

# The Way by Estimating the Variation of TCP Packet Round Trip Time to Detect Stepping-Stone Intrusion-Extended Abstract

## Abstract

We propose Std-ratio approach to detect stepping-stone intrusion, as well as Std-clustering method to find TCP packet round trip time of a connection chain. Std-ratio is a method to use the ratio between the standard deviation of the round trip time gaps to the downstream neighbor host and to the end side of a connection chain. Std-clustering is a method to determine the round trip time cluster by comparing the standard deviation of each candidate cluster. Comparing with previous approaches, Std-ratio has the advantages: 1) it can be applied to encrypted sessions; 2) it is not necessary to monitor a connection chain all the while; 3) it has low false negative and positive error; 4) it can overcome the unit bar problem.

## Keywords

Network security, round trip time, stepping-stone, intrusion detection, standard deviation

## 1. Introduction

A common way to detect stepping-stone [1] intrusion is to estimate the number of hosts compromised by computing the length of a connection chain [2][3][4]. One typical way to compute the length of a connection chain is to estimate a TCP send packet [5] round-trip time (RTT). Most of the previous approaches [2][3][4] focused on a point to compute a packet RTT by matching a send packet with its corresponding echo packet. The first problem is that so far there is no way to match up send and echo packets exactly, especially when there is send-echo pair overlap which occurs often on the Internet. The second problem is even if we can estimate the length of a connection chain, but how we do know if that length is long or short. That is to say we need a unit bar to measure how long a connection chain is. The paper [2] proposed an idea to use the RTT between a send packet and its corresponding acknowledgement from a downstream neighbor host as its unit bar. This will incur false negative error if the unit bar is too long or short. The step-function method proposed in [3] can overcome the above difficulty but introducing a prob-

lem that a connection chain must be monitored all the while.

In this paper we propose an approach to find the RTTs of a connection chain by using standard deviation based clustering algorithm (Std-clustering) rather than traditional clustering method [6][7][8], and a method to detect stepping-stone intrusion by computing the ratio between the standard deviation of the RTTs to the downstream neighbor host and the standard deviation of the RTTs to the victim site (Std-ratio). The smaller the ratio is, the high probability a connection is used by an intruder. The experiment results showed that Std-clustering algorithm can find more accurate RTTs than the Greedy algorithm, and much more RTTs than the Conservative algorithm [4]. Std-ratio algorithm can solve unit bar problem of paper [2], decrease the false positive error and the false negative error.

## 2. Main Technical Ideas

Suppose a connection chain established by using OpenSSH (here it can be any tools, such as telnet, rlogin) [9][10] is monitored at host  $h_i$ , its down stream neighbor host is  $h_{i+1}$ , the end host of the connection chain is host  $h_n$  which is called the victim site as well. We capture the send and echo packets of the chain in a certain time interval and put them into two sequences S and E respectively. Here we assume S with n elements and E with m elements are as the following,

$$S = \{s_1, s_2, \dots, s_n\},$$
$$E = \{e_1, e_2, \dots, e_m\},$$

here s represents a send packet, e represents a echo packet, and the subscript of them represents the packet chronological order.

Each send packet at  $h_i$  is going be acknowledged at host  $h_{i+1}$  first, and finally echoed by  $h_n$ . We also capture the packets acknowledged by  $h_{i+1}$ (ack packet) at host  $h_i$  and put them into sequence A. Assuming A has k elements as the following,

$$A = \{a_1, a_2, \dots, a_k\}.$$

The RTT between a send packet and its echo packet is denoted e-RTT, as well as the RTT between a send packet and its ack packet is denoted a-RTT. We make an assump-

tion that all the send packets in S are echoed or acknowledged by the packet in E or A.

## 2.1 Std-clustering algorithm to find RTT sequence

Given two sequences S and E or A, we form n sets  $S_1, S_2, \dots, S_n$  with size  $N_1, N_2, \dots, N_n$  respectively, as the following,

$$\begin{aligned} S_1 &= \{s_1e_1, s_1e_2, \dots, s_1e_m\} \\ S_2 &= \{s_2e_{i_2}, s_2e_{i_2+1}, \dots, s_2e_m\} \\ &\dots \\ S_n &= \{s_n e_{i_n}, s_n e_{i_n+1}, \dots, s_n e_m\} \end{aligned}$$

Here,  $s_i e_j$  represents the gap between  $i^{\text{th}}$  send packet and  $j^{\text{th}}$  echo packet. Each element of  $S_1, S_2, \dots, S_n$  must be positive value.

We generate cluster  $C_i$  ( $1 \leq i \leq N_1 * N_2 \dots * N_n$ ) and find the RTT cluster as the following steps:

- 1)  $C_i$  has n elements  $c_{i1}, c_{i2}, \dots, c_{in}$  with standard deviation  $\sigma_i$ ;
- 2)  $c_{i1} \in S_1, c_{i2} \in S_2, \dots, c_{ik} \in S_k, \dots, c_{in} \in S_n$ ;
- 3) For any two clusters  $C_i$  and  $C_j$ , there is at least one element different.
- 4) Find the cluster  $C_{RTT}$  among all the clusters formed with smallest standard deviation.
- 5) Filter  $C_{RTT}$  by removing the elements which have the same echo packet and only keep the one with smallest gap.

## 2.2 Std-ratio algorithm to detect stepping-stone intrusion

- 1) Find the a-RTT cluster and e-RTT cluster from S, E and A sequences by using Std-clustering algorithm;
- 2) Compute the standard deviation  $\sigma_e$  of e-RTT and  $\sigma_a$  of a-RTT clusters respectively;
- 3) Compute the ratio r between  $\sigma_e$  and  $\sigma_a$ .
- 4) Compare the ratio r to one with any two predefined numbers  $\epsilon, \delta$  which are all between 0 and 1, and  $\epsilon < \delta$  is satisfied.

If  $|r - 1| > \delta$ , then there is an intrusion;

Or if  $|r - 1| < \epsilon$ , then there is no intrusion at all;

Or if  $\delta < |r - 1| < \epsilon$ , then it is undetermined.

## 3. Experiment Results

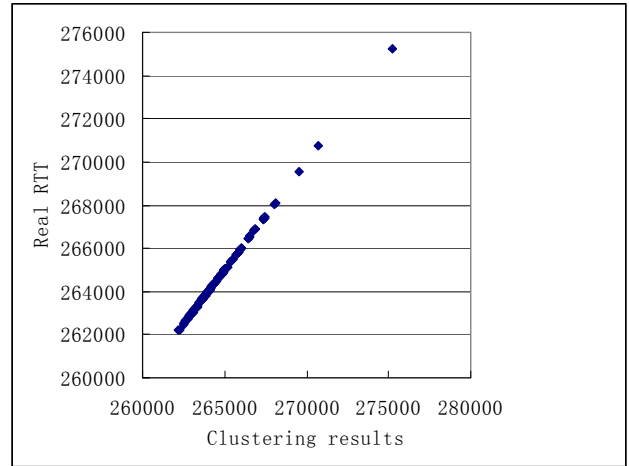
We designed two experiments on the Internet to verify the above two algorithms. In the experiments one connection chain that spanned USA and Mexico was established. The hosts used in the experiments are all located in University of Houston but the host Mex which is located in Mex-

ico. Our program made by using Libpcap [11][12] based on the algorithms proposed in this paper were running at Acl08 which is a host under control.

## 3.1 Experiment to verify Std-clustering algorithm

The best way to verify Std-clustering algorithm is to compare the clustering result with real RTTs. The way to get real RTTs of a connection is to control the typing speed to make each send packet to be echoed before the next packet is sent. We built a connection chain that had four more hosts connected after Acl08 and monitored the chain, captured the send and echo packets only when the chain had been established. We control our typing speed as slow as possible so as to be sure that each packet had been echoed before the next packet was sent. It is easy to get the RTT for each send packet because there is no send-echo overlap. We use Std-clustering algorithm to get the cluster results which are supposed to correspond to the RTTs of the chain. We compare them to see if all the RTTs are clustered. The comparison result is shown in Figure 1 in which X axis represents clustering RTTs and Y axis represents the real RTTs each with unit microsecond.

From Figure 1 we know that all of the real RTTs are clustered by Std-clustering algorithm. In this experiment



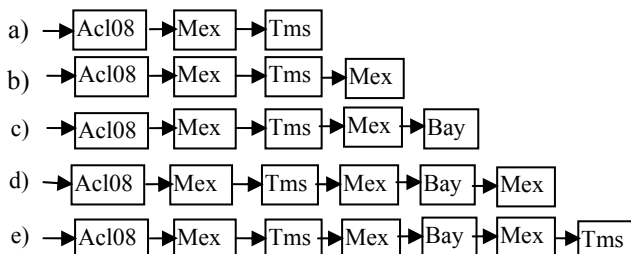
**Figure 1: To verify the accuracy of clustering algorithm**

we captured 119 send packets and had 119 real RTTs. Std-clustering algorithm output 119 RTTs with each value very close to the real one.

## 3.2 Experiment to verify Std-ratio algorithm

The purpose of this experiment is to verify if the Std-ratio algorithm works in detecting stepping-stone intrusion. We established a connection chain as Figure 2 a) shows, and then increased the chain one more host each time as Figure 2 b)-e) show respectively. For each chain we captured send, echo and ack packets at Acl08, carried out Std-

clustering algorithm to get a-RTT and e-RTT clusters, computed their standard deviations. The results are presented in Table 1. It shows that the more hosts a chain has, the more possible the chain is used by intruders. We also have another experiment result that showed the unit bar problem can be solved by Std-ratio algorithm. Duo to the limitation of the pages, we will not show the result here.



**Figure 2: Connection chains with two to six hosts compromised downstream from monitor point Acl08**

Item Host N	$\sigma_e$	$\sigma_a$	r
2	1475.08	2072.27	0.7118
3	1224.06	3545.01	0.3453
4	1024.81	5046.09	0.2031
5	1140.68	4881.66	0.2337
6	917.14	5409.90	0.1695

**Table1: Experiment results with different length of a connection chain**

#### 4. Conclusion and Future Work

We have proposed Std-ratio approach to detect stepping-stone intrusion, as well as Std-clustering algorithm to find the RTTs of a connection chain. Std-ratio is a method to detect stepping-stone intrusion by estimating the length of a connection via the ratio between the standard deviation of the a-RTTs and of the e-RTTs for the same chain. Comparing with other approaches, it has the advantages: 1) it can be applied to encrypted sessions; 2) it is not necessary to monitor a connection chain all the while; 3) it has low false negative and positive error; 4) it can overcome the unit bar problem.

The work presented was carried out under the conditions that the network traffic is uniform and there is no intruder's manipulation over the connection chain. In the future, we need to verify Std-clustering, and Std-ratio algorithm under different network traffic, as well as the chain is manipulated by an intruder, such as packet random delay and chaff [13]. Another work in the future is to prove that the cluster with smallest standard deviation has the highest possibility to be RTT sequence in theory.

#### References

- [1] Yin Zhang, Vern Paxson, "Detecting Stepping-Stones", Proceedings of the 9th USENIX Security Symposium, Denver, CO, August 2000, pp. 67-81.
- [2] Kwong H. Yung, "Detecting Long Connecting Chains of Interactive Terminal Sessions", RAID 2002, Springer Press, Zurich, Switzerland, October 2002, pp. 1-16.
- [3] J. H. Yang, S. Huang, "A Real-Time Algorithm to Detect Long Connection Chains of Interactive Terminal Sesiions", Proceedings of 3rd International Conference on Information Security (Infosecu'04), Shanghai, China, November 2004, pp. 198-203.
- [4] Jianhua Yang, Shou-Hsuan Stephen Huang: Matching TCP Packets and Its Application to the Detection of Long Connection Chains, Proceedings (IEEE) of 19th International Conference on Advanced Information Networking and Applications (AINA'05), Taipei, Taiwan, China, March (2005) 1005-1010.
- [5] University of Southern California, Transmission Control Protocol, RFC 793, 1981.
- [6] M. Friedman, A. Kandel, "Introduction to Pattern Recognition : Statistical, Structural, Neural, and Fuzzy Logic Approaches", NJ World Scientific Publishing Co., River Edge, London, 1999.
- [7] B. Mirkin, "Mathematical Classification and Clustering", Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996, pp. 169-198.
- [8] A. Jain, R. Dubes, "Algorithms for Clustering Data", Prentice Hall, Inc., New Jersey, 1988, pp. 55-143.
- [9] Ylonen, T., "SSH Protocol Architecture", draft -IETF document, <http://www.ietf.org/internet-drafts/draft-ietf-secsh-architecture-16.txt>, June 2004.
- [10] Ylonen, T., "SSH Transport Layer Protocol", draft IETF document, <http://www.ietf.org/internet-drafts/draft-ietf-secsh-transport-18.txt>, June 2004.
- [11] Lawrence Berkeley National Laboratory (LBNL), "The Packet Capture library", <ftp://ftp.ee.lbl.gov/libpcap.tar.z>, accessed March 2004.
- [12] Data Nerds Web Site, "Winpcap and Windump", <http://www.datanerds.net>, accessed July 2004.
- [13] D. L. Donoho (ed.), Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay. Proc. 5th International Symposium on Recent Advances in Intrusion Detection, Zurich, Switzerland, 2002, 45-59.