Stepping-Stone Detection via Request-Response Traffic Analysis[1]

Shou-Hsuan Stephen Huang[2]
Robert Lychev[3]
Jianhua Yang[4]

Computer Science Department
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

UH-CS-06-14
December 4, 2006

Keywords: Network security, intrusion detection, network traffic, stepping-stone, clustering, network request-response.

Abstract

In this paper, we propose a new method to detect stepping-stone intrusion by computing the linearity between the numbers of send packets and the numbers of echo packets. The linearity of two relayed connections is better than that of two non-relayed connections. We develop a connection-chain detection procedure that may be used as a stepping-stone detection tool. Our procedure is based on analyzing correlations between the frequencies at which cumulative numbers of packets are sent in outgoing connections and at which packets are sent in that of the incoming connections. The experiment and simulation results show this method can resist intruders' time and chaff evasion better than other approaches.

[1] This project is supported in part by an REU grant from NSF (SCI-0453498) and DoD's ASSURE Program. The authors would like to thank Scott Nielsen, Mykyta Fastovets for their participation in the experiments.
[2] *Department of Computer Science, University of Houston, E-mail: shuang@cs.uh.edu*
[3] *Department of Computer Science, University of Massachusetts, E-mail: rlychev@umass.edu*
[4] *Department of Mathematics and Computer Science, Bennett College, E-mail: jhyang@bennett.edu*

Stepping-Stone Detection via Request-Response Traffic Analysis

Shou-Hsuan Stephen Huang[1], Robert Lychev[2], Jianhua Yang[3]
[1]*Department of Computer Science, University of Houston*
E-mail: shuang@cs.uh.edu
[2]*Department of Computer Science, University of Massachusetts*
E-mail: rlychev@umass.edu
[3]*Department of Mathematics and Computer Science, Bennett College*
E-mail: jhyang@bennett.edu

Abstract

In this paper, we propose a new method to detect stepping-stone intrusion by computing the linearity between the numbers of send packets and the numbers of echo packets. The linearity of two relayed connections is better than that of two non-relayed connections. We develop a connection-chain detection procedure that may be used as a stepping-stone detection tool. Our procedure is based on analyzing correlations between the frequencies at which cumulative numbers of packets are sent in outgoing connections and at which packets are sent in that of the incoming connections. The experiment and simulation results show this method can resist intruders' time and chaff evasion better than other approaches.

Keywords

Network security, intrusion detection, network traffic, stepping-stone, clustering, network request-response.

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, 3, 6, 7], effective measures for tracking stepping-stone attacks are yet to be found. The focus of our research is to address the stepping-stone detection problem, a portion of the stepping-stone attack tracking problem, in interactive attacks.

The stepping-stone detection problem can be stated as follows: when an adversary, Eve, lounges an interactive attack on a victim, Bob, by forming a stepping-stone chain via other members of the network, the challenge is to determine whether any given member, Alice, belongs to that chain while provably minimizing time, false positive and false negative rates. Please note, while alias Eve refers to a real person behind a specific machine, aliases Alice and/or Bob may refer to stepping-stone and/or victim machines only. This is because there is no need for anyone to be using the machines in the stepping-stone chain while the attack is taking place, although the machines do need to be turned on. Consider the following scenario. Bob was discovered to be a victim of an interactive attack whose immediate source was found to be machine *C*. Simply shutting off *C* from the network is effective in stopping the attack, but it does not do anything to ensure that the adversary Eve is caught, since *C* could be just the immediate stepping-stone, Alice, used by Eve to indirectly connect to Bob. However, with the ability to correctly determine whether *C* is a stepping-stone or not, we can either go up the chain to discover other stepping-stones and/or catch Eve, or simply shut down *C* if it is not a stepping-stone (in which case it must be Eve). In fact, 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 very interesting and 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 [12], 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 par-

ticular host as a stepping-stone rely on finding strong correlations between that host's incoming and outgoing traffic. Such correlations can be based on the log-in activity [5, 8], packet content [9, 10], periodicity of network activity [15], the timing properties [11, 14], and the packet frequency of the connections [1]. The first two techniques are not practical because, respectively, it is conceivable that Eve 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. Eve can easily countermeasure correlation techniques such as the one described in [15] by introducing random time delays in between individual and/or collections of packets—jittering. It was shown in [2] that, in principle, there is no effective way for Eve to avoid timing-based detection techniques such as the ones described in [11,14]. However, this is true only under the assumption that Eve's jittering of the packets is independently and identically distributed and that the connection is long-lived. Also, timing-based detection approaches are prone to chaff—introduction of superfluous packets 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 [2], 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 [16], but all of these methods require a significant amount of intercepted packet in order to ensure a small false positive and negative rate. It is yet to develop a technique that is provably successful against jitter and chaff with a reasonable false positive rate while requiring the observation of a number of packets that is within practical limits. Our goal is to develop such a technique.

The rest of this paper is arranged as the following. Section 2 discusses our technical method to detect stepping-stone. Section 3 gives the experimental setup. In Section 4, we analyze the experimental results and present some discussions. Finally, we summarize the whole work and present the future work in Section 5.

2. Technical Method

Our research is primarily inspired by algorithms discussed in [1, 19, 20]. As mentioned above techniques that were discussed in [1] yielded pretty good results against time jittering. The idea is that sufficiently long strong correlations between the frequencies at which packets are sent in outgoing connections and at which packets are received in incoming connections should imply high probability that such connection pairs are stepping-stone witness. Vice versa should hold as well. However, in this paper only correlations between streams with the same direction were discussed. In other words, only the observation of traffic that is relayed from stepping-stone to stepping-stone is required by technique they propose (see Figure 1). We want to check whether our stepping-stone detection algorithm that focus primarily on determining such frequency relationships between request and response streams yields results comparable to, with respect to false positive and negative rates, what has been achieved in [1], while requiring less packets to observe.

2.1 The Basics behind Our Approach

Our algorithm requires the observation of traffic that holds Eve's commands as well as the traffic that carries Bob's responses to Eve's commands while going through the stepping-stone Alice (see Figure 2). In other words, we want to measure correlations of outgoing stream of outgoing connections and outgoing stream of incoming connections. Throughout the rest of this paper we will refer to the former as the SEND and the latter as the ECHO (see Figure 3).

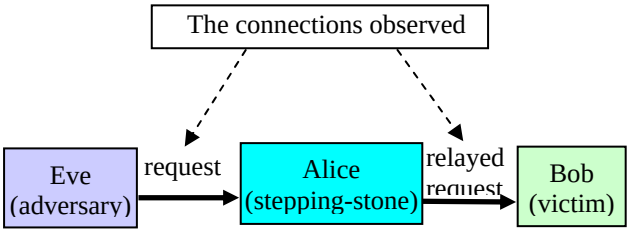


Figure 1: Traffic is monitored in the same direction only

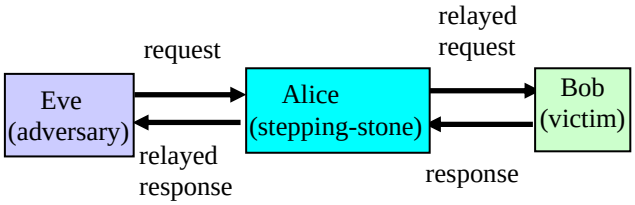


Figure 2: Traffic is monitored in both directions

Figure 3: SEND and ECHO streams are monitored

Our hypothesis is that the 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 linearly related to the frequency with which packets leave a stepping-stone in the SEND stream. Therefore, if a computer has a SEND-ECHO pair that satisfies a particular margin of linearity, there is a high probability that it is a stepping-stone. Our hypothesis is based on the fact that interactive attacks consist of adversaries obtaining information from the victims for every command the former send. Thus, the faster the adversary is sending the commands, the faster the victim is responding to those commands.

In order to see how correlated a particular ECHO is to a particular SEND stream we can study the data by relating ECHO – SEND versus ECHO + SEND (respectively, the difference and the sum of the number of packets in the ECHO stream and the number of packets in the SEND stream). This allows us to look at the packet frequency relationship between request and response traffic, where we can treat ECHO + SEND as the time and ECHO – SEND as the variable of interest. In this space, SEND-ECHO pair that belongs to a real connection chain should yield a curve that strongly resembles a line and is relatively smooth. The latter and the former properties may be quantified by measuring the correlation coefficient and the average distance of the curve from its linear fit respectively. It should be noted that we cannot expect the ideal case which would yield a truly linear curve because it cannot be predicted what the victim's responses in the ECHO streams may be. The quantity of the packets in this stream may vary greatly.

2.2 Computing Correlation Coefficient

The two most popular methods for obtaining *r*, which is the correlation coefficient, are the Spearman's Rank Correlation Coefficient and the Product-Moment Correlation Coefficient. However, we use Excel's CORREL() function which takes as input two arrays of equal length and outputs a correlation coefficient *r* regarding the input arrays as a number between -1 and +1. If $r = -1$, then there is a perfect negative correlation; if $r = 1$, then you have a perfect positive correlation. If $r = 0$, then the correlation is zero and there is no relationship between the variables. Here is how Excel computes *r* [17]:

1. Transform the values in each of the arrays of data into z-scores. The latter is a measure of how far any particular value is from the mean of the entire array in units of standard deviations.
2. Multiply together the corresponding z-scores in each array.
3. Sum all the results produced by step 2.

Divide the result obtained in step 3 by the total number of pairs of values in the input arrays. The average of the figures obtained in step 2 is thus obtained. This process will always produce a number between -1.00 and +1.00.

2.3 Measuring the Smoothness of a Curve

As stated above, the smoothness of a curve can be measured by calculating the average distance of the curve from its linear fit. The following formula is a standard way to calculate the slope *m* and the y-intercept *b* of the line $y = mx + b$, where x = E+S, and y = E-S [18].

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum x^2 - (\sum x)^2},$$

and

$$b = \frac{\sum y - m\sum x}{n}.$$

Given the slope-intercept form of a linear fit of a curve, it is trivial to measure the average distance of that curve from this line.

3. Experimental Setup

We performed two types of experiments: typing and secret-stealing. Both experiments involved 2-3 individuals logging onto 2-3 distinct remote hosts via SSH through a single stepping-stone, located at University of Houston, Texas, from three different hosts. The stepping-stone computer was running our software that was monitoring the streams of interest and recording packets in those streams. At the end of the experiment our software would relate each ECHO stream to each SEND stream in the space of ECHO – SEND versus ECHO + SEND and output it to a file. These files were later analyzed via Excel based on the procedures described 2.2 and 2.3 above. The point of these experiments 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. The idea here is that if it's possible to correctly distinguish connection chains in such a situation, then our procedures should work very well in situations where there is only one connection chain and many other completely unrelated incoming and outgoing connections. Further experimentation if this idea of ours is correct.

The typing experiment consisted of the participants opening a text editor on a victim computer and typing up 15-minutes worth of text. The following are the connection chains:

Jianhua: home computer (via SBC) → stepping-stone→ themis.cs.uh.edu → Mexico
Robert: UH-TLC → acl08.cs.uh.edu → stepping-stone → UMASS

Scott: UH-TLC → stepping-stone → TAMU

For the first three trials all participants were to type identical texts simultaneously although at slightly different rate. The last trial involved all three individuals typing different texts at different rates and not simultaneously. UMASS is a host located in University of Massachusetts, and TAMU is a host located in Texas A&M University.

The file downloading experiment consisted of the participants searching for a file on a victim computer by going through a number of directories that contained fake files. The test directory, consisting of the test files, was prepared in advance. The target file was copied onto the attacker's machine upon discovery. This experiment was repeated three times. The following are the connection chains:

Jianhua: cs.uh.edu → stepping-stone→ bayou → Mexico
Robert: TLC^2 → acl08.cs.uh.edu → stepping-stone → UMASS
Scott: TLC^2 → stepping-stone → TAMU

We found that implementing time-jittering and chaff is more complicated. Since we were not able to obtain SST from Guan Yong et al. [13], we decided to simulate time-jittering and chaff capabilities by changing the results that we obtain from regular experiments and performing the same analysis on the changed data as we usually do on the regular data.

The time-jittering and chaff were implemented to take place during the last portion of stepping-stone experiments. It is more convenient to introduce perturbations to the data while the program that records the packet data is still running because packet data is organized into a matrix that is easier to handle. For every stream, SEND and ECHO, of every connection the time-jittering perturbation is introduced as an addition of a randomly-chosen time extension, limited by a pre-specified limit, to the time stamps of the packet record. Not every packet record was thus time-jittered; every packet had a probability of only one-half to be time-jittered. Therefore, at least one-half of all packet records had their time-stamp be increased by a time chosen uniformly between 0 and some pre-specified limit. Microsecond resolution was used.

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 amount, to the original stream that is recorded inside a matrix. Two different ways of chaff were performed. The first technique consisted of generating a stream of superfluous packets, whose capture-time difference 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 .05-.10, inter-packet time intervals of the original packet stream. For both techniques, such parameters represent the worst-case scenario where the most chaff is introduced. Experiments performed with other time limits are not discussed here.

4. Analysis and Discussion

The main assumption for our analysis is that the relationship between E-S and E+S should be linear. As the reader will see, this assumption is justified by the looks of the curves of the plots and each curve's correlation coefficient. In other words, not only do the curves look linear, but correlation coefficients, computed using the procedure described in Section 2.2, for the correlation coefficients of the curves that correspond to real connection chains are always above 90%. The fact allows us to use the procedure described in Section 2.3 in order to measure the smoothness of the curves. It makes sense that the correlation coefficient for experiments without time-jittering and chaff is positive because our whole study is focused on interactive attacks, where Eve will get back more packets from Bob than she sends to Bob. However, this is not the case for experiments with time-jittering and chaff simulations.

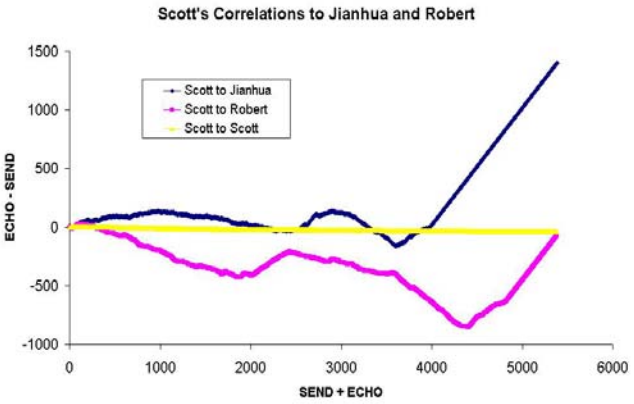
4.1 Basic Experiments

It turns out that for both types of the experiments 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 user. This can be seen just by looking at the curves on the plots of ECHO - SEND versus ECHO + SEND of experiments we took at the beginning of this project (see Figures 4 and 5 below). Data obtained from experiment shown on Figure 4 was not quantitatively analyzed as this experiment does not really model a real interactive attack, and basic qualitative analysis here yields the right result. Data obtained from experiment shown on Figure 5 was quantitatively analyzed with procedure described in 2.3 above. Results of this analysis are shown in the legend. To summarize, the experiments shown in Figures 4 and 5 give reassuring results because they show 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.

Contrary to the strategy used in [1], based on the results from both types of experiments, it is clear that the magnitude ECHO – SEND is not bounded for a real stepping-stone chains. Even when participants typed the same commands at the same time, due to differences in typing speed, each participant yielded data with different upper

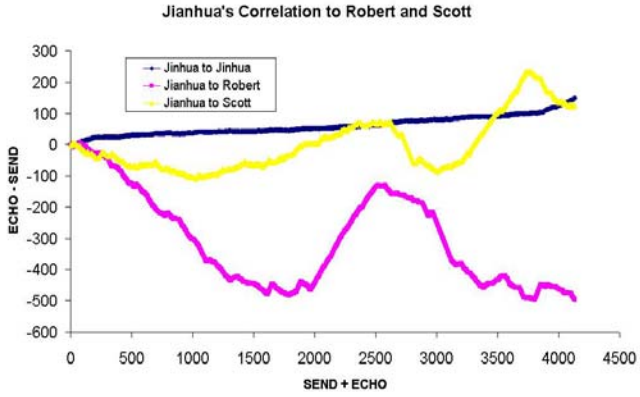
boundaries. Therefore, it is not sufficient for magnitude of ECHO – SEND to be within a particular limit in order to qualify its corresponding request-response connection pair as a part of a stepping-stone chain.

Results of the typing experiment showed that for some connection chains there were more packets in the SEND stream than in the ECHO stream (see Figure 4). Such situation could be due to the fact that our software counted retransmissions in the SEND stream, and/or because ECHO stream consisted of fewer packets, each with a larger payload, than the SEND stream. The former was fixed prior to the secret-stealing experiment, and the latter could be due to the fact that the victim computer received more than one packet from the stepping-stone chain before replying. Overall, however, the number of packets in the ECHO stream was about the same as the number of packets in the SEND stream for the typing experiment. At no point were there more packets in the SEND stream than in the ECHO stream for the secret-stealing experiment. All curves based on data from this experiment had a clear positive slope (see Figure 5).

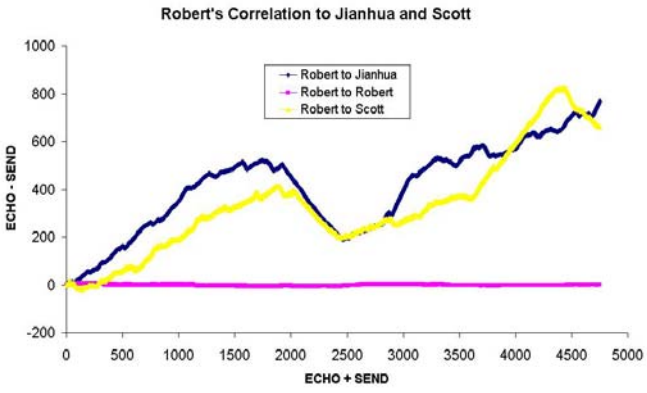


(c)

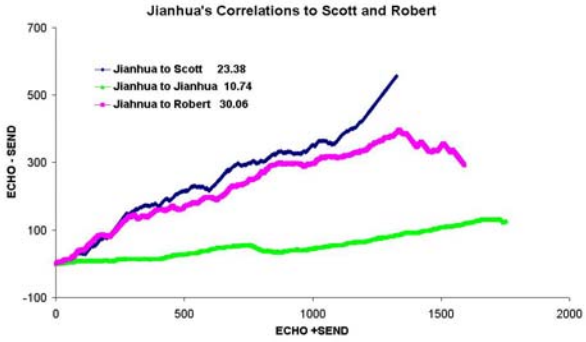
Figure 4: Correlations of the ECHO stream of a particular participant (Jianhua, Robert, and Scott in (a), (b), and (c), respectively) to the SEND streams of all the participants in the typing experiment.



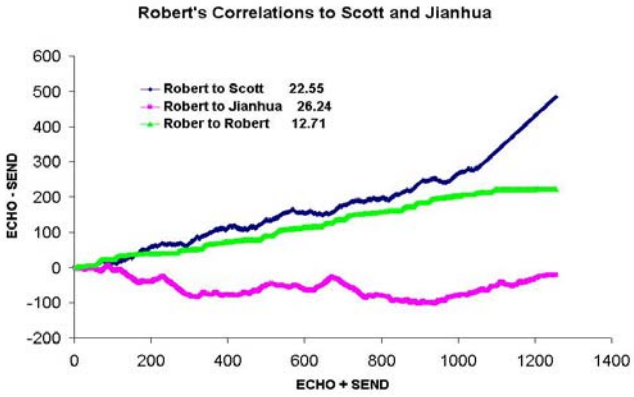
(a)



(b)



(a)



(b)

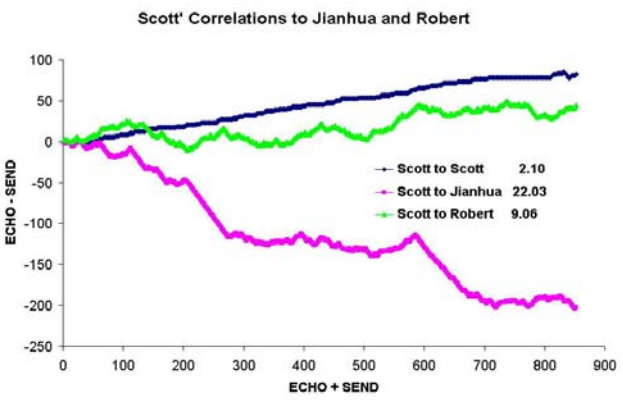
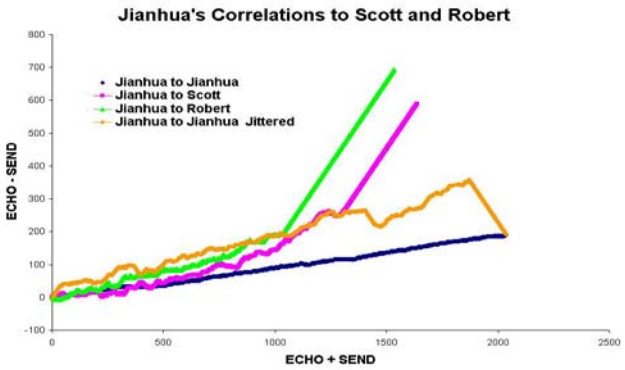


Figure 5: Correlations of the ECHO stream of a particular participant (Jianhua, Robert, and Scott in (a), (b), and (c) respectively) to the SEND streams of all the participants in the secret-stealing experiment.

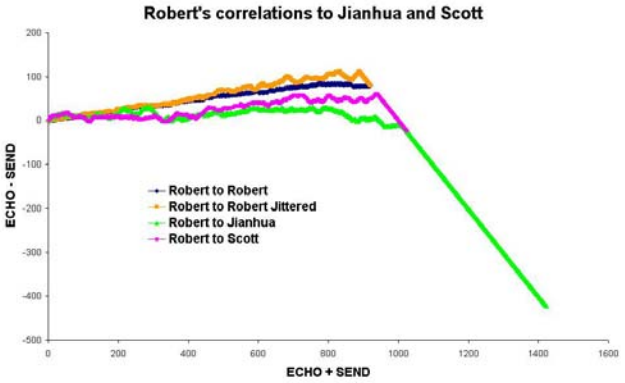
4.2 Detection with Time Jittering Evasion

We mostly looked at 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 from Figure 6. All the SEND packets were pushed so far ahead that the curve that corresponds to time-jittered connection chain has a positive slope and high negative slope at the beginning and the end of the curve respectively. The last portion of the curve cannot have a high negative slope because there should be more packets in the ECHO stream than in the SEND stream. This shortcoming causes our simulation to yield unrealistic results because it does not take into account that some ECHO packet can come only after their corresponding SEND packets. As can be noticed from the beginning portions of the curves that correspond to connection chains on Figures 6a and 6b, time-jittering does cause noise, but it is insignificant. The ends of these curves also exhibit the shortcomings of our simulation.

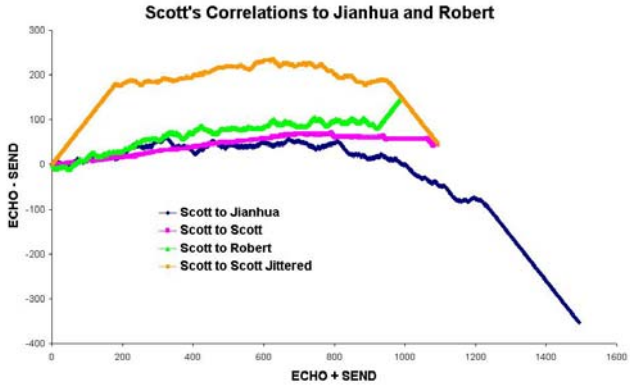
There are two procedures we could try to utilize in order to preserve the order of SEND packets with respect to ECHO packets. The first one deals with matching SEND packets to ECHO packets by taking into account the time it takes a packet to reach the victim and come back to the adversary. This technique requires us to implement special algorithms into our software, and it is not 100% reliable. The second one deals with matching the content of the SEND packet to the content of the ECHO packets. This technique requires us to use telnet, whose use is not allowed for most servers and much time in order to match the content of packets correctly. We think 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 think 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 packet can come only after their corresponding SEND packets and vice versa, this disruption is not expected to be significant. Therefore, we decided to shift our main focus to implementing chaff simulation.



(a)



(b)



(c)

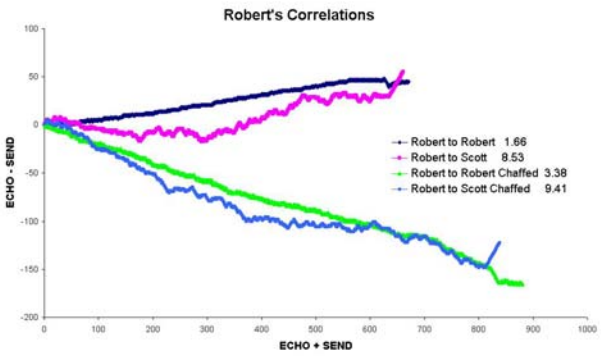
Figure 6: Correlations of the ECHO stream of a particular participant (Jianhua, Robert, and Scott in a, b, and c respectively) to the SEND streams of all the participants in the secret-stealing experiment with a time-jittered simulation.

4.3 Detection with Chaff Evasion

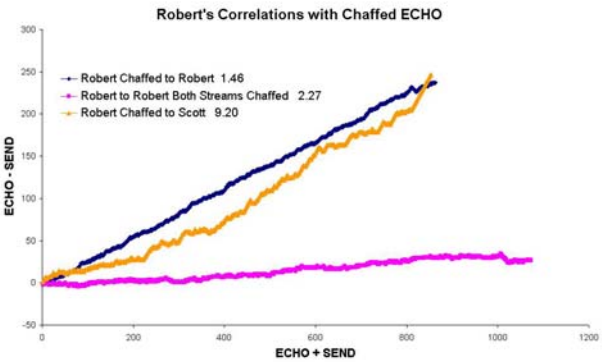
We looked at data that resulted from chaffing the SEND, the ECHO and both SEND and ECHO streams of various connections. After undergoing such perturbations, every vector with perturbed data was merged with data of various ECHO streams. In order to see whether chaff really makes a difference for each participant we generated two plots. The first one is used to compare the ECHO data of a particular user to all the users' regular and chaffed SEND data (Figures 7a, 7c, 8a, 8c). The second plot is used to compare the chaffed ECHO data of a particular user to all the users' regular SEND data and his/her complementary chaffed SEND data (Figures 7b, 7d, 8b, 8d). An important assumption that we use here is that Eve only chaffs data of one user, therefore we do not correlate chaffed data of one user to that of another user.

As can be seen from Figure 7, the first chaff technique does not introduce much noise to the data, but it stretches the data a bit. When only the SEND stream is chaffed, the curve has a negative slope. When only the ECHO stream is chaffed and the both streams are chaffed, the curve has a positive slope. It must be mentioned that all the experiments with the first chaff technique yielded data where the curve that corresponded to the real connection chain could be distinguished due to less noise by the means described in section 2.3.

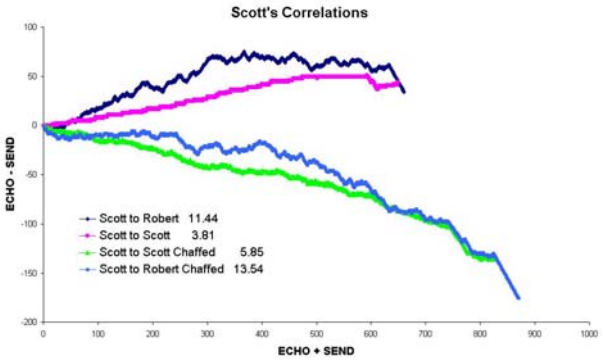
As can be seen from Figure 8, the second chaff technique is more aggressive than the first one and it introduces significant noise to the data. Like in the first case, when only the SEND stream is chaffed, the curve has a negative slope; when only the ECHO stream is chaffed and when both streams are chaffed, the curve has a positive slope. Here we cannot always distinguish the curve that corresponded to the real connection by the means described in section 2.3 because thus-introduced chaff makes data so noisy that when one's ECHO data is correlated with that participant's complementary chaffed SEND data, analysis based on procedure described in Section 2.3 yields a number that is larger than the curve of that users ECHO data related to the other participant's SEND data (see Figures 8a and 8c). A similar behavior happens when both ECHO and SEND streams are chaffed (see Figures 8b and 8d). Such results, however, are not very discouraging as they just signify that it is difficult to distinguish data of two users performing the same task at the same time when at least one of the streams is chaffed. There is still a way to distinguish these connections simply by eliminating contradiction that various plots may have. Thus, for example, in the case that Robert's SEND stream is chaffed, if curve *Scott to Scott* in Figure 8c is closer to its liner fit than curve *Robert to Scott* is in Figure 8a, then we know that even though curve *Robert to Scott* may be closer to its linear fit than curve *Robert to Robert Chaffed* is in Figure 8a, Robert's ECHO data cannot correlate to Scott's SEND. By process of elimination, Robert's ECHO data may then only correlate to Robert's chaffed SEND data. Such process of elimination requires $O(n^2)$ operations, and it needs to be checked whether it works with many connections. This shows that the second chaff technique, as opposed to the first one, is probably what Eve might want to use to evade our stepping-stone detection approaches. Further experimentation where participants are not performing the same tasks should show how well our detection mechanism works with chaffed connections.



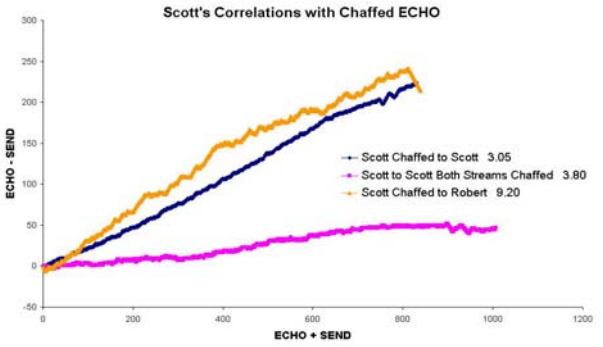
(a)



(b)

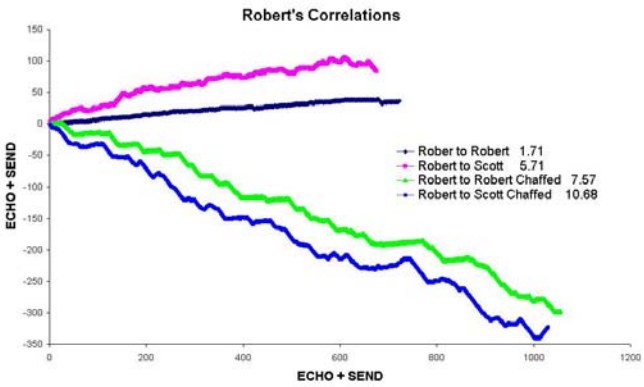


(c)

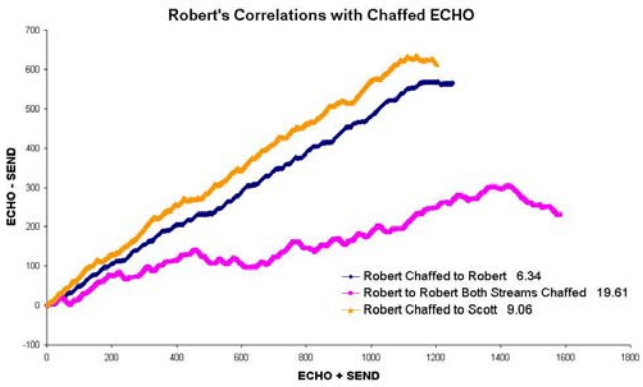


(d)

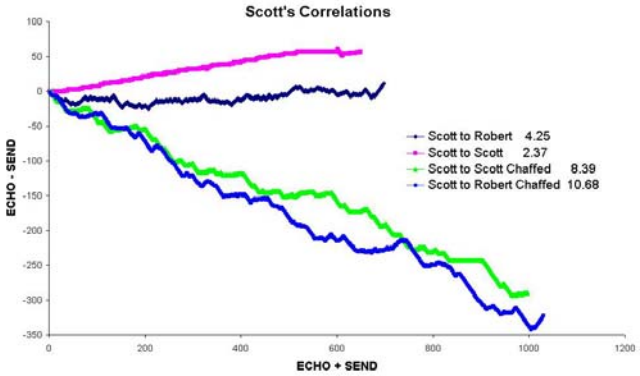
Figure 7: Correlations of the ECHO stream of a particular participant (Robert and Scott in (a), (b), (c), and (d), respectively) to the SEND streams of all the participants in the secret-stealing experiment with chaff simulation via the 1st technique.



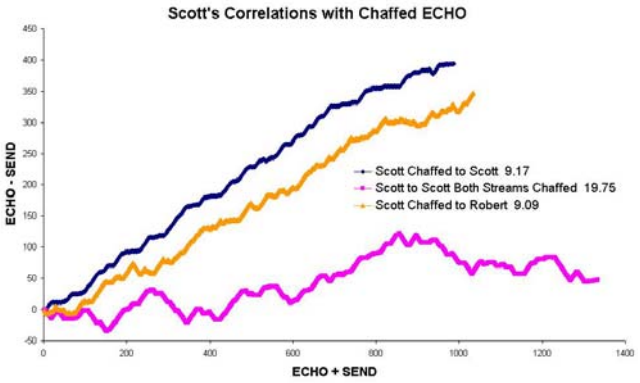
(a)



(b)



(c)

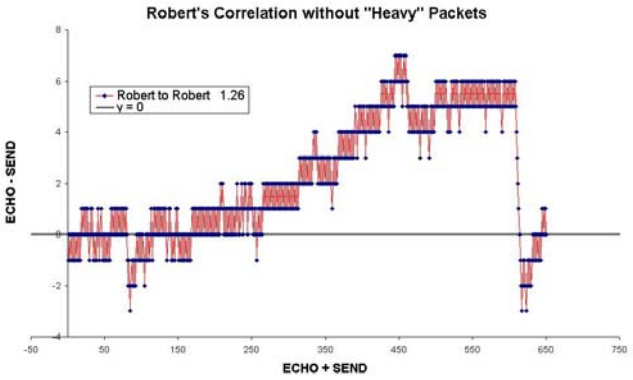


(d)

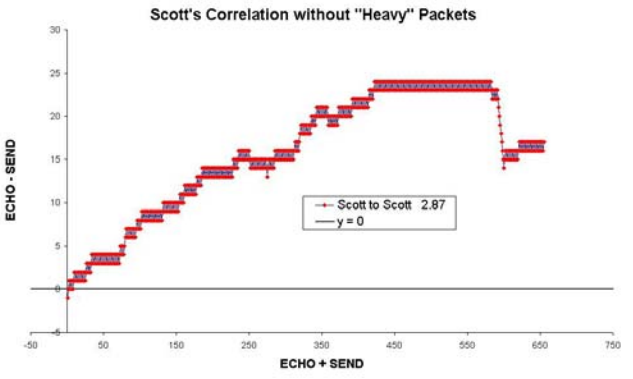
Figure 8: Correlations of the ECHO stream of a particular participant (Robert and Scott in (a), (b), (c), and (d), respectively) to the SEND streams of all the participants in the secret-stealing experiment with chaff simulation via the 2nd technique.

We have tried to see if our data would yield more satisfying results with respect to chaff by analyzing only packets that hold single characters. Such analysis ignores any heavy packets, usually occurring in the ECHO stream that

might be a result of a computer putting more than just one character into a single packet. We were expecting that such analysis would provide for an easier way to distinguish a curve that represents a real connection chain because such curve would oscillate about the x-axis within particular limits, while every other curve would go beyond those limits. This assumption makes sense because without the "heavy" packets every packet in the ECHO stream would be directly mapped to something that the adversary typed into the prompt. Our experiments showed that sometimes our assumptions are true and sometimes they are wrong as depicted in Figures 9a and 9b respectively. Further analysis of filtered streams shows no significant differences from that of unfiltered streams and its discussion is, therefore, omitted from this paper.



(a)



(b)

Figure 8: Correlations of the ECHO stream of a particular participant (Robert and Scott in (a) and (b), respectively) to their respective SEND streams after filtering out the "heavy" packets in the secret-stealing experiment.

5. Conclusions

Based on our experiments we can with confidence say, that procedure described in section 2.3 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. As discussed in section 4.3, we cannot say the same when chaff is involved. Further investigation is needed in order to check whether it is possible to resolve this issue by iteratively eliminating contradictions as discussed at the end of Section 4.3.

Our project is not complete as more experimentation is needed before and definitive claims could be regarding our procedure for finding connection chains. Moreover, further investigation is needed to check if our procedure for establishing connection chains works well in situations where there is only one connection chain and many other completely unrelated incoming and outgoing connections. After all these experiments, some theoretical research is necessary in order to come up with the minimum number of monitored packets in order to decided whether a computer belongs to a connection chain or not (this is equivalent to identifying it as a stepping-stone or not) with sufficient negative and positive false rates.

Acknowledgement

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

References

[1] A. Blum, D. Song, and S. Venkataraman, "Detection of Interactive Stepping Stones: lgorithms and Confidence Bounds," in Proc. of 7th International Symposium on Recent Advances in Intrusion Detection (RAID '04). SpringerLNCS 3224, pages 258-277, 2004.

[2] D. Donoho, A.G. Flesia, U. Shankar, V. Paxson, J. Coit, S. Staniford, "Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay," in Fifth International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science 2516, 2002.

[3] B. Duwairi, A. Chakrabarti, and G. Manimaran, "An Efficient Probabilistic Packet Marking Scheme for IP Traceback," in Proc. of NETWORKING '04, LNCS 3042, pages 1263-1269, 2004.

[4] M.T. Goodrich, "Efficent Packet Marking for Large-Scale IP Traceback," in Proc. of ACM CCS '02, pages 117-126, November 2002.

[5] H.T. Jung, H.L. Kim, Y.M. Seo, G. Choe, S.L. Min, C.S. Kim and K. Koh, "Caller Identification System in the Internet Environment," in Proc.of the UNIX Security Symposium, pages 69-78, 1993.

[6] S. Savage, D. Wetherall, A. Karlin and T. Anderson, "Practical Network Support for IP Traceback," in Proc. of the ACM SIGCOMM '00, April 2000.

[7] D. Song and A. Perrig, "Advanced and Authenticated Marking Scheme for IP Traceback," in Proc. of IEEE INFOCOM '01, April 2001.

[8] S. Snapp, et al., “DIDS (Distributed Intrusion Detection System) – Motivation, Architecture and Early Prototype,” in Proc. of 14th National Computer Security Conference, pages 167- 176, 1991.
[9] S. Staniford-Chen, L. T. Heberlein, “Holding Intruders Accountable on the Internet,” in Proc. of the IEEE Symposium on Security and Privacy, May 1995.
[10] X. Wang, D.S. Reeves, S.F. Wu and J. Yuill, “Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework,” in Proc. of 16th International Conference on Information ecurity (IFIP/Sec’01), pages 369-384, June 2001.
[11] X. Wang and D.S. Reeves, “Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Interpacket Delays,” in Proc. of ACM CCS ’03, October 2003.
[12] X. Wang, “The Loop Fallacy and Serialization in Tracing Intrusion Connections through Stepping Stones,” in Proc. of the 2004 ACM Symposium on Applied Computing, ACM Press (2004)
[13] J. Xin, L. Zhang, B. Aswegan, J. Dickerson, T. Daniels, and Y. Guan, "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 (TridentCom 2006), Barcelona, Spain, March 1-3, 2006.
[14] K. Yoda and H. Etoh, “Finding a Connection Chain for Tracing Intruders,” in F. Guppens, Y. Deswarte, D. Gollmann and M. Waidner, editors, 6th European Symposium on Research in Computer Security—ESORICS 2000 LNCS-1795, October 2000.
[15] Y. Zhang and V. Paxson, “Detecting Stepping Stones,” in Proc. of the 9th USENIX Security Symposium, pages 171-184, 2000.
[16] L. Zhang, A. G. Persaud, A. Johnson, Y. Guan, “Detection of Stepping Stone Attack under Delay and Chaff Perturbations," in 25th IEEE International Performance Computing and Communications Conference, Pheonix, USA, Apr. 2006.
[17] BizEd, “Correlation Explained on time web.” URL: http://www.bized.ac.uk/timeweb/crunching/crunch_relate_expl.htm, accessed at 06-30-06.
[18] Clemson University, “Physics Tutorial: Linear Regression.” URL: http://phoenix.phys.clemson.edu/tutorials/regression/index.html, accessed at 06-30-06.
[19] Jianhua Yang, Shou-Hsuan Stephen Huang, “A Real-Time Algorithm to Detect Long Connection Chains of Interactive Terminal Sessions”, Proceedings of 3rd International Conference on Information Security (Infosecu’04), Shanghai, China, November 2004, pp. 198-203.
[20] Jianhua Yang, Shou-Hsuan Stephen Huang: Matching TCP Packets and Its Application to the Detection of Long Connection Chains, IEEE Proceedings of 19th International Conference on Advanced Information Networking and Applications (AINA’05), Taipei, Taiwan, March 2005, pp. 1005-1010.