Wang and Yang *EURASIP Journal on Wireless Communications and Networking* (2018) 2018:276
https://doi.org/10.1186/s13638-018-1303-2

EURASIP Journal on Wireless Communications and Networking

REVIEW Open Access

A research survey in stepping-stone intrusion detection

Lixin Wang* and Jianhua Yang

Abstract

Attackers on the Internet often launch network intrusions through compromised hosts, called stepping-stones, in order to reduce the chance of being detected. In a stepping-stone attack, an intruder uses a chain of hosts on the Internet as relay machines and remotely log in these hosts using tools such as telnet, rlogin, or SSH. A benefit of using stepping-stones to launch attacks is that intruders can be hidden by a long interactive session. Since each interactive TCP session between a client and a server is independent of other sessions even though the sessions may be relayed, so accessing a server via multiple relayed TCP sessions can make it much harder to tell the intruder's geographical location unless all the compromised servers collaborate with each other and work efficiently. Due to such a nature of TCP protocol, the final victim host can only see the traffic from the last session of the connection chain, and it is extremely difficult for the victim host to learn any information about the origin of the attack. This paper provides a research survey in the area of stepping-stone intrusion detection. Most of the significant approaches developed by far for stepping-stone intrusion detection are included in this paper. These detection methods are put into two categories: host-based and network-based (i.e., connection-chain based), according to whether multiple hosts in the connection chain are involved in the design of detection algorithms. In each category, the detection algorithms are divided into several different subsections based on the key techniques used in the algorithms. At the end of the paper, several important and challenging open problems are proposed in this area.

Keywords: Stepping-stones, Intrusion detection, Connection chain, TCP connection, Sensor, Computer networks

1 Introduction

Attackers on the Internet often launch network intrusions through compromised hosts, in order to reduce the chance of being detected. The compromised hosts used by the attacker are called stepping-stones. In a stepping-stone attack, an attacker uses a chain of hosts on the Internet as relay machines and remotely log in these hosts using tools such as telnet, rlogin, or SSH. On the intruder's local machine, he enters commands that are relayed via the stepping-stone hosts in the chain till they finally reach the victim machine. Since each interactive TCP session between a client and a server is independent of other sessions even though the sessions may be relayed, so accessing a server via multiple relayed TCP sessions can make it harder to tell the intruder's geographical location unless all the compromised hosts collaborate with each other and work efficiently. Due to such a nature of TCP protocol, the final victim machine can only see the traffic from the last session of the connection chain. So it is extremely difficult for a victim host to learn any information about the origin of an attack.

An obvious benefit of using stepping-stones to launch an attack is that intruders can be hidden by a long interactive session. If a stepping-stone intrusion can be detected within the attacking period, then the session can be cut off and the victim machine can be protected. Even though there are still a few researchers working on the traceback of stepping-stone intrusion, most researchers work on stepping-stone intrusion detection (SSID).

Stepping-stone intruders can make a connection chain shown as in Fig. 1 using telnet/rlogin/ssh to launch their attacks. In Fig. 1, we assume that Host 0 is used by an intruder to launch an attack to Host N via compromised hosts Host 1, Host 2, … , Host $i-1$, Host i, Host $i+1$, … , and Host $N-1$. SSID can occur at one of the

* Correspondence: wang_lixin@columbusstate.edu
TSYS School of Computer Science, Columbus State University, Columbus, GA 31907, USA

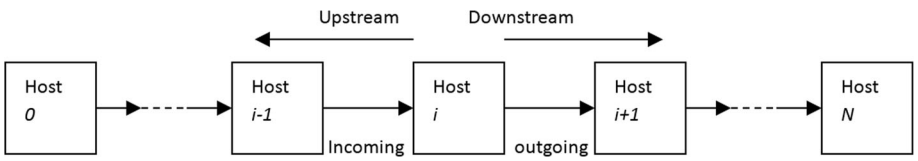

Fig. 1 A sample connection chain. Host 0 is used by an intruder to launch an attack to Host *N* via compromised hosts Host 1, Host 2,..., Host *N* – 1. SSID can occur at one of the stepping-stones. It is assumed that the detection program resides in Host *i* which is called a (detecting) sensor. SSID is to determine whether the 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*

stepping-stones. It is assumed that the detection program resides in Host i which is called a (detecting) sensor. SSID is to determine whether the 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. A necessary condition that Host i is used as a stepping-stone is that there is at least one relayed pair between all the incoming connections and all the outgoing connections.

One type of approach to detect stepping-stone intrusion is to compare all the incoming connections with all the outgoing connections of the same host to see if there exists a relayed pair. This type of approach is called *host-based* SSID. We will discuss all the significant research work for host-based SSID in Section 2. The primary issue of this type of approach is that high false-positive errors can be easily introduced since some legal applications may use stepping-stones to access remote servers.

Another type of approach to overcome the issues of host-based detection is to estimate the number of connections from Host 0 to Host N (as shown in Fig. 1), which is referred to as the length of the connection chain. If there are more than three connections involved in a connection chain, it indicates that the user obviously tries to access Host N via more than three computer hosts. Clearly, the more hosts involved in an interactive session to access a server, the slower the network communication, unless there are something hidden; otherwise, it does not make sense to access a remote sever via more than three hosts. The number "three" is used because it was found that most legal applications rarely used more than three stepping-stones to access a remote server. This type of approach is called *network-based* (or *connection-chain based*) SSID.

Estimating the length of the connection chain from Host 0 to Host i as shown in Fig. 1 is called upstream detection. Similarly, estimating the length of the connection chain form Host i to Host N is called downstream detection. Accurate estimation of the length of the whole connection chain requires both downstream and upstream detections. Unfortunately, performing upstream detection is extremely challenging. Such a problem is still open and remains unsolved. Therefore, it is extremely hard to estimate the length of the whole connection chain. So most researchers in SSID primarily focused on using the length of downstream connection to decide whether there is a stepping-stone intrusion. Compared to using the length of the whole connection chain, this simplified method may introduce false negative errors, but it performs much better than host-based detection as well as largely reduces false positive errors.

The remaining of this paper is organized as follows. In Section 2, we present some significant host-based approaches for SSID. In Section 3, we summarize some typically known network-based (connection-chain based) approaches for SSID. In Section 4, we propose several open problems in this area. Finally, we conclude our paper in Section 5 and provide the funding information of this research work in the declarations section.

2 Host-based SSID

In this section, we present some significant host-based approaches for SSID in the literature. Detecting stepping-stone intrusion is to decide whether a host is used as a stepping-stone. For host-based SSID, determining whether a host is used as a stepping-stone is to examine its incoming and outgoing connections to see if there is a relayed or matched connection pair. If such a pair exists, the host is most likely used as a stepping-stone by an attacker. The host-based approaches for SSID to be discussed include content-thumbprint, time-thumbprint, packet counting, random-walk detection, cross-over packets, and watermarking detection. At the end of this section, we introduce a very recent work for sniffing and chaffing network traffic.

2.1 Content-thumbprint

Using a content-thumbprint is one way to find a relayed connection pair on a host. As shown in Fig. 1, we assume that Host i is used as a sensor. To determine whether Host i is used as a stepping-stone using content-thumbprint, it is necessary to sniff the TCP packets from the incoming connection of the sensor for a certain time interval, e.g., 5 min. Applying a hash method to the contents from all the sniffed packets, we can obtain a hashed result which is called a content-thumbprint *Ctb-in* for the incoming

connection. In the same time interval, we sniff TCP packets on the outgoing connection of the sensor and get another content-thumbprint *Ctb-out*. We can determine whether Host i is used as a stepping-stone by comparing Ctb-in with Ctb-out to see if they are close enough. Of course, this approach to detect stepping-stone intrusion can only apply to *unencrypted* sessions. For the details about applying a content-thumbprint to detect stepping-stone intrusion, please refer to the paper [1].

Another content-based method is proposed in [2] by using an approach called sleepy watermark tracing (SWT). SWT is "sleepy" in that it does not introduce overhead when no intrusion is detected. When an intrusion is detected, the target will inject a watermark into the backward connection of the intrusion and wake up and collaborate with intermediate routers along the intrusion path. By integrating a sleepy intrusion response scheme, a watermark correlation technique, and an active tracing protocol, SWT provides an efficient and accurate source tracing on interactive intrusions through chained telnet or rlogin.

2.2 Time-thumbprint

Since most intruders use encrypted sessions to launch their attacks, content-thumbprint stepping-stone detection does not work in such cases. A time-thumbprint approach (see [3]) was proposed to overcome the primary issue of the content-thumbprint approach. Time-thumbprint uses the timestamp of each packet to decide whether there is a relayed connection pair. This approach can apply to encrypted sessions to detect stepping-stone intrusion because the timestamp of each packet is not encrypted.

We still use the scenario of Host i shown in Fig. 1 to demonstrate the mechanism of using time-thumbprint to detect stepping-stone intrusion. Monitoring and sniffing TCP packets from the incoming and outgoing connections of Host i in the same time interval, we record the timestamp of each TCP packet captured. Here, a TCP packet is either a send or an echo packet. We obtain a timestamp sequence $TS_{in} = \{p_1, p_2, p_3, \ldots, p_i, \ldots, p_{n-1}, p_n\}$ from the incoming connection of Host i, as well as another timestamp sequence $TS_{out} = \{q_1, q_2, q_3, \ldots, q_j, \ldots, q_{m-1}, q_m\}$ from the outgoing connection, where p_i $(1 \le i \le n)$, and q_j $(1 \le j \le m)$ denote the timestamp of each packet captured. Simply comparing the two sequences TS_{in} and TS_{out} to get the similarity, we can decide whether there exists a relayed connection pair. The higher the similarity between TS_{in} and TS_{out}, the higher the probability that the two connections are relayed, indicating that Host i is used as a stepping-stone. So the problem of SSID is converted into the mathematical problem of sequence comparison. For the details of using a time-thumbprint to detect stepping-stone intrusion, please refer to paper [3].

2.3 Packet counting

When an intruder launches an attack using stepping-stones, the commands are indirectly sent to the victim host through a chain of stepping-stones. The use of encrypted connections by such intruders make the detection process much harder and even more difficult if a connection is manipulated by intruders, such as time-jittering and/or chaff-perturbation.

The packet counting method was introduced to handle such challenges for SSID [4]. The strategies proposed in [4] are to identify stepping-stone connections when the attacking packets are encrypted and their timing is jittered. Furthermore, an attacker can inject chaff packets into an attacking stream. He and Tong [4] considered stepping-stone connections subject to packet-conserving transformations by an attacker. In order to defeat intruders' manipulation, two activity-based algorithms were proposed in [4] to detect stepping-stone intrusion. The goal of these two algorithms is to detect stepping-stone intrusion with bounded memory or bounded delay perturbation. This paper also addressed the detection of stepping-stone intrusion if the connection is manipulated by an intruder with both time-jittering and chaff-perturbation. The authors of [4] proved that their algorithms can tolerate a number of chaff packets proportional to the size of the attacking traffic sent from the intruder and have vanishing false alarm probabilities if traffic arrivals follow Poisson distribution.

Under the bounded memory assumption, the algorithm developed in [4] has linear complexity. The key idea used in this algorithm is that the maximum variation statistics diverges unboundedly for independent traffic. But when the memory is limited, it stays bounded for relayed traffic going through a stepping-stone. Under the bounded delay assumption, a timing-based algorithm was derived in [4] based on the idea of matching arriving packets with departing packets, but it has exponential complexity. However, by restricting the search to order-preserving packets, the complexity of this algorithm can be reduced from exponential to linear. Both of these algorithms have no miss detection for their targeting stepping-stone pairs and exponentially decaying false alarm probabilities for network traffic arrivals following Poisson distribution.

In paper [5], Donoho et al. proposed monitoring the incoming stream and outgoing stream of a network at a gateway node to detect stepping-stone intrusion. They considered that not only a host but also a network can be used as a stepping-stone. A pair of incoming and outgoing streams is called a stepping-stone pair if it is part of a stepping-stone attack. Otherwise, it is called a normal pair. The algorithms proposed in [5] attempted to find the stepping-stone connection pairs by analyzing the traffic from both incoming connection and outgoing

connection of a network (also see [6]). It is desirable that the detection strategy does not require synchronization between incoming and outgoing streams. Besides, the connections may be encrypted so that the algorithms cannot rely on the content of the traffic. Furthermore, a careful attacker may even modify the traffic on purpose each time it passes through the machine.

To reduce the false positive error, the two connections can be monitored in more than one time intervals, for example, six distinct time intervals. This means we can obtain a set of number pairs from incoming and outgoing connections, respectively. Each number pair represents the number of packets sniffed in one time interval from the incoming and outgoing connections. So the problem to detect stepping-stone intrusion using packet counting becomes the problem of sequence comparison. The larger the size of the sequence, the lower the false positive detection error.

2.4 Random-walk detection

Using stepping-stone hosts to launch attacks is one of the indirect ways used by hackers for hiding their identities. If a machine is used as a stepping-stone, it must have an incoming connection that originally comes from a hacker's machine and an outgoing connection that eventually goes to the victim host. Also, these two connections must be a relayed pair. In other words, if the two connections are relayed, then the machine must be used as a stepping-stone.

Based on the nature of TCP/IP protocol, the final victim machine can only see the network traffic from the last connection of a chain used by attackers. The information of a TCP/IP connection is only visible to its adjacent host in either downstream or upstream, that is, if an intruder uses a chain of multiple compromised hosts to invade a victim host, only the host having a direct TCP/IP connection to the victim host is visible, but the geographic location about the origin intruder host cannot be obtained. Therefore, it is very challenging for the victim machine to learn any information about the original machine used by the attacker.

To detect stepping-stone intrusions when the network traffic are encrypted, a lot of approaches have been proposed in the literature. One of these approaches is the time-thumbprint method proposed in [3] by Zhang and Paxson based on the periods of activity of the connections between the stepping-stones. This method can be used for SSID if the network traffic is encrypted. But this time-thumbprint approach also has several drawbacks that are listed below:

1. Intruders can easily use time-jittering and chaff-perturbation manipulations to fail SSID;
2. The time-thumbprint approach must use and rely on the synchronized and precise timestamps of the captured packets; and
3. There are many applications which make legal use of some hosts as stepping-stones, but the applications are apparently not malicious. For example, a typical legitimate application is when we access a remote file server and may need to get some information/file from another remote server. In this case, stepping-stones are used legitimately.

A deviation-based approach for SSID was proposed in [7] by Yoda and Etoh. In this paper, the authors calculated the deviations between a known intruder stream and all other concurrent streams on the Internet, compared the packets of streams which have small deviation from the intruder's stream, and used such an analysis to find a set of streams that might match the intruder's stream. Through their observation, the authors claimed that the deviation of two un-relayed connections is large enough to be noted from the deviation of those relayed connections. This deviation-based method has the same issues as the time-thumbprint method proposed in [3]. Also, if there are too many incoming and outgoing connections to/from the host, this approach is inefficient because computing the deviations is too complicated in such a case.

Yang and Huang in [8] proposed a better approach than the one presented in [9] using TCP/IP packet RTTs to detect stepping-stone intrusion. The method proposed in [8] makes the intrusion detection more accurate and significantly decreases both the false positive rate and the false negative rate. The authors designed a clustering–partitioning algorithm to compute the TCP packet RTTs from the timestamps of send and echo packets in a connection chain. This paper showed that the occurrence of TCP packet RTTs in the same level obeys Poisson distribution, which means that most of the packet RTTs in the same level are around its expectation value. The number of RTTs is hard to be manipulated by intruders because most chaff packets do not have responses so it is hard to match. When matching send and echo packets, the algorithm proposed in [8] filters out the un-chaffed send or echo. If we can obtain respectively the numbers of RTTs for incoming and outgoing connections, then it is trial to compute the difference of these two RTT numbers. Although this difference may vary, it is bounded regardless of whether chaff-perturbation or time-jittering is performed by intruders.

In order to resist network intruders' evasions using manipulation tools such as time-jittering or chaff-perturbation, a packet count difference-based approach was proposed in [10] by Blum et al. to detect stepping-stone intrusions through checking the difference of send packet

counts between two connections, one from the sensor and the other one to the sensor. The key idea in this approach is that the difference of Send packet counts is bounded if and only if the two connections are relayed. In other words, if two connections are not relayed, such a difference would not be bounded. This method has also its drawbacks. In order to detect a chaff connection even with a small amount of chaff packets, this approach requires to capture a large number of packets. Therefore, in practice, Blum's packet count difference-based method is difficult to implement and may not work in detecting stepping-stone intrusions when a number of chaff packets were injected by attackers.

A better approach was proposed by He and Tong in [4] in resisting intruders' chaff evasion with tolerance of a number of chaff packets injected. Their algorithm is called DBDC (detect-bounded-memory-chaff) for detecting stepping-stone intrusion with bounded memory or bounded delay perturbation. The authors claimed that their algorithm DBDC can tolerate a number of chaff packets proportional to the size of the attacking traffic in resisting chaff evasion of intruders. This paper proved that the chaff-packets injection rate of DBDC can be at least $1/(1 + \lambda\Delta)$, where Δ is an upper bound of the packets delay and λ is a parameter of a Poisson distribution which indicates the expected number of occurrences during a given time interval. Clearly, if algorithm DBDC uses smaller values for λ and Δ, then it can tolerate more chaff packets. However, this would also make DBDC have a high false positive rate for a wide range of normal network traffic.

Another innovative approach was proposed in [11] by Yang et al. in resisting intruders' chaff evasion using the idea of random-walk. The authors in this paper discovered that a random-walk process can be used to model the differences between the number of requests and the number of responses.

In this section, we present two algorithms for SSID using the random-walk approach. One is the routine random-walk detection algorithm proposed in [11], and the other one is the RTTs-based random-walk detection algorithm proposed in [12].

Routine random-walk detection: First, we introduce the routine random-walk algorithm proposed in [11] for SSID.

We capture and analyze an interactive TCP connection that is established using the open source tool OpenSSH for a given period of time. We will capture all the send packets in the outgoing connection of a sensor and all the echo packets in the incoming connection and store them in two different sequences respectively represented by S (with n send packets) and E (with m echo packets).

In an interactive TCP session, the user will enter a command such a Linux command "ls" by typing a list of letters (for example, l, s, etc.) and then execute the command on the server machine. The result responses from the remote sever will be a number of packets containing the output generated by executing the command. In general, the user machine will receive an Echo packet as a response whenever the user enters one letter on his machine as a command or a part of a command. If the command is a single letter, the packet is called a single-letter send or a single-letter echo, respectively. To this end, we only consider the single-letter send packets of the outgoing connection and the single-letter echo packets of the incoming connection. If the two connections of the sensor are a relayed stepping-stone pair, then the number of the send packets in the outgoing connection should be very close to the number of the echo packets in the incoming connection of the sensor.

Figure 2 contains host h_i with one incoming connection C_i^1 and one outgoing connection C_i^2; the incoming connection C_i^1 has one request (send) stream $S_i^{(1)}$ and one response (echo) stream $E_i^{(1)}$, and the outgoing connection C_i^2 has one request stream $S_i^{(2)}$ and one response stream $E_i^{(2)}$, where the index i represents the host h_i.

Considering a time interval of a pre-determined length, we analyze the packets captured during the same time interval from both the incoming connection and outgoing connection of host h_i. Let $N^{(1)}$ denote the total number the TCP packets in the incoming connection of host h_i, $N^{(2)}$ the total number the TCP packets in the outgoing connection of host h_i. The difference between these two numbers $N_i^{(1)}$ and $N_i^{(2)}$ is denoted as N_i^{Δ}. That is:

$$N_i^{\Delta} = N_i^{(1)} - N_i^{(2)}.$$

The paper [11] proved that if the two connections are relayed (i.e., they are a stepping-stone pair), then the behavior of the difference N_i^{Δ} follows a random-walk process [13]. The value of N_i^{Δ} may be negative or positive, but must be close to zero, that is, abs(N_i^{Δ}) is bounded. In other words, if the difference N_i^{Δ} calculated

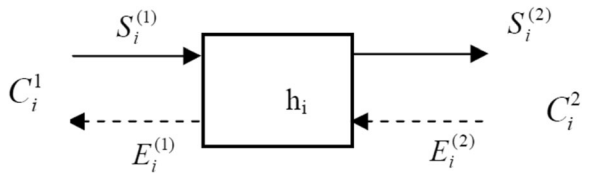

Fig. 2 Modeling of a stepping-stone host. This figure contains host h_i with one incoming connection C_i^1 and one outgoing connection C_i^2; the incoming connection C_i^1 has one request (send) stream $S_i^{(1)}$ and one response (echo) stream $E_i^{(1)}$, and the outgoing connection C_i^2 has one request stream $S_i^{(2)}$ and one response stream $E_i^{(2)}$, where the index i represents the host h_i

from the incoming and outgoing connections is a random-walk process, then paper [11] concluded that the two connections are relayed and they are a stepping-stone pair.

RTT-based random-walk detection: In this subsection, we present the RTT-based random-walk algorithm proposed in for SSID.

In the case of no packet dropped, combined, or decomposed, the number N_{in} of packets in an incoming connection must be the same as the number N_{out} of packets in the corresponding outgoing connection. Even if packet dropping occurs during the course of data communication, these two numbers should be very close with each other, compared with those of any two connections that are non-relayed. However, hackers can manipulate connections through chaff-perturbation or time-jittering. If either an incoming connection or an outgoing connection is manipulated by an attacker, there would be a big gap between the two numbers N_{in} and N_{out}. With chaff-perturbation and time-jittering, it is possible that the intruder may be evaded from stepping-stone detection. All previously known approaches in the literature including the papers [8, 10, 14, 15] assumed that:

1. If an intruder uses the time-jittering technique to hold a packet at any place, the holding time of any packet must be bounded;
2. If an intruder uses the chaff evasion approach to insert meaningless packets into interactive connections at any time, the inserting rate operated by the attacker must be bounded.

The paper [12] proposed a new method to determine whether an incoming connection and an outgoing connection are a relayed pair using the number of RTTs in a connection together with the random-walk approach, instead of using the number of packets monitored. This new approach can defeat intruders' chaff-perturbation and time-jittering manipulation without the two assumptions described above that were made in all prior work.

Next, we introduce the approach proposed in [8] for computing the packet RTTs of a TCP/IP session.

2.4.1 Computation of packet RTTs

Again, we use Fig. 2 to model a stepping-stone host, say, the host h_i. The paper [8] developed a data mining approach (a clustering–partitioning algorithm) to find the round-trip time from the timestamps of TCP Send and Echo packets. This paper showed that the occurrence of TCP packet RTTs in the same level obeys Poisson distribution, which means that most RTTs in the same level are around its expectation. The key idea to compute the packet RTTs is to match the Send and Echo packets in the same session. The clustering–partitioning algorithm can filter out the chaffed Send or Echo packets and detect intruders' evasion.

2.4.2 RTT-based random-walk detection algorithm

We use N_i^{RTT} to represent the number of RTTs for the incoming connection of host h_i, N_o^{RTT} the number of RTTs for the incoming connection of host h_i, and Δ_{io}^{RTT} the difference between these numbers. The value of this difference can be used to determine the probability that the two connections are relayed. The larger Δ_{io}^{RTT}, the lower the probability that the two connections are relayed, and the lower the probability that the host h_i is used as a stepping-stone.

The authors of [12] observed that if the two connections of a host are relayed, then the value of Δ_{io}^{RTT} must follow a random-walk behavior round zero. Due to this random-walk property of Δ_{io}^{RTT}, the value of Δ_{io}^{RTT} must have an upper bound, say Γ, that is, $|\Delta_{io}^{\text{RTT}}| < \Gamma$.

Therefore, the two connections are relayed if and only if the above inequality holds. The paper [12] proposed an RTT-based random-walk algorithm RBRW for SSID. Yang and Zhang [12] presented a technical proof and showed that the algorithm RBRW can resist intruders' evasion from detection, such as chaff-perturbation and time-jittering. Chaff-perturbation is a tool that allows intruders to insert some meaningless packets (chaff) which can make intruders escape from detecting methods. These chaffed packets are meaningless, and they must be excluded from the connection before they arrive at the victim host at the end. Time-jittering means that intruders can hold any packets for a random amount of time to evade detection. In practice, intruders do not perform time-jittering to the Echo packets. If the send packets are held by an intruder for certain amount of time and then release, it will not affect the number of RTTs because the corresponding echo packet also has the same delay. The timestamp difference between the delayed Send and Echo packets is thus not changed.

2.5 Cross-over packets

Most of intruders use long connection chains of stepping-stones to reach a victim target at the end of the chain in order to avoid being detected. The previously known SSID work has concentrated on detecting intermediate stepping-stones, not the victim host at the end of a chain. Because a stepping-stone host can perform timing and correlation analysis with all of the information sent between the attacker and the victim, the detection of a malicious connection chain is much more challenging from a victim's perspective than at intermediate stepping-stones. The packets from the intermediate stepping-stone to the target victim and back

form a closed loop. However, these intermediate stepping-stone-based detection approaches have several flaws. Obviously, most of the benefits go to the target victim at the end of the chain. Also, the intermediate stepping-stone is only able to gauge the maliciousness of a connection by the number of downstream connections it detects. If the stepping-stone host is very near the target victim in the connection chain, it is very difficult to distinguish a malicious chain from a benign connection.

Victim-based detection attempts to address these issues. It is much more important for a host to protect itself from being a victim host. This method also has some challenges to implement. Thanks to the nature of tunneled SSH connections and the fact that SSH is an interactive terminal session, estimating the full RTT for the length of a connection chain is usually very difficult. During an SSH session, there is no point in time at which the server sends data to the client and the client's machine automatically sends a reply back to the server. Also, the SSID completely relied upon the time difference between the attacker sending data downstream and a reply from the victim at the end of the chain passing back.

The paper [16] proposed an approach to detect long SSH connection chains at the victim host. This method of detection uses an approach to investigate the time delay between the time a user presses enter to finish a command and the time that the user types the next character. With the user's typing speed taken into consideration, we can estimate if the user is connected through a long or a short connection chain.

The key idea of this approach is to find the time difference between the server sending a response to the client and the clients' machine sending the next packet to the server with relative certainty. The time difference typically represents the full round-trip time plus the time it takes the user to generate the next packet (via keystroke) as follows:

$$\begin{aligned} \text{Time Diff} &= \text{Time to send an echo packet} \\ &\quad + \text{User delay time} + \text{Time to send the next packet} \\ &= \text{Full round trip time} + \text{User delay time} \end{aligned}$$

In this formula, the Echo Time of the previous Echo packet and Send Time of the next packet are combined in order to derive an estimate round-trip time (eRTT).

Thus, if the user delay time (the time needed by the user to type the next key) is subtracted from this time difference given above, the full round-trip time remains. The paper [16] seeks to estimate the user delay time in order to find the full round-trip time. More specifically, the paper calculated the time difference between the client's typing of a keystroke to submit a command and the client's typing of next keystroke to enter a new command. Having this time difference obtained, then the authors analyze this time gap based on several other features of the connection. Based on all these information, the authors try to find the full round-trip time and finally are able to estimate the length of the client's connection chain.

The user delay time is estimated to be the client's average typing speed. While this estimation does not account for the time the user might spend reading/thinking before starting to type, the method proposed in this paper try to target those pairs of commands which require minimal cognitive delay. By subtracting the estimated user delay from the total gap time, we obtain the estimated full round-trip time (eRTT).

We need to examine some features of an SSH connection session as we want to find accurately the measures such as the user typing speed and the occurrence of a new command. Session characteristics which allowed the detection of keystrokes, new commands, and nearest connection round-trip time are all needed for the analysis.

Keystroke detection—The TCP header of all packets in TCP sessions holds information about the source port, destination port, sequence number, and acknowledgment number of the packet among other pieces of information. The latter two numbers were used extensively to detect nearly all of the desired SSH session characteristics.

Command detection—The intuition behind command detection is that after the client enters a command, the amount of data that is sent back will be large enough to exceed one block size for the encryption algorithm being used.

Near connection round-trip time detection—After every packet sent by the client to the server, the server responses with an ACK packet to the last host in the connection chain. The last host then automatically sends an ACK packet back to the server. The RTT to the nearest host can then be calculated by investigating the time difference between the server sending the ACK and the client replying with its own ACK back to the server.

The paper calculated an eRTT to distinguish a long connection chain from a short one. The authors pick the time gap between an echo packet and the next send packet received at the victim's host and then use this time gap to subtract the average user delay. The paper argued that the value of eRTT can be used to differentiate the chain length.

The paper [17] improved the accuracy rate of detecting long connection chains based on the paper [16]. Both papers make the same assumption that there exist packet crossovers in a long connection chain. Huang et al. [17] further assumed that there are more crossovers generated in a long connection chain than in a short

one. The authors of [17] used these assumptions to identify a long connection chain.

This paper investigated the relationship between the number of crossovers and the length of a given connection chain. The goal is to distinguish and identify long connection chains from short ones. A connection chain is defined to be "a short chain" if its length is equal to one connection. One connection is defined as a session between an SSH client and an SSH server. A connection chain is defined to be "a long chain" if its length is at least three connections. All three connections in their experiment were routed on the Internet, not on any local LAN.

If a connection chain is long enough, the RTT of a packet may be longer than the time interval between two consecutive keystrokes. For data transfer, the client is allowed to send further packets without waiting for the response to the request. Thus, if the interval of a client's keystroke is longer than the RTT, the response packet will arrive at the client's host before another request is sent out. On the other hand, if the time interval of a client's keystroke is shorter than the RTT of the previous packet, there will be two or more consecutive request packets sent before a response packet arrives. Therefore, the probability that an RTT is greater than the time interval of the keystroke is higher in a long connection chain. When a response packet arrives at the client's machine later than another request packet is sent out, this response packet will cross over the coming request packet on its way before arrival, which is called a "crossover". The two crossovers occurred in such a situation are shown in Fig. 3.

The algorithm proposed in [17] is based on the existence of the packet crossover and its relationship with the length of a chain. Obviously, a large number of crossover packets can change the distribution of the packet gaps. This algorithm captures those packet gap variances resulting from a large number of packet crossovers. The algorithm can be used to distinguish and identify long connection chains from short ones.

Using RTT is an appropriate way to estimate the length of a connection chain. When a monitoring algorithm is running at the target server, the RTT cannot be calculated by using the algorithm. Instead of calculating the RTT, we can calculate an upstream RTT (uRTT), which is defined as the time gap between sending a response packet and the receiving of the next request packet at the target server. An example of RTT and uRTT is provided in Fig. 3. Generally speaking, the uRTT does not represent the real RTT because there is possible a delay before next request is being sent.

We focus on the minimum of all uRTTs, some of which are much smaller than the real RTTs. This happened because of the existence of the packet crossovers in a long connection chain. For example, if we compute the uRTT of the second pair of timestamps at Host 4, the $uRTT_2$ is much smaller than the RTT or the first $uRTT_1$ in Fig. 3. The reason for this difference is precisely due to the crossover of a response packet (Echo 2) with a request packet (Send 3).

From the experiments conducted in [17], the paper concluded that long connection chains generate a number of packet crossovers. Based on observations, the paper claimed that the number of crossovers is proportional to the length of a connection chain.

2.6 Watermarking detection

To detect the attacker's machine behind the stepping-stone hosts, we need to correlate the incoming and

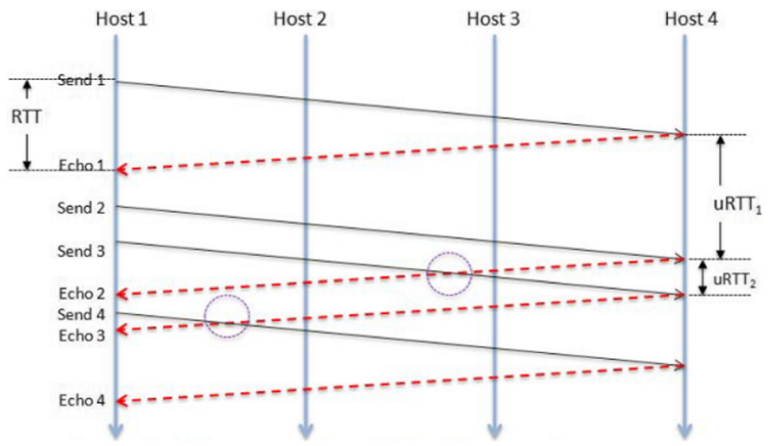

Fig. 3 Illustrations of an uRTT and two packets crossovers along a long connection chain with three hops. This figure provides an example of RTT and uRTT. We focus on the minimum of all uRTTs, some of which are much smaller than the real RTTs. This happened because of the existence of the packet crossovers in a long connection chain. If we compute the uRTT of the second pair of timestamps at Host 4, the uRTT2 is much smaller than RTT or the first uRTT1 in the figure. The reason for this difference is precisely due to the crossover of a response packet (Echo 2) with a request packet (Send 3)

outgoing connection sessions of a stepping-stone host. To resist attempts at correlation, the intruder may encrypt network traffic. The proposed timing-based correlation methods have been proven to be quite effective in correlating encrypted connections, as the time stamps of packets are not encrypted. However, timing-based correlation methods are subject to timing perturbations performed by attackers at stepping-stone hosts.

The authors in [18] proposed a watermark-based correlation scheme that was developed specifically to be effective against timing perturbations. Unlike most previous timing-based correlation methods, the key idea of the watermark-based approach proposed in [18] is that a unique watermark is embedded into the encrypted traffic flows by slightly adjusting the time stamps of selected packets. The embedded watermark in the encrypted flow gives us a number of advantages over previously known timing-based correlation approaches in resisting timing perturbations performed by the intruder. In contrast to all the existing correlation approaches, the watermark-based correlation in [18] did not have any limitations about the distribution or random process of the original inter-packet timing of the packet flow. In theory, it is possible for the watermark-based approach proposed in [18] to achieve almost 100% correlation true positive rate and almost 0% false positive rate at the same time if the packet flows are sufficiently long. This paper not only identified the accurate quantitative tradeoffs between the achievable correlation effectiveness and the defining characteristics of the timing perturbation, but also developed, when the amount of timing perturbation is given, a provable upper bound on the number of packets needed to achieve a desired effectiveness of correlation. Furthermore, the watermark-based approach proposed in [18] requires substantially fewer packets than any previously known timing-based correlation approach to achieve a given level of robustness.

Network flow watermarking approaches have been used to detect stepping-stone intrusion, including watermark embedding scheme and detection scheme. Most known watermark embedding schemes use a randomly selected operation time interval and then generate a watermark sequence in carrier flow of packets. Such existing approaches have an issue that the randomness of watermark operating can make the watermark be vulnerable for security attacks. The authors of [19] proposed another watermark embedding scheme based on entropy to overcome this problem. They first use entropy analysis to pre-process the carrier traffic flow and then determine the optimum time intervals for embedding the watermark. Within these determined time intervals, the watermark was randomly embedded. The authors of [19] proved analytically and verified by experiment that the proposed scheme is robust for timing perturbation by intruders and the embedded watermark is invisible for intruders. This new watermarking approach based on entropy also improved the stepping-stone detection rate and required less number of packets observed. The key idea of this method is to hide information in network traffic flow. All the watermark information are embedded in the time intervals of a large amount of information by preprocessing the carrier flow using entropy.

2.7 Sniffing and chaffing network traffic

Since stepping-stone hosts are widely used to launch attacks by intruders on victim machines on the Internet, a lot of stepping-stone intrusion methods have been proposed. Most of these proposed approaches need to sniff and analyze network traffic to detect stepping-stone intrusions. The paper [20] discussed how to write a C# program to sniff TCP/IP packets. It is well known that some intruders try to evade detection by manipulating TCP/IP connection sessions [21], such as using the technique of chaff-perturbation. In order to help SSID researchers understand how TCP/IP sessions are manipulated, [20] introduced a tool called Fragroute which can be used to inject meaningless packets into TCP/IP sessions on the Internet. Such results will help develop more advanced approaches not only detecting stepping-stone intrusion, but also resisting intruders' manipulation. Unlike other packet-sniffing tools such as Wireshark or Tcpdump, self-making programs to sniff network packet can be easily integrated into various detecting approaches that were proposed in the literature. Session manipulation can help intruders escape from detection. Therefore, knowing how to manipulate TCP/IP sessions can help SSID researchers develop more advanced detection approaches to resist intruders' manipulation in detecting stepping-stone intrusion.

3 Network-based SSID

If a host is detected to be compromised as a stepping-stone, we can only say that it is highly suspicious the session is an intrusion. But we cannot conclude whether an intrusion must exist. Otherwise, false positive error would be introduced. So stepping-stone intrusion can be detected depending on if a host is used as a stepping-stone, but the price is to introduce false alarms. In some cases, the error is fatal. The approaches we have mentioned in Section 2 can detect if a stepping-stone exits, but have their limits in terms of detecting the existence of a stepping-stone intrusion.

To avoid/reduce false positive error for SSID, new approaches need be developed [8, 9, 22–24]. Using one host as a stepping-stone is very common in legitimate applications, but we also found that it is rare to use more than three hosts as stepping-stones in legitimate

applications. The rationale behind this is clear. The more hosts to pass through to access a target, the slower the network traffic would be. If there were no malicious behavior to be hidden, it would be not wise or necessary to access a target indirectly via more than three hosts because this would generate lots of traffic and make the accessing very inefficient. So it is reasonable to assume that anybody doing so would try to hide some malicious activities.

So one approach of detecting stepping-stone intrusion is to estimate the number of compromised hosts used as stepping-stones in a connection chain. In other words, this detection method is to estimate the length of a connection chain. The longer the connection chain is, the more suspicious the connection is. It makes a more reasonable sense to determine an intrusion based on the number of hosts used as stepping-stones, other than a single host detected to be used as a stepping-stone. As we mentioned before many times, detecting intrusion based on stepping-stone detection may bring false positive errors. Similarly, detecting intrusion using the length of a connection chain may incur false negative detection error. The reason is that we can only estimate the length of the downstream section in a connection chain, other than the length of the whole connection chain. If a sensor happens to be close to the victim host, the length of the downstream connection chain would be trivial. In such a case, the upstream connection would dominate the length of the whole chain, but it cannot be estimated. In this scenario, the intrusion would escape from the detection. But this approach can definitely bring down false positive error because it does not count the upstream connection. If a downstream connection has more than three hosts used as stepping-stones, plus upstream connection, it would definitely be more than three compromised hosts. Obviously, if we want to minimize false negative detection error, it is necessary to know the length of the upstream connection which is currently still an open problem even though some researchers proposed several methods toward solving this problem. In the following, if we mention the length of a connection chain, it always means a downstream connection.

There are three basic approaches proposed to estimate the length of a connection chain. One approach is proposed in [9] by Yung in 2002; the second one is using a step-function proposed in [22] by Yang and Huang in 2004; and the third one is the clustering-partitioning data mining approach proposed in [8] by Yang and Huang in 2007. Even though the ideas used to estimate the length of a connection chain are different, all the three approaches use RTT to represent the length of a connection chain. The concept of RTT discussed here is slightly different from the one used in most computer network textbooks. In any computer network textbook, an RTT indicates the sum of a packet delivery and response times between a computer host serving as a sender and another computer host serving as a receiver, assuming no other computer hosts existing in between the two hosts (i.e., there are only routers in between). But for all the three approaches we will discuss here, an RTT means the round-trip time between two computing hosts, and there might be many other hosts existing in between the two.

At the end of this section, we introduce a recent work on stepping-stone detection in software-defined networks (SDN).

3.1 Yung's approach

In order to detect stepping-stone intrusion in a lower false positive rate, [9] by Yung proposed to estimate the length of a connection chain. As shown in Fig. 4, instead of estimating the number of hosts used as stepping-stones, Yung estimates the connection length from H_1 to H_{n+1}. To obtain how long the downstream connection chain is, Yung introduced using the length of connection from H_1 to H_2 as a yardstick to roughly measure how long the length of the connection is from H_1 to $Hn+1$.

The timestamp gap between a send packet and its matched echo packet collected at H_1 (the sensor) can be used to indicate the length of the connection chain from H_1 to H_{n+1}. It is denoted as $\mathrm{RTT}_e = t_e - t_s$. If we want to use the connection from H_1 to H_2 as a scale to measure the length of other connections, the connection length from H_1 to H_2 must be estimated. It is apparently not correct to use the timestamp gap between a send and its matched echo to represent the length of the connection from H_1 to H_2. What Yung used is the timestamp gap between a send packet and its acknowledgement packet

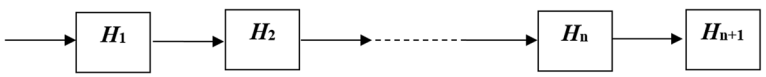

Fig. 4 Estimation of the connection length from H1 to Hn + 1. The timestamp gap between a send packet and its matched echo packet collected at H1 (the sensor) can be used to indicate the length of the connection chain from H1 to Hn + 1. The timestamp gap between a send packet and its matched echo packet collected at H_1 (the sensor) is denoted as $RTT_e = te - ts$. It is apparently not correct to use the timestamp gap between a send and its matched echo to represent the length of the connection from H_1 to H_2. What Yung's approach used is the timestamp gap between a send packet and its ACK packet collected at H_1. We denote this gap as $RTT_a = t_a$-t_s. The ratio RTT_a/RTT_e can approximately tell how long the connection from H_1 to H_{n+1} is

collected at H_1. We denote this gap as $\mathrm{RTT}_a = t_a - t_s$. Since H_2 is directly connected to H_1 in an ssh connection, so any packet sent from host H_1 is acknowledged at host H_2. The acknowledgement will sooner or later go back to H_1. Upon receiving a packet, an acknowledgement packet would be generated immediately. From the packet acknowledgement process, we understand even though it is not accurate to use RTT_a to represent the length of a connection chain, it still makes some sense. The reason that it is not accurate is because RTT_a tends to be smaller due to lacking of packet processing time. Any acknowledgement packet is generated at transport layer which needs less time than an echo packet which is generated at application layer in a ssh connection.

The ratio $\mathrm{RTT}_a/\mathrm{RTT}_e$ can approximately tell how long the connection from H_1 to H_{n+1} is. If this ratio is close to 1, it indicates the downstream connection length is not long. However, if this ratio is close to 0, it strongly indicates the downstream connection length is long. Yung verified his idea in [9] using the computers located throughout the USA and some of them located in Europe.

3.2 Step-function

The step-function approach proposed in [2] takes a different way from Yung's method. It is an approach for detecting stepping-stone intrusion by estimating the length of a connection chain. It collects and matches all the send and echo packets to compute the RTTs for all send packets. The RTTs would form different steps since the RTTs belong to different stages of a connection chain would go to different clusters and each of them can be considered as one step. Simply counting the number of steps can tell us how many hosts are used as stepping-stones.

As shown in Fig. 4, host H_1 is the sensor where we can monitor a connection chain which passes through hosts H_2, H_3,. . . , and finally reaches the victim host H_{n+1} at the end. As discussed before, if we can collect all the TCP packets when the chain only extends to H_2 from H_1 and compute the RTT for each send packet, we would get a set of RTTs which are different but are close enough in terms of their values. We denote this set of RTTs as RTT_1. When the chain extends to host H_3 from H_2, we would get another set RTT_2. The difference between RTT_1 and RTT_2 is that RTT_1 represents the connection chain that has only one connection from the sensor, but RTT_2 represents the connection chain that has two connections from the sensor. Most of the values in RTT_2 are larger than those in RTT_1.

Similarly, as long as the chain has one more connection extended, we would get a different RTT set. At the end of connection chain, we would get RTT_1, RTT_2, RTT_3, . . . , RTT_n. If we put all the RTT sets into a two dimensional coordinate system, we get different clusters with each cluster forming one step. The number of steps can tell the number of connections in the chain which is also the number of computer hosts connected in the chain. The core of the step-function approach is to match the send and echo packets; therefore, each RTT can be computed correctly. The step-function approach uses first-match packet matching algorithm which is a key step. This approach was tested at a local area network in [22], and OpenSSH was used to establish an interactive session connecting seven hosts with six connections from host H_1 to host H_7.

3.3 Clustering-partitioning

The clustering-partitioning data mining approach proposed in [8] is a method to estimate the length of a connection chain with no need to match any TCP/IP packets. In some cases, matching TCP/IP packets is infeasible. Other than matching packets first, this approach makes use of the distribution of RTTs to find the RTTs using a data mining approach.

It is well known that TCP/IP is a protocol between two directly connected hosts. This means host h_i can only know host h_{i+1} is connected with it (see Fig. 5). Host h_i has no idea about the connecting situation after h_{i+1} in the downstream of the connection. If we monitor the outgoing connection of the host h_i, what we know is the TCP/IP packets coming from h_i and going to host h_{i+1}, rather than any other hosts from h_{i+2} to h_{n+1}. But if there is a session between h_i and h_{n+1}, each packet sent from host h_i must be acknowledged by host h_{i+1} first and then forwarded to the following hosts of the chain until to host h_{n+1} at the end of the chain (also called the destination host). Even though the echo of a send from h_i comes from host h_{i+1} from the point of h_i, the reality

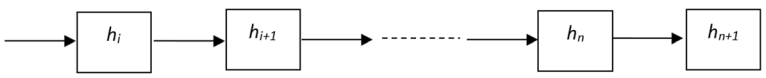

Fig. 5 An interactive connection chain. Host h_i can only know host h_{i+1} is connected with it. It has no idea about the connecting situation after h_{i+1} in the downstream of the connection chain. If we monitor the outgoing connection from the host h_i, what we know is the TCP/IP packets coming from h_i and going to host h_{i+1}, rather than any other hosts from h_{i+2} to h_{n+1}. But if there is a session between h_i and h_{n+1}, each packet sent from host h_i must be acknowledged by host h_{i+1} first and then forwarded to the following hosts until to the destination host h_{n+1} of the chain. Even though the echo of a send from h_i comes from host h_{i+1} from the point of h_i, this packet is actually echoed by the destination host, rather than its directly connected host

is that this packet is echoed by the destination host, rather than its directly connected host. This feature of the TCP/IP protocol motivated us to use the fact that if we compute the time gap between each send and its corresponding echo coming from h_{i+1}, the gaps should vary, but depend on the number of hosts connected after h_{i+1}. The time gaps are increased while more connections are extended along the session.

As shown in Fig. 5, we monitor host h_i, capture all the sends and echoes from the time that the session is only extended to host h_{i+1}, and compute all the gaps between each send and its corresponding echo. We find that those gaps are different, but vary slightly. They are bounded within a certain range which is called a "level," denoted as L_1. If we monitor this host and capture all the sends and echoes continuously, when one more host is connected, we can get level 2, denoted by L_2. In L_2, even though the RTTs are different, on average, they are larger than the RTTs in L_1. When more and more hosts are connected one after another, more and more levels can be obtained. As long as we monitor this session from the beginning to the end continuously and capture all the sends and echoes, we are able to determine the number of the levels. The number of the levels is exactly the same as the length of downstream connection of the chain. If we call each level a step, this method to detect stepping-stone intrusion is called the step-function approach.

In order to obtain the number of RTT steps, a step-counting algorithm was proposed in [8]. There are some other known algorithms proposed to match TCP/IP packets to detect stepping-stone intrusion, such as conservative and the greedy algorithm proposed in [25]. The primary issue of these previous known algorithms is that they always search for a "candidate" echo packet locally, rather than search globally, when they try to match a send packet.

Unlike the policy used in the conservative and the greedy algorithm in [25], the algorithm proposed in [8] is to use data clustering and partitioning technique to match TCP/IP packets and find the RTTs of the packets of a connection chain. All the previous known approaches to match send and echo packets work locally, i.e., these known algorithms process one packet at a time. The data mining algorithm in [8] is a global approach by which it checks all the packets together to determine TCP packet matches. It captures all the send and echo packets of a connection chain in a certain time interval and computes the difference between each send packet and all echo packets received after it. It is for sure that the correct RTTs are among these differences. Based on this observation, the approach is to find the subset that truly represents the RTTs.

It is obvious that these RTTs will cluster around several levels. It uses the maximum-minimum distance clustering algorithm (MMD) to find the real RTTs and determine the number of connections in a chain. The experimental result showed that this algorithm can match TCP/IP packets with both high matching rate and high matching accuracy.

Suppose it monitors and captures the TCP packets of a connection chain from the time the chain is being established to the time that the chain has four connections. At the time when the chain has only one connection, based on the analysis of the distribution of the RTTs of TCP packets in [8], most of its RTTs should be around RTT_1, which is the average value of the RTTs of the chain. Similarly, if the chain is extended incrementally until it has four connections, we have RTTs concentrating around RTT_2, RTT_3, and RTT_4, respectively. This result was first observed in [22]. This RTT clustering at different levels is used to identify the RTTs.

The second observation that will help us with this algorithm is the disjointed partitioning of the RTTs at the different clusters. If we identify the RTTs by the indices of the send packets, we will see that these indices are partitioned into four subsets, one for each cluster. Furthermore, each subset is an interval of the form $[i, i + 1, .., j]$, inclusively. By combining these two observations, clustering and partitioning, it can find the RTTs of a connection chain in a more accurate manner [8].

There have been many data clustering methods proposed in the literature, such as distance function classifiers, minimum distance classifiers, statistical classifiers, fuzzy classifiers, syntactic classifiers, and neural nets classifiers [26–28]. The special properties of the data set coming from send and echo packet timestamps determine that the MMD algorithm is most suitable for the purpose of packet matching. There is no idea about the final clustering results, and there is no a prior knowledge about the number of clusters. The threshold in MMD is only determined if a new cluster is created, rather than if an element is added. Once a cluster center is fixed, it does not shift any more during the process of clustering.

3.4 Stepping-stone detection in SDN

In this section, we introduce the work presented by Bhattacherjee in [29] to apply some of the known SSID and anti-evasion techniques to SDN which use network function virtualization (NFV). NFV is an initiative to virtualize network services traditionally run on proprietary, dedicated hardware. With NFV, functions like routing, load balancing, and firewalls are packaged as virtual machines (VMs) on commodity hardware.

The SDN network architecture has three tiers (see Fig. 6): the application tier, control tier, and infrastructure tier. The control tier consists of the logically

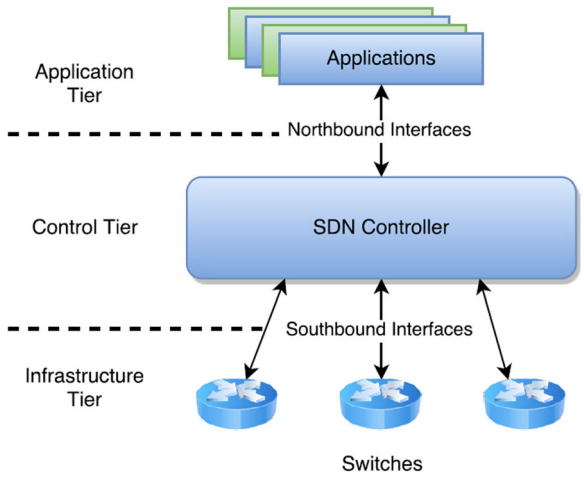

Fig. 6 SDN architecture. The SDN network architecture has three tiers: the application tier, the control tier, and the infrastructure tier

centralized controller, which provides a network-wide view of the forwarding elements (switches) and their states to the application tier via north-bound interfaces (NBI). Distributed routing protocols are replaced in SDN by algorithms that make use of the global view of the network. The centralized control plane is the single point of configuration for the network administrators. The controller in turn manages the forwarding elements. Hence, the traffic can be dynamically shaped by the administrators without configuring the individual forwarding elements. The controller directs the switches to deliver network services wherever they are needed. This process is a move away from traditional network architecture, in which individual network devices make traffic decisions based on their configured routing tables.

All the applications reside in the application tier, including load balancers, monitoring apps, and intrusion detection systems. They use the network-wide view provided by the control tier to monitor and control the data plane. The infrastructure tier (data plane) consists of forwarding elements (switches) which typically forward packets based on layer-2 and layer-3 headers. The controller communicates with the switches using south-bound interfaces (SBI). The SBI is used by the controller to send instructions to the switches and by the switches to consult the controller when they are not able to make the forwarding decision based on the previous instructions.

SDN is an easily programmable network architecture which separates the control plane from the data plane. Naturally, stepping-stone detection mechanisms must be adapted to these new environments on SDNs. The author in [29] first theoretically analyzed the stepping-stone attack techniques and their applicability to SDN and NFV based on the network architectures. He then proposed an architecture to support the detection techniques in SDN and NFV environments. In the experimental part, he implemented the detection techniques and evaluated their effectiveness on various network topologies.

The challenge on SDN is that the commonly used south-bound protocols are not suitable for traffic monitoring. The controller of the SDN can gather flow-level statistics using these protocols but cannot gather useful information on monitored individual connections. Instead, one has to use protocols such as sFlow (see [30]) to gain detailed information about the traffic passing through the individual switches on the SDN. With the traffic monitoring technique sFlow, low-cost sFlow agents are installed in the switches which sample packets and forward sampled data to a data collector for analysis.

More specifically, [29] presented an SDN-based architecture which can be implemented to monitor the data plane for correlated connections and stepping-stones. The sFlow enabled switches sample packets passing through them and forward the header information to a central collection and analysis module. The data analysis module removes redundancy in received information and maps sampled headers to connections. The timing-based stepping-stone detection techniques as well as the anomaly-based chaff detection techniques were implemented within its analyzer module.

4 Open problems

Monitoring an interactive TCP session and sniffing network traffic can be used to detect stepping-stone intrusion. To this end, host-based and network-based approaches have been developed. While we develop more advanced methods to detect stepping-stone intrusion, intruders also develop new techniques to evade detection. Time-jittering and chaff-perturbation are the two most popular techniques used by most intruders in order to evade detection.

Time-jittering is a technique that intruders can hold a packet for a while and then release it. The purpose of using time-jittering is to change the time gap between the TCP/IP packets in a session and further to defeat all the existing time-based SSID (stepping-stone intrusion detection) approaches. Chaff-perturbation is an approach that intruders can inject some meaningless packets into a TCP/IP session to change not only the time gap between the network traffic, but also the amount of packets in a certain time period. Chaff-perturbation technique can easily defeat the stepping-stone intrusion approaches using the amount of network traffic.

In Section 2.4 when we presented SSID algorithms using random-walk approach, we discussed quite a few detection algorithms to handle intruders' time-jittering or chaff-perturbation for detecting evasion in existing research papers in this area. However, all these known detection algorithms to handle intruders' time-jittering

or chaff-perturbation for detecting evasion either are not feasible to implement or do not work effectively. More advanced approaches must to be developed to resist in intruders' time-jittering and/or chaff-perturbation evasion. Any proposed approach that is feasible to implement and works effectively in practical computer networks would be significant.

As we have seen in the discussion, packet matching can more or less resist intruders' evasion. The primary reason is that some meaningless chaffed packets may be filtered out via packet matching. But there is no guarantee that all the chaffed packets can be filtered out. So innovative approaches are urgently needed to be proposed to match packets and resist intruders' chaff evasion effectively in stepping-stone intrusion detection and the issues are pressing.

Another open problem is stepping-stone intrusion upstream detection. The problem is significant because without upstream detection, most methods we discussed in this survey bring false negative errors. For example, estimating the length of a connection chain is a method to detect stepping-stone intrusion effectively and accurately. Unfortunately, the detecting accuracy really depends on where a sensor is located in a connection chain. The more close to the victim a sensor is, the less accuracy of the detection, and the higher false negative errors introduced by detection methods. The main reason is that when we estimate the length of a connection chain, we only count the downstream connection which is from a sensor to the victim. The upstream connection which is from a sensor to the attacker's host is not counted. Only counting both downstream and upstream of a connection can allow detecting approach to obtain an accurate estimation of the length of a whole connection chain.

Another significant part of upstream detection is due to the need to detect stepping-stone intrusion from victims' hosts. As we all know, detecting stepping-stone intrusion from a sensor which is in between intruder's host and victim's host can only protect somebody else other than the sensor itself. Detecting stepping-stone intrusion from the end of a connection chain is an upstream detection issue.

Upstream detection becomes more difficult and complex because of the loose coupling of send and echo packets. The send packets received from the upstream have little relation with the echo packets returned back to an intruder's computer. Unlike downstream detection, computing the length of an upstream connection precisely would be much more difficult. However, upon the observation of user's keystroke behavior which can be modeled as Poisson distribution, we found that, as an interactive process for an intrusion, the intruder needs to think and pause after his/her input to determine what to do next upon the responses. We can explore the relations among the signals: user's behavior (signal 1), user's input (signal 2), and the response (signal 3). Through carefully processing these three signals, it is feasible to identify if an upstream connection is long or short. The third open problem is to apply signal processing technique to stepping-stone intrusion detection. Chaffed packets are hard to be filtered completely and make the existing stepping-stone intrusion detecting approaches not successful. Due to different distributions, chaffed packets can be treated as a signal which is different from regular traffic packets. So a chaffed connection can be considered as a signal which contains regular traffic packets and chaffed packets. The signal is in time domain. If we can use FFT or some other methods to transfer the time domain signal to be a frequency domain signal, it is feasible to use signal processing technique to remove the chaffed packets in frequency domain.

5 Conclusion

In this paper, we provided a survey of research in the area of SSID that includes most of the important known algorithms developed for SSID. We put all these detection methods into two categories: host-based and network-based (i.e., connection-chain based), according to whether multiple hosts in the connection chain are involved in the design of the detection algorithms. In each category of the algorithms, they were discussed in different subsections based on the key techniques used in the algorithm design. For each of the algorithms we included, the key ideas used in the algorithms and differences among the related algorithms were presented. Both advantages and disadvantages of these detection algorithms were also discussed. Finally, several important and challenging open problems were proposed for future research directions in SSID.

Abbreviation

SSID: Stepping-stone intrusion detection

Funding

This work of Drs. Lixin Wang and Jianhua Yang is supported by National Security Agency CDI grant H98230-17-1-0396 with Columbus State University, GA, USA.

Availability of data and materials

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

Authors' contributions

JY worked on Section 2 "Host-based SSID" and proposed the open problems in Section 4. LW worked on Section 1 "Introduction" and Section 3 "Network-based SSID" and was a major contributor in writing the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 7 August 2018 Accepted: 16 November 2018
Published online: 04 December 2018

References

1. S. Staniford-Chen, L.T. Heberlein, *Holding Intruders Accountable on the Internet*, Proc. IEEE Symposium on Security and Privacy, Oakland, CA (1995), pp. 39–49
2. X. Wang, D. Reeves, S. Wu, J. Yuill, *Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework*, Proceedings of the 16th International Information Security Conference (IFIP/Sec'01) (2001), pp. 369–384
3. Y. Zhang, V. Paxson, *Detecting Stepping-Stones*, Proc. of the 9th USENIX Security Symposium, Denver, CO (2000), pp. 67–81
4. T. He, L. Tong, Detecting encrypted stepping-stone connections. *Proceedings of IEEE Transaction on signal processing* **55**(5), 1612–1623 (2007)
5. D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, S. Staniford, in *The 5th International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science*. Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay (2002)
6. S. Wu, U. Manber, G. Myers, W. Miller, An O(NP) sequence comparison algorithm. Inf. Process. Lett. **35**(6), 317–323 (1990)
7. K. Yoda, H. Etoh, *Finding Connection Chain for Tracing Intruders*, Proc. 6th European Symposium on Research in Computer Security, Toulouse, France (2000), pp. 31–42
8. J. Yang, S.S.-H. Huang, Mining TCP/IP packets to detect stepping-stone intrusion. *Journal of Computers and Security* **26**, 479–484 (2007) Elsevier Ltd.
9. K.H. Yung, *Detecting Long Connecting Chains of Interactive Terminal Sessions*, Proc. of International Symposium on Recent Advance in Intrusion Detection (RAID), Zurich, Switzerland (2002), pp. 1–16
10. A. Blum, D. Song, And S. Venkataraman, Detection of Interactive Stepping-Stones: Algorithms and Confidence Bounds, Proceedings of International Symposium on Recent Advance in Intrusion Detection (RAID), Sophia Antipolis, France, 20–35, 2004
11. J. Yang, B. Lee, S.S.–.H. Huang, *Monitoring Network Traffic to Detect Stepping-Stone Intrusion*, the Proceedings of 22nd IEEE International Conference on Advanced Information Networking and Applications (AINA 2008), Okinawa, Japan (2008), pp. 56–61
12. J. Yang, Y. Zhang, *RTT-Based Random Walk Approach to Detect Stepping-Stone Intrusion*, IEEE 29th International Conference on Advanced Information Networking and Applications (2015), pp. 558–563
13. Teknomo, Kardi. (2017) Stochastic Process Tutorial. http://people.revoledu.com/kardi/tutorial/StochasticProcess/RandomWalk/RandomWalk.html
14. S.S.–.H. Huang, R. Lychev, J. Yang, *Stepping-Stone Detection via Request-Response Traffic Analysis*, To Be Published in Lecture Notes in Computer Science (LNCS) by Spring-Verlag, 4th IEEE International Conference on Automatic and Trusted Computing, Hong Kong, China (2007), pp. 276–285
15. T. He and L. Tong, "Detecting Stepping-Stone Traffic in Chaff: Fundamental Limits and Robust Algorithms", 9th International Symposium On Recent Advances In Intrusion Detection (RAID 2006), 2006
16. W. Ding, M. J. Hausknecht, S.-H. S. Huang, and Z. Riggle, "Detecting Stepping-Stone Intruders with Long Connection Chains", 2009 Fifth International Conference on Information Assurance and Security, 2009
17. S. S.-H Huang, H. Zhang, and M. Phay, "Detecting Stepping-Stone Intruders by Identifying Crossover Packets in SSH Connections ", the Proceedings of 30th IEEE International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, IEEE proceedings and Digital Library, pp. 1043–1050, 2016
18. X. Wang, D. Reeves, Robust correlation of encrypted attack traffic through stepping stones by flow watermarking. IEEE Trans Dependable Secure Comput **8**(3), 434–449 (2011)
19. Y. Chen, S. Wang, in *Proceedings of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE), WorldComp*. A novel network flow watermark embedding model for efficient detection of stepping-stone intrusion based on entropy (2016)
20. J. Yang, Y. Zhang, R. King, T. Tolbert, in *2018 32nd IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. Sniffing and chaffing network traffic in stepping-stone intrusion detection (2018), pp. 515–520
21. H.T. Jung et al, "Caller Identification System in the Internet Environment", Proceedings of the 4th Usenix Security Symposium, 1993
22. J. Yang, S.-H.S. Huang, *A Real-Time Algorithm to Detect Long Connection Chains of Interactive Terminal Sessions*, Proceedings of 3rd ACM International Conference on Information Security (Infosecu'04), Shanghai, China (2004), pp. 198–203
23. S. Snapp et al., "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype". In Proc. 14th National Computer Security Conference, 1991
24. V. Paxson, S. Floyd, Wide-area traffic: the failure of Poisson modeling. IEEE/ACM Trans. Networking **3**, 226–244 (1995)
25. J. Yang, S.–.H.S. Huang, *Matching TCP Packets and Its Application to the Detection of Long Connection Chains*, Proceedings of 19th IEEE International Conference on Advanced Information Networking and Applications (AINA 2005), Taipei, Taiwan, China (2005), pp. 1005–1010
26. M. Friedman, A. Kandel, *Introduction to Pattern Recognition: Statistical, Structural, Neural, and Fuzzy Logic Approaches* (NJ World Scientific Publishing Co., River Edge, London, 1999)
27. A. Jain, R. Dubes, *Algorithms for Clustering Data* (Prentice Hall, Inc., Englewood Cliffs, 1988)
28. B. Mirkin, *Mathematical Classification and Clustering* (Kluwer Academic Publishers, Dordrecht, 1996)
29. Bhattacherjee, Debopam. "Stepping Stone Detection for Tracing Attack Sources in Software-Defined Networks", Degree Project in Electrical Engineering, Stockholm, Sweden (2016)
30. Phaal, P., Panchen, S., and McKee, N., "InMon Corporation's sFlow: A method for monitoring traffic in switched and routed networks". RFC 3176, IETF, 2001