Journal of Network and Computer Applications 39 (2014) 83–92

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection ☆

Yuxin Meng *, Lam-For Kwok

Department of Computer Science, City University of Hong Kong, Hong Kong, China

ARTICLE INFO

Article history:
Received 15 November 2012
Received in revised form
8 April 2013
Accepted 16 May 2013
Available online 24 May 2013

Keywords:
Network intrusion detection
Packet filter
Blacklist generation
Adaptive system
Signature matching

ABSTRACT

Network intrusion detection systems (NIDS) are widely deployed in various network environments. Compared to an anomaly based NIDS, a signature-based NIDS is more popular in real-world applications, because of its relatively lower false alarm rate. However, the process of signature matching is a key limiting factor to impede the performance of a signature-based NIDS, in which the cost is at least linear to the size of an input string and the CPU occupancy rate can reach more than 80% in the worst case. In this paper, we develop an adaptive blacklist-based packet filter using a statistic-based approach aiming to improve the performance of a signature-based NIDS. The filter employs a blacklist technique to help filter out network packets based on IP confidence and the statistic-based approach allows the blacklist generation in an adaptive way, that is, the blacklist can be updated periodically. In the evaluation, we give a detailed analysis of how to select weight values in the statistic-based approach, and investigate the performance of the packet filter with a DARPA dataset, a real dataset and in a real network environment. Our evaluation results under various scenarios show that our proposed packet filter is encouraging and effective to reduce the burden of a signature-based NIDS without affecting network security.

1. Introduction

Malware software such as malicious code, AutoRun worm, password-stealing Trojan, downloader and web-based exploit have become more and more prevalent than ever (i.e., about 20% increase of malware threats in Mcafee Threat Report: Second Quarter, 2012). To better protect network security, network intrusion detection systems (NIDSs) have become an essential component to protect different networks (e.g., an organizational network) against various network attacks. The wide use of NIDSs is a powerful complementary solution to current firewall technology to defend against attacks and suspicious network traffic that are missed by a firewall.

Traditionally, network intrusion detection systems can be mainly categorized into two folders: signature-based NIDS and anomaly based NIDS. A signature-based (also called *rule-based*) NIDS (Roesch, 1999) detects possible network attacks by comparing its signatures against observed events. A signature (or called *rule*) is a pattern that describes a known threat. An anomaly based NIDS (Paxson, 1999) identifies significant deviations by comparing pre-defined normal events (or *normal profile*[1]) against current observed events.

In real-world applications, the signature-based detection is more prevalent than the other detection approaches because of its relatively lower false alarm rate (Sommer and Paxson, 2010). In particular, these detection systems usually maintain a signature database for conducting the process of signature matching and update the database periodically. Take *Snort* (Snort) as an example, this open source light-weight signature-based NIDS contains four major components (including a packet capture engine, preprocessor plug-ins, a detection engine, and output plug-ins) and the core component is the detection engine that compares incoming packets with the stored signatures in the database. The number and complexity of the signatures are the main factors in affecting the performance of signature matching (or called *string matching*). Generally, a bigger number of signatures can cover a wider range of network attacks than a smaller group of signatures; and a tighter signature, by describing an attack with more features, can achieve a lower false alarm rate than a loose signature.

Problem. Although the computing power of computers becomes stronger and stronger nowadays, the limited computing capability is still an issue for a signature-based NIDS, which is caused by

☆A preliminary version of this paper appears in Proceedings of the International Conference on Information Assurance and Security (IAS 2011), pp. 74–79, 2011 (Meng and Kwok, 2011).

* Corresponding author. Tel.: +852 34423729.
E-mail addresses: yuxin.meng@my.cityu.edu.hk (Y. Meng), cslfkwok@cityu.edu.hk (L.-F. Kwok).

[1] The profiles can be constructed through monitoring the characteristics of typical activity over a period of time (Scarfone and Mell, 2007).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.05.009

high-volume traffic and the increasing number and complexity of NIDS signatures. For instance, Snort will quickly exhaust a computer memory when deployed in a high-volume traffic network, and this situation will cause Snort to drop a large number of incoming packets (Dreger et al., 2004). In addition, the process of string matching is so expensive in searching for an attack pattern in an incoming packet that the processing consumption is at least linear to the size of an input string (Rivest, 1977). Specifically, Snort usually spends about 30% of its total processing time in comparing input strings with its stored signatures, but the consuming time can increase up to 80% when it deals with intensive web traffic (Fisk and Varghese, 2002). With the increasing trend of network threats, it seems that the workload of a NIDS will become heavier with the growth of the number and complexity regarding the NIDS signatures.

To mitigate the above problem, several approaches have been proposed such as the improvement of signature matching algorithms and the pre-filtration of network packets (see details in Section 2.2). In real deployment, we notice that the blacklist technique has been widely used in spam detection (West and Lee, 2011), but it has not been studied much in constructing a packet for an IDS. Our motivation is therefore to explore the use of a blacklist in constructing a packet filter in helping filter out network packets for signature-based NIDSs.

Contributions. In this paper, we advocate the method of enhancing the performance of a signature-based NIDS by constructing a packet filter and filtering out packets, since a packet filter is easy for implementation and is flexible for utilization (i.e., as a plug-in component without modifying the NIDS). In particular, we propose a promising method of using the *blacklist* technique in constructing a packet filter to alleviate the burden of a signature-based NIDS by filtering out a number of network packets and reducing the target packets. Our contributions can be summarized as below.

- We develop an adaptive blacklist-based packet filter using a statistic-based approach to improve the performance of a signature-based NIDS. The filter consists of two major components: a *blacklist packet filter* and a *monitor engine*. The monitor engine updates the blacklist periodically so that the blacklist packet filter can filter out network packets in an adaptive way (i.e., the blacklist generation can be adaptive to traffic changes).
- We propose and implement a weighted ratio-based method (*statistic-based method*) in the component of monitor engine to calculate IP confidence and generate the blacklist in an adaptive way. Moreover, we explore the impact of *weight values* and identify a range of appropriate values to achieve a lower false positive and false negative by conducting simulations on a real dataset.
- To study the performance of our proposed packet filter, we evaluate it with the DARPA dataset, a real dataset and in a real network environment. The experimental results under various scenarios show that our proposed packet filter is effective to reduce the burden of a signature-based NIDS by filtering out a number of network packets without affecting network security.

The remaining parts of this paper are organized as follows. Section 2 briefly introduces the process of signature matching in Snort and review some related work in improving and mitigating this expensive process. In Section 3, we present the architecture of our adaptive blacklist-based packet filter and describe each component in detail; we illustrate how to select an appropriate weight value in Section 4, and present our experimental methodology and experimental results in Section 5; at last, Section 6 concludes our work.

2. Background and related work

In this section, we first introduces the background of Snort rules and its process of signature matching, we then review some related work in mitigating the expensive process of signature matching including designing novel matching algorithms, and classifying and filtering out network packets.

2.1. Background

Snort is an open-source signature-based NIDS which is widely used in the IDS research. We also utilize this tool in this work to explore the performance of our proposed method. Therefore, in this section, we briefly introduce the process of Snort signature matching.

To better illustrate the process of signature matching, we present a specific Snort rule, which is extracted from the Snort rule database (a folder of *scan.rules*) related to *scanning* as below.

Snort rule alert udp $EXTERNAL_NET any -> $HOME_NET 7 (msg: “SCAN cybercop udp bomb”; content: “cybercop”;)

The purpose of this Snort rule is to detect attempts of scanning using *CyberCop scanner*. Attackers can use this scanner to determine whether a computer system is vulnerable to a UDP bomb denial-of-service (DoS) attack or not. This attack targets on a UDP port by flooding it with ECHO and CHARGEN packets and connecting a host ECHO service to a local or remote CHARGEN service. The content of “*cybercop*” is the *target string* in this rule, and Snort makes an alert if it detects such a string in a UDP packet.

For the process of signature matching, Snort separates its rule into two parts: *rule header* and *rule option*. The *rule header* contains the protocol field, the source IP address and port, and the destination IP address and port. The *rule option* mainly contains strings (signatures) for the signature matching. After parsing the rules, Snort obtains three port groups: *destination port group*, *source port group* and *generic port group*. In particular, the *destination port group* contains rules with a unique destination port; the *source port group* has a unique source port for each rule; and the rules in the *generic port group* have no unique destination port and source port. For the above scanning attack rule, it has a clear destination port number 7, but the source port is generic. Thus, this rule is grouped into the *destination port group*. In the best scenario, a packet is only compared once for one of the three groups, whereas a packet is scanned up to three times for all the three groups in the worst case.

2.2. Related work

To resolve the problem of signature matching, several approaches have been proposed which can be roughly classified into two categories. One category is to enhance the process of string matching by directly designing new matching algorithms to speed up the process. For example, Boyer and Moore (1977) algorithm is the most widely used single pattern matching algorithm. It compares the target string with the input content beginning with the rightmost character of the string and uses two heuristics to reduce the number of searches in the matching process. Then, Horspool (1980) improved the Boyer–Moore algorithm by using only the bad-character heuristic with the purpose of achieving a more efficient implementation.

The multi-string matching, which searches a string by using only a single iteration, is also an effective method in improving the process of string matching. The well-known Aho–Corasick algorithm (1975) preprocesses the patterns to construct a deterministic finite automaton (DFA) aiming to search for all strings at the same time. The automaton compares the input string one character at a time and finds the matched patterns. Several