DPI based Forensic Analysis of Network Traffic using Grid Infrastructure

A thesis submitted
for the award of the degree of
DOCTOR OF PHILOSOPHY

by

Jyotsna Sharma
(90703503)

under the guidance of

Dr. Maninder Singh
Associate Professor
Computer Science and Engineering Department
Thapar University, Patiala -147004

Computer Science and Engineering Department
Thapar University, Patiala – 147004, INDIA

March, 2016

Dedicated to

the Supreme power

Certificate

I hereby certify that the work which is being presented in this thesis entitled *DPI based Forensic Analysis of Network Traffic using Grid Infrastructure*, for the award of degree of Doctor of Philosophy submitted to the Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Maninder Singh and refers other researchers' works which are duly listed in the references section.
The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Jyotsna Sharma)
Reg. No. 90703503

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Maninder Singh)
Associate Professor
Department of Computer Science & Engineering
Thapar University, Patiala 147 004, Punjab, INDIA

Acknowledgements

I express my gratitude to the many people who have helped and guided me with this research. First and foremost, I thank my advisor, Dr. Maninder Singh, a highly learned academician and a thorough gentleman with great human qualities, who has exceptional positive attitude and energy. This thesis would not have been possible without his invaluable sagacious research guidance and continuous support. He continually stimulated my analytical thinking and greatly assisted me with scientific writing. The opportunity to learn under his guidance has been like a blessing.

I am very grateful to Dr. Deepak Garg, Associate Professor and Head, Computer Science and Engineering Department, Thapar University, Patiala, for providing all the necessary guidance, advice and infrastructure to carry out the research work. Thanks to Dr. Rajesh Kumar for his valuable advice and critical observation that greatly contributed to the improvement in the thesis and research work. Dr. Anil Kumar Verma has always been there with his encouraging and generous advice, for which I am so grateful. I am also deeply indebted to Prof. Dr. Seema Bawa, for her efforts in guiding and supporting my study of Grid Computing. Her suggestions, ideas and comments were more

than helpful to me and did never stop. Thanks to Dr. Inderveer Chana who always obliged whenever any help and advice was needed. I also register my gratitude to Dr. Prateek Bhatia who provided important feedback on the work.

I acknowledge Dr. Inderpreet Singh for his genuine support, ideas and enriching technical discussions. With his help the Globus setup was a breeze. I express deep appreciation to my fellow colleague, Ratinder Kaur, for her continuous support, the much needed motivation and her readiness to help at any time of the day. I wish to heartily thank Dr. Shashi Bhanwar who provided incredible help and support. I am also grateful to Dr.Rajni Aaron and Dr. Sanmeet Bhatia for their generous support.

I would like to express my deepest gratitude to the people I love very much and to whom I dedicate this thesis: First and foremost my mother, Darshan Sharma, who has unconditionally supported me at every stage of my life and has always been a source of inspiration and motivation; my husband, Ajay Sharma; and my brother, Aditya Sharma who have aided me during my studies and have literally taken pains to help me in every possible way. I acknowledge the support of my aunt, Raj Kumari Sharma, who had the belief that I have the ability for greater achievement in my life. I am thankful to my aunt, Kusum Grover for her support and motivation. I miss my father, Late Sh.A.C.Sharma,a highly educated man, who would have felt happiness with my endeavours for academic fulfillment. Finally, my heartiest ac-

knowledgement of gratitude to my lovely children, Ananya, who always supported and motivated me and Aditi, who exhibited exceptional patience while I dedicated long hours to the writing of the thesis. The girls interspersed the long struggling days of research, with their smiles, laughter and antics.

Finally, I feel overwhelmed with the sense of gratitude for the great teachers who inspired me throughout my academic life right from the school time. I remember many of them who displayed confidence in my abilities and instilled into me the sense of hard work, sincerity and the thirst for knowledge.

Jyotsna Sharma

Abstract

Security threats have evolved from simple attacks such as virus infections to more sophisticated ones like the Application-layer buffer overflow, DDoS, Phishing and many zero-day variants. Such threats have significantly altered the requirements for modern network security architecture. To detect and prevent these threats, a completely new kind of security system is required which is highly proactive as well as reactive. To protect a network from complex, sophisticated attacks, the security system should have the ability to learn from the behaviour of the past attacks and get prepared to thwart similar attacks or attacks from similar sources in the future. A quick response time for such a detection, analysis and learning system is the key to a strong and reliable security system.

Intrusion Detection systems(IDS) and Intrusion Prevention Systems (IPS) monitor network and/or host activities for anomalous behaviour and react in real-time to block or prevent them. Traditional IDS/IPS use signature matching or anomaly detection techniques which work fine for known attacks but fail to detect new attacks. Another drawback is the generation of too many false positive alerts in which the IDS mistakes legitimate traffic for an attack. An Intrusion Detection

System based on Deep Packet Inspection (DPI) technology, where the appliance has the mechanism to look within the application payload of the traffic by inspecting every byte of every packet, has the ability to detect intrusions which are more difficult to detect as compared to the detection of simple attacks.

The real-time monitoring of the payload at any level requires significant human and hardware resources, and does not scale to networks larger than a single workgroup. It is more practical to archive all traffic and analyze subsets as necessary. The process, also known as reconstructive traffic analysis, or network forensics, can enhance the security of the network and also be useful for the investigation of the attacks. Forensic Analysis can analyze network traffic and lead to the source of attacks. The attribution of an attack to a host or a network helps in predicting and preventing future attacks. Attack patterns can be created from existing knowledge. The identification of attacks, and their categorization thereafter, followed by attack reconstruction can help to prevent a similar attack in the future. The traceback to the source of the attack also aids in criminal investigation and legal prosecution of the attackers.

In this work a DPI based network forensics analysis framework has been proposed for performing reconstructive traffic analysis to analyze the traffic according to the user's needs and discover useful and interesting insights into the analyzed traffic to protect against future attacks. The network traffic is captured and DPI based intrusion detection is

performed to identify attack and create evidence. This network based evidence alongwith the host based evidence(logs, file checksums), is used to create forensic profiles using the data mining tools. Bayesian algorithms and decision trees are used for the machine learning for creating the forensic profiles. The essence of a pattern-recognition DPI model, is a multiple pattern-matching algorithm. where the payload is inspected to detect malicious patterns. Fast string matching is the key element to DPI based IDSs and is a vital component for earlier attack detection. The popular algorithms are studied so that the best possible pattern-matching algorithm for the work can be selected. The quest for the further improvement in the performance of the algorithm, led to the exploration of the possibilities of mapping the algorithm to a GPU and after a careful review of the related literature, the Rabin-Karp multiple pattern matching algorithm is found most suitable for this work. The implementation on the GPU is done in CUDA.

The storage and processing requirements for analysis of the huge volume of network traffic, are high. Large storage capacities are needed to store the captured traffic and high processing power is needed to process this huge volume of data. Adding special hardware translates to high costs as the data requirements for even medium size networks soon run into terabytes. Grid architectures are analyzed in the research that lead to the attempts for leveraging the Grid Service technology as an answer to the high storage and computational requirements of the framework. The convergence of web services and grid computing

simplifies the design, development and deployment of grid services in virtual organizations with diverse compute and resource characteristics. Thesis work focuses on the realization of flexible and scalable service offers in a Grid environment, therefore many technology frameworks are considered to find the best choice as the web service architecture for the proposed model. An established and well supported Service Oriented Architecture based on Web Services Resource Framework(WSRF) has been presented, which can ensure long term scalability and extensibility of the application. Thesis gives an overview of the many currently available Grid service standards and supports the selection of the WSRF and describes the implementation of the WSRF compliant java web services on the Grid for the machine learning functions for the framework, such as classification and clustering.

The forensic analysis framework is designed based on behaviour-based machine learning, also known as Statistical modeling. Supervised machine learning is performed by training the classifier with the datasets created after the intrusion detection. The cross-validation techniques build a model which can be used to predict intrusions on newly captured datasets. The classifier is trained with a labelled dataset and several machine learning algorithms are tested before selecting the one with the lowest error rate and moderate model building time. The model is validated with the standard measures like F-Measure, Accuracy, Precision and Recall, on a Grid Prototype setup using Globus Toolkit 4.0 at the Thapar University Grid lab.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In this chapter, a general introduction to the network security threats and network security approaches is given to identify the necessity of a network forensics framework. The concepts of Deep Packet Inspection (DPI) are discussed and an overview of the various techniques to perform DPI is given. The types of network traffic analysis, viz. Realtime Analysis and Forensic Analysis are discussed. The introduction to the technologies used in the framework to make it a high performance, scalable and reliable framework, viz. Graphical Processing Units (GPUs) and Grid Computing is given. Efforts have been made in this research to leverage the abilities of these two technologies for designing the framework which attempts to meet the requirements of a General Network Forensics system and deliver the best possible performance while addressing the issues and constraints faced in the classical approaches.

Chapter ends with a discussion on the research motivation and the description of the organization of the rest of the thesis.

1.1 Network Security threats

Network Security threats multiply and become worse with each passing day. Network Security threats can be categorized into two types:

1.1.1 Malware

Malware is malicious software that damages a computer without the knowledge or permission of the user. It includes viruses, worms, trojans, rootkits, and spyware.
Computer viruses and **Computer worms** are self-replicating. Worms differ from a virus as they dispatch copies of themselves to other computers over a network and are not required to be attached to any existing programs.
Trojan Horse Viruses pretend to do one thing, but when loaded, deliver a malicious payload which might allow a hacker remote access to the attacked system, start a keystroke logger to record every keystroke, capture passwords/information, plant a backdoor on the system, destroy files, cause a denial of service (DoS), or even disable the antivirus protection or software firewall. Such malware can spread through Email attachments, Peer-to-peer networks (P2P) such as Kazaa and Gnutella, Instant messaging (IM), Internet Relay Chat (IRC) and Wrappers which bind a Trojan program to a legitimate program.
A **rootkit** takes unauthorized fundamental control (Root/Administrator access) of a computer system. It has the ability to mask its presence.
Spyware gets surreptitiously installed on a computer to intercept the user's interaction with the computer, without the user's informed consent.
Modern malware include **keyloggers**, **dialers**, **adware**, **ransomware** and **browser**

hijackers.

1.1.2 Network intrusions

Network intrusions consist of packets introduced intentionally and specifically to do one or all of the following:

- To consume resources uselessly
- To interfere with any system resource's intended function
- To acquire system information (which can be later used in attacks)

The most commonly used protocol at the network-layer, Internet Protocol (IP), forwards IP packets across nodes. An IP packet has an IP header and a data section called the payload. The transport layer adds a TCP header to the IP packet. These headers contain fields that hold information such as the source/destination IP address and source/destination port numbers. These fields and/or the payload could be modified by an attacker to create attacks on the network.

Types of Network Intrusions: A few major types of network intrusion threats are discussed below:

- ***Sniffing*** gives the attacker a way to capture data and intercept passwords.
- ***Session Hijacking*** is an attack in which, the attacker actively injects packets into the network to take over an authenticated connection to erase or modify information.

- ***Port Scanning*** is a reconnaissance technique where a hacker can scan the ports available on the victim. It leads to the discovery of services that can be broken into.

- ***Denial of Service (DoS)*** attacks can prevent legitimate users of a service from accessing that service. These include, Smurf/Fraggle attack, which uses band-width consumption to disable a system's network resources by sending a large amount of ICMP/UDP echo traffic at IP broadcast addresses [Gre06] ; SYN Flood attacks, which use resource starvation to achieve the DoS; DNS Attacks, which temporarily modify the DNS records on DNS servers such that queries destined for the original web site divert to the fake web site; and IP DoS attacks (Teardrop or Land attack), which exploit the IP protocol by sending malformed or poisoned packets.

- ***Distributed Denial of Service (DDoS)*** attack is a variation of the basic DoS attack, in which a system is flooded with requests falsified in such a way that the server uses up more resources trying to set up illegitimate connection than it would spend in setting up legitimate ones. Reportedly, there have been such attacks on high profile websites like Amazon.com, eBay and Yahoo [Sto01].

- ***Buffer overflows***, send a larger than anticipated quantity of data to the server. It can cause the server to crash or execute malicious code from the attacker.

- ***SQL injection*** attack involves injection of malicious SQL queries into user input forms, which can directly manipulate a database. The malicious code

is inserted into strings passed to SQL server for parsing and execution. An example of an SQL injection is quoted from [KGJE11]. In this example, an SQL injection attack is conducted through a login form where "OR 1=1" is submitted in the input login field and an empty value is submitted for the password field.

```
login=' ' OR 1=1 AND pass=' '
```

This statement is always true, making the application infer that authentication of the user is successful. As a result, the user(hacker) is granted access even without any real credentials.

- ***Cross Site Scripting (XSS)*** vulnerability is typically seen in search engines, http error messages, input forms and web message boards. It can potentially impact any site that allows users to input data. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to an unsuspecting end user. It works by embedding script tags in HTTP requests and inviting victims to click on them. The malicious script then gets executed on victim's machine.

- ***Cross Site Request Forgery(CSRF)*** is another kind of injection attack. Also known as XSRF attack or Session Riding attack, it can trick the user to execute actions of the attacker's choice on a web application. Social Engineering when used in conjunction with such an attack can enable the attacker to inherit the identity and privileges of the victim and perform malicious actions on the victim's behalf.

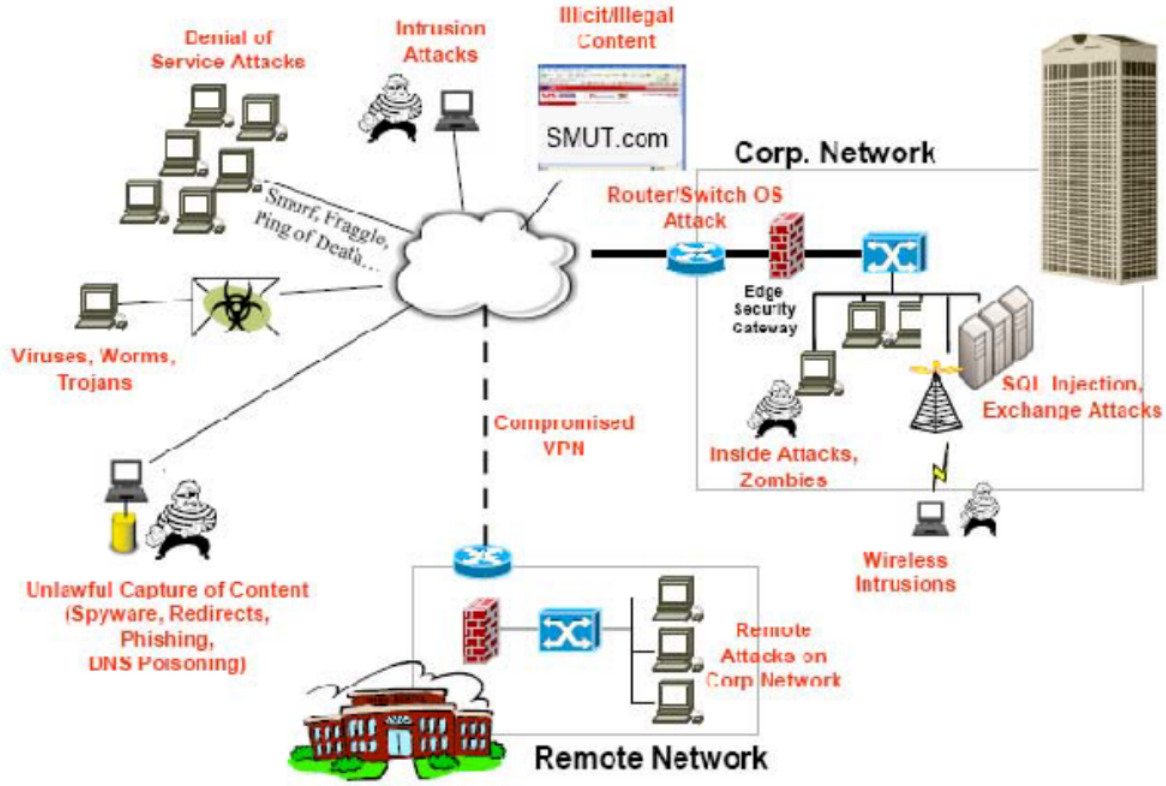

Figure 1.1: Network Security Threats

1.2 Network Security Approaches

There are two types of network security approaches to meet the network security challenges. The two approaches are described as follows:

- **Reactive Network Security Approach**

  Reactive Network Security Approach is where organizations use procedures once they discover that some of their systems have been compromised by an intruder or an attack program. The responses to an attack with this

approach include Recovery Plans, re-installation of operating systems and applications on compromised systems, hiring private investigation services and loss recovery specialists, or switching to alternate systems in other locations. Network Forensics appliances can passively record network traffic and provide useful data for network monitoring. Proper and indepth analysis can help lead to the source of attacks and present the possibility to mitigate further attacks.

- **Proactive Network Security Approach**

  The Proactive Network Security Approach keeps an eye on the vulnerabilities in the network security and develop preventive measures for safeguarding against the security threats. An Intrusion Detection System (IDS) would block traffic detected as malicious. The common techniques used for the proactive approach are antivirus, firewalls, honeypots and Intrusion Detection/Intrusion Prevention Systems (IDS/IPS).

1.3 Security Mechanisms

Traditional security methods that include antivirus softwares, firewalls and intrusion detection systems (IDS) are easy and simple to employ but suffer from several limitations. Deep Packet Inspection (DPI) based IDS addresses many of the limitations of these traditional methods.

1.3.1 Anti-malware Software

Anti-malware software, more commonly known as antivirus software can detect and remove malicious software. Originally developed to protect from computer viruses, antivirus software can protect against other malware too. The most common techniques adopted are signature-based detection and heuristic-based detection.

Limitations of antivirus software: Undoubtedly, anti-virus software are an integral part of a network security mechanism and are quite sophisticated too, but the virus programmers are always a step ahead of the anti-virus software. Anti-virus software employ scanning and integrity checking which are quite effective against known viruses but completely ineffective in detecting new viruses.

1.3.2 Firewalls

Firewalls inspect traffic as it enters and exits the network. They enforce an access policy by operating as a gateway between networks. **Packet filter firewalls** decide if a packet passing though the link be allowed to proceed or not, by filtering on IP addresses. **Stateful firewalls** track the status of a connection. If an arriving ACK packet claims to be from an established connection, the stateful firewall would deny it if it did not have a record of the three-way handshake ever taking place. **Proxy firewalls** terminate flow at one side, provide some service and recreate the flow at the other side.

Limitations of Firewalls: Firewalls cannot recognize and block all attacks. They have been found to be ineffective against smart attacks. Firewalls are usually located at the edge of the network, and therefore cannot stop attacks that originate inside the network perimeter [Sch07] and cannot protect against backdoor attacks [RK10]. Hackers can bypass the firewall and gain access through an unsecured wireless point. Attacks such as port-scanning, firewalking and banner grabbing, can be used by an attacker to exploit the weakness of a firewall [Gre06]. Software Firewalls that are built into the kernel, cannot detect stealthy programs. Hardware Firewalls allow only outgoing connections from the machine. The limitation is that they cannot monitor outgoing traffic and prevent connection hijack attacks which mimic responses instead of initiations. To summarize, it can be said that a firewall alone is not a dynamic protection system.

1.3.3 Intrusion Detection Systems (IDS)

Another way to address security concerns is deploying an Intrusion Detection System (IDS) which is much more dynamic than firewalls. It can detect and draw attention to odd behaviour. IDS could be passive or reactive. In a passive system, the IDS sensor detects a potential security breach, logs the information and signals an alert. In a reactive system, also known as an Intrusion Prevention System (IPS), the IDS responds to the odd activity by resetting the connection or by reprogramming the firewall to block traffic from a suspected source.

Types of IDS : There are two types of IDS based on the way data is collected [YH03] (See Figure 1.2).

- **Network intrusion detection systems (NIDS)** - An NIDS monitors a

network and discovers intrusion by matching an attack pattern to a database of known attack patterns.

- **Host based intrusion detection system (HIDS)** - An HIDS monitors the actual target machines, by monitoring event, security and systems logs or checking for changes to the system. When the HIDS detects any change in any of the files, it compares the new log entry with its configured attack signatures to check if there is a match. If a match is detected, it is an indication of an attempt of intrusion.

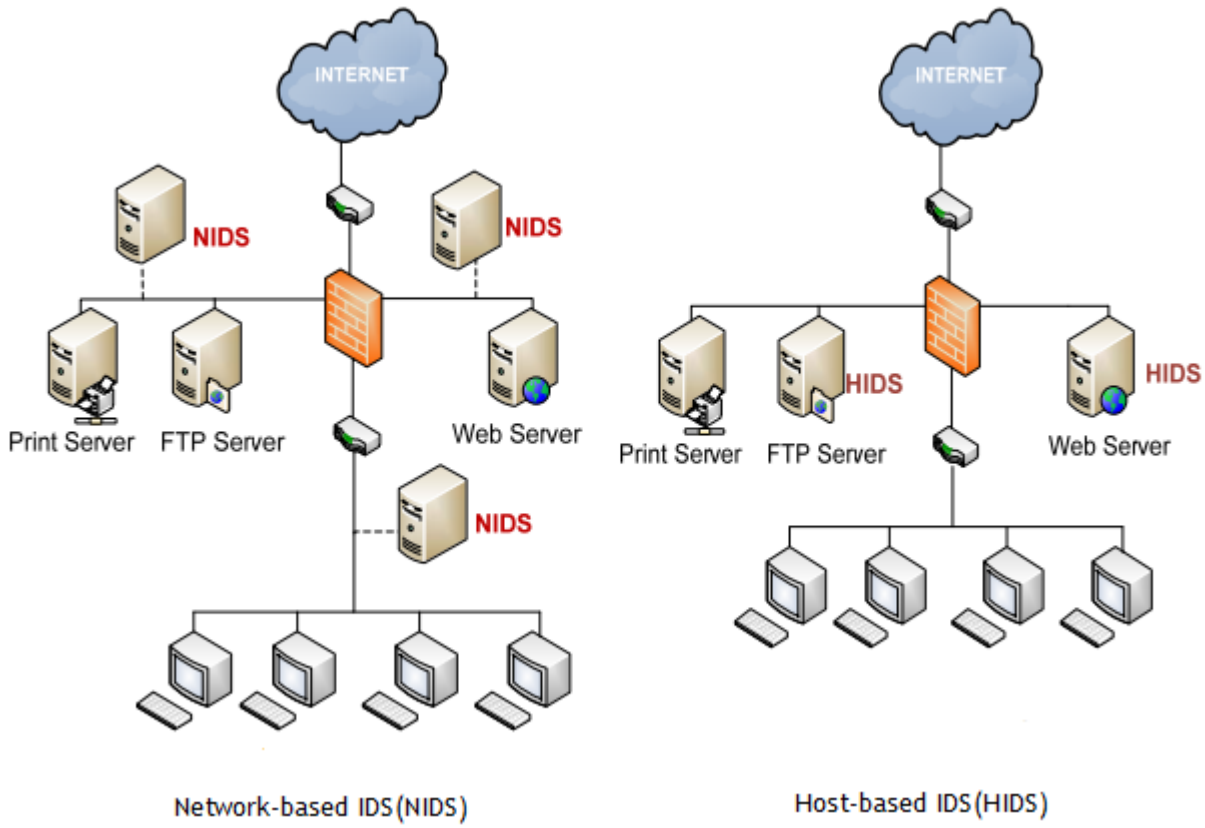

Figure 1.2: Network-based IDS(NIDS) and Host-based IDS(HIDS)

Working of NIDS : Network Intrusion Detection works by three ways :

- **Anomaly detection:** The administrator specifies a baseline of statistics such as CPU utilization, file/disk activity, user logins, etc. by making use of profiles of authorized activities or by placing the IDS into a learning mode so that it can learn what constitutes normal activity. An alert is generated when there is a deviation from the baseline.
- **Signature recognition/ Pattern Matching:** Known attack patterns are loaded into the system as signatures and are given an identification. Alerts are triggered for fragmented IP packets, malformed Internet Control Message Protocol (ICMP) packets, or streams of SYN packets (DoS).
- **Protocol decoding:** This IDS can pick out abnormal activity, if it knows the normal activity of the protocol.

Limitations of IDS: Early IDSs used signature detection. They detected all the attacks captured in their signature databases, but suffered from unacceptably high false-alarm rates [NST02]. The anomaly detection IDS fails horribly in detecting malicious activity that does not violate an accepted behavioral norm [BROS12]. Attackers can use a range of techniques such as Flooding, Evasion and Session Splicing, as an attempt to prevent detection by the IDS. *Flooding* refers to the attempts to overload the IDS by flooding it with traffic. *Evasion* occurs when an IDS discards the packet that is accepted by the host it is addressed to. As an example, TCP segments marked with a SYN flag might also include data. As this is an infrequent occurrence, an IDS might ignore the contents of these packets, thereby allowing the packets to go undetected. *Session Splicing* works by delivering

the payload over multiple packets, which defeats simple pattern matching without session reconstruction. This payload can be delivered in different manners and even spread out over a long period of time. The inability of the IDS to keep all fragments in memory for reassembling, can lead to an attack being unnoticed.

1.4 Deep Packet Inspection

Deep Packet Inspection (DPI) addresses the limitations of firewalls and traditional IDS techniques. DPI gives a complete view of the traffic. This is accomplished by reassembling IP datagrams, TCP datastreams and UDP packets as they flow through the device. DPI looks at not just the header but also at the payload of the packets. Figure 1.3 and Figure 1.4 are given to differentiate the concept of Deep Packet Inspection of the network data from the Shallow packet Inspection.

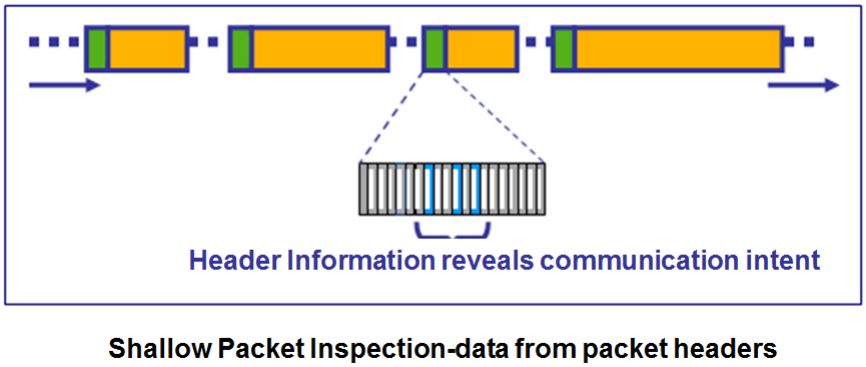

Figure 1.3: Shallow Packet Inspection

The main difference between DPI and a traditional firewall is that a firewall only looks at layer 3 and sometimes layer 4 of the OSI protocol stack, while DPI operates at Layer 3-7 of the OSI model. Figure 1.5 explains this concept.

Any system that performs DPI relies on the ability to compare bytes within the

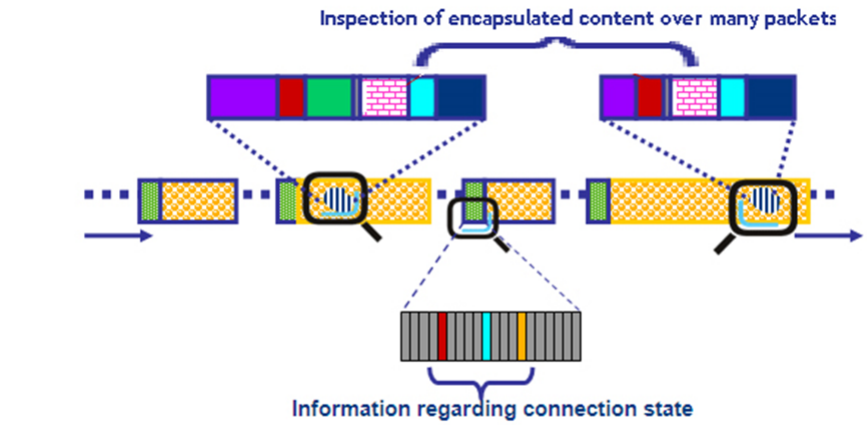

Figure 1.4: Deep Packet Inspection

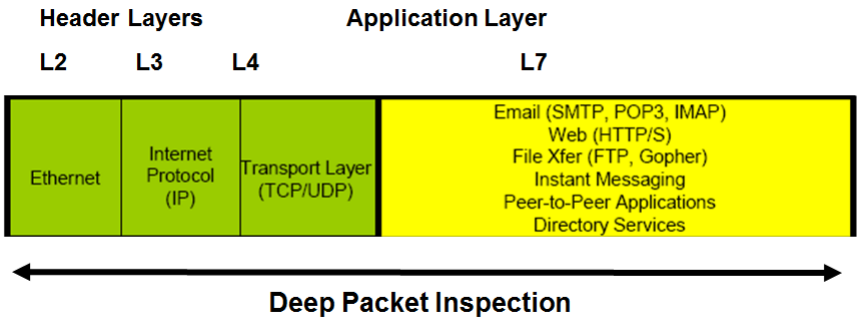

Figure 1.5: DPI (Layer 2 to Layer 7)

payload to a known list of malicious bytes. Fast string matching comparison is a key element to DPI based IDSs and is a vital component for earlier attack detection. While software approaches to enhancing the string comparison efficiency do exist [NTCV04], they are mostly slight improvements of the Aho-Corasick algorithm discovered in 1975 [AC75].

1.5 Network Traffic Analysis

Network Traffic Analysis can be defined as the process of capturing network traffic for the purpose of close inspection to deduce information which can be helpful for the monitoring and analysis. In the past, router-based traffic monitoring was performed with Simple Network Management Protocol (SNMP), Remote Monitoring (RMON)and Cisco's Netflow. These techniques are hard-coded into the router. SNMP and RMON can collect massive amounts of data but lack significant information that would be needed for an indepth analysis therefore they have limited use for forensic analysis. Netflow collects IP traffic as it enters the interface and has the ability to determine information like the source and destination of the traffic and the class of service. A wide variety of data analyzers are available for pulling out information from Netflow which can be used for a better analysis. Non-router based techniques involving tools like Ping, Traceroute, Iperf perform active monitoring by introducing probes into the network. This helps in collecting metrics from the active measurements but these can cause interference to the normal traffic on the network. A better and more popular technique is passive monitoring, which does not inject any probes into the network. It just sits and listens. Packet sniffers perform passive monitoring which by themselves have no overhead

and can capture huge amounts of data. This data can be analysed offline and is useful for forensic analysis. Combinational monitoring uses a combination of both active and passive monitoring techniques by monitoring actively when the traffic is low and monitoring passively during the time when the traffic is high. Watching Resources from the Edge of the Network (WREN)[LZ04] and Self Configuring Network Monitor (SCNM)[AGJT03] are two such tools.

The above discussion gives information on type of network analysis techniques. Based on the time of the analysis and the purpose of the analysis, Network Traffic Analysis can be categorized into two types as discussed in the following subsection.

1.5.1 Types of Network Traffic Analysis:

- **Real-Time Analysis**

  Real-Time Analysis of network traffic is performed over the network data as soon as it is obtained right after a security event has occurred. The main feature of real-time analysis is quick response as there is low delay in the time when the traffic analysis is done after the data is obtained. There is quick identification and retrieval of pertinent information for the analysis. This feature however demands high computational resources.

- **Forensic Analysis**

  Forensic Analysis of network traffic is performed when an event has occurred as in the case of an intrusion detection. The data needs to be captured and stored for an indepth analysis. Also known as Network Forensics, it can discover the source and nature of security threats. Chapter 2 discusses network forensics in more detail.

1.6 GPU

A Graphical Processing Unit(GPU), capable of running thousands of lightweight threads in parallel, is designed for intensive, highly parallel computations, exactly for graphic rendering purposes. Current-generation GPUs, designed to act as high performance stream-processors derive their performance from parallelism at both the data and instruction-level. In GPU's architecture, much more transistors are devoted to data processing and, less to data caching (see Figure 1.6) [WZDT10].

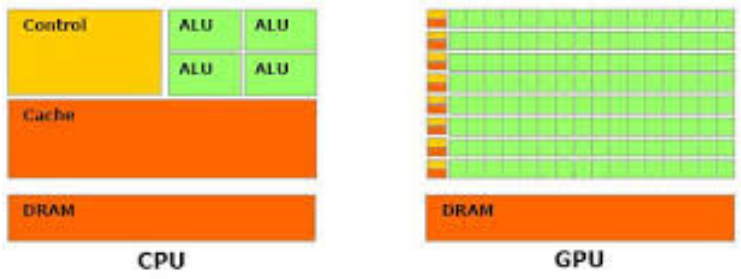

Figure 1.6: Processor Architecture(CPU Vs. GPU))

A GPU is a single-chip processor, just like the CPU (Central Processing Unit), the difference is it has a large number of cores as compared to a CPU. The majority of Intel's desktop-grade consumer CPUs have 1,2,4 or 6 cores. As on date, the fastest Intel CPU, The Core i7-5960X Extreme Edition, has 8 cores. This is a major contrast to the hundreds to thousands of cores in a GPU. The primary job of the GPU is to compute 3D functions. Because these types of calculations are very heavy on the CPU, the GPU can help the computer run more efficiently.
While talking about technologies that offer manipulation of data in a parallel manner, there are basically two types of parallelism we think of. The types of parallelism are discussed in the following paragraph:

- Task Parallelism:

  - Different tasks are performed at the same time over the same or different data

  - The parallelism is driven by the tasks

  - This type of parallelism is difficult to scale if the number of tasks is fixed

- Data Parallelism:

  - The same task is performed at the same time over different data

  - The parallelism is driven by the amount of data

  - This type of parallelism is easily scalable with data size

In this research, the data-parallel threading model is chosen, where, the same, independent operation of pattern-matching is performed repeatedly to different data by the GPU.

1.7 Grid Computing

Grid computing is a cost-effective and scalable means of solving large scale computing problems that span not only locations but also organizations, machine architectures and software boundaries to provide unlimited power, collaboration and information access to everyone connected to a grid [FK03].
A grid is a collection of distributed computing resources available over a network that appear as one large virtual computing system, to an end user or application. Workloads can be provisioned and deployed by locating available pools of computing resources. The resources of many computers in a grid can be applied to a single

problem at a time [JF04]. The most common resource is computing cycles provided by the processors of the machines on the grid, which specifically is referred to as a Computational Grid. Another type of grid, the Data Grid/Information Grid, utilizes memory and/or secondary storage in the grid to increase capacity, performance, sharing, and reliability of data while providing its access across multiple organizations [JBFT05]. Grids help optimize the infrastructure to balance workloads and provide extra capacity for high-demand applications [Cha07]. Grid computing also enables research-oriented organizations to solve problems that were infeasible to solve due to computing and data-integration constraints. It enables sharing of expensive scientific equipment too.

1.7.1 Benefits of Grid Computing:

- **Better Resource Utilization:** An organization may have unexpected peaks of activity that demand more resources. If the applications are grid enabled, they can be moved to underutilized machines during such peaks [JBFT05].

- **Virtual organizations for collaboration:** Grid computing can enable heterogeneous systems to work together to form the image of a large virtual computing system offering a variety of resources.

- **Reduction in energy consumption:** By providing a quick solution to usage changes, grid computing provides businesses with extra capability when needed, without consuming large amounts of energy [IPS08]. Using servers to capacity cuts down on the energy needed. Chesapeake Energy Corp., the largest independent producer of natural gas in the United States, who have

deployed oracle grid computing, also believes that Grid Computing supports reduction in energy consumption by enabling them to use fewer physical servers in their environment [Ora07]. Manufacturing computers is materials intensive; the total fossil fuels used to make one desktop computer weigh over 240 kilograms, around 10 times the weight of the computer itself. The environmental impacts associated with using fossil fuels (e.g. climate change) and chemicals (e.g. possible health effects on chip production workers) are significant and deserve attention [KW03].

1.8 Research Motivation

The need for a strong, scalable and quick-reacting security system is now more pronounced than ever. As computer networks continue to grow in size and complexity, there is a proportional rise in the number, variety and complexity of security threats as well. The traditional solutions like antivirus, firewalls and intrusion detection systems can detect and prevent some of them but they lack features for investigation and a deeper insight into the attack. Network Forensics is the solution meant to detect, analyze, prevent, defeat and investigate security threats. It can provide evidence for the purpose of legal prosecution while also providing knowledge which can be used for enhancing the security perimeter by gaining a deeper and better understanding of the attack. The study carried out for the research identified two major challenges, the forensic analysis task is characterized with. The objective to address these two challenges viz., **Large Quantity of Data** and **Quick Response Time**, is the motivation for the research.

1.8.1 Challenges

- **Large Quantity of Data**

  The amount of data available for the forensic analysis is directly proportional to the quality of the analysis results depending on how deep one can dig into the data to discover useful knowledge. Gathering data for the forensic analysis is not a big or expensive problem anymore, with good sophisticated tools (even free and open-source), but the cost to store the large volumes of data and then the effort to process them is huge. Raghavan[Raz13] discussed the serious challenges for digital forensic research which are equally applicable to network forensic research as well. Quick and Choo (2014) presented the statistics of the volume of forensic data examined by the US Federal Bureau of Investigation (FBI) Regional Computer Forensic Laboratory (RCFL) of the past few years (FBI RCFL 2003-12) [QC14]. The ever increasing total data volume in multiples of terabytes, is just an indicator of the quantum of the storage challenge.

- **Quick Response Time**

  A forensic analysis system should have a quick response time in order to be useful. The response time for the forensic analysis is very important as a longer delay means the network resource or service is vulnerable for that amount of time. The performance of the intrusion detection system used for the network traffic analysis should be reliable, and fast and becomes a challenge as it needs to address the 'Quantity' and 'Complexity' aspect of the network traffic.

Section 2.8 in Chapter 2 explains considerations for providing a solution to these challenges, and this has contributed to the formulation of the problem. A hybrid architecture involving the use of GPUs and Grid has been attempted to meet the research challenges. The wide availability and the price-performance benefit of GPUs make them an excellent candidate for their manipulation in tasks which call for high-performance and efficiency. The grid computing technology provides the solution for the storage issue and also enables efficient processing of the large quantity of forensic data.

The proposed approach calls for optimum utilization of the features of the technologies involved. By themselves, each of the technologies like Deep Packet Inspection, GPGPU, Grid Computing and WSRF, are very promising and have proven increased contribution towards several performance measures including speedup, efficiency, reliability and scalability, to name a few. Careful consideration of the parameters involved is required while setting up the framework so that the end result also delivers the performance as good as the individual performance of the technologies involved. This of course led to a lot of research effort and several runs of trying and testing.

1.9 Thesis Outline

This section discusses the framework of the thesis, which is organized as follows:

Chapter 1 *Introduction*

The first chapter gives the basic background information and introduces the core terminology. In this chapter, an introduction to Network Security, its need in the

current era of information technology and the challenges in the development of a network security system have been discussed. The details of different approaches to network security, concepts of grid computing, features of Graphics Processing Unit (GPU), and the advantages of General-Purpose Computing on GPU are also given. This chapter provides introduction to all these concepts. This chapter concludes by presenting the organization of rest of the thesis.

Chapter 2 *Literature Survey*

Chapter 2 extends the previous chapter to contain more theoretical backgrounds and a survey of potential solutions to provide flexible service offers in a Grid environment. The chapter presents the literature survey for meeting the first three objectives of the research. It presents the analysis of intrusion detection and prevention techniques. It discusses the need and importance of forensic analysis and includes the study of literature on the research efforts in the field of network forensics.

Additional literature reviewed in this chapter reviews and analyzes DPI as an intrusion detection technique. It includes the details pertaining the techniques for DPI, the hardware platforms for DPI and the algorithms for DPI implementation. The study of pattern-matching algorithms helps to select the most appropriate algorithm for this work. A history of GPU and its evolution and use in GPU Computing (GPGPU), is presented.

In this chapter, the history, evolution and benefits of Grid Computing, have also been provided. It also discusses the WSRF web service technology as a more useful service architecture as a choice for the Grid infrastructure. The implementation presented in this thesis, is the Globus Toolkit Java version of WS-Core. Apart

from this, there are at least four more different popular implementations of the SOAP, differentiated on the basis of the programming model and the programming language. The discussion in this chapter supports the selection of the Globus Java WS-Core from over these implementations. The chapter presents the problem formulation, lists the research objectives, and concludes by suggesting the methodology for fulfilling the objectives, in the section 'Thesis Overview' (see section 2.9.1).

Chapter 3 *Proposed Framework for DPI based Forensic Analysis using Grid Infrastructure*

The chapter presents the design and architecture of the proposed framework for DPI based Forensic Analysis using Grid Infrastructure. The various tools used to implement the framework are mentioned. The process of forensic analysis is explained followed by the design overview. The detailed architecture describes the components of the framework. A detailed discussion on the use of GPGPU for Rabin-Karp based pattern matching is followed by the design of the parallel algorithm for the GPU.

Chapter 4 *Implementation Details of the Framework*

Chapter 4 provides the implementation details of the framework on the Grid. The deployment is done on the GT4 container, as a web service based on WSRF. Furthermore, machine learning on Weka is discussed. The framework works on a classifier model which learns from the labeled datasets created after applying the DPI based intrusion detection techniques. The new datasets can be classified by the statistically built model which has learnt from the behaviour of the training datasets. The machine learning algorithms are discussed in detail. Thereafter, the

datasets' preparation for the data mining is explained. The details of the implementation of the Weka data mining algorithms on the grid are given in the chapter.

Chapter 5 *Results and Discussions*

Chapter 5 describes the test environment, deployment of the proposed model, detailed flow of work and the experiments performed. Phased validation of the framework is presented. First, the results of the performance of the CUDA DPI engine component of the framework is reported. The results of the serial implementation of the algorithm on the CPU are compared with its parallel implementation on the GPU. Thereafter, the validation of the statistical model built using Weka machine learning algorithms is done using standard statistical evaluation metrics like Accuracy, F-measure, Detection rate, Precision and Recall. Once a model has been evaluated and selected it is used to classify new unlabelled datasets. The performance of the data mining algorithms is then evaluated for the testing datasets. It reports the results and a detailed discussion on the results is presented.

Chapter 6 *Conclusions and Future Work*

Finally, Chapter 6 presents the conclusion of the thesis, describes the main contributions of the research work and also discusses the limitations of the work. The chapter also highlights the future work related to the possibilities of using a GPU Grid and making the framework completely open-source. The huge proliferation of wireless devices on wireless networks presents a new direction of wireless network forensics. Social Networking has gained an explosive growth in the recent times. Twitter, Facebook and Whatsapp have made unimaginable proliferation in usage and proportional is the increase in the security threats. The increasing number

of cases with an ever-growing amount of data calls for specialized Social Forensic Analysis research. Social Forensic Analysis research is another possible direction for future work.

Chapter 2

Literature Survey

2.1 Introduction

This chapter begins with the review of literature on intrusion detection and intrusion prevention techniques and tools, followed by the details of findings of research in the area of Deep Packet Inspection(DPI). The review of literature has been performed by studying the historic developments in this field. The literature reported in this work includes extensive details about DPI, the hardware platforms for DPI and the algorithms for DPI implementation. An in-depth study of pattern-matching algorithms is done which is very important. The selection of an appropriate pattern-matching is a very significant factor in this research work. It also highlights the DPI challenges. Network Forensics, its details, challenges and requirements are given which develop the required understanding necessary for the design considerations of the framework. The introduction to the model used for the forensic analysis framework in this research is given.

To accelerate the task of deep packet inspection of the large quantities of network

traffic, Graphical Processing Units(GPUs) are used. The history and evolution of GPUs is given in the chapter. A brief introduction to the GPUs of the most popular GPU manufacturer, NVIDIA is also included. A discussion on the General Purpose GPU Computing research efforts indicates how these efforts provided encouragement to attempt the use of GPGPU for the design of the framework proposed in this research.

Further in the chapter, history, evolution and benefits of Grid Computing, have been provided. The framework proposed in the thesis is based on Grid Infrastructure. The service oriented architecture, its benefits and convergence with the grid provides insights into the Web Service Resource Framework(WSRF) which is used in this work.

The chapter concludes by presenting the problem formulation, followed by the thesis objectives and a brief discussion on how these are achieved.

2.2 Intrusion Detection and Intrusion Prevention Techniques

The very first objective of the thesis is to analyze intrusion detection and prevention techniques. The literature review of the existing commercial and open-source tools provided the required knowledge of the strengths and limitations of the available methods.

The following table briefly presents the information pertaining both kinds of tools for Host-based as well as Network-based IDS/IPS (see Table 2.1).

Intrusion detection and Intrusion Prevention techniques are classified into two ma-

Table 2.1: Comparative Analysis of Intrusion Detection and Intrusion Prevention Tools

Description	**Tools**	
	Commercial	**Open-Source**
Host-based	Verisys [Sys15] - File Integrity Monitoring - Windows and Linux Platform - Based on Cryptographic Hashing - Client-server Components: Verisys Console and Verisys Agents -Communication between Console and Agents secured through (Transport Layer Security)TLS	OSSEC[Oss15] - Free - Open-Source -Cross-Platform - Provides IDS for Linux, OpenBSD, FreeBSD, Mac OS X, Solaris, Windows - Performs log analysis, integrity checking, rootkit detection
	Deep Security [Tre15] -from TrendMicro, - Integrity monitoring encryption log inspection, application scanning, and,SSL certificates	Tripwire [Sou13] -Open Source Security & Data-Integrity Tool -Monitors and alerts on Specific file changes on a range of systems -Cryptographic hashes used to detect changes to file system objects
Network based	Sourcefire [Sou15] Next-Generation IPS -Sourcefire -A distributed appliance-based offering modeled on the Snort detection engine.	Snort [Sno15] - From Sourcefire (Martin Roesch) - Cross Platform NIDS and NIPS -Anomaly Based & Signature Matching -Large ruleset - Sniffing, Packet Logging,Protocol Analysis, Content-Matching
	CISCO Secure IDS [CS01] -from CISCO Triggers intrusion alarms by using a signature database - Two Interfaces: Monitoring and Command & control	Bro [Bro15] -Linux, FreeBSD, Mac OS X - Performs Traffic Baselining, Network Measurements, Forensic Investigation - Two-Layered Design: 1.Bro Event Engine(C++ code analyzes traffic, Dynamic protocol detection 2.Bro Policy Scripts- Analyze events to create action policies
	Nitroguard IPS [Nit11] - actively detects and protects from malware and zero-day attacks	Suricata [Sur15] -Open Information Security Foundation - Cross-Platform - Protocol Detection, IP-Matching, File Matching (MD5 Checksums) - Highly Scalable (Multithreaded)
	Netprowler [Sym03] -from Symantec	Fail2ban IPS [Fai15] -Protects from brute-force attacks monitors log files(auth.log, access log) - Python code for UNIX Platform

jor categories: Anomaly Detection and Signature Recognition. Different methods are employed for both these techniques. Research efforts of various researchers in studying,analysing and proposing intrusion detection systems by employing the different techniques have been studied and are discussed in the following paragraphs.

Statistical Modeling and machine learning are two of the earliest and established techniques for anomaly detection. Bayesian Networks, Neural Networks, and Support Vector Machines are some machine learning methods which are widely researched for intrusion detection. Ekta Gandotra, Divya Bansal and Sanjeev Sofat discussed machine learning techniques for the intrusion detection of malware in their contribution [GBS14]. Hua Tang and Zhuolin Cao [TC09], Changjun et al. [HLYH11], Laheeb Mohammad Ibrahim [Ibr10] and Mukhopadhyay et al. [MCCC11] have conducted good research in the field of neural networks based intrusion detection. The neural network can quickly gain the experience in misuse detection initially by training the system to correctly identify preselected examples of the data. The response is reviewed and the system configuration is refined until a satisfactory level of neural network's analysis of the training data is reached. In addition to the initial training period, the neural network also gains experience over time while it conducts analyses on data related to the problem [Can98]. The review of the literature has led us to the conclusion that the strength of neural networks based IDS/IPS is their flexibility. This flexibility is attributed to the capability of a neural network to analyze the data from the network, even if the data is incomplete or distorted. Another advantage is the high speed of detecting intrusions. Neural networks based intrusion detection suffers from a few limitations too. It has been observed that there is a lot of variation in the taxonomy

which might affect the wide acceptance of neural networks for IDS/IPS. Also the correctness, quality and quantity of training data is very critical for its performance. Another issue identified is the 'Black Box Problem' [Fu92]. The rules for analysis are adapted to a training, and once success is achieved, this information is locked inside the neural network.

R.K. Challa and N. Srivastav presented an approach to IDS by integrating the neural network technique with the layered framework technique and demonstrated that the proposed approach had a higher accuracy in the detection rate and lower false positive rate as compared to the other approaches based on only the neural network or the only the layered framework technique [SC13].

Genetic Algorithms find great applications in search problems. An implementation of an intrusion detection system using genetic algorithms has been attempted by [HMB+12]. El-Sayed and N. Feras discussed an intrusion detection technique based on multicriterion fuzzy classifier with genetic attribute selection [EAAO15].

The following table presents the comparative analysis of the intrusion detection and intrusion prevention techniques proposed by researchers. The contributions of each of the proposals is given which helps to achieve a better understanding of the several methodologies, their design and implementation issues and their constraints (see Table 2.2).

2.3 Deep Packet Inspection

DPI engines parse the entire IP packet and make forwarding decisions by means of a rule-based logic. They compare the data within a packet payload to a database of predefined attack signatures. The essence of a signature based scanner is a

Table 2.2: Comparative Analysis of Intrusion Detection and Intrusion Prevention Techniques

IDS/IPS	Author	Methodology	Results/Contribution
Semantic Based Intrusion Detection [Mao10]	Yuxin Mao	Security ontology to represent the formal semantics for intrusion detection	Intrusion Detection on Wireless Sensor Network
Neural Network Approach [TC09]	Hua Tang and Zhuolin	Artificial neural networks & support vector machine applied to KDD CUP'99 dataset	Detection Rate: SVM better than NN Accuracy Comparison: NN better than SVM
Back Propagation Neural Network [HLYH11]	Changjun Han Yi Lv Dan Yang Yu Hao	Training Neural Network on KDD99 dataset	Detection rate:80.5% False alarm rate:7.4% Omission rate:11.3%
Back Propagation Neural Network Approach [MCCC11]	I. Mukhopadhyay M. Chakraborty S Chakrabarti T Chatterjee	Back Propagation Neural Network Classifier on KDD99 dataset	Training Success Rate:95.6% Failure Rate:4.4% Validation Success Rate:95.6% Failure Rate:4.4%
Distributed Time-Delay Artificial Neural Network [Ibr10]	Mohammad Ibrahim Laheeb	DTDNN,Training dataset: 5 classes(5000 instances) Test dataset: 500 instances	Detection Rate DoS:97.6%, Probe:96.2% U2R:95.8%, R2L98% Normal:98.4%
Genetic Algorithms [HMB+12]	Mohammad Sazzadul Hoque Md. Abdul Mukit Md. Abu Naser Md. Bikas	Implementation Of Intrusion Detection System Using Genetic Algorithm	Detection Rate:95% FPR: 3%

multi-pattern matching algorithm. Based on a certain set of string patterns, such as worm code signatures [NKS05], the scanner checks each byte of the payload to find out whether it contains one of these patterns. Signature based scanning can be used to check payloads, maintain flow states and scan flow streams to find specified signatures [CL05]. This takes more time than application protocol analysis [DFM+06] and usually becomes the bottleneck of the system. Analysis of packet headers can be done economically as the location of header fields is restricted by protocol standards. But, payload contents are unconstrained. Searching through the payload for multiple string patterns within the datastream is a computationally expensive task. The engine that drives deep packet inspection typically includes a combination of signature matching technology along with heuristic analysis of the data in order to determine the impact of that communication stream. The inspection engine must use a combination of signature-based analysis techniques as well as statistical, or anomaly analysis techniques.The requirement that it be performed at wirespeed adds to the cost.

Thus, the design of high performance signature based scanners is very important in high-speed network security applications [CCU07]. Figure 2.1 summarizes the various hardware and software implementations available for Deep Packet Inspection.

Based on these algorithms, there are two very popular tools which are widely used.

- **Snort**, a pattern matching IDS tool, was designed to be a lightweight, easy-to-use and cheap system for intrusion detection for small to medium size networks, as an alternative to commercial systems [Sno15].

- **Bro**[Bro15] features a sophisticated policy based approach and also allows

for signature based matching.

2.3.1 Hardware Platforms for DPI

A DPI system performs a set of time-critical operations to sense certain network patterns or behavior while trying to minimize packet processing latency. In order to identify traffic at the speeds necessary to provide sufficient performance, different platforms have been used to implement DPI including Application Specific Integrated Circuits(ASICs) [TS06], Field Programmable Group Arrays(FPGAs) [DL06] and network processor units(NPUs) [PY06] which provide for fast discrimination of content within packets while also allowing for data classification. The first step is to capture packets from network interface cards, reassemble and buffer them for processing. The second step is to search known signature patterns in the payload of packets. Following that, a DPI system will analyze the matched packets against semantic-rich policies to determine if an attack is present or an alert should be triggered. Traditionally, ASICs are used for the first task. The other task of policy processing is better carried out with general purpose processors because of the complex semantic processing. The matching of signature patterns falls in the middle of the spectrum, and can benefit from programmable hardware acceleration based on FPGAs or network processors. Most of these methods rely on state machines or finite automata to match patterns. However, one of the key issues is that the size of the finite automata is so large (often at least tens of megabytes) that they have to be stored in off-chip memory modules. As a result, the searching on the automata incurs a large number of off-chip accesses. [CD07] have proposed a hardware based DPI engine which uses techniques like modified content address-

able memory (mCAM), interleaved memory banks, and data packing, to reduce the storage requirements for the finite automata. ASIC-based systems take advantage of off-the-shelf silicon for fast-path packet processing. With most such hardware implementations, the pattern-matching engine provides acceleration for a limited number of expressions in parallel; once this number is exceeded performance drops to un-accelerated levels because the higher-level controls need to swap in and out pattern-match sets. When this occurs it is usually no longer possible for the matching engine to run in the "fast path" so not only does matching performance suffer, packet processing performance can be degraded as well. ASIC-based systems generally are designed without a hard-disk-backed virtual memory architecture, and consequently are unable to do long-term storage of information relevant to detection. B. Bloom proposed the Bloom Filter algorithm [Blo70], which is widely implemented in hardware, like FPGA which contains multiple hash function blocks and paralleled memory accesses. Due to their customizable architecture, FPGAs are ideal platforms for performing large numbers of string comparisons rapidly. They can be configured to accept a regular language consisting of all suspicious strings and to nearly instantaneously perform this for a given string being inspected. Recent efforts have yielded architectures capable of processing packets at multi-gigabyte speeds [SL04] [DKSL03] [AL05] The main oversight with this trend in research is that it ignores the lower end of the user spectrum. Previous hardware approaches attempt to solve problems where the speed benefits of the FPGA are necessitated by the volume of traffic. This occurs on large networks common in corporate offices or other similar enterprises. Individual clients operating stand-alone machines on the other hand often do not have the necessary CPU resources for a formidable software solution or the consumer

pricing threshold to accommodate a hardware solution, like an additional FPGA [SSMF03] [Le02].

2.3.2 Algorithms for DPI Implementation

There are two types of algorithms which have been proposed and utilized for pattern-matching; single pattern matching algorithms and multiple pattern matching algorithms. The most efficient algorithm for matching a single pattern against an input was proposed by Boyer and Moore [BM77]. The Boyer-Moore algorithm is based on skipping heuristics, therefore when the suffix of the pattern appears infrequently in the Text string, the execution time can be sub-linear. Naive or brute force is the most straightforward algorithm for string matching. It simply attempts to match the pattern in the target at successive positions from left to right by using a window of size m. In case of success in matching an element of the pattern, the next element is tested against the text until a mismatch or a complete match occurs. After each unsuccessful attempt, the window is shifted by exactly one position to the right, and the same procedure is repeated until the end of the text is reached. Aho and Corasick proposed the AC algorithm [AC75], which has proven linear performance, making it suitable for searching a large set of signatures. The AC algorithm and its extensions deal well with regular expression matching, but they are not optimal for fixed pattern matching like worm scanning for its large number of states and frequent I/O operation compared to BM and its extensions. The classical single-pattern matching BM algorithm, however, cannot be transplanted to multi-pattern matching trivially, in which cases, some extensions of the BM algorithm are proposed, such as the WM algorithm [SM94], the

Setwise BMH algorithm [FV02], or the AC_BM algorithm [CSM01]. These popular extensions change some features of the traditional algorithm to adapt to the multi-pattern case. Knuth-Morris-Pratt [KMP77] is similar to the Naive since it uses a window of size m to search for the occurrences of the pattern in the text but after a mismatch occurs, it uses a precomputed array to shift several positions to the right. Multiple-pattern matching scales much better than the single pattern matching. Aho-Corasick(AC), Wu Manber(WM) and AC_BM [CSM01] are the classical multiple-pattern matching algorithms. The AC algorithm has good linear performance, making it suitable for searching a large set of signatures. The AC algorithm and its extensions are ideal for regular expression matching, but they are not optimal for fixed pattern matching like worm scanning because of its large number of states and frequent I/O operations. WM algorithm is a very efficient multi-pattern matching algorithm, which implements multi-pattern matching using bad character block transfer mechanism and search the pattern with the hash function. AC-BM combines the AC and BM algorithms. Instead of using the suffix of patterns as in AC, it uses the prefix. It uses the BM technique of bad character shift and the good prefix shift. The multiple-pattern matching algorithm, studied and experimented with in this research, is the Rabin-Karp algorithm[KR87]. The algorithm is a good candidate for this work because of its rolling hash feature. The pattern-matching algorithm, Rabin Karp has been coded in CUDA for the GPU. A brief description of the classical pattern-matching algorithms is given in the following subsection.

2.3.2.1 Pattern-matching algorithms

The following are the classical algorithms used for the pattern-matching in Deep Packet Inspection.

- Naive Brute Force

  The complexity of the Naive Brute Force is O(mn). In this, First character of pattern P(having a length m) is aligned with first character of text T (having a length n). Thereafter, the scanning is done from left to right. The efficiency is affected as shifting is performed at each step.

- Aho-Corasick(AC)

  The AC algorithm is a dictionary-matching string-search algorithm. Invented by Alfred V. Aho and Margaret J. Corasick in 1975, this algorithm constructs a finite state machine resembling a trie with additional links between the various internal nodes. It avoids backtracking as the automaton can transition between pattern matches. The complexity of the algorithm is O(m + k), where k is the total number of occurrences of the pattern strings in the text.

- Boyer-Moore(BM)

  Boyer Moore, improves the performance of pattern-searching into the text by performing larger shift-increment whenever mismatch is detected. Its manner of scanning is different from other string matching algorithms. It uses the Good-Suffix and the Bad-Character shifts. It scans the string from right to left; i.e. P is aligned with T such that the last character of P will be matched to the first character of T. If the character is matched then the pointer is shifted to the left to match the rest of the characters of the pattern.

If a mismatch is detected at say character *c*, in T which is not in P, then P is shifted right to *m* positions and P is aligned to the next character after *c*. If c is part of P, then P is shifted right so that c is aligned with the right most occurrence of c in P. The worst complexity is still *O(m + n)*.

- Bloom Filter

  A Bloom filter is a probabilistic data structure, that is used to test whether an element is a member of a set. A Bloom filter has a 100% recall rate because False positive matches are possible, but false negatives are not. The limitation of a simple bloom filter is that as more elements are added to the set, the probability of false positives also increases, but with advanced techniques this problem has been addressed. Despite the risk of false positives the bloom-filter is a very space-efficient algorithm. A Bloom filter with 1% error and an optimal value of *k*, requires only about 9.6 bits per element, regardless of the size of the elements.

- Knuth Morris Pratt(KMP)

  This algorithm is based on automaton theory. Firstly a finite state automata model M is created for the given pattern P. The input string T is processed through the model. If pattern is present in text, the text is accepted otherwise rejected. One limitation of the KMP algorithm is that it does not report the number of occurrences of the pattern.

2.3.3 Combination of Hardware and Software Implementations

Since conventional software-based algorithms for string matching have not kept pace with high network speeds, Dharmapurikar et al. have proposed the use of specialized high-speed, hardware-based parallel bloom filters [DKSL03]. A Bloom filter is a data structure that stores a set of signatures compactly by computing multiple hash functions on each member of the set. With this technique, a database of strings is queried for the membership of a particular string. The answer to this query can be false positive but never a false negative. The data structure has a unique characteristic that the computing time spent in performing the query does not depend on the number of strings in the database. The grouping of signatures is done according to their length and stored in parallel Bloom filters. Each of these Bloom filters scans the streaming data for the presence of strings of its corresponding length. Whenever a Bloom filter detects a malicious string, an analyzer is probed with this string to check if it actually belongs to the given set of strings. Taskin and Kaya have proposed an energy efficient low power bloom filter architecture for DPI which exploits the pipelining technique [KK06].

To provide scalable deep packet inspection, Huang et al. have proposed the pipeline architecture with multiple Ternary Content Addressable Memory (TCAMs) to obtain wire-speed classification for IPv6 and multimedia applications. The packet classification engine is capable of handling multiple QoS requirements at ultra-high speed [HCC05].

The contribution of Cho and Smith is a novel architecture of programmable pattern matching co-processor. Based on the architecture, a scalable co-processor best

suited for FPGA is implemented. This processor is then combined with an efficient reconfigurable pattern matcher to form a hybrid system. When new patterns are added to the set, they can be immediately updated and detected by the co-processor. This update process gives the compiler enough time to generate a new filter that meets specified area and performance constraints [CS05].

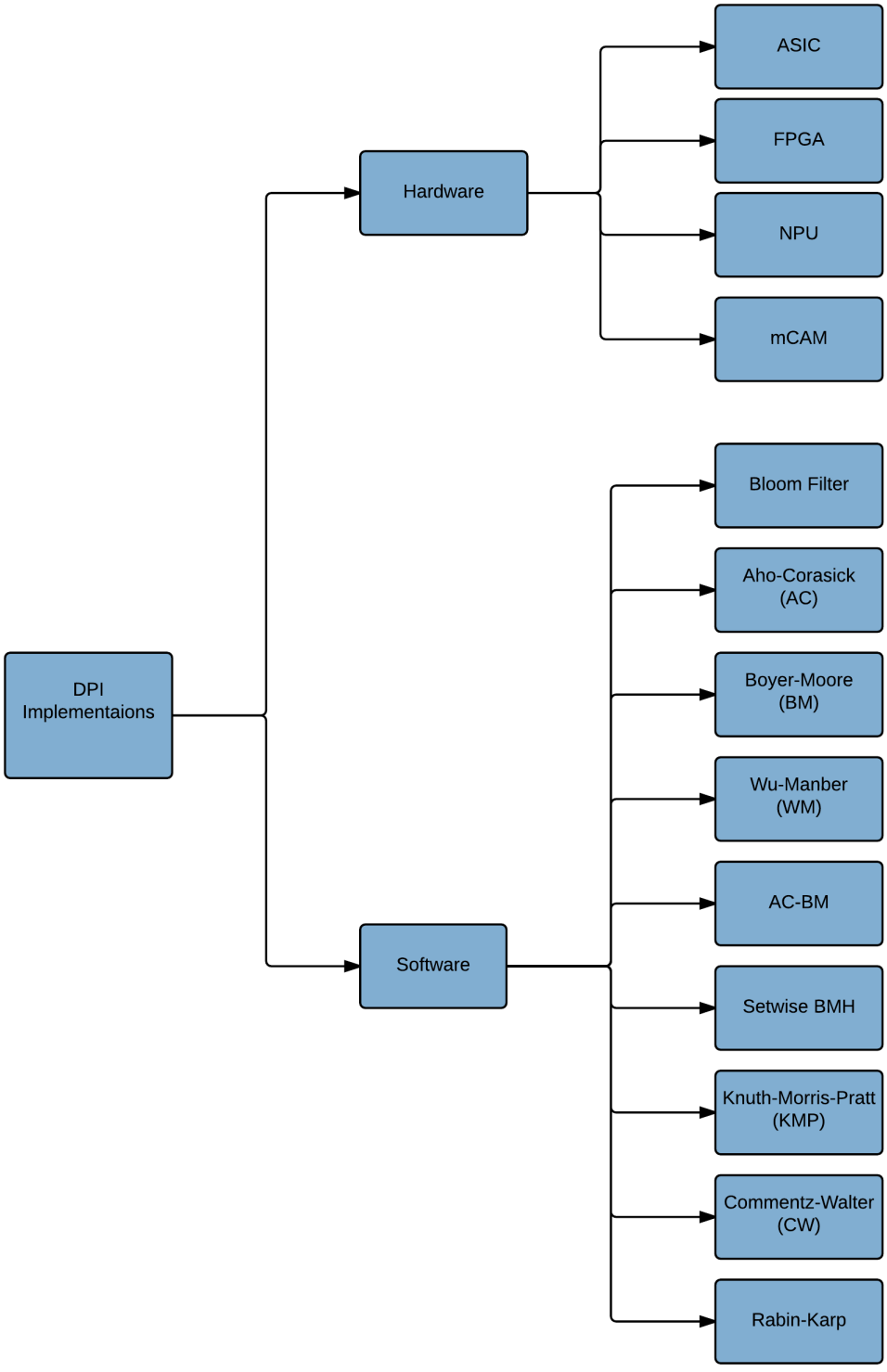

Figure 2.1: DPI Implementations

2.3.4 DPI Challenges

An understanding of the challenges to Deep Packet Inspection is of high significance because it contributes to the determination of the performance criteria for the forensic analysis framework. The framework proposed in the thesis is expected to meet the requirements set forth by these challenges. AbuHmed et al. [TMN07] summarized the challenges for a DPI as follows:

1. Complexity of the Pattern-Matching Algorithm: This is of great significance since an NIDS spends 75% of time in the pattern-matching process.

2. Increasing variety and volume of intrusion signatures : The constant expansion in the volume, variety and velocity of the attacks makes the pattern-matching process overwhelming.

3. Need for preprocessing the packets before the pattern-matching process because of overlapping of signatures.

4. Unknown location of a signature: The attack payload can be encoded in a way to evade the IDS, so there is a need to inspect the complete payload against the attack pattern.

5. Encrypted Data : Inspection of ciphered data can not be done by the DPI. The way-around is to plug the DPI component behind the decryption device.

These challenges contribute to the recognition of the design objectives, to design a framework which performs well under load. Under contrived theoretical conditions, a framework might give a good performance, but the challenge is that the system performs well under real heavy traffic situations too.

2.4 Network forensic analysis

Marcus Ranum is credited with defining network forensics as "*capture, recording, and the analysis of network events in order to discover the source of security attacks or other problem incidents*" [Ran14]. Network forensics monitors the network traffic to determine the anomalies in the traffic that might indicate a threat. When an attack is detected, the nature of the attack is also determined. Attack patterns can be created from the acquired knowledge. The techniques enable investigators to track the attackers and provide sufficient evidence to allow the perpetrator to be prosecuted [YM10]. The attribution of an attack to a host or a network helps in predicting and preventing future attacks.

2.4.1 Network traffic analysis challenges and requirements

Carrier identifies the following two challenges of digital forensics, the complexity of the process and the quantity of data being analyzed [Car03]:

- Complexity Problem: The acquired data are typically in a raw format, difficult for humans to comprehend.
- Quantity Problem: The enormous amount of data to be analyzed.

Both problems are exacerbated in the case of network forensics. Wang and Daniels summarized the major technical challenges for network forensics [WD05]:

- The complexity problem as identified by Carrier [Car03] is applicable here because of the continuous increase in the variety and velocity of attacks. The highly sophisticated attacks span over span over a large number of hosts, resulting in the complexity of the analysis.

- The quantity problem as identified by Carrier [Car03], is magnified for the forensic analysts, who are overwhelmed by the large amounts of low-level event logs. IDS alerts are just a basic element of the intrusion indication, not adapted reasonably for the forensic investigations. The redundancy of signatures and unrelated information while obscuring parts of useful information, is a problem.

With respect to these challenges and the difficulty of the human analysis of the huge volume of irrelevant information and the increasingly sophisticated attack strategies, they argued that Short response times and a Friendly interface, are important requirements for a network forensics analysis system [WD05]. Their model attempted to reduce the response time and perform the analysis in a timely manner, and presented the analysis results in an intuitive manner.

2.4.2 Network Forensic Analysis Techniques

Tools and techniques for network forensics are reviewed in this section . Pilli et al. surveyed network forensics' frameworks and reported the classification of network forensics systems based on their characteristics of purpose, collection and nature, in the following manner.

1. Purpose:
    a) General Network Forensics for enhancing the security of a network
    b) Strict Network Forensics for collecting evidence for legal requirements
2. Collection of Traffic:

a) Catch-it-as-you-can approach, where all packets passing through a certain traffic point are captured and written to storage with analysis being done in batch mode, posing the requirement of large amounts of storage.

b) Stop-look-and-listen approach, where every packet is analyzed in a rudimentary manner in the memory and only certain information is saved for future analysis, posing the requirement of a faster processor to keep up with volume and velocity of the incoming traffic[Gar14].

3. Nature: The network forensic system as,

a) An appliance with hardware and pre-installed software or

b) An exclusive software tool.

In this research, the purpose of designing the forensic analysis framework, is for the enhancement of the security perimeter. Thus, the proposed framework falls under the category of General forensic analysis. The collection of network data is done with the catch-it-as-you-can approach. With this purpose and traffic collection method, the framework is designed as a software tool exclusively. In this thesis, the review of the forensic analysis tools is conducted based on the 'nature' criterion. Accordingly, two types of Network Forensics tools are studied, Software-based and Appliance based. The appliance based tools deliver good performance but are quite expensive. RSAEnvision, NetIntercept, NetDetector from Niksun and InfiniStream are some popular network forensics appliances. The software based tools that are available from commercial vendors are more flexible as compared to the appliance-based but they also do cost quite a bit. The commercial software tools available for network forensics are: Sleuthkit, NetWitness, DragonIDS and

Omnipeek. Omnipeek from Savvius has the ability to work with multi-gigabit networks and also has functionality for VOIP. Open source forensic tools are also available. One good tool , which has been tested is Xplico. This free and open Source Network Forensic Analysis Tool can perform content reconstruction from packet captured by a packet sniffer(tcpdump/Wireshark). It also has the feature to decode VOIP calls.

Soft Computing based network forensics techniques have been proposed by Liu and Feng (2005), Kim et al.(2004), Liao et al.(2009), Anaya et al. (2009) and Zhang et al. They have proposed techniques for reduced Storage by correlating features and attacks which also have an increased classification accuracy.

Yassinac and Manzano (,2002) [YM10]; Merkle (2008) [Mer08] and Thonnard and Dacier (2008) have discussed honeypot based forensic analysis techniques. Nikkel,2006 discussed a Portable Network Forensic Evidence Collector. Krasser et al. in 2005 proposed a Network Visualization framework with animated, time-sequence scatter plots and parallel coordinate plots in both 2D and 3D to rapidly analyze network traffic [KCG+05]. Other techniques include Self Organizing Maps; Attack Graph-based; Haptic Feedback and Spinning Cube of Potential Doom.

For the problem of handling huge amount of forensic data, several researchers have proposed frameworks based on distributed computing. Daniels and Tang(2005) have proposed a simple framework for distributed forensics. The attack attribution graph mechanism is a contribution of their work. Ren and Jin proposed a real time distributed forensics which considered integration of logs with the network data for a deeper analysis [RJ05]. A Framework based on JADE Mobile architecture has been proposed by Nagesh (2006) [Nag06]. Shanmugasundaram, (2003) [SMSB03] proposed ForNet, a distributed network logging mechanism to aid digital forensics

over wide area networks. These distributed frameworks are based on traditional distributed computing technology which does not provide the flexibility and control on sharing relationships needed to establish Virtual Organizations(VOs). A virtual organization comprises on a set of individuals and institutions sharing resources regarding a set of rules. The distributed infrastructure that can efficiently co-ordinates problem-solving and resource-sharing within virtual organizations at different locations having heterogenous architectures. The proposed framework in this research is based on the Grid Computing Infrastructure which can execute data mining on distributed datasets through a WSRF compliant web service.

Despite the presence and use of the various software tools, current practices in network forensics analysis are to manually examine logs, a time-consuming and error prone process [IST04] . A technical report from the First Digital Forensic Research Workshop points to the desirable characteristics of the technologies being used to be reliable, precise, accurate, non-reputable, secure, flexible, and inexpensive and suggests the researchers to carefully consider these requirements [Pal01].

The framework presented in the thesis attempts to address the 'complexity' and 'quantity' issues by making efficient use of the GPU technology and the web-service oriented Grid infrastructure. The idea is to perform the forensic analysis as fast as possible so that alerts can be raised early on and timely counter-measures can be taken to prevent and control such attacks in the future.

2.4.3 DPI based forensic analysis of network traffic

Many tools perform real-time monitoring of network traffic, but this requires appreciable human and hardware resources. The computational and resource intensive

real-time deep packet inspection of all network traffic, would not scale to large networks. The practical solution is to perform reconstructive traffic analysis by capturing and archiving all traffic and analyzing subsets as necessary. Also known as network forensics, the method can identify the source and nature of the security attacks.

The Deep Packet Capturing(DPC) of complete data packets, crossing a network can be done by using tools such as TcpDump, Windump, Ettercap, dSniff or Wireshark (earlier known as Ethereal). Once captured and stored, either in short-term memory or long-term storage, Deep Packet Inspection is then performed for forensic analysis to uncover the root cause of network problems, identify the security threats and ensure data communications and network usage complies with outlined policy. Once an intrusion has been detected, historical data may allow a system administrator to determine, conclusively, exactly how many systems were affected [Mus11]. Once a particular attack or signature has been identified, every packet comprising that event is available, both in raw packet form or accurately rendered to their original format [Ven03].

The availability of packet-by-packet records of the network traffic related to an attack would allow for its in-depth scrutiny and exact documentation of an incident for purposes of forensics. The NIDS may perform analysis of past events which it might not have considered interesting until more recently seen traffic hinted at their relevance.

2.4.4 Behaviour based Intrusion Detection System/ Forensic Analysis

Behaviour based Intrusion Detection Systems, also known as Statistical Intrusion Detection Systems have been an interesting research topic because of the limitation of the Signature-based NIDS in detecting zero-day attacks. An NIDS based on advanced statistical modeling techniques can detect anomalies and classify new network traffic data. A very good work in this direction was carried out by Dorothy E. Denning in as early as 1987, where she proposed a real-time intrusion detection model based on the behaviour of subjects with respect to the objects to detect security violations[Den87]. Mahdi and Bensebti proposed an IDS model based on the Bayesian classifier by fitting the mixture model parameters to the expectation-maximisation (EM) algorithm[CS06] and also discussed adaptations of their detection algorithm for the case of Gaussian mixture models resulting in the linear complexity for the algorithm [MB09].

The research by [Pea10][FRP][GBB] also reported good results with Statistical Modeling for building Intrusion Detection models.

A dataset is the input source to the forensic analysis component. Datasets can be partitioned into smaller subsets for specific analyses and/or data mining. Data mining techniques can be used for datastream or email content mining, by utilizing artificial intelligence approaches to identify special features [MS03]. IP trace back approaches [BC00] help in the attack origin identification. Thereafter, content reconstruction can be performed. Statistical modeling techniques can be applied to the forensic analysis. Training of the classifier is done with the help of a labelled dataset. The classifier learns and then it can be used for the prediction of class

labels for future data instances from an unlabelled dataset also. This is a very useful feature for the forensic analysis as quick predictions can help thwart attacks sooner than other techniques.

2.5 Graphical Processing Unit(GPU)

Traditionally, GPUs have been designed to perform very specific type of calculations on textures and primitive geometric objects, and it was very difficult to perform non-graphical operations on them. One of the first successful implementation of parallel processing on GPU using non-graphical data was registered in 2003 after the appearance of shaders [iL13]. Latest generations of graphics cards incorporate new generations of GPUs, which are designed to perform both graphical and non-graphical operations. One of the pioneers in this area is NVIDIA [NVI13d] which rules the market with its parallel computing platform and API model, CUDA (Compute Unified Device Architecture) [NVI13a], gradually taking over its major competitor, AMD. Others also have their own programmable interfaces for their technologies: the vendor neutral open-source, OpenCL which gives a major competition to CUDA, AMD ATI Stream, HMPP, RapidMind and PGI Accelerator [GPKB12].

2.5.1 GPU Architecture

GPUs use numerous programmable computational cores and fine-grained threads. Moreover, to achieve scalability, GPUs minimize use of global structures. For example, unlike CPUs, the streaming-multiprocessors (SMs) in GPUs have simple in-order pipelines. GPU chips spend more die-space on ALUs and less on caches, thus

instead of seeking performance via out-of-order processing over large instruction windows and large caches, they incorporate low-overhead thread scheduling and hide memory latencies via multithreading. Further, to exploit data access locality, GPUs feature large register files, shared memory and relatively small caches. For example, Intel's Itanium 9560 CPU uses 32MB last level cache (LLCs). In contrast, the GT200 architecture GPUs did not feature an L2 cache, the Fermi GPU has 768KB LLC and the Kepler GPU has 1536KB LLC. Multi-level hardware-managed caches are relatively recent addition to GPUs which also marks a paradigm shift in GPU architecture towards mainstream computing. Effective management of caches is vital to fully exploit their potential in boosting GPU performance and energy efficiency.

2.5.2 NVIDIA GPUs

NVIDIA's GPU offered the opportunity to utilize the GPU for GPGPU(General Purpose computing on Graphics Processing Unit). In 2001, NVIDIA GeForce 3 exposed the application developer to the internal instruction set of the floating point vertex engine(VS & T/L stage). Later GPUs extended general programmability and floating-point capability to the pixel shader stages and exploited data independence. With the introduction of the GeForce 8 series, GPUs became a more generalized computing device. The unified shader architecture introduced in the series was advanced further in the Tesla microarchitecture backed GeForce 200 series which offered double precision support for use in GPGPU applications. The successor GPU microarchitectures viz. Fermi[NVI13f] for the GeForce 400 and GeForce 500 series, then Kepler [NVI13b] GeForce 600 and GeForce 700 and

subsequently Maxwell [NVI13c] GeForce 800 and GeForce 900 series, have dramatically improved performance as well as energy efficiency over the previous ones. Very recently NVIDIA announced major architectural improvements such as unified memory so that the CPU and GPU can both access both main system memory and memory on the graphics card, in its yet to be released Pascal architecture[NVI13e]. The new GPU generation also boasts of a feature called NVLink which would allow data between the CPU and GPU to flow at 80 GB per second, compared to the 16GB per second, available presently.

2.5.3 GPGPU

GPU computing or GPGPU or the General Purpose Computing on a GPU, is the use of a GPU (graphics processing unit) to do general purpose scientific and engineering computing. The model for GPU computing is to use a CPU and GPU together in a heterogeneous co-processing computing model. The sequential part of the application runs on the CPU and the computationally-intensive part is accelerated by the GPU. From the user's point of view, the application is faster because it is using the better performance of the GPU to improve its own performance. Long back, NVIDIA reported that the GPU is 2.5 times faster (on average) than the Intel 3.2 Ghz i7 CPU 14 times faster under certain circumstances. Now, with highly advanced GPU architectures, the possibility of greater improvements are a reality.

A lot of research efforts for the efficient use of GPU for improving its performance for the pattern-matching, with a specific focus on decreasing the high processing time, have been studied in this work. Smith et al. examined the viability of

SIMD-based architectures for signature-matching and presented a detailed architectural analysis [RGO+09]. Several researchers have experimented with various algorithms and techniques to leverage the abilities of the GPU in the field of intrusion detection [Hea08] [Lea08]. Cascarano et al. addressed the state space issues of the traditional DFA based approaches and presented a NFA(non-deterministic automata) based regular expression engine for large and complex rule sets [Nea10]. Vasiliadis et al. ported the open source IDS, Snort to a GPU, Gnort which has since been extended and improvised by several researchers to gain higher speed improvements [Gea08]. Jacob et al. also offloaded the packet-processing task of SNORT to the GPU; PixelSnort was programmed with the complex high level shading language Cg [JB06]. Tumeo et al. implemented the Aho-Corasick pattern-matching algorithm on a GPU [TVS10]. Chung et al. proposed a GPGPU based parallel packet classification method to decrease the huge computing time to filter large number of packets[HLL+11]. Efforts in this research work aim to address the performance issues studied in the related endeavours. One of the most common CUDA-optimization strategies is Memory Coalescing [TEL11], which maximizes the global memory bandwidth usage, by reducing the number of bus transactions, threads with adjacent global indexes in a block are forced to request contiguous data(packets) from global memory. A perfectly coalesced pattern greatly improves throughput. GPUS have multilevel set associative caches, which gives rise to the need for careful performance analysis due to the set associativity. Gusev and Ristov have notable contributions in this direction [MR11]. Several researchers including Fatahalian et al., Sim et al. and Ristov et al., to name a few, have highlighted the fact that GPU Caches affect application performance in a significant manner [KSH04] [JDKV12] [MR11]. Mittal S. presented the classification of techniques

for managing and leveraging the GPU cache [Spa13]. An unintended application of GPUs is by hackers for accelerated password cracking. Olufun et al. in their work for developing a security model for WLANS, demonstrated the use of a GPU based encryption breaking tool, pyrit which cracked a dictionary file much faster than a CPU based tool [TCH+14]. Such efforts are a big motivation to continue the GPU based security efforts.

2.5.4 CUDA architecture and programming model for GPGPU

There are a variety of GPU programming models, the popular ones being the open source OpenCL [JGS10], the free BSD licensed BrookGPU [BTFF03], and CUDA [NVI07], the Nvidia's proprietary framework. CUDA(Compute Unified Device Architecture) is supported well by Nvidia hence it has become the most popular of all. It has features like highly optimized data transfers to and from the GPU and efficient management of the GPU shared memory. The parallel throughput architecture of CUDA can be leveraged by the software developers through parallel computing extensions to many popular high level languages, such as, C, C++, and FORTRAN; CUDA accelerated compute libraries and compiler directives. Support for Java, Python, Perl, Haskell, .NET, Ruby and other languages is also available. The CUDA programming model is a C-like language where the CUDA threads execute on a the device(GPU) which operates as a coprocessor to the host(CPU). The GPU and CPU program code exist in the same source file with the GPU kernel code indicated by the *_global_* qualifier in the function declaration. The host program launches the sequence of kernels. A kernel is organized as a hierarchy of threads. Threads are grouped into blocks, and blocks are grouped into a grid.

Each thread has a unique local index in its block, and each block has a unique index in the grid, which can be used by the kernel to compute array subscripts. Threads in a single block will be executed on a single multiprocessor, sharing the software data cache, and can synchronize and share data with threads in the same block; a warp will always be a subset of threads from a single block. Threads in different blocks may be assigned to different multiprocessors concurrently, to the same multiprocessor concurrently (using multithreading), or may be assigned to the same or different multiprocessors at different times, depending on how the blocks are scheduled dynamically. There is a physical limit on the size of a thread block for a GPU determined by its compute capability. In this work, for the compute capability 2.1 GPU, it is 1536 threads or 32 warps.

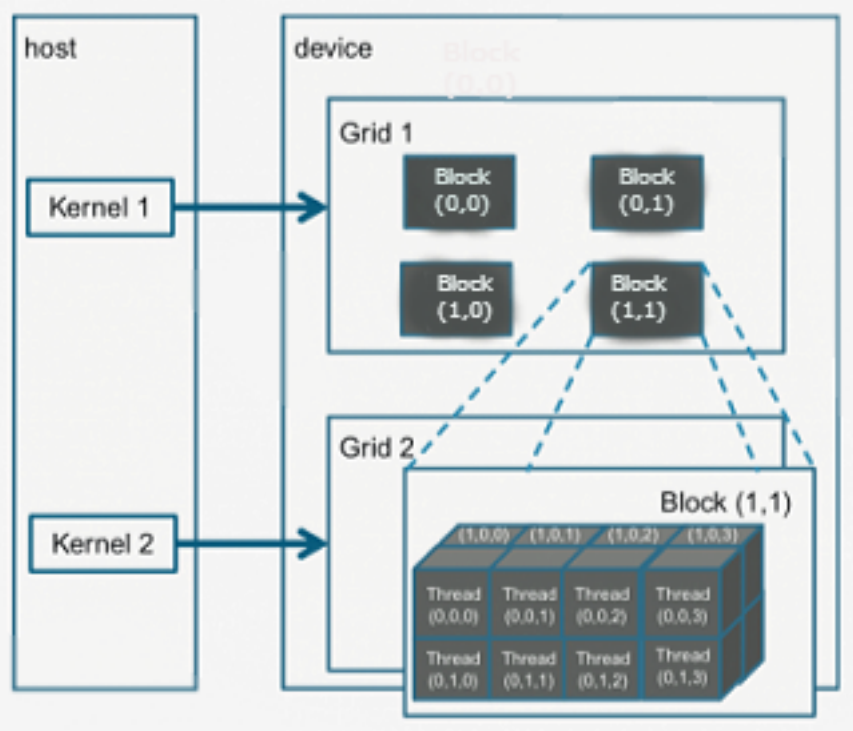

Figure 2.2: CUDA Programming Model

The CUDA programming model assumes that the host and the device maintain their own separate memory spaces. Figure 2.2 shows the CUDA programming model. The CUDA program manages the device memory allocation and de-allocation as well as the data transfer between host and device memory [NVI07]. CUDA devices provide access to several memory architectures, such as global memory, constant memory, texture memory, shared memory and registers, with their certain performance characteristics and limitations. Figure 2.3 illustrates the memory architecture of CUDA device, specifically the Nvidia GT635M.
The sequence of CUDA processing flow is as follows:

- Copy data from host memory to device memory
- CPU instructs the process to GPU
- GPU execute parallel code in each core
- Copy the result from device memory to host memory

The appropriate use of the read-only constant and texture memory can improve performance and reduce memory traffic when reads have certain access patterns.

2.6 Service-Oriented Architecture for the Grid

2.6.1 Web Services

Web Services is a software system that supports interoperable machine-to-machine interaction over a network. Web services interface is described in a machine processable format. Web services provide flexible, extensible, and widely adopted

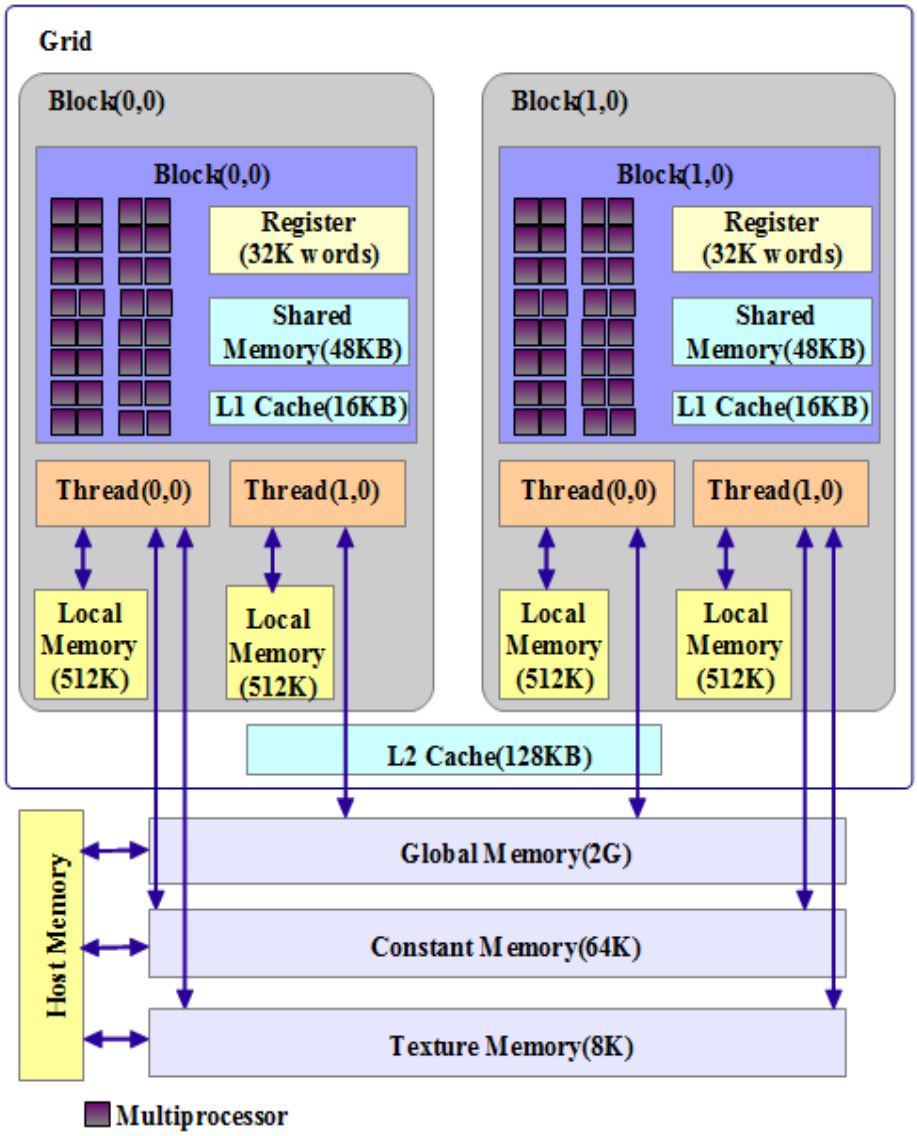

Figure 2.3: Memory Architecture of NVIDIA GT635M

XML-based mechanisms for describing, discovering, and invoking network services; in addition, its document-oriented protocols are well suited to the loosely coupled interactions that many argue are preferable for robust distributed systems [KWWW94]. A Web Service is callable by another program across the Web in a platform/language/object model neutral fashion (using standardized Web Ser-

vices protocols). In short, a Web service is to an application what a Web page is to a person. The Web Services protocols include Simple Object Access Protocol (SOAP); Universal Description, Discovery, and Integration (UDDI); and Web Services Description Language (WSDL). Web services have emerged as the architecture of choice for grid standards such as the Web Services Resource Framework (WSRF) [AIH04]. Web services publish details of their functions and interfaces, but they keep their implementation details private; thus a client and a service that support common communication protocols can interact regardless of the platforms on which they run, or the programming languages in which they are written. This makes Web services particularly applicable to a distributed heterogeneous environment. The key specifications used by Web services are:

- XML (eXtensible Markup Language)- A markup language for formatting and exchanging structured data
- SOAP (originally Simple Object Access Protocol, but technically no longer an acronym), an XML-based protocol for specifying envelope information, contents and processing information for a message
- WSDL (Web Services Description Language)- An XML-based language used to describe the attributes, interfaces and other properties of a Web service. A WSDL document can be read by a potential client to learn about the service. Although a Web service can support any communication protocol, and may offer its clients a choice, the most common is SOAP over either HTTP or HTTPS. This contributes to the appeal of Web services, as HTTP and HTTPS are ubiquitous and typically do not raise issues of firewall traversal in an organization that allows bi-directional HTTP traffic.

2.6.2 OGSA, OGSI, WSRF and GT4

The relationship between OGSA, WSRF and GT4 is shown in figure 2.4. The Open Grid Services Architecture (OGSA) represents an evolution towards a Grid system architecture based on Web Services concepts and technologies. The goal of the OGSA is to standardize all the services of the grid application by specifying a set of standard interfaces for them. The Globus Toolkit(GT) is an open source software toolkit used for building grid-based applications, being developed by the Globus Alliance and many others all over the world. The first prototype Grid service implementation was demonstrated on January 29, 2002, at a Globus Toolkit tutorial held at Argonne National Laboratory. Since then, the Globus Toolkit 3.0 and 3.2 offered an OGSA implementation based on the Open Grid Services Infrastructure (OGSI), a precursor to WSRF [All14]. The Grid Community invested a lot of effort into the specification and implementation of OGSI but soon realized that it was a heavyweight specification with too much definition in one specification; and was too much object-oriented and also stateful. Globus Toolkit 3(GT3) is an implementation of the OGSI, but since it failed to converge with existing web services standards, a new standard to supersede it, was announced in 2004 and fully implemented in the Globus Toolkit version 4(GT4). GT4 is the widely used, mature web services based toolkit for building grid applications [LB05]. It provides a set of OGSA capabilities based on the Web Services Resource Framework(WSRF) which specifies the creation, addressing, inspection, and lifetime management of stateful resources. Grids and web services started far apart in technology and applications, but WSRF completed the convergence, see (Figure 2.5).

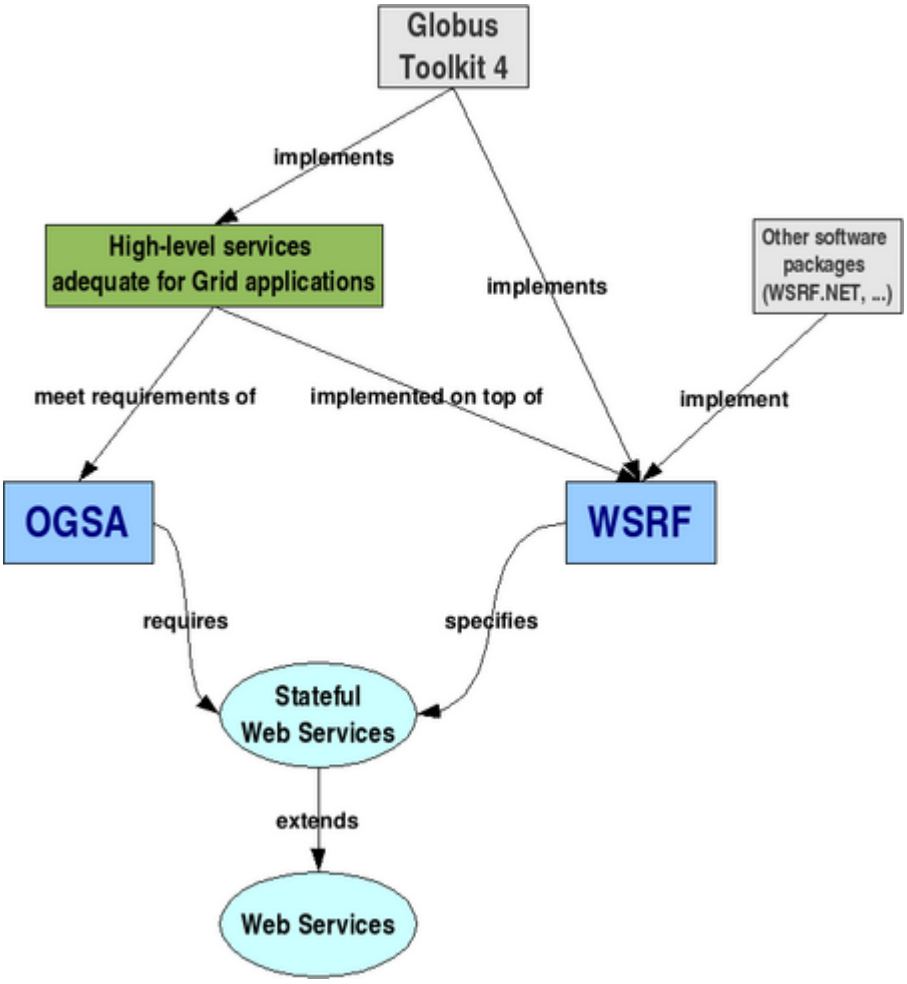

Figure 2.4: OGSA ,WSRF and GT4

2.6.3 WSRF Research Efforts

The service implementation for the Grid presented in the thesis, is the Globus Toolkit Java version of WS-Core. Apart from this, there are at least four more different popular implementations of the SOAP, differentiated on the basis of the programming model and the programming language. The discussion in this section supports the selection of the Globus Java WS-Core from over these implementa-

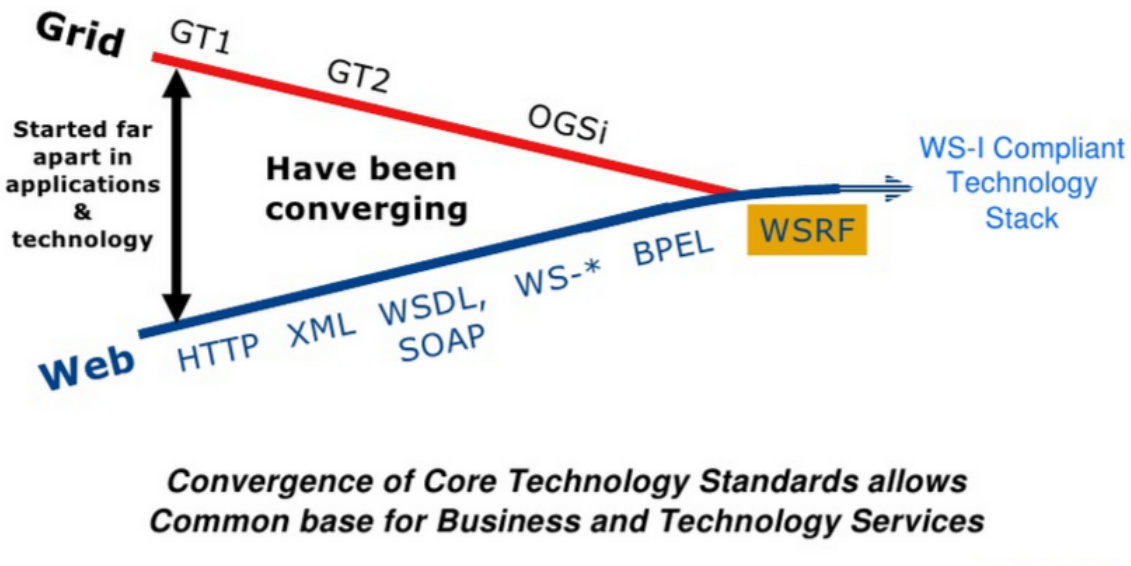

Figure 2.5: Convergence of Grid and Web Services

tions.

An additional implementation of the standards which is part of the Globus Toolkit is implemented in C and lets one develop Grid services in C. pyGridWare [pyG07] is also part of the Globus Toolkit, which allows the user to rapidly develop Grid services in Python. Similarly, WSRF.NET [HWMB04] is used to develop Grid services in any .NET language. With this implementation developing Grid services is not much different than programming Web services, the developer only needs to annotate what parts of the service should be made persistent. Lastly, there is also a Perl implementation of the standards called WSRF::Lite [Lit04]. These five implementations have been compared in terms of functionality and performance by Humphrey et al. [HWMB05].

This comparison concludes that the Globus Toolkit provides the most complete implementation of WSRF and has superior response times to invocation calls when

compared to WSRF.NET, WSRF::Lite, and pyGridWare. Heinis et al. extended the single-client setup study by using a multiple client setup to show how the system performs with a larger number of concurrent requests [HPDA05].

The Globus Toolkit has a robust implementation of WSRF, therefore it eliminates the problem of dealing with an incomplete server-side implementation of WSRF and concentrates work on the client-side issues of communicating with WSRF-based web services from various platforms. In addition, the Globus Toolkit offers a high-performance version written in the C language and a platform-agnostic version written in Java [Hua09].

Humphrey et al. have clearly indicated that the C WS Core of the Globus Toolkit 4, gives the best performance when evaluated for the key primitive functions and the notification function. GT4 Java WS Core and WSRF.Net were reasonably better than the WSRF::Lite and the pyGridWare. pyGridWare, the Python WS Core from the Lawrence Berkeley National Laboratory was faster in certain scenarios only. Perl-based WSRF::Lite from the University of Manchester implements TLS caching as it works on the Microsoft IIS, hence is faster with https [HWMB05]. Even though GT4 C WS Core performs fastest, still, GT4 Java WS Core seems to be a better choice as it has more support and use at both the academic institutions as well as the industry and is faster and easier to develop and implement. If a user wants to create a C client application he must write C implementation of the delegation by means of C WS Core API, because for C, the Delegation API is not available, only C WS Core API and C WS-GRAM API are there, whereas for writing the Java client application, submitting and managing job, Java WS Core API, Delegation API and WS-GRAM API are all available respectively [JDt15].

Kubert and Thai have also investigated the performance of the Globus Toolkit

WSRF and WSRF::Lite implementations. The latency, serialization and de-serialization benchmark results indicated WSRF::Lite performed better than GT4 [KT11].

REST(Representational State Transfer) is emerging as an easier and more flexible alternative to SOAP and WSDL based Web Services [Fie00]. RESTful web services make use of JavaScript Object Notation(JSON) [Cro06] instead of XML for resource representations and exploit HTTP methods explicitly for the resource manipulation instead of defining their own [SRK14]. The adoption of REST by mainstream Web 2.0 service providers, including Google, Yahoo and Facebook and large organizations like Wal-Mart, indicates its growing acceptance, but REST can not replace SOAP everywhere, as it has its own set of issues. A very good comparison is made between REST and SOAP & WS by Cesare et al. [CZL08], which indicates that the choice should be made after careful consideration of key aspects according to the requirements of an application.

2.7 Machine Learning for the forensic analysis

The amount of network traffic captured and available for analysis determines how deep one can dig into historical data to discover threats. The huge quantity of such network data is captured and after the initial preprocessing and/or preliminary inspection and analysis, is stored on the Grid. Once the datasets are stored on the grid, these can be pulled out for the network forensics as and when needed. The forensic analysis of network traffic has three major issues: High Velocity, Large Volume and Wide Variety. To effectively carry out forensic analysis for network traffic with such characteristics, Machine Learning tools are experimented with. A statistical model is built for training a statistical classifier and then use it for

Table 2.3: Types of Machine-Learning

Type	Description
Supervised Learning	The output datasets are provided which are used to train the machine and get the desired outputs e.g.(Decision Tree)
Unsupervised Learning	No datasets are provided, instead the data is clustered into different classes (e.g.clustering)
Reinforcement Learning	An agent interacts with the environment to learn the behaviour that maximizes its cumulative reward over time

predictive modeling.

Machine Learning is the study of algorithms that can learn from and make predictions on data. Smola and Vishwanathan predicted 5 years ago that with ever increasing amounts of data, smart data analysis will become all pervasive [SV10]. Machine Learning algorithms are used to discover patterns in the datasets, and classify or summarize the data. The availability of these datasets on the grid not only offers a solution to the ever growing demand of storage, but also offers the abilities of the grid to parallelize the time consuming and compute intensive tasks of the machine learning algorithm. The implementation of the machine learning algorithms on the Grid, reduces the overall execution time because of resource sharing feature of the grid.

There are three fundamental classes of Machine Learning. The classes with their description are shown in Table 2.3.

Machine learning techniques have applications in several fields including intrusion detection.

Sangeeta Rani and Geeta Sikka presented a survey of clustering techniques [RS12].

McClean et al. discussed model-based clustering approaches for the discretization of data which can lead to improved prediction [MGC+11].

2.8 Problem Formulation

The problem formulation has been done after careful consideration of the research challenges and the research objectives. The research challenges and have been discussed in the previous chapter and the thesis objectives are presented in this chapter in the following section.

The forensic analysis of network traffic has three major issues :High Velocity, Large Volume and Wide Variety. These issues relate to the traffic captured in the raw state and the traffic preprocessed and prepared for the forensic analysis

For providing complete solutions to the network security, the combination of reactive and proactive network security approach for intrusion detection is to be followed. Deep Packet Inspection based Forensics Analysis of network traffic allows indepth and unhurried analysis of anomalies which otherwise might go unnoticed. Currently IDS cannot deal with high speed networks where traffic speed is greater than 1 Gbit/sec. Packets can be missed by the IDS in such situations. The literature review of existing Deep Packet Inspection, Deep Packet Capture and Network Forensics clearly highlights the fact that by every passing day, the number of packets to be captured is increasing at a high rate. From gigabytes, to terabytes, it can go on to petabytes and maybe in the near future in exabytes too. One solution to this problem is adding special hardware but this translates to high costs. Faster capturing of packets is desirable so that all the packets can be captured and none are dropped. Network forensics can generate enormous quan-

tities of information. The forensics analysis too requires to be conducted at high speed. Providing for high speed computational resources translates to high costs. A better cost effective solution to this problem is using the capabilities of a grid to detect enormous intrusion packets and improve the drawbacks of traditional IDSs. Grids can provide high computational abilities alongwith the features of distributed resource usage and communication.

The packet capture rate is very important. For successful forensic analysis, it is important to capture and store traffic from a fully saturated network. Further complicating data integrity is the fact that the forensics analysis tool should never participate in the traffic it monitors. In a typical TCP session, if a machine at one end of a connection misses a packet, it can expect that the information will be resent but a forensics analysis tool cannot request packet retransmission. Performing all of these tasks require huge resources. Reliably capturing the packet from the network requires an interface that doesn't lose packets as the network approaches saturation. Writing captured traffic to disk requires a system bus and storage subsystem that can keep up with network speeds.

Storage size plays the final critical role: the amount of storage dedicated to traffic capture determines how far into the past is the availability of data for forensic analysis. For recording the network traffic of large networks, the storage requirements too are large. For storage or performance concerns, irrelevant traffic can be eliminated by applying a filter during packet capture. But filtering sometimes increases processing overhead to the point that packets are dropped. To be useful, easy access to any particular piece of information should be allowed [cor02]. For forensics analysis the voluminous repositories of network traffic need to be maintained for long durations of time. The traditional solutions to the storage problem

have their limitations in terms of capacity, access and cost. Grids, specifically data grids seem to hold a more better solution to this problem also.

A forensic analysis system based on the DPI intrusion detection system(IDS), is expected to perform the highly complex inspection and analysis of each and every packet crossing the network, and that too at a multi-Gigabit speed. An IDS based on DPI suffers from two types of performance limitations :[JB06]

- CPU-bound limitations: These arise due to the string-matching process;
- I/O-bound limitations. These are caused by the overhead of reading packets from the network interface card.

The efforts of various researchers notably of Ranum and that of Jalote, have measured the string-matching time of the popular IDS,Snort, and reported that string-matching contributes to a significant chunk of the total runtime. The indication is that the pattern-matching of signatures to packets is highly CPU intensive.

The network security solution has to satisfy the specific objectives of memory efficiency, scalability, dynamic updation of intrusion signatures, and deterministic performance independent of traffic characteristics or signature characteristics, to sustain the traffic rate and intrusion signatures growth. A Deep Packet Inspection(DPI) based Forensic Analysis framework, reviews the network packet data and performs network forensics to analyze the attack. The research motivation is to design a high throughput general forensic analysis framework that attempts to satisfy the objectives of the DPI based IDS.

While thinking about the ways to accelerate the task of forensic analysis, the following the concepts of decomposition are considered:

- Task decomposition: Dividing the task of pattern-matching (rabin-karp) algorithm into individual tasks . In this case, the use of the GPU.
- Data decomposition: Dividing the task of forensic analysis of the huge volume and variety of data(network traffic) All the data crossing the network is captured and stored on the data grid for its later analysis(forensics analysis). The network data is preprocessed to do initial and basic analysis, then deep packet inspection is performed to detect attacks. Labelled datasets are created which are also stored on the grid. The large datasets are divided into discrete chunks that can be operated in parallel on the grid. Statistical modeling using Weka is performed for the machine learning of the Weka classifier, which can quickly make predictions and classify new datasets and detect attackers .

2.9 Thesis Objectives

The objectives of the proposed work are:

i. To analyze intrusion detection and intrusion prevention techniques

ii To explore the techniques for forensic analysis of network traffic

iii. To explore the Deep Packet Inspection(DPI) technology

iv. To design a DPI based forensic analysis system for network traffic using grid infrastructure

v. To test and validate the proposed DPI based network forensics system on the grid

2.9.1 Thesis Overview

To achieve the **first objective**, an extensive review of literature is carried out to study the techniques for intrusion detection and prevention and presented in Chapter 2. Both Host-based and Network based techniques are discussed and a comparative analysis of the commercial and open-source tools/software is presented. Furthermore, the efforts of several researchers employing different methodologies for IDS/IPS are also discussed. The results and contributions of the works studied in the review of literature and the study of the features of the available IDS/IPS tools has contributed to develop the technical knowledge needed for the design of the proposed framework. Based upon the literature survey, the gaps and challenges in the present solutions are identified.

Towards the fulfillment of the **second objective**, a study of Digital forensics and Network-forensics has been conducted and presented in chapter 2. It has led to the development of the understanding of the challenges and requirements for the forensic analysis system and provides insights into the process of designing the forensic analysis framework. The literature review of the publications presenting the network forensics framework based on different design and architectural principles has led to the realization of the design and architecture of the proposed framework.

For the accomplishment of **third objective**, detailed literature review has been done to understand the DPI technology, its evolution and challenges(as given in section 2.3. The hardware approaches to implement DPI including Application Specific Integrated Circuits(ASICs), Field Programmable Group Arrays(FPGAs) and network processor units(NPUs), yield architectures capable of processing pack-

ets at multi-gigabyte speeds. The main oversight with this trend in research is that it ignores the lower end of the user spectrum who often do not have the necessary CPU resources for a formidable software or the consumer pricing threshold to accommodate a hardware solution, like an additional FPGA. The literature review of software approaches to the intrusion detection techniques is also done, which helped develop the idea to work on a software solution to make the solution cost-effective and scalable. The various algorithms for the Deep Packet Inspection are studied in order to select the most appropriate algorithm for the work that can yield the best possible performance in view of the DPI challenges as identified in the literature review. The classical algorithms used for signature matching including the Bloom Filter algorithm, the Aho-Corasick(AC) algorithm and the Boyer-Moore Algorithm(BM), but the Rabin-Karp algorithm is chosen for the implementation, considering the final aim of the research and the challenges of the design of the framework.

To achieve the **fourth objective**, a service oriented architecture using Grid Infrastructure, implemented on a GPU using CUDA has been proposed. Related literature as been reviewed to understand the gaps and challenges. The hybrid solution for network security has been designed and implemented to handle the forensic analysis. This hybrid architecture relies on three approaches to handle the high computational demands of the approach. The combination of web services and grid computing in conjunction with GPU makes it an effective, highly efficient, scalable and cost-effective defense mechanism for detecting new attacks, generating their signatures and updating them in IDS. This solution is implemented using various modules which are developed in GT4 WS Core Java. NVIDIA GPU is programmed with the NVIDIA's general purpose parallel computing architecture,

Compute Unified Device Architecture(CUDA), that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU. The proposed CUDA based implementation of the multiple pattern matching, Rabin-Karp algorithm on a multicore GPU outperforms the Intel quadcore processor. The development of the grid application is done using WSRF to maximize the potential of the Grid capabilities.
The **fifth objective** of testing and validation is divided into two phases:

1. The validation of performance of the pattern-search algorithm on the GPU.

2. The validation of the DPI based forensic analysis framework on the grid using the standard statistical metrics.

2.10 Summary

This chapter describes the Deep Packet Inspection and provided insights into the DPI Challenges. The Network Forensics Systems' challenges and requirements are discussed. This helps in gaining understanding of the gaps and challenges and leads to the problem formulation. The search for solutions to gain the desired performance and achieve the objectives, provided the encouragement to explore the GPU, GPGPU and CUDA framework. The benefits of a service-oriented architecture for the grid are also explored and the study of the different web service implementations for the Grid is done. The conclusion is that WSRF wonderfully completes the convergence of Web Services and Grid and hence the motivation to attempt the design of the framework as a DPI based Forensic Analysis System for Network Traffic using Grid Infrastructure.

Chapter 3

Proposed Framework for DPI based Forensic Analysis using Grid Infrastructure

3.1 Introduction

This chapter discusses the design and implementation of the framework. This chapter is organized as follows: Section 3.2 discusses the development tools and methodology. In Section 3.3 the architecture of the framework is presented. Section 3.4 presents a summary of section 3.3, the design and architecture of the framework.

3.2 Development Tools and Methodology

- Grid environment setup using Globus 4.0

  The Globus Toolkit 4.0 is the grid environment over which the implementation of the framework is accomplished. The framework uses the grid setup to perform the task of processing the huge volume of captured network traffic which is otherwise impossible to handle.

- tcpdump, the command line packet analyzer.

- Wireshark, the free and open-source packet analyzer. It is used as an extension to tcpdump as it provided the necessary sorting and filtering options. The network interface controller(NIC) was put in promiscuous mode, to make visible all the traffic(addressed to the interface or nor) crossing the interface. Along with that, there is a provision to add a network tap, if need be, in particular cases, like where high VOIP traffic is observed. Port mirroring or various network taps extend capture to any point on the network. The main features which have immensely helped in the development, include

  - Capturing data "from the wire" from a live network connection or read from a file of already-captured packets(.pcap file)
  - Reading of Live data from a number of types of network, including Ethernet, IEEE 802.11, PPP, and loopback.
  - Perusing the Captured network data via the GUI, or via Tshark, the command line interface.
  - Captured files can be converted to a different file formats.

  - Filtering of the data display.
  - Detection of VOIP traffic

- Libpcap, the packet capture library, on which wireshark is based.
- Winpcap, the port of libpcap used by windows
- openDPI, the open source DPI analyzer
- Snort AD 4.0, anomaly detector. Snort rules are used alongwith the custom-built rulesets(.rules) system to deal with various network attacks.
- Oracle Java SDK 6.0 Java is used for the development of the framework components.
- Eclipse IDE for Java
- Weka, the open source machine learning tool for data mining[WFH11]. Weka is used for the data pre-processing, classification, regression, clustering, association rules, and visualization. The following list mentions a few classes of Weka used in this framework
  - weka.core
  - weka.clusterers.SimpleKMeans
  - weka.classifiers.trees.J48
  - weka.classifiers.bayes.BayesNet
  - weka.filters
  - weka.filters.unsupervisedattribute

  - weka.classifiers.trees.RandomForest
  - weka.classifiers.bayes.NaiveBayes

- LIBSVM , the Library of Support Vector Machines
  - weka.classifiers.functions.LIBSVM
  - A standalone program interfaced into WEKA, incorporates many different variants of SVM for building supervised machine learning models to analyze data for classification and regression analysis.

- CUDA Toolkit 5.0

  The NVIDIA CUDA Toolkit 5.0 provides the development environment for building GPU-accelerated applications in C or C++. The CUDA Toolkit includes a compiler for NVIDIA GPUs(nvcc), the math libraries, and tools for the debugging and optimization of the performance of the GPU application.

- Microsoft Visual Studio

  The CUDA project is technically a C++ project (.vcxproj) that is preconfigured to use NVIDIA's Build Customizations. It is imperative to include CUDA Build Customizations for the CUDA program. All standard capabilities of Visual Studio C++ projects will be available. It is very important that before compiling the CUDA program in Visual Studio, the $CUDA_PATH environment variable is set to point to the location of the CUDA Toolkit and the associated .props files.

- Nvidia Visual Profiler, the cross-platform profiling tool that delivered the important feedback which guided to incorporate the code optimization tech-

niques for the CUDA application. The profiler helped in the identification of the bottlenecks and the issues are addressed to yield the best possible and efficient solution. The NVIDIA Visual Profiler is available as part of the CUDA Toolkit.

- TCPTrace, TCPDstat, the statistical tools, helped in forensic analysis phase.
- SiLK and TCPReplay, for the final step of manipulation.
- Python, a Weka wrapper in python is written for converting the .csv file(the original dataset) to .arff(for Weka) format.

3.3 Framework Design and Architecture

3.3.1 Introduction

The forensic investigation of a security event leads to the root cause of the event. It can lead to the source of the attack. Procedure 1 briefly describes the steps involved in the forensic investigation after a security event has occurred.

Procedure 1 Forensic Investigation

Step 1: Incident Response :

Step 1.1: Data Capturing

Step 1.2: Intrusion Detection

Step 1.3: Incident Reporting

Step 2: Conduct Indepth and Conclusive Analysis:

Step 2.1: Data Acquisition, Preparation and Integration

Step 2.2: Analysis(Correlation of intrusion detection information with new Data Captured)

Step 2.3: Reporting and Resolution

Each step is further comprised of several steps. The detailed forensic analysis process is graphically explained in Figure 3.1.

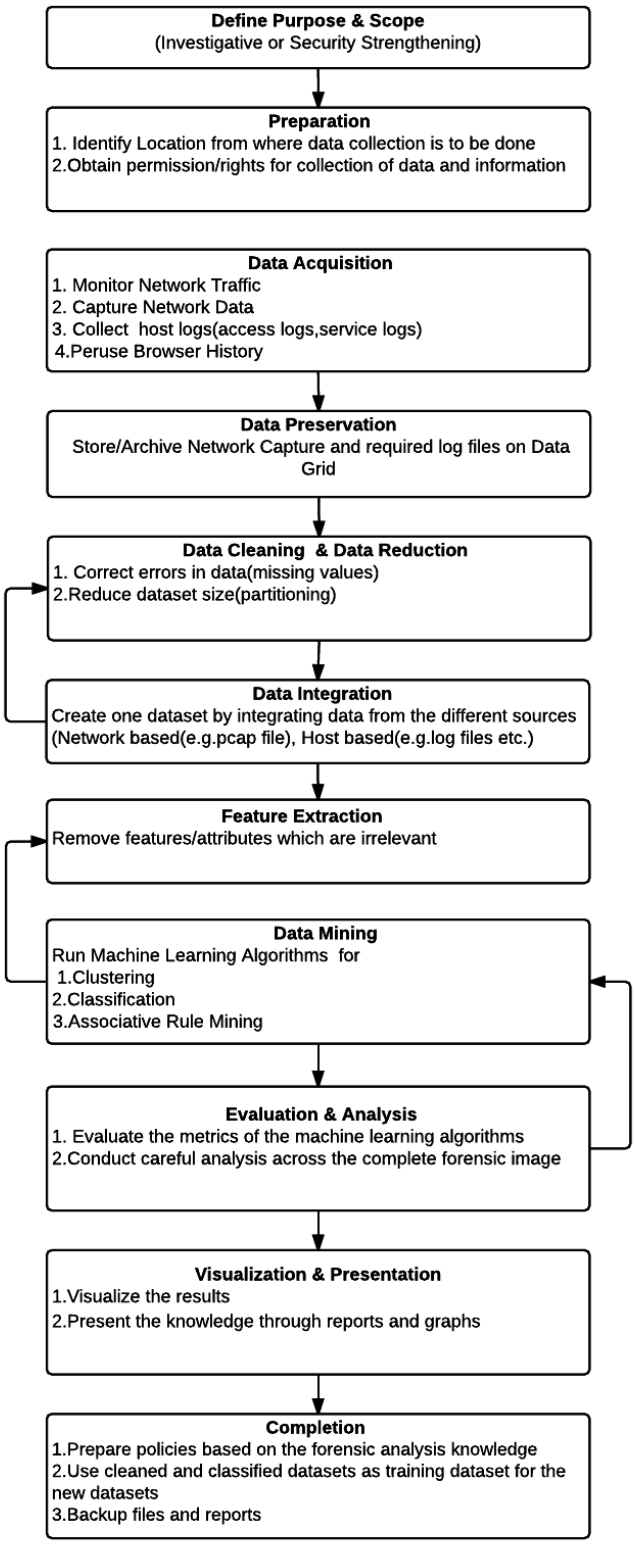

Figure 3.1: Forensic Analysis Process

The process begins by the **Definition of the scope and purpose** of the task. The purpose of performing network forensics could be to identify the vulnerabilities and get prepared to prevent any such future occurrences, and/or to be led to the attack source and produce legal evidence for prosecution of the perpetrators.

In the next step of **Data Preparation**, the location from where the data is to be captured/collected is identified. Proper permissions and rights are obtained from the administrators and the authorities.

Data Acquisition step is where the network traffic is monitored. The network data is captured using the tools like tcpdump or Wireshark. The log files from the host are collected. These are web logs(access logs) and service log files. A web log analyzer tool can be very useful at this stage. Sanjay Malik, Nupur Prakash and Sam Rizvi discussed a web mining technique for extracting useful and meaningful information from a plethora of web documents [MPR10].

The data is stored on the data grid and this is called **Data Preservation**. **Data Cleaning and Data Reduction** are performed by correcting errors in data eg. missing values or incorrect types of attributes. The dataset at this stage can also be considered for partitioning if size is huge. Dataset Reduction can also be done by eliminating redundant data.

The next step of **Data Integration** is very important.The integration of Network based evidence alongwith host based evidence can help find conclusive information of the events comprising the attack. The network based evidence is gathered from the traffic capture files(.pcap files created by tcpdump/wireshark), session data (gathered by tcptrace), statistical information like protocol distribution and flows(analysed by openDPI) and the intrusion detection alerts and data(generated by the CUDA DPI engine). The host based evidence includes browsing history, log

files(e.g. access.log, auth.log), launched programs, keystrokes, accessed files and modified files. Time-based correlation is performed on both network and host data. Bayesian-based correlation algorithm can perform this task with good accuracy. The dataset is now prepared for Data Mining.

Feature Extraction is performed before the data mining step. Irrelevant features or attributes are removed which have less or no contribution to the machine learning. This is discussed in detail in the section labelled Feature Selection/Extraction in Chapter 4 (see 4.3.4.4).

Data Mining involving machine learning and statistical modeling is used for knowledge discovery in the data. Supervised machine learning is used in this research for building the classifier model. Data mining using Weka is discussed in more detail in Chapter 4.

Evaluation and Analysis is the key step of the process. The machine learning algorithms are evaluated for their performance with a set of metrics as discussed in Chapter 5 in the section Evaluation of the Statistical Model (See 5.3.1). First, The training of the classifier is evaluated and then the performance of the classifier with the testing datasets is performed. The evaluation is done for both grid and non-grid platforms. A careful analysis is conducted across the complete forensic image.

Visualization, Presentation and Completion are the concluding steps of the process. The implementation of the framework on the Grid, performs the forensic analysis task. Once the analysis is done using data mining, the forensic analysis reports and logs are generated. Detailed reports of the analysis are prepared. The reports created are both textual and visual. The knowledge gained is presented through visual images and charts. Data visualization is the graphical interpretation

of high-dimensional data, which is particularly appropriate for obtaining an overall view and locating important aspects within a dataset . This is useful in network forensics because the data encountered in digital investigations is often significant in size, multidimensional, and complex. Thereafter, policies are prepared based on the knowledge gained from the forensic analysis. The files and logs are backed up which can be used as evidence. The datasets are held on the data grid as these are to be used for the testing of new datasets.

3.3.2 Design

The framework has been designed using the tools and technologies as mentioned in the previous section. The design objectives are as following:

1. The framework should have a **quick incident response system**. Quick detection, can minimize the amount of damage.

2. The framework should be **efficient**. It should perform the forensic analysis in a reasonable amount of time independent of the quantity of data.

3. The framework should be able to **store forensic data in long-term storage** .

4. The framework should **learn from existing patterns** so that **new analysis takes lesser time**. This provides increased protection against future attacks.

5. The framework should be **scalable**. Addition of new users, data and resources should be seamless and should not affect the performance.

6. The framework should be **reliable** and have high accuracy and low error rate.

7. The frame should be **extensible**. Addition of new features, algorithms and applications should be easy, simple and quick.

The research focusses on how the paradigms of the cost-effective technologies of Grid Computing, GPU Computing and Data Mining are used to meet the design objectives.
In the framework, the high performance computing infrastructure of the grid is the platform where the data-mining techniques of machine learning, clustering and associate rule-mining are applied in a parallel way to datasets distributed over the grid. The key benefits are:

- The compute-intensive forensic analysis is distributed amongst a large number resources which could be geographically dispersed.

- The processing algorithms are available to the data where it is stored, avoiding unnecessary data transmission.

Figure 3.2 gives an overview of the design of the framework. The detailed design is given a little later in the chapter in the next subsection.

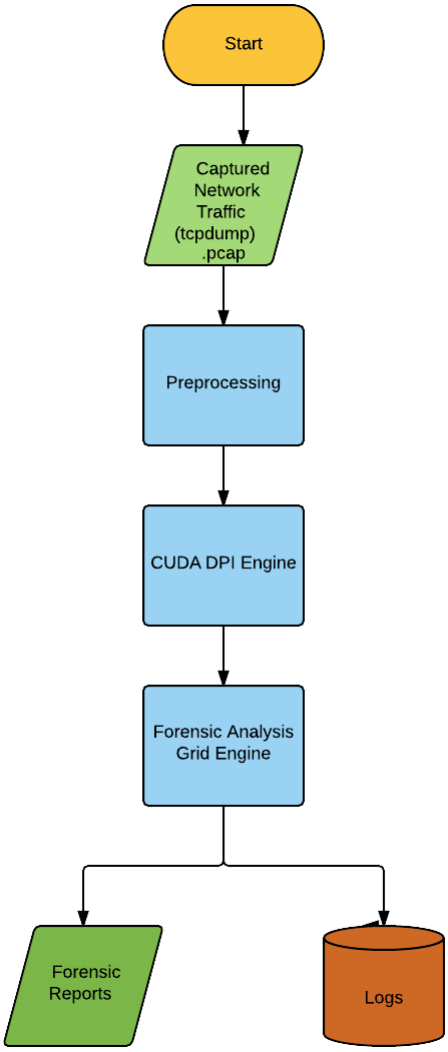

Figure 3.2: Design Overview

Broadly, the design can be classified into three phases. For the first phase of preprocessing, several network security tools have been used. Mainly open-source tools like, tcpdump, Tshark alongwith the libpcap library have been used. Another tool, openDPI has been used to gain insights into the traffic for the particular dataset. Once the primary analysis of the traffic is done, in the second phase, the pattern-matching is done on the data. For this purpose, as identified in the

previous chapters, the Rabin-Karp pattern-matching algorithm is chosen over the classical pattern-matching algorithms. The reason for this was that the objective has been to leverage the power of a GPU to perform the task as fast as possible. For the best parallel processing results, the Rabin-Karp algorithm, because of its key characteristics is the best candidate, based on the study of the various available algorithms. The algorithm is simple yet fast and efficient. This phase raises alerts and creates logs containing information about the attacks. For the new attacks, signature generation can also be done and stored in a signature repository. For the next phase, to further speed up the task and manage the great volume of the captured traffic, the grid infrastructure is used. The web service resource framework on the grid has been implemented. The data mining tool, Weka is thereafter used that helps in the indepth forensic analysis. Figure 3.3 details the architecture of the complete system.

The main components of the architecture are discussed as follows:

3.3.3 Preprocessing

The WinPcap library has been used for capturing the network traffic.The amount of network traffic data captured in a given scenario is huge. Different datasets are created for the implementation and testing phase.

The first step of preprocessing is required to perform network traffic sampling and traffic classification. Different network security tools are used for this purpose. The major ones being, Tcpdump, Wireshark, tshark and Nmap and OpenDPI. Tcpdump, wireshark and openDPI for used for packet classification. OpenDPI

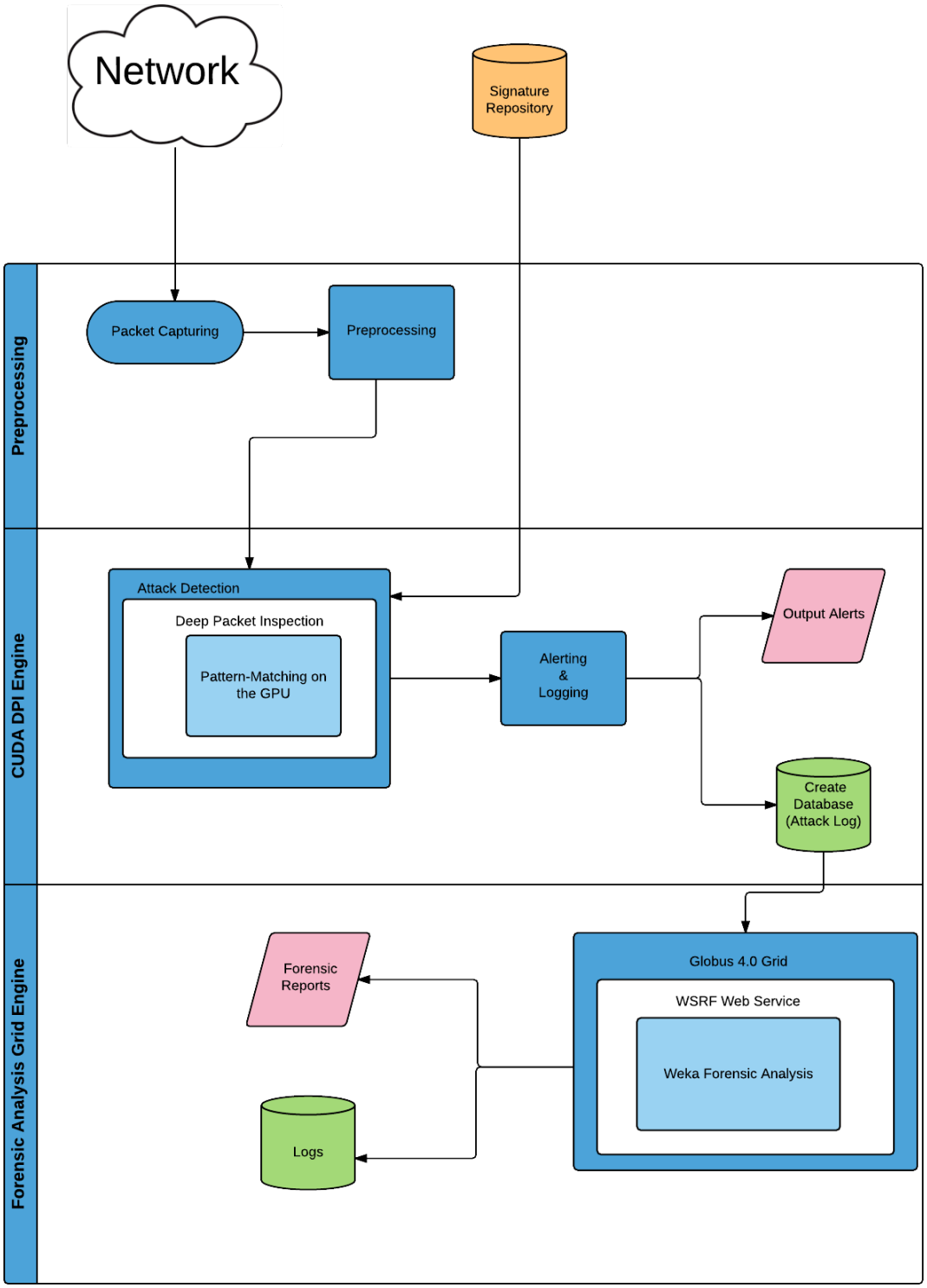

Figure 3.3: Architecture of the Framework

has now been superseded by nDPI. nDPI extends the original library of openDPI. It is a very useful tool in the detection of protocols on both standard and non-standard ports. After the identification of the application protocols, it is possible to differentiate the packets for different flows. Packet filters can perform the task of traffic sampling. Saklani and Tiwari discussed integrated sampling and filtering for enhancing Network Intrusion Detection by introducing randomness as a first object in the packet filter, for the in-kernel random sampling of packets, hosts and connections; Fixed-size associative tables indexed with hash tables also provided richer filter control [ST13]. A good-quality random number generator can be very useful for such applications in network security. Ruppa Thulasiram et al. discussed the use of GPUs together with histogram equalization for true random number generation [CSL+11].

In the preprocessing phase, general monitoring and gaining a quick insight to the nature and statistics of the network traffic is done.

Figure 3.4 displays the statistics for the protocol hierarchy in a typical university traffic dataset.

Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00 %	1778	100.00 %	1508750	0.071	0	0	0.000
Ethernet	99.94 %	1777	99.90 %	1507316	0.070	0	0	0.000
Internet Protocol Version 4	99.94 %	1777	99.90 %	1507316	0.070	0	0	0.000
User Datagram Protocol	1.57 %	28	0.40 %	6102	0.000	0	0	0.000
Domain Name Service	1.57 %	28	0.40 %	6102	0.000	28	6102	0.000
Transmission Control Protocol	98.37 %	1749	99.50 %	1501214	0.070	1518	1362783	0.064
Hypertext Transfer Protocol	12.99 %	231	9.18 %	138431	0.006	116	51135	0.002
Compuserve GIF	1.86 %	33	1.41 %	21202	0.001	33	21202	0.001
eXtensible Markup Language	1.91 %	34	2.08 %	31357	0.001	34	31357	0.001
JPEG File Interchange Format	2.59 %	46	2.27 %	34175	0.002	46	34175	0.002
Media Type	0.11 %	2	0.04 %	562	0.000	2	562	0.000
Text item	0.06 %	1	0.10 %	1434	0.000	1	1434	0.000

Figure 3.4: Statistics of protocol hierarchy in a typical dataset

Figure 3.5 displays the statistics for the load distribution of HTTP traffic in a usual session.

Figure 3.6 depicts the protocol distribution in a typical dataset. The statistics are derived by OpenDPI tool.

Topic / Item	Count	Rate (ms)	Percent
⊟ HTTP Requests by Server	116	0.000734	
⊞ HTTP Requests by Server Address	116	0.000734	100.00%
⊟ HTTP Requests by HTTP Host	116	0.000734	100.00%
⊟ ax.itunes.apple.com	5	0.000032	4.31%
8.18.65.67	5	0.000032	100.00%
⊞ metrics.apple.com	33	0.000209	28.45%
⊞ ax.search.itunes.apple.com	30	0.000190	25.86%
⊞ a1.phobos.apple.com	48	0.000304	41.38%
⊟ HTTP Responses by Server Address	116	0.000734	
⊟ 8.18.65.67	5	0.000032	4.31%
OK	5	0.000032	100.00%
⊞ 66.235.132.121	33	0.000209	28.45%
⊞ 8.18.65.32	4	0.000025	3.45%
⊞ 8.18.65.58	10	0.000063	8.62%
⊞ 8.18.65.82	22	0.000139	18.97%

Figure 3.5: Load Distribution of HTTP

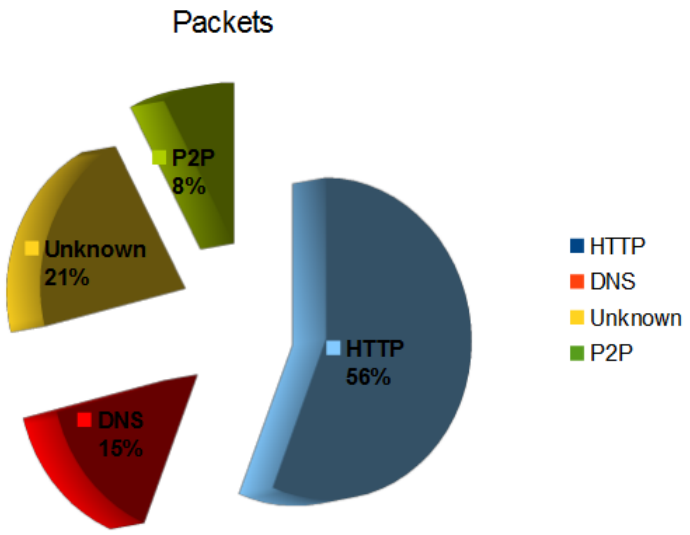

Figure 3.6: Protocol Distribution in a typical dataset

This phase helped in the profiling of the network traffic. It enabled the writing of the ruleset for the next phase of intrusion detection. The university network

traffic mainly consists of TCP (HTTP) followed by DNS queries. Therefore, three attack categories are identified, Denial of Service(DoS), SQL Injection and Cross-Site Scriptng(XSS). These are three dangerous attacks and always are in the top list of the annual vulnerability reports.

A denial of service (DoS) attack is a malicious attempt to make a server or a network resource unavailable to users, usually by temporarily interrupting or suspending the services of a host connected to the Internet.

SQL Injection occurs when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization by inserting SQL meta-characters and commands into Web-based input fields in order to manipulate the execution of the back-end SQL queries.

Cross Site Scripting allows attackers to execute scripts in the victim's browser by embedding script tags in URLs. The unsuspecting user clicks on them, which causes the execution of the malicious Javascript on the victim's machine and can result in hijacking of user sessions, defacing web sites, or redirecting the user to malicious sites.

3.3.4 CUDA DPI Engine

3.3.4.1 Pattern matching based DPI on GPU

The Graphics Processing Unit (GPU) which was primarily associated with processing graphics is rapidly evolving towards a more flexible architecture and this has encouraged research and development in several computationally demanding non graphic applications where its capability can be utilized for computations

that can leverage parallel execution [Oe08]. General-Purpose Computing on GPU (GPGPU), also known as GPU Computing exploits the capabilities of a GPU using APIs such as OpenCL and the Compute Unified Device Architecture (CUDA) [CUD13]. The opportunity is that one can implement any algorithm, not only graphics, but the challenge is to obtain efficiency and high performance in the field where they replace or support the traditional CPU computing.

A good candidate for such GPGPU is Deep Packet Inspection (DPI), the technology, where the appliance has the mechanism to look within the application payload of the traffic by inspecting every byte of every packet, and detect intrusions which are more difficult to detect as compared to the simple network attacks [TMN07]. The computational and storage demand for such inspection and analysis is quite high. An NIDS spends 75% of the overall processing time of each packet in pattern matching [Cea04]. Under high load conditions, high processing abilities are needed to process the captured traffic. A vast majority of earlier research focuses on specialized hardware for the compute intensive Deep Packet Inspection [RE04]. Application Specific Integrated Circuits (ASICs) [REKG06], Field Programmable Group Arrays(FPGAs) [DL06] and network processor units(NPUs) [PY06] provide for fast discrimination of content within packets while also allowing for data classification. Engaging special hardware translates to higher costs as the data and processing requirements for even medium size networks soon increase exponentially. Some researchers have discussed an interesting entropy based technique for fine-grained traffic analysis [WP05] [TWWL12]. Jeyanthi et al. presented an enhanced approach to this behavior-based detection mechanism, the “Enhanced Entropy” approach, by deploying trust credits to detect and outwit attackers at an early stage. The techniques seem good, but for or best performance, call for

methods like bi-directional measurements and this imposes additional computing overhead [JIKK13].

The task of pattern-matching for DPI can be performed on a GPU, at a much more reasonable cost as GPUs, being a necessary component of most computers these days, are now readily and easily available. The task is split into parallel segments resulting in searching the string much faster than a CPU.

There are several pattern-matching algorithms for DPI, Aho-Corasick and Boyer-Moore being the most popular [AC75][BM77]. In this framework, the Rabin Karp algorithm [KR87] has been selected from over these and several other popular algorithms because it involves sequential accesses to the memory in order to locate all the appearances of a pattern and is based on compute-intensive rolling hash calculations. The algorithm has delivered good performance and efficiency when executed on a multicore GPU(in this study, the Nvidia GeForce 635M).

3.3.4.2 Signature Based DPI

Pattern matching is the act of checking a given sequence of tokens for the presence of the constituents of some pattern. In contrast to pattern recognition, the match usually has to be exact. Signature-matching or Pattern-matching intrusion detection systems have major I/O bound performance limitations caused by the overhead of reading packets from the network interface card. Packet reading becomes a bottleneck when the number of packets overwhelms the IDS host's internal packet buffers[JB06]. The solution to this problem in this approach is to capture the network traffic using an open source packet capturing software. The packet capture file is the input to the CUDA application, The CUDA program rabinkarpGpu.cu implements the rabin-karp pattern-matching algorithm on the

GPU.

3.3.4.3 Rabin-Karp Algorithm

The literature review of related work supported the choice of the multiple-pattern matching algorithm, Rabin-Karp [KR87].

The key to the Rabin-Karp algorithm's performance is the efficient computation of hash values of the successive substrings of the text. The Rabin fingerprint is a popular and effective rolling hash function. The Rabin fingerprint treats every substring as a number in some base, the base being usually a large prime. The essential benefit achieved by using a rolling hash such as the Rabin fingerprint is that it is possible to compute the hash value of the next substring from the previous one by doing only a constant number of operations, independent of the substrings' lengths.

The Rabin-Karp algorithm is inferior for single pattern searching to Knuth-Morris-Pratt algorithm, Boyer-Moore string search algorithm and other faster single pattern string searching algorithms because of its slow worst case behavior. However, it is an algorithm of choice for multiple pattern search, most suitable for this work, because of its characteristics. The rolling-hash computations' based algorithm is as follows :

A naive way to search for k patterns is to repeat a single-pattern search taking O(n) time, totalling in O(n k) time. In contrast, the variant algorithm above can find all k patterns in O(n+k) time in expectation, because a hash table checks whether a substring hash equals any of the pattern hashes in O(1) time.

The key to RabinKarp is to incrementally update the hash as the potential match moves along the string to be searched. The hash function uses a prime q, whose

Algorithm 2 Rabin Karp Algorithm (Serial Implementation for CPU)

$matches = \{\}$

$pattern\_hash = hash(pattern)$

$substring\_hash = hash(s[0 : m])$

for *position* from 0 to $n - m - 1$ do **do**

update *substring_hash* to hash ($s[position : position + m]$)

if $substring\_hash = pattern\_hash$ **then**

add *position* to *matches*

end if

end for
return *matches*

value should be chosen such that 256q is no larger than a computer word, which makes the algorithm truly efficient. Exhibit 3 describes the checksum(hash) function as given in [KR87].

Karp et al. devised a theorem for parallel pattern-matching. They suggested that the computation of the hashes for the substrings and their comparisons with the hash of the text string can be done in parallel using multiple processors in constant time. This motivated to experiment with the algorithm on a multicore GPU.

The string matching problem for a pattern of length n and a text of length m ($n <= m$), where one can find all matches, can be solved by m processors in time $O(logm)$ with probability of error smaller than $0.697/m_k$.

Given a text string s of length n, and a pattern of length m, the algorithm computes

Exhibit 3 Checksum Function

A binary string $X = x_1, x_2...x_n$ can be regarded as a binary representation of the integer

$$H(X) = \sum_{i=1}^{n} x_i 2^{n-1} \tag{3.1}$$

For any integer q, the following is the hash function:

$$H_q(X) = H(X) \ mod \ q \tag{3.2}$$

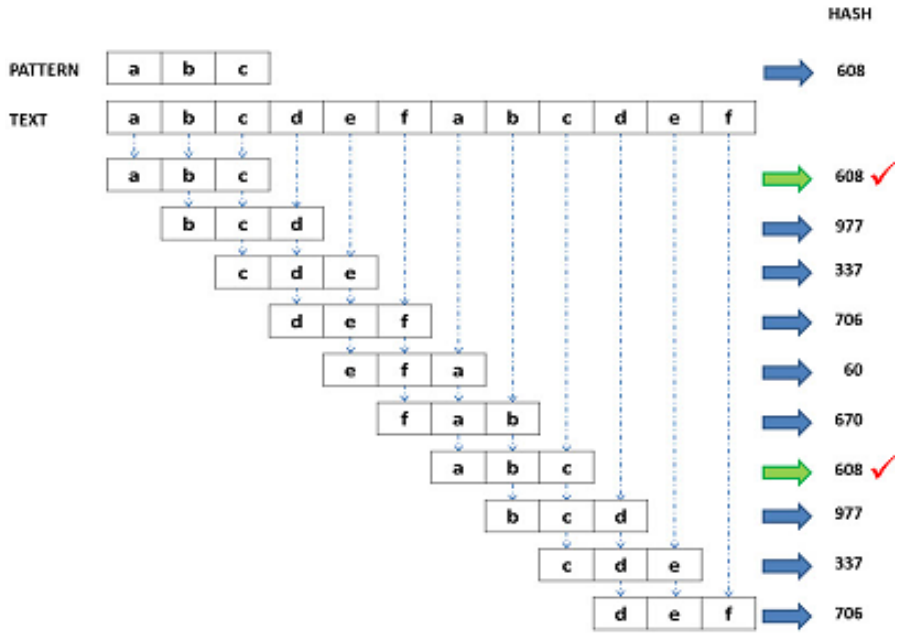

Figure 3.7: Rabin-Karp Algorithm

rolling hash for both the pattern and consecutive substrings of length m of the searched string. Every time a substring's hash equals that of the pattern, a match is reported. Figure 3.7 illustrates the idea behind the Rabin-Karp algorithm. The rows with a red checkmark indicate the situations where the hash for the pattern and the text match. The algorithm can find any of a large number, say k, fixed length patterns in a text.

3.3.4.4 Parallel Implementation of the Rabin-Karp Algorithm

The CUDA framework has been used to develop and execute the algorithm on the GPU. The experiments that have been conducted on random as well as real traffic indicate that the proposed algorithm is upto 14X times faster than the Rabin Karp algorithm being executed on a CPU for the pattern search. The host computer issues a kernel for the pattern-match to the GPU, which is executed on the device as several threads organized in thread blocks. One or more thread blocks, organized as warps are executed by each multiprocessor. Each thread that executes the kernel is given a unique thread ID that is accessible within the kernel through the built-in *threadIdx* variable. The number of threads per block and the number of blocks per grid are specified in the <<< ... >>> syntax as type *int*. Each block within the grid can be identified by a one-dimensional or two- dimensional index accessible within the kernel through the built-in *blockIdx* variable. The dimension of the thread block is accessible within the kernel through the built-in *blockDim* variable. The architecture of the pattern-matching process is given in Figure 3.8.

The implementation is listed in a step-by-step manner as below in Procedure 4:

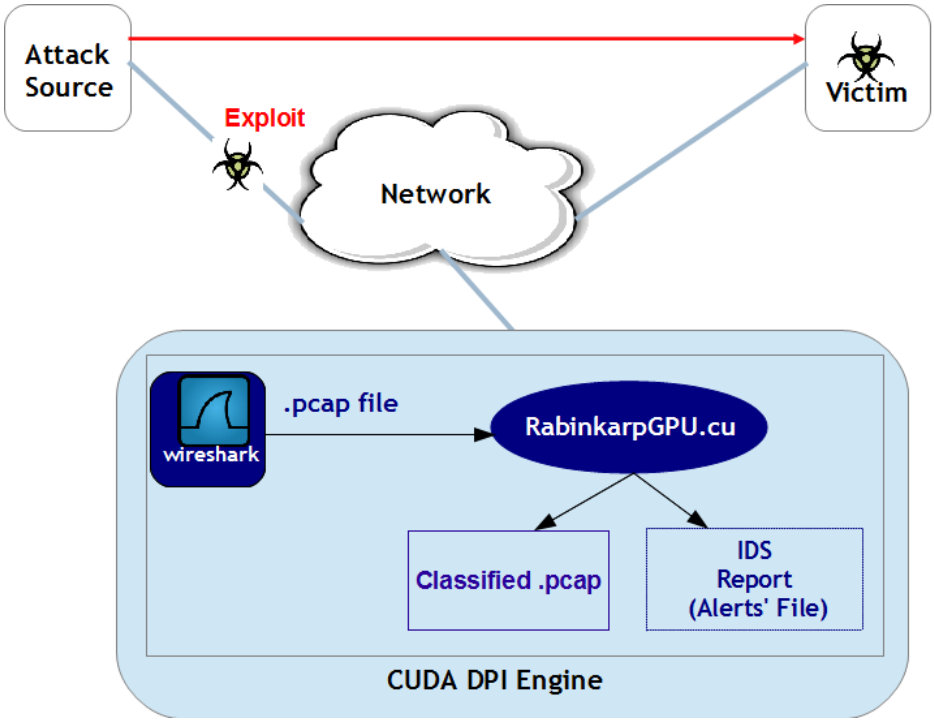

Figure 3.8: Architecture of the CUDA DPI Engine

Procedure 4 DPI on the GPU

Step 1: **Network Traffic Capture on the Host(CPU)**
Tool: The open source protocol analyzer Wireshark [Wir10] (known as Ethereal till 2006), or the classic old sniffer, Tcpdump [Tcp10], whose GUI is not as good as Wireshark but it requires fewer resources and also has fewer security holes. Both tools rely on the packet sniffing libraries, WinPcap and Libpcap[Tcp10].

Step 2: Transfer **of packets(*Text*) to the Device(GPU)**, through a covert channel

Step 3: **Copy patterns from the host to the shared memory**
(cudaMemcpyAsync() is used as it is non-blocking to the host, so control immediately returns to the host thread.)

Step 4: Considering each thread in a block responsible for one test pattern, **load a chunk of text into device memory**, with each thread doing one test pattern

Step 5: **Loop through the chunk of text looking for matches**

Step 6: Once pattern-matching completed with the loaded chunk, **load in the next chunk of text and repeat**

Step 7: When **end of file, store matches in a global array**

The pseudocode for the parallel implementation of the Rabin-Karp Algorithm for the GPU is given in Procedure 5:

Procedure 5 Rabin Karp Algorithm for GPU (rabinkarpGpu.cu)

{P is the set of all fixed-string patterns (string patterns of Snort V2.8)}
{T is the input string elements}
{q is the prime}

Phase 1: Data Loading (by the host)

1. Host Loads the patterns from the dictionary in P
2. Reads-in the PCAP file(s) for the Text and loads a packet at a time in T
3. Set the value of the prime number for computation
4. Host program calls the Rabin-Karp pattern-matching GPU Kernel with the parameters P, T and q

Phase 2: Pattern-matching (in the kernel)

1. GPU kernel calculates the checksum(hash) value of the pattern and the first substring of text(length m)
2. Slide the pattern over text one by one
3. Compare the checksum values of the substring (current window of text) and pattern
4. If the checksum values match then only check for characters one by one
5. If text and pattern match character by character then add position to matches
6. Calculate hash value for next substring and repeat from step 4
7. Return matches to the host

The pattern-matching algorithm processes the network data and raises alerts which are stored in an 'alerts file'. The forensic analyst can peruse these alerts for a kind of proactive security approach if the time gap between the processing and the data capture is minimal. The CUDA DPI engine marks the packets as 'Benign' or 'Attack', which are made available to the Forensic Analysis Engine as the Input.

3.3.5 Forensic Analysis Engine

Once the DPI based intrusion detection is done and the alerts are stored in an alert file, a dataset of the packet instances is created. The dataset contains pertinent packets' info. like Timestamp, Source/Dest. IP Address, Source/Dest. Port Number, etc. and an additional nominal attribute, *Class*, which is assigned the attribute value 'Attack', if the CUDA DPI engine successfully detected an intrusion by pattern-matching with the attack rule else it is marked as 'Benign'. The dataset is then provided as an input to a 'Correlation Engine' which performs correlation for packet classification to the appropriate *Attack Type* category which can be 'DoS', 'XSS', 'SQL-Injection' etc.. DoS, XSS and SQL-Injection are the major attack types identified in the university captured data. Different data-subsets are created for each different *Attack-Type*. The data-subsets created are then used for the machine learning. Three datasets are created, which are named as 'DoS Attack Dataset', 'XSS Attack Dataset' and 'SQL-Injection Attack Dataset'. The volume of even three types of such datasets would be increasing by each day. For the purpose of forensic analysis, these datasets need to be maintained in the storage. Data Grid is used for archiving the datasets which would allow an indepth and unhurried analysis.

Forensic analysis engine is implemented on a Globus 4.0 grid. Weka is executed on the grid which performs machine learning from the training dataset created by CUDA DPI engine. New datasets can be classified accurately by the chosen classifier. After the implementation and comparison of the performance of the results, J48 classifier is selected as it has provided the lowest error rate and a quick learning time. The details of the implementation are given in Chapter 4.

3.4 Summary

The chapter discussed the design and architecture of the framework. The detailed architecture of the framework is presented. The chapter discusses the main components of the architecture: pre-processing, CUDA DPI Engine and the Forensics Grid Engine. After the initial preprocessing, the pattern-matching is done for detecting DoS, XSS and SQL Attacks. Once the malicious packets are identified, the alerts are stored in an alert file and a dataset created from the processed traffic capture(.csv file now with an added class attribute) and the log files from the host. The datasets are stored on the data grid. When the forensic analysis task submitted to the Grid, this dataset is analysed on Weka, whenever processing resources are available on the grid.

Chapter 4

Implementation Details of the Framework

4.1 Introduction

This chapter discusses the process of realizing the forensic analysis framework in accordance to the proposed work. Globus 4.0 setup is used for the Grid environment. The framework is implemented as a WSRF web service on the Grid setup. Section 4.2.2 discusses the details about the environment and the technologies used for the implementation. The intrusion detection is done in CUDA on a GPU. Section 4.2 discusses the configuration of the GPU.

The chapter presents the details of the web service based data mining approach for the forensic analysis. The machine learning algorithms used in the framework for classification and clustering are also presented. The chapter also discusses the preparation of the datasets for the machine learning experiments.

Table 4.1: NVIDIA GPU Configuration

NVIDIA GT 635M	
GPU Engine Specs:	
Unified Shaders	144
Texture Mapping Units	24
Render Output Units	24
Graphics Clock (MHz)	Up to 675 MHz
Processing Power	388.8
Memory Specs:	
Memory Interface	DDR3
Global Memory	2G
Constant Memory	64K
Texture Memory	8K
L2 Cache	128K
Local Memory	512K
Memory Interface Width	Up to 192bit
Memory Bandwidth (GB/sec)	Up to 43.2
Other Specs:	
Compute Capability	2.1
Threads per block	1024

4.2 Experimental Setup

4.2.1 GPU Configuration

The GPU used for the GPGPU computing in the framework is the nVIDIA GEFORCE GT 635M. The configuration of the GPU is given in Table 4.1.

4.2.2 Grid Setup

Proposed model is tested on the Grid environment at Thapar University (TU), setup using Globus toolkit. Globus toolkit 4.0 has been installed on different resources and the service they provide were deployed as grid services. Grid en-

vironment setup consisted of 44 Intel Dual Core 2.2GHz processor Windows XP nodes, 20 dual 2.4GHz Xeon Linux nodes and 5 nodes 450MHz PII Linux nodes. Each node has 1GB RAM and 80GB HDD.

4.3 Implementation of the framework on the Grid

4.3.1 Web services oriented architecture of the Network Forensics Grid

SOA (Service Oriented Architecture) for the grid provides methods for exposing services and allowing computers to talk to each other in a highly heterogeneous environment. The benefits of a SOA architecture on the grid is that it provides loose coupling which results in the system being flexible and scalable, services are replaceable and it also adds fault tolerance [ST05].

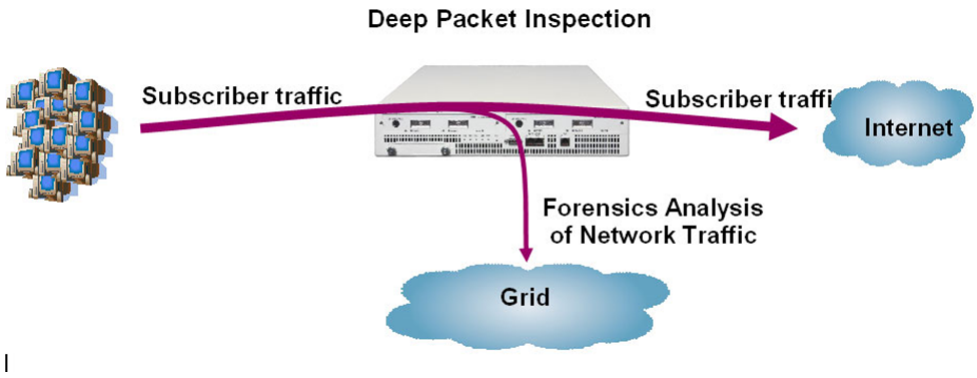

Figure 4.1: Forensic Analysis on Grid

A Deep packet Capturing (DPC) machine archives the network traffic and forwards it to the web-services based application on the Grid, which is going to be referred as the DPI-FA Grid in this chapter. In the web service oriented architecture model

of the DPI-FA Grid, every application runs as a service and is associated with a WS-resource. The web service is defined in the Web Services Description Language (WSDL) document. The Resource Properties Document schema, which explicitly describes a view of the resource with which the client interacts, is referenced by the WSDL description of the service. By exploiting the Resource Properties Document schema, WSRF enables the mechanical definition of simple, generic messages which interact with the WS-Resource. A WS-Resource is accessed through a Web service at a particular location which is described by a WS-Addressing [WS-Addressing] EndpointReference (EPR), which contains the URI of the Web service endpoint as follows:

```
. . . String url = "http://local/dpiWebService"
EndpointReferenceType epr = new EndpointReferenceType();
epr.setAddress(new Address(url));
epr.setProperties(props);
```

This EPR is known to the client/requester as the service-name. The contents of the EPR are generated by the service provider and are not intended to be understood by the client. The client takes the EPR from the response message and associates it with the client's programmatic representation of the resource (for example, a program object, reference or variable) so that all calls which use that representation are directed to the resource. The 'GetResourcePropertyDocument' operation can then be used to retrieve the values of all resource properties associated with the WS-Resource. The SOAP envelope for the service returns the response to the request with the 'GetResourcePropertyDocumentResponse' message [Ban05].The

DPI based Forensic Analysis Web Service (DFA-WS) will be a network accessible entity that will process the SOAP messages. SOAP messages will be received by the hosting environment (Globus Container), which is domain independent. It would identify and invoke the appropriate code to handle the message. The WS implementation handles the SOAP message through domain specific code.

The service interface (a WSDL document) is published to the service registry. A service requester (a potential client) queries the service registry to search for a service that meets its needs. The registry returns a list of suitable services, and the client selects one and passes a request message to it. The client then binds to the service provider in order to execute the service (using SOAP). The web services are deployed into the GT4 web container. The GT4 Java WS Core Code implements WSRF and WS-Notification as well as supporting code for security and management[Sta06]. The Service is coded using Java and the Globus java ws-core API, which depends on a GT component, the Java Commodity Grid (CoG) Kit, and some 3rd party software including Apache Axis, Apache Xerces, Apache Tomcat JNDI, Apache Addressing, and Apache XML Security, to name a few.

Apache Axis is one of the best free tools available for implementing and deploying web services, and also for implementing the web service clients. The code is designed to be used with Apache Axis[Apa14] as a SOAP engine plus other relevant WS components such as the WS-Addressing and WS-Security [Rec06], [Oas06]. The end user can upload the .pcap file to the portal server based on the WSRF through a web browser. The portal is based on the Java Portlet model. The deployment of the GT4 Java Web Service in a step-by-step manner is described in Procedure 6.

Procedure 6 Deploying the GT4 Java Web Service

Step1: Define the service's interface in the Grid Web Service Description language(GWSDL) file, which describes the service's abstract interface.

Step2: Implement Service by coding the service in java (extends GridSrviceImpl) and specify resource properties.

Step3: Configure the Deployment by defining Deployment Parameters in Web Services Deployment Descriptor (WSDD) file and Java Naming and Directory Interface (JNDI)) deployment file that describes various aspects of the service's configuration.

Step4: Compile and generate the Grid Archive(GAR) file, which is a single file containing all the files and information the grid services container would need to deploy the service. The GAR files are deployed using *globus-deploy-gar* and undeployed using *globus-undeploy-gar*.

This step converts the GWSDL into WSDL and creates the stub classes from it. It compiles the stub classes and the service implementation. All the files are organized into a very specific directory structure. Compilation generates application specific interface routines that handle the de-marshalling/marshalling of the Web service's arguments from/to SOAP messages. (Ant,build.xml)

Step5: Deploy the service into a grid service using Ant. The Ant task calls the *generateLauncher* target which is specified in `$GLOBUS_LOCATION/share/globus_wsrf_common/build-launcher.xml`.

4.3.2 Data Mining using WEKA

Weka is an easy to use, yet powerful tool for machine learning and data mining. The collection of machine learning algorithms for data analysis and predictive modeling has good graphical user interfaces for easy access to their functionality [WFH11]. The following features of Weka are the factors for motivation to select Weka as a tool for statistical modeling :

- Portability, because of full implementation in the Java programming lan-

guage and hence the ability to run on almost any modern computing platform.

- Good graphical user interface.

Table 2.3 in Chapter 2 discussed the types of machine learning methods. The approach is based on the 'Supervised Learning' model. Essential information is presented to the machine in .arff format. The machine will take in cases from the information included and infer expectations on their basis. The forensic analysis component can be trained with the machine learning techniques so that for new attacks, no time is wasted in re-conducting Deep Packet Inspection of the traffic.

4.3.2.1 Dataset Clustering:

The datasets are clustered to groups by applying k-means clustering algorithm. Clustering techniques apply when there is no class to be predicted but rather when the instances are to be divided into natural groups. These clusters presumably reflect some mechanism at work in the domain from which instances are drawn, a mechanism that causes some instances to bear a stronger resemblance to each other than they do to the remaining instances. The k-means cluster algorithm forms clusters in numeric domains, partitioning instances into disjoint clusters. Clustering allows for unsupervised learning. The machine will learn on its own, using the data. The weka clustering algorithm is executed on the three attack datasets. Figure 4.2 shows the screenshot of the following weka clustering algorithm on the DoS dataset.

weka.clusterers.SimpleKMeans -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

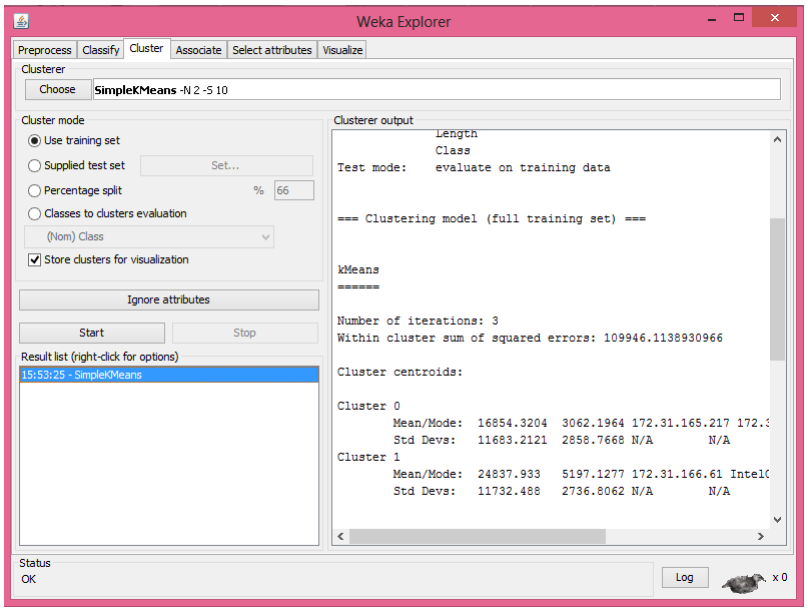

Figure 4.2: Labelled Dataset(Weka Clustering on DoS Dataset)

4.3.2.2 Dataset Classification

Classification is an important aspect of Machine Learning. Figure 4.3 explains how the classification model built by the classifier receives the training dataset as the input, learns and then can label new test datasets.

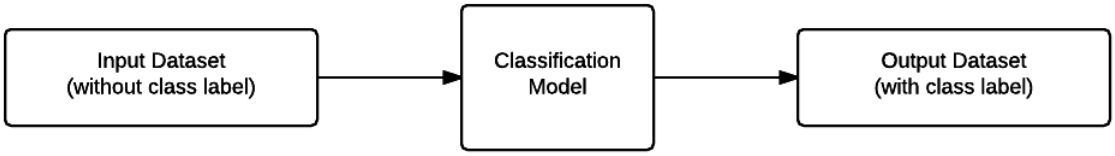

Figure 4.3: Classification Model

The classifier model built by training with the labelled dataset can classify new datasets quickly by applying probabilistic algorithms, regression models, or clustering methods based on Principal Component Analysis. Thus, it has been possible to build a Behaviour-based intrusion detection system which has the ability to examine existing traffic behaviour (from the captured traffic which has once been exposed to the DPI based intrusion detection).

The classifier learning process is a very time-consuming process for large datasets. Though it is possible to use datasets with fewer instances for training the classifier, it has been observed that more the number of instances in the dataset available for training, better the model that is built.

Random sub-sampling, Leave-one-Out and Cross-Validation are some of the methods used by the classifier. The k-fold cross-validation has been studied in this section. This method builds a good model which is not only built faster but also has a lower error-rate. In n-fold cross-validation, a labelled dataset is partitioned into n folds and n iterations of training and testing are performed. One fold is used as a test set for each iteration and the rest of the dataset of k-1 folds is used as the training set. The classifier learns on the training set and is validated on the test data. Figure 4.4 shows the approach for building the classification model.

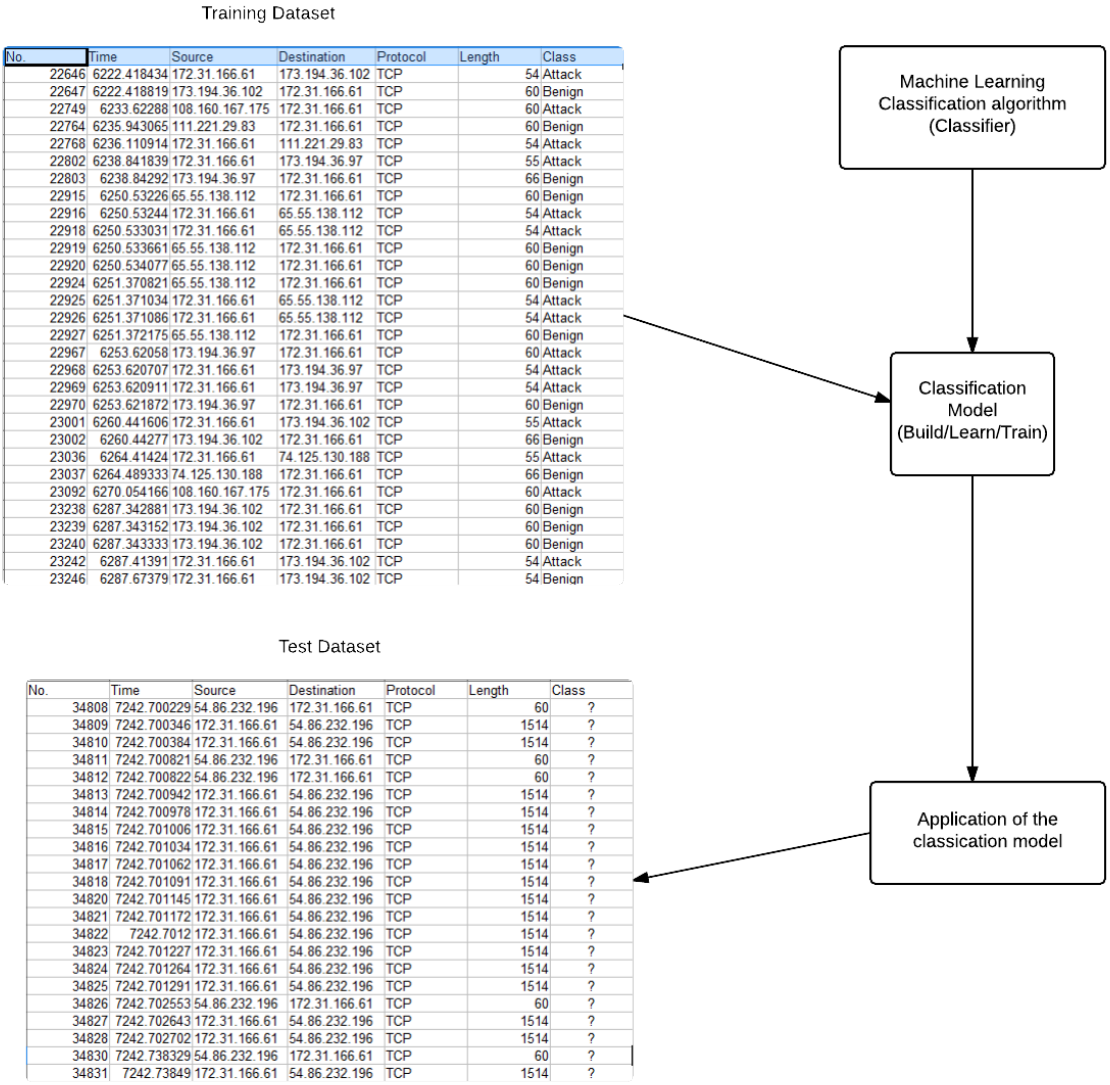

Figure 4.4: Approach to build the classification model

4.3.3 Machine Learning Classifier Algorithms

The following section discusses the popular Supervised Learning Classifiers:

1. **Naive Bayes**

Naive Bayes classifiers are a group of probabilistic classifiers that apply the Bayes' theorem with the naive assumption that the features of the training dataset are independent to the given class labels. These classifiers require a small amount of training data for the estimation of the necessary parameters. The Naive Bayes classifier relies on the discriminant function as seen in equation 4.1:

$$p(C_i|x) = p(C_i) * p(x|C_i)/p(x) \tag{4.1}$$

Let the dataset be D with the Features $X = (x_1, x_2, ...x_n)$ and Class C with j labels $C_j, j = 1, 2, ...N$. This algorithm computes the conditional probabilities $P(x_j|c_i)$ and prior probabilities $P(c_i)$ on given training dataset to build the predictive model. $P(c_i)$ are computed by counting data which present in the Class label C_i divides the resultant count based on the number of the training data. The same way is followed to compute the probabilities through observed frequency of feature distribution in x_j within the training dataset which is labelled. The posterior probability is computed on each class to predict the unknown labelled data.

2. **Bayesian Net**

A Bayesian network or a Bayes Net, is a probabilistic graphical model that represents the dependencies of the variables via a directed cyclic graph(DAG). The task in Bayes Net structure learning is to find a structure of the Bayes Net that describes the observed data the most. It is impossible to enumerate all possible structures, and then score them for even few number of nodes

in Bayes Net. K2, a heuristic-based algorithm to speed up the search to reach a local maximum. K2 is a greedy search algorithm, which starts at an initial structure in the structure space, considers all nearest neighbours of the current point, and moves to the neighbour with the highest score than the current point(which means the local maximum has been reached), the algorithm stops. The algorithm is as follows:

Algorithm 7 Bayesian Network

Input: Training Dataset;

Output: A Bayesian Network

1. Generate the initial Bayesian Network
   evaluate it and set it as the current Bayesian network

2. Evaluate the neighbour of the current Bayesian Network

3. If the best score of the neighbour is better than the score of the current Bayesian network,
   set the neighbour with the best score as the current Bayesian Network and return to Step 2

4. Otherwise, stop the learning process.

Bayesian network classifiers learn using hill climbing(K2, B) and general purpose(tabu search) algorithms under the presumptions:

a.) All attributes are nominal attributes. In Weka, numeric attributes are predescretized by using the filter,

weka.filters.unsupervised.attribute.Discretize

b.) There are no missing values in any instance. Any such values in Weka are replaced globally by using the filter

weka.filters.unsupervised.attribute.ReplaceMissingValues.buildClassifier

If the data conforms to to these two assumptions, the classifier is built, else it gives a warning message.

3. **J48 (C4.5 Decision Tree)**

C4.5 algorithm is a very popular algorithm which was developed by Ross Quinlan [**?**]. Weka classifier package has its own version of C4.5 known as J48. Algorithm J48 is an optimized open-source java implementation of C4.5 rev. 8.

Algorithm 8 J48 Classifier

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision node that splits on a_best
5. Recur on the sublists obtained by splitting on *a_best*,and add those nodes as children of node

4. **Random Forest**

This highly accurate machine learning algorithm can run efficiently on large datasets. It is an ensemble classifier which means that it consists of several decision trees. It outputs the class that is the mode of the classes output by the individual trees. Each tree is constructed using the algorithm 9 as given below:

Algorithm 9 Random Forest

1. Let the number of training cases be N, and the number of variables in the classifier be M.

2. m is the number of input variables to be used to determine the decision at a node of the tree; m should be much less than M.

3. Choose a training set for this tree by choosing n times with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.

4. For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.

5. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

In a random forest algorithm, these two parameters are important:

- ntree : The number of tree grown
- mtry : The number of variables used at each split

4.3.4 Statistical Modeling and Datasets' Anomaly Detection in Weka

The following discussion highlights the important aspects involved in the statistical modeling with Machine Learning techniques in Weka.

4.3.4.1 Datasets for Data Mining

For the purpose of this research synthetic datasets have not been used even though several standard datasets are freely available like, the NSL-KDD dataset. After profiling the network traffic and observing the pattern of protocol distribution,three attack categories are identified. Original datasets have been created and intrusion detection carried out on GPU using the Rabin-Karp pattern-matching algorithm. Two datasets are created, one for the DoS Attack and another one for both the XSS as well as the SQL attack. The two datasets are classified by using the class label, *Class*, which is assigned the value '*Benign*' or '*Attack*' after the deep packet inspection by the CUDA DPI Engine. The datasets which are going to be referred to as the attack datasets are loaded into weka and the different tasks of data mining can then be performed.
The two datasets associated with the three types of attacks are:

1. DoS Attack Dataset
2. XSS & SQL-Injection Dataset

- **DoS Attack Dataset:**

The DoS Attack dataset has been created from the correlation of the network data output file and the host data(service logs etc). The network data output file is created by the intrusion detection module(CUDA DPI engine) after it has processed the original datasets created from the network data(gathered via tcpdump) and the host web log data, and marked the packets as 'Benign' or 'Attack'. This file contains all the attributes of the network capture(right up to the flags), and has now an additional nominal attribute named *class*.

- **XSS & SQL-Injection Dataset:**

The XSS & SQL Attack Dataset has been created from the Apache web server web logs. Their feature set is based on the HTTP CISC 2010 dataset's structure. HTTP CISC 2010 is developed by the Consejo Superior de Investigaciones Cientificas - CSIC at the Spanish National Research Council(www.csic.es). Sanmeet et al. identified eight features from the HTTP CSIC 2010 based on the Information gain score dataset [KS14].

4.3.4.2 Machine Learning of Datasets in Weka

The datasets for Weka are created after the processing of the network capture dataset and the web logs dataset by the CUDA DPI Engine, for intrusion detection. These datasets now have an additional attribute, Class, which marks each instance of the dataset as 'Benign' or 'Attack'. This dataset in .CSV format needs needs to be prepared before it is clustered or classified. The process is explained in procedure 10:

4.3.4.3 Dataset file format Conversion

An ARFF (Attribute-Relation File Format) file is an ASCII text file that is the file format that has been created by the Machine Learning Project at 'The University of Waikato' for use with the Weka machine learning software. The file describes a list of instances sharing a set of attributes. The file has two sections:

1. Header
2. Data

The Header contains the name of the relation, and a list with the names and types

Procedure 10 Preparation of DoS and XSS & SQL Attack Datasets for Weka

Step 1: For DoS Attack Dataset :
Step 1.1: Open .pcap file in *wireshark* to **identify the network data attributes needed for the dataset.** The packet header browserdetails panel displays the names.
Step 1.2: Run *tshark*, add the identified attributes to the -e arguments list and print to file in CSV format.

Step 2: For XSS & SQL Attack Dataset
Step 2.1: Collect attributes from the host files(inspect log files,service files)
Step 2.2: Perform Data Integration(network data and host data)and create dataset as .CSV file

Step 3: Input the datasets to the CUDA DPI engine where deep packet inspection is performed

Step 4: After pattern-matching, the instances in the datasets are marked as 'Attack' or 'Benign' in the newly added *Class* attribute

Step 5: Convert .csv to .arff format(*Python Wrapper Code*) The type of each attribute, *numeric* or *nominal*, is specified.

Step 6: Perform Feature Selection

Step 7:Perform clustering

Step 8:Perform training of classifier

Step 9:Test new data

of the attributes.

The Data section contains the data declaration line and the actual instances represented on a single lines, the end of the instance denoted by a carriage return. The attribute values for each instance are delimited by commas.

Algorithm 11 shows the python pseudocode to convert the .csv to .arff file format.

Algorithm 11 Python Weka Wrapper (.csv to .arff Converter)

```
import weka.core.jvm as jvm

jvm.start()

from weka.core.converters import Loader, Saver

loader = Loader(classname="weka.core.converters.CSVLoader")

data = loader.load_file("d:\dpi\thapar3.csv")

print(data)

saver = Saver(classname="weka.core.converters.ArffSaver")

saver.save_file(data, "d:\dpi\thapar3.arff")

clusterer = Clusterer(classname="weka.clusterers.SimpleKMeans",
options=["-N", "3"])

clusterer.build_clusterer(data)

print(clusterer)
```

4.3.4.4 Feature Selection/Extraction

This is a crucial step because the original dataset not only has a large number of instances, but also a big number of attributes. Not all attributes contribute to the accuracy of the classification process. The training time of the classifier is highly important. The classifier with higher accuracy and lower training time is considered to be good. With careful consideration of which attributes are necessary for the machine learning algorithm, features' vectors are identified and the attribute set is reduced. This process significantly alters the dimensionality of the task. El-Sayed and N.Feras discussed an approach to attribute selection based on multicriterion decison-making fuzzy classification and demonstrated higher accuracy with feature selection [EAAO14]. Table 4.2 describes the DoS Dataset Features.

Table 4.2: DoS Attack Dataset Features

Feature	**Description**
Time	The time of the beginning of the session
Source IP	The Network Layer Source IP Address
Destination IP	The Network Layer Destination IP Address
Source Port	The transport Layer source Port
Destination Port	The Transport Layer Destination Port
Protocol	The protocol type (e.g. TCP or UDP)
Source bytes	Bytes sent from source IP to destination IP
Count	Number of connections to the same host as the current connection in the past five seconds
Different Services Rate	% of connections to different services
Same Services Rate	% of connections to the same Service
Destination IP SError Rate	% of connections to the current Destination IP and specified service that have an S0 error
Same Services Rate from different Source IP	% of connections to the same service to the destination IP from different Source IP
Different Services Rate from Same IP	% of connections having the same destination host to different services
Class	Attack Class Category: Benign or Attack

Table 4.3: XSS & SQL Attack Dataset Features

Feature Name	Description
Index	Content Location
Method	Method of Request
URL	Source URL
ContentLength	Length of the Response Body
ContentType	MIME type of the Response Body
Cookie	Content of the HTTP Cookie
Payload	Packet Payload
Host	Domain Name of the Server
Protocol	Protocol Type
Class	Benign or Attack

The attribute dataset for the XSS attack and the SQL attack is created from the host web logs. Various log files are merged like the apache access log and the cookie log. A python script is used to convert the web log collected from the host to the .CSV format and then the Python Weka wrapper code converts the .CSV file to .ARFF format(See 11). The feature selection has been done with the 'Principal Components'. This algorithm uses the 'Ranker Search algorithm' as the search method. Evaluation has been done on the full training data.

Evaluator: *weka.attributeSelection.PrincipalComponents -R 0.95 -A 5*

Search: *weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1*

Table 4.3 shows the structure of the XSS & SQL Attack dataset. The dataset has 10 attributes. Out of the 18 attributes of the original dataset, 9 attributes are chosen by the weka attribute evaluator. The 10th attribute is the 'Class' attribute which has any one of the values, 'Attack' or 'Benign'. This class attribute should be the last attribute in the training dataset file.

4.3.5 Forensic Analysis on The Grid with Weka

The data mining tool, Weka, can be invoked by each client so that the n-fold cross-validations for the classifier can be distributed on the parallel framework of the grid. We have utilized the Grid-Weka code from [oD15]. The individual iterations for the different folds are executed on different grid nodes. In this framework, some nodes are configured as the Weka Server(s) and some as the Weka Client(s). Weka runs on the Weka server and the Weka Client calls the Weka service on the Grid. Several classifiers can be trained in parallel by launching multiple Weka services from the Grid client. The client requests are handled by the Weka Server by calling the appropriate Weka functions.

WekaServer Package :

The WekaServer package has general purpose task executors and a Weka Explorer plugin which allows distributed k-fold cross-validation where each fold is converted to a separate task and sent to the server. The package can be installed through the package manager in Weka.

The following command launches the WekaServer:

```
java -Djava.awt.headless=true weka.Run WekaServer -host DPIServer -port 6714
```

For this purpose, GridWeka2.jar from [oD15] has been used. The following steps list the procedure to parallelize the classifier (See Procedure 12).

Procedure 12 Weka on Grid

Step 1: Weka-Servers List

Create a file, weka-servers.csv, a file containing the information about the weka servers. The file looks like this:

```
localhost,6714,-,-
DPIServer,6714,-,-
```

This tells Weka that there is a server running on localhost and another server running on DPIServer, and both listen on port 6714.

Step 2: Inform the client about the Weka servers

Copy weka-servers.csv to the directory from where the weka client is to be started(i.e. the GUI Chooser, the Explorer or the Experimenter).

Step 3:Start Weka on the server(s)

```
java -cp GridWeka2.jar weka.ucd.WekaServer 6714 10
```

The above command starts a server on port 6714 that allows up to 10 concurrent requests.

Step 4: Start GUIChooser on the client.

```
java -cp GridWeka2.jar weka.gui.GUIChooser
```

The above command starts the GUIChooser on the client. The GUIChooser extends javax.swing.JFrame. It lets the user choose the GUI which they want to run. The two important methods of this class are:

- showExplorer()

  This launches the Weka Explorer window as shown in Figure 4.5. The .arff file is loaded and further actions of Classification, Clustering, Feature Selection, Filtering etc are performed.

- showKnowledgeFlow()

  This method calls the GUI for the Weka KnowledgeFlow window.

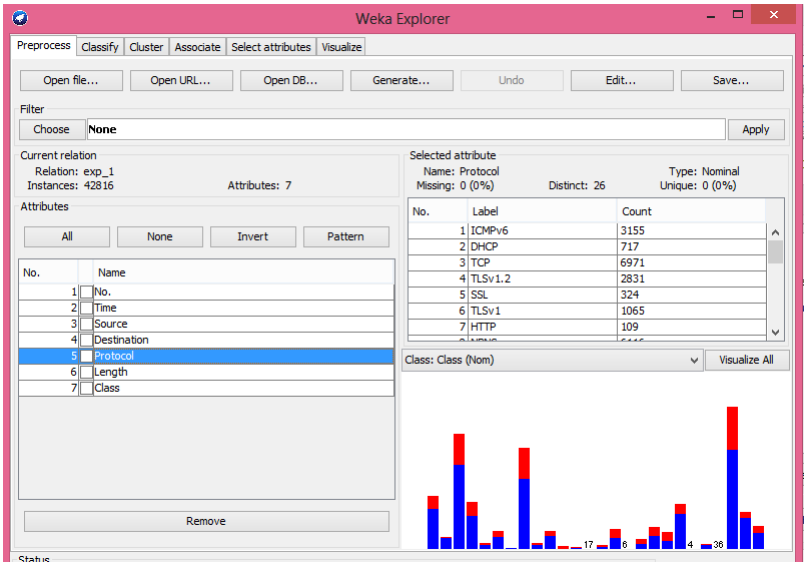

Figure 4.5: Weka Explorer GUI

Figure 4.6 is the screenshot of running the weka server on hosts, localhost and DPIServer(renamed asus).

```
D:\DPI>java -cp GridWeka2.jar weka.ucd.WekaServer 6714 10
Welcome to the GridWeka2 Server.
To get a short help, start with the -help option.
Allowing 10 concurrent requests.
Listening on port 6714
Wed Jul 15 16:00:32 IST 2015 Waiting for incoming connection...
Wed Jul 15 16:01:40 IST 2015 [0] Connection from /127.0.0.1:54244
Wed Jul 15 16:01:40 IST 2015 Waiting for incoming connection...
Wed Jul 15 16:01:40 IST 2015 [0] User - ok
Wed Jul 15 16:01:40 IST 2015 [1] Connection from /172.31.236.56:54245
Wed Jul 15 16:01:40 IST 2015 Waiting for incoming connection...
Wed Jul 15 16:01:40 IST 2015 [1] User - ok
Wed Jul 15 16:01:40 IST 2015 [2] Connection from /127.0.0.1:54246
Wed Jul 15 16:01:40 IST 2015 Waiting for incoming connection...
Wed Jul 15 16:01:40 IST 2015 [2] User - ok
Wed Jul 15 16:01:40 IST 2015 [3] Connection from /127.0.0.1:54247
Wed Jul 15 16:01:40 IST 2015 Waiting for incoming connection...
Wed Jul 15 16:01:40 IST 2015 [3] User - ok
Wed Jul 15 16:01:40 IST 2015 [4] Connection from /172.31.236.56:54248
Wed Jul 15 16:01:40 IST 2015 Waiting for incoming connection...
Wed Jul 15 16:01:40 IST 2015 [4] User - ok
Wed Jul 15 16:01:40 IST 2015 [5] Connection from /127.0.0.1:54249
Wed Jul 15 16:01:40 IST 2015 Waiting for incoming connection...
Wed Jul 15 16:01:40 IST 2015 [5] User - ok
```

Figure 4.6: Weka Server

Figure 4.7 is the screenshot when the client calls the GUIChooser, loads the .arff file and selects the 'Classify' tab from the Weka Explorer.

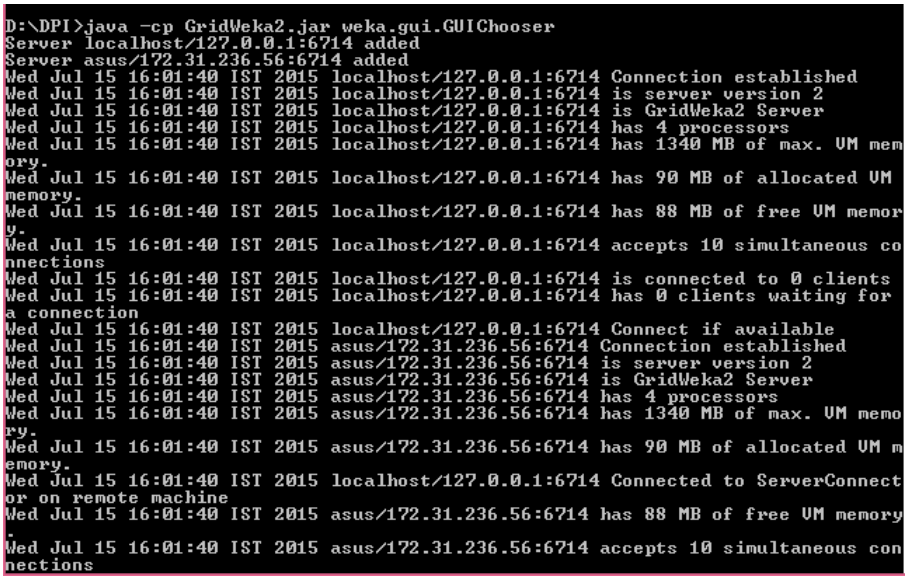

Figure 4.7: Weka Client

Procedure 13 lists the steps for training the classifier in Weka.

Procedure 13 Weka Statistical Modeling

Step 1: Data Loading (ARFF format)

1. Choose *Explorer* application
2. Get the input file in ARFF format(this file is created from the CSV file by executing the python wrapper code).

   The .arff file features correspond to the *weka.core.FastVector*. Each feature is contained in a *weka.core.Attribute* object.

Step 2: Build(Train) the statistical classifier

1. A training set of instances is required.
2. A classifier is to be selected from *weka.classifiers.Classifier*
3. The experimentation is performed with different classifiers before choosing the best classifier to create the model. The performance of each classifier(build-time and error-rate) is compared. The classifier with a reasonable model building time and highest accuracy is then selected. It is important to experiment with the different parameters specific to each classifier and evaluate the ones which give better performance.

Step 3: Test the classifier

1. After training the classifier and building the classifier model, to test the classifier, the testing dataset is fed to the weka evaluation module, *weka.classifiers.Evaluation*.

   The evaluation module outputs several statistics, summary statistics, confusion matrix etc.

Step 4: Analyze the Confusion-Matrix

1. The Confusion Matrix for each of the three types of attacks is analyzed
2. Output is the data adjust matrix.

Step 5: Calculate the following parameters

1. Precision/Detection Rate
2. Recall/TPR
3. FPR

Step 6: Calculate the Accuracy and F1 Score for each attack.

1. F1 Score : Also called the or F-Score or the F-Measure, it is the harmonic mean of the Recall(Sensitivity) and Precision.
2. Accuracy : The accuracy is computed by summing up all instances in the main diagonal(TP+TN) and dividing by total number of instances.

4.4 Summary

The chapter details the experimental setup and highlights its importance. Without a proper testbed, the realization of the solution would not have possible. The description of various tools and softwares used in the research is given in the chapter. To make the solution extensible, well-supported and affordable for all kinds of users, open-source tools have been used wherever possible. The implementation phase has been a spiral process. Each step of successful implementation of a module, has been followed by a careful review and matching with the objectives to ensure that the solution adheres to the requirements and is in accordance to the expected standards of evaluation. Several times, backtracking and re-implementation of the various modules has been done.

The chapter also discusses the hardware implementation as well as the software implementation of the framework.

Chapter 5

Results and Discussions

5.1 Introduction

The experiments have been conducted, tested and evaluated on two platforms: GPU and Grid. Section 5.2 and section 5.3 report and discuss results for these two categories of testing. The validation of the CUDA DPI engine on the GPU provides the results that indicate that the pattern-matching based DPI on the GPU delivers more than 14 times speedup as compared to the serial execution of the algorithm on just the CPU . The validation of the forensic analysis framework is done with Weka on the Grid setup. A statistical model is built with the labelled dataset and can then be used to build predictive models using Weka classifiers. The metrics for the evaluation of the classifier have been Detection Rate, Sensitivity, Specificity, Accuracy, F-Measure and ROC. Visualization is very important for the forensic analyst as it provides comprehensive results and gives a quick and fair insight to the performance. The execution time of the framework is evaluated in two different scenarios; grid setup and a non-distributed single-node setup.

Table 5.1: DPI traffic analysis

Filesize (Packet capture file)	Quadcore CPU	Nvidia GPU (UnOptimized)	Nvidia GPU (Optimized)
85MB	197.6ms	75.8ms	21.2ms
124MB	298.4ms	98.0ms	32.4ms
238MB	413.3ms	121.0ms	39.2ms
434MB	563.8ms	167.5ms	43.9ms
528MB	899.9ms	234.5ms	61.4ms
776MB	984.9ms	266.0ms	70.3ms
876MB	1197ms	354.6ms	84.3ms

5.2 Validation of the Results on GPU

The proposed algorithm is executed on a commodity graphics card equipped with the programmable NVIDIA GeForce GT 635M having 144 CUDA cores and a Graphics Clock upto 675 MHz which means it has high parallel computation power to perform the task of multiple pattern-matching in parallel thereby delivering a significant speedup as compared to the regular scenario where the intrusion detection is performed on a CPU. The GPU uses a unified clock instead of a shader clock which leads to higher efficiency. For measuring the data transfer time and the kernel execution time, Nvidia Profiler tool is used. Table 5.1 compares the execution time of the serial implementation on a quadcore Intel CPU with the parallel implementation of the Rabin-Karp pattern-matching algorithm on a multicore GPU with and without code optimization techniques.

It is observed that there is a good speedup gained with the optimized implementation on the GPU as compared to its serial implementation as shown in Figure 5.1. The average speedup observed for the optimized code GPU implementation is 12X. The best case speedup observed is 14X. The filesize of the .pcap file, the packet

capture file is a significant factor. The performance continues to demonstrate a good speedup but for very large sizes, it begins to plateau, because of latency and memory throughput. Figure 5.2 shows the GPU performance results for different packet capture file sizes.

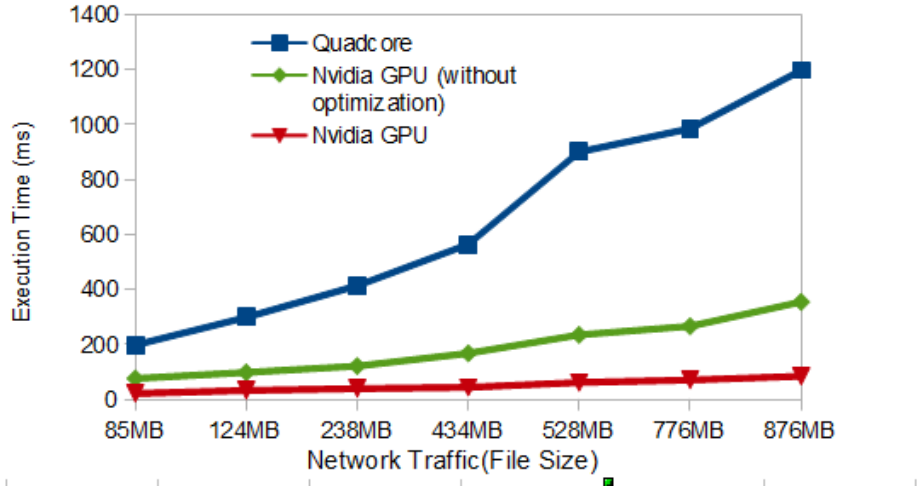

Figure 5.1: Performance Result Comparison

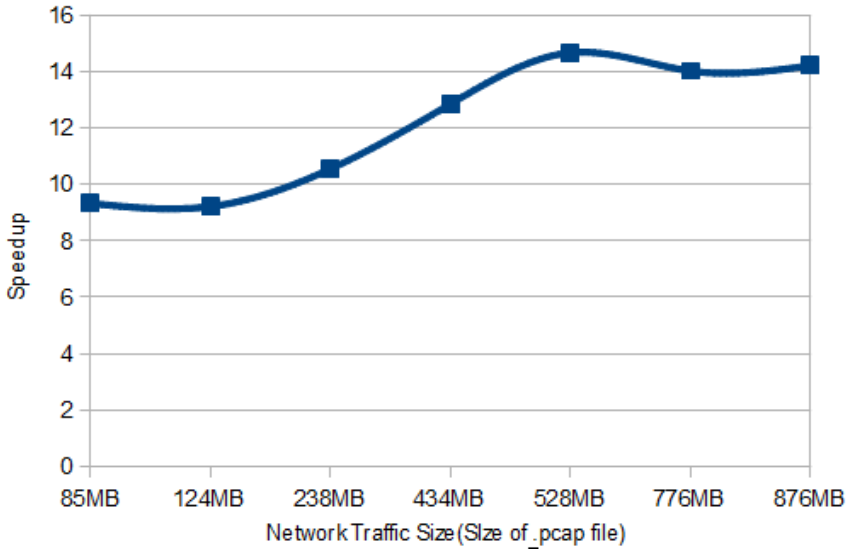

Figure 5.2: Speedup on Nvidia GPU

5.2.1 Code Optimization

- Memory Optimization The on-chip shared memory available for each multiprocessor is a great feature for writing a well optimized CUDA code. The shared memory is a small high bandwidth memory which is private to the threads in a block. The shared memory latency is 100x lower than the global memory latency. Storing frequently reused data to the shared memory can deliver substantial performance improvements [RGO+09], therefore the pattern and the hash table are stored in the shared memory to deliver the maximum performance. The data transfers between the CPU and GPU are done using the asynchronous variants for copying data (cudaMemcpyAsync()), which immediately return control to the host.

- Global Synchronization This is achieved by decomposition into multiple kernels, and this is a good technique because the hardware and software overhead of the kernel launch is very low.

5.3 Validation of the Forensic analysis Framework on Grid Infrastructure

For the purpose of the traffic identification task, the experimental setup captures packets from the academic network of the Thapar University. After the initial preprocessing and the Pattern-Matching, the dataset is staged on the data grid. Data mining algorithms are performed on the dataset on the grid. For the performance analysis of the statistical model built using Weka, the TP-Rate, FP-Rate,

F-Measure, Recall, Precision and ROC Area for each of the three attack categories under test, are calculated and then the average F-measure, recall, and precision values are calculated. The results are presented and discussed in the next section. The performance of the framework is evaluated on the grid by observing execution time for different dataset sizes.

5.3.1 Evaluation of the Statistical Model

Evaluating the performance of the data mining technique is a very important aspect of machine learning. It helps one to gain a better understanding of the technique which results in the refinement of the parameters, in the iterative process of the learning. Evaluation provides the prediction how well the model will work in the future. The different data mining components for the major data mining tasks of Feature Selection, Classification, Clustering and Association Rule Mining are Attribute Evaluator, Classifier, Clusterer and the Associator.
The classifier is the most important component for this framework as it builds the statistical model for the forensic analysis process. The following is the list of the methods for the performance evaluation of the classifier.

- Confusion Matrix
- ROC Curves
- K-fold Cross-Validation
- Leave One Out(LOO)
- Hold Out

Table 5.2: Confusion Matrix

Class	Classified as Benign	Classified as Attack
Benign	TN	FP
Attack	FN	TP

- Random Sub Sampling
- Bootstrap

In the research, the framework is evaluated with three methods: Confusion Matrices, K-fold Cross-Validation and ROC Curves

- **Confusion Matrix**

  The basic performance evaluation is done from the confusion matrix(Table 5.2. The confusion matrix also known as an error matrix or a contingency table, represents the visualization of the outcome of Weka analysis in this case of supervised learning.

  The fields of the confusion matrix are described as follows:

  – TP

    True Positive: The number of malicious packets, classified correctly as malicious

  – FN

    False Negative: The number of packets, classified incorrectly as normal

  – FP

    False Positive: The number of normal packets, classified incorrectly as malicious

– TN

True Negative: The number of normal packets, classified correctly as normal

- **K-fold Cross-Validation**

1. Partition randomly the available examples into k disjoint subsets.
2. Use one of the partitions as a testing set to evaluate a model generated by considering the other subsets as the training set.
3. Repeat this process with all subsets and average the obtained errors.

- **ROC Curve**

The accuracy of the analysis is evaluated using the Receiver Operating Characteristic (ROC) curve analysis [CE78]. In an ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between the two groups (attack/normal). A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test [MG93].

5.3.2 Experiments Results on a single host setup

The network traffic of the university is profiled. The protocol distribution profiling suggests that typically it is Web Traffic (Nearly 50% is HTTP and 50% of the rest is DNS(See Figure 3.6. The research considers the major categories of attacks for such a protocol profile. The attacks have been discussed in Section 1.1.2). The three types of attacks in this section considered for the research are as follows.

1. Denial-of-Service Attack(DoS)
2. SQL Injection Attack
3. Cross-Site Scripting Attack(XSS Scripting)

The following performance metrics are used for the evaluation of the solution[WFH10]:

- Accuracy

  $Accuracy = (TP + TN)/TP + FN + FP + TN;$

- Precision or Detection Rate

  $Precision = TP/(TP + FP);$

  Precision is the measure of exactness or quality.

- False Alarm or False Positive Rate or Fall-Out

  $FPR = FP/FP + TN;$

- Recall or Sensitivity

  This is the True Positive Rate, also called as Hit Rate, it is the measure of the completeness or quantity.

  $Recall = TP/(TP + FN);$

- Specificity or True Negative Rate

  $SPC = TN/(FP + TN)$;

Based on the above values, the F_1 Score is calculated, which is the harmonic mean of precision and recall.

$$F_1 = 2TP/(2TP + FP + FN) \tag{5.1}$$

The open source machine learning package, Weka(Waikato Environment for Knowledge Analysis is used for the statistical modeling . The latest stable version of the data mining software in Java, Weka 3.6.12 is used for the data mining tasks that contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization.

Different classifiers are experimented with, and their performance compared before the final selection of the classifier to be used for building the test model is done. In this research, four classfiers are experimented with, varying their execution with different parameters. Procedure 14 describes in detail the steps for building the statistical model using the Weka Classifiers.

Procedure 14 Weka Classifier Model

Step 1: Build Classifier Models

Execute weka classifiers one by one on the training dataset to build the model, and record the performance of each.

1. Naive Bayes: *weka.classifiers.bayes.NaiveBayes*
Naive Bayes algorithm is executed in the test mode with 10-fold cross validations.

2.Bayes Net:
weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 – -P 1 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator – -A 0.5

BayesNet algorithm is run with the wwo different algorithms for estimating the conditional probability tables of the network: *SimpleEstimator* and *K2 search* algorithm without using ADTree.
10-fold cross-validation is performed.

3. J48 (C4.5 Decision Tree): *weka.classifiers.trees.J48 -C 0.25 -M 2*
J48 is experimented is this study with the parameters:
confidenceFactor = 0.25; numFolds = 10; seed = 1; unpruned = True

4. Random Forest: *weka.classifiers.trees.RandomForest -I 100 -K 0 -S 1*
A Random forest of 100 trees created with the above algorithm.
Each tree is constructed while considering 3 random features.
The independently trained forests are stitched together with the combine() function from the randomForest package.
The Test mode was 10-fold cross validation.
The building time is quite long, but the accuracy is good.
The number of trees should be enough to stabilize the error, but not too much that it leads to an overfit. First, an odd value is chosen for this, ntree=501, so that ties can be broken, and then the random forest object is plotted which depicts the error convergence based on the OOB error.

Step 2: Select the classifier model

Following classifier metrics are considered for the selection of the best classifier model:
1. Training Time: It is the Total time taken by the classifier to build the model
2. Error Rate: The classifier can misclassify in which the class label of a test instance contradicts the class labels of other similar records in the training set.

$$ErrorRate = 1 - Accuracy \tag{5.2}$$

Performance of each classifier tested in the previous step is recorded in a table(See Table 5.10).

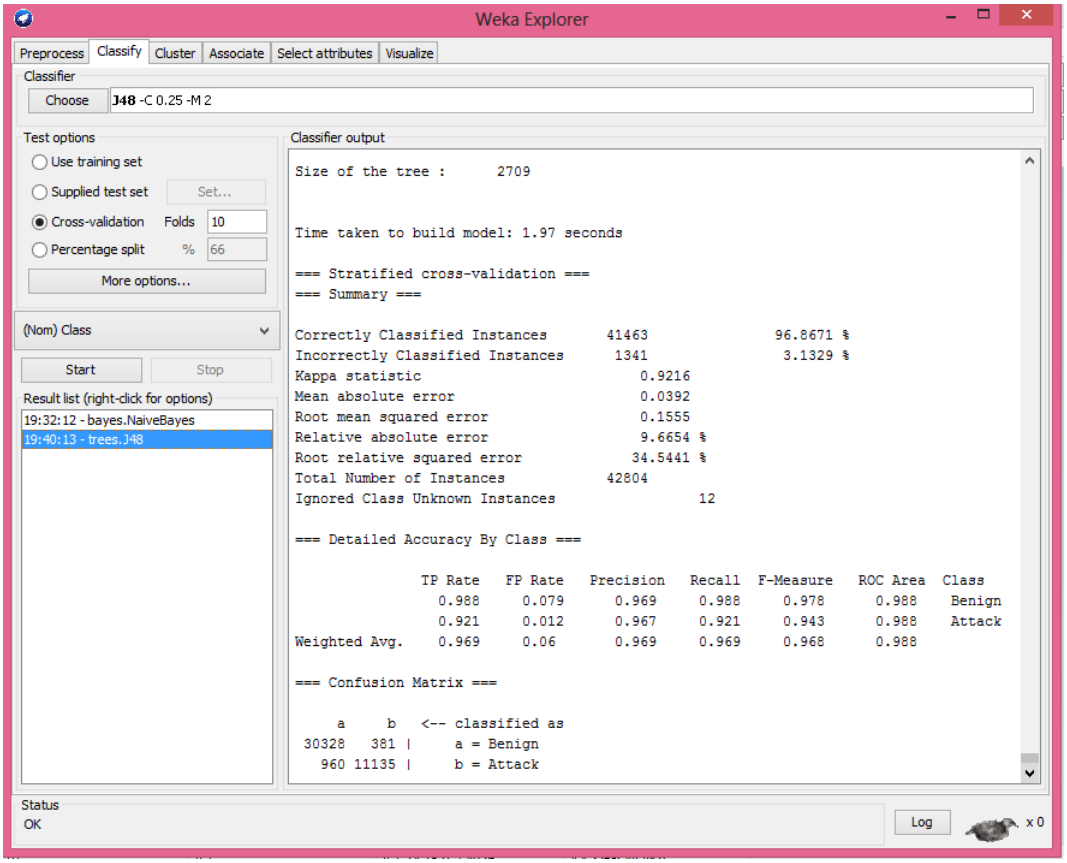

Figure 5.3: J48 Classifier

5.3.2.1 Results for DoS Attack Dataset

Table 5.3 shows the confusion matrix for the DoS Attack dataset.

Table 5.3: Confusion Matrix for Dos Attack Type

Class	Classified as Benign	Classified as Attack
Is Benign	30328	381
Is Attack	960	11135

Table 5.4 lists the accuracy characteristics of the classifier alongwith their description. Table 5.5 through Table 5.9 depict the detailed accuracy characteristics for the DoS Attack Dataset when classified with the four different classifiers.

Table 5.4: Accuracy Statistics

Metric	Description
Correctly Classified Instances	Instances those are correctly classified
Incorrectly Classified Instances	Instances those are incorrectly classified
Kappa statistic	Degree to which raters agree to assign qualitative data to categories
Mean absolute error	The average of absolute errors
Relative absolute error	The total absolute error
Root mean squared error	Difference between actual value and the predicted value
Root relative squared error	Normalized total square error

Table 5.5: J48: Accuracy Statistics-DoS Attack Dataset

333333 ++++

Metric	Result
Correctly Classified Instances	96.8671%
Incorrectly Classified Instances	3.1329%
Kappa statistic	0.9216
Mean absolute error	0.0392
Root mean squared error	0.1555
Relative absolute error	9.6654%
Root relative squared error	34.5441%

Table 5.6: J48-Detailed Accuracy by class

TP Rate	FP Rate	Precision	Recall	FMeasure	Class
0.988	0.079	0.969	0.988	0.978	Benign
0.921	0.012	0.967	0.921	0.943	Attack

The evaluation of the accuracy of the model is also presented visually using the Receiver Operating Characteristic (ROC) curve analysis in which the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between the two groups (attack/benign). The AUC for DoS is 98.82% and

Table 5.7: Bayes Net:Accuracy Statistics-DoS Attack Dataset

Correctly Classified Instances	84.693%
Incorrectly Classified Instances	15.307%
Kappa statistic	0.6171
Mean absolute error	0.1931
Root mean squared error	0.331
Relative absolute error	47.619%
Root relative squared error	73.521%

Table 5.8: Naive Bayes-Detailed Accuracy by class-DoS Attack Dataset

Correctly Classified Instances	79.43%
Incorrectly Classified Instances	20.58%
Kappa statistic	0.4405
Mean absolute error	0.26
Root mean squared error	0.376
Relative absolute error	64.13%
Root relative squared error	83.65%

Table 5.9: Random Tree-Detailed Accuracy by class-DoS Attack Dataset

Correctly Classified Instances	96.297%
Incorrectly Classified Instances	3.703%
Kappa statistic	0.9084
Mean absolute error	0.038
Root mean squared error	0.189
Relative absolute error	9.376%
Root relative squared error	42.182%

for XSS & SQL Dataset is 99.1%. The ROC curves for DoS, XSS & SQL Dataset, respectively, are all closer to the upper left corner which indicates higher accuracy. Therefore the proximity of the ROC curve to the upper left corner indicates that it is closer to perfect discrimination. The following figures depict the ROC curve for all the three test cases with the three different datasets.

ROC Curves (DoS Attack Dataset)

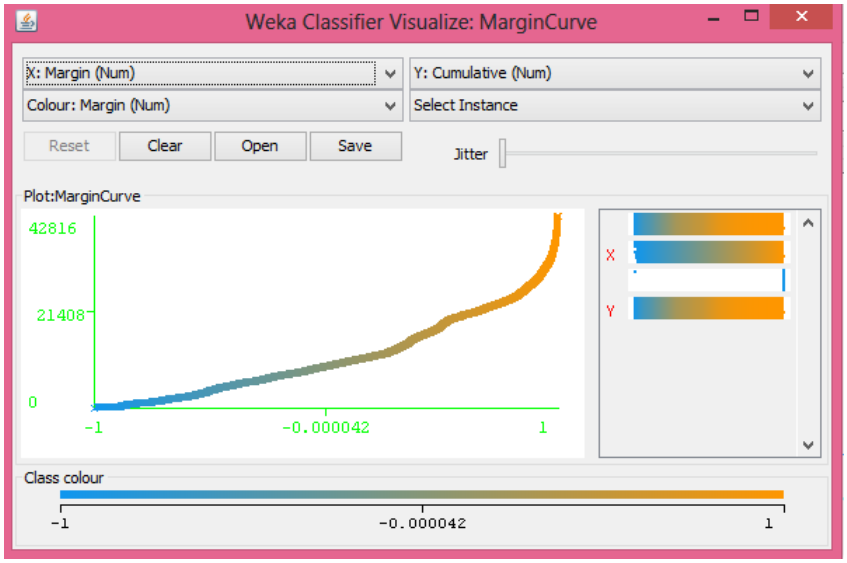

Figure 5.4: Margin Curve

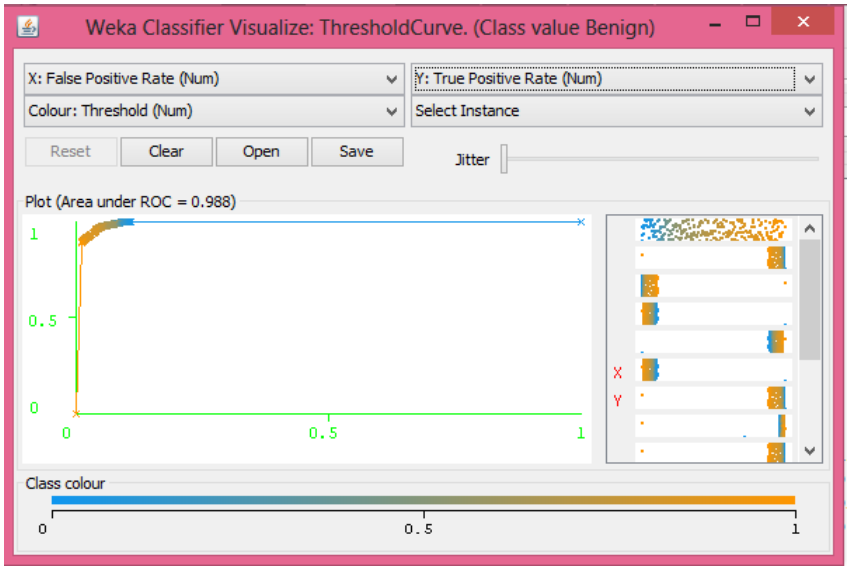

Figure 5.5: Threshold Curve(Benign)

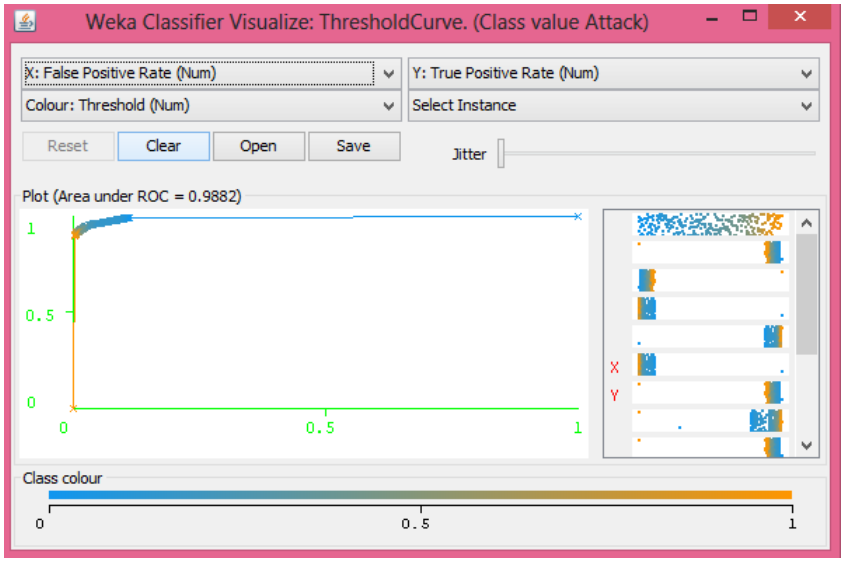

Figure 5.6: Threshold Curve(Attack)

After the observations from the table(Table 5.10), **the classifier model is chosen where training time is low yet accuracy is high(low error rate)**.

Once the training of the classifier is done on a single-host(non-grid) setup, the classifier training is repeated on the dataset on the Grid as shown in Section 4.3.5. It is observed that the error rate/accuracy statistics are same in both cases. The difference is in the model building time. In both cases the Naive Bayes algorithm builds the model fastest of all, but the error rate is high. The J48 algorithm is chosen for further analysis as it has a low error rate and a reasonable model building time in both the non-grid as well grid environment.

Table 5.10: Comparison of performance of Classifiers on Single-host

Classifier	Time taken to Build the model	Error Rate
Naive Bayes	0.59 seconds	20.57
Bayes Net	1.59 seconds	15.3
J48	2.48 seconds	3.1
Random Forest	25.42 seconds	3.7

Table 5.11: Confusion Matrix for XSS & SQL Attack Dataset

Class	Classified as Benign	Classified as Attack
Benign	204438	89
Attack	1242	18896

Table 5.12: Metrics for all Attack Datasets-J48 on single host

Measure	DoS Attack Dataset	XSS & SQL Attack Dataset
False Positive Rate(FPR)	3.06%	0.6%
TPR	96.69%	99.53%
Accuracy	96.87%	99.4%
Precision	92.06%	93.83%
Recall	96.69%	99.53%
True Negative Rate (TNR)	96.93%	99.39%
F-Measure	94.32%	96.59%

5.3.2.2 Results for XSS & SQL Attack Dataset

Table 5.11 shows the prediction results of the classifier in the form of a confusion matrix for the XSS Attack and SQL Attack Dataset.

There were 204527 number of benign instances, out of which 204438 were classified correctly but 89 instances were classified incorrectly. The total number of 'attack' instances were 20138, out of which 18896 were correctly classified and 1242 were not classified correctly. 18 instances were ignored because of unknown class.

Table 5.12 summarizes the results for the classifier metrics for the J48 classifier in the non-Grid scenario. The results are shown for both datasets, DoS Attack Dataset and the XSS &SQL Attack Dataset.

5.3.3 Results on Grid

Procedure 12 discussed the steps to request the weka classifier on the grid. When the J48 classifier is selected to be executed by a client on the grid for each of

Table 5.13: Metrics for all Attack Datasets-J48 on Grid

Measure	DoS Attack Dataset	XSS & SQL Attack Dataset
False Positive Rate(FPR)	3.06%	0.6%
TPR	96.69%	99.53%
Accuracy	96.87%	99.4%
Precision	92.06%	93.83%
Recall	96.69%	99.53%
True Negative Rate (TNR)	96.93%	99.39%
F-Measure	94.32%	96.59%

Table 5.14: Comparison of performance of Classifiers on the Grid

Classifier	Model Training Time	Error Rate
Naive Bayes	0.18 seconds	20.57
Bayes Net	0.63 seconds	15.3
J48	1.3 seconds	3.1
Random Forest	1.78 seconds	3.7

the datasets, we observe the same results for the FPR, TPR, Accuracy, Precision, Recall, TNR and F-Measure as we observe on the non-grid setup. The error rate is also same. Table 5.12 summarizes the results for the two datasets when the classifier is executed on a Grid. The significant difference is in the build time of the models on the grid, as shown in Table 5.14.

5.3.4 Comparison of the Results on a single host setup and Grid Setup

J48 technique is a good algorithm for performing this task. As we have seen that in both cases(Grid and Non-Grid), the error rate was lowest and the build time also reasonable. Though Naive Bayes was the fastest, it had high error rate.

5.3.4.1 Accuracy Evaluation

The accuracy of the data mining Java web service when compared to the accuracy of the data mining algorithm on single host are comparable. Both implementations identify the same number of correctly classified and incorrectly classified instances. The confusion matrices are same, therefore the metrics like precision, recall and Accuracy are also same.

5.3.4.2 Execution Time Evaluation

The amount of time that the Web services require to perform the data mining classification process on a Grid is compared to the execution time of the classifier on a non-grid environment. For all the classifiers the execution on the grid is several times faster as compared to the execution on a single host setup. Surprisingly, random forest was very fast on the grid(more than 100 times faster). The execution times for each run may vary a little bit depending on other services being run on the host/client machine.

Figure 5.7 demonstrates the benefit of using the Grid Computing infrastructure for processing the dataset using distributed web services. The speedup is manifold.

5.3.5 Summary of results of the forensic analysis engine

The performance of the forensic analysis engine is evaluated through the standard metrics for the statistical algorithms which are discussed in section 5.3.2. The metrics recall and precision are first calculated for each of the three attack categories under test, and then the average recall and precision values are calculated. The results are presented and discussed in this chapter.

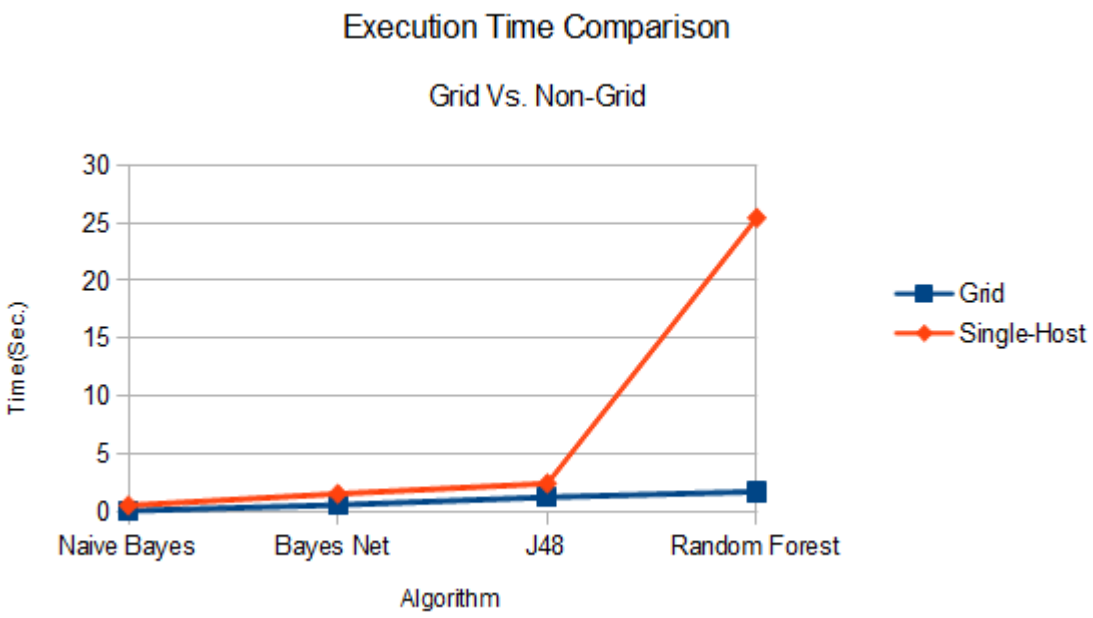

Figure 5.7: Comparison of Execution Time on Grid and Non-Grid Platform

It is observed that the False Positive Rate(FPR) and the False Negative Rate(FNR) are quite low whereas the Precision and Recall are nearly perfect. A perfectly tuned intrusion detection based forensic analysis system should have a rate of TPR = 100% and FPR = 0%, which means that every attacks have raised alerts and that the normal(benign) traffic has never raised any alert. But practically, this has never been seen. In actual conditions with a high amount of network traffic, many False Positive alerts are generally raised up by the IDS. False Positive alerts mean that an alert was raised even in the absence of a true attack. It is debatable to arrive at an acceptable TPR and FPR rate. However, at the Symantec Website [Kev10], it has been discussed that an acceptable level of TPR should be at least 60% . In this case it is much better than the generally acceptable rate. The proposed model is considered to be performing well with the TP (true positive) rate

at 96.69% for DoS, 99.53% for XSS-SQL. FP (false positive) rate was generated at a level of 3.06% for DoS attack category and for XSS it was merely 0.6%. The high recall and precision values(more than 90%) indicate the better suitability of the technique to the framework, as compared to the other solutions. F-measure, the measure of the test's accuracy, is also nearly 95% in both cases which is an indicator of the efficacy of the model.

The accuracy of the framework is the same in both grid and non-grid platforms, 96.87% and 99.4% for the two attack datasets. The notable difference is in the execution time. In all cases the speedup when the algorithms are executed on the grid is good. The highest speed gain is noticed in case of Random forest algorithm. The algorithm in this case is made from 100 trees. Three random features are considered in the construction of each tree. Such an algorithm is quite suitable for implementation on a parallel platform like grid. Though, J48 has been chosen in this framework, Random forest also is a good candidate as it also has a low error rate(though a little higher than J48, still much lesser than Naive Bayes and Bayes Net).

Chapter 6

Conclusions and Future Work

This chapter summarizes the results and contributions of the research work. The chapter also highlights future research direction based on the results obtained.

6.1 Conclusions

The thesis results in the realization of a framework for forensic analysis of large volume of network traffic captured at wirespeed. The main objective has been to perform the forensic analysis at an highly accelerated speed, so that the network security system becomes almost proactive while also being reactive. The early analysis helps in updating the signature repository soon after an attack has occurred so the continuation of the attack can be controlled.

The efforts to involve the use of concepts for high performance computing have delivered good results, much beyond the expectations. The parallel implementation on a multicore GPU of the pattern-matching algorithm achieved a speedup of upto 14X. The performance begins to show a plateau effect when the traffic

size increases beyond a limit as the GPU is limited with its memory, nonetheless the system delivers optimal performance at increasing traffic. Much better GPUs Peak performance has been achieved by removing dependencies on the global memory and improving the memory coalescing. The Nvidia Profiler [NVI14b] and the CUDA Occupancy Calculator[NVI14a] have been greatly helpful in profiling the application and discovering the bottlenecks. The satisfactory speedup gains motivated to improve the code further by even more optimization of the code in the area of memory management. The important considerations in the work have been :

1. Selection of an appropriate candidate for the desired results on the GPU is very important. Not every serial code implemented on the GPU gives better performance.
2. Simple conversion of a serial code to the parallel code will not result in the desired improvement. Implementing proper CUDA code optimization results in efficient use of the GPU.

The results in the thesis indicate that definitely DPI performed on a GPU yields excellent performance especially when the underlying algorithm is carefully selected and tuned. The research on network security indicates that Deep Packet Inspection (DPI) based IDS has a better capability in securing networks from the new and emerging threats. DPI based forensic analysis of network traffic can carry out in-depth and unhurried analysis of the attacks and help to design a stronger security system, but is quite resource intensive. It calls for a lot of computational power as well as storage space(both short term and long term). Industry giants are successfully embracing web services and utilizing the Grid abilities, one recent example is the growing popularity of the Amazon Web Services being offered by

Amazon [Ama14][Raz13]. This thesis proposes the use of the high computational abilities of Grid infrastructure for performing the task in a more cost effective, fast and reliable way. The development of the grid application using WSRF maximises the potential of the Grid capabilities, enables it to be loosely coupled and scale to arbitrary size. Amongst the various WSRF implementations and the emerging implementations like RESTful WS implementations, GT4 Java WS Core is a good choice as it is established, well supported, and has a proven good performance.

6.1.1 Contributions

Major contribution of thesis has been highlighted below:

- An extensive review of literature provides insight into the current intrusion detection and prevention techniques, deep packet inspection approaches and network forensics techniques. A review of the commercial and open-source tools available for these is included in the thesis. This forms a good ground work for future research.

- A network forensics framework has been proposed, designed, implemented and validated which can perform forensic analysis of large volume of network traffic captured at wirespeed.

- An important contribution has been to perform the forensic analysis at an highly accelerated speed, so that the network security system becomes almost proactive while also being reactive.

- The proposed model is capable of performing Rabin-Karp multiple pattern matching algorithm on a GPU using CUDA. The efforts to involve the use of

concepts for high performance computing have delivered good results, much beyond the expectations. The parallel implementation on a multicore GPU of the pattern-matching algorithm achieved a speedup of upto 14X.

- The thesis proposes the use of the high computational abilities of Grid infrastructure for performing the task in a more cost effective, fast and reliable way. The development of the grid application using WSRF maximises the potential of the Grid capabilities, enables it to be loosely coupled and scale to arbitrary size.

- This solution has been implemented using GT4 Java WS Core components which makes it highly scalable and portable. Amongst the various WSRF implementations and the emerging implementations like RESTful WS implementations, GT4 Java WS Core is a good choice as it is established, well supported, and has a proven good performance.

- The system is able to identify intrusions and record them in an alerts repository.

- The evaluation is performed using standard IDS evaluation metrics like detection rate, Precision and Recall. The development of the grid application using WSRF maximizes the potential of the Grid capabilities, enable it to be loosely coupled and scale to arbitrary size. Amongst the various WSRF implementations and the emerging implementations like RESTful WS implementations, GT4 Java WS Core is a good choice as it is established, well supported, and has a proven good performance. Results show that proposed model is able to speedup upto 12x when implemented on a GPU. The per-

formance continues to demonstrate a good speedup but for very large sizes, it begins to plateau, because of latency and memory throughput

- The framework is developed, while keeping in mind even the user on the lower end of the user spectrum. level of end user in mind. The aim is to make the framework available to the user/organization who need a strong and the best network security system but unlike a large organization, cannot afford to invest in a custom-built DPI appliance or any other hardware appliance. Most tools and technologies used in the design and implementation are free and open-source.

6.1.2 Limitations

The main limitations of this work are as listed below:

- Partially open-source

  Despite the best intentions to design the system based on only open-source tools and technologies, there were limitations owing to the lack of sufficient support and documentation of some tools.

- Inability to inspect ciphered payload

  The framework works well for all network traffic, except encrypted data. Though, if a continuously encrypted session is noticed, it raises a flag and such traffic can be flagged under the 'Suspicious Connection' category.

- Under-utilization of the Grid Infrastructure

  The research work has not been able to exploit the grid to its full potential because of several constraints like limitations in interoperability.

6.2 Future Work

This section briefly discusses the future directions, to further improve/apply the research. The future work based upon the results obtained and the limitations of this work.

- **Completely Open-Source Framework** : Most of the tools used in the research have been open-source except for CUDA. The open-source API for GPGPU, OpenCL, would be considered for future work, so that the entire work becomes open-source and not vendor-locked. The other major advantages of using OpenCL are:

  1. Much wider range of hardware and platform support. Apart from NVIDIA, it equally supports AMD and Intel GPUs.

  It can also be used on iPhones and Android Phones.

  2. Synchronisation over multiple devices is possible.

  Since for the research work, there was the availability of NVIDIA GPUS, working with CUDA was considered as opposed to OpenCL because CUDA is the native API for NVIDIA GPUs. The other reasons are:

  1. NVIDIA is yet to release public drivers to support OpenCL versions.

  2. For the evaluation of the work, good debugging and profiling tools are needed which are readily provided by CUDA. CUDA comes with excellent debugging and profiling tools(NVIDIA Profiler).

  3. CUDA is very well documented.

4. Easier to learn, it uses C-like or C++ like language for writing the kernel code.

- **GPU-Grid** : To ensure higher speedups even with highly increasing traffic size, future efforts will be to utilize the power of grid computing and perform the forensic analysis on a GPU-grid. If all the phases of the framework implementation, can be performed on a grid of GPUs, the acceleration can be impressive. The other benefits of the Grid are a big bonus.

- **Wireless forensic analysis :** Future work could include, wireless forensic analysis. Nowadays wireless networks are being so popular. Great apps for mobile phones and tablets and hand-held devices have led to the manifold increase of wireless traffic. Wireless computer networks pose numerous security challenges as identified in [CLM09].

- **Social Network Forensics :** Social Networking has gained an explosive growth in the recent times. Twitter,Facebook and Whatsapp have made astronomical proliferation in usage and proportional is the increase in the security threats. Stalking, Trolling, identity theft and espionage are a few of the risks associated with such sites and applications. Criminals use social networks for perpetuating not only cybercrime but heinous crimes like murder and blackmailing. The increasing number of cases with an ever-growing amount of data calls for specialized Social Forensic Analysis research.

6.3 Summary

The experimental evaluation has been done by considering datasets collected from the university-vide network on different days. From the results, it has been indicated that the convergence of web services, grid and GPU computing provides an adequate solution to the problems of classical DPI techniques and the mostly manual forensic analysis task. The key attainment is that in-time alerts lead to timely intervention and prevent similar future attacks early-on.

References

[AC75] A. V. Aho and M. J. Corasick. *Efficient string matching: An aid to bibliographic search.* Communications of the ACM, 18(6):pp. 333-340, June 1975. 14, 35, 91

[AGJT03] Deb Agarwal, Jose Maria Gonzalez, Goujun Jin, and Brian Tierney. *An Infrastructure for Passive Network Monitoring of Application Data Streams.* Proceedings of the 2003 Passive and Active Monitoring Workshop, 2003. 15

[AIH04] Globus Alliance, IBM, and HP. *The WS-Resource Framework.* http://www.globus.org/wsrf/, 2004. 58

[AL05] M. Attig and J. W. Lockwood. *A framework for rule processing in reconfigurable network systems.* Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), April 2005. 34

[All14] Globus Alliance. *The Globus toolkit.* http://www.globus.org/tookit/, January 2014. 59

[Ama14] Amazon. *Amazon Web Services*. http://aws.amazon.com, September 2014. 148

[Apa14] Apache. *Apache AXIS*. ws.apache.org/axis/, January 2014. 104

[Ban05] T. Banks. *OASIS WSRF Primer. Committee Draft 01*. docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf, November 2005. 103

[BC00] H. Burch and B. Cheswick. *Tracing anonymous packets to their approximate source*. Proceedings of the USENIX Large Installation Systems Administration Conference, pp. 319-327, December 2000. 49

[Blo70] B. H. Bloom. *Space/time trade-offs in hash coding with allowable errors*. Communications of the ACM, 13(7):pp. 422-426, January 1970. 34

[BM77] R.S. Boyer and J. Strother Moore. *A fast string searching algorithm*. Communication of ACM, 20(10):pp. 762-772, October 1977. 35, 91

[Bro15] Bro.org. The bro network security monitor. *Retrieved 15 April 2015, from http://www.bro.org*, 2015. 28, 32

[BROS12] R. Bragg, M. Rhodes-Ouslty, and K. Strassberg. *Network Security: Complete Reference*. Tata McGraw Hill, August 2012. 11

[BTFF03] I Buck, J. Sugerman P. Hanrahan M. Houston T. Foley, D. Horn, and K. Fatahalian. *BrookGPU*. http://graphics.stanford.edu/projects/brookgpu/, May 2003. 54

[Can98] James Cannady. Artificial neural networks for misuse detection. In *National information systems security conference*, pages 368–81, 1998. 29

[Car03] B. Carrier. *Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layters.* International Journal of Digital Evidence, 1(4), January 2003. 43, 44

[CCU07] J. N. Chuang, L. Z. Chen, and P. Ungsunan. *A fast multi-pattern matching algorithm for deep packet inspection on a network processor.* International Conference on Parallel Processing (ICPP 2007), September 2007. 32

[CD07] H.L. Christopher and Y. L. Dpico. *A high speed deep packet inspection engine using compact finite automatas, Orlando, Florida.* www.bitcricket.com., December 2007. 33

[CE78] Metz CE. *Basic principles of ROC analysis.* Seminars in Nuclear Medicine 8:283-298, 1978. 132

[Cea04] JBD. Cabrera and et. al. *On the statistical distribution of processing times in network intrusion detection.* Proceedings of the 43rd IEEE Conference on Decision and Control, volume 1: pp. 75-80, December 2004. 90

[Cha07] N. Chawla. *Grid computing, Developer IQ.* Techmedia Publications, 6(1), January 2007. 18

[CL05] Z. Chen and C. Lin. *Antiworm NPU Based Parallel Bloom Filters For*

TCP/IP Content Processing in Giga-Ethernet Lan. IEEE Proceedings of Local Computer Network, The First Workshop on Network Security(WoNS), November 2005. 32

[CLM09] Hakima Chaouchi and Maryline Laurent-Maknavicius. *Vulnerabilities of Wired and Wireless Networks.* 2009. 152

[Cro06] D. Crockford. *The application json media type for javascript object notation (JSON).* https://tools.ietf.org/html/rfc4627, July 2006. 63

[CS01] Earl Carter and Rick Stiffler. Intrusion detection:cisco ids overview. *Retrieved 14 April 2015, from http://www.ciscopress.com*, 2001. 28

[CS05] Y. H. Cho and W. H. M. Smith. *Fast reconfiguring deep packet filter for 1+ gigabit network.* Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05), January 2005. 40

[CS06] P. Cheeseman and J. Stutz. *Bayesian classification (Auto Class): theory and results in Advances in Knowledge Discovery and Data Mining.* The AAAI Press, edited by U.M. Fayyad et al., California: pp. 61-83, 2006. 49

[CSL+11] Jose Juan Mijares Chan, Bhanu P Sharma, Jiaqing Lv, Gael Thomas, Ruppa Krishnamachary Thulasiram, and Parimala Thulasiraman. True random number generator using gpus and histogram equalization techniques. In *High Performance Computing and Communica-*

tions (HPCC), 2011 IEEE 13th International Conference on, pages 161–170. IEEE, 2011. 86

[CSM01] C. J. Coit, S. Staniford, and J. McAlerney. *Towards faster string matching for intrusion detection or exceeding the speed of snort.* DARPA Information Survivability Conference & Exposition II, Vol. 1, pp. 367-373, June 2001. 36

[CUD13] CUDA. *Parallel Computing Platform.* http://www.nvidia.in/object/cuda_home_new.html, September 2013. 90

[CZL08] Pautasso Cesare, Olaf Zimmermann, and Frank Leymann. *Restful web services vs. big'web services: making the right architectural decision.* Proceedings of the ACM 17th international conference on World Wide Web., April 2008. 63

[Den87] Dorothy E. Denning. *An Intrusion-Detection Model.* IEEE Transactions on Software Engineering, Vol. SE-13,No. 2, feb 1987. 49

[DFM+06] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. *Dynamic application-layer protocol analysis for network intrusion detection.* Proc. USENIX Security Symposium, January 2006. 32

[DKSL03] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, and John Lockwood. *Deep packet inspection using parallel bloom filters.* Proceedings of the 11 th Symposium on High Performance Interconnects (HOTI03), IEEE Computer Society, 2003. 34, 39

[DL06] S. Dharmapurikar and J. W. Lockwood. *Fast and scalable pattern*

matching for network intrusion detection systems. IEEE Jounal on Selected Areas in Communications, 24(10): pp.1781-1792, October 2006. 33, 90

[EAAO14] El-Sayed M El-Alfy and Feras N Al-Obeidat. A multicriterion fuzzy classification method with greedy attribute selection for anomaly-based intrusion detection. *Procedia Computer Science*, 34:55–62, 2014. 118

[EAAO15] El-Sayed M El-Alfy and Feras N Al-Obeidat. Detecting cyber-attacks on wireless mobile networks using multicriterion fuzzy classifier with genetic attribute selection. *Mobile Information Systems*, 2015. 30

[Fai15] Fail2ban. Main page. *Retrieved 14 April 2015, from http://www.fail2ban.org*, 2015. 28

[Fie00] R. Fielding. *Architectural styles and the design of network-based software architectures.* Ph.D. dissertation, University of California, Irvine, 2000. 63

[FK03] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, December 2003. 17

[FRP] Gilberto Fernandes, Joel J.P.C. Rodrigues, and Mario Lemes Proenca. *Autonomous Profile-based Anomaly Detection System using Principal Component Analysis and Flow Analysis.* 49

[Fu92] LiMin Fu. A neural network model for learning rule-based systems.

In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, pages 343–348. IEEE, 1992. 30

[FV02] M. Fisk and G. Varghese. *An analysis of fast string matching applied to content-based forwarding and intrusion detection.* Technical Report CS2001- 0670, University of California at San Diego, January 2002. 36

[Gar14] S. Garfinkel. *Network forensics: tapping the Internet.* http://www.oreillynet.com/pub/a/network/2002/04/26/nettap.html, September 2014. 45

[GBB] Prasanta Gagoi, Bhogeshwar Borah, and Dhruba K. Bhattachraya. *Network Anomaly Detection using Unsupervised Model.* 49

[GBS14] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. Malware analysis and classification: A survey. *Journal of Information Security*, 2014, 2014. 29

[Gea08] V. Giorgos and et. al. *Gnort: High performance network intrusion detection using graphics processors.* In Recent Advances in Intrusion Detection, Springer Berlin Heidelberg, 40(5):pp. 116-134, January 2008. 53

[GPKB12] J. Ghorpade, J. Parande, M. Kulkarni, and A. Bawaskar. *Gpgpu processing in cuda architecture.* arXiv preprint arXiv:1202.4347, 2012. 50

[Gre06] M. Gregg. *Certified Ethical Hacker Exam Prep.* Pearson IT Certification, April 2006. 4, 9

[HCC05] N.F. Huang, K.B. Chen, and W.E. Chen. *Fast and scalable multi-tcam classification engine for wide policy table lookup.* Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA05), January 2005. 39

[Hea08] N.F. Huang and et. al. *A gpu-based multiple-pattern matching algorithm for network intrusion detection systems.* 22nd IEEE International Conference on Advanced Information Networking and Applications-Workshops, pp. 62-67, April 2008. 53

[HLL+11] Che-Lun Hung, Yaw-Ling Li, Kuan-Ching Li, Hsiao-Hsi Wang, and Shih-Wei Guo. *Efficient GPGPU-based parallel packet classification.* IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom): pp. 1367-1374, January 2011. 53

[HLYH11] Changjun Han, Yi Lv, Dan Yang, and Yu Hao. An intrusion detection system based on neural network. In *Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on*, pages 2018–2021. IEEE, 2011. 29, 31

[HMB+12] Mohammad Sazzadul Hoque, Md Mukit, Md Bikas, Abu Naser, et al. An implementation of intrusion detection system using genetic algorithm. *arXiv preprint arXiv:1204.1336*, 2012. 30, 31

[HPDA05] T. Heinis, C. Pautasso, O. Deak, and G. Alonso. *Publishing Persistent Grid Computations as WS Resources*. Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing, Melbourne, Australia, December 2005. 62

[Hua09] et al. Huang, Shihong. *Remote computing resource management from mobile devices by utilising WSRF.* International Journal of Computer Aided Engineering and Technology, 2(2-3):pp.199-217, December 2009. 62

[HWMB04] M. Humphrey, G. Wasson, M. Morgan, and N. Beekwilder. *An Early Evaluation of WSRF and WS-Notification via WSRF.NET.* Grid Computing Workshop associated with Supercomputing, Pittsburgh, PA, USA, January 2004. 61

[HWMB05] M. Humphrey, G. Wasson, M. Morgan, and N. Beekwilder. *State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations*. Proceedings of the IEEE International Symposium on High Performance Distributed Computing, Research Triangle Park, NC, USA, 2005. 61, 62

[Ibr10] Laheeb Mohammad Ibrahim. Anomaly network intrusion detection system based on distributed time-delay neural network (dtdnn). *Journal of Engineering Science and Technology*, 5(4):457–471, 2010. 29, 31

[iL13] iXBT Labs. *Computer Hardware In Detail*. http://ixbtlabs.com/articles3/video/cuda-1-p1.html, September 2013. 50

[IPS08] IPSoft. *Resources, Articles, Green Computing.* www.ipsoft.com/us/component/content/article/146, January 2008. 18

[IST04] ISTS. *Law enforcement tools and technologies for investigating cyber attacks: Gap analysis report.* http://www.ists.dartmouth.edu, February 2004. 47

[JB06] Nigel Jacob and Carla E. Brodley. *Offloading IDS Computation to the GPU.* ACSAC, p. 371-380, December 2006. 53, 67, 91

[JBFT05] B. Jacob, M. Brown, K. Fukui, and N. Trivedi. *Introduction to Grid Computing.* IBM Redbooks, December 2005. 18

[JDKV12] S. Jaewoong, A. Dasgupta, H. Kim, and R. Vuduc. *A performance analysis framework for identifying potential benefits in GPGPU applications.* ACM SIGPLAN Notices, 47(8):pp. 11-22, January 2012. 53

[JDt15] KIAM JINR (Dubna) and SINP MSU team. *Report No.2 on Globus Toolkit 4 evaluation.* theory.sinp.msu.ru/dokuwiki/lib/exe/fetch.php?cache.=cache&media=egee:gt4:gt4eval_report2.pdf, July 2015. 62

[JF04] J. Joseph and C. Fellenstein. *Grid Computing.* Prentice Hall/IBM Press, January 2004. 18

[JGS10] E. John, D. Gohara, and G. Shi. *OpenCL: A parallel programming standard for heterogeneous computing systems.* Computing in science & engineering, 12(1-3):pp. 66-73, May 2010. 54

[JIKK13] N. Jeyanthi, N.Ch.S.N Iyengar, P C Mogan Kumar, and A. Kannammal. *An enhanced entropy approach to detect and prevent DDoS in cloud environment.* International Journal of Communication Networks and Information Security (IJCNIS), 5(2):pp. 110-119, April 2013. 91

[KCG+05] Sven Krasser, Gregory Conti, Julian Grizzard, Jeff Gribschaw, and Henry Owen. Real-time and forensic network data analysis using animated and coordinated visualization. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 42–49. IEEE, 2005. 46

[Kev10] Timm Kevin. *Strategies to Reduce False Positives and False Negatives in NIDS. Symantec. [Online] 03 11 2010. [Accessed: 14 02 2015] http://www.symantec.com/connect/articles/strategies-reduce-false-positives-and-false-negatives-nids.* Symantec, 2010. 144

[KGJE11] Adam Kiezund, Philip J. Guo, Karthick Jayaraman, and Michael D. Ernst. Automatic creation of sql injection and cross site scripting attacks. 2011. 5

[KK06] T. Kocak and I. Kaya. *Low-power bloom filter architecture for deep packet inspection.* IEEE COMMUNICATIONS LETTERS, 10(3):pp. 210-212, March 2006. 39

[KMP77] E. D. Knuth, J. H. Morris, and V. R. Pratt. *Fast pattern matching in strings.* SIAM journal on computing, 6(2): pp. 323-350., March 1977. 36

[KR87] R.M. Karp and M. O. Rabin. *Efficient randomized pattern-matching algorithms.* IBM J.Res. Dev, 31(2): pp. 249-260, March 1987. 36, 91, 92, 93

[KS14] Sanmeet Kaur and Maninder Singh. *A Newtork Security Model for Attack Signature Generation,Tracking and Analysis.* Thesis submitted to Thapar University,Patiala, 2014. 115

[KSH04] F. Kayvon, J. Sugerman, and P. Hanrahan. *Understanding the efficiency of GPU algorithms for matrix-matrix multiplication.* Proceedings of the ACM SIGGRAPH/ EUROGRAPHICS conference on Graphics hardware, pp. 133-137, January 2004. 53

[KT11] R. Kubert and H. Thai. *A performance comparison of four WSRF implementations.* International Journal of Electronic Business (IJEB), 9(5): pp: 499-515, July 2011. 63

[KW03] R. Kuehr and E. Williams. *Computers and the Environment:Understanding and Managing their impacts.* Kluwer Academic Publishers, Eco-Efficiency in Industry and Science Series, October 2003. 19

[KWWW94] S.C. Kendall, J. Waldo, A. Wollrath, and G. Wyant. *A Note on Distributed Computing.* Sun Microsystems, TR-94-29, January 1994. 57

[LB05] M. Li and M. Baker. *The Grid: Core Technologies.* John Wiley and Sons, 2005. 59

[Le02] W. Lee and et.al. *Lecture notes in computer science: Performance adaptation in real-time intrusion detection systems.* Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID 2002), October 2002. 35

[Lea08] C.H. Lin and et. al. *Accelerating pattern matching using a novel parallel algorithm on gpus.* IEEE Transactions on Computers, 62(10): pp.1906-1916, April 2008. 53

[Lit04] WSRF Lite. *Perl Grid Services.* http://www.sve.man.ac.uk/Research/AtoZ/ILCT, January 2004. 61

[LZ04] Bruce B. Lowekamp and Marcia Zangrilli. *Using Passive Traces of Application Traffic in a network Monitoring system.* IEEE Computer Society 2004, 2004. 15

[Mao10] Yuxin Mao. A semantic-based intrusion detection framework for wireless sensor network. In *Networked Computing (INC), 2010 6th International Conference on*, pages 1–5. IEEE, 2010. 31

[MB09] Merouane Mahdi and Messaoud BenSebti. *Statistical Model for Intrusion Detection System.* Fifth International Conference on Sciences of Electronic Technologies of Information and Telecommunications,March 22-26,Tunisia, mar 2009. 49

[MCCC11] I Mukhopadhyay, M Chakraborty, S Chakrabarti, and T Chatterjee. Back propagation neural network approach to intrusion detection system. In *Recent Trends in Information Systems (ReTIS), 2011 International Conference on*, pages 303–308. IEEE, 2011. 29, 31

[Mer08] L.D. Merkle. *Automated network forensics.* Proceedings of the conference on genetic and evolutionary computation (GECCO 2008), pp. 1929-1932, 2008. 46

[MG93] Zweig MH and Campbell G. *Receiver-operating characteristic (ROC) plots: A fundamental evaluation tool in clinical medicine. Clinical Chemistry 39:561-577.* na, 1993. 132

[MGC+11] Sally McClean, Lalit Garg, Priyanka Chaurasia, Bryan Scotney, and Chris Nugent. Using model-based clustering to discretise duration information for activity recognition. *In 24th International Symposium on Computer-Based Medical Systems (CBMS*, pages 1–7, 2011. 65

[MPR10] Sanjay Kumar Malik, Nupur Prakash, and SAM Rizvi. Ontology and web usage mining towards an intelligent web focusing web logs. In *Computational Intelligence and Communication Networks (CICN), 2010 International Conference on*, pages 443–448. IEEE, 2010. 79

[MR11] G. Marjan and S. Ristov. *Performance Gains and Drawbacks using Set Associative Cache.* Journal of Next Generation Information Technology, 3(3): pp. 87-98, January 2011. 53

[MS03] S. Mukkamala and A. H. Sung. *Identifying significant features for network forensic analysis using artificial intelligent techniques.* International Journal of Digital Evidence, 1(4), December 2003. 49

[Mus11] L. Musthaler. *Rewind and replay what happens on your*

network. www.networkworld.com/newsletters/techexec/2007/0716techexec1.html, January 2011. 48

[Nag06] Asha Nagesh. Distributed network forensics using jade mobile agent framework. *Student Trade Show and Project Reports*, pages 5–6, 2006. 46

[Nea10] C. Niccolo and et. al. *iNFAnt: NFA pattern matching on GPGPU devices*. ACM SIGCOMM Computer Communication Review, 40(5):pp. 20-26, April 2010. 53

[Nit11] NitroGuard. Nitroguard intrusion prevention system. *Retrieved 16 April 2015, from http://www.ndm.net/ips/pdf/nitrosecurity/NitroSecurity%20IPS_Datasheet_06_2011.pdf*, 2011. 28

[NKS05] J. Newsome, B. Karp, and D. Song. *Polygraph: Automatically generating signatures for polymorphic worms*. Proceedings of the IEEE Symposium on Security and Privacy, November 2005. 32

[NST02] D. Newman, J. Snyder, and R. Thayer. *Crying Wolf: False alarms hide attacks*. Network World, June 2002. 11

[NTCV04] T. Sherwood N. Tuck, B. Calder, and G. Varghese. *Deterministic memory efficient string matching algorithmS for intrusion detection*. 23rd Conference of IEEE Communications Society (Infocomm), March 2004. 14

[NVI07] NVIDIA. *NVIDIA CUDA compute unified device architecture*

programming guide. http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIACUDAProgrammingGuide1.0.pdf, May 2007. 54, 56

[NVI13a] NVIDIA. *NVIDIA CUDA Zone*. https://developer.nvidia.com/cuda-zone, September 2013. 50

[NVI13b] NVIDIA. *NVIDIA GeForce GTX 680 Whitepaper*. http://www.geforce.com/Active/en_US/en_US/pdf/GeForce-GTX-680-Whitepaper-FINAL.pdf, September 2013. 51

[NVI13c] NVIDIA. *NVIDIA GeForce GTX 750 Ti Whitepaper*. http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce-GTX-750-Ti-Whitepaper.pdf, September 2013. 52

[NVI13d] NVIDIA. *NVIDIA GPU and Processors*. http://www.nvidia.in/page/home.html, September 2013. 50

[NVI13e] NVIDIA. *NVIDIA Updates GPU Roadmap Announces Pascal*. http://blogs.nvidia.com/blog/2014/03/25/gpu-roadmap-pascal/, September 2013. 52

[NVI13f] NVIDIA. *NVIDIA's Next Generation CUDA Compute Architecture: Fermi Architecture*. http://www.nvidia.in/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf, July 2013. 51

[NVI14a] NVIDIA. *CUDA Occupancy Calculator*. http://developer.download.nvidia.com/.compute/cuda/CUDA_Occupancy_calculator.xls, March 2014. 147

[NVI14b] NVIDIA. *Profiler*. https://developer.nvidia.com/nvidia-visual-profiler, February 2014. 147

[Oas06] Oasis. *OASIS Web Services Security (WSS) TC*. https://www.oasis-open.org/committees/wss/, September 2006. 104

[oD15] University of Dublin. *Grid Weka2*. Retrieved 10 March,2015 from http://www.andreas-hess.info/projects/gridweka2/GridWeka2.jar, 2015. 120

[Oe08] J. D. Owens and et.al. *GPU Computing*. Proceedings of the IEEE, 96(5): pp. 879 -899, January 2008. 90

[Ora07] Oracle. *Chesapeake energy deploys oracle grid computing*. http://www.gridtoday.com/grid/1875224.html, 2007. 19

[Oss15] Ossec.net. Ossec — home — open source security. *Retrieved 15 April 2015, from http://www.ionx.co.uk/products/verisys*, 2015. 28

[Pal01] G. Palmer. *A Road Map for Digital Forensic Research*. Report from the First Digital Forensic Research, Technical Report DTRT0010-01, DFRWS, November 2001. 47

[Pea10] Andre Proto and Leandro A. Alexandre et al. 2010. 49

[PY06] P. Piyachon and L. Yan. *Efficient memory utilization on network processors for deep packet inspection*. Proceedings of the ACM/IEEE

symposium on Architecture for networking and communications systems, pp. 71-80, March 2006. 33, 90

[pyG07] pyGridWare. *Python Web Services Resource Framework*. http://dsd.lbl.gov/gtg/projects/pyGridWare/, January 2007. 61

[QC14] Darren Quick and Kim-Kwang Raymond Choo. Data reduction and data mining framework for digital forensic evidence: storage, intelligence, review and archive. *Trends & Issues in Crime and Criminal Justice*, 480:1–11, 2014. 20

[Ran14] M. Ranum. *Network Flight Recorder*. http://www.ranum.com/, March 2014. 43

[Raz13] K. Raz. *Amazon Web Services Deploys NVIDIA GRID GPUs*. http://nvidianews.nvidia.com/news/amazon-web-services-deploys-nvidia-grid-gpus-2775398, January 2013. 20, 148

[RE04] ANM. Rafiq and M. Ehtesham. *A fast string search algorithm for deep packet classification*. Computer Communications, 27(15): p: 1524-1538., June 2004. 90

[Rec06] W3C Recommendation. *Web Services Addressing 1.0 - Core*. http://www.w3.org/TR/ws-addr-core, September 2006. 104

[REKG06] A.N.M. Ehtesham Rafiqa, M.Watheq El-Kharashib, and Fayez Gebali. *Architectures for bit-split string scanning in intrusion detection*. IEEE Micro, 26(1): p: 110-117., March 2006. 90

[RGO+09] Smith Randy, Neelam Goyal, Justin Ormont, Karthikeyan Sankaralingam, and Cristian Estan. *Evaluating GPUs for network packet signature matching*. IEEE International Symposium on Performance Analysis of Systems and Software, pp: 175-184, April 2009. 53, 129

[RJ05] W. Ren and H. Jin. *Modeling the network forensics behaviors*. Proceedings of the first international conference on security and privacy for emerging areas in communication networks (SecureComm 2005), pp.1-8, September 2005. 46

[RK10] B. Rudis and P. Kostenbader. *The enemy within: Firewalls and Backdoors*. http://www.symantec.com/connect/articles/enemy-within-firewalls-and-backdoors, November 2010. 9

[RS12] Sangeeta Rani and Geeta Sikka. Recent techniques of clustering of time series data: A survey. *International Journal of Computer Applications*, 52(15):1–9, 2012. 64

[SC13] Nidhi Srivastav and Rama Krishna Challa. Novel intrusion detection system integrating layered framework with neural network. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 682–689. IEEE, 2013. 30

[Sch07] D. L. Schuff. *High-performance network intrusion detection through parallelism*. Master's thesis, Purdue University, May 2007. 9

[SL04] D. V. Schuehler and J. W. Lockwood. *A modular system for fpga-based tcp flow processing in high-speed networks*. Proceedings of 14th

International Conference on Field Programmable Logic and Application (FPL 2004): pp. 301-310, August 2004. 34

[SM94] W. Sun and U. ManBer. *A fast algorithm for multi-pattern searching.* Technical Report TR-94-17, January 1994. 35

[SMSB03] Kulesh Shanmugasundaram, Nasir Memon, Anubhav Savant, and Herve Bronnimann. Fornet: A distributed forensics network. *Computer Network Security*, pages 1–16, 2003. 46

[Sno15] Snort. Snort.org. *Retrieved 15 April 2015, from http://www.snort.org*, 2015. 28, 32

[Sou13] SourceForge. Open source tripwire. *Retrieved 15 April 2015, from http://sourceforge.net/projects/tripwire/*, 2013. 28

[Sou15] Sourcefire. Sourcefire ips. *Retrieved 14 April 2015, from http://www.ndm.net/ips/solutions/sourcefire*, 2015. 28

[Spa13] M. Sparsh. *A Survey Of Techniques for Managing and Leveraging Caches in GPUs.* Journal of Circuits, Systems, and Computers, 23(8) pp: 143-149, January 2013. 54

[SRK14] B. Schuller, J. Rybicki, and K.Benedyczak. *High-Performance Computing on the Web: Extending UNICORE with RESTful Interfaces.* Poceedings the Sixth International Conference on Advances in Future Internet, November 2014. 63

[SSMF03] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland. *Characterizing the performance of network intrusion detection sensors.* Proceedings

of the Sixth International Symposium on Recent Advances in Intrusion Detection (RAID 2003), September 2003. 35

[ST05] L. Srinivasan and J. Treadwell. *An Overview of Service-oriented Architecture*. Web Services and Grid Computing by HP Software Global Business Unit, November 2005. 102

[ST13] Ashish Saklani and Parveen Tiwari. How to enhance network intrusion detection technique with integrated sampling and filtering. *International Journal of Innovative Research & Studies : Vol2, Issue 7*, pages 304–327, 2013. 86

[Sta06] Oasis Standard. *Web Services Brokered Notification 1.3*. http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.htm, October 2006. 104

[Sto01] Henry Stocker. *Deconstructing DoS attacks*. archives.cnn.com/2001/TECH/internet/03/07/dosattacks.idg/, March 2001. 4

[Sur15] Suricata. Suricata. *Retrieved 14 April 2015, from http://suricata-ids.org*, 2015. 28

[SV10] Alex Smola and S.V.N. Vishwanathan. *Introduction to Machine Learning*. Cambridge University Press, October 2010. 64

[Sym03] Symnatec. Netprowler. *Retrieved 14 April 2015, from http://securityresponse.symantec.com*, 2003. 28

[Sys15] Ionx Verisys File Integrity Monitoring System.

http://ionx.co.uk. *Retrieved 15 April 2015, from http://www.ionx.co.uk/products/verisys*, 2015. 28

[TC09] Hua Tang and Zhuolin Cao. Machine learning-based intrusion detection algorithms. *Journal of Computational Information Systems*, 5(6):1825–1831, 2009. 29, 31

[TCH+14] Olufon Tope, Carlene EA Campbell, Stephen Hole, Kapilan Radhakrishnan, , and Arya Sedigh. *Mitigating External Threats in Wireless Local Area Networks*. International Journal of Communication Networks and Information Security (IJCNIS), 6(3), January 2014. 54

[Tcp10] Tcpdump. *Tcpdump/Libpcap Public Repository*. `Available:http://www.tcpdump.org/`, March 2010. 96

[TEL11] Y. Torres, A. G. Escribano, and D. R. Llanos. *Understanding the impact of CUDA tuning techniques for Fermi*. IEEE International Conference on High Performance Computing and Simulation (HPCS), pp. 631-639, January 2011. 53

[TMN07] A. Tamer, A. Mohaisen, and D. Nyang. *Deep packet inspection for intrusion detection systems: A survey*. Magazine of Korea Telecommunication Society, 24(11): pp. 25-36, November 2007. 42, 90

[Tre15] Trendmicro.com. Virtualization security - server virtualization - deep security - trend micro usa. *Retrieved 14 April 2015, from http://www.trendmicro.com/us/enterprise/cloud-solutions/deep-security/*, 2015. 28

[TS06] L. Tan and T. Sherwood. *Architectures for bit-split string scanning in intrusion detection.* IEEE Micro:Micros Top Picks from Computer Architecture Conferences, January 2006. 33

[TVS10] A. Tumeo, O. Villa, and D. Sciuto. *Efficient pattern matching on GPUs for intrusion detection systems.* 7th ACM international conference on Computing frontiers, pp. 87-88, December 2010. 53

[TWWL12] L. Ting, Z. Wang, H. Wang, and K. Lu. *An Entropy-based Method for Attack Detection in Large Scale Network.* International Journal of Computers Communications & Control, 7(3):pp. 509-517, March 2012. 90

[Ven03] P. Venezia. *Netdetector captures intrusions.* Infoworld, issue 27, July 2003. 48

[WD05] W. Wang and T. E. Daniels. *Building evidence graphs for network forensics analysis.* Proceedings of the IEEE Computer Security Applications Conference 21st Annual (ACSAC'05), pp. 254-266, Tucson, AZ, USA, December 2005. 43, 44

[WFH10] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, December 2010. 133

[WFH11] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, December 2011. 74, 105

[Wir10] Wireshark. *Network Protocol Analyzer*. http://www.wireshark.org/, March 2010. 96

[WP05] A. Wagner and B. Plattner. *Entropy based worm and anomaly detection in fast IP networks*. 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, June 2005. 90

[WZDT10] Z. Wu, C. Zhangxin, C. C. Douglas, and W. Tong. *High Performance Computing and Applications*. Second International Conference, HPCA 2009, Shanghai, China, January 2010. 16

[YH03] B. Yuebin and K. Hidetsune. *Intrusion detection systems: Technology and development*. Proceedings of the 17th International Conference on Advanced Information Networking and Applications(X'ian, China, March 27-29, 2003). AINA'03, March 2003. 9

[YM10] A. Yasinsac and Y. Manzano. *Policies to enhance computer and network forensics*. Proceedings of the IEEE workshop on information assurance and security, pp. 289-295, January 2010. 43, 46

Publications

2015

1. Sharma Jyotsna and Singh Maninder, "Web Services oriented architecture for DPI based Network Forensics Grid", International Journal of Energy, Information and Communications (IJEIC) , Volume 6, Issue 3 : pages 19-28, 2015.

2. Sharma Jyotsna and Singh Maninder, "CUDA based Rabin-Karp pattern matching for Deep Packet Inspection on a Multicore GPU", International Journal of Com. Networks and Information Security (IJCNIS).

2010

3. Sharma Jyotsna and Singh Maninder, "Grid Infrastructure Based Forensic Analysis for Combating Global Terrorism", Poster proposal has been accepted for the Grace Hopper Conference, September 28-October 2,2010, Atlanta, Georgia, USA.

2009

4. Sharma Jyotsna and Singh Maninder(2009,Sept. 30th - Oct. 2nd) , "DPI based Forensic Analysis of Network Traffic using Grid Infrastructure", Paper presented and published in proceedings of the Grace Hopper Celebration of Women

in Computing Conference(GHC 2009), 30th September-2nd October, 2009,Tucson, Arizona, USA, pp.146-152.

5. Sharma Jyotsna , "Grid Computing: The technology & the Indian Initiatives", Paper published in "Handbook of Research on Grid Technologies and Utility Computing", an IGI Global Publication, edited by Emmanuel Udoh, Indiana University-Purdue University, USA, and Frank Wang, Cranfield University, UK , pp. 292-302.