Detecting Stepping-Stone Intruders by Identifying Crossover Packets in SSH Connections

Shou-Hsuan Stephen Huang, Hongyang Zhang, Michael Phay

Department of Computer Science
University of Houston
Houston, TX, USA
shuang@cs.uh.edu

Abstract— Routing packet traffic through a chain of hosts is a common technique for hackers to attack a victim server without exposing themselves. Generally, the use of a long connection chain to log in to a computer system is an indication of the presence of an intruder. This paper presents a new solution to the problem of detecting such long connection chains at the server side. Our hypothesis is that a long connection chain will cause Request and Response packets to cross each other along the chain. So even though we cannot directly observe the packet crossovers from the server side, we can observe some of their side effects. Thus, our detection algorithm is based on detecting this side effect of packet crossovers. We validated the algorithm using test data generated on the Internet. The results show a high detection rate of long connection chains of length three hops with a reasonable false positive rate.

Keywords— intrusion detection; stepping-stone attacks; packet crossovers; long connection chain

I. INTRODUCTION

Cyber security has always been an important issue for the military and homeland security. One the consumer side, cyber attacks against businesses reached a new peak in 2014. Internet security is required in order to prevent hackers from stealing information or damaging government, corporation, and personal computers. Even though many measures have been deployed to prevent hackers from computer server intrusion, there are news reports about stolen data affecting millions almost weekly. For example, servers at LinkedIn were compromised and 6.5 million user passwords were stolen in 2012 [1]. In 2011, Sony PlayStation suffered from 24 days of network outage due to intrusions, which cost Sony a $171 million loss. The attack continued for 3 days, and Sony had to turn off the PlayStation Network for maintenance [2] [3]. More recent data losses include Home Depot, Sony Entertainment, Target, and other US retailers.

There are many different ways of attacking a computer on the Internet. Since network communication is divided into a 7-layer OSI model [17], each layer is vulnerable to attack. This research project focuses on the application layer, where the Secure Shell (SSH) protocol resides. A number of attacks can happen at this level, such as Denial of Service (DoS), SQL Injection, and Man-in-the-Middle attacks. Our goal is to detect hackers who try to tamper with a system after logging in to the system via an SSH server. We also assume the hackers will not spoof the IP address, because they need to receive information from the targeted server. In order to avoid detection, most intruders route through a long chain of previously compromised intermediate hosts, called stepping-stones, before logging in to a target server.

Stepping-stone intrusion is widely used by hackers to stay anonymous and to avoid trace backs. These stepping-stone hosts are usually picked by intruders and may be located in different countries. Therefore, it would be extremely difficult to trace back to the original attacker without the assistance of system administrators from all these intermediate hosts in real-time.

We can break up such attacks by identifying them anywhere along this chain of stepping-stone hosts. One such place is at a stepping-stone while the hackers are connected [4]. There have been many publications focusing on this area in the past two decades. Staniford-Chen and Heberlein [5] wrote the first paper addressing this issue. Without bothering the intermediate hosts, the paper proposed a method that helps to flag suspicious activities, maintains logs in the case of an intrusion from local sites and sets up connections through external nodes. Their approach also helps to enforce policies in terms of cross-traffic, and to detect insecure combinations of legitimate connections.

Previous research primarily concentrated on intermediate host-based stepping-stone detections, i. e., detecting stepping-stones by correlating streams of packet traffic. From the view of an intermediate host, the data generated by an attacker are sent downstream, and a response from the server (target host) is passed backward upstream to the attacker. The request/response packets form a closed loop between an intermediate host and the target server. Meanwhile, the time difference between the request and the response is captured for detection. Yang and Huang [6] used time gaps in the closed loop to detect a downstream connection chain in real time. They aimed to stand between the attacker and victim, in order toprotect the target host; this is usually a third party connected by a long chain originating from a hacker. However, most of the time we do not have privilege on intermediate hosts in a long chain. It would be extremely helpful to be able to detect intruders and avoid being compromised at the target host. There is no straightforward method that is able to measure the full Round Trip Time (RTT) from a target to a hacker. Since

SSH is an interactive terminal session, the client's machine will not automatically send a reply back to the server for anyone to compute the RTT. The responses that the server received are generated from the nearest node on the connection chain.

Li, Zhou and Wang [18] devised a novel real-time RTT calculation algorithm for stepping-stone attacks based on the estimation of the current RTT value. The authors show that this is more precise than the previous real-time RTT algorithms.

Ding et al. [7] proposed an intrusion detection algorithm at the targeted server, which is at the end of the connection chain shown in Figure 1 above. As presented in Figure 1, the neighboring hosts directly connected to the target server are the only machines visible to the victim (using IP addresses on the packets). The rest of the hosts in the chain are beyond the visibility of the target server. Ding et al.'s goal was to detect and prevent suspicious stepping-stone connection chains by using an anomaly-based algorithm. They first built a profile of short connection chains using inter-packet gaps. If an SSH connection deviates too much from this normal profile, it is considered a long chain and is thus suspicious.

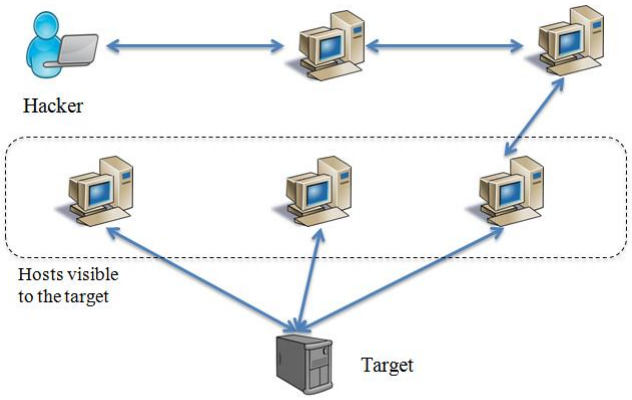

Figure 1: Logging in to a target server using a long connection chain

Ding et al.'s approach is to compute the difference in distribution of the packet gaps. Their hypothesis is that the longer the connection chain is, the larger the gaps are. In their experiments to validate their hypothesis, 4-hop and 6-hop connection chains were compared to short connection chains of 1-hop. The results show that the algorithm works, as 86% of long connection chains were identified, with a false positive rate of around 13% in the case of 6-hop chains. The accuracy for the 4-hop chains is less impressive.

The goal of this project is to improve the accuracy rate of detecting long connection chains based on such previous work [7]. To their credit, the authors discovered the "packet crossovers" in long connection chains, even though they did not find a way to use that observation to detect long connection chains. The authors also suggested that longer connection chains should cause more packet crossovers. Our paper makes the same assumption of the existence of these packet crossovers and uses this assumption to identify long connections. Thus, building on previous work, we are able to detect network intruders with better precision.

The rest of this paper is organized into four sections. Section 2 gives definitions of network security, intrusion detection, stepping-stones, and presents other related work. Section 3 presents the issue of packet crossover and its relationship with the length of the connection chain. Section 4 shows our new detection algorithm to identify long connection chains from short ones. The final section summarizes our work on stepping-stones detection.

II. Literature Survey and Definitions

In this section, we briefly review current research in stepping-stone intrusion detection.

2.1 Definitions and Assumptions

Network security has been a research topic for decades [7][8][9][10]. Intrusion detection provides a network, as well as resources that are stored in the network, accessible device protection from intruders attemping to steal information. Authentication can be deployed as the first step in the defense of network security. Username and password can be to authorize users into the system. There are also other authentication mechanisms such as tokens, fingerprints, and retina recognition.

Secure Shell (SSH) is a cryptographic network protocol for securing data communication, remote command-line login, remote command execution, and other secure network services between two networked computers. It connects via a secure channel over an insecure network, with a server and a client running SSH server and SSH client programs, respectively [16]. Although the data field of an SSH packet is encrypted, the header fields of a packet are given in plain text. This plain text header can reveal information such as the type of packet, the source/destination IP, and the size of a packet.

Stepping-stones are a typical way for a hacker to hide his identity. Using stepping-stones makes tracking the IP address of an intruder difficult, since these stepping-stones may be located in different countries. A hacker may log in to a stepping-stone computer and start a new connection to another computer from this machine and do the same thing repeatedly. Those intermediate stepping-stones computers form a connection chain. Anytime such long chains are detected, it is a sign that someone may be trying to intrude into a system and conceal their activities. This is the type of intruder that we are trying to detect and prevent.

During a connection with SSH, each client's keystrokes generate a request packet, sometimes called "send packet"; Upon receipt, the server sends one or more response packets, sometimes called "echo packets" back to the client, which appear in the client's terminal window.

Ideally, the client host sends out the next request after receiving the response packet(s) from the previous request. However, this significantly slows down communication between the client and the server. To speed up communication, the TCP/IP protocol allows the client host to send out the request packets before receiving responses, up to a limit determined by the size of a buffer. In these cases, the request packet will "meet" the incoming response packet halfway. We call this situation "crossover". Every time a

request packet meets a response packet, one crossover is counted.

In summary, the type of intrusion that we discuss in this paper has the following characteristics: (1) the intruder must log in to a target host in order to steal information or cause other damage to that system and (2) the data content of the connection may be encrypted and not visible to any detection algorithm. Denial of service (DoS) attacks, for example, do not fall under this category.

2.2 Previous Work

There have been quite a few studies focused on stepping-stone detection, which have used various techniques to discover the hosts used by a hacker to avoid tracing. For each stepping-stone host, there may be a number of incoming and outgoing SSH connections. If the packet pattern of an incoming connection and that of an outgoing connection are highly correlated, that host is certainly suspicious.

Staniford-Chen and Heberlein [5] are the pioneers in the stepping-stone detection area. Their early research was based on the content of captured packets. Thumbprints are short summaries of connection packets captured in cross-traffic. They compared thumbprints to determine whether the bi-directional packets contain the same content. If so, the host is likely to be an intermediate node in a long connection chain. However, this method is only able to detect stepping-stones using unencrypted messages, such as on a Telnet connection. The algorithm cannot be applied when dealing with encrypted connections such as SSH.

Wang and Reeves [11] proposed a novel intrusion framework called "Sleepy Watermark Tracing" (SWT). It integrated a sleepy intrusion response scheme, a watermark correlation technique, and an active tracing protocol. The term "sleepy" refers to the fact that no overhead is introduced when no intrusion is detected, but the overhead becomes active when detecting an intrusion. The target host will inject a watermark into the backward connection of the intrusion, and then wake up to collaborate with routers along the chain. Wang and Reeves' work is quite inspiring, since they made it possible to trace back to the original source node in real time when a hacker uses a long chain to cover their presence in order to achieve interactive intrusion.

Due to its protection of privacy, SSH is gradually becoming the most useful remote connection tool over Telnet. In order to adapt to new ways of connection, Zhang and Paxson [4] proposed a method using timing correlation of ON/OFF periods of different connections to detect stepping-stone attacks. Because they rely on the properties of interactive traffic, such as packet sizes and idle periods, they do not need to look at packet contents on the connections in order to detect intrusions. Their algorithm provides good accuracy and reduced workload in capturing packets by only keeping packet headers. However, there is still a weakness in their approach: it cannot distinguish legitimate stepping-stones from malicious ones.

Yung's research [12] focused on estimating the RTT for outgoing connections using request-and-response pairs between a client and a downstream server. Yung's approach monitors an outgoing connection to estimate two metrics. The first metric is the time gap between requests from the client and acknowledgements from the downstream server, which is used to estimate the RTT between the client and the server. The second metric is the time gap between the client's request and the server's response, which is used to estimate how far away the targeted server is. These two metrics are also used to estimate how many hops there are on the connection chain from hacker to victim. This method performs well in identifying connections with more than two downstream hops and can be used on interactive sessions, such as Telnet and SSH. However, the time gaps are greatly influenced by the network environment and the end machine, since any delay in network traffic or on the server side could impact the size of the RTT.

Yang and Huang [6] proposed an algorithm to estimate the length of downstream connection chains by monitoring outgoing and incoming packets. Their algorithm computes the RTT in a request and response pair. By capturing the changes in these RTTs, the number of nodes in the downstream chain can be estimated. With this approach, the user can determine if his machine is being used as a stepping-stone when the hacker is connected to a target server, and then take measures to stop the intrusion. This method is able to detect intrusion in real-time, which is important. It also delivers an accurate result for estimating the length of a downstream connection chain.

After Yang and Huang's work [6], they proposed two additional algorithms [13] for matching TCP send-and-echo packets. One algorithm is relatively conservative, and can accurately match a smaller number of packets, whereas the other one is heuristic, and can match a larger number of packets with less accuracy. The authors justified the correctness of their conservative algorithm. By applying these two algorithms in their experiments, results show that both algorithms could achieve the same performance. If the conservative algorithm failed to generate enough data, the heuristic one would be used as a supplement. The combined algorithms can detect intruders in real-time. This approach can also estimate an encrypted connection chain length accurately, even with fluctuating network traffic. The major weakness of this approach is that it requires capturing packets throughout a given connection session. Since previous research [13] was not able to capture enough send packets using a conservative algorithm, Yang and Huang [14] proposed a new clustering partitioning algorithm to extract the timestamps from the send-and-echo packets in a connection chain.

Ding et al. [7] continued detection work from another perspective by proposing an intrusion detection algorithm based on the target host at the end of a chain, instead of on the intermediate host. They analyzed the delay between the time a user presses the enter button to finish a Unix command and the time that the user types the next character, and used an approximated upstream round-trip time to separate long connection chains from short ones. Their results show that this proposed algorithm is able to distinguish long from short connection chains with relatively low false positive rate under certain conditions. Our paper attempts to find a better detection algorithm under the same assumptions.

III. Packet Crossovers in the Connection Chain

Ding et al. [7] discovered the "crossover" issue in their examination of long connection chain packets. In this section we discuss the relationship between the number of crossovers and the length of a given connection chain. Throughout this paper, we will be using the following setup as our test environment. Our main goal is to distinguish and identify long connection chains from short connection chains. The length of a short chain is obviously where length equals 1 hop. One hop is defined as one connection between an SSH client and an SSH server. The length of a long chain in our experiment was set at 3 hops. The four servers used to create a 3-hop connection chain are shown in Figure 2. All three hops were routed on the Internet, not on our campus LAN. Note that we collected all packets at all four nodes for this experiment, but only used the data at the target host in Georgia for the detection analysis. The other data have been saved for future analysis. For the data in short chains, a connection was created between the University of Houston and Pittsburg.

Each time a user on the client-side presses a key, the client host generates a request packet with the padded and encrypted character as the content for the next server. The server sends a response packet back to the client after processing the request packet. If the server is an intermediate stepping-stone host, it turns around and sends the request via a separate packet to the next SSH server and waits for its response. If the server is the intended target host, then the host processes the request, which includes sending a response packet back with the same character in it. Occasionally, if a complete command has been received at the server side, the server sends one or more packets back with the reply. Thus RTT can be measured in the first host by computing the difference between the timestamps of the request and response packets. The first two communications in Figure 3 illustrate how such an RTT can be computed. Note that the processing time at the end of the chain is typically very small. A server may also send an acknowledgement packet if no response is ready to be sent. When the user closes the connection, the server process terminates, and the auxiliary server continues to listen for new SSH connection requests [15].

According to the TCP/IP protocol, a client is allowed to send a limited number of packets to the server without having to wait for a response. The packet RTT for different request packets will be different. This can be modeled as a random variable, depending on network traffic and the availability of the hosts involved. Normally, request and response packets stay in the same sequence at both ends of the connection. Also, RTTs between two consecutive hosts are much shorter than intervals between manual keystrokes.

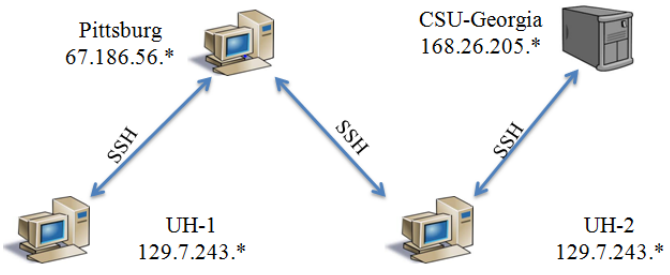

Figure 2: Set up of a three hops connection chain used in our experiments

We then formally define packet crossovers and state our assumptions about the crossovers. When TCP/IP services are started on an SSH server host, the auxiliary server creates a listening socket for the SSH, enabling it to accept a remote connection request. When the client executes an SSH command on a remote client host, the SSH client is initiated. The client reads the configuration file and initiates a TCP connection to a server host using the specified destination port. On an SSH server host, the auxiliary server creates a copy of the server process, which reads the server's configuration file. The SSH client and server exchange information about supported protocol versions. During the connection, the SSH server runs in a loop, accepting request messages from the client, performing required actions, and returning response messages to the client. For a stepping-stone type of connection, there is a different packet between every pair of successive hosts in a chain.

If a connection chain is long enough, the round-trip time of a packet may be longer than intervals between two consecutive keystrokes. For data transfer, the client is allowed to send further messages, subject to a limit, without waiting for the response to the request [16]. Therefore, if the client's keystroke intervals are longer than the RTT, the response packet will arrive at the client's machine before another request is sent out. On the other hand, if the client's keystroke intervals are shorter than the round-trip time of the previous packet, there will be two or more consecutive request packets sent before a response packet arrives. Thus, the probability that an RTT is greater than the key intervals 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 "meet" the coming request packet halfway before arrival, which is called "crossover." Two crossovers are shown in Figure 3.

Crossover of the request and response packets requires more explanation. When a packet "crossover" happens, the downstream node stays in the normal request/response packet order, whereas the upstream node sends out the next request

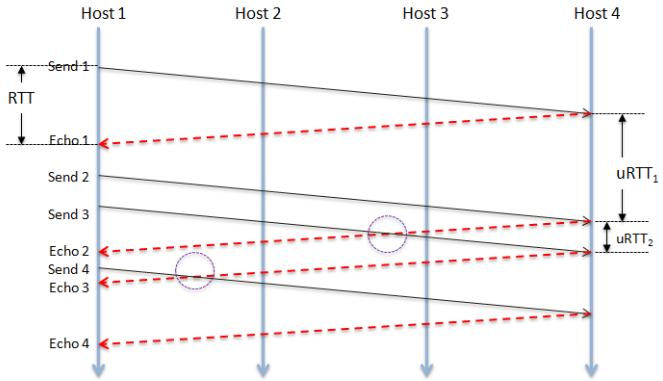

Figure 3: Illustrations of a uRTT and two packet crossovers (circled) along a chain of three hops

packet before receiving a response packet for the previous one. Then at the upstream node, there will be two consecutive request packets whose sequence is different from those at the downstream node.

The rest of this paper is based upon the assumption that there are more crossovers generated in a long connection chain than in a short one. In other words, the longer a connection chain is, the more crossovers there are, which means the number of packet crossovers is positively correlated to the length of the connection chain. We plan to validate this crossover assumption in a separate work.

IV. Detection of Long Connection Chains

The existence of the packet crossover and its relationship with the chain length provided a base for our algorithm. Although we made the assumption that a high number of crossovers implies a long connection chain, we were unable to use this to identify long connection chains, because of the lack of packet information. We did not have packet information along the whole chain, except for the last host where the monitoring algorithm resides (Figure 1). Nevertheless, a large number of crossover packets will alter the distribution of packet gaps. This section describes an algorithm that captures those gap variances resulting from a large number of packet crossovers. With this algorithm, it is possible to distinguish and identify long connection chains from shorter chains.

A reasonable way to approximate the chain length is by using the RTT. However, the monitoring algorithm is unable to calculate the RTT when the algorithm is located at the target server. The best we can do is an Upstream RTT (uRTT), which is defined as the time gap between sending a respnose packet and the receiving of the next request packet at the target server An example is provided of RTT and uRTT in Figure 3. The uRTT in general does not represent the real RTT because there is potentially a delay before the next request packet is sent. An example of uRTT is shown in Figure 3; it is the difference between the timestamps of a response packet and that of the next request packet received at the target server.

Even though uRTTs are not very accurate in general, we were hoping that one of them would provide a good estimation of the true RTT. This is possible when the client queues the key strokes pending the reply packets. So we looked at the minimum of all uRTTs, and we were surprised by some of the values, as some were much smaller than what real RTTs should be. It turned out that the problem was caused by the packet crossovers, particularly 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 RTT or the first $uRTT_1$ in Figure 3. The reason for this difference is precisely due to the crossover of a response packet (Echo 2) with a Request packet (Send 3).

We can classify all the uRTT gaps into two types: "Inter-command Gap" and "Intra-command Gap" [7]. Each operating system command is usually followed by an "Enter" key. Inter-command Gap refers to the time gaps between a return character and the first character of the next Unix command entered by the user. Intra-command Gap refers to the time gaps between two keystrokes within a single command, i.e., with no return or end-of-line characters.

The reason for separating Inter- and Intra-command Gaps is to filter out some of the packet gaps that are not contributing to our detection algorithms. The Intra-command Gaps essentially measure the typing speed of a user and do not depend on the length of the chain. On the other hand, Inter-command Gaps may be influenced by the chain length. The user may have to digest the result from the prior command to determine what to do next. An algorithm was presented in [7] to filter out Intra-command Gaps. The Inter-command Gaps are used to build a profile characterizing a given connection chain.

From our experiments, we conclude that long connection chains will generate a number of packet crossovers. Even though we do not have solid numbers to support this claim, from our observations, the number of crossovers is proportional to the length of the connection chain. These crossovers resulted in some relatively small Inter-command Gaps that were smaller than the actual RTT. Thus, if there are an unusual number of small Inter-command Gaps, it is highly likely to be a long connection chain.

Figure 4 shows two sample "profiles" of Inter-command Gaps, one for a 1-hop chain and the other for a 3-hop chain. The profile is sorted in ascending order of the gap values after filtering out the Intra-command Gaps. In general, the gap values are higher for the longer chain. However, the gap values dropped at the beginning of the profile for the long chain, i. e., there are some very small uRTTs for the long chain, visible in the two curves in Figure 4. This dip happened in all 20 long chain cases that we tested. On the other hand, there is no significant dip for the short chain, represented by the dotted curve in Figure 4. Consequently we can view the existence of very small uRTTs as a signature for a long chain. Unfortunately, we cannot measure the packet crossover by only examining the packets at the target server. We are able to observe a side-effect of the dip in the profile. After the first few very small uRTTs, the gap value has to increase to their normal, higher value. So, there is a sharp rise in the gap values in the profile.

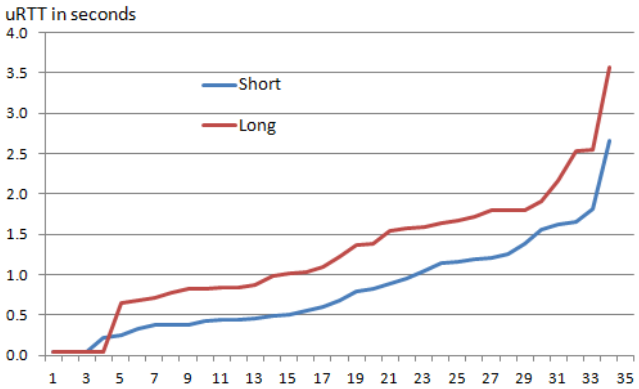

Figure 4: Comparison of Inter-command Gap profiles (sorted by uRTT) of long chain vs. short chain

We then reduced the problem of finding a long connection chain to finding a sharp rise in the profile of its Inter-command Gaps. There are two ways to measure the "sharp rise" of a curve: (a) measure the difference of the successive gaps and look for the largest difference, or (b) measure the ratio of two successive gaps and look for the largest ratio. Both of these methods were tested and the ratio approach turned out to be the better method. An algorithm to find a long connection chain is shown below. In the algorithm, we computed the ratio of two successive gaps as a measurement of the sharpness of the rise of the profile. If the largest rise in the ratio is higher than a threshold value, the algorithm returns "long connection chain".

Detection Algorithm: to determine whether the connection chain is a long or short one, given all the packet information at a target server.

Step 1: Extract request/response SSH packets from data collected at the targeted server of the selected connection.

Step 2: Compute the uRTT gaps of successive packets and sort them in ascending order.

Step 3: Filter out the Intra-command Gaps and keep only the Inter-command Gaps $G[i]$ sorted in ascending order.

Step 4: Compute the ratios of successive gaps over their previous ones, $R[i] = G[i]/G[i\text{-}1]$. (An alternative is to find the difference of the two gaps.)

Step 5: Find the maximum gap ratio

$$mgr = max\{R[i] | \, i = 2, \ldots, \mathrm{n}\}$$

and if this maximum ratio *mgr* is greater than a predetermined threshold *t*, it is considered to be a long connection chain. Otherwise, it is a short one.

The above algorithm did not specify what threshold value to use, because there is no universal threshold that can be used in all situations. Each installation should conduct its own experiment and derive the best threshold to use for that server. We shall describe the algorithm to find such a threshold below. There is an obvious trade-off between the accuracy of our ability to detect long connection chains (lower *t* value) and the false positive rate.

V. VALIDATION OF THE ALGORITHM

To validate the algorithm, we designed an experiment to compare long connection chains with short connection chains. The experiment was conducted over the Internet starting from the University of Houston (UH) campus. A short chain is defined as one single connection (1-hop) from our campus to a host off-campus. A long connection is defined as a 3-hop chain, i. e., a chain of 4 hosts connected with three connections. All intermediate stepping-stone hosts are located within the U.S., but are not geographically close to our campus. A total of 20 short chains and 20 long chains were collected for our analysis.

To illustrate how the algorithm works, we shall look at some selected sample profiles. Table I presents three of the twenty long-connection cases tested. For each case, we listed the Inter-command Gaps and the gap ratio computed in Step 4 of the algorithm.

Due to the packet crossovers, the first few gaps in the profiles for long chains are very small; thus, it is easier for us to identify the sharp increase in the gap ratio. As one can see, there is one ratio in bold that stands out among all of the ratios in each of the three cases. The first two cases are the *typical* cases from the experiments, while Case 3 is the *worst* case, i. e., the profile with the lowest *mgr* value among all twenty test cases. Note that we only listed the first fifteen values of the profile to save space. Showing more values does not give more insight into how the algorithm works.

TABLE I. THREE CASES OF RATIOS OF CONSECUTIVE GAPS AMONG TWENTY LONG CONNECTION CHAINS COLLECTED IN OUR EXPERIMENTS (CHAIN LENGTH = 3 HOPS).

	Case 1		Case 2		Case 3	
	Gap	Ratio	Gap	Ratio	Gap	Ratio
1	0.045		0.020		0.043	
2	0.046	1.022	0.043	2.190	0.045	1.030
3	0.046	1.001	0.044	1.021	0.045	1.003
4	0.047	1.021	0.045	1.020	0.045	1.001
5	0.512	**10.972**	0.045	1.005	0.069	1.550
6	0.520	1.017	0.551	**12.138**	0.073	1.054
7	0.523	1.005	0.553	1.004	0.190	**2.604**
8	0.551	1.053	0.686	1.241	0.354	1.863
9	0.591	1.073	0.690	1.005	0.459	1.295
10	0.651	1.103	0.724	1.049	0.473	1.031
11	0.655	1.005	0.734	1.014	0.527	1.114
12	0.660	1.009	0.765	1.042	0.554	1.050
13	0.672	1.017	0.798	1.043	0.613	1.108
14	0.698	1.039	0.851	1.067	0.637	1.039
15	0.700	1.003	0.855	1.004	0.660	1.036

We did a similar analysis for short connection chains, as illustrated in Table II. The ratios for the short chains were supposed to be low, but we did see some outlier values, as shown in Case 3, which is, again, the *worst* case of the twenty short connection chains. The reason for this very strange worst case is unknown. However, for most of the cases, the *mgr* for the short chains, highlighted in Table II, is smaller than those of the long chains. We can use this as a feature to separate the two cases.

The algorithm then took the forty *mgr* numbers, mixed them together, and sorted them in ascending order. As shown in Figure 5, most short chain (blue circles) and long chain (red squares) data points stay in the lower part and the upper part of the chart, respectively. Figure 5 may be used to evaluate the accuracy of our algorithm. For example, if we select the threshold *t* to be 6.0, then there is one short chain misclassified: the only blue circle above 6.0, which is a false positive. At the same time, there are three long connection chains that we were unable to detect, or three false negative cases. The system administrator at the target server can determine the threshold value depending on the false rate he/she is willing to tolerate.

TABLE II. THREE CASES OF RATIOS OF CONSECUTIVE GAPS AMONG TWENTY SHORT CONNECTION CHAINS COLLECTED IN OUR EXPERIMENTS (CHAIN LENGTH = 1 HOP).

	Case 1		Case 2		Case 3	
	Gap	Ratio	Gap	Ratio	Gap	Ratio
1	0.051		0.052		0.050	
2	0.052	1.022	0.052	1.000	0.052	1.035
3	0.052	1.004	0.055	1.059	0.499	**9.558**
4	0.127	**2.443**	0.127	**2.308**	0.513	1.027
5	0.195	1.538	0.241	1.905	0.519	1.012
6	0.196	1.006	0.260	1.077	0.526	1.014
7	0.204	1.041	0.283	1.090	0.583	1.108
8	0.287	1.409	0.310	1.095	0.596	1.023
9	0.306	1.063	0.339	1.094	0.598	1.003
10	0.363	1.189	0.350	1.032	0.614	1.027
11	0.397	1.093	0.372	1.064	0.630	1.026
12	0.429	1.079	0.388	1.042	0.678	1.076
13	0.431	1.004	0.469	1.208	0.701	1.033
14	0.442	1.026	0.510	1.088	0.702	1.003
15	0.470	1.065	0.533	1.044	0.706	1.005

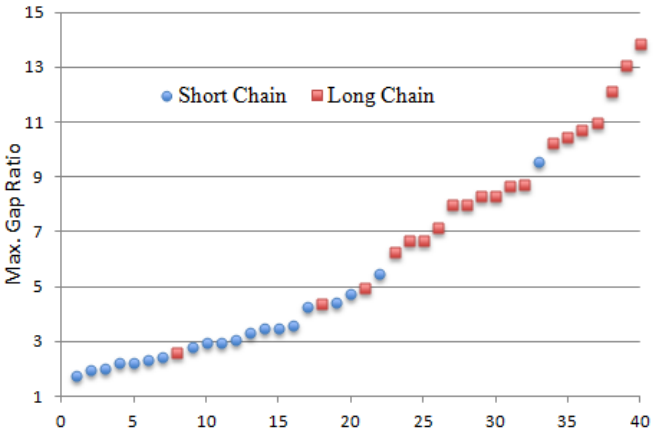

Figure 5: Comparison of the Maximum Gap Ratio (MGR) of 20 short chains and 20 long chains

In order to better examine the detection rate of this method, we set the ratio threshold at different levels to distinguish long connection chains from short ones. For example, if we set the ratio threshold as 6, there is only one data point falling into the false positive category, which means a false positive ratio of 5%. Meanwhile, seventeen of the twenty long connection chains are correctly detected with a accuracy (true positive) rate of 85%.

In this way, we can generate an ROC (Receiver Operating Characteristic) curve with different combinations of false/true positive ratios. The ROC curve gives a simple visual evaluation of the accuracy and false alarm rate of our method in long connection chain detection. Depending on the amount of false positives one is willing to tolerate, one can estimate the accuracy of the detection algorithm. The ROC curve is presented in Figure 6, which turns out to be better than that of previous research. Note that in our experiment, the goal was to be able to identify long chains of 3 hops vs. 1 hop. This is much more challenging than the prior research which is to

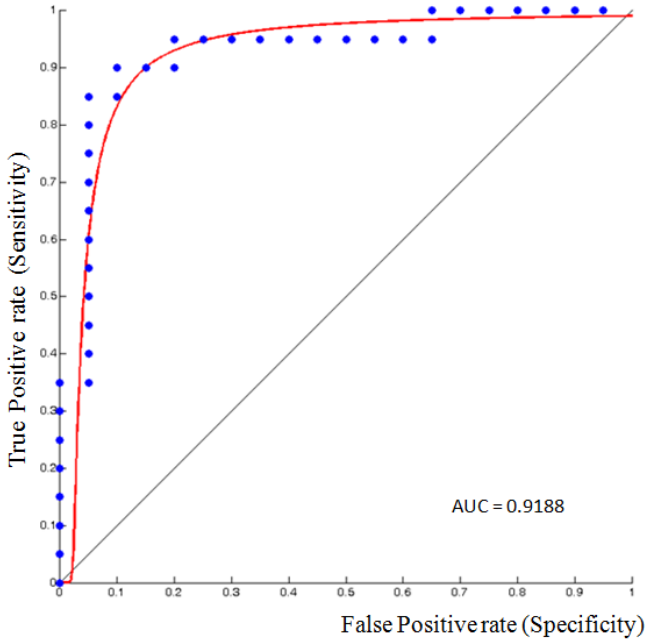

Figure 6: ROC Curve showing the accuracy of the detection algorithm vs. false positive rate

separate a length of 4 hops vs. a length of 1 hop. We also tested 5 hops vs. 1 hop and the detection rate is 100% with no false positives. That result is not shown here, to save space.

VI. CONCLUSION

Cyber attacks through stepping-stones have been widely used by hackers to avoid being detected. Stepping-stones intrusion detection can effectively prevent the hackers from being traced back to their source. It is important for a server to be sure that there are no long stepping-stone chains that are connecting to it through an SSH server.

In this paper, we proposed a new approach to detect long connection chains based on the hypothesis that the number of packet crossovers is approximately proportional to the length of a given connection chain. Our approach is able to detect connection chains of three hops with a better accuracy than the previous research. Our algorithm is based on the sharp rise of the Inter-command Gap profile of a chain. Experiments were conducted to validate our algorithm, and for a 3-hop chain we were able to detect 85% of the long chains with only a 5% false positive rate. This result is based on a particular threshold we picked in the analysis. The threshold is not arbitrary; it depends on each installation of the algorithm depending on network speed, among other factors. By studying the training cases, each installation can determine a threshold that gives them an acceptable False Positive rate.

Note that this paper did not address the issue of estimating packet crossovers. It remains to be seen if we can collect enough evidence to validate the hypothesis about packet crossovers. The strategy proposed in this paper was designed for SSH protocol only. Future work will investigate whether

this method works for other protocols, for example, a chain of HTTP or SOCKS proxies.

Our research helps servers to identify potentially dangerous connections by detecting users connecting to a SSH server by using long connection chains to hide their identity. The algorithm requires packet data at the server, i. e., end of chains only. The passive algorithm does not interfere with the connection. In our experiment, we used Wire Shark to collect packet data for off-line analysis. The algorithm may be converted to an online real-time monitor algorithm that can alert a system administrator during an attack.

ACKNOWLEDGEMENTS

We would like to thank Chenlei Zhang, Lingjia Deng, and Dr. Jianhua Yang for their assistance in making their servers available for our experiments. This research was supported in part by NSF Grant 1062954.

REFERENCES

[1] LinkedIn Corp. Linkedin Passwords Compromise. URL: http: //blog.linkedin.com/20-12/06/06/linkedin-member-passwords-compromised/, accessed April 8, 2014.

[2] Sony Corp. Sony PlayStation Network/Qriocity Services Outage. URL: http://bl-og.us.playstation.com/2011/04/22/update-on-playstation-network-qriocity-services/, accessed April 8, 2014.

[3] Wikimedia Foundation. Sony PlayStation Network Outage. URL: http://en.wiki-pedia.org/wi-ki/PlayStation_Network_outage/, accessed April 8, 2014.

[4] Y. Zhang and V. Paxson. Detecting stepping stones. In Proceedings of the 9th Conference on USENIX Security Symposium, volume 9, pages 171-184, 2000.

[5] S. Staniford-Chen and L. Heberlein. Holding intruders accountable on the Internet. In Proceedings of IEEE Symposium on Security and Privacy, pages 39-49, May 1995.

[6] J. Yang and S. Huang. A real-time algorithm to detect long connection chains of interactive terminal sessions. In Proceedings of the 3rd International Conference on Information Security, pages 198-203, 2004.

[7] W. Ding, M.J. Hausknecht, S. Huang and Z. Riggle. Detecting stepping-stone intruders with long connection chains. In Proceedings of the 5th International Conference on Information Assurance and Security, Volume 2, pages 665-669, 2009.

[8] Peter G. Neumann and Phillip A. Porras. Experience with emerald to date. In 1st USENIX Workshop on Intrusion Detection and Network Monitoring, pages 73-80, April 1999.

[9] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. In Technical Report 99-15, Department of Computer Engineering, Chalmers University, March 2000.

[10] Richard P. Lippmann. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In Proceedings of DARPA Information Survivability Conference and Exposition, pages 12-26, 2000.

[11] X. Wang and D. Reeves. Sleepy watermark tracing: An active network-based intrusion response framework. In Proceedings of the 16th International Information Security Conference, pages 369-384, 2001.

[12] K. H. Yung. Detecting long connection chains of interactive terminal sessions. In Proceedings of Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, pages 1-16, 2002.

[13] J. Yang and S. Huang. Matching TCP packets and its application to the detection of long connection chains on the Internet. In 19th International Conference on Advanced Information Networking and Applications (AINA), Volume 1, pages 1005-1010, 2005.

[14] J. Yang and S. Huang. A clustering-partitioning algorithm to find TCP packet round-trip time for intrusion detection. In 20th International Conference on Advanced Information Networking and Applications (AINA), Volume 1, pages 231-236, 2006.

[15] Hewlett-Packard. How the SSH Client and Server Communicate. URL: http://h71000.www7.hp.com/doc/83final/ba548_90007/ch01s04.html, accessed April 8, 2014.

[16] T. Ylonen and C. Lonvick, The Secure Shell (SSH) Connection Protocol, IETF RFC 4254, January 2006; http://www.ietf.org/rfc/rfc4254.txt.

[17] Wikimedia Foundation. OSI model. URL: http:// en.wikipedia.org/ wiki/OSI_model, accessed April 8, 2014.

[18] Ping Li; Wanlei Zhou; Yini Wang, "Getting the Real-Time Precise Round-Trip Time for Stepping Stone Detection," 2010 4th International Conference on Network and System Security (NSS), pp.377-382, 1-3 Sept. 2010.