Detecting Long Connection Chains of Interactive Terminal Sessions

Kwong H. Yung

Stanford University Statistics Department
390 Serra Mall; Stanford CA 94305-4020; USA
khyung@stat.stanford.edu

Abstract. To elude detection and capture, hackers chain many computers together to attack the victim computer from a distance. This report proposes a new strategy for detecting suspicious remote sessions, used as part of a long connection chain. Interactive terminal sessions behave differently on long chains than on direct connections. The time gap between a client request and the server delayed acknowledgment estimates the round-trip time to the nearest server. Under the same conditions, the time gap between a client request and the server reply echo provides information on how many hops downstream the final victim is located. By monitoring an outgoing connection for these two time gaps, echo-delay comparison can identify a suspicious session in isolation. Experiments confirm that echo-delay comparison applies to a range of situations and performs especially well in detecting outgoing connections with more than two hops downstream.

Keywords: Stepping stone, connection chain, intrusion detection, computer security, network security, network protocol, terminal session, delayed acknowledgment, reply echo, echo delay.

1 Introduction

Network security and intrusion detection have become important topics of active research [1,4,5,3]. As the use of the Internet becomes more common and widespread, so have network attacks and security breaches. Because the growing number of network attacks is ever more costly, network security and intrusion detection now play a crucial role in ensuring the smooth operation of computer networks.

1.1 Motivation

A skilled computer hacker launches attacks from a distance in order to hide his tracks. Before launching an actual attack with noticeable consequences, a skilled hacker will break into many computers across many political and administrative domains, to gather computer accounts. With access to multiple computer

A. Wespi, G. Vigna, and L. Deri (Eds.): RAID 2002, LNCS 2516, pp. 1–16, 2002.

$$-m \longrightarrow -m+1 \longrightarrow \ldots \longrightarrow -1 \longrightarrow 0 \longrightarrow 1 \longrightarrow \ldots \longrightarrow n-1 \longrightarrow n$$

Fig. 1. Typical connection chain. Relative to stepping stone 0, machines $-m, -m+1, \ldots, -1$ are *upstream* and machines $1, 2, \ldots, n$ are *downstream*.

accounts, the hacker can chain these computers through remote sessions, using the intermediate computers in the chain as stepping stones to his final victim.

Figure 1 shows a typical connection chain. Computer $-m$ attacks computer n, via stepping stones $-m+1, -m+2, \ldots, 0 \ldots, n-2, n-1$. Since the logs of the final victim n traces back only to the nearest stepping stone $n-1$, the originating point $-m$ of the attack cannot be determined without logs from the upstream stepping stones. Although the originating point used by the hacker may be found through repeated backtrace, this process is slow and complicated because the stepping stones belong to different political and administrative domains and often do not have logs readily available. Even when the originating attack point is found after costly investigation, the hacker will have already left and eluded capture.

This report presents a simple technique to detect interactive terminal sessions that are part of long connection chains. Once an outgoing session is found to have many hops downstream, the server machine can terminate the outgoing connection, to avoid being used as a stepping stone in the long connection chain. Moreover, the suspicious outgoing terminal session is useful as a basis for finding other sessions on the same chain.

1.2 Previous Approaches

Staniford-Chen and Heberlein [8] introduced the problem of identifying connection chains and used principal-component analysis to compare different sessions for similarities suggestive of connection chains. Because the packet contents were analyzed, the technique did not apply to encrypted sessions.

Later, Zhuang and Paxson [10] formulated the stepping stones problem and proposed a simpler approach to finding two correlated sessions, part of the same connection chain. By using only timing information of packets, the technique also applied to encrypted sessions.

Both [8] and [10] aim to match similar session logs, indicative of connection chains. Unless sessions on the *same* connection chain are in the same pool of collected sessions, however, this approach fails to identify suspicious sessions. For example, [10] analyzed logs from a large subnet at UC Berkeley. A hacker initiating an attack from a computer on the subnet would not be detected unless his chain connects back into the subnet.

The strategy of grouping similar sessions also inadvertently detects many benign, short connection chains because legitimate users often make two or three hops to their final destinations. For example, the Stanford University database

research group has one well-protected computer that allows incoming connections only from trusted computers on the Stanford University network. To connect to this protected machine, an off-campus user must connect via another computer on the campus network. Because restrictions on incoming connections are quite common in heavily protected networks, short connection chains are often necessary and harmless.

1.3 New Strategy

This report presents an alternative strategy for identifying connection chains. Because sessions on a long chain behave differently than sessions on a direct connection, a suspicious session can be detected in isolation without finding other similar sessions on the same connection chain. The proposed technique makes use of delayed-acknowledgment packets, response signals found in typical protocols for interactive terminal sessions, such as telnet, rlogin, and secure shell.

Like [10], *echo-delay comparison*, the technique proposed here, relies only on timing information of packets and so applies equally well to encrypted sessions. Rather than comparing sessions in a large pool, echo-delay comparison operates on a single session and thus solves the shortcomings of [8] and [10]. A suspicious session that is part of a long connection chain can be identified even without finding another correlated session.

Clearly many factors determine the behavior of a connection, including the network, the machine, the user, and the session transcript. Therefore, isolating the distinctive properties of sessions in long connection chains is extremely difficult. Yet research in this strategy is worthwhile because looking for similar sessions from a large pool has many inherent drawbacks. Of course, these two strategies are not mutually exclusive but rather complementary.

To balance out the many uncontrolled factors in a connection, echo-delay comparison relies on the logistics of interactive terminal sessions. Because details of specific protocols involved are used, echo-delay comparison does not apply to all types of connections. Nevertheless, the technique is quite simple and can be extended to handle related protocols.

1.4 Elementary Solutions

The original Internet protocols were not designed with security as the main objective. Under the shield of anonymity offered by the Internet, malicious users can attack many computers remotely. Although the prevention and detection of attacks are now important, the widespread use of older protocols is difficult to change because the global connectivity of the Internet often requires compatibility with older software. Upgrading the vast number of computers on the Internet proves to be nearly impossible.

Perhaps the simplest way to prevent connection chains from forming is to forbid all incoming sessions from executing any outgoing terminal sessions. Some servers indeed do adopt this policy and forbid most outgoing connections. Yet implementing such a strict policy severely limits legitimate users.

Any policy blindly disabling outgoing connections is too restrictive in most settings because there are many legitimate reasons to connect via a short chain. On many networks, users are allowed external connections only through a dedicated server, which is heavily protected and closely monitored. So to connect to an outside host, users must connect via the dedicated server. In this case, the gateway server cannot blindly forbid outgoing connections.

Legitimate computer users often use short connection chains to get from one host to another. Malicious hackers generally use long connection chains to cover their tracks before executing an attack. To protect a machine from being used as a stepping stone in a long chain, a reasonable policy would be to terminate sessions that continue *multiple* hops downstream.

1.5 Outline of Report

Following this introduction, Section 2 provides a brief background of the network signals sent during an interactive terminal session between a client and a server in a direct connection. Section 3 then explains the dynamics of a connection chain and introduces two time gaps useful for detecting long connection chains. Next, Section 4 presents the mathematics for calculating and comparing the two time gaps. Afterwards, Section 5 presents two sets of experiments used to test the proposed technique. Towards the end, Section 6 discusses the advantages and limitations of the ideas proposed in this report. Finally, Section 7 summarizes this report's main conclusions.

2 Background

By comparing its incoming sessions with its outgoing sessions, a machine can determine that it is being used as a stepping stone in a connection chain. Standard protocols for remote sessions do not provide information about the length of the connection chain. So in isolation, the machine cannot in principle determine whether it is being used as part of a long chain or just a short chain.

2.1 Reply Echo

In most client implementations of interactive terminal sessions, each individual character typed by user will initiate a packet sent from the client to the server. Once the server receives the character packet, it usually echoes the character back to the client, instructing the client to display the typed character on the client screen. This reply echo is the typical response of the server to a non-special character from the client.

When the user types a carriage return, the carriage return received at the server usually triggers the server to execute a special command. After executing the command, the server then sends back command output. Figure 2 illustrates a typical exchange between a client and a server on a direct connection.

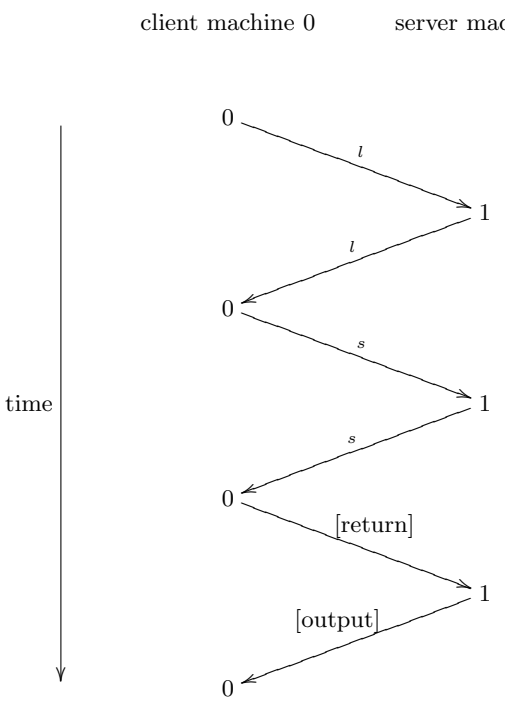

Fig. 2. Interactive session on direct connection. Client 0 sends the `ls` command to server 1 and receives back the directory listing. The vertical time axis is not drawn to scale.

2.2 Delayed Acknowledgment

Most often, the server responds soon enough with a nonempty packet, which also functions to acknowledge the client request. If the requested command requires a long execution time, then the server times out and sends a so-called *delayed acknowledgment*. This delayed-acknowledgment packet contains no content but signals to the client that the server indeed received the client request. Thus, the delayed acknowledgment functions to keep the conversation between the client and the server alive, at the expense of sending an empty packet. The server sends a delayed acknowledgment only when it cannot send a nonempty response in time. The server implementation determines the actual delay tolerance before a delayed acknowledgment is sent from the server to client.

Similarly, when the server sends a nonempty packet to the client, the client must acknowledge the server. Usually this acknowledgment to the server is sent by the client along with the next character packet. If the user is slow to type the next character, however, the client times out and sends a delayed acknowledgment to the server. The client implementation determines the delay tolerance before a delayed acknowledgment is send from the client to the server.

3 Theory

A machine used as a stepping stone in a connection chain only knows about its incoming and outgoing sessions. Typically the stepping stone only passes the packet information from its client onto its server. In the Figure 1, the stepping stone 0 acts as a server to receive the incoming connection from upstream client machine -1. The stepping stone 0 then acts as a client to forward the outgoing connection onto downstream machine 1. This propagation of signals starts from the very first originating client $-m$ to the final victim destination n.

After the final victim n receives the packet and executes the command, the output from n is then forwarded back to the originating client $-m$ in a similar manner. The intermediate stepping stones $-m+1, -m+2, \ldots, n-2, n-1$ act mainly as conduits to pass the packets between $-m$ and n. Along the way, packets may be fragmented or coalesced. Moreover, packets may be lost and retransmitted. Because human typing generates character packets separated by relatively large time intervals, character packets are generally not coalesced.

3.1 Interactive Terminal Sessions

Protocols for standard interactive terminal sessions do not provide information about an outgoing connection beyond the first hop. So there is no certainty about how many additional hops an outgoing connection will continue downstream. In this scenario, the client machine only communicates with the nearest server one hop downstream and does not know about additional servers several hops downstream.

After sending out a character packet to the server, the client waits for a reply echo from the server. If the final victim machine is many hops downstream from the client, then the nearest server must forward the character packet downstream. Upon receiving the character packet, the final victim then sends the reply echo to the client via the client's nearest server. To the client, the reply echo appears to come from the nearest server.

3.2 Dynamics of Connection Chains

In a connection chain with many hops downstream, there is a long time gap between the client request and the server reply echo because the nearest server must forward the client request to the final victim and then pass the reply echo back to the client. A long delay between the client request and the server reply echo also results if the client and the nearest server are separated by a noisy connection or if the server is just slow. *Consequently, a long echo delay alone is not sufficient to suggest that there are many hops downstream.*

As soon as the nearest server receives the client request, the nearest server forwards the request. If the nearest server cannot pass the reply echo from the final victim back to the client in time, the nearest server sends a delayed acknowledgment to the client in the meantime. So if there are many hops downstream of the nearest server, the client will first receive the delayed acknowledgment

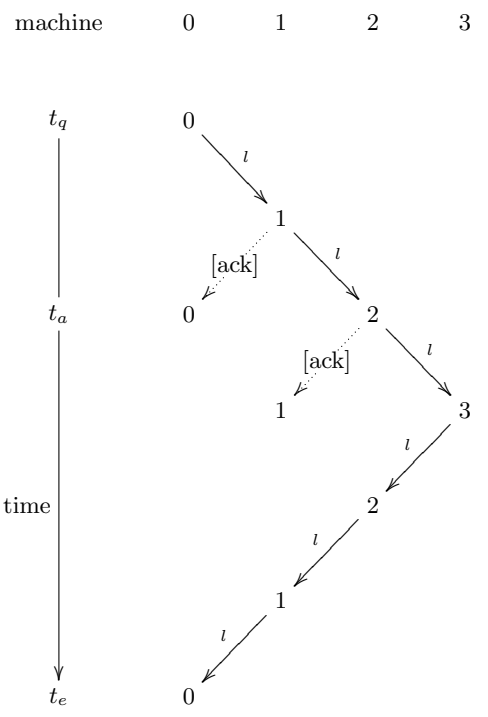

Fig. 3. Interactive session on connection chain. Client 0 issues a character packet containing letter l. Downstream servers 1 and 2 forward the packet to the final victim 3. After executing the packet, the final victim 3 sends the reply echo back to the client 0, via the stepping stones 1 and 2. In the above scenario, client 0 logs three packets, at times $t_q, t_a,$ and t_e.

from the nearest server before receiving the reply echo. Figure 3 illustrates the dynamics of a connection chain in the simplest scenario.

3.3 Two Time Gaps

In the simplest scenario, a client sending out a character packet can record three different signals at three points in time,

$$t_q < t_a < t_e. \tag{1}$$

Here t_q is the time of character request sent from the client, t_a is the time of delayed acknowledgment received from the server, and t_e is the time of reply echo received from the server. All three times refer to when a packet leaves or arrives at the client. Figure 3 illustrates these three time points.

The delayed-acknowledgment gap $t_a - t_q$ provides an overestimate of the travel time between the client and the nearest server. The reply-echo gap $t_e - t_q$ provides an estimate of the travel time between the client and the final victim.

The difference between these two gaps provide an estimate of the number of additional hops downstream beyond the nearest server.

4 Analysis

In encrypted connections, as with secure shell, the packet contents are obfuscated and cannot be compared. Thus, delay times cannot be calculated easily by matching a character packet to its reply echo. Any technique that applies equally well to encrypted connections, therefore, cannot rely on the packet content. The analysis technique described below, by design, applies to interactive terminal sessions, including telnet, rlogin, and secure shell.

4.1 Reducing Network Logs

After ignoring the packet content, the communication between the client machine and the server machine is reduced to a sequence $X = (x_1, x_2, \ldots, x_i, \ldots)$ of packets recorded at machine 0. Let t_i denote the time of packet x_i, as recorded at machine 0. Let o_i denote the issuing host of packet x_i. Here $o_i = 0$ if packet x_i is sent from client 0 to server 1, and $o_i = 1$ if packet x_i is sent from server 1 to client 0.

The calculations described below provide estimates of the time gaps instead of the exact values. The packet content is not used in these simplified calculations. Instead, packet header provided the necessary information: the logging time, the issuing host, the receiving host, the port numbers, and the packet flags.

4.2 Estimating Gap of Reply Echo

The gap $t_e - t_q$ between the client request and the server reply echo is estimated by considering only *nonempty* packets, essentially those packets whose flags are not delayed-acknowledgment flags. Define $(x_{i_k}) = (x_j : x_j \text{ nonempty})$ as the subsequence of packets with nonempty content. Then the set

$$E(X) = \{t_{i_{k+1}} - t_{i_k} : (o_{i_k}, o_{i_{k+1}}) = (0, 1)\} \tag{2}$$

captures all the gaps for a (0,1)-transition in the packet sequence X.

Certain (0,1)-transitions correspond to the server execution of a client command. Other (0,1)-transitions correspond to the reply echoes to the client's single-character packets. Because a user can type a short burst of characters before the first character is echoed back, a (0,1)-gap generally measures the time between a character, followed by the server reply echo to an earlier character in the burst of characters typed. The set E of (0,1)-gaps include both long command-execution gaps and also include short gaps for reply-echo lags. Thus, the distribution (0,1)-gaps in E has wide variance and skew.

4.3 Gap of Delayed Acknowledgment

The gap $t_a - t_q$ between the client request and the server delayed acknowledgment is calculated by considering the sequence of all packets. Delayed-acknowledgment packets from the server to the client is easy to identify from the packet header. Each such delayed acknowledgment can be matched to the most recent nonempty packet from the client. More precisely, let a_i be the gap between packet i and its most recent nonempty packet from the client, defined as

$$a_i = \min\{t_i - t_l : l < i, o_l = 0, x_l \text{ nonempty}\} \tag{3}$$

$$= t_i - t_{[i]}, \tag{4}$$

where $[i] = \max\{l < i : o_l = 0, x_l \text{ nonempty}\}$. Then the calculated set A of delayed-acknowledgment gap is

$$A(X) = \{a_i : o_i = 1, x_i \text{ delayed acknowledgment}\}. \tag{5}$$

This distribution of these delayed-acknowledgment gaps in A tends to be sharply focused, compared the distribution of (0,1)-gaps in E. In any case, $\alpha(X) = \max A(X)$, the maximum of the delayed-acknowledgment gaps in A, provides an over-estimate the round-trip time between the client and the server.

4.4 Comparison of Delay Gaps

Because the set E of (0,1)-gaps contains many different types of gaps, the distribution of these (0,1)-gaps depend on many factors. In general though, if there are many slow connections downstream, then the (0,1)-gaps tend to be large. Comparing the maximum delayed-acknowledgment gap α to the distribution of (0,1)-gaps in E provides insight into the connections downstream.

A simple comparison of α to E is the quantile $\text{quan}(E, \alpha)$ of α with respect to E. This quantile is robust estimate of how large E is compared to α. If the downstream delay is large, then $\text{quan}(E, \alpha)$ would be small.

5 Results

Experiments under a variety of settings were conducted to test the technique proposed in this report. The author had root access to one machine at Stanford University and user access to many remote machines. Connection chains were tested on these accounts, with different numbers and orderings of computers. All the results presented here use the secure shell protocol for the connections.

5.1 Experimental Setting

The author's machine ST ran the Red Hat 7.2 distribution of Linux. This logging machine used Snort [7] 1.8.3 to collect network packet dumps for incoming and outgoing connections.

Most of the remote machines were located throughout the US, and several were located in Europe. They ran a variety of operating systems, including Linux, FreeBSD, Solaris, VMS, and S390. All of them supported incoming and outgoing terminal sessions, with support for secure shell, either version 1 or version 2.

Each logged session lasted between one to ten minutes. In these experiments, the network dumps were not analyzed online but after the experiments were completed. The simple analysis on the network packet dumps, however, could have been performed in an online setting.

In all experimental setups, the logging machine ST acted as a client 0 to the nearest server 1. Simple one-pass online algorithms were used to calculate the two delay gaps from connection between the logging client and the nearest server.

5.2 Experimental Results

Echo-delay comparison proved useful under many situations, as confirmed by many experiments. Two different sets of experimental results are reported here. Both sets used the machine ST to record network logs. In the first set, the recording machine was a *stepping stone* in the connection chain. In the second set the recording machine was the *originating point* of the chain.

5.3 Recording on Stepping Stone

In this set of experiments, each connection chain had a topology of the form $-1 \longrightarrow 0 \longrightarrow 1 \longrightarrow \ldots \longrightarrow n$, where machine 0 refers to the recording machine ST. The length of the chain and the identity of the machines varied. The same sequence of commands were executed in all the sessions. To control for variability in network congestion, all the experiments were conducted in the same time frame, within one hour.

Table 1 shows three groups, each of three chains. There are nine network logs, corresponding to the nine chains presented. Because the outgoing connection from client 0 is ST-rs, Essentially same delayed-acknowledgment gap α is used for all nine chains. The distributions E of (0,1)-gaps differ.

The second group of three chains replicates the first group of three chains. Yet the comparison quantiles differ slightly because there are still uncontrolled variability between the two replications.

In the third group of three chains, the quantile $\text{quan}(E, \alpha)$ is not monotonically increasing as the chain length decreases. On the other hand, the quantile $\text{quan}(E, 2\alpha)$ is better-behaved. The three groups also seem more consistent on the quantile $\text{quan}(E, 2\alpha)$. Apparently, the $\text{quan}(E, 2\alpha)$ feature discriminates more sharply the three chain lengths.

By rejecting chains whose $\text{quan}(E, 2\alpha)$ is less than 0.95, the two long chains of each group would be identified as suspicious. Decreasing this fraction would be less restrictive but also allow more suspicious chains. In any case, the rejection

Table 1. Quantile behavior under varying topology but identical session transcript. In each chain, the leftmost initial denotes the originating point, and the rightmost initial denotes the final victim. All network logs were recorded on machine ST. All sessions executed the same sequence of commands, using only the simple command line under character mode.

$\text{quan}(E, \alpha)$	$\text{quan}(E, 2\alpha)$	connection chain
0.46	0.73	e5 → ST → rs → e13 → zi → e14 → vm → e15
0.63	0.91	e5 → ST → rs → e13 → zi → e14
0.77	1.00	e5 → ST → rs → e13
0.48	0.61	e5 → ST → rs → e13 → zi → e14 → vm → e15
0.54	0.80	e5 → ST → rs → e13 → zi → e14
0.69	1.00	e5 → ST → rs → e13
0.34	0.57	e5 → ST → rs → e6 → zi → e7 → vm → e8
0.63	0.88	e5 → ST → rs → e6 → zi → e7
0.57	1.00	e5 → ST → rs → e6

region S of suspicious chain, based on the $\text{quan}(E, 2\alpha)$ value alone, would have the form

$$S(c) = \{\text{packet sequence } X : \text{quan}(E(X), 2\alpha(X)) \leq c\}, \tag{6}$$

where $0 \leq c \leq 1$ is an adjustable parameter. A larger value of c enlarges the rejection region S and generally results in more false alarms.

5.4 Recording on Originating Point

In this second set of experiments shown in Table 2, the recording machine ST acted as the originating point in all the chains. The length of the chain varied, but the ordering of machines remained fixed. The session transcript also varied substantially. To control for variability in network congestion, all the experiments were conducted in the same time frame, within two hours.

The bar graph in Figure 4 plots the $\text{quan}(E, 2\alpha)$ values versus the chain length for the 14 sessions in Table 2. On the whole, the quantile $\text{quan}(E, 2\alpha)$ decreases as the chain length increases. Deviations from the general trend are expected because these sessions do not have a common transcript.

In this set of experiments, not only does the chain length vary, but so does the session transcript. Using rejection region based on the $\text{quan}(E, 2\alpha)$ alone, as in Equation 6, gives mixed results. Figure 5 shows the ROC curves for three different tests. To test for more than two hops downstream, the rejection region of the form in Equation 6 provides a perfect test. To test for more than five hops downstream or to test for more than nine hops, however, the results are uniformly worse.

In test samples with variability in session transcript, the simple rejection region based on $\text{quan}(E, 2\alpha)$ alone still works reasonably well, especially in testing

Table 2. Quantile behavior under varying session transcripts but fixed topology. The 14 logs were all recorded from the original point ST. The session transcripts involved shell commands and more complicated commands within the EMACS editor. Each log had a different session transcript.

quan$(E, 2\alpha)$	connection chain
0.40	ST→e2→zi→e3→cp→e4→ls→sp→e6→cs→e7→xb→df→bs→e8
0.36	ST→e2→zi→e3→cp→e4→ls→sp→e6→cs→e7→xb→df→bs
0.28	ST→e2→zi→e3→cp→e4→ls→sp→e6→cs→e7→xb→df
0.39	ST→e2→zi→e3→cp→e4→ls→sp→e6→cs→e7→xb
0.42	ST→e2→zi→e3→cp→e4→ls→sp→e6→cs→e7
0.44	ST→e2→zi→e3→cp→e4→ls→sp→e6→cs
0.21	ST→e2→zi→e3→cp→e4→ls→sp→e6
0.68	ST→e2→zi→e3→cp→e4→ls→sp
0.57	ST→e2→zi→e3→cp→e4→ls
0.45	ST→e2→zi→e3→cp→e4
0.70	ST→e2→zi→e3→cp
0.62	ST→e2→zi→e3
0.92	ST→e2→zi
0.99	ST→e2

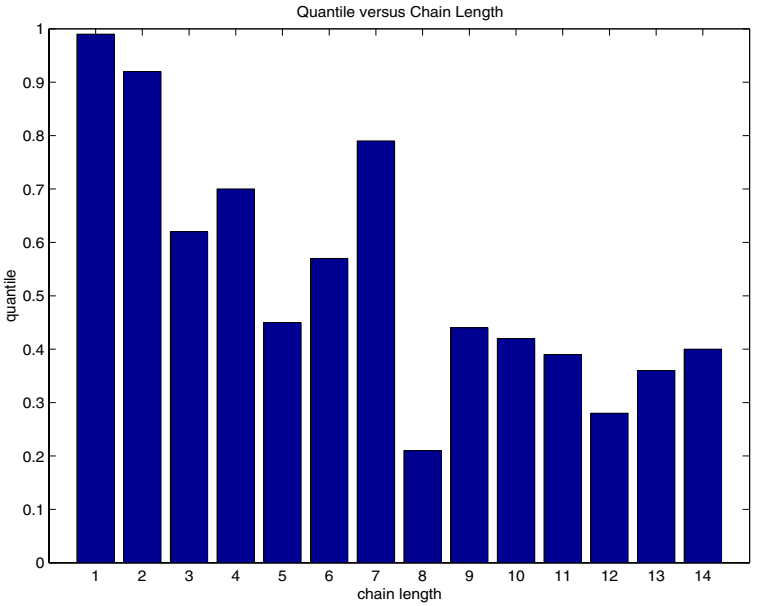

Fig. 4. Plot of quan$(E, 2\alpha)$ versus chain length.

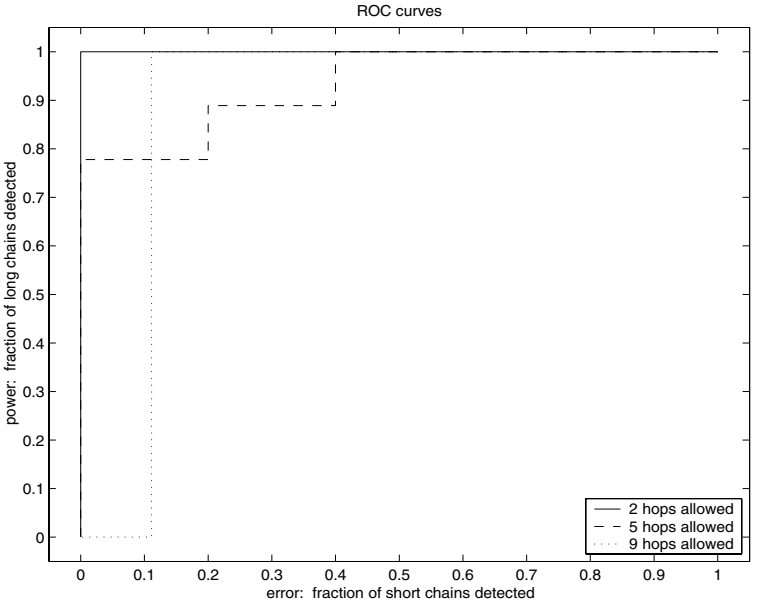

Fig. 5. ROC curves for rejection region $S(c)$.

for more than two hops downstream. As a test for more than five hops downstream or as a test for more than nine hops downstream, the performance of the simple rejection region based on $\text{quan}(E, 2\alpha)$ alone deteriorates.

When there are many hops downstream, the variability introduced by the machines downstream adds considerable complication. As expected, tolerance for a big number of downstream hops is more difficult to implement than tolerance for a small of downstream hops. In practice, the low tolerance policy that rejects sessions with more than two hops downstream is more useful and realistic.

6 Discussion

The rejection region in Equation 6 classifies an outgoing connection as having too many hops downstream if the outgoing connection has a packet sequence X whose $2\alpha(X)$ value is too small compared to the gaps in set $E(X)$. To test for an outgoing connection with more than two hops, experiments indicate that the using cut-off parameter $c = 0.9$ would give reasonably accurate results without high false alarms.

The experiments used a machine on the Stanford University network as the logging machine. Although a wide range of experiments under different settings showed that the cut-off parameter $c = 0.9$ performed well, machine and network properties can vary considerably. For best results, each deployment can train on its own machines and networks to determine the optimal cut-off to meet prescribed requirements for accuracy and precision.

6.1 Accuracy and Precision

Most intrusion-detection techniques suffer from too many false positives [4]. Because there is a wide range of network attacks, general intrusion-detection systems are difficult to design. By narrowing the scope of detection to special types of attacks, the number of false attacks can be lowered considerably, [2] for example.

The problem of connection chains is well-defined and narrow enough to be considered as a specific attack. Yet, earlier techniques [8,10] for detecting stepping stones identify many harmless, short chains common in practice. Echo-delay comparison proposed here specifically addresses these logical false positives and detects only connection chains with many hops downstream.

On the other hand, echo-delay comparison does not detect the number of *upstream* hops in connection chain. From the viewpoint of the recording machine, only the nearest upstream client is known. If the recording machine is a stepping stone in a long connection chain but is only one single hop away from the final victim, then the session will not trigger any warning because there are not many hops *downstream*. This logical false negative will be addressed in future work. Because delayed acknowledgments are sent in both directions, an extension of the current work may prove useful in detecting many hops upstream.

6.2 Hacker Intervention

In the ever-raging battle between hackers and intrusion-detection systems, intelligent hackers always search for new ways to elude detection. Echo-delay comparison and previous approaches [8,10] are all susceptible to hacker intervention. In fact, in a theoretical sense [6], any intrusion detector relying solely on network logs can be circumvented by carefully manipulating network signals.

The time gap between the client request and the delayed acknowledgment of the nearest server provides an overestimate of the travel time for one hop downstream. Since not all downstream hops are equally noisy, two problems may arise. A fast connection between the client and the nearest server may over-amplify the slow hops downstream. This configuration does not benefit the hacker trying to avoid detection though.

Likewise, a slow connection between the client and the nearest server may mask the fast hops downstream. If the detector is used at its optimal settings, to detect more than two hops downstream, then there is minimal leeway for hiding many hops behind the slow first connection. Hiding many quick connections on machines within close proximity would defeat the purpose of using a connection chain.

A knowledgeable hacker can manipulate the network signals. To elude detection, the hacker may delay and suppress the delayed-acknowledgment signal and the reply-echo signal. Because the analysis in this paper uses aggregate statistics, targeting a few signals will not thwart the detector. Manipulating many signals simultaneously without adversely affecting the dynamics of the connection chain would be difficult even for the skilled hacker.

7 Summary

Echo-delay comparison monitors an outgoing connection to estimate two important time gaps. First, the gap between the client request and the server delayed acknowledgment estimates the round-trip travel time between the client and the server. Second, the gap between the client request and the server reply echo estimates the how far downstream the final victim is away. Together these two time gaps provide a simple test to identify a session whose final victim is many hops downstream.

Unlike previous approaches for detecting stepping stones, echo-delay comparison works in isolation, without matching for similar sessions on the same connection chain. Moreover, this new strategy will allow benign, short connection chains common in practice. Echo-delay comparison makes use network signals found in interactive terminal sessions, such as telnet, rlogin, and secure shell. Experiments demonstrate that the technique is effective under a wide range of conditions and performs especially well in identifying sessions with more than two hops downstream.

Acknowledgments. This research project was funded in part by the US Department of Justice grant 2000-DT-CX-K001. Jeffrey D. Ullman of the Stanford University Computer Science Department introduced the author to the field of intrusion detection and offered invaluable advice throughout the past year. Jerome H. Friedman of the Stanford University Statistics Department provided important feedback in several discussions. The author is grateful for their help and extends his delayed acknowledgment, long overdue.

References

1. Stefan Axelsson. "Intrusion Detection Systems: A Survey and Taxonomy." Technical Report 99-15, Department of Computer Engineering, Chalmers University, March 2000.
2. Robert K. Cunningham, et al. "Detecting and Deploying Novel Computer Attacks with Macroscope." *Proceeding of the 2000 IEEE Workshop on Information Assurance and Security.* US Military Academy, West Point, NY, 6–7 June, 2001.
3. Harold S. Javitz and Alfonso Valdes. "The NIDES Statistical Component: Description and Justification." Technical report, Computer Science Laboratory, SRI International. Menlo Park, California, March 1993.
4. Richard P. Lippmann, et al. "Evaluating Intrusion Detection Systems: The 1998 ARPA Off-Line Intrusion Detection Evaluation." *Proceedings of DARPA Information Survivability Conference and Exposition.* DISCEX '00, Jan 25–27, Hilton Head, SC, 2000. `http://www.ll.mit.edu/IST/ideval/index.html`
5. Peter G. Neumann and Phillip A. Porras. "Experience with EMERALD to Date." *1st USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 73-80. Santa Clara, California, USA, April 1999.
6. Thomas H. Ptacek and Timothy H. Newsham. "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection." Secure Networks, Inc., January 1998. `http://www.aciri.org/vern/PtacekNewsham-Evasion-98.ps`

7. Martin Roesch. “Snort: Lightweight intrusion detection for networks.” 13th Systems Administration Conference (LISA’99), pages 229–238. USENIX Associations, 1999.
8. Stuart Staniford-Chen and L. Todd Heberlein. “Holding Intruders Accountable on the Internet.” *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 39–49. Oakland, CA, May 1995.
9. W. Richard Stevens. *TCP/IP Illustrated Volume 1: The Protocols*. Addison-Wesley: Reading, Massachusetts, 1994.
10. Yin Zhang and Vern Paxson. “Detecting stepping stones.” *Proceedings of 9th USENIX Security Symposium*. August 2000.