Anti-evasion Technique for Packet Based Pre-filtering for Network Intrusion Detection Systems (Poster)

Salvatore Pontarelli and Simone Teofili

Consorzio Nazionale InterUniversitario per le Telecomunicazioni (CNIT)
University of Rome "Tor Vergata", Via del Politecnico 1,
00133, Rome, Italy

Abstract. This work proposes a method to extend packet pre-filtering for Network Intrusion Detection Systems (NIDS). The aim of the method is to avoid the false negatives occurring when a malicious content has been sent splitted in several packets. In this paper we propose a method that is able to identify even the fragmented malicious content avoiding false negative limiting the false positive rate.[1]

1 Introduction

Network Intrusion Detection Systems (NIDS) like Snort [1] are used for monitoring the presence of different kind of attacks in a network. The packets or flows carrying these attacks are detected looking for in the header and payload "malicious content" defined by specific IDS rules. IDS software need high computational resources and processing time to examine all the packets of a network. These problems strongly limit the usage of IDS in the backbone of Internet Service Provider network. The porting of all the components of a software based IDS to an hardware platform is not feasible. Instead it is possible to identify some tasks that form one side allow to offload the software IDS and from the other can be easily implemented in hardware. Starting from this consideration hardware based packet pre-filtering that allow inspection at wire speed has been proposed in [2]. The main limitation of this approach is that the it does not perform any reordering of the packets composing the data stream. The absence of reordering causes a false negative (i.e. the flow contained an attack is not identified) when the malicious contents are transmitted in different packets. The obvious solution to avoid these false negatives is to implement in hardware the TCP reassembly and IP defragmentation tasks [3]. In Snort [1] this tasks are accomplished by STREAM5 and Frag3 pre-processors.

The implementation of this task is,however, very complex, and the papers that describe hardware implementation of TCP/IP reassembly usually can cope with a very limited number of flows. The solution we propose in this paper

[1] This work has been partially supported by the European Commission in the frame of the DEMONS project `http://fp7-demons.eu/`

J. Domingo-Pascual, Y. Shavitt, and S. Uhlig (Eds.): TMA 2011, LNCS 6613, pp. 185–188, 2011.

overcomes the limitation described above preserving a per-packet anomalous content analysis that allows us to manage the thousand of flow travelling into a network. Our solution is based on searching also for half-content and limits the false positive rate by selecting a best content representing each rule.

2 Proposed Strategy

In this section we show how to modify the IDS rules to avoids false negative due to fragmentation. First of all we apply the method presented in [4] to simplify the inspected rule taking the most representative content of each rule. This choice allows to limit the false positive rate due to the use of a relaxed version of the IDS rules. In Fig. 1 is shown how a content can be split between two packets. We suppose that the content is at least long L bytes, while the packets whose payload is longer than L. Three cases can occur:

1. the content is contained in a single packet (Fig. 1 a)
2. the content is equally divided between two packets (Fig. 1 b)
3. one packet contains more than $L/2$ bytes, the other less than $L/2$ (Fig. 1 c).

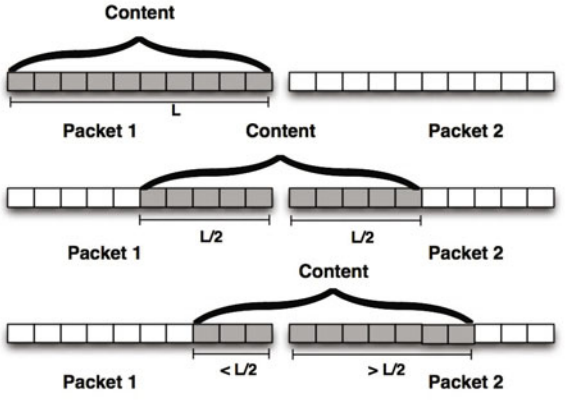

Fig. 1. Different split of a content in consecutive packets

The figure shows that, supposing all the packets longer than L, one of the packet containing a splitted content has to contain at least $L/2$ bytes. Moreover we note that half of the malicious content is contained always in the last $L-1$ bytes of a packet, or in the first $L-1$.

To check if a content is present in a flow it is sufficient to check the presence of the first half part of the content in the last $L-1$ bytes of the packet or the presence of the second half part of the content in the first $L-1$ bytes of a packet. If one of these checks is successful it is possible that a suspicious content has been sent into the network by using consecutive packets. Forwarding the flow that contains this packet to the software IDS we are therefore able to avoid false negative due to fragmented packets. We notice that this method works

only under the assumption that all the packets had at least a length of L bytes. In fact, suppose to have a packet with length of $L-2$ bytes, an attacker can split a content L into three packets. This fragmentation can evade the presented technique. To avoid the problem, we propose to forward to the software IDS the flows that had a packet with length less than L for further inspection.

3 Experimental Result

This section present the results obtained by applying the method described above to a realistic network scenario. In this experiment, we fix the parameter L to twenty and therefore $L/2$ is set to 10 bytes. In order to validate our results we select a large subset of the available snort IDS rules [1] to evaluate the false positive rate of our methodology. The rules has been used to analyze about 700 MBytes of traffic collected on the local area network of the Tor Vergata University network group. The collected trace is composed by a million of packet belonging to about 13000 different flows. The rules have been modified in order to match also the half-content and the begin or at the end of a packet, as described in the previous section. The packet distribution length is presented in Table 1.

Table 1. Distribution of packet length and number of flow with small packets

packet length (L)	number of packets	(%)	number of flows	(%)
< 100	10462	1%	1862	15 %
< 50	7606	0.7 %	1291	10 %
< 20	2786	0.2 %	842	7 %

From the above table we estimate that the 7% of the traffic had a packet with a length that can not be properly managed by our half-content method. Finally, we further reduce this amount of flow considering that many of these flows had as small packet the last packet of the stream, (if the small packet is the last one the content can not be split in three packets). Fig. 2 depict how a content can be split between these two packets.

It can be seen that in the second to last packet at least half-content is stored. Considering this refinement the number of flow that should be forwarded to the software decrease to 286 (the 2%), because 556 flows had the last packet with length less than L bytes. In Table 2 the result of our experiment are presented. The reassembly column use the original set of rules, while the half-content column use our method.

All the flow detected as suspected (the 507 flows of Table 2) and the ones composed with small packets (286 flows) are sent to an IDS with reassembly for further analysis. We obtain a total of 793 flows that are detected as suspected by our method again the Snort IDS that identify only 287 flows. Our method therefore identifies as flows needing further analysis about 6% (793 flows over a total of 13000 flows) of the incoming flows, offloading the software from the analysis of the overall flow travelling into the network.

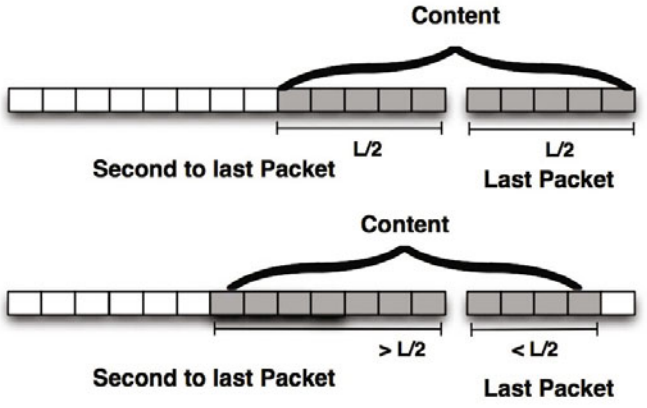

Fig. 2. Different split of a content in second to last and last packets

Table 2. Alert generated

	Snort with reassembly	proposed pre-filtering
inspected flows	13000	13000
flow sent to Snort	-	793
flow detected by Snort	287	287
Lost attacks	-	0
traffic reduction	-	7%

4 Conclusions

In this paper has been presented a methodology that avoids false negative due to packet fragmentation in packet pre-filtering for NIDS. The experiments performed on real traffic case are presented in order to prove the effectiveness of our proposed solution. Our measurement shows that our method forwards only the 6% of the incoming traffic to the second stage of the NIDS.

References

1. Sourcefire, Snort: The Open Source Network Intrusion Detection System (2003), `http://www.snort.org`
2. Song, H., Sproull, T., Attig, M., Lockwood, J.: Snort offloader: a reconfigurable hardware NIDS filter. In: International Conference on Field Programmable Logic and Applications, August 24-26 (2005)
3. Necker, M., Contis, D., Schimmel, D.: TCP-Stream Reassembly and State Tracking in Hardware. In: 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2002 (2002)
4. Teofili, S., Nobile, E., Pontarelli, S., Bianchi, G.: Snort pre-filter for data-reduced intrusion detection: hardware design issues and trade-offs. In: International Tyrrhenian Workshop on Digital Communications (ITWDC 2010), Ponza, Italy, September 6-8 (2010)