ABSTRACT

The authors propose in this paper an on-line algorithm based on Bloom filters to detect port scan attacks in IP traffic. Only relevant information about destination IP addresses and destination ports are stored in two steps in a two-dimensional Bloom filter. This algorithm can be indefinitely performed on a real traffic stream thanks to a new adaptive refreshing scheme that closely follows traffic variations. It is a scalable algorithm able to deal with IP traffic at a very high bit rate thanks to the use of hashing functions over a sliding window. Moreover it does not need any a priori knowledge about traffic characteristics. When tested against real IP traffic, the proposed on-line algorithm performs well in the sense that it detects all the port scan attacks within a very short response time of only 10 seconds without any false positive.

Keywords: Attack Detection, Bloom Filter, IP Traffic, Internet Measurements, On-Line Algorithms

INTRODUCTION

Problem Statement

We address in this paper the problem of designing an on-line algorithm for identifying port scan attacks in IP traffic. A port scan is a method of determining whether particular services are available on a host or a network by observing responses to connection attempts (Devivo, 1999). The received information is exploited to identify weaknesses and vulnerabilities of the host and to launch therefore more serious attacks. Several attack tools are now available and can easily be used (see (Nmap; Foundstone) and (Nessus)). Port scan can be launched from one or several sources. In this latter case, we are dealing with distributed attacks, which are more difficult to detect as the contribution of each source can be considered as legitimate traffic. According to Staniford (2002) port scan attacks can be classified into two categories:

1. Vertical scan consisting of scanning a big number of destination ports of a single destination address.
2. Horizontal scan when many IP addresses are scanned (generally within the same subnet), on one or several ports.

**Related Works**

Many port scan detecting methods had been developed in the literature. In Monowar (2010) Monowar et al. provide a survey on a large number of detection approaches. They classified them into many types (algorithmic, threshold-based, soft computing, rule-based, and visual approaches). They also established a comparison based on type, mode of detection and accuracy of the algorithms. Their main conclusion is that methods combining data mining and threshold-based analysis are the most efficient in terms of false positive rates, scalability and robustness. Most of detection approaches are single source and can not detect distributed attacks (e.g. (Robertson, 2003; Roesch, 1999)). Very few methods can be applied on-line and give real time response. Moreover a common weakness of port scan detecting methods, particularly the threshold-based methods, is that their accuracy is closely dependent on traffic characteristics. Quite often, algorithms depend on constants directly related to the traffic intensity. For a limited set of traces, they can be tuned by hand to get reasonable performances. This procedure is, however, not acceptable in the context of an operational network. As a general requirement, it is highly desirable that the constants used by algorithms automatically adapt, as simply as possible, to varying traffic conditions. As an example, in Heberlein (1990) Heberlein et al. propose an IDS (Intrusion Detection System) where a source is considered as malicious if it contacts more than 15 other IP addresses in a given time window.

In this paper we focus on vertical port scan where many ports are scanned for a given destination address. Our objective is to design an adaptive algorithm that detects on-line this kind of attacks. The algorithm will be deployed on operators IP backbone network carrying traffic at a very high bit rate. At this level one can have an aggregated view of the global traffic (generated by many hosts), which is very useful to detect attacks. Moreover at this level, network operator can stop the attack to avoid its propagation until the destination and therefore save network resources. An on-line analysis of IP traffic in the core network is a challenging issue. The algorithm has to perform a very quick data processing and to store only relevant information. Data analysis should be faster than the arrival rate of data stream of a real-time execution. So the processing time of a packet has to be lower than the inter-arrival packet time, which is of the order of few nanoseconds. This is typically a context of data mining.

A natural solution to cope with the huge amount of data in IP traffic is to use hash tables. A data structure using hash tables, a Bloom filter, proposed by B. Bloom (1970) in 1970, has been used to test whether an element is a member of a given set. Bloom filters have been used in various fields: In McIlroy (1982) and Mullin and Margoliash (1990) they are used to represent words of a dictionary with a small memory. They make the search in the dictionary much faster. Bloom filters are also very useful for some distributed database applications. With a simple representation of a table content in a Bloom filter, one can speed up significantly semi-join operations and reduce the overhead communication between machines as they only exchange Bloom filters (see Bratbergsengen (1984) and Mullin (1990)). Many distributed network applications rely on Bloom filters to improve their performance. We can for example mention the distributed web cache sharing proposed by Fan et al. in (Fan, 2000) or the detecting loops algorithm introduced by Whitaker and Wetherall (Whitaker, 2002), in the context of packet switching. Bloom filters are well adapted to many other applications such as multicast (Gronvall, 2002), resource routing in peer-to-peer networks (Druschel, 2001; Stoica, 2001) and queue management (Dilip, 2001).

Bloom filters have been used by Estan and Varghese (Estan, 2002) to detect large flows. A Bloom filter consists of \( k \) tables of counters indexed by \( k \) hash functions. The general principle is the following: for each table, the flow ID of a given packet (that is the addresses and port numbers of the source and the destination) is
Adaptive Web Service Composition: An Aspect-Oriented Approach
www.igi-global.com/chapter/adaptive-web-service-composition/115432?camid=4v1a

A Tool Support for Secure Software Integration
www.igi-global.com/chapter/tool-support-secure-software-integration/65852?camid=4v1a