

Stepping-Stone Detection Via Request-Response Traffic Analysis

Shou-Husan Stephen Huang¹, Robert Lychev², and Jianhua Yang³

¹ Department of Computer Science, University of Houston,
4800 Calhoun Rd., Houston, TX 77004, USA
shuang@cs.uh.edu

² Department of Computer Science, University of Massachusetts Amherst,
Amherst, MA 01003, USA
rlychev@cs.umass.edu

³ Department of Mathematics & Computer Science, Bennett College,
900 E. Washington St., Greensboro, NC 27401, USA
jhyang@bennett.edu

Abstract. In this paper, we develop an algorithm that may be used as a stepping-stone detection tool. Our approach is based on analyzing correlations between the cumulative number of packets sent in outgoing connections and that of the incoming connections. We present a study of our method's effectiveness with actual connections as well as simulations of time-jittering (introduction of inter-packet delay) and chaff (introduction of superfluous packets). Experimental results suggest that our algorithm works well in the following scenarios: (1) distinguishing connection chains that go through the same stepping stone host and carry traffic of users who perform similar operations at the same time; and (2) distinguishing a single connection chain from unrelated incoming and outgoing connections even in the presence of chaff. The result suggests that time-jittering will not diminish our method's usefulness.

1 Introduction

The study of detection and/or prevention of network-based attacks requires much attention as perpetrators are becoming more and more capable of compromising much of critical information infrastructure that we so highly depend on. Network-based attacks can be either interactive, where a perpetrator is interested in stealing information from another member of the network, or non-interactive, where a perpetrator's goal is to trigger a malicious software or perform a denial-of-service attack on another member of the network. Attackers can use a number of techniques to avoid revealing their identification and location. Two of the most-commonly used evasion measures include IP-spoofing and the construction of stepping-stone chains. The latter involves an intruder connecting to a victim indirectly through a sequence of hosts called stepping-stones. Although, some work has already been done to show a number of effective techniques for tracing spoofed traffic [4, 5, 7, 8], effective measures for tracking stepping-stone attacks are yet to be found. The focus of our research is to study a

connection-chain detection scheme that could help us address the stepping-stone detection problem, a portion of the stepping-stone attack tracking problem, in interactive attacks.

To understand why stepping-stone detection may be an important subject to study, consider the following scenario. Machine V is discovered to be a victim of an interactive attack whose immediate source was found to be machine S . Shutting off S from the network is effective in stopping the attack, but it does not do anything to ensure that the adversary A is caught, since S could be just the immediate stepping-stone used by A to indirectly connect to V . However, with the ability to correctly determine whether S is a stepping-stone or not, one can either go upstream along the chain to discover other stepping stones and/or catch the perpetrator, or simply shut down S if it is not a stepping stone (in which case it must be A). Even when it is not known that an attack is launched, being able to correctly determine whether any member of the network is a stepping-stone should allow for an effective way of policing interactive attacks. Stepping-stone detection problem is a useful subject to study, but it must be noted that just having the capability of even perfect stepping-stone detection is not enough to solve the stepping-stone attack tracking problem. As explained in [13], to track stepping-stone attacks one also needs to have correct methods of serializing stepping-stones into a connection chain.

Much research has already been done in this area, and, ultimately, all established techniques of identifying a particular host as a stepping-stone rely on identifying a connection-chain based on strong correlations between that host's incoming and outgoing traffic. Such correlations can be based on the log-in activity [6, 9], packet content [10, 11], periodicity of network activity [16], the timing properties [12, 15], and the packet frequency of the connections [1]. The first two techniques are not practical because, respectively, it is conceivable that hackers should be able to forge authentication sessions, and, since most users use SSH instead of Telnet, it is not clear how to correlate traffic that is encrypted as it is passed from host to host. A hacker can easily countermeasure correlation techniques such as the one described in [16] by introducing random time delays in between individual and/or collections of packets—jittering. It was shown in [3] that, in principle, there is no effective way for an adversary to avoid timing-based detection techniques such as the ones described in [12, 15]. However, this is true only under the assumption that the adversary's time-jittering of the packets is independently and identically distributed and that the connection is long-lived. Also, effectiveness of timing-based detection methods is likely to diminish in the presence of chaff—superfluous packets introduced at various stepping-stones. Although techniques based on finding correlations between packet frequency of incoming and outgoing traffic, as presented in [1], were shown to be successful against jittering without the assumptions that were necessary in [3], these techniques do not perform well with chaffed traffic. Several effective algorithms to detect stepping-stone chains with chaff and jittering have been proposed in [17], but all of these methods require a significant amount of intercepted packets in order to ensure small false positive and negative rates. A testbed that may be very useful in testing various stepping-stone detection mechanisms in different scenarios was proposed in [14].

Section 2 is dedicated to describing our approach, Section 3 explains our experimental setup and methodology, Section 4 presents and analyzes results we obtained

from various experiments, and Section 5 wraps up this paper with a discussion of conclusions and possible directions for future work.

2 Technical Method

Our research is primarily inspired by algorithms discussed in [1]. However, in [1] only correlations between streams with the same direction were discussed, so only the observation of traffic that is relayed from stepping-stone to stepping-stone is required by techniques they proposed. We want to check whether our connection-chain detection algorithm, that focuses on determining frequency relationships between request and response streams, could be used to design a stepping-stone detection algorithm that yields results comparable to, with respect to false positive and negative rates, what has been achieved in [1, 17], while requiring less packets to observe. Given that some machine S is being used as a stepping-stone by some adversary A , the challenge of detecting a connection-chain lies in finding the exact S 's incoming-outgoing connection pair that is carrying traffic relevant to A 's stepping-stone attack.

2.1 The Basics

Our algorithm is based on measuring correlations of the outgoing streams of outgoing connections to the outgoing streams of incoming connections. Throughout the rest of this paper we will refer to the former as the SEND and the latter as the ECHO. Traffic in both directions should be monitored at stepping-stone.

Our hypothesis is that for a SEND-ECHO pair that belongs to a real connection chain, the frequency with which packets leave a stepping-stone in the ECHO stream is a function of the frequency with which packets leave a stepping-stone in the SEND stream. This is based on the fact that interactive attacks consist of adversaries obtaining information from the victims for every command the former sends. We study the relationship of ECHO - SEND versus ECHO + SEND (the difference and the sum of the number of packets in the ECHO stream and the number of packets in the SEND stream, respectively) to see how correlated a particular ECHO stream is to a particular SEND stream. Based on our hypothesis, we assume that the relationship between ECHO - SEND and ECHO + SEND should be linear. Thus we are able to analyze the packet frequency relationship between request and response traffic for a particular incoming-outgoing connection pair independently of other connections, where we can treat ECHO + SEND as the time, which is actually independent of real time and inter-packet delay, and ECHO - SEND as the variable of interest. Time independence gives us an advantage over the time-jittering detection evasion that will be discussed in Section 4.2. We suspect that in ECHO - SEND vs. ECHO + SEND space, SEND-ECHO pair that corresponds to a real connection chain should yield a curve that resembles a smooth line more than all curves that correspond to other SEND-ECHO pairs. We use this to find connection chains. Also, if a computer is discovered to have a SEND-ECHO pair that satisfies a particular margin of linearity, there is a high probability that it is being used as a stepping-stone. This assumption may be used for stepping-stone detection. Method of measuring linearity is explained in Section 2.2.

2.2 Computing Linearity of a Curve

We measure linearity of a curve by calculating the average square distance of that curve from its linear fit. According to our assumption, the curve with the smallest average square distance from its linear fit should correspond to the SEND-ECHO pair of the real connection chain. Linear fit $y = mx + b$ is calculated via standard linear regression method, where $x = \text{ECHO} + \text{SEND}$, and $y = \text{ECHO} - \text{SEND}$ [2].

3 Experimental Setup

3.1 Experiment of Worst Case Scenario

The first experiment targeted the worst-case scenario: all participants login onto different hosts via SSH through a single stepping stone, located at UH (University of Houston), from three different hosts at the same time and perform the same tasks. The point of studying the worst-case scenario was to see if we can distinguish connection chains that go through the same stepping-stone and carry traffic of users who perform similar operations at the same time. We conjecture that if it is possible to correctly distinguish connection chains in such a situation, then our procedure should work very well in situations where there is only one connection chain and many other completely unrelated incoming and outgoing connections. The former can happen if an adversary makes loops in his/her connection chain before reaching the victim's machine for the purposes of stepping-stone-detection evasion. Two types of experiments were done this way: typing and secret-stealing. The stepping-stone computer was running our software that was monitoring the streams of interest and recording packets in those streams. The following are the connection chains:

Participant A: home computer (SBC) \rightarrow UH stepping stone \rightarrow UH1 \rightarrow Mexico

Participant B: UH2 \rightarrow UH3 \rightarrow UH stepping stone \rightarrow UMASS Amherst

Participant C: UH4 \rightarrow UH stepping stone \rightarrow Texas A&M University

For the first three trials all participants were to type identical texts simultaneously. The last trial involved all three individuals typing different texts not simultaneously for different amounts of time.

The secret-stealing experiments consisted of the participants searching for a secret file on a victim computer by going through a number of directories that contains fake files. The test directory, consisting of secret directories/files was prepared in advance. The secret file was copied onto the attacker's machine upon discovery. The following are the connection chains:

Participant A: UH1 \rightarrow UH stepping stone \rightarrow UH5 \rightarrow Mexico

Participant B: UH2 \rightarrow UH3 \rightarrow UH stepping stone \rightarrow UMASS Amherst

Participant C: UH4 \rightarrow UH stepping stone \rightarrow Texas A&M University

3.2 Experiments with a Single Connection Chain

The case where a stepping-stone machine had only one connection-chain and other unrelated connections was addressed in the last experiment with the following connection chain:

Participant A: UH2 → UH stepping stone

Participant B: UH stepping stone → UMASS Amherst

Participant C: UH4 → UH stepping stone → Texas A&M University

Participant A connected to the stepping stone at UH and was writing a Hello World application. Participant B connected from the stepping stone at UH to UMASS and was copying electronic copies of some files. Finally, Participant C was performing a secret-stealing attack. The results of this experiment were also chaffed via the second chaff technique and are discussed in Section 4.4. It is reasonable to assume that in real-life computer networks it is very unlikely for a single computer to have more than one connection chains that go through it. Even if a perpetrator decides to make loops, as mentioned in Section 3.1, he/she will not be likely to loop through every stepping-stone because this is likely to slow down his/her attack by a large margin. With this in mind, the point of performing the experiment described in this section is to model something that is more likely to happen in real life situations.

3.3 Time-Jittering and Chaff

We studied time-jittering and chaff by changing the results that we obtained from regular experiments and analyzing the changed data the same way as the regular data.

Time-jittering perturbation was introduced as an addition of time extensions, chosen uniformly between 0 and some pre-specified limit, to the time stamps of packet records. The original order of packets within a stream is preserved. Every packet had probability of .5 to be thus time-jittered.

For every stream, SEND and ECHO, of every connection the chaff perturbation is introduced as an addition of packets, whose amount is limited by a pre-specified margin, to the original stream. Two different methods were performed. The first method consisted of generating a stream of superfluous packets, whose inter-packet delay is a random variable with a uniform distribution in the interval of 100-900 thousand microseconds, and merging this stream with an actual stream of packets that was recorded during the experiment. The second technique consisted of inserting a random number of superfluous packets, ranging from 1-20, into pseudo-randomly-chosen, with probability of 0.1, inter-packet time intervals of the original packet stream. For both methods, such parameters represent the worst-case scenario where the most chaff is introduced. Experiments performed with other chaff limits are not discussed in this paper.

4 Analysis and Discussion

As the reader will see, our main assumption that the relationship between ECHO - SEND and ECHO + SEND is close to linear is justified by the fact that correlation coefficient r [2] for curves that correspond to real connection chains are all above 0.95 when no stream is time-jittered or chaffed. It makes sense that r of curves that correspond to experiments without time-jittering and chaff are positive as our study is focused on interactive attacks, where the adversary gets back more packets from the victim machine than he/she sends to the latter. However, this is not always the case for experiments with time-jittering and chaff simulations.

4.1 Basic Experiments

For both types of experiments described in Section 3.1, without time-jittering and chaff the packet data of ECHO stream of a particular participant yields the smoothest curve when related to the packet data of SEND stream of that participant. This can be seen just by looking at the curves on the plots of $ECHO - SEND$ versus $ECHO + SEND$ of typing and secret stealing experiments we took at the beginning of this project (see Fig. 1a and Fig. 1b). On all the figures' legends ECHO and SEND streams

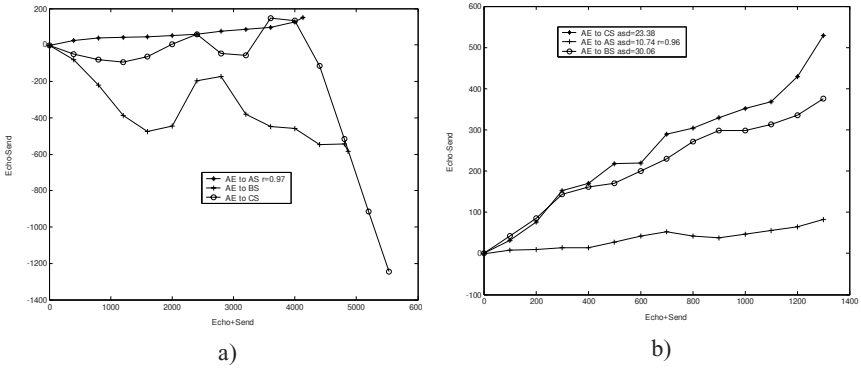


Fig. 1. Correlations of the ECHO stream of a particular participant to the SEND streams of all the participants in a) the typing experiment; b) the secret-stealing experiment

are referred to as E and S, respectively. All legends, except for Figures 1 and 3, show the average square distance (*asd*) of a curve from its linear fit (for each curve) and r (for the curve that corresponds to the real connection chain). Data obtained from the typing experiment was not quantitatively analyzed as this experiment does not really model a real interactive attack, but basic qualitative analysis should be enough here to obtain the correct result. Data obtained from experiment shown on Fig. 1b was quantitatively analyzed with procedure described in 2.2. Overall, experiments shown in Fig. 1a and Fig. 1b indicated that even when participants perform the same set of operations at the same time it is possible to pair each SEND stream with its complementary ECHO stream correctly using procedure described in 2.2.

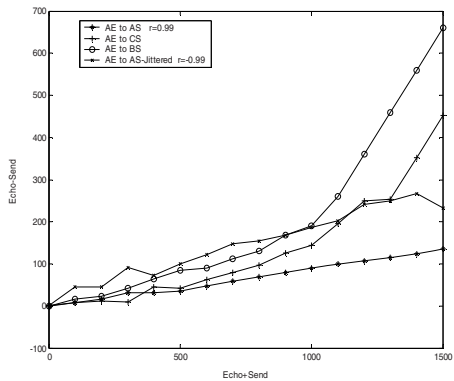


Fig. 2. Correlations of the ECHO stream of a particular participant to the SEND streams of all the participants in the secret-stealing experiment with a time-jittered simulation performed

4.2 Time Jittering Simulations

We mostly studied data that resulted from time-jittering the SEND streams of various connections where no time extension exceeded 200 thousand microseconds. After undergoing perturbations, every SEND-stream-packet-record vector was merged with data of various ECHO streams. After time-jittering, while the order of SEND packets with respect to each other was preserved, the order of SEND packets with respect to ECHO packets was not. This can be seen in Fig. 2. The ends of these curves also exhibit the shortcomings of our simulation.

We claim that the results that we might obtain once we solve the shortcomings of our current time-jittering simulation are not going to be very interesting. We claim so because in order for time-jittering to really affect our results, the order of SEND packets with respect to the ECHO packets has to be significantly disrupted. However, because some ECHO packets can come only after their corresponding SEND packets and vice versa, this disruption is not expected to be significant.

4.3 Chaff Simulations

We looked at data that resulted from chaffing the SEND stream, the ECHO stream and both streams of various connections. After undergoing such perturbations, every vector with perturbed data was merged with data of various ECHO streams. We assume that the adversary can chaff only his own traffic.

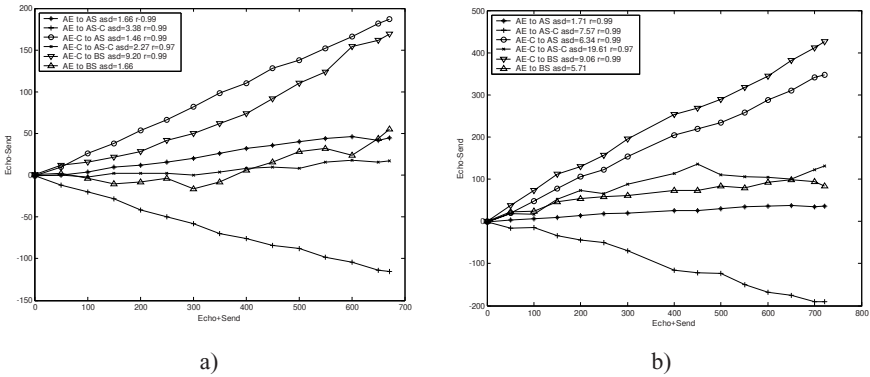


Fig. 3. Correlations of the ECHO stream of a particular participant to the SEND streams of all the participants in the secret-stealing experiment with a) the 1st chaff technique; b) 2nd chaff technique

As can be seen from Fig. 3a in which ‘-C’ indicates ‘Chaffed’ (the same for other figures), the first chaff technique does not introduce much noise to the data; it stretches the curve a bit. When only the SEND stream is chaffed, the curve has a negative slope. When only the ECHO stream or when both streams is/are chaffed, the curve has a positive slope. As can be seen from Fig. 3b, the second chaff technique is more aggressive than the first one and it introduces significant noise to the data. This is why when the second chaff technique is used, we cannot always distinguish the

curve that corresponded to the real connection by the means described in section 2.2. This is not discouraging because this experiment models an unrealistically difficult situation when users perform the same task at the same time and at least one of the streams is chaffed. It is interesting to note that the second technique may be useful to the adversary for stepping-stone detection evasion.

4.4 Experiment with a Single Connection Chain

The goal of our last experiment was to see if we can distinguish a chaffed (via the second chaff technique) connection chain from unrelated connections. As can be seen from Fig.4, it is possible to distinguish participant C's connection chain when its SEND and ECHO streams are chaffed, but not when both SEND and ECHO streams are chaffed. Curves AE to CS and AE to CS-chaffed exhibit rather weak correlations; this is because they correspond to unrelated connections. Such results are encouraging as they show that even though our procedure may not work very well in the worst-case scenario, it should work fine in the case of a single connection chain unless the hacker chaffs both the ECHO and the SEND streams via the second chaff technique.

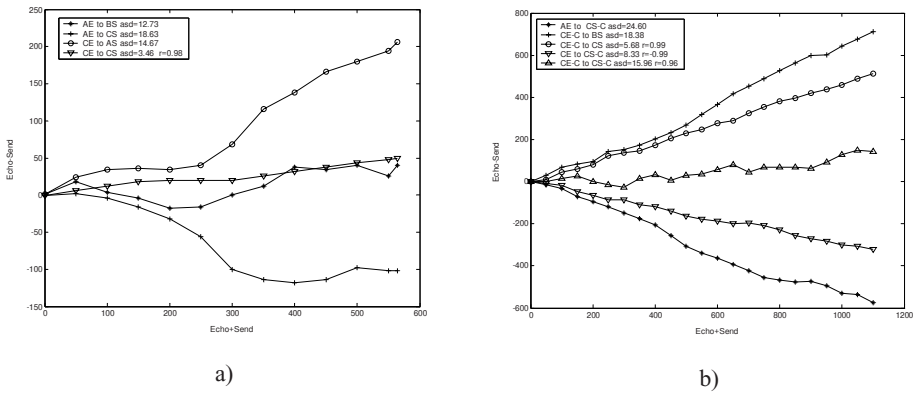


Fig. 4. Correlations of the ECHO stream of a particular participant to SEND streams of participants B and C in the experiment with only a single connection chain. a) without chaff; b) with chaff

5 Conclusions and Future Work

Even though more experimentation is needed before any definitive claims could be made regarding our procedure for finding connection chains, based on our experiments we can with confidence say that procedure described in Section 2.2 always works in distinguishing connection chains that go through the same stepping-stone and carry traffic of users who perform similar operations at the same time when neither time-jittering nor chaff is introduced by the adversary to his/her traffic. Our procedure works well when the first chaff technique is used. The second chaff method is more aggressive, and, therefore, may qualify as a good method for hackers

to use for stepping-stone-detection evasion. Our procedure works well in distinguishing a single connection chain from unrelated incoming and outgoing connections, even when chaff is introduced via the second technique, unless both streams are chaffed.

In the future, we would like to test our connection-chain detection mechanism when chaff and time-jittering are introduced into real-life connections as opposed to simulating these stepping-stone detection evasion tools with data obtained from regular experiments. It would be interesting to address the following questions. How well does our connection-chain method work when more than one user's stream is chaffed and/or when streams are both time-jittered and chaffed? Are there any other methods of measuring linearity of a curve that could yield better results than our procedure with respect to connection-chain detection? How well do other stepping-stone detection mechanisms work when the second chaff technique is used? How successful are our connection-detection procedure and other stepping-stone detection methods when introduction of chaff is not based on probability distributions that are i. i. d.? Ultimately, we would like to design a stepping-stone detection mechanism that would efficiently use our connection-chain detection method and experimentally and/or formally compare the former to other stepping-stone detection mechanisms with respect to running-time complexity, false positive and negative rates and the number of packets required to observe.

Acknowledgement

This project is supported in part by a grant from NSF (SCI-0453498) DoD's ASSURE Program. The authors would like to thank Scott Nielsen, Mykyta Fastovets for their participation in the experiments.

References

1. Blum, A., Song, D., Venkataraman, S.: Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 258–277. Springer, Heidelberg (2004)
2. Brunk, H.D.: An Introduction to Mathematical Statistics, Ginn and Company (1960)
3. Donoho, D., Flesia, A.G., Shankar, U., Paxson, V., Coit, J., Staniford, S.: Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 45–59. Springer, Heidelberg (2002)
4. Duwairi, B., Chakrabarti, A., Manimaran, G.: An Efficient Probabilistic Packet Marking Scheme for IP Traceback. In: Mitrou, N.M., Kontovasilis, K., Rouskas, G.N., Iliadis, I., Merakos, L. (eds.) NETWORKING 2004. LNCS, vol. 3042, pp. 1263–1269. Springer, Heidelberg (2004)
5. Goodrich, M.T.: Efficient Packet Marking for Large-Scale IP Traceback. In: Proc. of ACM CCS '02, Washington, DC, USA, pp. 117–126 (2002)
6. Jung, H.T., Kim, H.L., Seo, Y.M., Choe, G., Min, S.L., Kim, C.S., Koh, K.: Caller Identification System in the Internet Environment. In: Proc. of 4th USINEX Security Symposium, Santa Clara, CA, USA, pp. 69–78 (1993)

7. Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Practical Network Support for IP Traceback. In: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Stockholm, Sweden, pp. 295–306 (2000)
8. Song, D., Perrig, A.: Advanced and Authenticated Marking Scheme for IP Traceback. In: Proc. of IEEE INFOCOM, Anchorage, AL, USA, pp. 878–886 (2001)
9. Snapp, S., et al.: DIDS, (Distributed Intrusion Detection System) – Motivation, Architecture and Early Prototype. In: Proc. of 14th National Computer Security Conference, Columbus, OH, USA, pp. 167–176 (1991)
10. Staniford-Chen, S., Heberlein, L.T.: Holding Intruders Accountable on the Internet. In: Proc. of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 39–49 (1995)
11. Wang, X., Reeves, D.S., Wu, S.F., Yuill, J.: Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework. In: Proc. of 16th International Conference on Information Security, Paris, France, pp. 369–384 (2001)
12. Wang, X., Reeves, D.S.: Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Inter-packet Delays. In: Proc. of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA, pp. 20–29 (2003)
13. Wang, X.: The Loop Fallacy and Serialization in Tracing Intrusion Connections through Stepping Stones. In: Proc. of the ACM Symposium on Applied Computing, Nicosia, Cyprus, pp. 404–411 (2004)
14. Xin, J., Zhang, L., Aswegan, B., Dickerson, J., Daniels, T., Guan, Y.: A Testbed for Evaluation and Analysis of Stepping Stone Attack Attribution Techniques. In: Proc. of the 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, Barcelona, Spain (2006)
15. Yoda, K., Etoh, H.: Finding a Connection Chain for Tracing Intruders. In: Proceedings of 6th European Symposium on Research in Computer Security, Toulouse, France, pp. 191–205 (2000)
16. Zhang, Y., Paxson, V.: Detecting Stepping Stones. In: Proc. of the 9th USENIX Security Symposium, Denver, CO, USA, pp. 171–184 (2000)
17. Zhang, L., Persaud, A.G., Johnson, A., Guan, Y.: Detection of Stepping Stone Attack under Delay and Chaff Perturbations. In: Proc. of 25th IEEE International Performance Computing and Communications Conference, Phoenix, AZ, USA (2006)