KNOWLEDGECUDDLE PUBLICATION

International Journal of Computer Engineering and Science, August- 2014

Efficient String Matching Algorithm for Intrusion Detection

Bhargavi Patel
Computer department, B.V.M Engineering College, India
bhargavi71291@yahoo.in

ABSTRACT: ***Intrusion Detection Systems (IDSs) have become widely recognized as powerful tools for identifying, deterring and deflecting malicious attacks over the network. Intrusion detection systems (IDSs) are designed and installed to aid in deterring or mitigating the damage that can be caused by hacking, or breaking into sensitive IT systems. . The attacks can come from outsider attackers on the Internet, authorized insiders who misuse the privileges that have been given them and unauthorized insiders who attempt to gain unauthorized privileges. IDSs cannot be used in isolation, but must be part of a larger framework of IT security measures. Essential to almost every intrusion detection system is the ability to search through packets and identify content that matches known attacks. Space and time efficient string matching algorithms are therefore important for identifying these packets at line rate. In this paper we examine string matching algorithm and their use for Intrusion Detection.***

Keywords: ***System Design, Network Algorithm***

I. INTRODUCTION

With each passing day there is more critical data accessible in some form over the network. Any publicly accessible system on the Internet today will be rapidly subjected to break-in attempts. These attacks can range from email viruses, to corporate espionage, to general destruction of data, to attacks that hijack servers from which to spread additional attacks. Even when a system cannot be directly broken into, denial of service attacks can be just as harmful to individuals, and can cause nearly equal damage to the reputations of companies that provide services over the Internet. Because of the increasing attacks held by the various users of the internet, there has been widespread interest in combating these attacks at every level, from end hosts and network taps to edge and core routers. Intrusion Detection Systems (or IDSs) are emerging as one of the most promising ways of providing protection to systems on the network.

The Basis for Acquiring Idss

At least three reasons justify the acquisition of IDS. The three are:

1. To provide the means for detecting attacks and other security violations that cannot be prevented.
2. To prevent attackers from probing a network.
3. to document the intrusion threat to an organization.

As with firewalls, intrusion detection systems are growing in popularity because they provide a site resilience to attacks without modifying end-node software. While firewalls only limit entry to a network based on packet headers, intrusion detection systems go beyond this by identifying possible attacks that use valid packet headers that pass through firewalls. Intrusion detection systems gain this capability by searching both packet headers and payloads to identify attack signatures.

To define suspicious activities, IDS makes use of a set of rules which are applied to matching packets. A rule consists at minimum of a type of packet to search, a string of content to match, a location where that string is to be searched for, and an associated action to take if all the conditions of the rule are met.

In addition, as IDSs move from end-hosts into edge and core routers, the needs placed on algorithms for intrusion detection will change. While common-case performance can be an acceptable metric for end-hosts that are based on commodity processors, in order to be successful inside the network infrastructure, algorithms must satisfy stringent worst-case performance bounds and tight constraints on memory. At the heart of almost every modern intrusion detection system is a string matching algorithm. String matching is crucial because it allows detection systems to base their actions on the content that is actually flowing to a machine. From this sea of packets, the string identifies those packets that contain data matching the fingerprint of a known attack. Essentially, the string matching algorithm compares the set of strings in the rule-set to the data seen in the packets that flow across the network.String matching is computationally intensive. Because string matching dominates the performance in this and many other IDS, in this we concentrate our efforts on building smaller and faster string matching algorithms. We present optimized techniques for matching large sets of strings in incoming packets in the context of network intrusion detection.
We characterize the properties of a real set of IDS string matching rules and examine both how the rules have changed over time, and the effect of those changes on the data structures used. An important contribution of this work is the development of an algorithm that performs well and has useful bounds on worst case performance.

II. TYPES OF IDS

There are several types of IDS available. They are characterized by different monitoring and analysis approaches. Each type has distinct uses, advantages, and disadvantages. IDSs can monitor events at three different levels: network, host, and application. They can analyze these events using two techniques: signature detection and anomaly detection. Some IDSs have the ability to respond automatically to attacks that are detected.

1. Ids Monitoring Approaches

One way to define the types of IDSs is to look at what they monitor. Some IDSs listen on network backbones and analyze network packets to find attackers. Other IDSs reside on the hosts that they are defending and monitor the operating system for signs of intrusion. Still others monitor individual

applications.

1.1 Network base IDs

Network-based IDSs are the most common type of commercial product offering. These mechanisms detect attacks by capturing and analyzing network packets. Listening on a network backbone, a single network-based ID can monitor a large amount of information. Network-based IDSs usually consist of a set of single-purpose hosts that "Sniff" or capture network traffic in various parts of a network and report attacks to a single management console. Because no other applications run on the hosts that are used by network-based IDS, they can be secured against attack. Many of them have "stealth" modes, which make it extremely difficult for an attacker to detect their presence and to locate them.

Advantages: A few well-placed network-based IDSs can monitor a large network. The deployment of network based IDSs has little impact on the performance of an existing network. Network-based IDSs are typically passive devices that listen on a network wire without interfering with normal network operation. Thus, usually, it is easy to retrofit a network to include network-based IDSs with a minimal installation effort. Network-based IDSs can be made very secure against attack and can even be made invisible to many attackers.

Disadvantages: Network-based IDSs may have difficulty processing all packets in a large or busy network. Therefore, such mechanisms may fail to recognize an attack that is launched during periods of high traffic. IDSs that are completely implemented in hardware are much faster than those that have been totally realized in software. In addition, the need to analyze packets quickly forces vendors to try and detect attacks with as few computing resources as possible. This may reduce detection effectiveness.

Many of the advantages of network-based IDSs do not always apply to the more modern switch-based networks. Switches can subdivide networks into many small segments; this will usually be implemented with one fast Ethernet wire per host. Switches can provide dedicated links between hosts that are serviced by the same switch. Most switches do not provide universal monitoring ports. This reduces the monitoring range of a network-based IDS sensor to a single host. In switches that do provide such monitoring ports, the single port is frequently unable to mirror all the traffic that is moving through the switch.

Network-based IDs cannot analyze encrypted information. Increasingly, this limitation will become a problem as the use of encryption, both by organizations and by the attackers, increases. Most network-based IDSs do not report whether or not an attack was successful. These mechanisms only report that an attack was initiated. After an attack has been detected, administrators must manually investigate each host that has been attacked to determine which hosts were penetrated.

1.2 Host based Ids

Host-based IDSs analyze the activity on a particular computer. Thus, they must collect information

from the host they are monitoring. This allows IDS to analyze activities on the host at a very fine granularity and to determine exactly which processes and users are performing malicious activities on the operating system. Some host-based IDSs simplify the administration of a set of hosts by having the administration functions and attack reports centralized at a single IT security console. Others generate messages that are compatible with network administration systems.

Advantages Host-based IDSs can detect attacks that are not detectable by network-based IDS because this type as a view of events that are local to a host. Host-based IDSs can operate in a network that is using encryption when the encrypted information is decrypted on (or before) reaching the host that is being monitored. Host based IDSs can operate in switched networks.

Disadvantages: the collection mechanisms must usually be installed and maintained on every host that is to be monitored. Because portions of these systems reside on the host that is being attacked, host-based IDSs may be attacked and disabled by a clever attacker. Host-based IDSs are not well-suited for detecting network scans of all the hosts in a network because the IDS at each host sees only the network packets that the host receives. Host- based IDSs frequently have difficulty detecting and operating in the face of denial-of-service attacks. Host based IDSs use the computing resources of the hosts they are monitoring.

1.3 Application based Ids

Application-based IDSs monitor the events that are transpiring within an application. They often detect attacks by analyzing the application's log files. By interfacing with an application directly and having significant domain or application knowledge, application- based IDSs are more likely to have a more discerning or fine-grained view of suspicious activity in the application.

Advantages: Application-based IDSs can monitor activity at a very fine granularity, which allows them, often, to track unauthorized activity to individual users. Application-based IDSs can work in encrypted environments, because they interface with the application that may be performing encryption.

Disadvantages: Application-based IDSs may be more vulnerable than host-based IDSs to being attacked and disabled because they run as an application on the host that they are monitoring. The distinction between application-based IDS and host-based IDS is not always clear. Thus, for the remainder of this article, both types will be referred to as host- based IDSs.

2. IDS Event Approaches

There are two primary approaches to analyzing computer and networks events to detect attacks: signature detection and anomaly detection. Signature detection is the primary technique used by most commercial IDS

products. However, anomaly detection is the subject of much research and is used in limited form by a number of IDSs.

2.1 Signature based IDSs

Signature-based detection looks for activity that matches a predefined set of events that uniquely describe a known attack. Signature-based IDSs must be specifically programmed to detect each known attack. This technique is extremely effective and is the primary method used in commercial products for detecting attacks.

Advantages: Signature-based IDSs are very effective in detecting attacks without generating an overwhelming number of false alarms.

Disadvantages: Signature-based IDSs must be programmed to detect each attack and thus must be constantly updated with the signatures of new attacks. Many signatures based IDSs have narrowly defined signatures that prevent them from detecting variants of common attacks.

2.2 Anomaly based IDS

Anomaly-based IDSs find attacks by identifying unusual behaviour(i.e., anomalies) that occurs on a host or network. They function on the observation that some attackers behave differently than "normal" users and thus can be detected by systems that identify these differences. Anomaly-based IDSs establish a baseline of normal behaviour by profiling particular users or network connections and then statistically measure when the activity being monitored deviates from the norm. These IDSs frequently produce a large number of false alarms because normal user and network behaviours can vary widely. Despite this weakness, the researchers working on applying this technology assert that anomaly-based IDSs are able to detect never-before-seen attacks, unlike signature based IDSs that rely on an analysis of past attacks. Although some commercial IDSs include restricted forms of anomaly detection, few, if any, rely solely on this technology. However, research on anomaly detection IDS products continues.

Advantages: Anomaly-based IDSs detect unusual behaviour and thus have the ability to detect attacks without having to be specifically programmed to detect them.

Disadvantages: Anomaly detection approaches typically produce a large number of false alarms due to the unpredictable nature of computing and telecommunication users and networks. Anomaly detection approaches frequently require extensive "training sets" of system event records to characterize normal behaviour patterns.

III. EVASION TECHNIQUES

Fragmentation: by sending fragmented packets, the attacker will be under the radar and can easily by pass the detection system's ability to detect the attack signature.

Avoiding defaults: The TCP port utilised by a protocol does not always provide an indication to the protocol which is being transported. For example, IDS may expect to detect a Trojan on port 12345. If an attacker had reconfigured it to use a different port the IDS may not be able to detect the presence of the Trojan.

Coordinated, low-bandwidth attacks: coordinating a scan among numerous attackers (or agents) and allocating different ports or hosts to different attackers makes it difficult for the IDS to correlate the captured packets and deduce that a network scan is in progress.

Address spoofing/proxying: attackers can increase the difficulty of the ability of Security Administrators to determine the source of the attack by using poorly secured or incorrectly configured proxy servers to bounce an attack. If the source is spoofed and bounced by a server then it makes it very difficult for IDS to detect the origin of the attack.

Pattern change evasion: IDS generally rely on 'pattern matching' to detect an attack. By changing the data used in the attack slightly, it may be possible to evade detection. For example, an IMAP server may be vulnerable to a buffer overflow, and IDS is able to detect the attack signature of 10 common attack tools. By modifying the payload sent by the tool, so that it does not resemble the data that the IDS expect, it may be possible to evade detection.

IV. STRING MATCHING FOR INTRUSION DETECTION

In the Introduction we motivated the need for string matching in Intrusion Detection Systems. In this section we further demonstrate how string matching is used in an actual intrusion detection system. We also examine the state of the art in string matching as it relates to intrusion detection.

Aho-Corasick string-matching algorithm

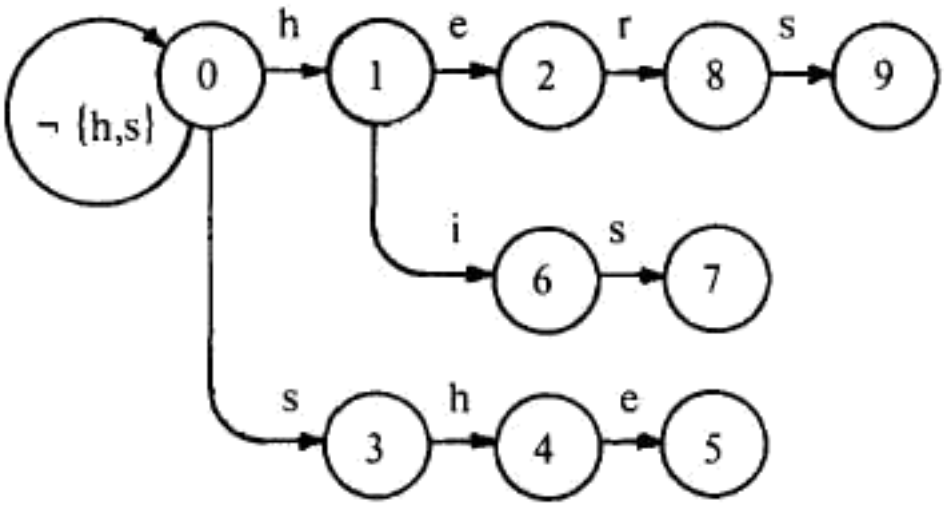
One of the earliest algorithms in precise multi-pattern string matching is due to Aho- Corasick , which is able to match strings in *worst case* time linear in the size of the input. Aho-Corasick works by constructing a state machine from the strings to be matched. The state machine starts with an empty root node which is the default non-matching state. Each pattern to be matched adds states to the machine, starting at the root and going to the end of the pattern. The state fig(2) machine is then traversed and failure pointers are added from each node to the longest prefix of that node which also leads to a valid node in the trie.(Fig 3). Beyond this basic notion, there are two choices for the algorithm. We can optimize the data structure further by using the failure pointers to precompute the next state for every character from every state in the machine (Fig 1), or we can leave these transitions undefined and traverse the failure pointers at run-time (Fig 2).If the data structure is optimized, then Aho- Corasick requires only a single memory reference (albeit a very wide memory reference)

per character in the input. If the data structure is left unoptimized, one can show via amortizedanalysis that only two (again wide) memory references per character of input string are required to traverse the data structure. We use the unoptimized data structure because the undefined pointers allow us significant opportunity for space optimizations.

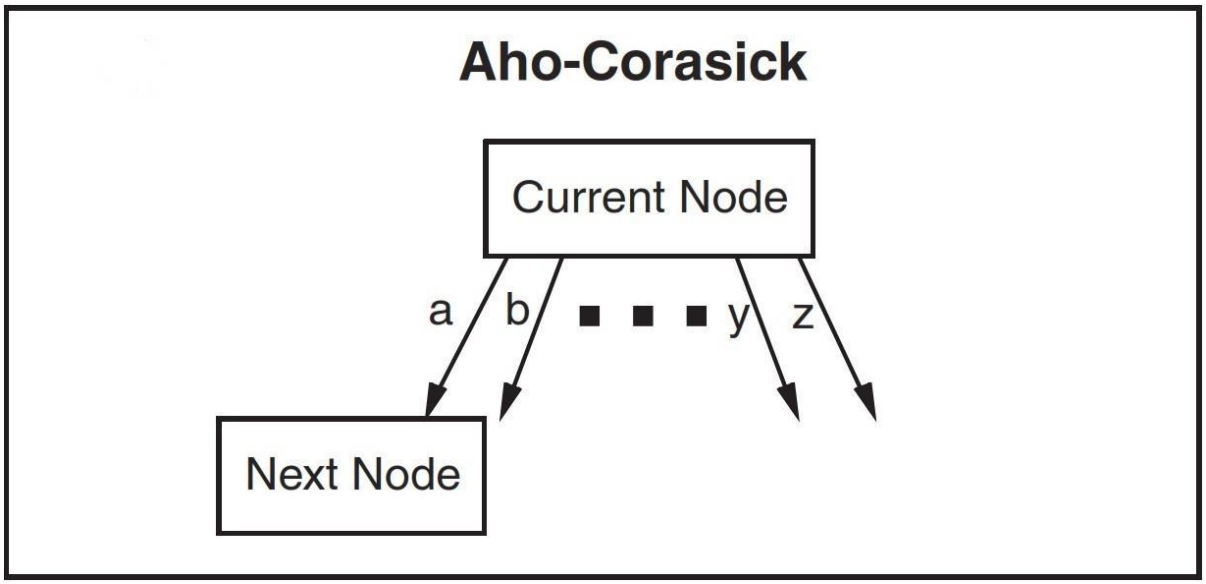
V. FIGURES

(1) Pattern matching machine.

Fig. 1. Pattern matching machine.



(2) Aho-Corasick



VI. CONCLUSION

Guided by the analogy between IP lookup and string matching, our paper builds on the worst-case guarantees of the classical Aho-Corasick string matching algorithm. As with multibit tries, Aho-Corasick is the only string matching algorithm we know of that has deterministic worst-case lookup times and a data structure friendly enough to use for wire speed hardware matching. Unfortunately, the classical Aho-Corasick data structure takes more storage than is likely to fit in on-chip SRAM or the cache of a commodity processor.

The principal contribution of our paper is to apply bitmap node compression and path compression to AhoCorasick to gain both compact storage and worst-case performance. In particular, we show that the use of such compression gains factors of almost 50 times in database size reductions on current rule sets. While the case is less clear for software implementations unless more predictable performance is desired; we believe that our compressed AhoCorasick algorithms are the best choice for hardware implementations
of string matching for IDS using an FPGA, ASIC or network processor designs of the future.

REFERENCES

[1] Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection- By Nathan Tuck, Timothy Sherwood, Brad Calder, George Varghese, Department of Computer Science and Engineering, University of California, San Diego, Department of Computer Science, University of California, Santa Barbara.

[2] EDPACS-THE EDP Audit, Control, And Security Newsletter, NOVEMBER 2001 Vol.XXIX, No.5

[3] M. Consens, G. Navarro, "N-Gram Similarity and Distance". SPIRE 2005, LNCS 3772, 2005, pp. 115-126

[4] Andreas Wespi, Marc Dacier, Herve Debar. "Intrusion Detection Using Variable-Length Audit Trail Patterns". Third International Workshop on the Recent Advances in Intrusion Detection(RAID 2000). Toulouse, France, 2000, 110-129

[5] Andreas Wespi, Marc Dacier, Herve Debar. "An Intrusion-Detection System Based on the Teiresias Pattern-Discovery Algorithm". EICAR Proceedings, 1999, pp. 1-15

[6] Duan You-xiang, Huang Min, Xu Jiuyun, "Initialization Method of Gene Libray Based on Teiresias Algorithm". Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC 09. 2009, pp, 294-297

[7] Jonassen Inge, "Efficient Discovery of Conserved Patterns using a Pattern Graph". Comput Appl Biosci, 1997