An Algorithm to Detect Stepping-Stones in the Presence of Chaff Packets

Ying-Wei Kuo and Shou-Hsuan Stephen Huang

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
Email: {ykuo, shuang}@cs.uh.edu

Technical Report Number UH-CS-08-11
July 23, 2008

Keywords: Stepping-stone, network security, connection chain, intrusion detection, chaff.

Abstract

A major concern for network intrusion detection systems is the ability of an intruder to evade the detection by routing through a chain of the intermediate hosts to attack a target machine and maintain the anonymity. Such an intermediate host is called a stepping-stone. The intruders have developed some evasion techniques such as injecting chaff packets. A number of algorithms have been proposed to detect stepping-stones, but some of them failed to detect correctly when the network traffic is somehow corrupted or with the chaff packets. We discuss the viability of solving those issues by improving a previous methodology. The algorithm is based on finding as many matched pairs of incoming and outgoing packets on the same host as possible and then decide whether it is a stepping-stone connection by the mismatched rate. We examine a number of tradeoffs in choosing the threshold values by simulating network traffic. Our experiments report a very good performance with very low false detection rates when using carefully selected parameter values.

1. Introduction

Network Intrusion Detection Systems (NIDS) are responsible for detecting malicious, inappropriate, anomalous activity or any other data, which may consider as an unauthorized occurring within a network. Unlike a firewall, which is configured to allow or reject access to a particular service/host based on certain rules, a NIDS captures and inspects all traffic and then looks for suspicious patterns. The goal of NIDS is to detect the intrusion and trace back to the intruders.

Unsurprisingly, intruders want to maintain anonymity by hiding their true identities. A common method to do so is to use a sequence of previously compromised, intermediary hosts to initiate attacks rather than directly connected from the attacker's own computer. In this way, one can trace back only one host. Therefore, by performing their attacks through a sequence of several intermediate hosts, the intruder's identity can be hidden. This sequence of intermediate hosts is called "*stepping-stones*". Nowadays, network intruders have learned to inject some chaff packets into the attacked traffic flows which make tracing the source of stepping-stone attack substantially more difficult.

In this paper, we assume the packets are encrypted and there are several stepping-stone hosts to form a connection chain. In reality, traffic may often be corrupted or chaffed and such corrupted or chaffed traffic causes some of the previous detection algorithms to fail. Hence, we modify previous algorithm on matching input and output streams so that it tolerates some chaff packets or corrupted data.

We summarize previous related work and discuss the issues related to stepping-stone detection in Section 2. A new definition for matching streams is presented in Section 3. We then describe an algorithm to detect stepping-stone pairs in Section 4. We then examine our algorithm and report the performance along with several factors in Section 5. Finally, we conclude the paper.

2. Related Work

The earlier algorithms to detect stepping stones were based on the packet contents. However, one major limitation for packet contents approach is not able to detect the encrypted connections. In order to detect intrusions on the encrypted traffic, the timing characteristics have been taken into the consideration. There are some well known timing-based approaches [2, 4] which used the incoming and outgoing times of packets to correlate connections. In addition, some algorithms [1, 3, 9] proposed to detect stepping-stones are based on the round-trip time (RTT). This RTT is very small for normal connections but increases proportionally with the number of intermediate hosts in the chain.

Donoho *et al.* [6] presented the limitations of timing perturbation by the intruders on timing-based correlation. They provided a multi-scale approach, which pointed out that intruders might transform their traffic to defeat stepping-stone detection. However, there were limitation on the ability of intruders to hide their traffic through timing perturbation and packet padding. Yet, some issues such as the chaff perturbation, and the tradeoff of false positive rates (probability of incorrectly identified normal pairs) were not considered.

Blum *et al.* [5] proposed the packet matching schemes that considered both delay and chaff perturbations into encrypted stepping stone connections. They introduced the Detect-Attacks method, which achieves polynomial upper bounds on the number of packets needed to confidently detect and identify stepping stone flows with guaranteed false positive rates. Moreover, they addressed Detect-Attack-Chaff method which was used to calculate the bounds on the amount of chaff packets that intruders might plug in to evade detection. However, Blum's methods could not deal with large amount of packets with delay and chaff perturbations existed simultaneously.

He and Tong [7, 12] proposed Detect-Match (DM) algorithm, which is similar to the detection scheme in [13]. Moreover, DM algorithm restricted its search to all the order preserving mappings between an incoming stream and an outgoing stream. If each incoming packet is matched to an outgoing packet within a fixed time window, this connection would be defined as an ATTACK (stepping-stone) connection; otherwise, it would be a NORMAL connection. Since only the order preserving mapping needed to be considered, the major advantage was to greatly reduce the complexity of DM algorithm from exponential to linear.

Kuo and Huang [14] discovered that DM algorithm did not take all the possible ATTACK pairs into consideration. In other words, it is possible to have some missed detection. They provided a new algorithm called Detect-Min-Index-Match (DMIM) which was able to detect the stepping-stone (ATTACK) pairs within the same time complexity. The paper provided a correctness proof and the experimental results of the algorithm. DMIM algorithm is guaranteed to find the mapping, if there is one; if DMIM algorithm cannot find the ATTACK mapping, then no mapping exists.

However, the algorithm used the same assumption that every incoming packet will appear on the outgoing stream within a maximum tolerable delay. The assumption sounded reasonable and was used in [7, 12]. In reality, the detection algorithm DMIM is very sensitive of the assumptions. In both DM and DMIM, the goal is to find 1-1 mappings between the two packet streams. If there is an extra unmatched packet, the algorithm may produce incorrect result. One mismatched packet may cause all the subsequent packets to match incorrectly. The assumptions do not always hold due to corruption (such as packet loss). Hence, it is possible that an ATTACK connection is accidentally defined as a NORMAL connection due to violations of the assumption. The intruder may take advantage of this and add chaff packets to one of the stream to evade the detection. For these reasons, this paper further investigated the problem and introduced an improved algorithm in order to solve this problem.

3. Problem Definition

We first justify why traffic corruption becomes a problem in the detection process and then define the overall problem of tracing intrusion connections through stepping-stones.

3.1. Network Traffic Problem

The experimental results on the DMIM algorithm in [14] made the assumption that every packet in the incoming stream will appear on the output stream with some delay. If the collected data violated the assumptions (even if there is only one corrupted packet), the false negative detection may occur. Note that false negative detection means that an

ATTACK connection is mistakenly classified as a NORMAL one, and the false positive means a NORMAL connection is somehow declared as an ATTACK one.

According to our experiments, the assumptions can be easily violated under following two situations: *(n > m)* or *(n < m)* where the number of incoming packets is *n* and the number of outgoing packets is *m*. If the number of incoming packets is more than the number of outgoing packets *(n>m)*, it is impossible to map every incoming packet to one outgoing packet without violating the assumption that no outgoing packet can be mapped more than once. Even if the number of incoming packets is less than or equal to the number of outgoing packets *(n ≤ m)*, the assumption can still be violated if there is at least one incoming packet failing to find a mapping within the maximum tolerable delay Δ (user defined parameter). Such an example is illustrated in Fig. 1. Suppose there is a pair of incoming stream *X* and outgoing stream *Y*. Fig. 1a represents a stepping-stone (ATTACK) connection pair Fig. 1b represents a NORMAL connection pair due to more incoming packets (n = 5) than outgoing packets (m = 4); Fig. 1c represents the other NORMAL connection pair due to a long idle on the outgoing stream.

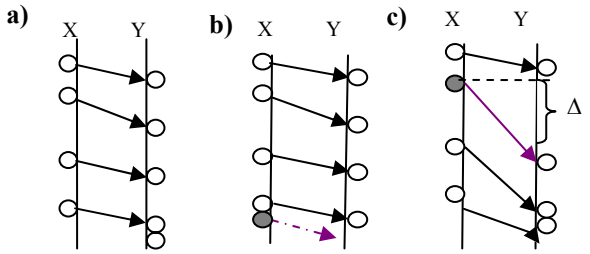

Fig. 1. a) ATTACK; b) NORMAL due to larger incoming packets size; c) NORMAL due to long delay

The corrupted traffic often occurs during packet lost, garbled data, congestion, buffer overrun, etc. For instance, suppose that a sender transmits a packet. From the sender's viewpoint, the sender does not know whether a data packet was lost, an acknowledgement segment (ACK) was lost, or if the packet or ACK was simply overly delayed.

We discovered that the packet retransmission is sometimes the reason to cause every incoming packet failing to map to its relayed outgoing packet within the maximum bounded delay in the NORMAL connection (failure of the 1-1 mapping). An example of such a failure is illustrated in Fig. 2. Suppose there are three hosts: A, D, B and we want to detect if the stepping-stone problem exists between host A and host B by installing a monitor on the middle host D. If an incoming packet sending from host A is somehow lost on its way to host D before passing to host B, the packet will be retransmitted after certain timeout interval. In other words, the same incoming packet is sent to the host D twice while host B only receives one outgoing packet. Since there are more incoming packets than the outgoing packets in the traffic flow, the monitor fails to find all the mappings. Even if the assumption is changed to map every outgoing packet to its relayed incoming packet, one-to-one mapping can still fail under certain situations such as ACK lost. If an outgoing packet sending from the host D is correctly arriving host B, host B then replies the ACK back to host D and the ACK is somehow lost on its way to host D, then host D will retransmit the packet after certain timeout interval even though

host B has successfully received the packet. In this case the same outgoing packet is received twice in host B while host A only sends it once.

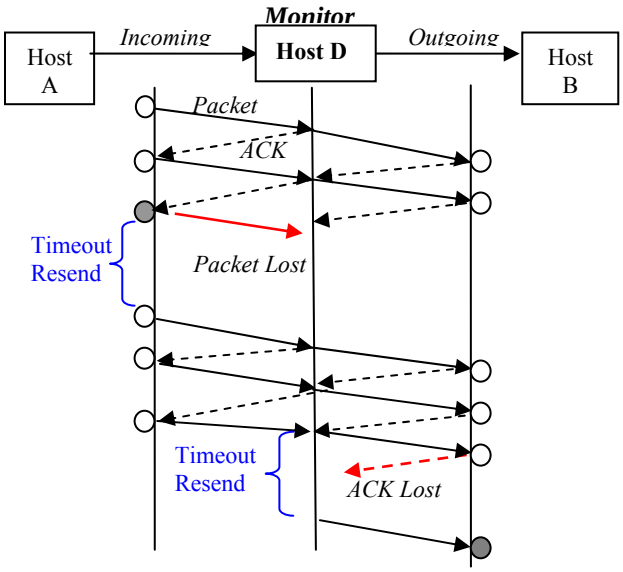

Fig. 1. Two examples where 1-1 mapping failed

In order to test the effectiveness of DMIM algorithm [14], an experiment that followed the same implementation and setting (capturing each of 80 packets from three incoming streams and three outgoing streams simultaneously) as in [14] has been done. Fig. 3 illustrated the results with one incoming streams mapping to all three outgoing streams. Note that only one of them is stepping-stone (ATTACK) connection. The time difference (or delay) is the incoming timestamp of the outgoing packet minus the timestamp of its relied incoming packet. Obviously, the delay between each pair of stepping-stone packets should be relatively small (less than 0.1 seconds from incoming packets 1 to 71) compared with the normal pair. However, the failure occurred when the collected data was somehow corrupted (circled). In this attack connection, the difference between the 72nd incoming packet and its corresponding outgoing packet was approximately 14 seconds. So, in order to avoid any miss detection, the threshold that used to declare the ATTACK and NORMAL must be set to no less than 14 seconds. However, the NORMAL connection 2 would be defined as ATTACK since all its time difference were smaller than 14. On the other hand, the threshold that used to declare the ATTACK and NORMAL must be set no more than 0.1 seconds to avoid the false alarm. Hence, it is possible that DMIM may failure to find all the attack mappings.

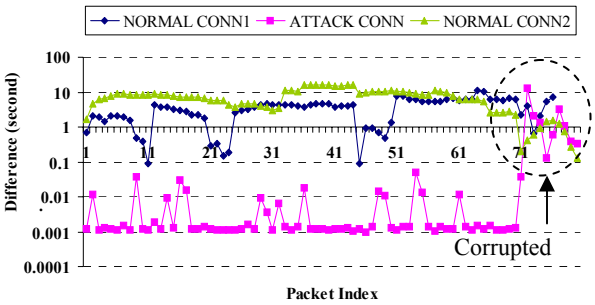

Fig. 3. An example with corrupted data

As we discussed earlier, corrupted traffic may occur due to a various number of reasons. There is really no way to prevent them from happening especially the traffic flow may go through several intermediate hosts before being collected. Hence, we concluded that DMIM algorithm in [14] will cause the false detection whenever the collected data is corrupted if the assumption is removed. Although the network traffic does not corrupt every time, this situation does happen. This paper introduces a modified algorithm which overcomes the problem of corrupted traffic later on.

3.2. Attack Definition

The definition in [14] must be modified to solve the problem of detecting the attack from encrypted and interactive stepping-stone connections. We assume that it is impossible to have a long idle between each pair of the related packet. In addition, most of the incoming packets can form a pair with their related outgoing packets in a stepping-stone connection. As a result, a maximum bounded delay (denotes as Δ) and a threshold (denotes as ρ) have been used to reduce the problem to match as many arriving packets with relayed departing packets within the provided Δ as possible. And then the mismatched rate will be used to compare with the given ρ.

Suppose the packets arrive on both incoming stream $X = \{x_1, \ldots, x_n\}$ and outgoing stream $Y = \{y_1, \ldots, y_m\}$, of the same host. The i^{th} incoming timestamp of incoming stream X is defined as x_i ($i \geq 1$). Similarly, y_i ($i \geq 1$) is the i^{th} incoming timestamp of outgoing stream Y. In addition, we assume the timestamps of both incoming stream X and outgoing stream Y are sorted in increasing order. Hence, x_i always arrives before x_{i+1} which denotes by $x_i < x_{i+1}$ and $y_i < y_{i+1}$. We present a new definition for ATTACK connection which tolerates some corrupted packets.

Definition 1: *Given a pair of packet streams X and Y, a maximum tolerable delay Δ and an acceptable mismatched rate ρ, streams X and Y forms a stepping-stone ATTACK connection if there exists a subset U of X and a one-to-one function $f:(X-U) \rightarrow Y$ such that $0<|f(x)-x| \leq \Delta$ for all $x \in X-U$ and $|U|/|X| \leq \rho$.* Otherwise, *X and Y are a NORMAL connection*.

The definition allows a subset of the incoming packets to be excluded from matching with the outgoing packets. Clearly this new definition is a generalization of the previous definition by simply letting U to be an empty set.

Given that searching for all the possible mappings require exponential complexity, the *order-preserving concept* (introduced in [8, 13]) is used to limit the search by assuming the mappings are sorted in an increasing order $f(x_i) \le f(x_j)$ *for all* $x_i \le x_j$ *in X.* Although it is possible that the attackers intentionally change the order of packets arriving to the host, delaying a packet will normally cause the delay for all its subsequent packets. In this case, the complexity is reduced when searching for the valid mappings. The problem is then further reduced into matching each incoming packet with its *earliest* correlated departing packet which purposed in [14].

After finding all the possible mappings, a mismatched rate can then be computed by dividing the number of incoming mismatched packet to the total number of incoming packets. Because the corrupted or chaffed traffic does not occur every time, the mismatched rate of normal connection should be lower than the stepping-stone connection. Consequently, the decision of whether a connection should be classified as a stepping-stone connection or as a normal one depends on the computed mismatched rate and a given acceptable mismatched rate ρ. The connection is a stepping-stone ATTACK connection if the mismatched rate is smaller than or equal to ρ. Otherwise, it is defined as a NORMAL one.

4. Algorithm

The purpose of this Algorithm is to detect whether a pair of the incoming stream X and the outgoing stream Y forms the stepping-stone connection using our new definition above.

Given:

- *X[1..n], incoming packets in an increasing order,*
- *Y[1..m], outgoing packets in an increasing order,*
- *Δ, maximum tolerable delay,*
- *ρ, acceptable mismatched rate,*
- *n, number of packets in stream X,*
- *m, number of packets in stream Y.*

Return: ATTACK, if the ratio of mismatched packets is smaller than or equal to ρ, NORMAL, otherwise.

Table 1. Algorithm DM2

```
DM2(X, Y, m, n, Δ, ρ)
 Queue QX,QY;
 int i, j, k;
 int U = 0; //# of unmatched packets

 for i = 1 to n do   QX.enqueue(X[i]);
   for j = 1 to m do   QY.enqueue(Y[j]);

 x = QX.dequeue();
 y = QY.dequeue();
 while(!QX.isEmpty()))do
  if (QY.isEmpty())
```

```
    U++;
    x = QX.dequeue();
  else if (y-x ≤ 0)  //y is of no use
    y = QY.dequeue();
  else if (y-x > Δ) //no y in range
    U++;
    x = QX.degueue();
  else //found a y to match x
    x = QX.dequeue();
    y = QY.dequeue();
  end
end

if ((U/n) ≤ ρ) return ATTACK;
else return NORMAL;
```

The complexity of Algorithm DM2 depends on the number of incoming and outgoing packets. A careful study proves that each packet in an incoming stream X and an outgoing stream Y enters into the queues once and removed only once. Hence, the time complexity of the algorithm is $O(m + n)$.

Unlike the DMIM algorithm [14], algorithm DM2 has a weaker restriction and therefore has the ability to tolerate the corrupted traffic flow. Other than that, DM2 maintains most of the properties of DMIM. An example is shown in Fig. 4 (the same capturing data was used as in Fig. 3). Clearly, if the difference between any pair of incoming packet and outgoing packet is larger than Δ, it will be considered as a mismatch. According to the Fig., all the differences except the 72nd incoming packet are smaller than 0.1 seconds in the ATTACK connection and almost all the differences in both NORMAL connections are larger than 0.1 seconds. Thus, the maximum delay that used to declare the ATTACK and NORMAL should be set no less than 0.1 seconds. By setting Δ =0.1, the mismatched rates for both NORMAL CONN1 and CONN2 would be 98% and 99%, while the ATTACK Connection had only 1% mismatched rate. Since there is a huge gap in between (1%~98%), it is very easy to choose an appropriate ρ. It is worth noting that the effect of a corrupted packet (index number 72 in Fig. 4) does not propagate to the subsequent packets (compared to Fig. 3).

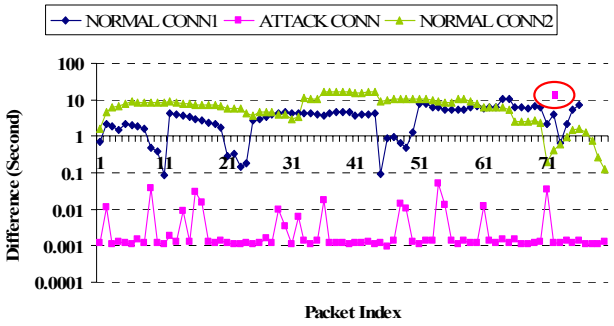

Fig. 4. An example of using DM2 algorithm with corrupted data (Δ = 0.1 seconds)

Theorem 1: *Given an incoming stream X and an outgoing stream Y where X and Y are stepping-stone connection pair satisfying the giving the maximum tolerable delay Δ, and a separate set of packet C, there exists a ρ such that DM2 will return ATTACK if we add C to X.*

Proof: Since inserting a packet (corrupted or chaff) into the incoming stream will only result in finding equal or more mappings, the number of matched pairs will be non-decrease. Note: we did not claim that all the matches remained the same. It is likely that some of the mapping will change. Let's select ρ = |C| /(C| +|X|). So the number of unmatched packets |U| can only be equal or less than the number of inserted chaff packets |U| ≤ |C|. Because |U| ≤ |C|, so |U| / (|C| +|X|) can only be less than or equal to |U| / (|C| +|X|). Therefore,

$$\rho = |C| /(|C| +|X|) \geq |U| /(|C| +|X|).$$

The worst case for inserting |C| extra packets is none of the extra packets finds the mapping and returns to the missed rate |U| /(|U| +|X|). Hence, as long as ρ ≥ |U| /(|U| +|X|), the algorithm will always return ATTACK. However, even it is very unlikely, but we still have to admit that there is a very small chance to misidentify a NORMAL stream pair as an ATTACK pair if the NORMAL stream pair is accidentally very similar to each other (false positive). The mapping has the minimum indices among all such mappings like in [14].

Corollary 1: *If Algorithm DM2 returns NORMAL, then no such mapping exists.*

The Corollary can be proved straightforward from Algorithm DM2. Since we have proved that the algorithm will always return ATTACK if two streams are really stepping-stone pair, the only possible situation for the algorithm to return NORMAL is to insert two streams which are not stepping-stone connection pair.

5. Experiments and Performances

We conduct experiments to answer the following questions: what are reasonable value for the parameters Δ and ρ, the number of packets for the detection to be effective, and how much chaff can the algorithm tolerate.

We implemented our algorithm DM2 with various parameter values during the stepping-stone detection. These parameters are: maximum tolerable delay *(Δ)*, the number of incoming packets *(n)*, and the acceptable mismatched rate *(ρ)*. We are interested in finding the appropriate values for each of the parameters though the experiments. Moreover, it is very important to study the reactions of our algorithm when some chaff packets are added into the stepping-stone connections.

In order to answer all the questions above, we first collected the data flow on this host which contained 50 incoming streams and 50 outgoing streams for the experiments. Of the 50X50 combinations, there are 50 stepping-stone ATTACK connections and the rest are NORMAL connections.

5.1. Maximum Tolerable Delay

Maximum tolerable delay *(Δ)* is one of the user defined parameters. The delay between each stepping-stone pair would be relatively small comparing with the normal pair. In other words, a stepping-stone connection has the smaller delay for all matched pairs in general, and the number of mismatched packets is relatively lower. Since Maximum tolerable delay affects the mismatched rate (the smaller the Δ is, the higher the number of mismatched packets is) and then the mismatched rate affects whether the connection is distinguished as ATTACK or as NORMAL, the accuracy of our algorithm indirectly relies on Δ. Consequently, how to define a maximum delay value is very important.

The first part of the experiment modified Algorithm DM2, so it computed the delay of every pair. Basing on the results, a range of maximum delay could be generated as the sample for user's consideration. This experiment examined 50 pairs of ATTACK streams and randomly selected 50 pairs of NORMAL streams with 100 packets in each connection. We then analyzed the results using descriptive statistics. A box plot, which depicted such analysis, is represented in Fig. 5. The minimum difference for the ATTACK pairs and the NORMAL pairs are 0.0005 seconds and 0.00004 seconds. The lower quartile for the ATTACK pairs and the NORMAL pairs are 0.0041 seconds and 0.9417 seconds. The median for the ATTACK pairs and the NORMAL pairs are 0.0041 seconds and 3.3515 seconds. The higher quartile for the ATTACK pairs and the NORMAL pairs are 0.0063 seconds and 8.8255 seconds. The maximum differences for the ATTACK pairs and the NORMAL pairs are 59.0150 seconds and almost 60.4372 seconds. Since any data observation which lied lower than the first quartile or higher than the third quartile was considered as outliers (the most frequent differences fell into the range of lower quartile and higher quartile), setting the maximum tolerable delay *(Δ)* within the range of 0.01 seconds to 1.00 second should help the algorithm detect most of stepping-stone attack.

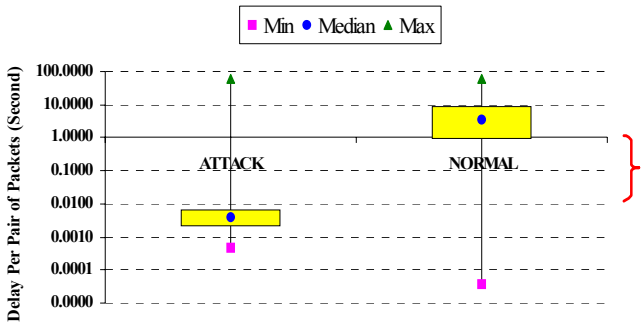

Fig. 5. Distribution of the packet delay times

We further investigated the range of the maximum delay. The second experiment examined 50 pairs of ATTACK streams and 2450 pairs of NORMAL streams in each connection. We used 0.02 to 0.10 seconds as the maximum delay with an increment of 0.02 seconds to test our algorithm. The results are presented in Fig. 6. Each mismatched rate was averaged by the number of total ATTACK or NORMAL streams. We could see the number of mismatched rate decrease when the value of the maximum delay increases for both ATTACK and NORMAL streams. The results also clearly represented the average gaps between ATTACK streams and NORMAL streams. According to Fig. 6,

even setting the smallest maximum delay to 0.02 seconds and number of packets n = 100, the difference (0.7144%) between the mismatched rate for both ATTACK and NORMAL was still large enough to safely distinguish ATTACK pairs from NORMAL pairs. Hence, maximum tolerable delay within a range between 0.01 seconds and 1.00 seconds is acceptable.

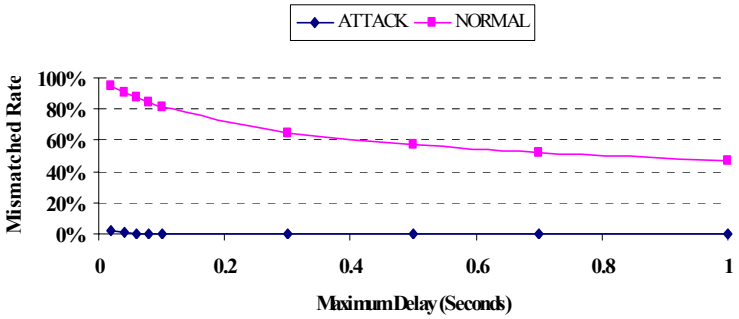

Fig. 6. Different Maximum Delay (n = 100)

5.2. Number of Incoming Packet

Given enough number of incoming packets, Algorithm DM2 should be able to properly declare ATTACK connection and NORMAL connection. The smaller the number of incoming packet is, the smaller the denominator of the mismatched rate is and the higher the mismatched rate may end up (especially for ATTACK connection). This means the false positive rate would be huge when *n* was too small. On the other hands, if *n* was too large, algorithm DM2 would take much longer time to compute the result. Hence, it is very important to define an appropriate *n* value.

Again, this part of the experiment examined 50 pairs of ATTACK streams and 2450 pairs of NORMAL streams in each connection. Of the results produced on the various numbers of incoming packets, it is clear that when the number of incoming packet increases, the mismatched rate also increases. According to the results in Fig. 7, even by using 10 incoming packets (n=10), the difference (0.7144%) between the mismatched rate for both ATTACK and NORMAL was still large enough to safely distinguish ATTACK pairs from NORMAL pairs.

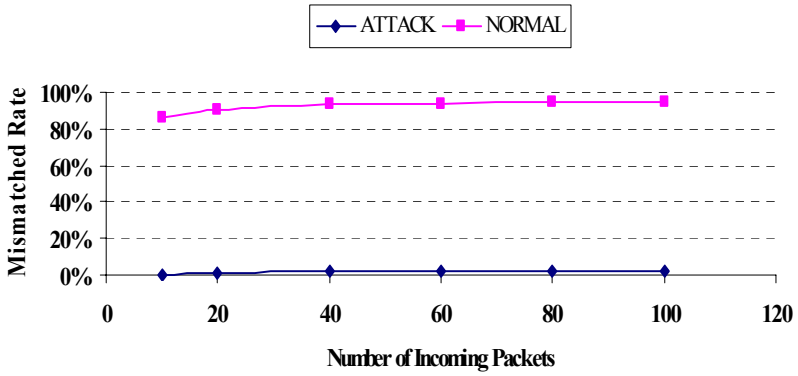

Fig. 7. Mismatched Rate using different amount of incoming packets (Δ = 0.02 seconds)

5.3. Threshold

The acceptable mismatched rate ρ is also one of the user defined parameters in algorithm DM2. Such value also needed to be known in prior. The mismatched rate for each pair of stepping-stone streams would be relatively small if we compared it with the normal pair and it affected whether the connection is distinguished as ATTACK or as NORMAL. As a result, the accuracy of our algorithm directly depends on ρ.

The goal of this part of the experiment was to find a range of reasonable ρ which could be used for help the user to define a correct value. By doing so, we computed the difference of the mismatched rate (or so called gap) between ATTACK and NORMAL connections and analyzed all the gaps for each different maximum delay (Δ=0.02~0.10 seconds) while number of the incoming packets is 20 (n = 20) in Fig. 8. Clearly, the minimum difference dropped when the mismatched rate increased. Each of the gaps was represented by the box. Therefore, there was a very strong negative correlation between mismatched rate and the minimum gap. However, the range of the gap was still no more than 20% difference.

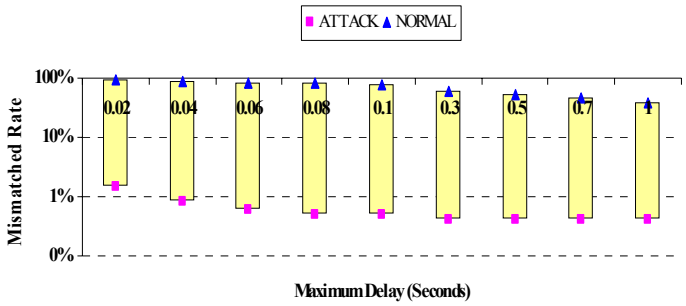

Fig. 8. Minimum Gap

5.4. Chaff

Although the pervious experiments returned the good performance, they were done without any chaff involved. But as a matter of fact, the intruders might inject some chaff packets into ATTACK traffic in order to help increase the possibility of escaping from the detection. For this reason, we are interested in examining the reactions when inserting some chaff into the traffic and checking our combined algorithm DM2 which can still distinguish stepping-stone ATTACK connection and NORMAL connection while tolerating certain amount of chaff.

In the following experiment, we tested the performances of using algorithm DM2 when the chaff packets were involved by using the same 50X50 combinations setup as described in previous sections. Except in this experiment, we also inserted chaff packets at different chaff rates *c* in each of the streams on either side. Note: suppose we insert the chaff packets on incoming stream, chaff rate is the number of chaff divided by number of incoming packets. These chaff packets were generated by Poisson distribution and added to the incoming packets collected from the experiment. This experiment was done by using 0% to 400% chaff rate with an increment of 50% at a time, and the number of the

testing packets was 20 (n = 20). Taking Δ = 0.02 as an example, Fig. 9 clearly showed that NORMAL connections had the constant 90% mismatched rate. However, while chaff rate increased from 50% to 400%, the ATTACK connection had increasing the mismatched rate rapidly (up to 80%). When inserting the chaff packets on the incoming stream, the ATTACK connection had the consistent mismatched rate of approximately 1% When inserting the chaff packets on the outgoing stream, we can conclude that no matter inserting chaff packets into incoming streams or outgoing streams, there was always enough gap to distinguish ATTACK and NORMAL connections for adding up to 400% additional chaff packets.

We further investigated the performance while tolerating certain amount of chaff on both incoming and outgoing streams. The performances were again presented in Fig. 9. Even though the mismatched rate of ATTACK connection raised when the chaff rate increased, the mismatched rate would never cross 40% when the chaff rate was up to 400%. Therefore, this experiment results still showed there was enough gap (at least 50% difference) to distinguish both ATTACK and NORMAL connections for up to 400% additional chaff packets on both incoming and outgoing streams.

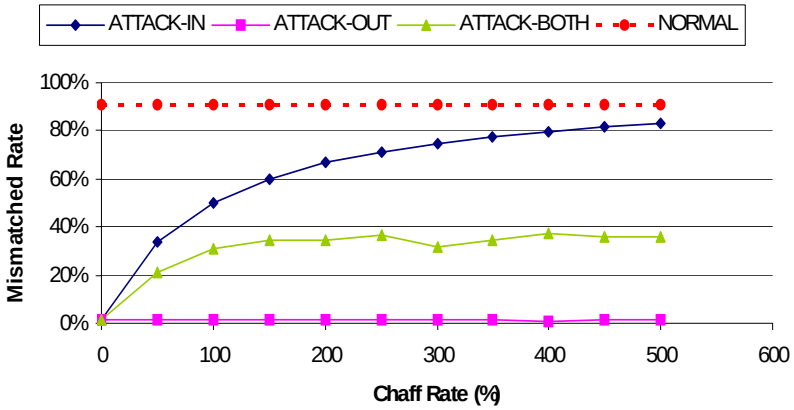

Fig. 9. Mismatched Rate when adding Chaff on the streams (n = 20 and Δ = 0.02seconds)

5.5. Performance Evaluation

In this section, we evaluate our algorithm performance with detection false positive rate (FPR) and the false negative rate (FNR) by using 50 pairs of ATTACK streams and 50 pairs of NORMAL streams with 20 packets (n=20) each. Note: the false positive rate represents the probability of incorrectly identified normal pairs and the false negative rate represents the probability of incorrectly identified attack pairs. We examine the performance of DM2 with different threshold (ρ) and the performance of DM2 with different maximum tolerable delay (Δ). Both of the performance results are illustrated in Table 2 below. This experiment reports a very good performance with low false detection rates when the right parameter values are selected (discussed in section 5.1~5.4). Note: it is impossible to compare our algorithm performance to other algorithms due to the fact that other algorithms are either having different restrictions or

only having theoretical algorithms without the experimental results and performance data.

Table 2. a) Performance of DM2 with different **ρ;**
b) performance of DM2 with different Δ **(n=20)**

ρ	DM2 Δ=0.02s	
	FPR	FNR
0.05	0%	8%
0.10	0%	2%
0.20	0%	0%
0.40	0%	0%
0.60	0%	0%

Δ sec	DM2 ρ=0.2	
	FPR	FNR
0.02	0%	0%
0.04	0%	0%
0.06	0%	0%
0.08	0%	0%
0.10	0%	0%

6. Conclusions

Network intrusion detection is still an infant field of research. However, it is beginning to assume the great significance in today's computing environment. The combination of facts, such as the intemperate growth of the Internet, the global financial possibilities opening up in electronic trade, etc. makes NIDS an important and pertinent field of research. In reality, network traffic may often be corrupted or chaffed and such corrupted or chaffed traffic causes some of the previous detection algorithms such as [7, 12, 14] to fail. Due to such significance, we propose and analyze an algorithm for encrypted stepping-stone detection. We also provide a proof of the correctness of our algorithm.

We use several factors, such as maximum delay, number of incoming packets, and the acceptable mismatched rate to measure the effectiveness of our detection algorithm. Moreover, this paper examines the performance of our algorithm when the network traffic is somehow corrupted or chaffed. According to the experimental results, the performance of our algorithm is very good and it is highly accurate in detecting the stepping-stone connection. Our algorithm can still detect the ATTACK connections correctly even when inserting the chaff rate up to 400% on incoming stream, outgoing stream or both incoming and outgoing streams.

Acknowledgement. Support of this research under a grant from Texas Learning and Computation Center (TLC2) is acknowledged.

References

[1] Yang, J and Huang, S., "Matching TCP packets and its application to the detection of long connection chains on the internet," in Proc. of the 19th International Conference on Advanced Information Networking and Applications, 2005, pp. 1005-1010.

[2] Zhang, Y. and Paxson, V., "Detecting Stepping Stones", in Proc. of the 9th USENIX Security Symposium, Denver, CO, 2000, pp. 171-184.

[3] Yung, K. H. "Detecting long connection chains of interactive terminal sessions," in Proc. of 5th International Symposium on Recent Advances in Intrusion Detection (RAID), Lecture Notes in Computer Science, 2002, pp. 1-16.

[4] Wang, X., Reeves, D., and Wu, S,. "Inter-Packet Delay-Based Correlation for Tracing Encrypted Connections through Stepping Stones", In D. Gollmann, G. Karjoth and

M. Waidner, editors, 7th European Symposium on Research in Computer Security (ESORICS), 2002.
[5] Blum A., Song D. and Venkataraman S., "Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds," in Proc. of 7th International Symposium on Recent Advances in Intrusion Detection (RAID), Springer LNCS 3224, 2004, pp. 258-277.
[6] Donoho, D., Flesia, A.G., Shankar, U., Paxson, V. and Staniford, S., "Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay", 5th International Symposium on Recent Advances in Intrusion Detection, 2002.
[7] He, T. and Tong, L., "Detecting Encrypted Stepping-stone Connections", IEEE Trans. on Signal Processing, Vol. 55, No. 5, 2007, pp. 1612-1623.
[8] Wang, X. and Reeves, D., "Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Inter-packet Delays", in Proc. of ACM Conference on Computer and Communications Security, ACM Press, 2003, pp. 20-29.
[9] Yang, J and Huang, S., "A real-time algorithm to detect long connection chains of interactive terminal sessions," in Proc. of the 3rd international conference on Information security. ACM Press, 2004, pp. 198-203.
[10] Wu, H and Huang, S, "Detecting Stepping-Stone with Chaff Perturbations," in Proc. of Symposium on Frontiers in Networking with Applications, 2007.
[11] Yoda, K. and Etoh, H., "Finding a Connection Chain for Tracing Intruders", 6th European Symposium on Research in Computer Security (ESORICS), 2000, pp. 31-42.
[12] He, T. and Tong, L.,"A Signal Processing Perspective to Stepping-stone Detection"', in Proc. of IEEE CISS, Princeton, NJ, 2006.
[13] Zhang, T., Persaud, A, Johson, A, and Guan, Y, "detection of Stepping Stone Attack under Delay and Chaff Perturbations," in Proc. of the 25th IEEE International Performance Computing and Communications Conference (IPCCC), Phoenix, AZ, 2006.
[14] Kuo, Y. and Huang, S., "Stepping-Stone Detection Algorithm based on Order Preserving Mapping", to appear in Proceedings of the 13th International Conference on Parallel and Distributed Systems (ICPADS), Hsinchu, Taiwan, December, 2007.