Network Intrusion detection on CPU GPU System

Miss. Ambekari Tabassum Dawood, Mrs A.N.Mulla

Annasaheb Dange College Of Enginnering, Ashta

Abstract- With the great increase in the usage of computer networks, the risk of having these network to be under attacks have been increased. Number of techniques have been created and designed to help in detecting and/or preventing such attacks. One common technique is the use of Intrusion Detection Systems (IDS). Today, number of open sources and commercial IDS are available to match enterprises requirements. However, the performance of these systems is still the main concern. This paper examines perceptions of intrusion detection architecture implementation, resulting from the use of graphics processor. It discusses recent research activities, developments and problems of operating systems security. Some exploratory evidence is presented that shows capabilities of using graphical processors and intrusion detection systems. The focus is on how knowledge experienced throughout the graphics processor inclusion has played out in the design of intrusion detection architecture.

Index Terms- Network analysis, Intrusion detection, GPU, CUDA, Parallel computing.

I. Introduction

A Network Intrusion detection System (NIDS) is an intrusion detection system that tries to detect malicious activity such as service attacks, port scans or even attempts to break into computers by monitoring network traffic. A network intrusion detection system reads all incoming packets and tries to find suspicious patterns known as signatures or rules. For instance, if it is observed that a particular TCP connection requests connection to a large number of ports, then it can be assumed that there is someone who is trying to conduct a port scan of all/most of the computers of the network. A network intrusion detection system is not limited to inspecting the incoming network traffic only. Patterns and outgoing intrusion can also be found from the outgoing or local traffic as well. Some attacks might also come from the inside of the monitored network, as in trusted host attack. At the heart of every modern network intrusion detection system there is a string matching algorithm. The network intrusion detection system uses the string matching algorithm to compare the payload of the network packet and/or flow against the pattern entries of the intrusion detection rules, which are a part of every network having a network intrusion detection system[1].

Intrusion detection is the process of monitoring the events occurring in system and analyzing them for signs of intrusions. These intrusions are the results of attackers accessing systems from the Internet, authorized users of the systems who attempt to gain additional unauthorized privileges, and authorized users who misuse the privileges given to them. Intrusion detection is a security technology that attempts to identify and isolate intrusions against computer systems. Intrusion detection is an important component of a security system, and it complements other security technologies.

II. NEED FOR INTRUSION DETECTION SYSTEM

The underlying reasons for using intrusion detection systems are relatively straightforward

-identifying attacks that firewall legitimately allow.

-Identifying attempts such as port scan or ping sweep.

-Notice insider hacking.

-Provide additional checks for holes/ports opened through firewalls, intentionally or unintentionally.

A. Uses of intrusion detection system:

The following are the basic needs that a intrusion detection system can fulfill,

- **Controlling the administrator activities:**

Network intrusion detection systems can act as an additional control tool which can check unauthorized configuration changes by the hosts that have been granted administrative privileges.

- **Protection against viruses:**

There has been an alarming increase in the number of viruses and worms that now invade the internet and affect numerous computers every day. Worm epidemics has demonstrated the danger of underestimating the dangers of malicious programs.

- **Detecting unknown devices:**

A network intrusion detection system can help in identifying the address of unknown/external hosts within the protected network segments. It can also detect increased traffic and special kind of activities from specific workstations which were not involved in such kind of activities before. Such activities can be a hint to malicious activities from the hosts and the network administrator must be informed about this.

- **Analyzing the information follows:**

Situations where the communication specialists have no trustworthy information on the protocols used in the protected network segments. A intrusion detection system can control all the protocols and services used in the corporate network, as well as the frequency of their use.

- **Analyzing data from the network equipment:**

Log files from routers and other network equipment can serve as an additional source of information on the various attacks that a data network can be prone to. A network intrusion detection system can be configured to do this work. The task of collecting such log file information and analyzing logged security events can be delegated to the intrusion detection system, which in this case, serves as a Syslog server. It can centralize such tasks of collecting log-file information and detect attacks and misuse of the network[2].

III. STRING MATCHING ALGORITHM

At the heart of every modern network intrusion detection system there is a string matching algorithm. The network intrusion detection system uses the string matching algorithm to compare the payload of the network packet and/or follow against the pattern entries of the intrusion detection rules, which are a part of every network having a network intrusion detection system. String matching needs significant memory and time requirements. In fact, the performance of all network intrusion detection systems depends almost entirely on the performance of the string matching algorithm[4]. String matching generally consists of finding a substring (called a pattern) within another string (called the text). The pattern is generally denoted as,

x = x[0..m - 1]
whose length is m and the text is generally denoted as,
y = y[0..n -1]

whose length is n . Both the strings- pattern and text- are built over a finite set of characters which is called the alphabet and denoted by Σ. whose size is denoted by σ. While searching the pattern within the text, at one time, we consider a subset of the text generally with the help of a window whose size is equal to m. Then the window is aligned with the pattern and they are matched for equality- either from the right or from the left- this specific work is known as an attempt. If they completely match, then either the algorithm ends or it continues to find any more occurrences of pattern in text. If they do not match, then the window is shifted to a new position. This is known as sliding window mechanism.

A. Boyer-Moore Algorithm :

This is an algorithm that searches for the location of the first occurrence of pattern in text. During the search operation, the characters of pattern are matched starting from the last character of pattern. The first and simpler pre-computation step done by the Boyer-Moore algorithm is the creation of the bad character shift table. This table is essentially an array indexed by all characters in the alphabet storing integers that represent how far the algorithm may shift upon a mismatch. At all characters indices not in the keyword a value equal to the keyword's length is stored since if a character in the input is encountered that does not appear in the keyword we can shift entirely past it[3]. All table positions of characters present in the keyword store the distance from the right most character in the keyword. Therefore, if the character a started the keyword and did not appear anywhere else in the keyword then the bad character shift table's (call it bctable) value at position a bctable[a] would be m − 1. When the character does exist in the keyword this bad character shift allows the algorithm to immediately realign the keyword's right most appearance of the character that was mismatched to the character that cause the mismatch in the input.

The first notable difference of this algorithm is that it starts matching the characters from right to the left instead from left to right as in most string matching algorithms. When there is a mismatch, the window needs to be shifted and this algorithm uses two pre-computed functions corresponding to the good-suffix shift and the bad-character shift. Suppose, while the character comparison, we get to a point when the characters in the pattern and the text do not match, that is, ≠ . Also, in this condition, we have already matched some of the trailing characters of the pattern with that of the text, that is, . In this case, we employ the good-suffix shifting. We align the segment of the text with the right most occurrence in the pattern under the condition that it is preceded by a character different from the character at which the mismatch had occurred, that is, different from . This type of shifting is known as the good-suffix shift because there exists a suffix of the pattern which has already matched with the text. After the shifting, instead of starting the character comparisons from the extreme right of the pattern, we can start the comparison from suffix.

Under the bad-character shift, we align the character with the right most occurrence in . In some cases, does not occur anywhere in . In these cases, the window is just shifted by one position, that is, the window is shifted from to . The bad character shift can sometimes be negative, that is, the window can get shifted to the left instead to the right. So the Boyer-Moore algorithm uses the maximum of the bad-character and the good-suffix shift.

IV. STRUCTURE OF IDS

➢ *Design :*

The IDS developed is based on signature-network. An IDS comprises of Management console and Sensors. Management console is the management and reporting console. Sensors are agents that monitor hosts or networks on a real time basis. Our IDS has a database of attack signatures. The attack signatures are patterns of different types of previously detected attacks.

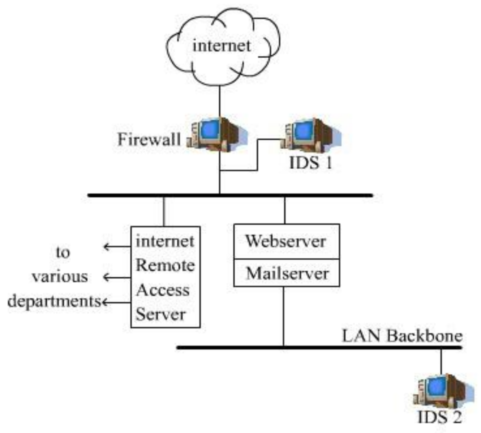

Figure 1: Schematic diagram of IDS placements

The snort sensors are placed in a key position to monitor all internet based traffic in and out of the organization. IDS-1 is a popular placement of IDS which detects attacks that successfully penetrate the firewall. By placing intrusion detection systems (IDS-2) throughout a corporate network, attacks by insiders will be detected [4].

➢ *Framework of IDS* **:**

As mentioned in figure software packages are integrated to form the following Intrusion Detection System framework.

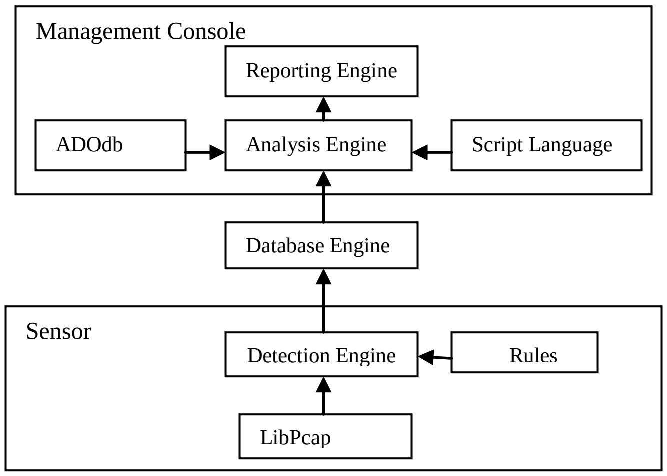

Figure 2: Framework of IDS

a) Packages :

- Snort - Detection Engine

Snort fills an important "ecological niche" in the realm of network security: a cross platform, lightweight network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attacks.

- libPcap- Packet Capturing Library

libPcap is an architecture for packet capture and network analysis for the Linux platforms. It includes a kernel-level packet filter, a low-level dynamic link library ,and a high-level and system-independent library .

- PHP- Web Based General-Purpose Scripting Language

PHP stands for Hypertext Pre-processors. PHP's strongest feature is its database interfacing capability. Connecting a database to the Internet has never been so easy. It supports many of the most popular database servers on the market, including MySQL, Oracle, Sybase, Generic ODBC, and PostgreSQL, to name a few.

- ADOdb- Database Abstraction Library

ADOdb stands for Active Data Objects Data Base. PHP is a wonderful language for building dynamic web pages. Unfortunately, PHP's database access functions are not standardized. Every database extension uses a different and incompatible API[5].

- Apache-Web server

Apache is the name of the software that allows you to run a web service on a server. Apache is very popular and provides access to most web sites on the internet. A recent survey of Web Servers around the world placed Apache Powered sites at over 50 percent of the total.

- ACID-PHP-Based Analysis

Analysis Console engine for Intrusion Detection (ACID), is a PHP-based analysis engine to search and process a database of security incidents generated by security-related software such as IDSs and firewalls.

V. OVERVIEW OF GPU AND CUDA

A. Graphical Processing Units **:**

The constant global demand for high-definition graphics in 3D games and real time visualization for applications have lead to massive investments in developing Graphic Processing Units (GPU) for harnessing computational power in an extremely efficient parallel processing computational model. The GPUs have grown almost exponentially in processing power and memory bandwidth [6].

B. CUDA:

CUDA is a parallel computing architecture that enables programmers to use both CPU and GPU processors to cooperate in a single program, using a computing paradigm known as heterogeneous computing. Software developers are able to program general purpose functions or routines to be run on the GPU by simply use \C for CUDA" while the rest of the program is still executed in the CPU.CUDA is the parallel computing architecture developed by NVIDIA which allows developers to use 'C for CUDA' language to code parts or functions of a general purpose program. The main three key abstractions that are available to the programmer as the C extensions are: a hierarchy of thread groups, shared memories and thread barrier synchronization.

VI. PACKET

A packet as a unit of data is routed between an origin and a destination on the Internet or any other network. The entire file downloads, Web page retrievals, email, all these Internet communications always occur in the form of packets. When any file is sent from one place to another, the Transmission Control Protocol (TCP) layer of TCP/IP divides the file into chunks of an efficient size for routing. Each of these packets is separately numbered and includes the Internet address of the destination. Basically as series of digital numbers, it conveys the following:

- The source IP address and port numbers;
- The destination IP address and port numbers;

Each packet header contains the proper protocols, the originating address, the destination, and the packet number. Routers in the network look at the destination address in the header and compare it to their lookup table to find out where to send the packet. For the purposes of this work, the adoption of this nature of protocol was used.

A. *Packet Capturing :*

The first phase of any network intrusion detection system is packet capturing. All data in the network are transmitted in the form of a packet, which comprises of a packet header, packet data. The packet header consists of several Open Systems Interconnection (OSI) layer information, checksums, fragmentation flags and offsets, source and destination **IP** addresses. source and destination port numbers, etc.; the packet data consists of the payload. The raw packets thus captured are processed to extract the source and destination addresses, source and destination ports, protocol information, and the packet payload all of which are essential for detecting intrusions. The information that is extracted is stored for comparison and reassembling the packet later.

B. Transferring packets :

The simplest approach is to transfer each packet to the GPU separately. However, there is an overhead associated with a data transfer operation to the GPU and many small transfers to the GPU result into a performance decreasing. Therefore the following approach is designed to achieve better performance for packets transferring. Packets obtained by CPU are collected in a buffer for some time. When CPU decides that it's time to process packets it transfers a batch of packets to the GPU for processing[7].

VII. ARCHITECTURE

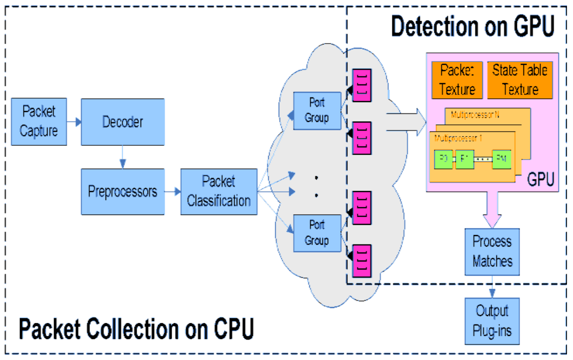

Fig 3: Architecture of Packet Processing on GPU

The overall architecture is shown here where we separate the architecture of our system in three different tasks: the transfer of the packets to the GPU, the pattern matching processing on the GPU, and finally the transfer of the results back to the CPU.

A. Transferring Packets to GPU :

The first thing to consider is how the packets will be transferred from the network interface to the memory space of the GPU. The simplest approach would be to transfer each packet directly to the GPU for processing. However, there is overhead associated with a transfer in the form of batching which require many small transfers into a larger one ,so it is much better to transfer directly.

We use a separate buffer for temporarily storing the packets of each group. After a packet has been classified to a specific group, it is copied to the corresponding buffer. Whenever the buffer gets full, all packets are transferred to the GPU in one Operation.

B. String Searching and Pattern Matching on the GPU :

Once the packets have been transferred to the GPU, the next step is to perform the pattern matching operation. In a NIDS the main component is the string matching algorithm. The objective of the algorithm is to search for a string (also called a pattern), or multiple strings. Pattern matching algorithms will search through a text for each known pattern individually, which means they will loop through the whole text as many times as the total number of patterns. example of single pattern algorithms is Boyer-Moore(BM) .

C. Transferring the Results to the CPU :

Every time a thread matches a pattern inside a packet, it reports it by appending it in an array that has been previously allocated in the device memory. The reports for each packet will be written in a separate row of the array, After the pattern matching execution has finished, the array that contains the matching pairs is copied to the host memory.

➢ **Cat Karat Packet Builder :**

Cat Karat Builder is a handy, easy IP4,IP6,IP4/IP,TCP,UDP,ICMP,IGM,ARP etc packet generation tool that allows to build custom packet for firewall or target testing and has integrated scripting ability for automated testing. This packet builder enables the user to specify the entire contents of the packet from the GUI. In addition to building packets ,it also supports saving packets to packet files and sending packets to network .It can be used at all kind of network areas like traffic generator, packet generator or protocol simulator.

EXECUTION:

For optimizing the code, the best way to transfer data into the GPU is to perform one large memory transfer requiring to only access memory once. This is quite important as for every network packet the information has to be copied over. Saving processing time is our number one objective.

The last step that needs to be done before launching the GPU kernel is to specify memory space where the kernel is to store the output of the algorithm it will be running. This memory space is reserved and GPU memory address pointer stored on the CPU side, and then sent to the GPU as a kernel parameter. This has to be done because the kernel does not have an actual output back to the calling function. We have to manually retrieve the data that now has been placed in the reserved memory space on the GPU from our CPU function that started the kernel in the first place. To save memory transfer bandwidth, the output of the algorithm is purely the ID of the pattern that was found and the location of the input text where it was located.

The final operations on the output are carried out by the CPU.

ANALYSIS:

a) CPU versus GPU Time Ratio *:*

The analysis is done using string searching and pattern matching algorithm in which every character gets check with the stored database information. A packets have an header and data information sections, each sections have their own fields and analyzing each field is very time consuming task so that part of work is assigned to the GPU that increases the performance with much better speedup.

Following graph and table shows the comparison status of time Complexity required for CPU and GPU with different scenarios of Packets capturing and analysis

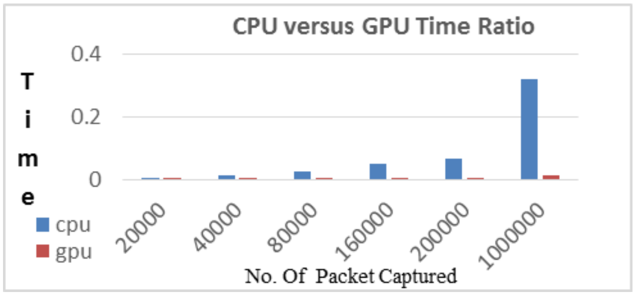

Table 1: CPU versus GPU Time Ratio

No	Packets	CPU Time	GPU Time	Speedup
1	20000	0.007159	0.000366	19.56011
2	40000	0.01263	0.000674	18.73887
3	80000	0.028274	0.001266	22.33333
4	160000	0.052642	0.002446	21.52167
5	200000	0.066887	0.003039	22.00954
6	1000000	0.321635	0.014848	21.66184

VIII. CONCLUSION

According to the analysis of the results obtained from application, we can conclude that :

a) The Intruder hosts are identified by detecting malicious packets successfully.

b) String searching and matching algorithm that is boyer moore works efficiently on GPU than CPU.

b) Transferring Complex part of work to the GPU resulting in much more speedup.

IX. FUTURE SCOPE

In future, extensive complex analysis could be carried out using the less complex algorithm, in order to gain more better performance in network security system.

REFERENCES

[1] Information Assurance Technology Analysis Center. Intrusion detection systems, 2009.

[2] Marc Sune Clos. A framework for network traffic analysis using GPUs. 2010.

[3] NVIDIA. NVIDIA CUDA C Programming Guide 4.2. 2012.

[4] CUDA Zone.http://www.nvidia.com/cuda

[5] A GPU-accelerated network performance monitoring system for large scale scientific collaborations. In IEEE LCN'11, 2011.

[6] Nvida CUDA programming Guide: http://kr.nvidia.com/object/cuda_develop_kr.html.

[7] NVIDIA. NVIDIA CUDA C Programming Guide 4.2. 2012.

AUTHORS

First Author – Researcher Scholar: Miss. Ambekari Tabassum Dawood, Annasaheb Dange College Of Enginnering, Ashta ambekari.tabassum28@gmail.com
Second Author – Guided By: Mrs A.N.Mulla, Annasaheb Dange College Of Enginnering, Ashta