observation. As a result, by mining network packets using the *k*-means clustering method, we can estimate the length of a connection chain.

Suppose that $Y = \{y_1, y_2, \cdots, y_n\}$ is a set of n data points. Let $C = \{c_1, c_2, \cdots, c_k\}$ be the set of k centers that will be used by the *k*-mean clustering algorithm. We partition Y into k disjoint subsets $Y_1, Y_2, \cdots, Y_k$, where the data points in Y_j are associated with their nearest center c_j in C for each $1 \le j \le k$ by associating each point in Y to the nearest center in C. For each $1 \le j \le k$, let $Y_j = \{y_{j_1}, y_{j_2}, \cdots, y_{j_Y}\}$. The standard derivation σ of the given dataset Y according to the *k*-means clustering is defined to be the square root of its variance:

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

Let dataset-1, dataset-2, and dataset-3 denote the three RTT datasets obtained from three connection chains of distinct length 2, 3, or 4, respectively. But we do not know which RTT dataset was obtained from which connection chain beforehand. Our proposed algorithm for SSI detection using the *k*-means clustering method can accurately determine which RTT dataset was obtained from the connection chain of length 2, 3, or 4, respectively. Our proposed detection algorithm for SSI is described in Algorithm 1.

Let us explain Algorithm 1 that is used to estimate the length of a connection chain. The value of the integer *k* is between 2 and 4.

At Step 1, the *k*-means clustering algorithm is called on dataset-1. Here is the procedure: initially, we randomly select *k* points in *Y* as the *k* centers for the clustering algorithm that executes the steps below:

```
Input: $k$, dataset-1, dataset-2, and dataset-3.
Output: the RTT dataset obtained from the connection chain of length $k$.
(1) Call the $k$-means clustering algorithm on dataset-1. Assume $\sigma_1$ represents the standard derivation outputted based on Equation (2)
    using the $k$ clusters obtained at the end of the $k$-means clustering algorithm execution
(2) Call the $k$-means clustering algorithm on dataset-2. Assume $\sigma_2$ represents the standard derivation outputted based on Equation (2)
    using the $k$ clusters obtained at the end of the $k$-means clustering algorithm execution
(3) Call the $k$-means clustering algorithm on dataset-3. Assume $\sigma_3$ represents the standard derivation outputted based on Equation (2)
    using the $k$ clusters obtained at the end of the $k$-means clustering algorithm execution
(4) If $\sigma_1 = \min\ \{\sigma_1, \sigma_2, \sigma_3\}$, return dataset-1; /* if $\sigma_1$ is the smallest one among all three standard derivations, the dataset-1 is obtained
    from the connection chain of length $k$ */
(5) If $\sigma_2 = \min\ \{\sigma_1, \sigma_2, \sigma_3\}$, return dataset-2; /* if $\sigma_2$ is the smallest one among all three standard derivations, the dataset-2 is obtained
    from the connection chain of length $k$ */
(6) If $\sigma_3 = \min\ \{\sigma_1, \sigma_2, \sigma_3\}$, return dataset-3; /* if $\sigma_3$ is the smallest one among all three standard derivations, the dataset-3 is obtained
    from the connection chain of length $k$ */
```

Algorithm 1: Algorithm to estimate the length of a connection chain

(i) To generate the k clusters, each point in Y is scanned and assigned to the cluster center whose distance from the cluster center is minimum of all these cluster centers

(ii) Calculate the standard derivation σ_{curr} using the current partition and cluster centers according to Equation (2)

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

(iv) Use these new cluster centers to reproduce the k clusters following the same procedure described in step (i)

(v) Recompute the standard derivation σ_{new} using the updated partition and cluster centers according to Equation (2)

(vi) If $\sigma_{\text{new}} \geq \sigma_{\text{curr}}$ then Exit; otherwise, go back and repeat step (iii) above

When the k-means clustering algorithm exits, the standard derivation is minimized using the cluster centers and partition obtained at the last round of the algorithm. Thus, after the algorithm's last round completes, the standard derivation can no longer be smaller through changing the positions of the cluster centers. According to Equation (2), the value of σ_1 is the standard derivation obtained by using the k clusters calculated at the last round of the k -means clustering.

Similarly, at Steps 2 and 3 of Algorithm 1, the k-means clustering algorithm is called on dataset-2 and dataset-3, respectively. Then, σ_2 and σ_3 are in turn computed. If σ_1 is minimum among the three values σ_1, σ_2, and σ_3, then the dataset dataset-1 is the one that was obtained from the connection chain of length k. The similar conclusions are also true for the two datasets dataset-2 and dataset-3.

Finally, Algorithm 1 will be executed for the values of $k = 2$, 3, and 4. Hence, Algorithm 1 can accurately determine the length of each of these three connection chains from which the three RTT datasets were collected.

4. Performance Analysis of the Proposed Detection Algorithm

In this section, we analyze the performance of our proposed Algorithm 1 above for SSI detection by mining network traffic using the k-means clustering approach. 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 20 collections of datasets, each of which contains three datasets obtained from connection chains of length 2, 3, or 4, respectively. Assume that we do not know the length of the connection chain where each of these datasets was collected from. Our proposed detection algorithm can accurately determine which dataset was collected from a connection chain of length 2, which dataset from a connection chain of length 3, and which dataset from a connection chain of length 4.

In our experiment, we initially created a connection chain with only one connection from the intruder's host to the first stepping-stone host S1 that will be used as the detecting sensor where a detection program such as TCPDump is installed. Then, the connection chain is extended to the host S2 and then to the host S3. Now the length of the downstream chain (from the sensor S1 to the host S3) is two. After that, the connection chain is extended to contain three connections and, finally, up to four connections in the downstream connection from the sensor host S1 to the victim host. See Figure 3 for the connection chain of length four.

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 by using a window of size three. We found that most of its RTTs are actually around the mean of these RTTs of the connection chain. Theoretically, this is true according to the distribution of the packets' RTTs of a connection chain we discussed in Section 2. Then, we

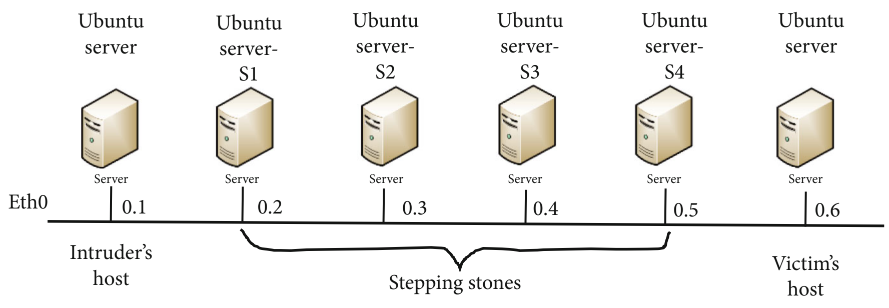

FIGURE 3: A connection chain of length four (from the sensor host S1 to the victim's host).

TABLE 2: The standard derivations outputted by the 2-mean clustering algorithm on the three RTT datasets collected from connection chains of lengths 2, 3, and 4, respectively.

RTT dataset	Chain length	Standard derivation
Dataset-1	2	678238.35
Dataset-2	3	760574.99
Dataset-3	4	758813.90

TABLE 3: The standard derivations outputted by the 3-mean clustering algorithm on the three RTT datasets collected from connection chains of lengths 2, 3, and 4, respectively.

RTT dataset	Chain length	Standard derivation
Dataset-1	2	68994.78
Dataset-2	3	5377.87
Dataset-3	4	6621.19

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 around 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, respectively, represent the datasets of packets RTTs generated from the captured TCP Send and Echo packets collected from the connection chains of lengths 2, 3, and 4, using a window of size 3.

Then, we run the 2-means clustering algorithm (i.e., all items in the dataset are divided into 2 clusters) on the datasets dataset-1, dataset-2, and dataset-3. For each input dataset, this 2-means clustering algorithm outputs the standard derivation calculated using the 2 clusters obtained at the last iteration of the 2-means clustering based on Equation (2). The standard derivations outputted by this clustering algorithm are listed in Table 2. Based on this table, the RTT dataset we collected from the connection chain of length 2 achieves the smallest standard derivation in this case.

Similarly, we run the 3-means clustering algorithm (i.e., all items in the dataset are divided into 3 clusters) on the datasets dataset-1, dataset-2, and dataset-3. The standard derivations outputted by this clustering algorithm are listed in Table 3. Based on this table, the RTT dataset we collected from the connection chain of length 3 achieves the smallest standard derivation in this case.

TABLE 4: The standard derivations outputted by the 4-mean clustering algorithm on the three RTT datasets collected from connection chains of lengths 2, 3, and 4, respectively.

RTT dataset	Chain length	Standard derivation
Dataset-1	2	7542.40
Dataset-2	3	1870.36
Dataset-3	4	1733.15

Finally, we run the 4-means clustering algorithm (i.e., all items in the dataset are divided into 4 clusters) on the datasets dataset-1, dataset-2, and dataset-3. The standard derivations outputted by this clustering algorithm are listed in Table 4. Based on this table, the RTT dataset we collected from the connection chain of length 4 achieves the smallest standard derivation among all three datasets.

Therefore, we can accurately determine the length of the connection chain where each of the three RTT datasets was collected from by using our proposed detection algorithm based on the k-means clustering.

5. Conclusion and Future Research Directions

We proposed in this paper an efficient detection algorithm for SSI by using the k-means clustering algorithm to estimate the length of a (downstream) connection chain. Our proposed detection algorithm using the k-means clustering can accurately determine the connection chain length. Also, the proposed algorithm does not require capturing and processing a large number of TCP packets. Therefore, our proposed detection algorithm for SSI is efficient. Due to the use of the k-means clustering approach, our proposed algorithm is easy to implement as well. Through well-designed network experiments by setting up three connection chains, the effectiveness and correctness of our proposed detection algorithm for SSI are verified. For future research work, one can appropriately modify some steps of the k-means clustering algorithm so that the detection algorithm for SSI can significantly reduce both the false positive and false negative errors compared to all existing approaches. Also, the detection methods based on the k-means clustering require that there should be as less outlier values of round-trip times as possible. Therefore, additional algorithms are needed to remove these outlier values of round-trip times from the input file of the detection algorithm.

Data Availability

All data generated or analyzed during this study are included in this published article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work of Drs. Lixin Wang and Jianhua Yang is supported by the National Security Agency NCAE-C grant H98230-20-1-0293 with Columbus State University, Georgia, USA.

References

[1] 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 *39th IEEE International Performance Computing and Communications Conference (IEEE IPCCC 2020)*, November 2020.

[2] B. Mathew, “UNIX security: threats and solutions,” in *Invited talk given at the 1995 system administration, networking, and security conference*, Washington, DC, April 1995.

[3] S. Staniford-Chen and L. T. Heberlein, “Holding intruders accountable on the Internet,” in *Proceedings 1995 IEEE Symposium on Security and Privacy*, pp. 39–49, Oakland, CA, USA, 1995.

[4] V. Paxson and S. Floyd, “Wide area traffic: the failure of Poisson modeling,” *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[5] L. Wang and J. Yang, “A research survey in stepping-stone intrusion detection,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, Article ID 1303, 15 pages, 2018.

[6] Y. Zhang and V. Paxson, “Detecting stepping-stones,” in *Proceedings of the 9th USENIX Security Symposium*, pp. 67–81, Denver, CO, August 2000.

[7] J. Yang, B. Lee, and S. S.-H. Huang, “Monitoring network traffic to detect stepping-stone intrusion,” in *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, pp. 56–61, Okinawa, Japan, March 2008.

[8] J. Yang, Y. Zhang, R. King, and T. Tolbert, “Sniffing and chaffing network traffic in stepping-stone intrusion detection,” in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 515–520, Krakow, May 2018.

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

[10] S. S. H. Huang, H. Zhang, and M. Phay, “Detecting stepping-stone intruders by identifying crossover packets in SSH connections,” in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 1043–1050, Crans-Montana, 2016.

[11] T. He and L. Tong, “Detecting encrypted stepping-stone connections,” *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 1612–1623, 2007.

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

[13] 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)*, pp. 20–35, Sophia Antipolis, France, September 2004.

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

[15] 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 *5th International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science*, Berlin, Heidelberg, 2002.

[16] W. Ding, M. J. Hausknecht, S. H. S. Huang, and Z. Riggle, “Detecting stepping-stone intruders with long connection chains,” in *2009 Fifth International Conference on Information Assurance and Security*, pp. 665–669, Xi'an, 2009.

[17] Xinyuan Wang and D. Reeves, “Robust correlation of encrypted attack traffic through stepping stones by flow watermarking,” *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 3, pp. 434–449, 2011.

[18] 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 (EEE)*, 2016.

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

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

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

[22] J. Yang and S. S.-H. Huang, “Mining TCP/IP packets to detect stepping-stone intrusion,” *Computers & Security*, vol. 26, no. 7-8, pp. 479–484, 2007.

[23] G. Hamerly and J. Drake, “Accelerating Lloyd’s algorithm for k-means clustering,” in *Partitional Clustering Algorithms*, pp. 41–78, Springer, Cham, 2015.

[24] “Data clustering algorithms: k-means clustering algorithm,” https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm.

[25] P. K. Agarwal and C. M. Procopiuc, “Exact and approximation algorithms for clustering,” *Algorithmica*, vol. 33, no. 2, pp. 201–226, 2002.

[26] 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.

[27] J. Yang, L. Wang, A. Lesh, and B. Lockerbie, “Manipulating network traffic to evade stepping-stone intrusion detection,” *Internet of Things*, vol. 3-4, pp. 34–45, 2018.

[28] J. Liu, W. Zhang, Z. Tang et al., “Adaptive intrusion detection via GA-GOGMM-based pattern learning with fuzzy rough set-based attribute selection,” *Expert Systems with Applications*, vol. 139, article 112845, 2020.

[29] H. Clausen, M. S. Gibson, and D. Aspinall, “Evading stepping-stone detection with enough chaff,” in *14th International Conference*, pp. 431–446, Melbourne, VIC, Australia, 2020.

[30] Q. Wang, B. Zheng, Q. Li, C. Shen, and Z. Ba, “Towards query-efficient adversarial attacks against automatic speech recognition systems,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 896–908, 2021.

[31] 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, 2020.

[32] 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.

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

[34] Q. Li and D. L. Mills, “On the long-range dependence of packet round-trip delays in Internet,” in *ICC'98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM'98 (Cat. No. 98CH36220)*, vol. 2, pp. 1185–1191, 1998.

[35] 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 *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, vol. 1, Vienna, 2006.