International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 03, March-2015

Unearthing the Stepping Stone Intrusion by TCP/IP Packet Matching Algorithm

S. Kranthi
Information Technology
VR Siddhartha Engineering College
Vijayawada, India

K. Pranathi
Information Technology
VR Siddhartha Engineering College
Vijayawada, India

***Abstract*- In this paper we present the matching algorithms to match TCP/IP packets in real time. Here we estimates the length of downstream TCP/IP packet to find stepping stone intrusion. The main intention of finding length of connection chain is to match the TCP/IP send and echo packets. Here we are going to come across SDC and SWAM algorithms. Where the disadvantages of SDC are overcome by SWAM algorithm .This new algorithm SWAM uses slide window format. The proposed approach algorithm can detect stepping-stone intrusion and resist intruders' time-jittering and chaff-perturbation manipulation to some extent.**

KEY WORDS - Stepping stone intrusion, TCP/IP matching packets, network security and intrusion detection.

1. INTRODUCTION:

Technology has become a major part of our lives. We have entered into an era where almost everyone who uses a computer should have basic knowledge about protecting their system from treats. Most of the intruders attack a system in a indirect way rather than direct way. Firstly they compromise some computers called steeping-stones, and then attack host computers.one reason about using stepping-stone intrusion is it makes intruders safe from being detected.

We formally give definitions for the following terms:

Connection: When a user from a host logs into another host, we call this a connection session between the two hosts.

Chain: Given n hosts H1,….,Hn , a sequence of connection is defined as a chain C =<C1,C2,C3……, Cn-1> where Ci is a connection from host Hi to Host Hi+1 for I=1,…..,n-1.

Downstream and upstream: If a direction is along a user's login direction it is called downstream. Otherwise, it is called upstream.

Send: A packet is defined as SEND if it propagates downstream and has flags both 'Push (P)' and 'Acknowledgement (A)' or only 'P'.

Echo: A packet is defined as ECHO if it propagates upstream and has flags both 'Push (P)' and 'Acknowledgement (A)'or only 'P'.

ACK: A packet is defined as ACK if it propagates either downstream or upstream and only has flag 'A'.

Matched packet: If a given echo is directly triggered by a send, then the Echo is defined as a matched packet of the Send. The method to find matched packets is called a Packet-matching algorithm.

Today's computer network security has been developed to a very advanced level and many algorithms have been developed. Such as Deviation-based approach, Round-Trip Time Approach and Packet Number Difference-Based Approach.

Staniford-Chen and Herberlin introduced a method which identifies intruders by comparing different sessions for suggestive similarities of connection chains. The main drawback of this is it cannot be applied to encrypted sessions. Then Zhang and Paxson proposed time based approach to detect stepping-stone intrusion. But this method has three major problems. First, the TCP/IP session can be easily manipulated. Second, these packets must have the precise and synchronized timestamps to correlate them properly .Yoda and Etoh proposed Deviation-Based Approach, this is a network-based correlation scheme. This method defines the deviation as the minimum average delay gap between the packet streams of two TCP connections. This time based approach has drawbacks like 1) computation deviation is not efficient .2) It depends on the size of packet.3) It can only correlate only when TCP connections have one-to-one correspondence in their TCP sequence numbers.4) correlation measurements are applicable only to post-attack traces because the correlation metrics are defined over entire duration of connection. The Round-Trip Time approach proposed by Yung this estimates the length by using gap between a request and its corresponding acknowledgement and as well as gap between a request and its corresponding response.

Then a Packet Number Difference-Based Approach (PND-based) proposed by Blum detects stepping-stones by

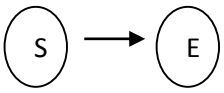
checking the difference between the send packet numbers of incoming connection and of an outgoing connection. If the two connections are relayed, the difference should always be bounded, otherwise, it should not. This method can resist time-jittering and chaff-pertubation.using wavelet and multi-scale methods the stepping-stone detection is still possible to monitor even the session is jittered by time and chaff perturbation. The major problem with the PND-based approach is that the upper bound of the number of packets required to be monitored is large and while the lower bound of amount of chaff needed to evade this detection is small. And also too many false positive errors are introduced. The longer the connection chain is, the higher the probability that the session is a stepping stone intrusion.

The key to compute the length of a downstream connection chain is to match TCP/IP send and echo packet.so many methods have proposed that match TCP/IP packets to detect stepping-stone intrusion. However, they all suffer from either inefficiency or in inaccuracy. In this paper, we will make use of some special features of computer network traffic to simply and improve the state of the art packet matching algorithm.

2. TRIALS TO SUIT TCP/IP PACKETS

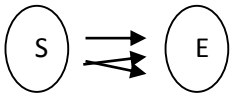
The packet matching problem is to find corresponding echoes for each send in TCP/IP stream. The packets transmitted on internet are complex, but they can be decomposed into four cases.

First case: Each send is followed by one Echo



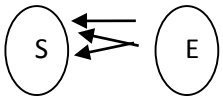
Echo is right one to match the send

Second case: Several sends are followed by one echo



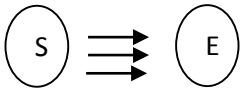
Echo gets matched with first send

Third case: One send is followed by several echoes



First echo is supposed to match the send

Fourth case: Several sends are followed by several echoes.



It is not so clear how to match them, first echo is supposed to match the first send.

There are many issues to affect matching TCP/IP packets. They are namely five main reasons

(1) Lost packet re-transmission
(2) Packet cumulative acknowledgement and echo
(3) Session transmit window
(4) Packets communication between adjacent hosts
(5) Multiple echoes from a server side

The lost packet gets re-transmitted when it receives an acknowledgement signal that is haven't received from sending client or when gets a request from receiving server. Re-transmission of same continues until it receives an acknowledgement is received or when connection timeout expires. So we are faced with one echo that matches with two or more sends.

Every TCP packet is not acknowledged individually rather cumulative acknowledgement may take place. The most vital advantage of this mechanism is that it reduces the number of acknowledgement messages. This mechanism makes one to one packet matching impossible. The same problem occurs for echo packets too.

To control data flow, TCP maintains a transmit window. The size of the window resolves how many unack signals of data that a transmitter is allowed to send before it must get ceased. In this way, if this size is set to one, which implicate each packet is sent if and only if the previous send has been acknowledged or echoed. If size is not set to one, then several packets get overlapped.

The problems of packet matching are inherited form the fact that send and echo packets may be in a many to many relationship, not one to one. It is difficult to match them in real-time. It is noteworthy that if we made any mistake in packet matching at one point of a packet stream, then that mistake would affect the entire packet matching after that point. In order to reduce the mistake, we divide a packet stream into some sub streams, one of which is the scope where we match the packets.

3. EXISTING PACKET MATCHING AND ITS PROBLEM:

The basic idea of matching TCP/IP packets is to exploit a fact that the Round Trip Time of matched packets has the smallest standard deviation comparing with the other non-matched packets. The send and echo packets are collected from the same connection chain at the same time interval and recorded in two sequences: S= $\{s_1, s2,\ldots,s_n\}$ and E=$\{e_1,e_2,\ldots\ldots e_m\}$. S_i can match any packet, such as e_j in E.A set S_i can be formed coming with that each element is the difference between S_i and a packet in E.

S1={s1e1, s1e2,…, s1em},

S2={s2e1, s2e2,…, s2em},

…

Si= {sie1, sie2, …,siem},

…

Sn={sne1, sne2,…,snem},

Where, $S_ie_j = t_{ej} - t_{si,}$ and here $t_{si}$and t_{ej} stand for the timestamps of a send packet Si and a echo packet e_j. Take one element from each set and form a cluster X. if all the cases are considered, there would be m^n clusters with each cluster n elements.

Algorithm SDC(S, E)

Begin:

(1) Generate data sets Ej (1≤j≤m): Ej={t(i,j) | t(i,j) = ej-si,

i=1, n &t(i,j)>0}

(2) Combine the elements in data sets Ej (1≤j≤m) to form

Clusters Xu (1≤u≤nm): Xu={t(ij,j)∈Ej | ∀1≤j≤m &

i1<i2…<im}

(3) For each cluster X: (a) if x(i, j), x(i, k) ∈ X, j<k, then

delete x(i, k), and (b) if x(i, j), x(k, j) ∈ X, i<k, then

delete x(i, j)

(4) Out R = {r1, r2,…rs} (1≤s≤n) which is the cluster with

smallest standard deviation among all clusters Xu

(1≤u≤nm).

End

the main problem of this algorithm is that to identify m^n clusters would be a big problem if m and n are of big numbers.

Efficient packet matching:

Because of complexity of algorithm it cant be applicable to online stepping-stone intrusion detection even application to offline detection is doubtfull.so we need to improve the algorithm that is to reduce the elements in $S_{i.}$

TCP/IP protocol tells us that they should be in chronological order. Because of chronological order, one send packet can't be matched twice.

The second point is that the packet in e_j in E can echo and only echo the packets in S which are before packet $e_j$chronically.so that the gap $s_ie_j$cannot be negative or zero.

The third point is that if a packet ein E could match a packet s_i in S, the gap between them cannot be too large. This is possible by a predefined parameter in TCP protocol. That is if a packet is set from a sender to a receiver, the packet will be either acknowledge and echoed or both with in a time threshold.

The fourth point is that the intruders may think and pause for a couple of seconds to wait for the response results or hoe to do the next step. This is usually as short time for humans but a large time for systems. This leads to the large time gaps which makes the packet matching algorithm more efficient.

The fifth point is that some consecutive packets have very close timestamps, those packets can be combined to one packet with average of their timestamps. This finally makes the packet matching algorithm efficient.

From the above motivations new algorithmic concept has been introduced called 'Sliding Window'. We put the packets of S and E together in a chronical order and set up a new sequence Q.the size of the sliding window is W and the 1st packet is send packet and the remaining W1 packets are the echoed ones. We slide the window from the first element of Q to the last one. These sets $S_1,S_2,\ldots,$Sn are computed with each one has W elements other than m.

4. SLIDING WINDOW PACKET MATCHING ALGORITHM (SWAM):

We can use an example to explain SWAM algorithm suppose there are 10 send, 14 echo packets collected and stored in two sequences.

S = {s1, s2, s3, s4, s5, s6, s7, s8, s9, s10}

E = {e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e13, e14}

By using SDC algorithm we can have 14^{10}=289254654976clusters .if SWAM is used these two sequences are combined in chronological order and put into sequence Q.

Q= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

if we assume the sliding window as 3 and by moving the window from beginning of Q we get the different packets as

Q1= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q2= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q3= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q4= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q5= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q6= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q7= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q8= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q9= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

Q10= {s1, s2, e1, s3, e2, s4, e3, e4, s5, e5, s6, e6, e7, s7, e8, e9, s8, e10, s9,

e11, e12, s10, e13, e14}

From the above sliding results, the following data sets are obtained.

S1={s1e1, s1e2}; S2={s2e1, s2e2}; S3={s3e2, s3e3}; S4={s4e3, s4e4};

S5={s5e5, s5e6}; S6={s6e6, s6e7}; S7={s7e8, s7e9}; S8={s8e10, s8e11};

S9={s9e11, s9e12}; S10={s10e13, s10e14}.

From this the numbers of possible clusters are 2^{10}=1024. It cannot be concluded by this because we don't know about the sliding window size so we need to take the different sliding window sizes and converged them to get a final stable result.

Generally, if there are n send and m echo packets collected, using SDC incurs m^n clusters. Assuming the sliding window size tried is from w1 to w2, SWAM incurs $\sum^{i=w_2, w_1} i^n$ clusters.

Later advancements where made on stepping stone intrusion detect where by using step function method which is used to estimate the length of a connection chain based on the changes in packet round trip times. The key point to compute the round trip time of a connection chain is to match a send and its corresponding echo.

First algorithm matches fewer packets and quality is high and the second algorithm matches up more packets with some uncertainty. This approach uses the changing RTT (Round Trip Time) between matched packets to estimate the length of a connection chain. The idea of using the changes of RTTs to signal the compromised hosts is demonstrated with experimental results from the interest.

They proposed two packet matching algorithm, a conservative one and greedy one to match TCP send and Echo packets. Conservative algorithm matches packet precisely but only for a small subset of the packets and greedy algorithm is used to know the correctness we are unsure of. If conservative algorithm fails to produce enough matched packet pairs, we can always use the greedy algorithm.

5. CONSERVATIVE ALGORITHM:

We cannot match all TCP packets, so if we can match a significant portion of the packets, it is sufficient for the purpose of estimating the length of a connection chain.

(1) Match only those that we are sure of their correctness.
(2) Include some that we are not completely sure about.

In first algorithm, we collect only the matches that we are truly confident in their correctness and we sacrifice on the matching rate. During an interactive terminal session, it is reasonable to divide a TCP/IP packet stream into some segments used on the third hypothesis we made each segment is started with one send. The gap between two continuous segments is supposed to be considerably larger than the RTT of a network. It is also safe to assume that no Echo packet will match a send packet across the segment gaps. If two consecutive send packets have time stamps difference that is more than TG, we will assume the existence of a gap. In our experiments, TG was set to one second, which worked well.

Algorithm 1 is used to match TCP packet, based on segment gap and two conditions stated above. In this algorithm, an empty send queue, which is used to store an unmatched send, is initialized once a packet is captured, we first need to resolve if it is a send on an echo.

Suppose we use E, S to stand for Echo, and send respectively each segment is going be expressed as either case1: {S_1E…..} or case 2: {S_1S_2…..S_nE……}

Initialize a SendQ queue;

correctMatch = true; // clear match flag

while(there are more packets){

capture the next packet p;

if p is a send packet{

reset the SendQ;

correctMatch = true;

}else {add p to SendQ;}

}else if p is an Ack packet{ //Ignore it

}else if p is an Echo packet {

Q = dequeue(SendQ);

If((Q.ack# = p.seq#) and (Q.seq# < p.ack#) and (correctMatch)){

Packets p and Q are matched;

Compute round trip time between p and Q;

} else {(// No match, set confusing match flag

correctMatch = false;

}

}

}

6. GREEDY ALGORITHM :

The main reason that algorithm 1 gives us low MR is that once we are confused about how to match the send in a send queue, we are going to discard al the sends of the queue once we get confused on which on in the send queue is supposed to match, we are going to match the very first send, and the following conditions must be satisfied.

Send SAN < Echo RSN

Send SSN < Echo RAN

The greedy algorithm tends to give us a higher RTT when it was confused in matching the packets in a send queue. Suppose there is a segment, {S_1 S_2 S_3……. S_N E_1 $S_{N+1}E_2$…..} , in which we already know that E_1 matches with S_1, and E_2 matches with S_3. The RTT between E_2 and S_2 obtained y the greedy algorithm is larger than it is supposed to be because S_2 is before S_3. The higher RTT doesnot hurt the purpose for stepping stone intrusion detection.

Initialize a SendQ queue;

While(there are more packets){

Capture the next packet p;

If p is a send packet {

Compute Time Gap TG;

If (TG > Threshold) {Reset the SendQ; }

Else {add p to SendQ;}

} else if p is an Echo packet {

Q = dequeue(SendQ);

If ((Q.ack# = p.seq #) and (Q.seq# < p.ack#)) {

Packets p and Q are matched;

Compute round trip time between p and Q;

} else if ((Q.ack# =< p.seq#) and (Q.seq# < p.ack#)) {

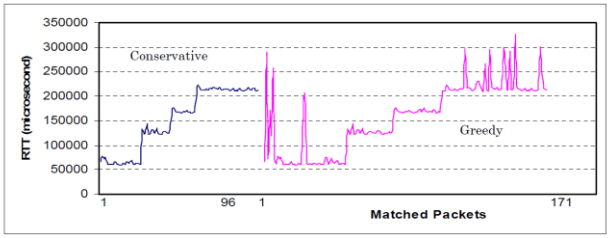
Packets p and Q are matched;

Compute RTT between p and Q;

} else {// No match;}

} else {Return; }

}



Fig(1) RTT Vs Matched packets graph for conservative and greedy algorithm

7. CONCLUSION:

These efficiency analysis shows that SWAM could reduce upto 99.99% and future development is to reduce the computation further and apply SWAM to online stepping stones for detection. We have also proposed the conservative and greedy algorithm to match TCP/IP packets online for computing the RTTs of a TCP interactive session. The conservative algorithm can give us correct matches, which have been proved, but with low MR. we evaluated the performance of the greedy algorithm and results showed that this algorithm is more useful and practical than the conservative one for the purpose of stepping stone intrusion detection.

REFERENCES

1. S. Staniford-Chen , L. T. Heberlein, Holding intruders accountable on the Internet, Proceedings of the 1995 IEEE Symposium on Security and Privacy, p.39, May 08-10, 1995
2. Kunikazu Yoda , Hiroaki Etoh, Finding a Connection Chain for Tracing Intruders, Proceedings of the 6th European Symposium on Research in Computer Security, p.191-205, October 04-06, 2000
3. Kwong H. Yung, Detecting long connection Chains of interactive terminal sessions, Proceedings of the 5th international conference on Recent advances in intrusion detection, October 16-18, 2002, Zurich, Switzerland
4. Jianhua Yang , Shou-Hsuan Stephen Huang, Matching TCP Packets and Its Application to the Detection of Long Connection Chains on the Internet, Proceedings of the 19th International Conference on Advanced Information Networking and Applications, p.1005-1010, March 25-30, 2005 [doi>10.1109/AINA.2005.240]
5. David L. Donoho , Ana Georgina Flesia , Umesh Shankar , Vern Paxson , Jason Coit , Stuart Staniford, Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay, Proceedings of the 5th international conference on Recent advances in intrusion detection, October 16-18, 2002, Zurich, Switzerland
6. Blum, A., Song, D., and Venkataraman, S. 2004. Detection of Interactive Stepping-Stones: Algorithms and Confidence Bounds. In Proceedings of International Symposium on Recent Advance in Intrusion Detection (Sophia Antipolis, France, 2004). Springer Press, New York, NY, 20--35.
7. Jianhua Yang , Shou-Hsuan Stephen Huang , Ming D. Wan, A Clustering-Partitioning Algorithm to Find TCP Packet Round-Trip Time for Intrusion Detection, Proceedings of the 20th International Conference on Advanced Information Networking and Applications, p.231-236, April 18-20, 2006 [doi>10.1109/AINA.2006.13]
8. Yang, J. and Huang, S.-H. 2007. Probabilistic Analysis of an Algorithm to Compute TCP Packet Round-Trip Time for Intrusion Detection. Journal of Computers and Security, Elsevier Ltd. Vol. 26(2) (Sep. 2007), 137--144.
9. Jianhua Yang , Shou-Hsuan Stephen Huang, A real-time algorithm to detect long connection chains of interactive terminal sessions, Proceedings of the 3rd international conference on Information security, November 14-16, 2004, Shanghai, China [doi>10.1145/1046290.1046331]
10. Yang, J. and Huang, S.-H. 2007. Mining TCP/IP Packets to Detect Stepping-Stone Intrusion. Journal of Computers and Security, Elsevier Ltd. Vol. 26(7--8) (Dec. 2007), 479--484.
11. Jianhua Yang , Byong Lee , Yongzhong Zhang, Finding TCP packet round-trip time for intrusion detection: algorithm and analysis, Proceedings of the 5th international conference on Cryptology and Network Security, December 08-10, 2006, Suzhou, China [doi>10.1007/11935070_21]