Matching TCP/IP Packets to Resist Stepping-Stone Intruders’ Evasion

Ni Long, Jianhua Yang, David Y. Song
1Department of Electrical and Computer Engineering, North Carolina A & T University
E-mail: {nilong, songyd}@ncat.edu
2Department of Math and Computer Science, Bennett College for Women
jhyang@bennett.edu

Abstract

Most network intruders tend to use stepping-stones to attack or to invade other hosts to reduce the risks of being discovered. There have been many approaches proposed to detect stepping-stone since 1995. One of the most commonly used methods proposed by Blum is to detect stepping-stone by checking if the difference between the number of the send packets of an incoming connection and the one of an outgoing connection is bounded. However, this method is weak in resisting to intruders’ evasion, such as chaff perturbation. In this paper, we analyze the resistance of packet matching approach to intruders’ evasion. The theoretical analysis shows that packet matching method is more effective and robust than most existing approaches in resisting to intruders’ chaff perturbation and time jittering evasion.

Keywords

Network security, stepping-stone, intrusion detection, chaff perturbation, time jittering, evasion

1. Introduction

Stepping-stones [YZ00] are popularly used by intruders to invade or to attack other computers because the attackers’ identity could be hidden through a long connection chain [KY02]. Detecting such kind of intrusion is difficult because of the inborn designing flaw of TCP/IP protocol in which one computer in a TCP/IP session is transparent only to its downstream and upstream neighbors. Many approaches have been proposed to detect stepping-stone intrusion since 1995, such as Content-based Thumbprint [SC95], Time-based Approach [YZ00], Deviation-based Approach [KY00], Round-trip Time Approach [JY05, KY02], and Packet Number Difference-based Approach [DD02, AB04].

Staniford-Chen and Heberlein proposed a method that identifies intruders by comparing different sessions for suggestive similarities of connection chains [SC95]. The fatal problem of this method is that it cannot be applied to encrypted sessions because their real contents are not available and thus are unable to make thumbprint. The time-based approach proposed by Zhang and Paxson [YZ00] can be used to detect stepping-stones or to trace intrusion even if a session is encrypted. However, there are three major problems in the time-based approach. First, it can be easily manipulated by intruders. Second, the method requires that the packets of connections have precise and synchronized timestamps in order to correlate them properly. This makes it difficult or impractical to correlate the measurements taken at different points in the network. Third, Zhang and Paxson also were aware that a large number of legitimate stepping-stone users routinely traverse a network for a variety of reasons. The deviation-based approach proposed by Yoda and Etoh [KY00] is a network-based correlation scheme. It defines the deviation as the minimum average delay gap between the packet streams of two TCP connections. This method is based on the observation that the deviation for two unrelated connections is large enough to be distinguished from the deviation of connections in the same connection chain. In addition to the problems the time-based approach has, the deviation-based approach has the following additional problems. 1) computing deviation is not efficient; 2) it is not applicable for a compressed session because it depends on the size of a packet; 3) it cannot correlate connections where padding is added to the payload because it can correlate only the TCP connections that have one-to-one correspondences in their TCP sequence numbers; 4) correlation measurements are applicable only to the post-attack traces because the correlation metrics are defined over the entire duration of the connections. The round-trip time (RTT) approach proposed by Yung [KY02] detects stepping-stone intrusion by estimating the downstream length using the gap between a request and its response, and the gap between the request and its acknowledgement. The problem of the RTT approach is that it makes inaccurate detection because it cannot compute the two gaps accurately.

The packet number difference-based (PND-based) approach proposed by Blum [AB04] detects stepping-stones by checking the difference of Send packet numbers between two connections. The method is based on the idea that if the two connections are relayed, the difference should be bounded; otherwise, it should not. It is claimed
that this method can resist intruders’ evasions such as time jittering and chaff perturbation to an extent. Donoho et al. [DD02] showed for the first time that there are theoretical limits on the ability of attackers to disguise their traffics using evasions during a long interactive session. They proved that detecting stepping-stone is still possible by monitoring a session long enough even if the session is jittered by time and chaff perturbation by using wavelet and multi-scale methods. However, Donoho et al. did not point out how long a session needs to be monitored in order to detect a stepping-stone. Blum [AB04] continued Donoho’s work and proposed a PND-based algorithm for stepping-stone detection using Computational Learning Theory. Blum achieved provable upper bounds on the number of packets required to be monitored in an interactive session in order to get a given confidence. The major problem with the PND-based approach is due to the fact that the upper bound on the number packets required to monitor is large, while the lower bound on the amount of chaff an attacker needs to evade his detection is small. This fact makes Blum’s method very weak in resisting to intruders’ chaff evasion.

The method, matching TCP/IP packets to detect stepping-stone intrusion, was proposed in 2005 [JY05]. In 2006, and 2007, some other methods (partitioning-clustering) used to match TCP/IP packets are published [JY06, JY07]. Their main idea is to detect stepping-stone intrusion through estimating the length of connection chain between the stepping-stone and the victim host. The longer the connection chain, the higher the probability that the user is an intruder. We also found that those packet matching methods can also be used to detect if a computer is used as a stepping-stone. The more important thing is that we found that packet matching methods are stronger than other methods mentioned above in resistance to intruders’ evasion. An intruder usually manipulates a connection chain through time jittering and chaff perturbation to evade stepping-stone intrusion detection. The ability in resisting to intruders’ evasion is an important benchmark in evaluating the performance of algorithms to detect stepping-stone. In this paper, we will analyze how packet matching approaches could resist intruders’ time jittering and chaff perturbation evasion.

The rest of this paper is arranged as following. In Section 2, we present the problem statement. Section 3 presents the ability of packet matching method in resisting to intruders’ evasion. Finally, in Section 4, the whole work is summarized, and the future work is presented.

2. Problem Statement

The basic idea of detecting a host or a network of computers used as a stepping-stone is to compare an incoming connection with one of the outgoing connections. If they are relayed, we call them a stepping-stone pair; otherwise, a normal pair. As Fig.1 shows, host \( h_i \) has one incoming connection \( C_i^1 \) and one outgoing connection \( C_i^2 \), while each connection has one request stream and one response stream. If we make the following assumptions, then in a period of time, the number of packets monitored in each connection should be close if the two connections are relayed:

1) Each packet appearing in one connection must appear in its relayed one;
2) An intruder could hold any packet at any place, but the holding time has an upper bound;
3) An intruder could insert meaningless packets into an interactive session at any time, but the inserting rate is bounded.

The assumption 1) means that there are no packet drops, combinations, or decomposing. It guarantees that the number of the packets in an incoming connection must be greater than or equal to the number of the packets in the relayed outgoing connection. If two connections are relayed, we can at least find a relationship between the number of the requests of the outgoing connection and the number of responses of the incoming connection. The problem of detecting stepping-stones becomes the problem of finding a correlation between the number of requests and the number of responses. The assumption 2) comes from the fact that each user has a time tolerance of using an interactive session; and the assumption 3) indicates that the rate in which a user can insert packets into an interactive session is bounded.

![Fig.1. Illustration of connections and streams of a host](image)

From the above assumptions, we know that if two connections are relayed, there should be a suggestive relationship between the requests and responses. We can use the existence of this relationship to determine if two connections are in the same chain. Our statement is that it is possible to detect stepping-stone by matching the Sends (requests) in an outgoing connection with the Echoes (responses) in an upstream connection. In other words, it is possible to detect stepping-stone intrusion by matching TCP/IP send and echo packets.

Most approaches detecting if host \( h_i \) is used as a stepping-stone focus on comparing the numbers of send packets \( S_i^{(1)} \) and \( S_i^{(2)} \). The closer the two numbers, the higher probability this host is used as stepping-stone. The bad news is that an intruder could hold some send packets for...
a while, or add some meaningless packets to the incoming or outgoing connection chain to make the two numbers are unrelated to evade such kind of detection. Most approaches are vulnerable in resisting to intruders time jittering and chaff perturbation. We found that packet matching approaches can overcome this difficulty. It is stronger than other approaches in terms of resisting intruders’ evasion.

3. Resistance Analysis to Intruders’ Evasion

3.1 Time Jittering and Chaff Perturbation

Intruders can evade detection by holding send packets of a session for a while. This is called time jittering evasion. Each packet is held for different time gaps. Usually the intruder randomly generates time gaps for the send packets held. But the send packets order must be guaranteed. Suppose we have \( n \) send packets: \( \{ s_1, s_2, s_3, \ldots, s_n \} \), their corresponding time stamps are: \( \{ t_1, t_2, t_3, \ldots, t_n \} \). The following relations must be satisfied if these packets belong to one interactive session,

\[
t_n > t_{n-1} > \ldots > t_3 > t_2 > t_1
\]

If the \( i^{th} \) packet is held for time gap \( \Delta t_i \), the time stamps of the jittered send packets would be: \( \{ t_1 + \Delta t_1, t_2 + \Delta t_2, t_3 + \Delta t_3, \ldots, t_i + \Delta t_i, \ldots, t_n + \Delta t_n \} \). Whatever how much the \( \Delta t_i \) is, the following relations must be satisfied to guarantee the order,

\[
t_n + \Delta t_n > t_{n-1} + \Delta t_{n-1} > \ldots > t_3 + \Delta t_3 > t_2 + \Delta t_2 > t_1 + \Delta t_1
\]

This is required by TCP/IP protocol. Many stepping-stone detection approaches [YZ00, SC95, KY00] are vulnerable to this time jittering evasion.

Another way used by intruders to evade detection is to insert meaningless packets to an interactive session. This is called chaff perturbation. With chaff perturbation, intruders can make two relayed connections unrelayed, or two unrelayed connections relayed. The methods to detect stepping-stone intrusion by counting the number of send packets fail to resist this chaff evasion because the packets number could be easily changed with chaff perturbation. Chaff is more difficult to implement than time jittering. We emphasize two points about chaff perturbation. One is chaff rate; another one is chaff removing.

Intruders usually do not insert too many packets into an interactive TCP/IP session. One reason is that it is too difficult to control by inserting too much packets, another one is that it is inefficient. The purpose of chaff perturbation is to evade detections which compare the number of send packets between incoming and outgoing connections to see if the two numbers are close enough. In other words, it checks if the difference of the two numbers is within \( \varepsilon \), where \( 1>\varepsilon>0 \). If we use \( \Delta N \) to represent the difference, then the following equation must be satisfied,

\[
|\Delta N|<\varepsilon
\]

The smaller the \( \varepsilon \), the more accurate the detection. If an intruder wants to evade the detection, he (she) just chaffs the session with rate little more than \( \varepsilon \). Our conclusion is that intruders do not have to insert too many packets into a session to evade the detection.

Another important thing is that intruders must remove all the chaffs before they go to the end host of an interactive session. The reason is that the meaningless packets cannot be executed at the end host. More serious thing is that if the chaffs go to the end host, they will interfere to the execution of the normal packets. Our conclusion is that all the chaffs do not have any echo packets.

3.2 Packet Matching Idea

One or more send packets in one session can be echoed by one echo packet in the same session. Packet matching is to find which echo packet is used to respond to which send packets. Simply to say, it is to pair the send and echo packets in the same chain. If we can match send and echo packets in different connections, we definitely say that the two connections are relayed, and further that the host is used as a stepping-stone. The problems of packet matching are inherited from the fact that the Send and Echo packets may be a many-to-many relationship, not one-to-one. It is impossible to match them deterministically even with a complete log. Therefore, it is much more difficult to match in real-time. To be noted is that if we make a mistake in packet matching at one point of a packet stream, the mistake would affect all other packet matches after that point. To prevent this kind of mistake from occurring, the policy used is to limit the possible mistakes in a certain range. By dividing a packet stream into sub-streams, which are the scopes in which we match the packets. Thus, if we made a mistake, it would only affect the packet matching within that sub-stream. Another benefit of dividing a packet stream into sub-streams is that we are relatively confident that in each sub-stream, there is one and at least one matched packet pair. The only thing left is how to divide a packet stream into sub-streams online. Usually the fact used is that an intruder would need to think about and pause prior to each step; consequently, there exists a gap, which is more than a predefined threshold, between the keystroke right before the thinking-pause-point and the keystroke right after that point. There are already many methods proposed to detect stepping-stone intrusion [JY05, JY05-1, JY06]. The Conservative matching algorithm [JY05] only matches the packets that we are relatively sure about their matching correctness in each sub-stream, while it discards the others. Otherwise, once we are not
sure about the packet matching correctness, we use the chronicle, sequence number, and size of a packet in a stream to decide packet matching, using the Greedy matching algorithm [JY05]. Other than the previous two methods to match TCP/IP packet locally, the clustering-partitioning method [JY06] can match send and echo packets globally. In this paper, we focus on analyzing how the packet matching methods resist intruders’ time jittering and chaff perturbation evasion. This point was not mentioned in the above papers.

### 3.3 Resistance to Time Jittering Evasion

As Figure 1 shows, an intruder could perform time jittering either at incoming connection by holding $S^{(i)}$ or at outgoing connection by holding $S^{(o)}$. Usually an intruder cannot hold echo packets. The reason is that each echo packet is the response of a send packet. If an echo packet is held for a while, it causes packet resent which complexes the network communication. Based on our investigation, observation, and technical possibilities, holding send packets is a usual and practical way for intruders to manipulate an interactive session.

To simplify our analysis, we assume that the send packets $S^{(i)}$ in outgoing connection are held randomly. Before time jittering is performed, we assume there are $n$ send packets $S: \{s_1, s_2, s_3, \ldots, s_n\}$ with time stamps $\{t_{s1}, t_{s2}, t_{s3}, \ldots, t_{sn}\}$ passing through the outgoing connection. Correspondingly, we also assume that there are $m$ echo packets $E: \{e_1, e_2, e_3, \ldots, e_m\}$ with time stamps $\{t_{e1}, t_{e2}, t_{e3}, \ldots, t_{em}\}$ coming from end host of the session at the incoming connection. We also assume that $s_i$ with time stamp $t_{si}$ matches $e_j$ with time stamp $t_{ej}$. After time jittering, the two time stamps should become $t_{si} + \Delta t$ and $t_{ej} + \Delta t$, where $\Delta t$ is the time jittering, if the two connections are relayed. The round trip time before time jittering is $\text{RTT1} = t_{ej} - t_{si}$, and the round trip time after time jittering is $\text{RTT2} = (t_{ej} + \Delta t) - (t_{si} + \Delta t) = t_{ej} - t_{si}$. We found that time jittering does not affect the round trip time which is used to match send and echo packets. Reversely, we can say that stepping-stone detection through packet matching cannot be affected by time jittering. Packet matching to detect stepping-stone intrusion can resist intruder’s time jittering evasion.

### 3.4 Resistance to Chaff Perturbation Evasion

To determine if packet matching could resist intruders’ chaff perturbation evasion, we need to analyze if chaff perturbation can affect packet matching. That is to see if chaff perturbation can affect RTT which is used to match send and echo packets. Unlike time jittering, intruders can perform chaff perturbation on either send packets or echo packets, or on both. We divide our discussion to two cases: one is that intruders insert meaningless packets on only send packets; another one is that intruders perform chaff on both send and echo packets.

1. **Chaff on send packets only**

Before chaff perturbation we assume the send packet stream has $n$ packets $\{s_1, s_2, s_3, \ldots, s_n\}$ with time stamps $\{t_{s1}, t_{s2}, t_{s3}, \ldots, t_{sn}\}$, and echo packet stream has $m$ packets $\{e_1, e_2, e_3, \ldots, e_m\}$ with time stamps $\{t_{e1}, t_{e2}, t_{e3}, \ldots, t_{em}\}$. Think about the situation that one packet (we call it $k$th packet with time stamp $t_{sk}$) is inserted between $i$th and $(i+1)$th send packets. Because this $k$th packet will be removed before it goes to the end host, so this send packet has no echoed packet at all. When time stamp $t_{sk}$ is very close to $t_{si}$, this packet will merge to the $i$th packet, or if it is very close to $(i+1)$th packet, it will merge to $(i+1)$th packet, or it is not close to either $i$th packet or $(i+1)$th packet, this packet will exist independently and match nothing. It does not affect the round trip time computation theoretically. Obviously packet matching method can resist intruders’ evasion with only send packets are chaffed.

2. **Chaff on both send and echo packets**

We assume two send packets $s_i, s_{i+1}$ are matched by two echo packets $e_j, e_{j+1}$ respectively before chaff perturbation. After chaff perturbation, they become $s_i, s_p, s_{i+1}$ and $e_j, e_q, e_{j+1}$, where $s_p$ and $e_q$ are chaffs. The same as the above case 1, if $s_p$ is either close to $s_i$ or to $s_{i+1}$, it does not affect the round trip time computation. Here let us think about the situation that $s_p$ is neither close to $s_i$ nor to $s_{i+1}$. In this case $e_q$ may match $s_p$ to form a different RTT to interfere the packet matching or to make two unrelayed connection relayed. It occurs only under two strict conditions. One is there are many chaffs, another one is that accidentally the chaffs can form RTTs which are very close, and these RTTs can interfere with the real RTTs.

Just as what we analyzed before. It is impractical and impossible to insert two many meaningless packets into an interactive session. Even if the intruder can handle everything and do not care if the network communication is efficient or not, the second condition is very difficult to be satisfied because the intruder needs to control most of the packets inserted to be matched and to form a different set of RTTs. The more packets inserted, the more difficult for intruders to do that. The key here is that few packets inserted do not affect packet matching at all. If intruders just randomly insert some meaningless packets into an interactive session, it is highly impossible that the chaffs matched each other. Our conclusion is that randomly chaff cannot affect packet matching. That is to say packet matching method could resist intruders’ chaff evasion.
4. Conclusions and Future Work

In this paper, we have analyzed the resistance of packet matching algorithm to intruders’ time jittering and chaff perturbation evasion. Our conclusion is that the packet matching algorithm can not only resist intruders’ time jittering evasion, but also resist intruders’ chaff perturbation evasion.

It is noted that an intruder could manipulate a session at more hosts concurrently. In such case, we need to analyze the resistance to intruders’ evasion under the situation that a session is manipulated at more than one hosts concurrently, further investigation along this line is under way and the results will be reported in the forth coming paper.

References


