Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 1807509, 8 pages
https://doi.org/10.1155/2021/1807509

Research Article

A Framework to Test Resistency of Detection Algorithms for Stepping-Stone Intrusion on Time-Jittering Manipulation

Lixin Wang,[1] **Jianhua Yang**,[1] **Michael Workman**,[1] **and Peng-Jun Wan**[2]

[1]*TSYS School of Computer Science, Columbus State University, Columbus GA, USA*
[2]*Department of Computer Science, Illinois Institute of Technology, Chicago IL, USA*

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

Received 25 June 2021; Accepted 27 July 2021; Published 10 August 2021

Academic Editor: Zhuojun Duan

Hackers on the Internet usually send attacking packets using compromised hosts, called stepping-stones, in order to avoid being detected and caught. With stepping-stone attacks, an intruder remotely logins these stepping-stones using programs like SSH or telnet, uses a chain of Internet hosts as relay machines, and then sends the attacking packets. A great number of detection approaches have been developed for stepping-stone intrusion (SSI) in the literature. Many of these existing detection methods worked effectively only when session manipulation by intruders is not present. When the session is manipulated by attackers, there are few known effective detection methods for SSI. It is important to know whether a detection algorithm for SSI is resistant on session manipulation by attackers. For session manipulation with chaff perturbation, software tools such as Scapy can be used to inject meaningless packets into a data stream. However, to the best of our knowledge, there are no existing effective tools or efficient algorithms to produce time-jittered network traffic that can be used to test whether an SSI detection method is resistant on intruders' time-jittering manipulation. In this paper, we propose a framework to test resistency of detection algorithms for SSI on time-jittering manipulation. Our proposed framework can be used to test whether an existing or new SSI detection method is resistant on session manipulation by intruders with time-jittering.

1. Introduction

Hackers on the Internet usually send attacking packets using compromised hosts, called stepping-stones, in order to avoid being detected and caught. With stepping-stone attacks, an intruder remotely logins these stepping-stones using programs like SSH or telnet, uses a chain of Internet hosts as relay machines, and then sends the attacking packets. To launch a stepping-stone attack, the intruder enters the attacking commands on his/her local machine which are relayed through the stepping-stone machines until the attacking packets arrive at the final target machine. It is well-known that every such TCP session between a server and a client is independent of one another even if they are relayed sessions. Such a nature of the TCP protocol makes it much more challenging to know the attacker's geographical location while accessing a remote machine via multiple relayed TCP sessions. The final target machine can only see the TCP packets from the last hub of the connection chain. Therefore, a target machine can hardly learn any information regarding the origin of the intrusion.

To launch a stepping-stone attack, the intruder could use a remote login program (SSH, telnet, or rlogin) and create a connection chain as shown in Figure 1. In this figure, Host 0 is the intruder's machine, Host N is the final target host, and Host 1, Host 2, $\cdots$, and Host $N-1$ are the stepping-stone machines. With stepping-stone intrusion detection (SSID), the detection program can be installed at any of the stepping-stones. The stepping-stone host with the detection program installed is called a detecting sensor. In Figure 1, we assume that Host i is the (detecting) sensor. The purpose of SSID is to know if the detecting sensor Host i is employed as a stepping-stone machine. Two important concepts related to a detecting sensor of a connection chain are the incoming and outgoing connections. 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 the detecting sensor Host

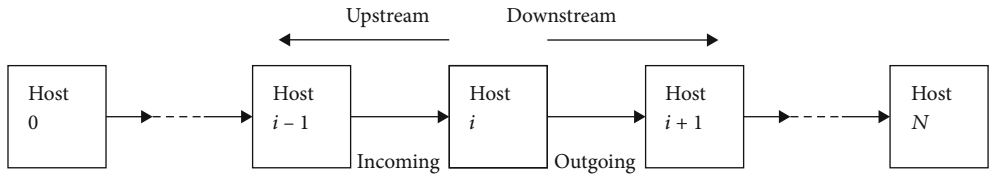

Figure 1: A sample connection chain.

i is employed as a stepping-stone machine, then, there exists at least one matched pair between all of its incoming connections and all of its outgoing connections.

1.1. Definitions of Send/Echo Packets. The definitions of Send and Echo packets will be illustrated using Figure 1. Assuming that Host i is the detecting sensor. First, let us look at its incoming connection. Send packets are those TCP packets with the flag bit TCP.Flag.PSH set to TRUE that are sent from Host $i-1$ to Host i; Echo packets are those TCP packets with the flag bit TCP.Flag.PSH set to TRUE that are sent from Host back to Host $i-1$. Now, let us look at the outgoing connection. Send packets of the outgoing connection from Host i are those TCP packets with the flag bit TCP.Flag.PSH set to TRUE that are sent from Host i to Host $i+1$; Echo packets are those TCP packets with the flag bit TCP.Flag.PSH set to TRUE that are sent from Host $i+1$ back to Host i.

Which Send packet is matched with which Echo packet? Let us answer this question by using an example on the command line. If an attacker enters the command "ps" on a command line in a terminal, the command could be sent to the target machine with one or two TCP packets. For simplicity, we assume that the command "ps" is delivered to the target host with two different TCP packets, one for "p" and the other one for "s." When the attacker types "p" on the command line, its packet is delivered to the target host. After this Send packet is echoed, an Echo packet is sent back to the attacker's machine, and then the letter "p" is visible on the screen of the attacker's host. The Send packet associated with the command "p" and its Echo packet are referred to as a *matched pair*, or sometimes called a *relayed pair*. Based on the TCP protocol design, an Echo packet may echo more than one Send packets. Similarly, a Send packet may be echoed by more than one Echo packets.

1.2. The Distribution of Packets' RTTs for a Connection Chain. In a TCP connection, a packet RTT is the sum of four delays including queuing delay, transmission delay, processing delay, and propagation delay. For connection chain-based SSI detection, packet RTTs can be used to estimate a connection chain length. The network traffic can be represented by the RTTs calculated from the matched pairs of a Send packet and an Echo packet. In the work [1] by Yang et al., the authors proved that a connection chain length is the same as the number of clusters that are generated by employing the RTTs calculated from the connection chain.

The work [2] by Paxson and Floyd discovered that the packet RTTs calculated from a connection chain follow the Poisson distribution. This important discovery can be employed to match TCP packets as well as calculate a connection chain length. Figure 2 shows that the packet RTTs obtained from a connection chain follow the Poisson distribution. In this figure, the RTTs were obtained from the TCP packets collected from a connection chain whose length is four. Based on this experiment in a connection chain with four connections, most RTT values are very close to the average μ which is 138,500 (microsecond) of all the RTT values. Clearly, at least 95% of the RTTs are larger than 137,000 (microsecond) as well as less than 141,000 (microsecond).

If a random variable X obeys the Poisson distribution, its mean and standard deviation are represented by μ and σ, respectively. It is well-known that

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

According to the above inequality, the majority values of the random variable X should be around its mean value μ. The absolute value of the difference between X and μ is at most 2σ. Therefore, the packet RTT values calculated from captured network traffic from a TCP connection chain follow the Poisson distribution. That is, most values of the packet RTTs calculated from a connection chain of fixed length must be close to its mean value which is inside a circle centered at X of radius 2σ.

1.3. Session Manipulation by Intruders Using Chaff-Perturbation or Time-Jittering. A great number of detection approaches for SSI have been developed in the literature [2–12]. However, malicious attackers never stop developing new session manipulation approaches to evade detection. The two most popular such techniques used by attackers are time-jittering and chaff perturbation. Time-jittering is a method that an attacker's host does not transmit packets immediately. Instead, every packet will be hold for a random period of time, and then it will be released. The timestamp of each packet will be jittered. Therefore, if the network traffic is manipulated by intruders using the time-jittering technique, the timestamp of every packet is modified. As a result, all the existing approaches for time-based SSID do not work anymore.

Chaff perturbation is a session manipulation approach with which attackers can create some meaningless packets and then insert them into a normal network traffic. Due to the injection of these meaningless packets into a normal network traffic, the total number of packets is changed, so are the time gaps between the normal packets. Therefore, if the network traffic is manipulated by intruders using chaff perturbation technique, the total number of packets and the time gaps of packets are all modified. As a result, those existing approaches (SSID) that are based on the amount

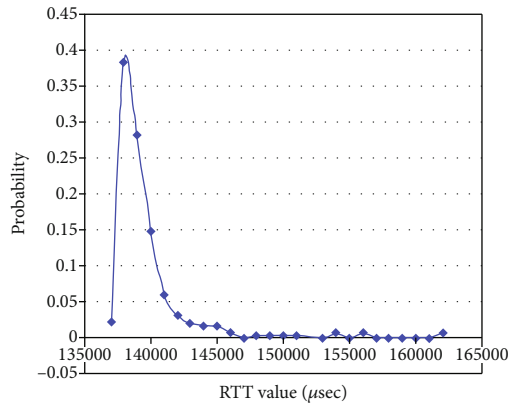

Figure 2: The distribution of packets' RTTs for a connection chain.

of network traffic or time gaps of packets do not work anymore. Therefore, it is very important to know whether an existing or new SSID method is resistant to intruders' session manipulation.

For chaff perturbation, software tools such as Scapy have been developed to inject meaningless packets into data streams. These software tools can be used to test whether a SSID algorithm is effective in resisting intruders' session manipulation with chaff perturbation. Scapy is a software for packet manipulation in computer networks. Its first version was implemented in Python. It can create or decode packets and then send them to the Internet. It can also capture the packets and match an Echo packet with its corresponding Send packet. Moreover, scanning, tracerouting, attacks, and network probing and discovery can all be done using Scapy.

However, to the best of our knowledge, there are no existing software tools or effective algorithms to test resistency of SSID algorithms on session manipulation with time-jittering. In this paper, we propose a framework to test resistency of detection approaches for SSI on time-jittering manipulation. Our proposed framework can be used to test whether an existing or new SSID method is resistant on session manipulation by intruders with time-jittering. The output file generated by our proposed algorithm satisfies the following properties: (1) it remains a valid list of captured TCP packets as for each Send packet with jittered timestamp, and its timestamp is still less than that of every following Echo packet; (2) only the timestamps of a given percentage of the Send packets will be jittered; (3) those Send packets whose timestamps will be jittered are randomly selected; and (4) for every Send packet whose timestamp will be jittered, the increment of its timestamp is a random and independent value.

Table 1 lists all the notations used in this paper to help readers for easy referencing.

The remaining of this paper is organized as follows. In Section 2, we give literature review on many existing and significant SSID methods. In Section 3, we present a framework to test resistency of detection algorithms for SSI on time-jittering manipulation by intruders. This paper will be summarized in Section 4, and the funding information of this research work will be provided in Acknowledgments.

2. Literature Review on SSID

Network security experts and researchers have proposed many SSID approaches since Staniford-Chen and Heberlein published their seminar work in SSID in 1995 [7]. In this section, we will conduct a literature review on SSID methods since 1995. There are two different types of SSID approaches: host-based SSID and network-based SSID. A host-based SSID approach is to detect stepping-stone intrusion by comparing all the outgoing connections with all the incoming connections of a single host (that is, the detecting sensor) to see if there exists a matched pair in these two connections. A network-based SSID is to estimate the length of the connection chain (the number of connections in the connection chain).

Let us begin with reviews on host-based SSID. Staniford-Chen and Heberlein [7] proposed a content-thumbprint method to find a matched pair on a single machine by comparing all the outgoing connections with all the incoming connections of the host. As law content of packets is used for the comparison, this content-thumbprint method does not work if the network traffic is encrypted. In order to overcome the problem of this content-thumbprint method, Zhang and Paxson [12] proposed a time-thumbprint method for SSID. As the timestamps of packets are usually not encrypted, this time-thumbprint method could work effectively if the network traffic is encrypted. If intruders send encrypted attacking packets to launch attacks by using SSH, for example, it makes the detection process of SSID much more challenging. Furthermore, if intruders use session manipulation techniques such as chaff perturbation and/or time-jittering to evade detection, it will make the SSID process even more challenging.

There are quite a few SSID methods have been proposed for network traffic with session manipulation by attackers using chaff perturbation and/or time-jittering to evade detection. He and Tong [9] proposed the packet counting approach aimed at addressing such challenges caused by intruders' session manipulation. The authors of [9] assumed that network traffic is encrypted, and the session is manipulated by intruders using the time-jittering and chaff perturbation techniques. Furthermore, an attacker can inject chaff packets into an attacking stream. This paper developed two detection algorithms for SSID that deal with the time-jittering and chaff perturbation manipulation. Donoho et al. [4] employed a gateway router as the detecting sensor and proposed detection algorithms for SSID. This paper considered that a stepping-stone host maybe a single machine on the Internet or a whole network associated with the gateway router.

As we mentioned earlier, Zhang and Paxson [12] proposed detection algorithms for SSID that work for encrypted network traffic. However, those approached proposed in [12] have problems when the session is manipulated by attackers using time-jittering and chaff perturbation. The detection algorithms for SSID developed in [12] depend on accurate timestamps of network packets; otherwise, these SSID approaches do not work effectively, even if packets' timestamps are slightly jittering. Yoda and Etoh [13] proposed a better approach to address this issue. This method for SSID

Table 1: All notations used in this paper.

X	A random variable
μ	Mean of a random variable
σ	Standard derivation
p	Percentage of which timestamps of Send packets will be jittered
N	Total number of Send packets in the input file
M	Largest integer less than or equal to pN
l_packets	A list of all the packets in the input file
l_Send	A list of all Send packets in the input file
l_random	An increasing list of M random numbers in the range $0 \sim N-1$

is called a deviation-based approach as the deviations between an existing intruder stream and all other concurrent data streams in the network are computed. This method tries to discover a set of data streams that could match the stream sent from the intruder. He and Tong [9] proposed a better way to resist attackers' session manipulation using chaff perturbation. The detection algorithms work effectively when the network traffic having meaningless packets chaffed into the data stream. These algorithms for SSID still work if the number of chaffed packets is proportional to the total number of packets sent from the attacker's machine. Yang et al. [14] used random walks to design a method for SSID to handle intruders' manipulation using chaff perturbation evasion. In this paper, the difference between the number of responses and the number of requests is modelled as a random walk. Yang and Zhang [15] proposed a better way of using random walks to design detection algorithms for SSID. The method used in [15] is referred to as an RTT-based random walk approach. The key idea of this paper is to decide if an outgoing connection and an incoming connection are a matched pair by applying the number of RTTs in a connection as well as the idea of random walks. The detection algorithms proposed in [15] work effectively when the network traffic is manipulated by attackers using time-jittering and/or chaff perturbation evasion.

Ding et al. [16] took a different approach that detected SSI at the target victim machine. This paper used and considered the time delay between the attacker completing typing an attacking command and the time when the next letter is entered. Later, Huang et al. [17] improved the detection algorithm for SSID proposed in [16]. The authors in [16, 17] assumed that cross-over packets must be present in a long connection chain. Huang et al. [17] discovered that a longer connection chain should produce more cross-over packets. Wang and Reeves [18] proposed a watermark-based method for SSID. This paper assumed that a unique watermark is injected into the network traffic. The matching between an incoming connection and an outgoing connection is based on the injected watermark.

Yang et al. [6] developed a computer program to inject TCP/IP packets into network traffic. The program developed in this paper could help network security researchers better understand how session manipulation works and design more innovative detection algorithms for SSID that are resistant to time-jittering and/or chaff perturbation manipulations by intruders.

Because stepping-stones may be employed by legal applications for remote access, host-based SSID approaches could produce high false positive errors. To avoid the problem caused by host-based detection methods, network-based detection approaches were proposed. This type of detection method for SSID is to calculate the length of a connection chain. It is well-known that most hosts access a remote server using at most three stepping-stones. If a host uses more than three stepping-stones to access a remote server, it is most likely an intrusion. This is the rationale of all network-based detection approaches.

Next, we present the literature review on network-based detection approaches for SSI. The first known detection algorithm via the network-based approach was presented in [19] in 2002. The key idea of this paper is to compute the RTT of a Send packet and then attempt to match this Send packet with its corresponding acknowledgment (ACK) packet transmitted from the next adjacent host in the connection chain. The method proposed in [18] reduced the false positive error a little bit. However, this method for SSID produces high false negative error as the ACK packet from the next adjacent host instead of the actual Echo packet was used for the matching. The problem with the work presented in [19] was that the way to set up the connection chain was not proper.

To overcome the problem caused by the improper connection chain setup in the paper [19], a step-function detection method was developed to calculate the length of a connection chain in [20] in 2004. The step-function method developed in this paper reduced both the false positive and false negative errors in the case of local area networks (LANs). The connection chain was properly created in [20] so that the corresponding Echo packet of a Send packet can be used for the matching. In this paper, a Send packet was matched with its corresponding Echo packet, and then the packet RTTs was calculated using the step-functions. The drawback of this detection approach presented in [20] is that this method works effectively only in LANs, but it does not work well in the context of the Internet. The conservative and greedy packet matching method for SSI detection presented in [21] worked effectively in the context of the Internet, but this detection method can only match very

few TCP packets, and thus, it is not practical in SSID for computer networks connected with the Internet.

The data mining approach with clustering and partitioning proposed in [5] is a very effective connection chain-based detection approach SSI. In this work, the packet RTTs are obtained by applying the maximum-minimum distance (MMD) clustering method, and all the possible packets were checked during packet matching. The clusters' number outputted by the MMD algorithm gives the length of the connection chain. Also, the detection method based on MMD reduced largely the false negative errors as well as the false positive. A drawback of this detection algorithm for SSI is that a large number of TCP packets must be captured and analyzed. Therefore, the detection method based on MMD presented in [5] is not efficient in terms of processing time. A SSID method using the k-means clustering approach was developed in [22] in order to overcome the weakness of the MMD-based SSID algorithm proposed in [5]. This k-means-based detection algorithm is very efficient as it did not require to capture and analyze a large number of TCP packets. It is well-known that packet RTTs cluster around a number of levels [5, 20]. In general, the k-means data mining algorithm has been widely used to put data-set items into groups of related observations without none of the prior knowledge regarding their relationships. As long as most of the RTT outliers are removed from the captured RTTs in the input file, the k-means-based SSID algorithm proposed [22] works effectively in LANs.

It is worth mentioning some recent significant results that are related to network security. Using a combination of social relationship and nonsensitive attributes, Cai et al. [23] investigated how social networks are exploited and an inference attack is launched. With differential privacy, Cai et al. [24] proposed a mechanism that employed a sampling approach to generate rough counting results. In theory, these counting results are verified to satisfy privacy guarantee as well as unbiasedness. An innovative method to upload data in smart cyber-physical systems was proposed in [25]. The method proposed in this paper considered privacy preservation as well as energy conservation. A framework to mimic the behaviors of stepping-stones was proposed in [3]. The proposed framework in [3] contains tools for evasion and some other tools that can be used for evaluating detection rates of existing SSID approaches. With industrial Internet of Things, a privacy-preserved data sharing scheme was proposed in [26] where competing customers can coexist in different stages of the IoT system. Gamarra et al. [27] developed a model that describes the propagation of SSI attacks in the IoT systems using a vulnerability graph whose topology is fixed as well as switching. The model can be expanded to a more realistic scenario when the vulnerability graph changes because the attack is discovered or the intrusion detection system of the IoT is trigged. Liu et al. [28] proposed an adaptive intrusion detection approach using the fuzzy rough set theory and a new pattern learning. Using a greedy approach, the authors of [28] introduced a Gaussian mixture model clustering method aiming at obtaining the intrinsic structure of instances of computer networks.

3. A Framework to Test Resistency of SSID Methods on Time-Jittering Manipulation

In this section, we first propose a framework to test resistency of detection algorithms for SSI on time-jittering manipulation. Our proposed framework can be used to test whether an existing or new SSID method is resistant on session manipulation by intruders with time-jittering. Then, we present the properties of the output generated by our proposed algorithm with jittered timestamps. Finally, the significance of our proposed framework is discussed.

3.1. An Algorithm to Test Resistency of SSID Methods on Time-Jittering. The algorithm for testing resistency of SSID methods on time-jittering manipulation is described in Algorithm 1.

Next, we explain the above Algorithm 1 for testing resistency of SSID methods on time-jittering manipulation.

Both the input file input.txt and the output file output.txt contain two columns: one lists packet timestamp and the other column lists the packet type for each packet. The input file is obtained from a PCAP file captured in the Internet environment. The output file contains jittered timestamps in the first column and the same packet type in the second column as in the input file, and the only timestamps of a given percentage of the Send packets will be jittered.

The algorithm begins with copying the content of input.txt into output.txt. Let p denote the percentage with which of Send packets' time stamps will be jittered, N the total number of Send packets in the file input.txt, and M the largest integer less than or equal to pN. Clearly, M is the number of Send packets that will be jittered.

Then, we generate M random numbers in the range $0 \sim N-1$, sort them in an increasing order, and store them in a list l_random. These random numbers are the indices of the Send packets in the list l_Send whose timestamps will be jittered, where l_Send represents the list of all the Send packets in the file input.txt.

After that, the for loop iterates each Send packet in the list l_Send. For each Send packet in l_Send, if its index belongs to the list l_random, then its timestamp will be jittered. The increment will be a random value between zero and diff, where diff represents the timestamp difference between this Send and the first following Echo packet in the list l_packets. Finally, update this Send's timestamp in the file output.txt by adding the increment to its original timestamp.

3.2. Properties of the Output Generated by the Above Algorithm 1 with Jittered Timestamps. Clearly, the output file output.txt generated by the above Algorithm 1 satisfies the following important properties:

(1) It remains a valid list of captured TCP packets as for each Send packet with jittered timestamp, its timestamp is still less than that of every following Echo packet

(2) Only the timestamps of a given percentage of the Send packets will be jittered

Input: a TXT file input.txt with two columns (including packet timestamps and the packet type) obtained from the packets captured in the Internet environment

Output: a TXT file output.txt with two columns (including packet timestamps and the packet type) and the timestamps of a given percentage of Send packets have been jittered

```
copy the file input.txt into the file output.txt
p = percentage; /* timestamps of this percentage of Send packets will be jittered */
N = total number of Send packets in the file input.txt;
M = largest integer less than or equal to pN;
/* number of Send packets that will be jittered */
l_packets = a list of all the packets in the file input.txt
l_Send = a list of all Send packets in the file input.txt
l_random = an increasing list of M random numbers in the range 0~N-1
/* the Send packets in the list l_Send with these indices in l_random will be jittered */
for each Send packet in the list l_Send, do
    if its index equals a number in the list l_random
        /* jitter its timestamp */
        diff = timestamp difference between this Send and the first following Echo packet in the list l_packets
        incr = a random number in the range 0~diff
        increase the timestamp of this Send packet by incr
        update this Send's timestamp in the file output.txt
```

Algorithm 1: An efficient algorithm for testing resistency of SSID methods on time-jittering.

(3) Those Send packets whose timestamps will be jittered are randomly selected

(4) For every Send packet whose timestamp will be jittered, the increment of its timestamp is a random and independent value

3.3. Significance of the Proposed Framework. Stepping-stones have been widely used by hackers to launch their attacks, especially after the emerging of the Internet. Network security researchers have been proposing approaches for SSID during the last two decades since Staniford-Chen and Heberlein published their seminar work [7] in 1995. However, intruders have also been developing new techniques to evade our detection. When SSI attacks are launched, intruders tend to use session manipulation techniques to evade detection. By far, the most two popular session manipulation techniques used by intruders for evasion are time-jittering and chaff perturbation. All the known SSID methods to handle intruders' time-jittering and/or chaff perturbation for detecting intruder's evasion either are not feasible to implement or do not work effectively. Some of such methods can only detect an intruder's evasion with very limited capacity. Therefore, newly proposed SSID methods should be resistant to intruders' session manipulation so that they can be used to protect practical computer networks against SSI attacks.

For chaff perturbation, software tools such as Scapy have been developed to inject meaningless packets into data streams. These software tools can be used to test whether a SSID algorithm is effective in resisting intruders' session manipulation with chaff perturbation. Currently, there are no existing software tools or effective algorithms to test resistency of SSID algorithms on session manipulation with time-jittering. Our proposed framework in this paper can be used by network security researchers to test whether their proposed SSID algorithms are resistant on session manipulation by intruders with time-jittering.

4. Conclusion

In this paper, we developed a framework to test resistency of detection approaches for SSI on time-jittering manipulation by intruders. Network security researchers have been proposing approaches for SSID during the last two decades. However, intruders have also been developing new techniques to evade our detection. When SSI attacks are launched, intruders tend to use session manipulation techniques to evade detection. Therefore, newly proposed SSID methods should be resistant to intruders' session manipulation so that they can be used to protect practical computer networks against SSI attacks. Currently, there are no existing software tools or effective algorithms to test resistency of SSID algorithms on session manipulation with time-jittering. Our proposed framework in this paper can be used by network security researchers to test whether their proposed SSID algorithms are resistant on session manipulation by intruders with time-jittering.

As a future research direction, we will develop new effective methods for SSID that are efficient and resistant to intruders' evasion manipulation using time-jittering and/or chaff perturbation. Our proposed framework can be used to verify the resistency of the proposed SSID methods on time-jittering manipulation by intruders.

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 (NSA) NCAE-C research grant H98230-20-1-0293 with Columbus State University, Columbus GA, USA.

References

[1] 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, Austria, 2006.

[2] 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.

[3] H. Clausen, M. S. Gibson, and D. Aspinall, "Evading stepping-stone detection with enough chaff," in *International Conference on Network and System Security*, pp. 431–446, Cham, 2020.

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

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

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

[7] 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, 1995.

[8] T. He and L. Tong, "Detecting stepping-stone traffic in chaff: fundamental limits and robust algorithms," in *9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, Hamburg, Germany, April 2006.

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

[10] 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.

[11] L. Wang and J. Yang, "A research survey in stepping-stone intrusion detection," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, Article ID 276, 2018.

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

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

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

[15] J. Yang and Y. Zhang, "RTT-based random walk approach to detect stepping-stone intrusion," in *IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 558–563, Gwangju, Korea (South), 2015.

[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*, Xi'an, China, August 2009.

[17] S. S. H. Huang, H. Zhang, and M. Phay, "Detecting stepping-stone intruders by identifying crossover packets in SSH connections," in *Proceedings of 30th IEEE International Conference on Advanced Information Networking and Applications*, pp. 1043–1050, Crans-Montana, Switzerland, March 2016.

[18] 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.

[19] K. H. Yung, "Detecting long connecting chains of interactive terminal sessions," in *Proc. 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] L. Wang, J. Yang, X. Xu, and P. J. Wan, "Mining network traffic with the -means clustering algorithm for stepping-stone intrusion detection," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6632671, 9 pages, 2021.

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

[24] 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.

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

[26] 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.

[27] M. Gamarra, S. Shetty, O. Gonzalez, D. M. Nicol, C. A. Kamhoua, and L. L. Njilla, "Analysis of stepping-stone attacks in internet of things using dynamic vulnerability graphs," *Modeling and Design of Secure Internet of Things*, vol. 12, pp. 273–294, 2020.

[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, p. 112845, 2020.