A Graphical User Interface Framework for detecting Intrusions using Bro IDS

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Technology
in
Computer Science and Applications

Submitted By:
Shaffali Gupta
(601003025)

Under the supervision of:
Ms. Sanmeet Kaur
Assistant Professor
SMCA

School of Mathematics and Computer Applications
THAPAR UNIVERSITY
PATIALA – 147004

June 2012

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*A Graphical User Interface Framework for detecting Intrusions using Bro IDS*" in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Applications* submitted in School of Mathematics & Computer Applications of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms. Sanmeet Kaur* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Shaffali Gupta)
601003025

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

(Ms. Sanmeet Kaur)
Assistant Professor
SMCA

Countersigned by

(Dr. S.S. Bhatia)
Head,
SMCA,
Thapar University,
Patiala.

(Dr. S. K. Mohapatra)
Dean (Academic Affairs),
Thapar University,
Patiala.

Acknowledgement

I would like to express my deepest appreciation to Ms. Sanmeet Kaur, my mentor and thesis supervisor for her constant support and motivation. She had been instrumental in guiding me throughout the thesis with their valuable insights, constructive criticisms and interminable encouragement.

I am also thankful to Dr. S.S. Bhatia, Head, School of Mathematics and Computer Applications and Mr. Singara Singh, P.G. Coordinator for their constant support and encouragement.

I would like to thank all the faculty members and staff of the department who were always there at the need of the hour and provided with all the help and facilities, which I required, for the completion of this work.

I express my thanks to my family for their support and affection and for believing in me always. I also want to thank my colleagues, who have given me moral support and their relentless advice throughout the completion of this work.

Shaffali Gupta
(601003025)

Abstract

Internet has transformed and greatly improved the way we do business, this vast network have opened the door to an increasing number of security threats from which corporations must protect them. Although network attacks are presumably more serious when they deal with businesses that store sensitive data, such as personal, medical or financial records, the consequences of attacks on any entity range from mildly inconvenient to completely debilitating-important data can be lost, privacy can be violated, and several hours, or even days, of network downtime can ensue. To protect the network, Network security is needed. Network security is the provision made in an underlying computer network or rules made by the administrator to protect the network and its resources from unauthorized access.

To make network secure, an Intrusion detection system is one of the efficient system. An intrusion is used to monitor network traffic, check for suspicious activities and notifies the system or network administrator. Taking a closer look at open source Network Intrusion Detection System, there is a very powerful open source system that is termed as Bro. It passively monitors network traffic and looks for suspicious activity by comparing network traffic against scripts. If Bro detects something of interest, it can be instructed to either issue a log entry or initiate the execution of an operating system command.

Some Policy Scripts are already built in Bro IDS. In this thesis, various types of live traffic is captured and analyzed. Some new policy scripts are built to filter out the needed packets from the captured traffic. Also, a Graphical User Interface is designed to ease the make and run of scripts that eliminates the need of writing the commands at terminal.

Table of Contents

List of Figures

List of Snapshots

List of Tables

Chapter 1

Introduction

1.1 Introduction to Network Security

Internet has transformed and greatly improved the way business is performed, this vast network and its associated technologies have opened the door to an increasing number of security threats from which corporations must protect. Although network attacks are presumably more serious when they deal with businesses that store sensitive data, such as personal, medical or financial records, the consequences of attacks on any entity range from mildly inconvenient to completely debilitating. Due to attack, important data can be lost, privacy can be violated, and several hours, or even days, of network downtime can ensue. With networks more vulnerable and hackers equipped to cause havoc, it is no surprise that network attacks are on the rise. Unfortunately security policies and rules needed to govern these networks have not progressed as rapidly. So considering all this there arise the need for fast reactionary capabilities in network security where network security consists of the provisions and policies adopted by the network administrator to prevent and monitor unauthorized access, misuse, modification or denial of the computer network and network accessible resources. Network security components often include:

- Anti-virus and anti-spyware.
- Firewall, to block unauthorized access to the network.
- Intrusion Detection System (IDS), to identify fast-spreading threats.
- Virtual Private Networks (VPN), to provide secure remote access.

Recent security incidents and analysis have proved that manual response to network attacks is no longer feasible. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Network security covers a variety of computer networks, both public and private that are used in everyday jobs conducting transactions and communications among businesses, government agencies and individuals. Networks can be private, such as within a company and others which might be open to public access. Network

security secures not only the network, as well as protecting and overseeing operations being done.

Initially to make the security automated firewalls provides a great solution. After some time it was found that firewalls usually detect data packet header to make traffic flow control decisions. But they do not inspect the entire content of the packet and cannot detect malicious code embedded within normal traffic. While firewalls and router based packet filtering are necessary components of network security topology, they are insufficient on their own. For overcoming from these insufficiencies IDS components are being deployed outside and inside firewall. Intrusion detection systems offer techniques for modeling and recognizing normal and abusive system behavior. They provide many of the methodologies that include statistical models, protocol verification, neural networks, expression matching, state transition analysis, Expert system and genetic algorithms that fall under anomaly and misuse intrusion detection system.

Different definitions of security in context of different perspectives are:

- The state of being free from danger or threat.
- The safety of a state or organization against criminal activity such as terrorism, theft is termed as "national security".
- In information technology, security is the protection of information assets through the use of technology, processes, and training.

1.1.1 Necessity of Security

Security provides a safe platform for computers, programs and users to carry out their allowable significant functions within a protected environment. A secure network protect its users from viruses, worms, Trojan Horses as they are enough dangerous to leak out all the information present in the system or to make the system resource unavailable [20].

- **Authentication**

  Authentication helps to prove the user that the entity to which they communicate is one that it claims to be. Authentication of user is essential to ensure proper authorization and access to system resources and services. In the simple way authentication can be conducted through

the user login passwords, single sign-on (SSO) systems, biometrics and digital certificates [35].

- **Confidentiality**

  Confidentiality ensures that information send by authenticated party does not get leaked or disclosed to any unauthorized individual, host or systems. For example, a credit card transaction on the internet requires the credit card number to be transmitted from the buyer to the merchant and from the merchant to a transaction processing network. The system enforces confidentiality by encrypting the credit card number during transmission and by limiting the places where it might appear i.e. in databases, log files, backups, printed receipts etc. Confidentiality also enforce by restricting access to the places, where the data is stored. Confidentiality is said to be compromised, if any unauthorized host obtains the card number.

- **Data integrity**

  Data integrity is the means of giving assurance that information can only be accessed or modified by those authorized to do so. Some measures taken to ensure integrity include controlling the physical environment of networked terminals and servers, restricting access to data and maintaining precise authentication policies [3].

- **Non-repudiation**

  It provides protection against denial by one of the entities involved in communication. Non-repudiation is the reassurance that someone cannot deny something. Typically, non-repudiation refers to the ability to ensure that a party cannot deny the authenticity of their signature or information provided by party on a document or the sending of a message that they actually originated.

1.1.2 Threats to Network Security

The numbers of security threats are increasing day by day. These attacks can range from mildly inconvenient to completely debilitating. Some of the common attacks are:

- **Buffer overflow:** A buffer overflow occurs when a program or process tries to store more data in a buffer than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information which has to go somewhere can overflow into adjacent buffers, corrupting or overwriting the valid data held in them. Although it may occur accidentally through programming error, buffer overflow is an increasingly common type of security attack on data integrity. In buffer overflow attacks, the extra data may contain code designed to trigger specific actions, in effect sending new instructions to the attacked computer that could, for example, damage the user's files, change data, or disclose confidential information.
- **Backdoor trojan:** A backdoor Trojan opens a backdoor to the system. They are also called as Remote Access Trojans (RAT). These are the most widespread and also the most dangerous type of Trojan. They are so dangerous because they have the potential to allow remote administration of the system. In this, hackers were virtually supposed to sit at their keyboards. There is almost no limit to what they can do.
- **Denial of service attack:** A denial-of-service (DOS) attack or distributed denial-of-service (DDoS) attack is an attempt to make a computer or network resource unavailable to its intended users. One common method of attack involves saturating the target machine with external communications requests, so that it cannot respond to legitimate traffic, or responds so slowly as to be rendered effectively unavailable. This attack usually leads to a server overload.
- **Arbitrary code execution:** Arbitrary Code Execution is defined as an attacker ability to execute any commands of the attacker choice on a target machine. It is commonly achieved through control over the program counter of a running process. The instruction pointer points to the next instruction in the process that will be executed. Control over the value of the instruction pointer therefore gives control over which instruction is executed next. In order to execute arbitrary code, many exploits inject code into the process and use a vulnerability to change the instruction pointer to have it point to the injected code. The injected code will then automatically get executed.
- **Privilege escalation:** It is the act of exploiting a bug in an operating system to gain prominent access to resources that are normally protected from an application or user. The result is that an application with more privileges can perform unauthorized actions.

1.1.3 Solutions to achieve Network Security

With an increasing amount of people getting connected to networks, the security threats that cause massive harms are increasing also. A no. of solutions to achieve security have also been increased from past to present. Some of them explained here are:

- **Intrusion detection system:** It is a device that monitors network traffic to find malicious activities, violation of rules and notifies the administrator. Some systems may attempt to stop an intrusion attempt but this is neither required from a monitoring system. In addition, organizations use IDS for other purposes, such as identifying problems with security policies, documenting existing threats, and deterring individuals from violating security policies. IDS have become a necessary addition to the security infrastructure of nearly every organization.
  IDS typically record information related to observed events, notify security administrator of important observed events, and produce reports. Many IDS can also respond to a detected threat by attempting to prevent it from succeeding. They use several response techniques, which involve the IDS stopping the attack itself, changing the security environment, or changing the attack's content [28].

- **Firewalls:** A firewall is a device or set of devices designed to permit or deny network transmissions based upon a set of rules and is frequently used to protect networks from unauthorized access while permitting legitimate communications to pass. Many personal computer operating systems include software-based firewalls to protect against threats from the public Internet. A network firewall builds a bridge between an internal network that is assumed to be secure and trusted and another network, usually an external network, such as the Internet, that is not assumed to be secure and trusted [39].

- **Honeypot:** A honeypot is a device that allows bad patterns to come inside the network so that their behaviour can be studied to further strengthen the system. Its primary function is to study the way in which the attackers progress and establish their lines of attack. Thus firewalls and IDS used defensive security whereas honeypot work as offensive security measure. A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. It means honeypot derives their value from threats using them. If the enemy does not interact with the honeypot, then it has little value. Honeypots are very different from most security mechanisms, and this difference that makes them a powerful

tool. Honeypots can be classified based on their purpose (production, research, and honey tokens) and level of interaction (low, medium, and high) [36].

1.1.4 Difference between Firewall and IDS

Initially to secure the networks, firewalls provides a great solution. Later it was found that firewall usually detect data packet header to make traffic flow control decisions. But they do not inspect the entire content of the packet. While firewall is necessary components of network security, they are insufficient in their own. For overcoming from these insufficiency IDS components are being deployed outside and inside firewalls.

- There are many forms of IDS. Network IDS and Host IDS are the examples while Firewalls are not limited to a perimeter of a Network, Firewalls can be sophisticated.
- The purpose of Intrusion detection system is to analyze the traffic that goes through it and detect possible intrusions to the system while firewall is a network device that separates the internal network from the external network. It allows internal users to go out, but prevents unauthorized external traffic to go inside.
- An Intrusion detection system (IDS) is software or hardware designed to detect unwanted attempts but Firewall is used to block unauthorized connections from other computers.
- IDS detects the attacks by using Signature and Patterns much like an Anti Virus do but Firewall will scan the packets based on Source and Destination address and block or allow traffic accordingly

1.2 Introduction to Intrusion Detection System

When user of an information system takes an action that actually was not legally allowed to take, it is called intrusion. The intruder may come from outside, or the intruder may be an insider, who exceeds his limited authority to take action. Whether or not the action is detrimental, it is of concern because it might be detrimental to the health of the system, or to the service provided by the system.

As information systems have come to be more comprehensive and a higher value asset of organizations, complex intrusion detection subsystems have been incorporated as elements of operating systems, although not typically applications. Intrusion detection system involves determining that some entity, an intruder has attempted to gain or worse, has gained unauthorized access to the system. Most intrusion detection systems attempt to detect suspected intrusion and then they alert system administrator. The technology for automated reaction to intrusion is just beginning to be fashioned. Original intrusion detection system assumed a single, stand-alone processor system and detection consisted of post-facto processing of audit records. Today systems consist of multiple nodes executing multiple operating systems that are linked together to form a single distributed system [3].

1.2.1 Architecture of Intrusion Detection System

IDS may be deployed using a centralized or distributed architecture. In a centralized architecture, the collection and analysis of data occurs on the same device. The device might even be in-line, becoming a potential bottleneck in the network throughput. In a distributed architecture, a load-balancing system might send portions of traffic to multiple sensors that in turn send the collected data to one or more analysis stations and management consoles. Figure 1.1 shows a generalized architecture of network IDS.

An IDS is a sequential process consisting of five basic components:

- **Load balancing**: It allows the IDS to efficiently utilize the processing power of the distributed sensors for scalability. Load balancers may be in-line with a border router so that all traffic must go through it or all traffic may be mirrored to it. If an IDS has no load-balancing component, the load may be statically spread out by placing sensors in separate subnets. High-bandwidth load balancers may allow the IDS to collect traffic higher up in the network, closer to the border router. The result will be more efficient use of sensors and better protection for the monitored network [7].
- **Sensors:** The sensors receive traffic from the load balancer (if any exists) and separate out the suspicious traffic for further analysis. Potentially a many-to-many (M:M) relationship between sensors and analyzers is possible. Separating sensing from analysis may allow better throughput by offloading the analysis burden, but separation adds

network overhead. Throughput may be improved more directly by efficient load balancing. Simple sensors are either signature-based or anomaly based.

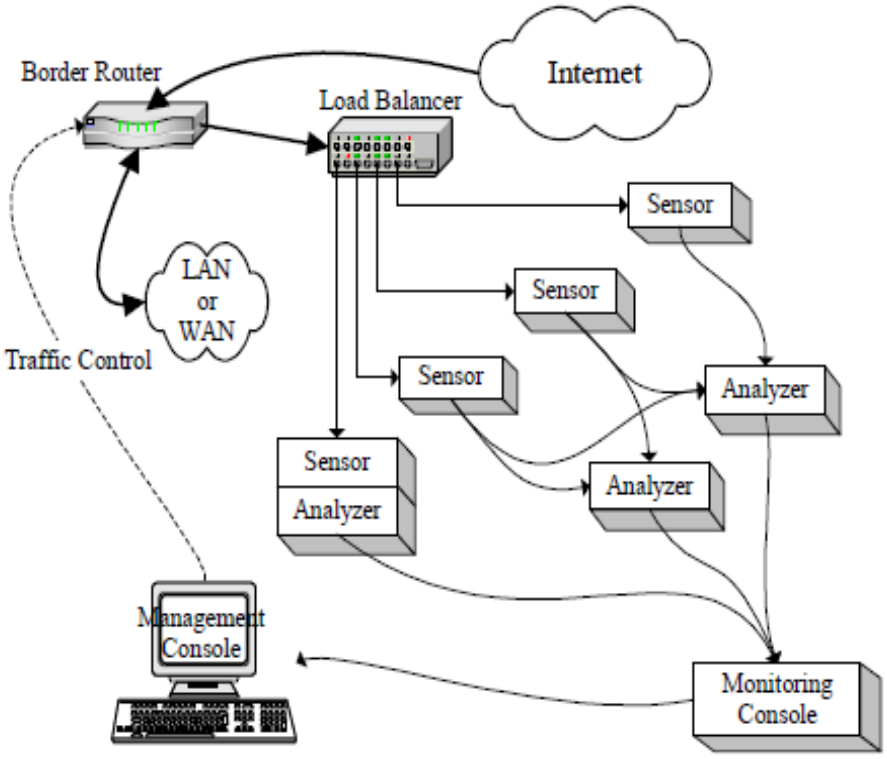


Figure 1.1: Generalized network IDS architecture [7].

- **Analyzers:** They determine the threat level of the raw data collected by the sensors. Suspicious traffic may be either classified as an attack of some severity, or it may be dismissed as safe. Typically a many-to-one (M:1) relationship exists between the analyzers and the monitors they report to. Firstly analysis determines threat severity. Secondly analysis determines scope, intent, or frequency of the threat. Accurate analysis may require storage of a significant amount of historical data that can be used to give context to a perceived threat. Good analysis can correlate one attack with another or determine that no such correlation is appropriate [7].

- **Monitor:** Monitor is required to notify an operator whenever a threat is severe according to a security policy. Monitor must be tuned according to the traffic patterns of the protected network.

1.2.2 Classification of IDS

Four major parameters on which an IDS can be classified are named as:

(i) Intruder Type

(ii) Detection Behavior

(iii) Detection Approaches

(iv) System Types

These parameters can be sub classified into sub parameters. The major parameters on which classification of IDS is done are shown below in Figure 1.2.

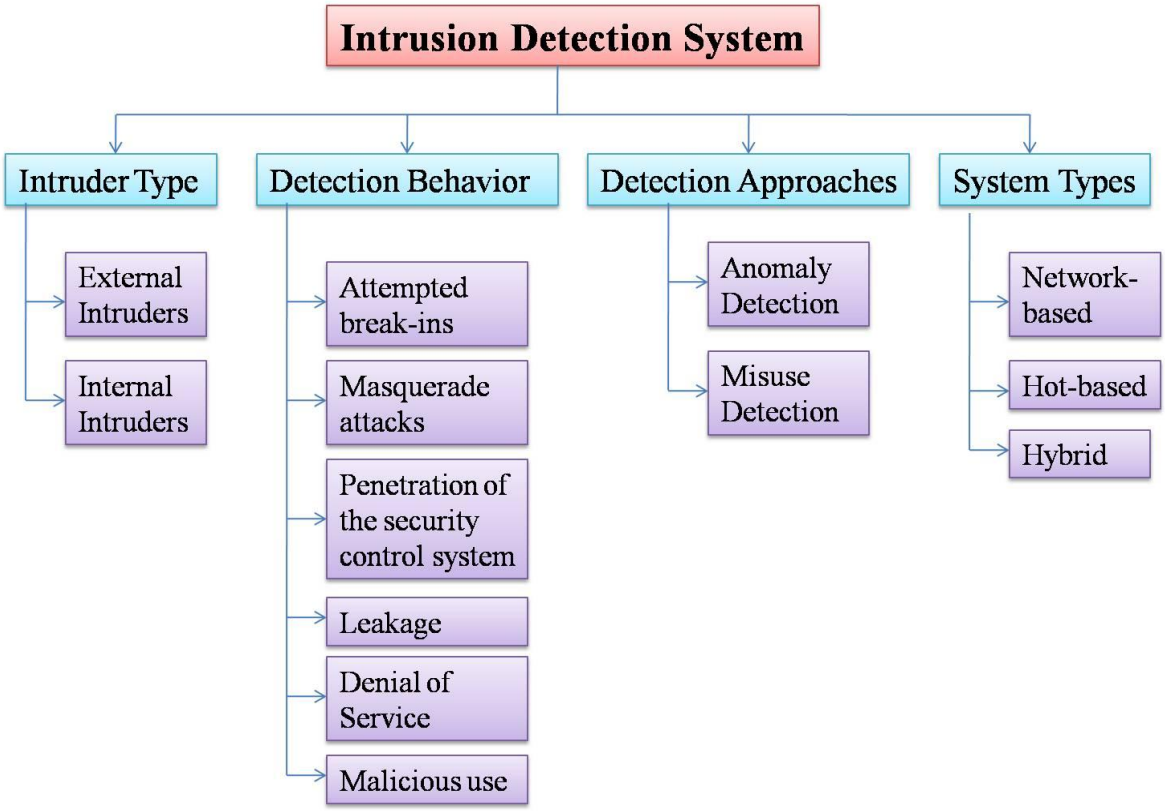


Figure 1.2: Classification of IDS [32].

(i) Intruder Type

IDS can be classified into external and internal according to intruder type. The external intruders are unauthorized users of the system they attack and internal intruders have permission to access the system, but not some portions of it. Furthermore, internal intruders divide into intruders who masquerade as another user, those with legitimate access to sensitive data and the most dangerous type, the clandestine intruders who have the power to turn off audit control for themselves [32].

(ii) Detection Behaviour

IDS can be divided into six main types according to detection behaviour:

- Attempted break-ins, which are detected by typical behaviour profiles or violations of security constraints.
- Masquerade attacks, which are detected by typical behaviour profiles or violations of security constraints.
- Penetration of the security control system, which are detected by monitoring for specific patterns of activity.
- Leakage, which is detected by typical use of system resources.
- Denial of service, which is detected by typical use of system resources.
- Malicious use, which is detected by typical behaviour profiles, violations of security constraints, or use of special privileges [16].

(iii) Detection Approaches

IDS that monitor computer systems and networks, analyze them for signs of security policy violation and respond accordingly, is based on one of the following approaches [22].

- **Anomaly detection approach**

  Anomaly detection systems notify activities that deviate significantly from the established normal usage profiles and termed them as anomalies, i.e., possible intrusions. Anomaly detection is an important tool for detecting fraud, network intrusion, and other rare events that have great significance.

- **Misuse detection approach**

  Misuse detection systems use patterns of well known attacks or weak spots of the system to match and identify known intrusions. In misuse detection approach, abnormal system behavior is defined at first, and any other behavior which is not defined is seemed as normal behavior. It fails to detect truly innovative attacks.

(iv) System Types

Intrusion detection system as the name implies is used to detect the intrusions. The area of intrusion can be limited to a single host or can be a large network. Based on the area, system types of IDS can be:

- **Network-based**

  A network-based IDS usually consists of a network appliance with a Network Interface Card (NIC) operating in promiscuous mode. The IDS is placed along a network segment or boundary and monitors all traffic on that segment. Network intrusion detection system (NIDS) is used to flag and sometimes to stop an attack before it gets information assets or causes damage. They capture network traffic and compare the traffic with a set of known attack pattern or signatures. NIDS devices compare these signatures with every single packet that they see, in hopes of catching intruders in the traffic. NIDS devices can be deployed passively, without requiring major modifications to system or networks. NIDS are effective for monitoring both inbound and outbound network traffic.

  Although HIDS is far better than NIDS in detecting malicious activities for a particular host, they have limited view of entire network topology and they cannot detect attack that is targeted for a host in a network which does not have HIDS installed [8].

- **Host-based**

  Host-based Intrusion Detection System (HIDS) monitor specific files, logs and registry settings on a single PC. HIDS alert on any access, modifications, detection and copying of the monitored object. The role of HIDS is to flag any tampering with a specific PC and can automatically replace altered files when changed to ensure data integrity. Most host-based IDSs will have components that parse system logs and watch user logins and processes. Some of the more advanced system will even have built-in capabilities to catch Trojan code deployments. Host-based systems are agent-based that is they require the installation of program on the systems they protect. This installation procedure allows Host-based systems to be more thorough on some levels, but also more of a headache to deploy and administer these kinds of systems.

  Agents are deployed software installed on particular PC in a HIDS. Agent software generally has a small footprint and uses very little processing power. The agents function is to monitor specific files or logs on the host and report when these particular files are accessed or modified to the central manager [8].

- **Hybrid-based**

  Hybrid Intrusion Detection System complements HIDS technology with the ability to monitor the network traffic coming in or out of specific host. This is very different from NIDS technology that monitors all network traffic. Hybrid agents combine the functionality of host based agent with network based sensor technology that is limited to analyzing only the network traffic addressed to the specific host where the hybrid agent is installed. A hybrid agents footprint is generally larger because of the additional functionality. The processor utilization of the hybrid agent is much greater than a host-based agent because of the continual processing of network traffic for the host [25].

1.2.3 Characteristics of IDS

The ultimate goal of IDS is the identification of all intrusive behavior within an environment, and the reporting of that behavior in a timely manner. However, in order for an IDS to be successful, some characteristics that will be needed are:

a) It must run continually without human supervision. The system must be reliable enough to allow it to run in the background of the system being observed. However, it should not be a black box. That is, its internal workings should be examinable from outside.
b) It must be fault tolerant in the sense that it must survive a system crash and does not require its knowledge base to rebuild at restart.
c) It must impose minimal overhead on the system. A system that slows a computer to a crawl will simply not be used.
d) It must observe deviations from normal behavior.
e) Listen to port activity and alert administrators when specific ports are accessed.

1.2.4 IDS in Network Topology

Depending on network topology, Intrusion Detection System can be placed at one or more places as shown in Figure 1.3. It also depends upon what type of intrusion activities are to be detected: external, internal or both. If only external intrusion activities need to be detected, and only one router is connected to the internet, the best place for an intrusion detection system is inside the router or a firewall. If there are multiple paths to the internet, one IDS box can be placed at each entry point. However, if internal threats need to be detected, place IDS box in every network segment. In many cases there is no need to detect intrusion detection activities in all network segments, and then limit IDS only to sensitive network areas. More intrusion detection system mean more work and more maintenance cost. The decision depends upon security policy and rules, which define what is, need to be protected from the hackers [27].

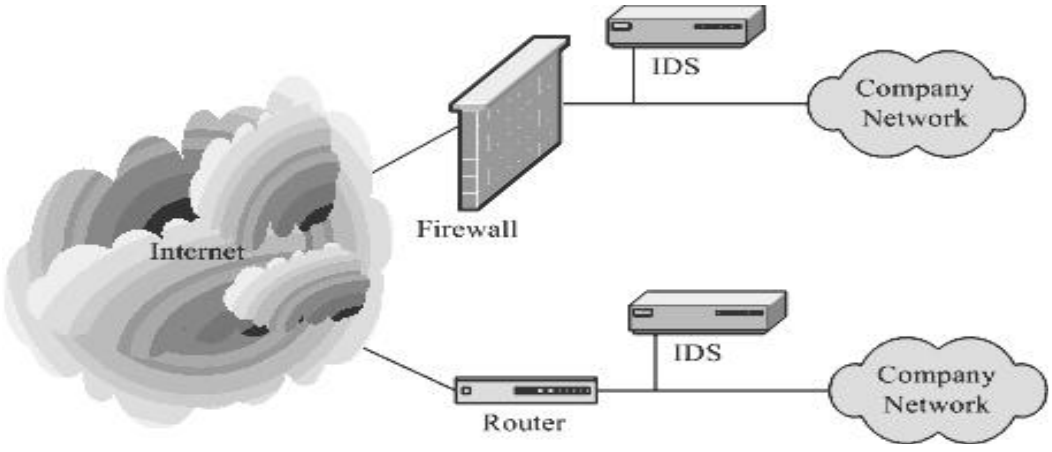


Figure 1.3: Position of Intrusion Detection System in Network [27].

Chapter 2

Literature Survey

2.1 Intrusion Detection System Approaches

The ability of intrusion detection is to distinguish normal system behavior from that system which is abnormal or harmful. For distinguishing the behavior the approaches used are shown in Figure 2.1 [22].

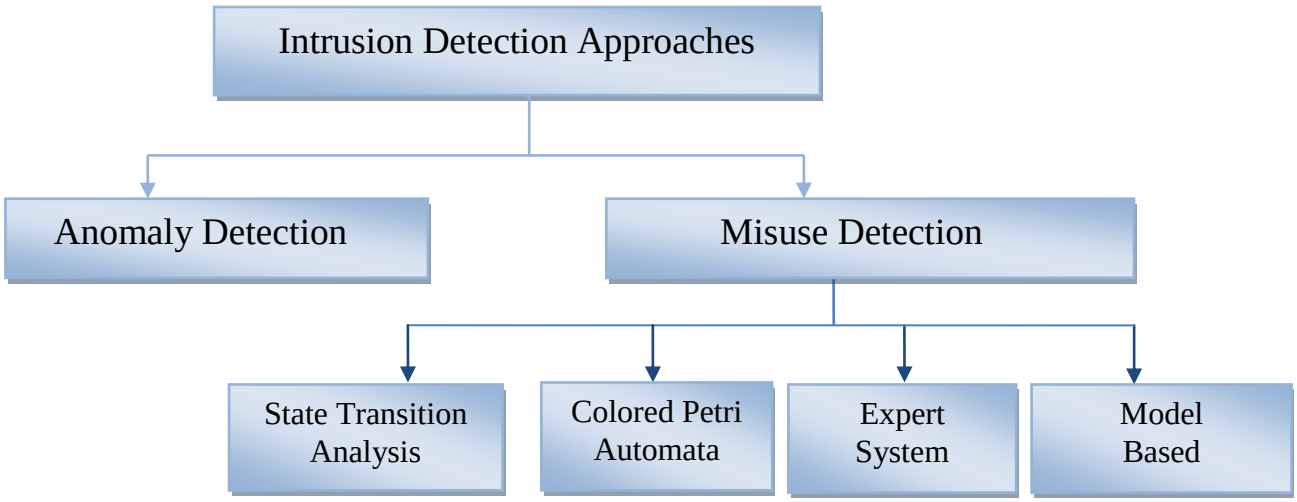


Figure 2.1: Detection Approaches of IDS

2.1.1 Anomaly Detection Approach

Anomaly detection systems flag observed activities that deviate significantly from the established normal usage profiles as anomalies, i.e., possible intrusions. The anomaly detector must be able to distinguish between the anomaly and normal. Anomaly detection can be divided into static and dynamic detector. A static anomaly detector is based on the assumption that there is a portion of the system being monitored that should remain constant. Dynamic anomaly detectors must include a definition of behavior. System behavior is defined as a sequence (or partially ordered sequence) of distinct events. When the defined sequence is meet, anomaly is said to occur.

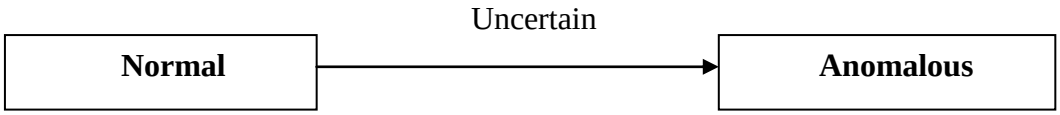


Figure 2.2: Boundary between Normal and Anomalous Behavior [16]

The most common way to draw this boundary is with statistical distributions having a mean and standard deviation. Once the distribution has been established, a boundary can be drawn using some number of standard deviations. If an observation lies at a point outside of the number of standard deviations, it is reported as a possible intrusion [16].

2.1.2 Misuse Detection Approach

Misuse detection systems use patterns of well known attacks or weak spots of the system to match and identify known intrusions. For example, a signature rule for the guessing password attack can be like as intrusion is detected if there are more than 4 failed login attempts within 2 minutes [22]. In an ideal case, a system administrator would be aware of all the known vulnerabilities and want to eliminate them. An organization may decide that eliminating known vulnerabilities is cost prohibitive. Users, who slowly modify their activity so that their profiles contain the abusive activity, are nearly impossible to detect with anomaly detectors. So, misuse detection systems look for known intrusions irrespective of the user normal behavior. A misuse detection system typically compares current system activity to a set of intrusion scenarios in an attempt to detect an intrusion in progress. Misuse detection systems can track the intrusion attempt against the intrusion scenario action by action. During the sequence of actions, the intrusion detection system can anticipate the next step of the possible intrusion. Given this information, the detector can more deeply analyze system information to check for the next step or can determine that the intrusion attempt has proceeded far enough and intervene to mitigate possible damage. The description of the detected intrusion scenario will substantially determine how efficient monitoring can be. Current system activity as seen by the intrusion detection system can be real time observations strictly for the use of the intrusion detection system, or it can be the audit records as recorded by the operating system [16].

The pros and cons of anomaly and misuse detection approach are discussed in Table 2.1 that helps to make these approaches differentiable in a better manner.

Table 2.1: Comparison of Intrusion Detection Approaches

Intrusion Approach	Advantages	Disadvantages
Anomaly Detection Approach	1. It does not require prior knowledge of intrusion and can thus detect new intrusions [14]. 2. Anomalies are recognized without getting inside their causes and characteristics. 3. Less dependence on operating system [11].	1 .Very low false alarm rate. 2. Very simple algorithm. 3. Easy creation of attack signature database. 4. Easy implementation and typical minimal system resource usage.
Misuse Detection Approach	1. It can accurately and efficiently detect instances of known attacks. 2. User behaviours can vary with time, requiring a constant update of normal behaviour profile database. [11]. 3. The necessity of training the system for changing behaviour makes system immune to anomalies detected during training phase.	1. It lacks the ability to detect the truly innovative attacks [14]. 2. Difficulties in updating information on new attack types. 3. They seemed to have difficulty in handling internal attacks.

2.2 Primary Approaches to Misuse Detection Approach

There are various approaches used for misuse detection system. Each of these approaches has their relative pros and cons. Some are easy to use and quick to implement. Others are complex and may be difficult to understand but also very effective in detecting the intrusions. A detailed

comparative analysis of various misuse detection approaches with their pros and cons is shown in Table 2.2.

2.2.1 State Transition Analysis Technique (STAT)

Initially, state transition analysis technique was adopted to facilitate the specification of the patterns of known attacks. It is based on the assumption that all penetrations share two common features.

(i) Penetrations require the attacker to possess some minimum prerequisite access to the target system.

(ii) All penetrations lead to the acquisition of some ability that the attacker does not have prior to the attacks.

It views an attack as a sequence of actions performed by an attacker that leads from some initial state on a system to a target-compromised state. STAT requires some critical actions, called signature actions, to identify the attacks. With the series of state changes and the signature actions that cause the state changes, an attack scenario is then represented as a state transition diagram, where the states are specified by assertions of certain conditions [21].

2.2.2 Colored Petri Automata (CPA)

It is presented as an abstract hierarchy for classifying intrusion signatures based on the structural interrelationships among the events that compose the signature. Events in such a hierarchy are high-level events that can be defined in terms of low-level audit trail events and used to instantiate the abstract hierarchy into a concrete one. User-specified actions may be associated with such patterns and are executed when patterns are matched. The adapted colored Petri nets are called colored Petri automata. A CPA represents the transition of system states along paths that lead to intruded states. A CPA is also associated with pre and post conditions that must be satisfied before and after the match [29].

2.2.3 Expert System

An expert system is defined [18] as a computing system capable of representing and reasoning about some knowledge-rich domain with a view to solve problems. Expert system detectors code knowledge about attacks as if-then implication rules. Rules specify the conditions requisite for an

attack in their if part. When all the conditions on the left side of a rule are satisfied, the action on the right side of the rule is performed. This may trigger the firing of more rules or conclude the occurrence of an intrusion based on available data.

2.2.4 Model Based System

This system combines models of misuse with evidential reasoning to support conclusions about its occurrence. There is a database of attack scenarios, where each scenario comprises a sequence of behaviors making up the attack. At any moment the system is considering a subset of these attack scenarios. It seeks to verify them by seeking information in the audit trail to substantiate the attack scenario (the anticipator). The anticipator generates the next behavior to be verified in the audit trail, based on the current active models and passes these behaviors to the planner. The planner determines how the hypothesized behavior will show up in the audit data and translates it into a system dependent audit trail match. This mapping from behavior to activity must be easily recognized in the audit trail, and must have a high likelihood of appearing in the behavior [17]. The evidential reasoning calculus built into the system permits the update of the likelihood of occurrence of the attack scenarios in the active models list.

Table 2.2: Analysis of Various Misuse Detection Approaches

Misuse Detection Approaches	**Advantages**	**Disadvantages**	**Real-Time Applications**
State Transition Analysis Technique	More robust in detecting unknown vulnerabilities because of the trade of monitored system state are emphasized [37].	1. Attack patterns can specify only a sequence of Events rather than more complex forms. 2. Some intrusion behaviors cannot be described by state transition diagrams.	STAT has been applied for misuse detection in distributed systems, and networks. USTAT is the first prototype of STAT, which is aimed at misuse detection in UNIX systems [5].

Colored Petri Automata	A benefit of this classification scheme is that it clarifies the complexity of detecting the signatures in each level of the hierarchy.	1. CPA is quite expensive. 2. If the intrusions are described in every detail, the attacker may be able to change his/her attacking strategy and bypass the IDS.	CPA has been implemented in a prototype misuse detection system called Intrusion Detection In Our Time (IDIOT) [15].
Expert System	1. The main advantage is in formulating of if-then implication rules are the separation of control reasoning from the formulation of the problem solution. 2. Its chief use in misuse detection is to symbolically deduce the occurrence of an intrusion based on the available data.	1. Working memory elements that match the left sides of productions to determine eligible rules for firing are essentially sequence-less [13]. 2. Other problems include software engineering concerns with the maintenance of the knowledge base and quality of rules.	The Next Generation Intrusion Detection Expert System (NIDES) developed by SRI applied the Expert system approach [13].

Model Based System	1. The advantage is its basis in mathematically sound theory of reasoning in the presence of uncertainty. 2. The structuring of the planner provides independence of representation of the underlying audit trail syntax. 3. Furthermore, this approach has the potential of reducing substantial amounts of processing per audit record.	1. It places burden on the person creating the intrusion detection models to assign meaningful and accurate numbers to various parts of the graph representing the intrusion model [10]. 2. The patterns for intrusion scenarios must be easily recognized. 3. The patterns must not be associated with any other normal behavior [10].	

2.3 Introduction to Wireshark

Packet Sniffer is used to observe the messages exchanged between executing protocol entities. A packet sniffer captures messages being sent or received from or by the computer. It will also typically store and display the contents of the various protocol fields in these captured messages. A packet sniffer is passive. As a sniffer, Wireshark is a very useful and powerful network packet analyzer. A network packet analyzer will try to capture network packets and display that packet data as detailed as possible. In the past, these tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all this has changed. Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix and Mac computers. It is an ideal packet analyzer.

2.3.1 Features of Wireshark

Some of the features that Wireshark provides are:

• It is available for UNIX and Windows.

• Capture live packet data from a network interface.

• Display packets with detailed protocol information.

• Open and Save packet data captured.

• Import and Export packet data from and to a lot of other capture programs.

• Filter packets on many criteria.

• Search for packets on many criteria.

2.3.2 Disadvantages of Wireshark

Wireshark does not provide:

• Wireshark is not an intrusion detection system. It will not warn when someone does strange things on the network that are not allowed to do.

• Wireshark will not manipulate things on the network, it will only "measure" things from it.

2.4 Components of Wireshark

When Wireshark program is run and packets are captured, the Wireshark graphical user interface as shown in Figure 2.3 will be displayed. Initially, no data will be displayed in the various windows.

The Wireshark interface has five major components:

• **Command menus:** These are standard pull down menus located at the top of the window. Mainly used are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data and exit the Wireshark application. The Capture menu allows to begin packet capture.

• **Packet-contents window:** It displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

• **Packet display filter field**: It is a field into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window.

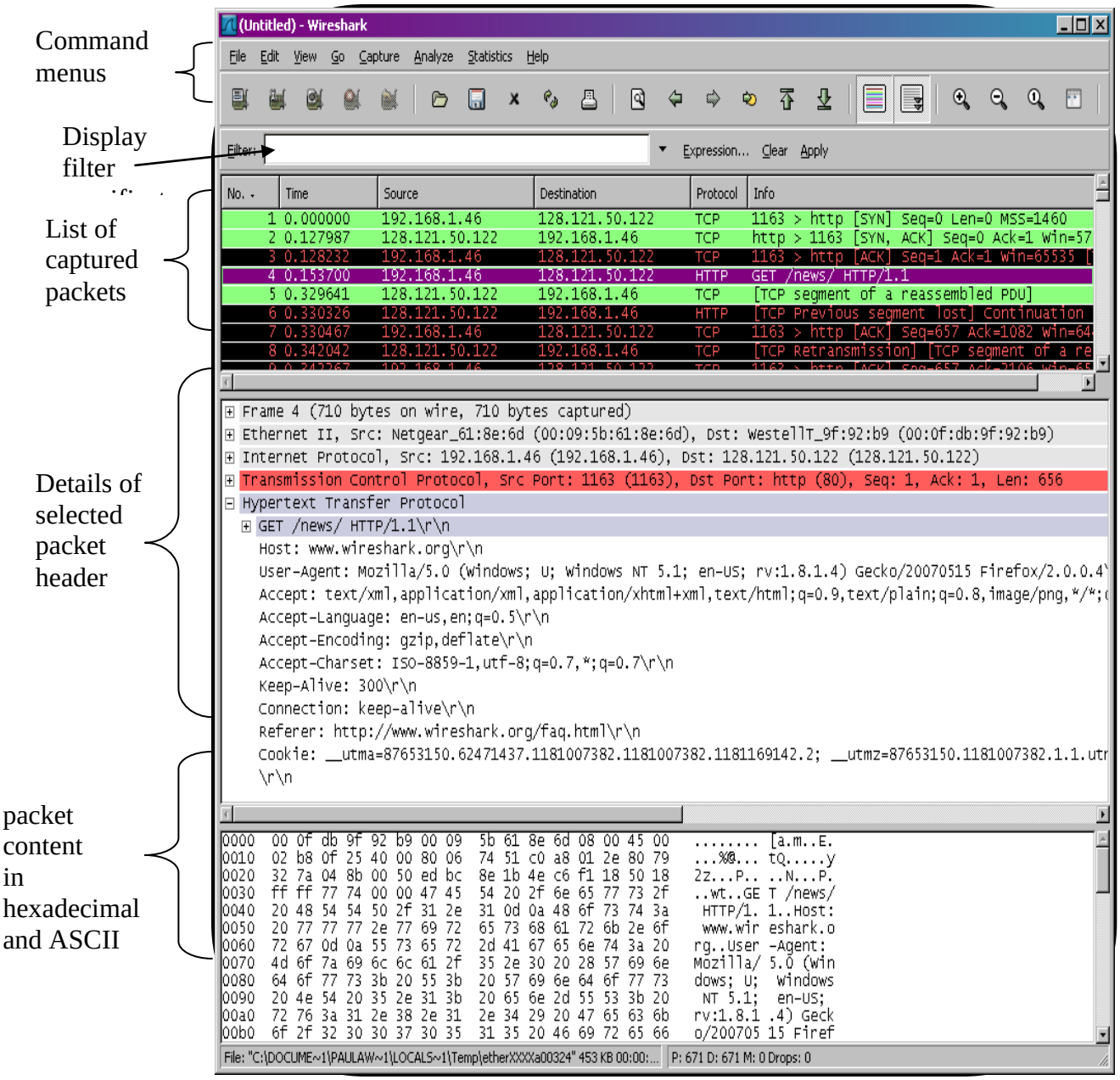


Figure 2.3: Wireshark Graphical User Interface [38].

- **Packet-listing window:** It display a one-line summary for each packet captured, including the packet number, the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The protocol type field lists the highest level protocol that sent or received this packet.

- **Packet-header details window:** It provides details about the packet selected in the packet listing window. These details include information about the Ethernet frame (assuming the packet was sent or received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the plus-or-minus boxes to the left of the Ethernet frame or IP datagram line in the packet details window.

2.5 Bro: An Intrusion Detection System

Bro is originated as a research system. It is designed and developed by Vern Paxson of ICSI Center for Internet Research (ICIR), Berkeley. The project is started in 1995 at Lawrence Berkeley National Laboratory (LBL), and Bro is under active development since then. It was first published in 1998. Major extensions of Bro are contributed by people at the University of California, Berkeley and Princeton University, New Jersey. Among other locations, Bro is operationally deployed at the University of California, Berkeley and at LBL. Our group at TU Munich, Germany has used and extended Bro since 2001.

Bro is a Unix-based Network Intrusion Detection System (IDS). Bro monitors network traffic and detects intrusion attempts based on the traffic characteristics and content. Bro Detects intrusions by comparing network traffic against rules describing events that are deemed troublesome [23]. These rules might describe activities (e.g., certain hosts connecting to certain services), what activities are worth alerting (e.g., attempts to a given number of different hosts constitutes a "scan"), or signatures describing known attacks or access to known vulnerabilities. If Bro detects something of interest, it can be instructed to either issue a log entry or initiate the execution of an operating system command. Bro targets high-speed (Gbit/second), high-volume intrusion detection. By judiciously leveraging packet filtering techniques, Bro is able to achieve the performance necessary to do so while running on commercially available PC hardware and thus can serve as a cost effective means of monitoring a site’s Internet connection [34].

Bro lends itself particularly well to forensic tasks due to its great data collection and analysis capabilities. Bro is a signature-based IDS, meaning that it attempts to match a signature to network traffic in order to find the attack. Bro is unique in that it utilizes regular expressions, rather than fixed strings to understand network activity. The creators of Bro translate this to

mean that a lower false-positive rate is achieved due to Bro being able to understand the context of the traffic, rather than merely matching a static signature. Bro also comes with its own language which advanced users can utilize to program policy scripts. Policy scripts allow network administrators to fine-tune their Bro installation in order to specifically search out certain types and patterns of traffic, and define them as malicious. Further, developers can extend Bro capabilities by having scripts execute in certain events to block, alert or log information about certain network traffic. Some of Bro's biggest shortfalls are that it only reports information to log files and does not have a graphical user interface (GUI). Log files are designed in such a way that humans can understand them and computers can easily parse them. The option to report events to a database might be nice in some cases, especially for long-term storage of data but it is not an absolute necessity for Bro. The lack of a GUI is understandable given Bro preference towards forensics and analysis rather than intrusion prevention techniques [19].

Two systems extensively explored under IDS are: Snort and Bro. Snort is the only one widely deployed in real networks. It appears that most people are not aware of alternatives. This is interesting for three reasons:

(i) Snort shows some technical limitations that indicate that it is not always an optimal solution.

(ii) In areas of security, there is usually a variety of open source software available.

(iii) There is at least one other very powerful open source system that is largely unknown: termed as Bro [33].

2.5.1 Features of Bro NIDS

Bro is a stand-alone system used to detect network intruders in real time by passively monitoring the network link. Bro provides a set of characteristics that make it a powerful intrusion detection system.

- **Network based**

  Bro is a network-based IDS. It collects filters and analyzes traffic that passes through a specific network location. A single Bro monitor, strategically placed at a key network junction, can be used to monitor all incoming and outgoing traffic for the entire site. Bro does not use or require installation of client software on each individual, networked computer.

- **Rich application-layer analysis**

  A primary feature of Bro is that it includes detailed, parser-driven analysis of many popular application protocols. The output of these analyzers is a stream of events that describe observed activity in semantically rich, high-level terms. These events themselves do not constitute security alerts, but rather provide the input for further, stateful processing using Bro's custom scripting language.

- **Custom scripting language**

  Bro policy scripts are programs written in the Bro language. They contain all the "rules" that describe what sorts of activities are deemed troublesome. They analyze the network activity and initiate actions based on the analysis. Although the Bro language takes some time and effort to learn but once mastered, the Bro user can write or modify Bro policies to detect and alert on virtually any type of network activity.

- **Powerful signature matching facility**

  Bro policies incorporate a signature matching facility that looks for specific traffic content. For Bro, these signatures are expressed as regular expressions, rather than fixed strings. Bro adds a great deal of power to its signature-matching capability because of its rich language. This allows Bro to not only examine the network content, but to understand the context of the signature, greatly reducing the number of false positives. Bro comes with a set of high value signatures policies, selected for their high detection and low false positive characteristics.

- **Network traffic analysis**

  Bro not only looks for signatures, but can also analyze network protocols, connections, transactions, data amounts and many other network characteristics. It has powerful facilities for storing information about past activity and incorporating it into analyses of new activity.

- **Detection followed by action**

  Bro policy scripts can generate output files recording the activity seen on the network (including normal, non-attack activity). They can also generate problem alerts to event logs, including the operating system syslog facility. In addition, scripts can execute programs, which can in turn, send e-mail messages, page the on-call staff, automatically terminate existing connections, or, with appropriate additional software, insert access control blocks into a router's access control list. With Bro's ability to execute programs at the operating system level, the actions that Bro can initiate are only limited by the computer and network capabilities that support Bro.

- **Snort compatibility support**

  The Bro distribution includes a tool, snort2bro, which converts Snort signatures into Bro signatures. Along with translating the format of the signatures, snort2bro also incorporates a large number of enhancements to the standard set of Snort signatures to take advantage of Bro's additional contextual power and reduce false positives.

2.5.2 Design Goals of Bro IDS

To achieve the functionality of Bro in a desired manner, the design of Bro has been guided by a set of goals. These goals must be achieved to detect the intrusions effectively by Bro IDS.

- **High speed, large volume monitoring:**

For an environment, the greatest sources of threats are the external hosts connecting to the hosts over the Internet. The network needed to protect has a single link connecting it to the remainder of the Internet (“DMZ”), greatest potential source of attacks can be monitored by passively watching the DMZ link [4].

- **No packet filter drop:**

If an application using a packet filter cannot consume packets as quickly as they arrive on the monitored link, then the filter will buffer the packets for later consumption. However, eventually

the filter will run out of buffer, at which point it drops any further packets that arrive. From a security monitoring perspective, drops can completely defeat the monitoring, since the missing packets might contain exactly the interesting traffic that identifies a network intruder.

- **Real time notification:**

One of the main dissatisfactions with the initial off-line system was the lengthy delay incurred before detecting an attack. If an attack, or an attempted attack, is detected quickly, then it can be much easier to trace back the attacker minimize damage, prevent further break-ins, and initiate full recording of all of the attacker's network activity. Therefore, one of requirement for Bro was that it detect attacks in real-time.

- **Extensible:**

Because there are an enormous number of different network attacks, with who knows how many waiting to be discovered, the system clearly must be designed in order to make it easy to add to it knowledge of new types of attacks [4].

- **Avoid simple mistakes:**

Of course, we always want to avoid mistakes. However, here to mean that a site defines its security policy is both clear and as error-free as possible.

2.5.3 Architecture of Bro

Architecturally, Bro is layered into three major components as shown in Figure 2.4. It uses the standard libpcap packet-capture library to filter the packet stream. The filtered packets are then passed along to Bro event engine, which reduces the filtered stream into a series of higher-level events. These events reflect network activity in policy-neutral terms. Finally, Bro policy script interpreter executes event handlers written in the custom Bro scripting language.

- **Network:** The lower-most layers process the greatest volume of data, and hence must limit the work performed to a minimum. As we go higher up through the layers, the data stream diminishes, allowing for more processing per data item.

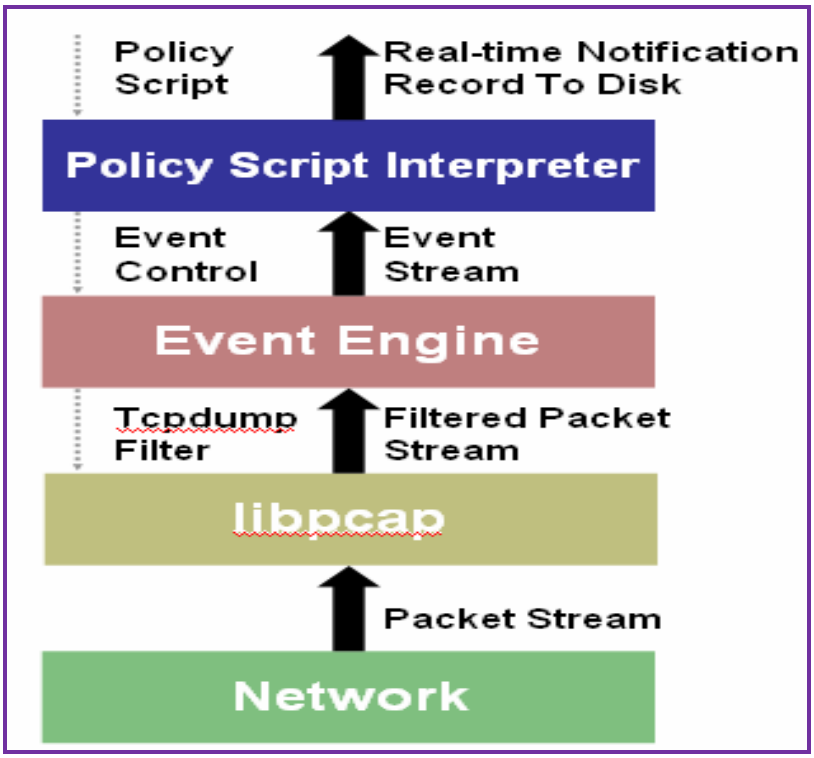


Figure 2.4: Structure of Bro system [6]

- **Libpcap:** Just above the network is libpcap, the packet-capture library used by tcpdump. Using libpcap gains significant advantages: it isolates Bro from details of the network link technology; it greatly aids in porting Bro to different UNIX variants as it becomes. If the host operating system provides a sufficiently powerful kernel packet filter, then libpcap downloads the filter used to reduce the traffic into the kernel. The key to packet filtering is, of course, judicious selection of which packets to keep and which to discard.
- **Event engine:** The resulting filtered packet stream is then handed up to the next layer, the Bro "event engine." This layer first performs several integrity checks to assure that the packet headers are well-formed, including verifying the IP header checksum. If these checks fail, then Bro generates an event indicating the problem and discards the packet. It is also at this point that Bro reassembles IP fragments so it can then analyze complete IP datagram. If the checks succeed, then the event engine looks up the connection state associated with the tuple of the two IP addresses and the two TCP or UDP port numbers,

creating new state if none already exists. It then dispatches the packet to a handler for the corresponding connection [6]

- **Policy script interpreter:** After the event engine has finished processing a packet, it then checks whether the processing generated any events (These are kept on a FIFO queue). It processes each event until the queue is empty. It also checks whether any timer events have expired, and if so processes them, too. A key fact of Bro design is the clear distinction between the generations of events versus what to do in response to the events. This structure reflects the separation between mechanism and policy. The "policy script interpreter" executes scripts written in the specialized Bro language. These scripts specify event handlers, which are essentially identical to Bro functions except that they do not return a value. For each event passed to the interpreter, it retrieves the semi-compiled code for the corresponding handler, binds the values of the events to the arguments of the handler, and interprets the code. This code in turn can execute arbitrary Bro scripting commands, including generating new events, logging real-time notifications, recording data to disk, or modifying internal state for access by subsequently invoked event handlers.

  The major tasks of policy script interpreter are:

  • For each event passed to the interpreter it retrieves compiled code for the corresponding handler and binds values of the events to the arguments of the handler and interprets the code.

  • That code can in turn generate new events, log notifications, record data to disk, or modify the current state.

2.5.4 Attacks on the Monitor

A number of attacks that attempt to subvert monitoring systems and defensive strategy used against these attacks are explained in this section.

- **Overload attacks**

  An attack is overload if the goal of the attacker is to overburden the monitor to the point where it fails to keep up with the data stream it must process. The attack has two phases, the first in which the attacker drives the monitor to the point of overload, and the second

in which the attacker attempts a network intrusion. Defensive strategy is for the monitor to shed load when it becomes unduly stressed. Of course, if the attacker knows the form of load-shedding used by the monitor, then they can exploit its consequent blindness and launch a now-undetected attack. Finally, to help defend against overload attacks, the event engine periodically generates a net stats update event. The value of this event gives the number of packets received, and the number reported dropped by the network interface because the kernel failed to consume them quickly enough [26].

- **Crash attacks**

  Crash attacks aim to knock the monitor completely out of action by causing it to either fault or run out of resources. The crash attack has two phases, the first during which the attacker crashes the monitor, and the second during which they then proceed with an intrusion. Crash attacks can be much more subtle than overload attacks, though. By careful source code analysis, it may be possible to find a series of packets, or even just one, that, when received by the monitor, causes it to fault due to a coding error. The effect can be immediate and violent. To defend against this form of crash attack by careful coding and testing.

- **Subterfuge attacks**

  In a subterfuge attack, an attacker attempts to mislead the monitor as to the meaning of the traffic it analyzes. These attacks are particularly difficult to defend against, because unlike overload and crash attacks, if successful they do not leave any traces that they have occurred.

  To thwart subterfuge attacks, at each stage Bro analyze the explicit and implicit assumptions made by the system and how, by violating them, an attack might successfully elude detection [26].

Chapter 3
Problem Statement

An intrusion detection system is used to monitor the network traffic, check for suspicious activities and notifies the system or network administrator. Bro is one of the most effective IDS which can be used to detect these threats. Bro monitors network traffic and detect intrusion attempts based on the traffic characteristics and content.

Policy scripts in Bro allow network administrators to fine-tune their Bro installation in order to specifically search out certain types and patterns of traffic, and define them as malicious. However, the policy scripts are few in number and there are various types of traffic that are still not captured by Bro IDS. Moreover, policy scripts that are executed to block, alert or log information about network traffic are yet needed to be explored. Much of the work has already been done to capture and filter traffic of various network classes. The network traffic of classes like SMTP, FTP is further needed to be worked on.

Currently, Bro biggest shortfall is that it only report information to log files and does not have a Graphical User Interface (GUI). Till now the traffic is captured by using the commands which is also an overhead for users. So a GUI framework is required that could capture traffic more effectively and easily. Also, a framework is needed that can automate the creation of policy scripts and run those scripts to capture and filter the network traffic.

In this dissertation, the objective is to:

1. Design a policy scripts to filter out the needed packets from traced traffic and alert the user about the captured traffic.

2. Implement a policy script to count the number of connections established by each local host.

3. Integrate a GUI framework in Bro that analyze and filters the traced network traffic. This eliminates the need of writing the commands at terminal and makes it easy for users to create the scripts and run them on captured traffic.

In this way by using the proposed scripts and GUI framework, the network traffic can be easily examined. Hence it provides a security system that captures network traffic and analyzes it for possible hostile attacks.

Chapter 4

Implementation Details and Experimental Results

4.1 Introduction

In this chapter, policy scripts are designed to filter out the needed packets from traced traffic and generate alert on desired incoming network packets. The term script here refers to a set of event handlers and related functions and variables written in the Bro language. A Bro GUI framework is developed that analyze and filter the traced network traffic. Bro GUI Framework creates the scripts and run them on the live traffic to get the desired output. The Bro GUI Framework is shown in Figure 4.1.

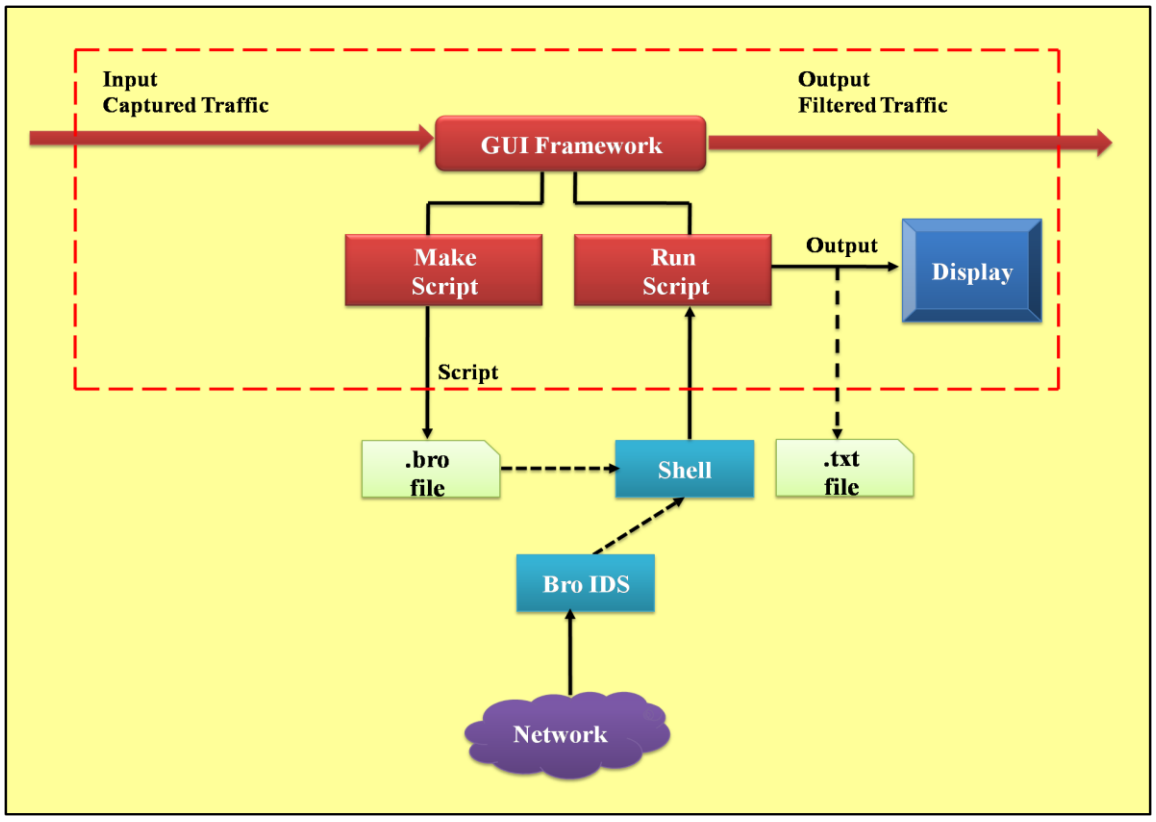


Figure 4.1: Bro GUI Framework.

4.2 Steps followed to Run Bro GUI Framework

Input: Live traffic

Step1: Choose the network protocol whose packets are to be filtered.

Step 2: Create the desired policy script using Bro GUI Framework.

Step 3: Run the script on the live traffic by using Bro GUI Framework.

Step 4: Filtered traffic is displayed as output and saved in a file.

Output: Filtered Traffic.

4.3 Components of Bro GUI Framework

Bro GUI Framework is used to automate the creation and execution of policy scripts. There are basically three components that make Bro GUI Framework. The components are listed below as:

4.3.1 Network Traffic

Network traffic comprises of various protocols like HTTP, ARP, TCP, SNMP etc. These protocol packets are captured and analyzed to detect the intrusions effectively. In this thesis, work has been done on the packets of SMTP, FTP, TCP, HTTP protocols. The network traffic is captured and these protocol packets are filtered in order to alert user about the usage of these packets. Further this filtered traffic can be analyzed and examined. The live traffic needed for experimentation is captured by running the following set of commands as:

- *bro –i etho file1 > fille2*

  This command is used to analyze the network traffic from an interface. To capture the live traffic using Bro, *–i* option is need to be specified. Instead, to read the data captured in a file, an option *–r* is used at the place of –i. *eth0* specifies the interface receiving the network traffic. So, it is recommended to replace it by the appropriate interface name. *file1* here refers to the bro script created by the Make Script Form. This script is saved with an extension of .bro. *file2* specifies the name of output file where filtered data is saved.

- *clear*

  Like clrscr() function used on windows platforms, *clear* command is used to clear the terminal on Linux platform. After capturing the live traffic screen is cleared by running the *clear* command.

- *broctl stop*

  After screen is cleared, *broctl stop* command is run to stop all the working processes attached with Bro.

These commands help to trace and examine the filtered traffic.

4.3.2 Creation of Policy Scripts

Policy scripts in Bro allow network administrators to fine-tune their Bro installation in order to specifically search out certain types and patterns of traffic, and define them as malicious. Different policy scripts are created for different types of protocols. The policy scripts in Bro GUI Framework are designed to filter live traffic.

Depending on the traffic to be filtered, various parameters are concatenated to make the policy scripts. Three parameters are discussed based on which various policy scripts are created and differentiated.

- **Analyzer**

Bro policy script is the basic analyzer used by Bro to determine what network events are alarm worthy. A policy can also specify what actions to take and how to report activities, as well as determine what activities to scrutinize. The simplest way to use an analyzer is to *@load* the standard script associated with the analyzer. Policy files are loaded by using an *@load* command. In general, the work associated with a particular analyzer is done if policy script defines one or more event handlers associated with the analyzer. For example, Bro will instantiate an FTP analyzer only if policy script defines an ftp_request or ftp_reply handler. If it does not, then when a new FTP connection begins, Bro will only instantiate a generic TCP

analyzer for it. The semantics of @load are "load in this script if it hasn't already been loaded", so there is no harm in loading something in multiple policy scripts.

Some policy files loaded are:

@load tcp: Initialize filter for SYN/FIN/RST TCP packets.

@load http: General HTTP analyzer.

@load smtp: Record and analyze email traffic.

@load ftp: Load FTP packets for analysis.

Some of the common parameters essentially loaded in all the scripts are:

@load weird: Initialize generic mechanism for detecting unusual events.

@load alarm: Open logging file for alarm events.

- **Event handler**

Specifies name of the handler in Bro language based on which filtration is done. A list of built in handlers is provided in Bro IDS. Some of them are:

a) The tcp analyzer processes traffic associated with SYN/FIN/RST TCP packets. Bro instantiates an tcp analyzer providing event handler *tcp_packet* is defined.
b) The ftp analyzer processes traffic associated with the FTP file transfer service. Bro instantiates an ftp analyzer for any connection with service port 21/tcp, providing event handler *ftp_request* is defined.
c) The smtp analyzer processes traffic associated with the email traffic. Bro instantiates an smtp analyzer for any connection with service port 25/tcp, providing *smtp_request* event handler is defined.
d) The http analyzer processes traffic associated with http packets. Bro instantiates an http analyzer providing event handler *http_request* is defined.

- **Arguments defined in handler:**

Bro contain a large list of arguments defined in event handler. A set of arguments is needed to be defined in the event handler. They help to define data types of various parameters like length, count, sequence, name etc Depending on the event handler, list of arguments is selected. Some event handlers contain few while some have a large list of argument list. Various Scripts contain some arguments in common. Some of the arguments used are *c: connection, is_orig: bool, command: string, arg: string, flag: string, seq: count, ack: count.*

The protocol whose packets are to be filtered is chosen. Based on the selected protocol, arguments are marked and a script is created. The script is saved with a .bro extension. The corresponding script is applied on the live traffic. The filtered traffic is the output and hence, this filtered traffic can be used to detect the intrusions. The Bro GUI Framework automates the process of creation of scripts.

4.3.3 Executing Policy Scripts

The policy scripts created in make script are run on the live traffic in order to get the desired filtered traffic of respective protocol. On selecting the appropriate option, the output i.e filtered traffic is generated. The filtered output is shown on the screen so that users can analyze it and also the filtered traffic is saved in a file on the pre-decided path so that it can be examined for future reference. The filtered traffic e.g. FTP Filtered traffic that serves as output is shown in Figure 4.2

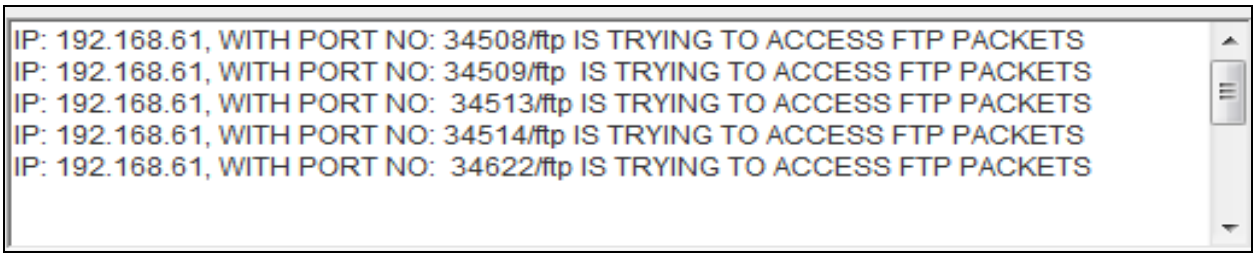


Figure 4.2: Filtered Traffic.

4.4 Proposed Policy Scripts

Various policy scripts are designed to filter out the needed packets from traced traffic. In this section, traffic of SMTP, HTTP, FTP, TCP protocol is filtered and analyzed.

4.4.1 Script to filter SMTP packets from the captured traffic

Bro instantiates a smtp [31] analyzer to processes the traffic associated with the email service. smtp analyzer listen on port 25, providing the appropriate event handler is defined. This policy script uses smtp as analyzer, smtp_packet as event handler and c: connection, is_orig: bool, command: string, arg: string as arguments used in handler. The policy script to filter SMTP traffic is written as:

smtp_sample.bro script:

```
@load weird
@load alarm
@load smtp
global path: string;
redef ignore_checksums= T;
event smtp_request(c: connection, is_orig:  bool, command: string, arg: string)
{
    print fmt ("IP: %s, WITH PORT NO: %s IS TRYING TO ACCESS SMTP
            PACKETS", c$id$orig_h, c$id$orig_p);
}
```

4.4.2 Script to filter HTTP packets from the captured traffic

Bro instantiates a http [12] analyzer to processes the traffic associated with the HTTP protocol. http analyzer listen on port 80, providing the appropriate event handler is defined. This policy script uses http as analyzer, http_request as event handler and c: connection, method: string, original_URI: string, unescaped_URI: string, version: string as arguments used in handler.

http_sample.bro script:

```
@load weird
@load alarm
@load http
global path: string;
redef ignore_checksums= T;
```

```
event http_request(c: connection, method: string, original_URI: string, unescaped_URI:
string, version: string)
{
        print fmt ("IP: %s WITH PORT NO:%s IS TRYING TO ACCESS HTTP
                PACKETS" ,c$id$orig_h ,c$id$orig_p);
}
```

4.4.3 Script to count the number of connections established by each local host.

Script to count the number of connections established by each local host is specified below as:

count.bro script:

```
@load weird
@load alarm
@load tcp
global hosts: table[addr] of count &default=0;
event connection_established(c: connection)
  {
        local orig = c$id$orig_h;
        if ( ! is_local_addr(orig) )
              return;
       ++hosts[orig];
  }
event bro_done()
  {
        for ( h in hosts )
          print h, hosts[h];
}
```

4.4.4 Script to filter TCP packets from the captured traffic

Bro instantiates a tcp analyzer to processes the traffic associated with the TCP protocol. tcp analyzer listen on the port, providing the appropriate event handler is defined. This policy script

uses tcp as analyzer, tcp_packet as event handler and c: connection, is_orig: bool, flags: string, seq: count, ack: count, len:count, payload: string as arguments used in handler.

tcp_sample.bro script:

```
@load weird
@load alarm
@load tcp
event tcp_packet(c: connection, is_orig: bool, flags: string, seq: count, ack: count, len: count, payload: string)
{
        print fmt("IP : %s WITH PORT NO:%s IS TRYING TO ACCESS TCP
                PACKETS" ,c$id$orig_h ,c$id$orig_p);

}
```

4.4.5 Script to filter FTP packets from the captured traffic

Bro instantiates a ftp analyzer to processes the traffic associated with the FTP file transfer service. ftp analyzer listen on the port 21, providing the appropriate event handler is defined. This policy script uses ftp as analyzer, ftp_request as event handler and c: connection, command: string, arg: string as arguments used in the handler.

ftp_sample.bro script:

```
@load weird
@load alarm
@load http-request
redef ignore_checksums= T;
event ftp_request (c: connection, command: string, arg: string)

{

        print fmt("IP ADDRESS: %s WITH PORT NO:%s IS TRYING TO ACCESS
                FTP PACKETS" ,c$id$orig_h ,c$id$orig_p);
```

```
}
```

4.4.6 Script to display source and destination address of the captured traffic

The IP address and port number of the source and destination machines are displayed as output by using this script. These addresses help user to analyze and examine the network intrusions. The analyzer used is http and event handler is http_header. The arguments selected are c : connection, is_orig: bool, name: string, value: string.

ip.bro script:

```
@load weird
@load alarm
@load http
global path: string;
redef ignore_checksums= T;
event http_header(c: connection, is_orig: bool, name: string, value: string)
{
        print fmt ("CONNECTION IS: %s: %s->%s: %s", c$id$orig_h, c$id$orig_p,
                c$id$resp_h, c$id$resp_p);

        }
```

4.5 Bro GUI Framework Implementation

Bro is a UNIX based Network Intrusion Detection System. Moreover, there is no GUI Framework for Bro IDS. The traffic is captured by using the commands which is an overhead for user. So a GUI framework is required that could capture traffic more effectively and easily. In this dissertation, Bro GUI Framework is implemented to get the filtered traffic. The Bro GUI Framework is developed in JDK 1.7 environment. The input to the Bro GUI Framework is live traffic. The scripts designed using Bro GUI Framework are saved in a file having .bro extension. The live traffic is run on the scripts. The output is the filtered traffic. The filtered output is shown on the screen so that users can analyze it and also the filtered traffic is saved in a file on the pre-decided path so that it can be examined for future reference.

4.6 Bro GUI Framework Snapshots

A Graphical User Interface called Bro GUI Framework is designed. GUI helps to make and run the policy scripts in easy and efficient manner. Sections contain information about bro and define sample scripts so that a user can understand execution of policy scripts and Bro GUI Framework in an easy manner.

4.6.1 Start-up Form

Snapshot 4.1 shows the start-up form. It is basically divided into two modules namely Bro Script and Read Me. Bro Script is further divided into sub modules namely Make Script and Run Script.

Snapshot 4.1: Start-up Form.

Make Script sub module is use to create scripts and Run Script is used to run the created script on the live traffic. The module Read Me is further divided into two sub modules called About Bro and Sample Scripts. About bro contain details about Bro and Sample scripts act as user guide. The structural relationship among modules is shown in Figure 4.3.

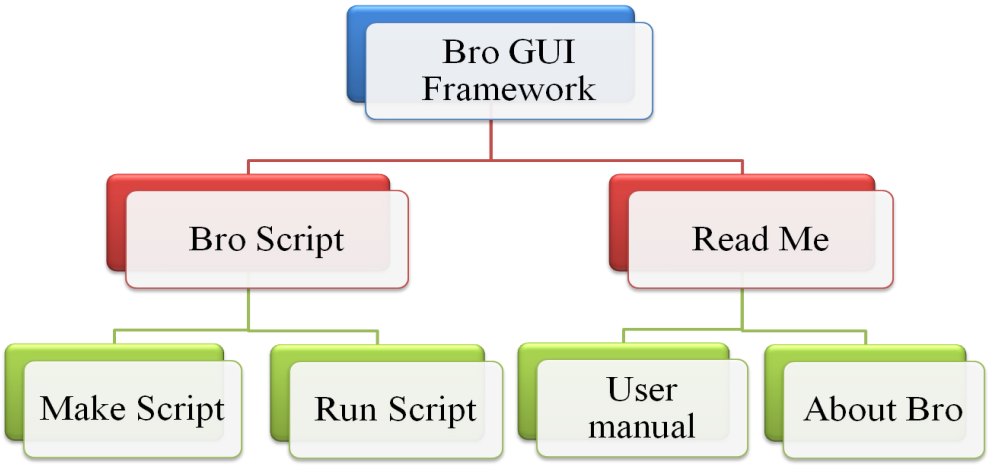
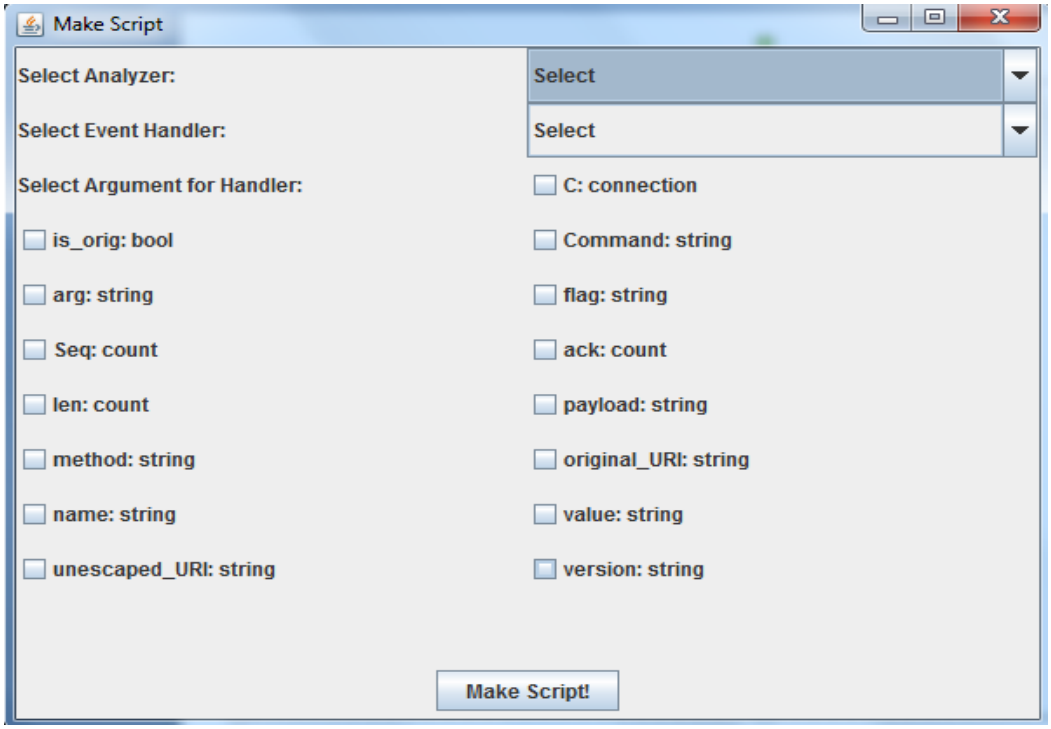


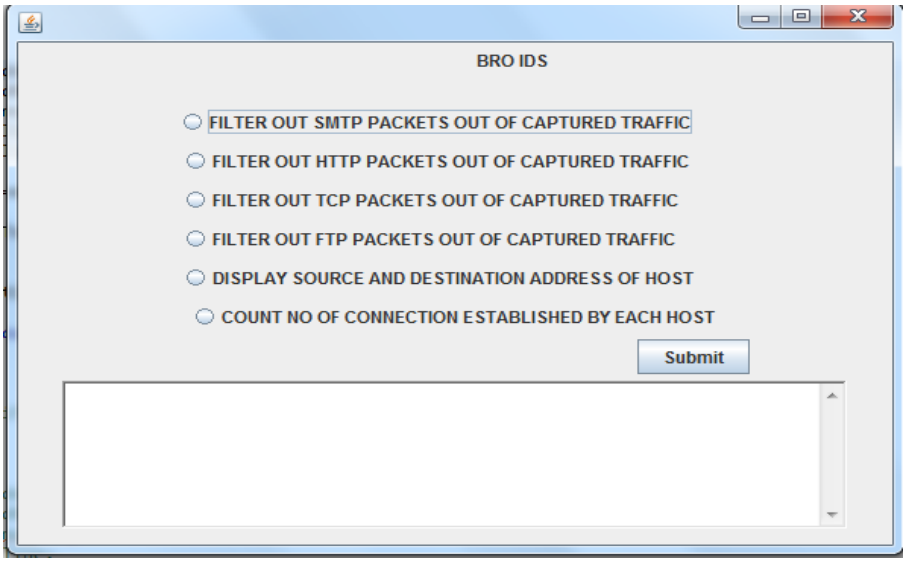
Figure 4.3: Tree Hierarchy of Bro GUI Framework.

4.6.2 Make Script: A form called Make Script is designed. Various combo boxes and check boxes are provided that contain different arguments and event handlers necessary for the scripts. All the data chosen from these boxes is added to the file after concatenation to make the script in a dynamic manner. After clicking on Make Script! button, desired policy script is saved in the form of a file on the pre-decided path. Snapshot 4.2 shows the necessary parameters for making the script.



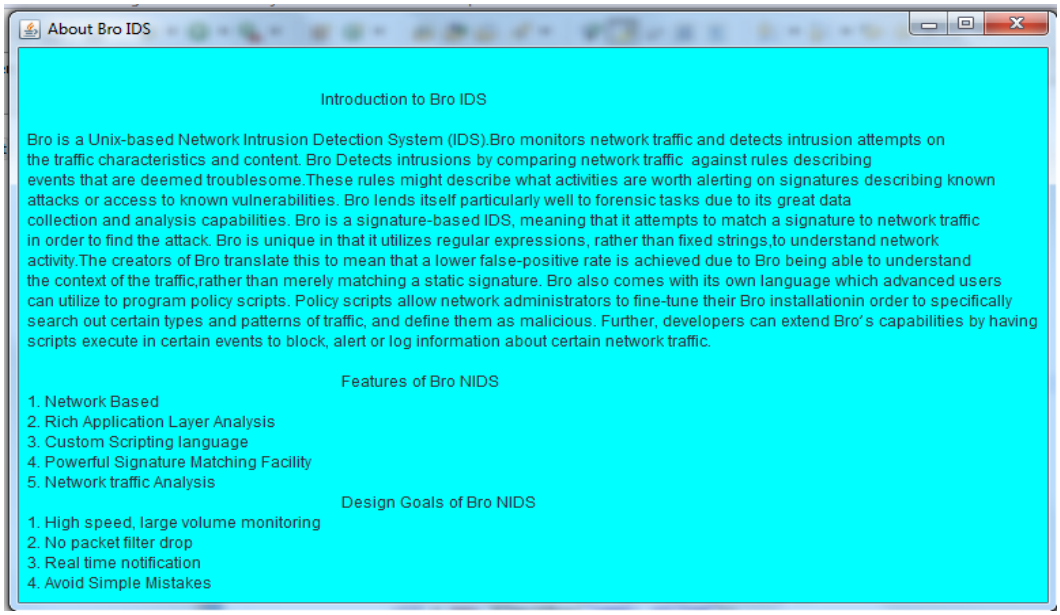
Snapshot 4.2: Make Script Form.

4.6.3 Run Script: This form makes the execution of scripts graphical, i.e by clicking a button. It eliminates the need of writing the command. Various buttons are built and on selecting the appropriate choice, output of corresponding script is displayed. The Run Script Form and available options are shown in Snapshot 4.3.



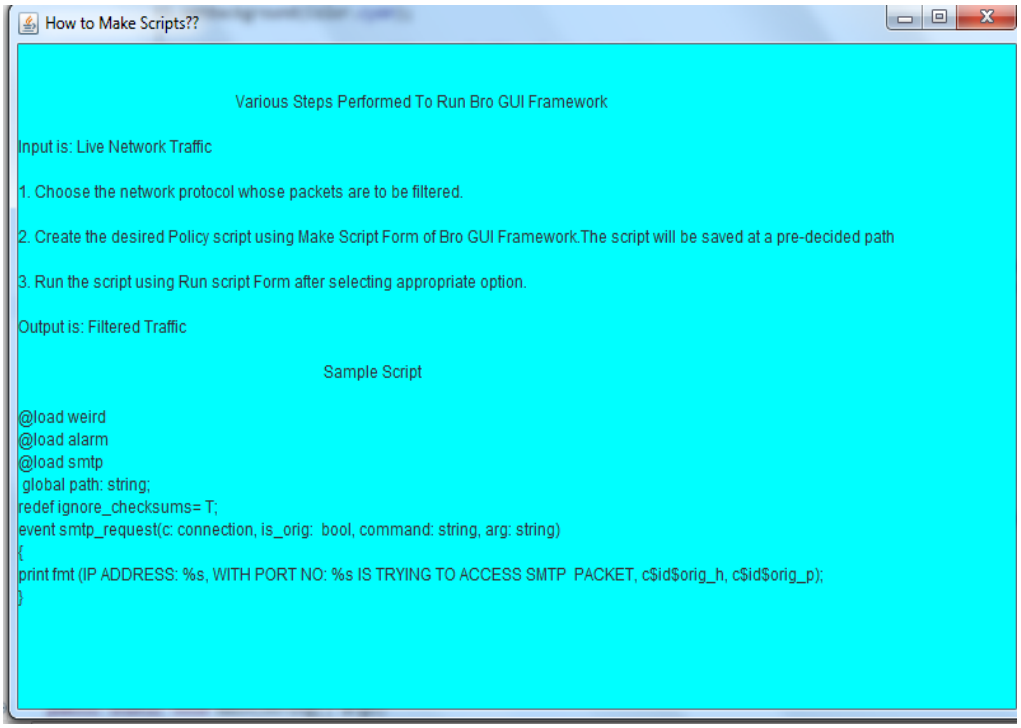
Snapshot 4.3: Run script Form.

4.6.4 About Bro IDS: A form About Bro IDS is displayed in Snapshot 4.4 that contain details of Bro. Features and design Goals of Bro are listed to give user a quick and better understanding of Bro. This acts as a user manual.



Snapshot 4.4: About Bro IDS Form.

4.6.5 Sample Scripts: In this form, steps needed to implement policy scripts are written and a sample script has been shown as an example as shown in Snapshot 4.5. This form is basically designed to give user an understanding of scripts. It help user to learn the way scripts are written.



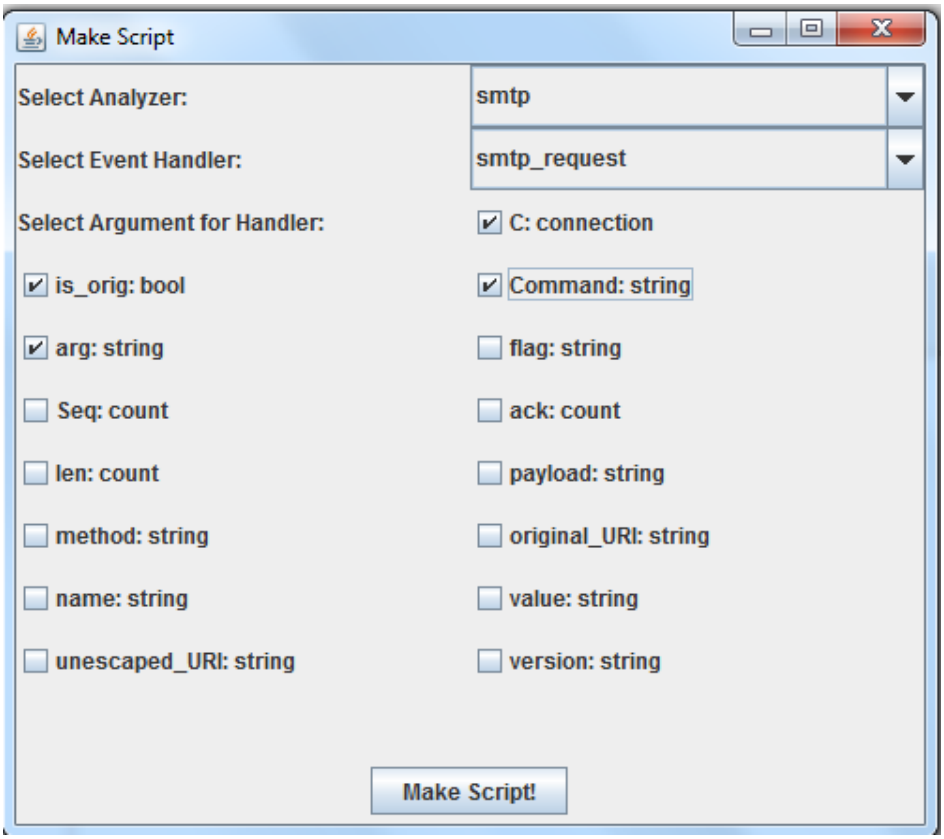
Snapshot 4.5: Sample Script Form.

4.7 Experimental Results

Various policy scripts are created and run by using Bro GUI Framework. These scripts help to filter the captured traffic in desired manner. The implementation and experimental result of policy scripts are shown in this section.

4.7.1 SMTP Script:

SMTP policy script is created by using Bro GUI framework. The SMTP policy script uses smtp as analyzer, smtp_packet as event handler and c: connection, is_orig: bool, command: string, arg: string as arguments used in handler as shown in Snapshot 4.6. Bro uses smtp analyzer to process the traffic associated with the email service coming on port no 25.



Snapshot 4.6: Make of SMTP Script.

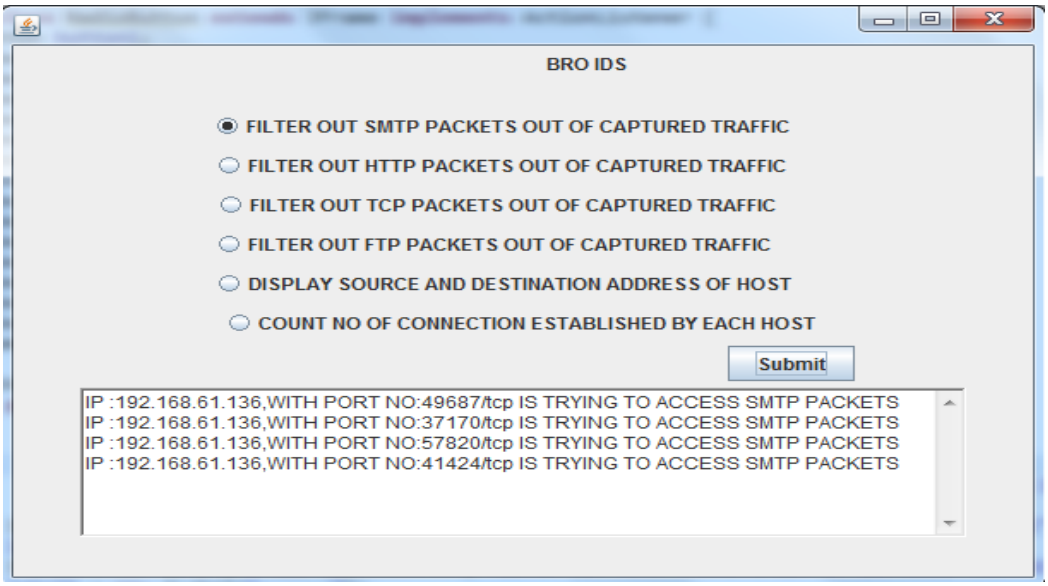
On clicking the Make Script! button, a file called out.bro as shown in Snapshot 4.7 is created at the pre- decided path.

```
@load weird
@load alarm
@load smtp
global path: string;
redef ignore_checksums=T;
event smtp_request(c: connection, is_orig: bool, command: string, arg: string)
{
print fmt("IP:%s,WITH PORT NO:%s IS TRYING TO ACCESS SMTP PACKETS",c$id$orig_h,c$id$orig_p);

}
```

Snapshot 4.7: File containing SMTP Script.

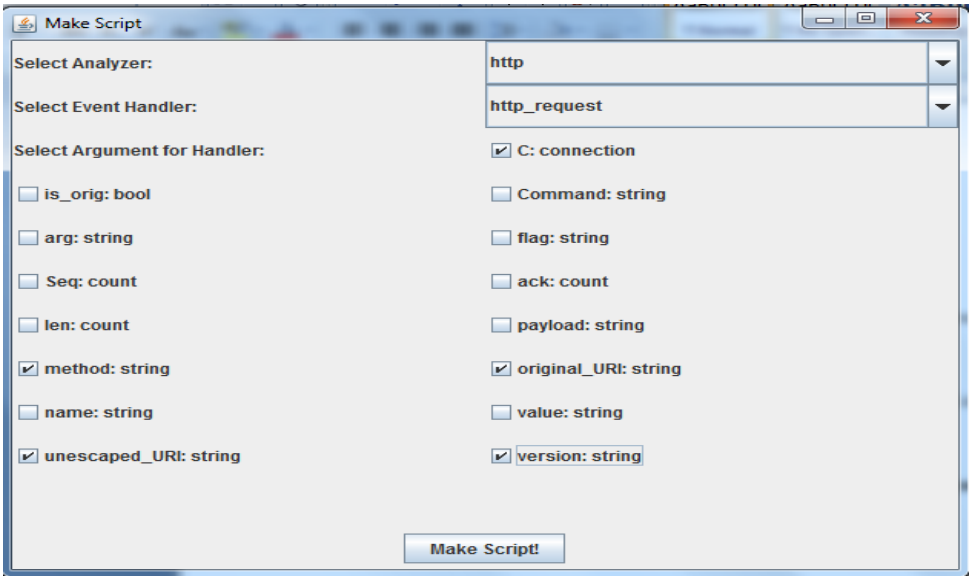
After clicking on desired option, i.e group button 1, filtered traffic containing only SMTP protocol packets are displayed as shown in Snapshot 4.8.



Snapshot 4.8: Run of SMTP Script.

4.7.2 HTTP Script

HTTP policy script uses http as analyzer, http_request as event handler and c: connection, method: string, original_URI: string, unescaped_URI: string, version: string as arguments used in handler. This is shown in Snapshot 4.9. Bro uses http analyzer to process the traffic associated with the http protocol coming on port no 80.



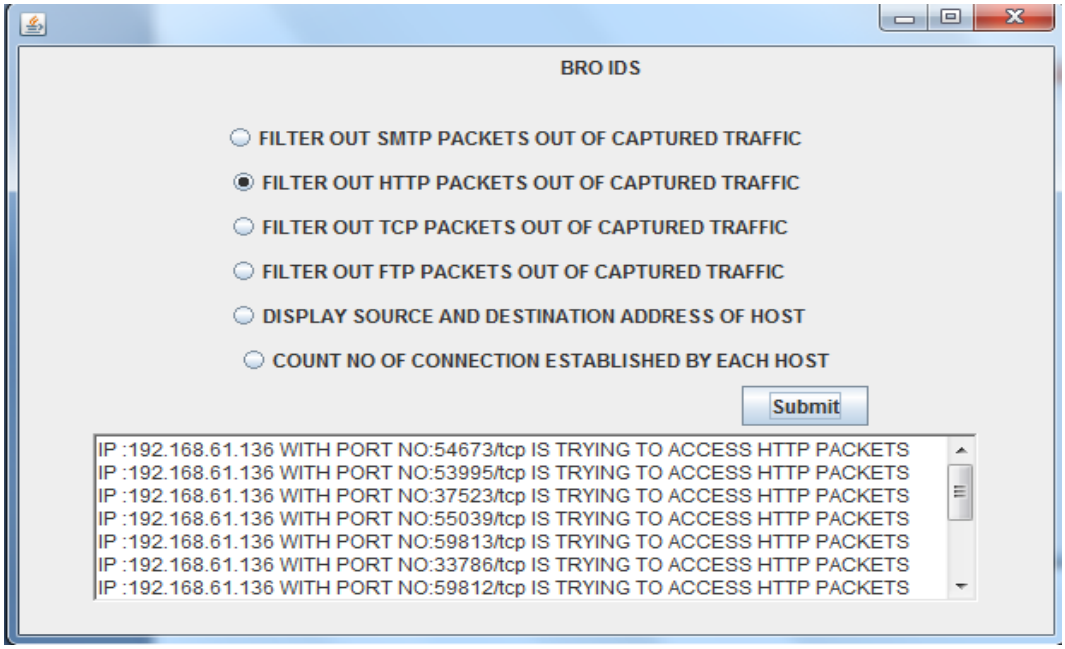
Snapshot 4.9: Make of HTTP Script.

On clicking the Make Script! button, a file called out.bro as shown in Snapshot 4.10 is created at the pre- decided path.

```
@load weird
@load alarm
@load http
global path: string;
redef ignore_checksums= T;
event http_request(c: connection, method: string, original_URI: string, unescaped_URI: string,
version: string)
{
print fmt("IP:%s WITH PORT NO:%s IS TRYING TO ACCESS HTTP PACKET",c$id$orig_h,c$id$orig_p);
}
```

Snapshot 4.10: File containing HTTP Script.

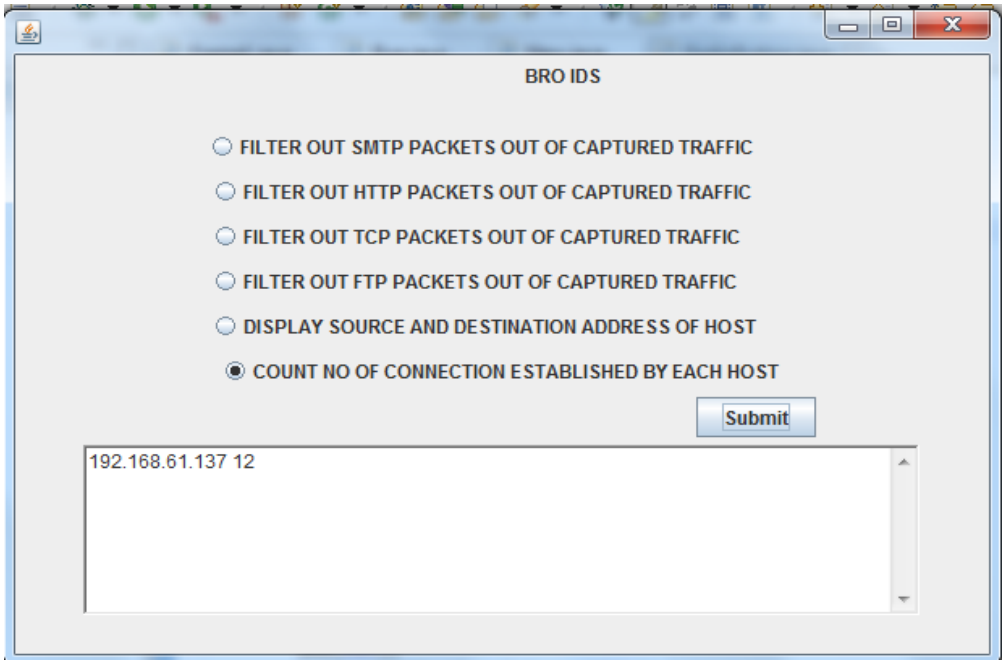
After clicking on desired option, i.e group button 2, filtered traffic containing only HTTP protocol packets are displayed as shown in Snapshot 4.11.



Snapshot 4.11: Run of HTTP Script.

4.7.3 To count the number of connections established by each host.

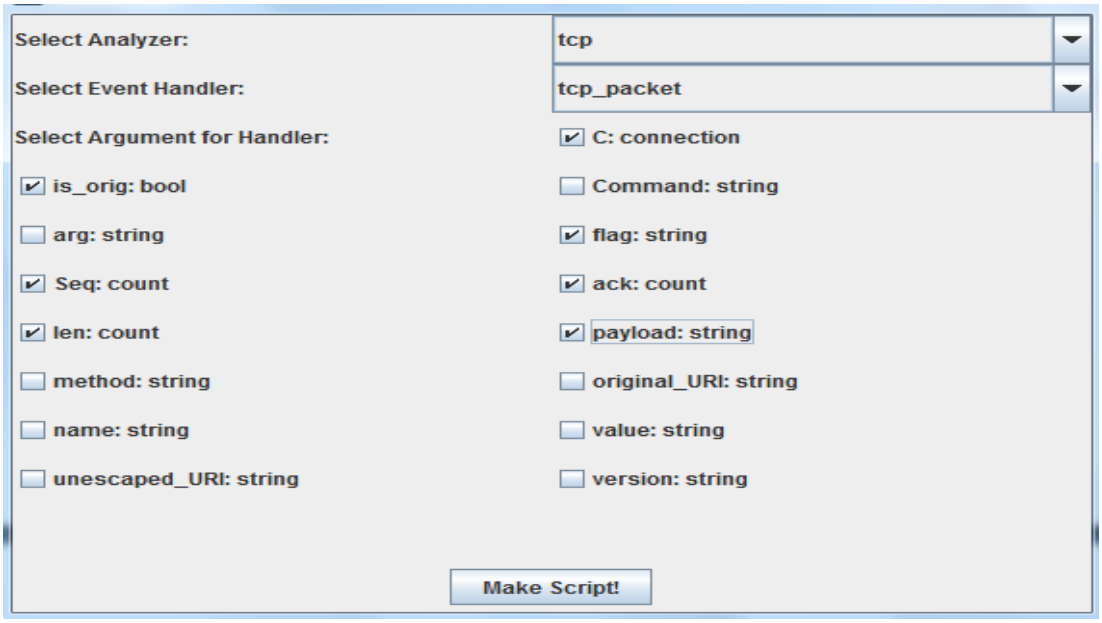
To count the number of connections established by each local host is specified in this script. The script is run after selecting the group button 6, the output is displayed as shown in Snapshot 4.12.



Snapshot 4.12: Run of Count Script.

4.7.4 TCP Script

TCP policy script uses tcp as analyzer, tcp_packet as event handler and c: connection, is_orig: bool, flags: string, seq: count, ack: count, len: count, payload: string as arguments used in handler. Bro instantiates a tcp analyzer to process the traffic associated with the TCP protocol. The making of script is shown in Snapshot 4.13.



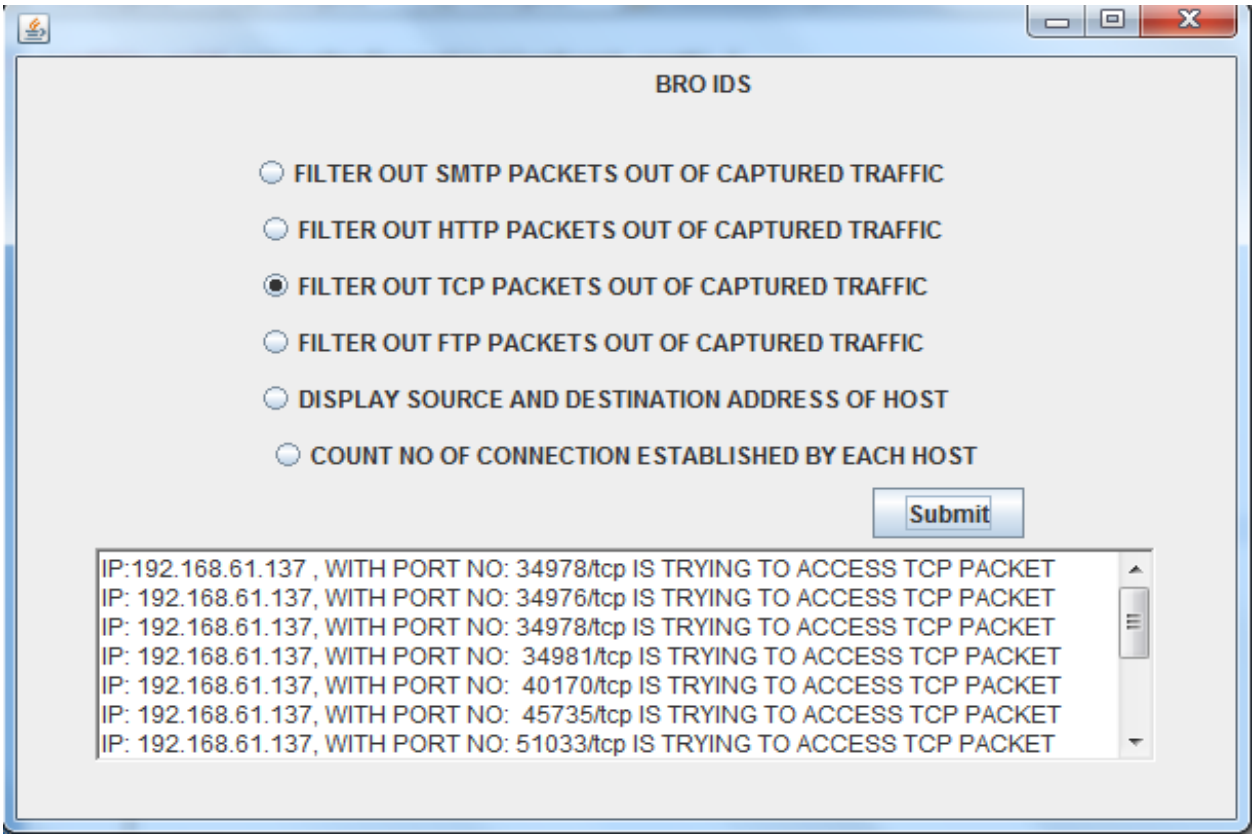
Snapshot 4.13: Make of TCP Script.

On clicking the Make Script! button, a file called out.bro as shown in Snapshot 4.14 is created at the pre- decided path.

```
@load weird
@load alarm
@load tcp
global path: string;
redef ignore_checksums= T;
event tcp_packet(c: connection, is_orig: bool, flags: string, seq: count, ack: count, len: count,
payload: string)
{
print fmt("IP:%s,WITH PORT NO:%s IS TRYING TO ACCESS TCP PACKETS",c$id$orig_h,c$id$orig_p);
}
```

Snapshot 4.14: File containing TCP Script.

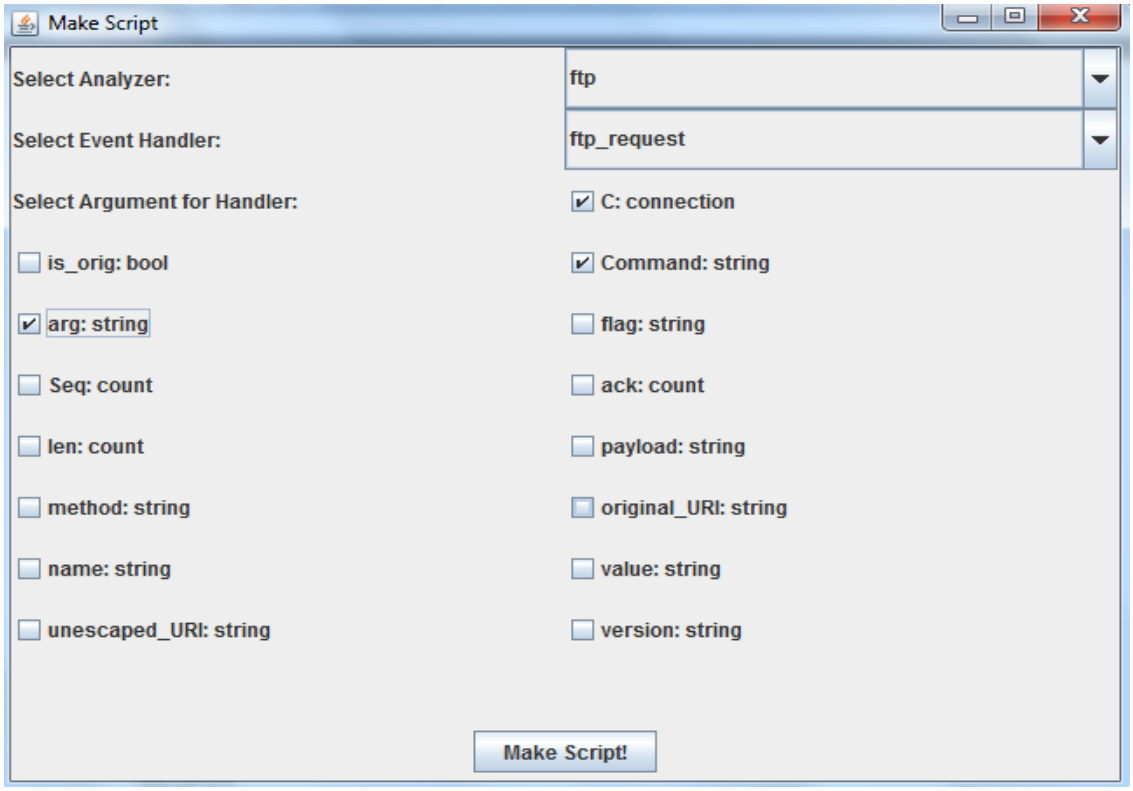
After clicking on desired option, i.e group button 3, filtered traffic containing only TCP packets are displayed as shown in Snapshot 4.15.



Snapshot 4.15: Run of TCP Script.

4.7.5 FTP Script

Bro instantiates a ftp analyzer to processes the traffic associated with the FTP file transfer service. ftp analyzer listen on the port number 21. The ftp policy script uses ftp as analyzer, ftp_request as event handler and c: connection, command: string, arg: string as arguments used in the handler. The making of script is shown in Snapshot 4.16.



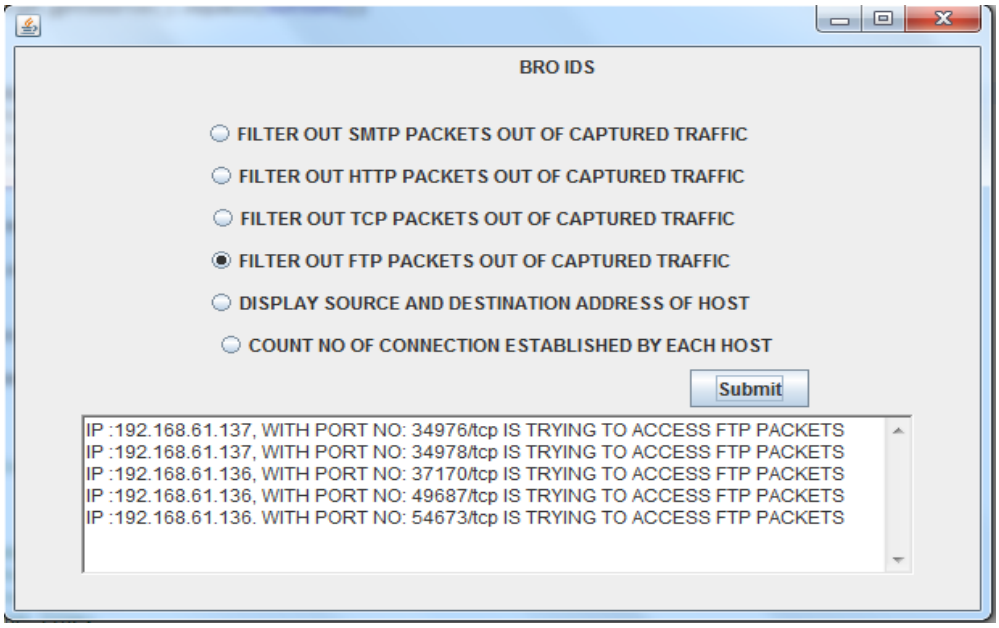
Snapshot 4.16: Make of FTP Script.

On clicking the Make Script! button, a file called out.bro as shown in Snapshot 4.17 is created at the pre- decided path.

```
@load weird
@load alarm
@load ftp
global path: string;
redef ignore_checksums= T;
event ftp_request (c: connection, command: string, arg: string)
{
print fmt("IP:%s WITH PORT NO:%s IS TRYING TO ACCESS FTP PACKET",c$id$orig_h,c$id$orig_p);
}
```

Snapshot 4.17: File containing FTP Script.

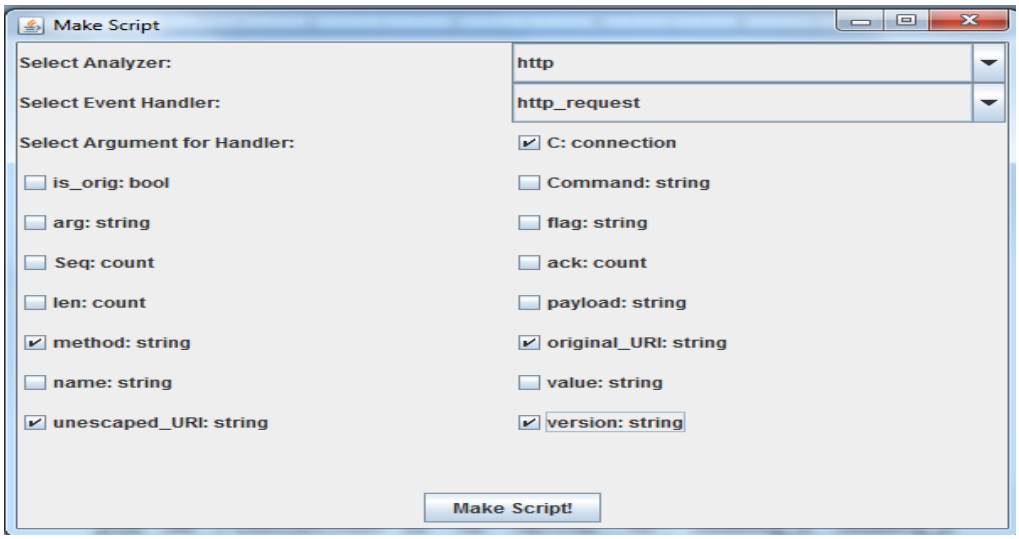
After clicking on desired option, i.e group button 4, filtered traffic containing only FTP protocol packets are displayed as shown in Snapshot 4.18.



Snapshot 4.18: Run of FTP Script

4.7.6 To display source and destination address of the captured traffic

The IP address and port number of the source and destination machines are displayed by running the script created by this Make Form. These addresses help user to analyze and examine the network intrusions. The analyzer used is http and event handler is http_header. The arguments selected are c: connection, is_orig: bool, name: string, value: string as shown in Snapshot 4.19..



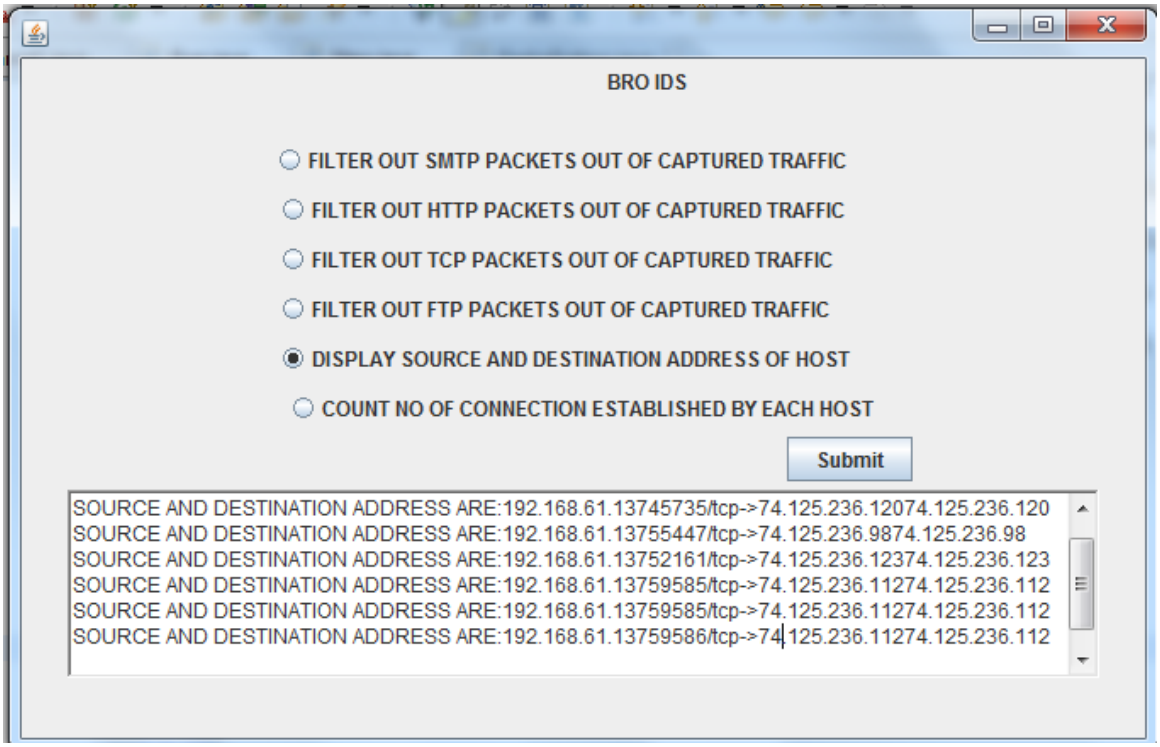
Snapshot 4.19: Make of IP Addresses Script.

On clicking the Make Script! button, a file called out.bro as shown in Snapshot 4.20 is created at the pre- decided path.

```
@load weird
@load alarm
@load http
global path: string;
redef ignore_checksums= T;
event http_header(c: connection, is_orig: bool, name: string, value: string)
{
print fmt("SOURCE AND DESTINATION ADDRESS ARE: %s:%s->%s:%s",c$id$orig_h,c$id$orig_p,c$id$resp_h,c
$id$resp_p);
}
```

Snapshot 4.20: File containing IP Addresses Script.

After clicking on desired option, i.e group button 5, filtered traffic containing only source and destination address are displayed as shown in Snapshot 4.21.



Snapshot 4.21: Run of IP Addresses Script.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

Today network is very complex and whole world is focusing on ease of use and functionality. Unfortunately security policies and rules needed to govern these networks have not progressed as rapidly. So there is huge need of detecting the threats and intrusions as rapidly as possible. An intrusion detection system is used to monitor network traffic, check for suspicious activities and notifies the system or network administrator. Bro is one of the most effective IDS which can be used to detect these threats.

In this dissertation, new policy scripts are designed to filter out the needed packets from traced traffic and generate alert on desired incoming network packets. Policy scripts are written to filter E-mail and File Transfer traffic going via SMTP and FTP protocol respectively. These policy scripts display web traffic sent on HTTP and TCP protocol after analyzing them. These scripts help to know the source and destination address of the captured traffic. Also a policy script is built to count the number of established connections by each local host.

A GUI framework is integrated in Bro that analyzes and filters the traced network traffic. It eliminates the need of writing the commands at terminal and makes it easy for users to create the scripts and run them on captured traffic.

5.2 Future Work

1. Presently the traffic of SMTP, TCP, HTTP and FTP is examined and filtered. The work can be enhanced to trace the traffic of SSL, bit torrent and P2P.
2. In this thesis, scripts are created that generate alerts on desired incoming network packets. In future, scripts can be created that will used to block the desired network traffic.
3. The proposed GUI framework can be further extended to report events to database instead of log files so that data can be stored safely for future reference.

References

[1] Allman E., Shapiro G. N. and Assmann C., "Sendmail Installation and Operation guide", US Patent 6865671, 6986037, October 2001.

[2] Anderson D., Lunt T., Javitz H., Tamaru A. and Valdes A., "Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES)", *SRI International Computer Science Laboratory Technical Report SRI-CSL-95-06*, May 1995.

[3] AXENT Technologies Intruder Alert, Available at: http://www.axent.com/product/smsbo/ITA/, 1999.

[4] Bai Y. and Kobayashi H., "Intrusion Detection System: Technology and Development", in *Proceedings of the 17th International Conference on Advanced Information Networking and Applications*, pp. 710-715, 2003.

[5] Compton C. and Tennenhouse D., "Collaborative load shedding for media based applications", in *Proceedings of IEEE International conference on Multimedia Computing and Systems*, pp. 496-501, May 1994.

[6] Fink G. A., Chappell B. L., Turner T. G. and O'Donoghue K. F, "A Metrics-Based Approach to Intrusion Detection System Evaluation for Distributed Real-Time Systems", in *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, pp. 93-100, 2002.

[7] Forrest S., Homeyr S. and Sommayaji A., "Computer Immunology", *Communications of the ACM*, vol. 40, no. 10, pp. 88-96, October 1997.

[8] Garvey T. D. and Lunt T. F., "Model based Intrusion Detection", in *Proceedings of the 14th National Computer Security Conference*, pp. 372-385, October 1991.

[9] Gong F., "Deciphering Detection Techniques: Part II Anomaly-Based Intrusion Detection", White Paper, March 2003.

[10] Hypertext Transfer Protocol, Available at http://en.wikipedia.org/wiki/ Hypertext_Transfer_Protocol.

[11] Ilgun K., Kemmerer R. A. and Porras P. A., "State transition analysis: A rule-based intrusion detection approach", *IEEE Transactions on Software Engineering*, vol. 21, no. 3, pp. 181-199, March 1995.

[12] Jalali C., Jagannathan, Denning P.G. and Dorothy E., "An Intrusion-Detection Model", *IEEE Transactions on Software Engineering-Special Issue on Computer Security and Privacy*, vol. 13, no. 2, pp. 222-232, February 1987.

[13] Jensen K., "Coloured Petri Nets. In: W. Brauer, W. Reisig, G. Rozenberg (eds.): Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986 Part I", *Lecture Notes in Computer Science*, vol. 254, pp. 248-299, 1987.

[14] Jones A.K. and Sielken R.S., "Computer System Intrusion Detection: A survey", Department of Computer Science, University of Virginia, United States, 1999.

[15] Kumar S. and Spafford E.H., "A Pattern Matching Model for Misuse Intrusion Detection", COAST Project, Department of Computer Science, Purdue University, Indiana, US, 1994.

[16] Lunt T. F., Tamaru A., Gilham F. and Neumann R., "A Real-Time Intrusion Detection Expert System (IDES)", Computer Science Laboratory, Final Technical Report, Fourth Ed., SRI International, Menlo Park, CA, February 1992.

[17] Myers S., Musacchio J., Bao N., "Intrusion Detection Systems: A Feature and Capability Analysis", *Technical Report*, May 2010.

[18] Need of Network Security, Available at: http://www.indiastudychannel.com/resources/105777Network-Security-Attackers Hackers.aspx.

[19] Ning P. and Jajodia S., "Intrusion Detection Techniques", in the Internet Encyclopedia, H. Bidgoli, Ed., New York: John Wiley & Sons, 2003.

[20] Oh S. H. and Lee W.S., “An Anomaly Intrusion Detection Method by Clustering Normal User Behavior”, *Computer and Security*, vol. 22, no.7, pp. 596-612, October 2003.

[21] Paxson V., “Bro: A System for Detecting Network Intruders in Real-Time”, in *Proceedings of 7th USENIX Security Symposium*, pp. 2435-2463, December 1999.

[22] Porras P. A. and Kemmerer R. A., “Penetration State Transition Analysis: A Rule Based Intrusion Detection Approach”, in Proceedings of *8th Annual Conference on Computer Security Application*, pp. 220-229, December 1992.

[23] Ptacek T. and Newsham T., “Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection”, Secure Networks, Inc. Available at http://www.aciri.org/vern/Ptacek-Newsham- Evasion-98.ps, January 1998.

[24] Rehman R.U., “Intrusion Detection Systems with Snort”, Prentice Hall, New Jersey, 2003.

[25] Scarfone K., and Mell P., “Guide to Intrusion Detection and Prevention Systems”, *NIST Special Publication 800-94*, February 2007.

[26] Sekar R., Bendre M., Dhurjati D. and Bollineni P., “A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors", in *Proceedings of 2001 IEEE Symposium on Security and Privacy*, pp. 144-155, 2001.

[27] Simple Mail Transfer Protocol, Available at http://en.wikipedia.org/wiki/ Simple_Mail_Transfer_Protocol.

[28] Sobh T.S., “Wired and Wireless Intrusion Detection System: Classifications, good Characteristic and State-Of-The-Art”, *Computers and Standards and Interfaces in Computing*, vol. 28, no. 6, pp. 670-694, 2005.

[29] Sommer R., “BRO: An Open Source Network Intrusion Detection System”, in Security, E-Learning, E-Services, 17 DFN-Arbeitstagung uber Kommunikationsnetze, vol. 44, Dusseldorf, Germany: Gesellschaft fur Informatik (GI), 2004, pp. 273–288

[30] Sommer R., Slides on the Bro Network Intrusion Detection System, Lawrence Berkeley National Laboratory, Berkeley, CA, 2009.

[31] Stallings W., "Cryptography and Network Security: Security Services", 4th Ed., Delhi: Prentice Hall, 2007.

[32] The Honeynets Project Know Your Enemy. Available: http://project.honeynet.org/papers/honeynet.

[33] Wagner D. and Dean D., "Intrusion Detection via Static Analysis", in *Proceedings of 2001 IEEE Symposium on Security and Privacy*, pp. 156-168, 2001.

[34] Wireshark, Man Pages. Available at: http://www.wireshark.org/docs/man-pages/wireshak.html.

[35] Yue X., Chen W. and Wang Y., "The Research of Firewall Technology in Computer Network Security," in *Proceedings of Conference on Computational Intelligence and Industrial Applications*, pp. 421-424, 2009.

List of Publications

1. Shaffali Gupta, Sanmeet Kaur, "*Comparative Analysis of Various Misuse Detection Approaches: A Survey",* International Conference on Advanced Computing Technologies, June 2012 at Gurukul Vidyapeeth, Chandigarh.
2. Shaffali Gupta, Sanmeet Kaur, *Design and Implementation of Policy Scripts to Detect Network Intrusions using Bro IDS,* International Conference On "Advanced Computing Technologies", June 2012 at Gurukul Vidyapeeth, Chandigarh.