Design and Development of an Efficient Alert Summarization Technique for Cloud Environment to Detect Intrusions

THESIS

Submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

In

Information Security

Submitted By:

Ankit Singh

(Roll No. 801233001)

Under the Supervision of:

Dr. Neeraj Kumar

Associate Professor (CSED)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY, PATIALA – 147004

June 2014

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*Design and Development of an Efficient Alert Summarization Technique for Cloud Environment*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Information Security* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Neeraj Kumar.*

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

Signature:

(Ankit Singh)

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

(Dr. Neeraj Kumar)
Associate Professor
Computer Science and Engineering Department

Countersigned by:

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

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

Acknowledgments

The written word acknowledgement has an unfortunate tendency to degenerate a feeling of genuine gratitude into a formality. However, I have no other means of recording my feelings permanently. First, and foremost I would like to thank my supervisor *Dr. Neeraj Kumar* for his thorough guidance, support, encouragement, and counsel throughout the research and painstakingly reading my reports. Without his invaluable advice and assistance it would not have been possible for me to complete this thesis.

I would also like to give my most sincere thanks to *Dr. Deepak Garg*, Head, CSED for providing me the opportunity to conduct the research work presented in this thesis; I would also like to thank *Dr. Maninder Singh*, Associate Professor for his invaluable guidance and advice on the topic. I am also grateful to *Mr. V.P. Singh*, System Analyst for helping me in removing errors during my implementation work. I also thanks to *Mr. Ajitpal Singh*-Ph.D. Scholar, *Mr. Vikram Ojha*- M.Tech Scholar and *Mr. Divesh Jain*-MCA Student for their support throughout my research work.

I am also indebted to all my family members, my friends and my well wishers for their inspirational and constant encouragement.

Ankit Singh.

Ankit Singh

ME (Information Security)

801233001

Abstract

The recent development of cloud computing has drastically modified everyone's perception of software delivery, infrastructure architectures and development models. Following the transition from mainframe machines to client/server deployment models, cloud computing incorporate elements from utility computing, grid computing and autonomic computing, into revolutionary deployment architecture. Cloud computing has emerged as a new computing paradigm in which users can access various resources from remote sites using 'pay-per-service'. This brisk transition regarding the clouds has fuelled concerns on a censorious issue regarding the success of information security, communication and information systems.

The open and distributed structure of cloud computing and services has become an appealing target for potential cyber-attacks by intruders. The conventional Intrusion Detection Systems (IDS) are inefficient to be deployed on cloud computing environments because of their openness and specific essence. Traditional IDSs are known for producing large volumes of alerts regardless of all the progress made over the last few years. The dissection of a large number of raw alerts from giant networks is usually labour intensive and time consuming because the relevant alerts are usually buried under the heaps of irrelevant alerts.

The work presented in the thesis showcases the development of an efficient alert summarization technique that is embedded in IDS implemented on cloud environment, which filters out the irrelevant alerts depending on various trust factors thus improving the quality of relevant alerts, hence enabling the analyst to focus on important alerts. The proposed prototype has been implemented in real environment and different types of vulnerabilities were examined using the proposed system. Also various parameters for intrusion detection have been observed in real cloud environment.

Keywords: Cloud Security, Intrusion Detection System (IDS), Alarm Management.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Cloud Computing (CC) can be visualized as the change into reality of a long held dream called 'Computing as a Utility'; CC assures on-demand services for a customer's infrastructure, software and platform needs. In its bunch, companies do not even require to plan or create blueprint for their IT growth in advance with this advanced 'pay as you go' system. So far, many upbeat assessment has been done about its great potential for scalability, utility and instant access -features; but on the another side many researchers are worried because of the security gaps, involving for instance, risk, trust and threats [1]. CC can be visualized as a new architecture of Information System (IS), which can be viewed as a destiny of computing, an energetic force nagging from its audience to rethink their understanding of client-server architecture, operating systems and browsers. CC has anchorage users from the view point of requirements of hardware, while minimizing overall client side complexity and requirements [2]. As the popularity of CC is increasing, many concerns are being voiced related to the security issues introduced through the acquisition of this novel model. The efficiency and effectiveness of traditional protection methods are being reconsidered, as the characteristics of this unprecedented deployment model differ very much from that of traditional architecture. Before we investigate the privacy and security issues in CC, it is beneficial to revisit the definition of CC. Various books and articles have been scrutinized and a new definition that is easy to understand and yet vast in its scope, which can be envisioned in diagrammatical form has been described in Figure.1. In words form the definition can be stated as: CC is a kind of a system/architecture, where the resources and information of a data centre are shared using virtualization technology, which also provides on demand, elastic and instant services to its authorized users and thus charges customer usage as utility bill.

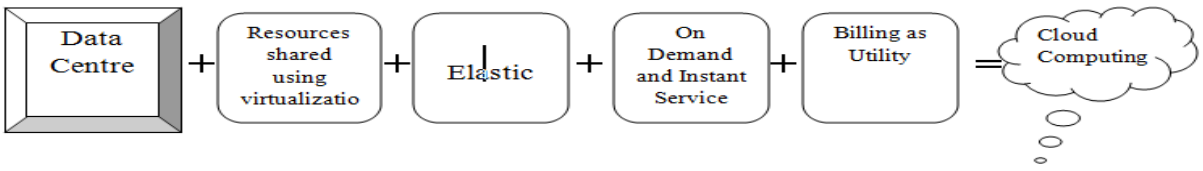


Fig.1: Schematic definition of cloud computing [1].

Elasticity, instant service, virtualization, on-demand and pay as you go are the main attributes that transform data centre into CC. With the change in the IT infrastructure where there are escorting new risks, CC is no exception. On-demand and shared nature of CC makes it vulnerable to some distinct/unique risks/threats that have not been experienced before.

Thus the area of information security plays a very crucial role in the safety and economic well being of a society as a whole. The speedy growth and extensive use of electronic data processing and electronic business conducted through the use of wireless and wired networks, web application, Internet, raises the need for implementing safe and secure information security system through the use of Intrusion Detection System (IDS), firewall, encryption and other software and hardware solutions. In the race of securing the systems and stored data, IDS can prove to be an indispensable tool, where its motive is to perform the detection of malicious activity thus preventing further damage to the systems. By implementing IDS in an organization one can easily detect any anomalous activity or an attack and thus can notify to the security administrator, which in turn can take the appropriate action to restrict attack [3]. Forensic evidences can be recorded with the help of IDS that can be used in legal proceedings if the performer of attack is prosecuted. However the performance of IDS is slowed down by high false alarm rate or by the occurrences of the repeated alarm that occur from the same source, thus the thesis focuses on summarizing and parsing the alarms of same origin/pattern (depending upon various attributes like source IP, signature_id, timestamp and many more), so if the mirrored attacks occurs having the same timestamp than instead of generating the same number of alarm as that of the number of mirrored attack, the IDS will only generate the single relevant alarm, which allows security administrator to focus on critical issues and also provide him more time to take necessary actions.

If we take a broader view, the thesis work is divided into 2 sections, the first one being the literature survey and the second one is the implementation and results. The literature survey is divided into 3 parts, first part gives the comprehensive description of the layered taxonomy of IDS, the second part explains the CC along with the security issue that are related to CC and the last part deals with the features, characteristics, challenges and requirements of implementing IDS in cloud environment. Second section (implementation) of the thesis is further divided into two parts, first part showcases the complex configuration and installation of the IDS along with its supportive software on a cloud node, while the second part deals with attaching the summarizer and parser algorithm to that of MySQL ‘Snort’ database that will refine the logs and thus generates the relevant alerts.

Chapter 2: Literature Review

2.1 Intrusion Detection System (IDS)

Outsider attacks are those that have their origin outside the perimeter of the organizations while insider attacks involves unauthorized internal employees trying to misuse and gain non-authorized access privileges. Intrusion detection is a technique of monitoring networks and computers for unauthorized activity, entry or file alteration [4]. Most of the time attacks occur in distinctive groups called incidents. However many incidents are malicious in nature, while some are not; for example a person might mistype a wrong address of a machine and accidently make an effort to connect to a different machine without authorization. IDS is a software that automates the process of intrusion detection process and thus unearth possible intrusions. The thesis works showcases the use of Snort, which is an open source auditing tool, and has received great acceptance in the market of IDS. Snort performs packet logging and real time packet analysis on the network, it analyses protocol and detect different network threats with the help of various signature matching algorithms.

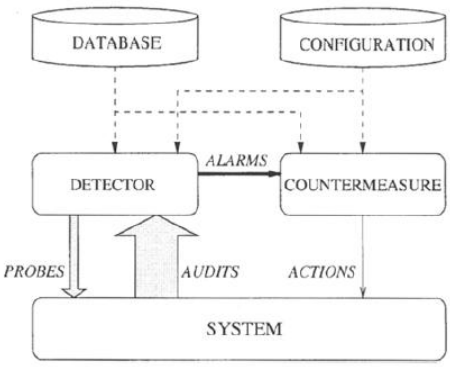


Fig.2: Scenario of simple IDS [5].

The thesis work concentrates on the Snort capability as a NIDS. As a NIDS, Snort is composed of packet sniffer, detection engine, pre-processor, and output plugins as shown in Figure 3.

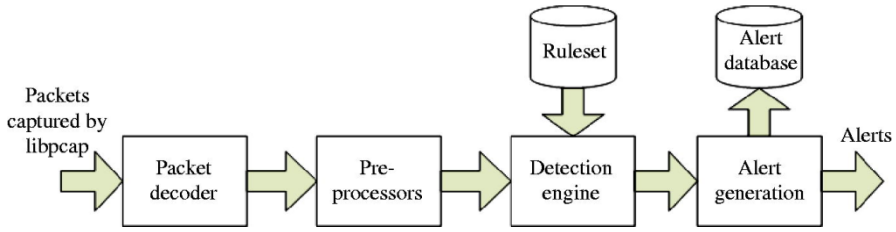


Fig.3: Components of Snort [7].

In a packet sniffer mode, it captures the network traffic and transfers it to pre-processor. Snort relies on packet capturing libraries such as libpcap and winpcap. The libraries function as a packet capturing interfaces to the corresponding operating system. Raw packets are the input to the preprocessor, the preprocessor than check those raw packets against certain type of behaviour. The packet is forwarded to the detection engine if a particular behaviour is discovered otherwise it is discarded. User can configure the application according to their need; this is possible because of the availability of different plug-ins. It also excludes the unnecessary processing of all incoming packets which may produce load on the machine. Detection engine is the most important component of a NIDS. It fetches data from the pre-processors and equates it against certain bunches of rules. The rules are the set of condition that trigger an alert on discovering certain strings of characters in a file, spotting known pattern of attacks etc. Snort permits the users to write their own customized rules on foreseeable threats/ network characteristics. To add up, authorized users have approval to download Source fire VRT rules present on Snort website [6]. If the data present in a packet matches with the rule in the detection engine than an alert is generated and sent to a log file.

2.1.1 A layered taxonomy of IDS [3]

2.1.1.1 Functional Layer: As shown in figure 4, IDSes provides four crucial security functions i.e. they monitor detect and analyze after that they respond to unauthorized activity as portrayed in the functional layer. Intrusions are detected by IDS after analyzing the collected data.

2.1.1.1.1 Monitoring Layer:

1. The observed environment can be host-based, network-based, or application-based:

a. Network-based IDS: observes network traffic for specific network devices or segment and examines the application and network protocol activity to identify disreputable activity.

b. Host-based IDS: monitors the state of the computer machine and identify which process accesses what resources.

c. Application-based IDS: focuses on events that take place in some particular application by way of scrutinizing the application log files or evaluating their performances.

2. Time of detects:

a. Real time: in case of real-time detection, attacks can be recognized or identified while the network or system can instantly flag any deviation and provide proper prevention.

b. Non-real time: processes audit data with delay.

3. Data Collection: From several distinct sources and location, audit data can be gathered in a distributed fashion or they can be gathered using a centralized approach from one particular source.

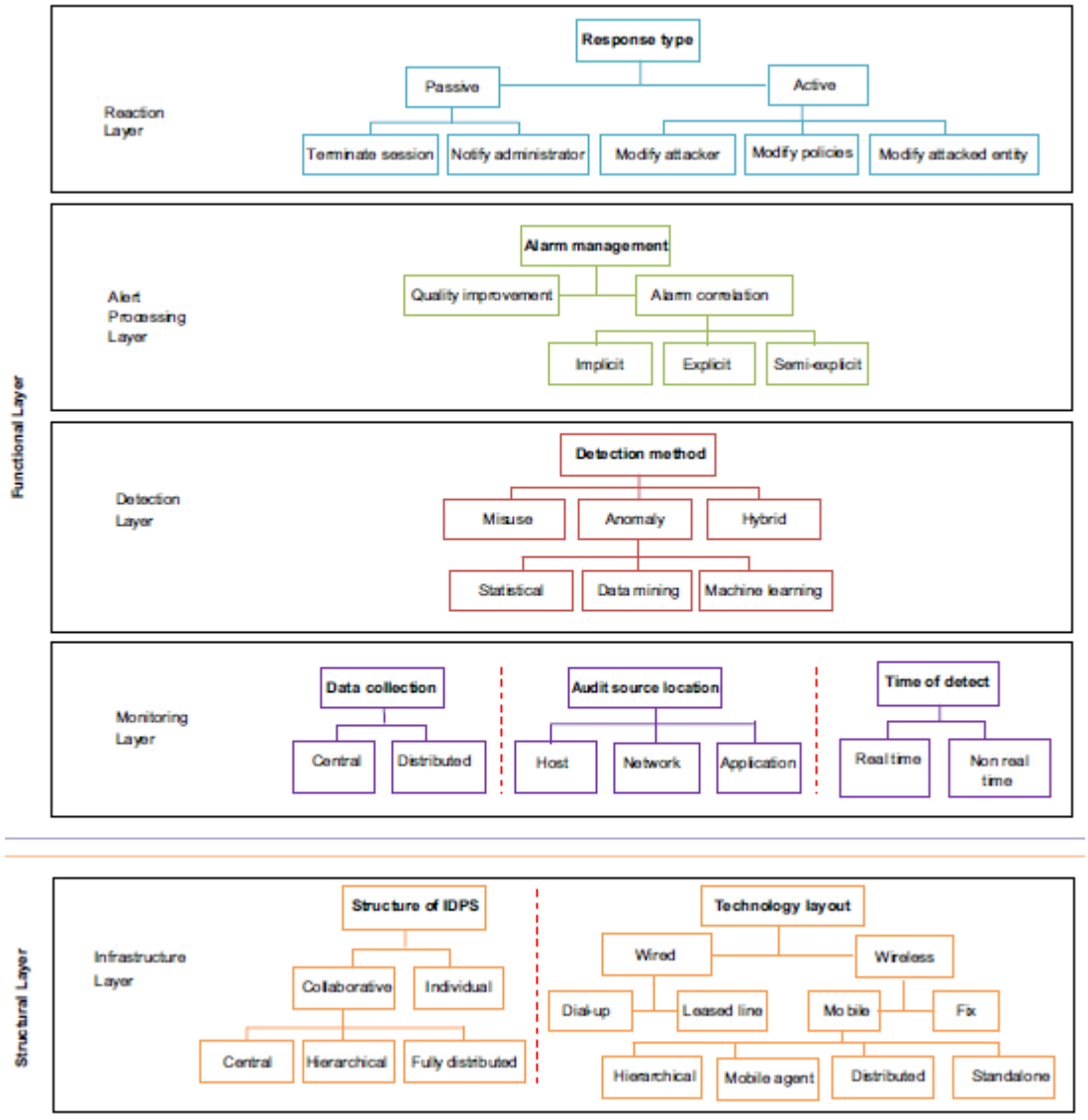


Fig.4: A layered taxonomy of IDS [3].

2.1.1.1.2 Detection Layer: The identified techniques of detection are categorized into three classes of anomaly, misuse and hybrid model amalgamating the first 2 classes.

1. Misuse Detection: this method utilizes specifically investigated patterns of illegitimate behaviour, called signatures, to forecast and identify subsequent similar attempts.

2. Anomaly Detection: invented to expose abnormal patterns of behaviour. IDS sets a baseline of standard usage patterns, and whatever drifts from this get identified as possible intrusions [8]. There are numerous categories of anomaly detection advocated, but the three mainly used are as follows [9]:

a. Statistical: in this method the system monitors the CPU usage or the no. of TCP connections made in terms of statistical distribution and generates profiles which symbolize their behaviours. Thus the system generates two types of profile: first one is generated during the training phase while the other is the latest profile during the detection phase. An anomaly can be identified if any difference is observed among the two profiles.

b. Machine Learning Based: this method has the potential of learning and improving its capabilities over time. It tends to concentrate on constructing a system which can improve its functioning in a loop cycle and can modify its implementation strategy according to feedback data.

c. Data mining based: the technique assist in improving the process of detection of intrusion by unfolding patterns, anomalies, association, changes and important events. Clustering and outline detection, classification and association rule discovery are the techniques of data mining used in IDS.

3. Hybrid: the approach combines the above two detection methodologies (misuse and anomaly) to enhance the capabilities of IDS. The main motive behind this was anomaly detects unknown attacks while the misuse detects known once.

2.1.1.1.3 Alarm Management: Alarm management can be classified into following two methods:

1. Alarm quality improvement: this approach uses information related to vulnerability reports or alert context to improve the quality of alert. There is one more approach in which vulnerability approach is matched with correlated alerts [10].

2. Alarm Correlation: in this approach low level alerts are used to reconstruct the high level incidents. There are three ways to perform alert correlation:

a. Implicit: data-mining technique is used to examine, collect and group large alert datasets. The method is appropriate for the analysis of gigantic number of alerts but it fails to boost the semantics of the alerts.

b. Explicit: this approach relies on language granting security experts to state temporal and logical constraint among alert patterns to recognize complex attack scenarios.

c. Semi-explicit: this approach is an expansion of the explicit approach which links post conditions and pre conditions, reflected by first order formula, with isolates attacks and actions. Thus, it presupposes that complicated intrusions scenarios involves attacks whose prerequisite correlate to the outcome of some earlier ones.

With the increase in the number of anomaly detection application, a fresh trend in under research, which concentrates on how to manage and handle alarms [9]. Table 1 showcases the most recent research striving to deal with the problems related to alarm management.

Reference	Year	Method	Performance	Technique category	IDPS technique	Management model
Tjhai et al. (2010)	2010	Two-stage classification system using Self-Organizing Map (SOM) neural networks and k-means algorithm	More than 50% reduction in false Positive rate (FPR)	Similarity between alert attributes and filtering algorithm	Hybrid	Alarm correlation
Mansour et al. (2010)	2010	Data mining technique based on a Growing Hierarchical Self-Organized Map (GHSOM)	Reduces FPR from 15% to 4.7% and false negatives from 16% to 4% for the real-world data	Filtering algorithms and similarity between alert attributes	Anomaly	Alarm correlation
Spathoulas and Katsikas (2010)	2010	Post-processing filter based upon statistical properties of the input alert set	Up to 75% reduction in FPR	Filtering algorithms	Misuse	Alarm quality
Al-Mamory and Zhang (2010)	2010	Filtering using clustering algorithm	Average 82% reduction of FPR	Filtering algorithms	Hybrid	Alarm quality
Li and Tian (2010)	2010	XSWRL ontology technique	No test	Pre-conditions and post conditions of attacks and predefined attack scenarios	Misuse	Alarm correlation
Maggi et al. (2009)	2009	Fuzzy measures and fuzzy sets	Decrease the FPR, but with a small reduction of the detection rate	Similarity between alert attributes	Anomaly	Alarm correlation
Zeng et al. (2009)	2009	Antibody Concentration (NIDMBAC)	FPR was reduced by 8.66%, 4.93% and 6.36% without affecting detection rate	Multiple information sources	Anomaly	Alarm correlation
Vincent Zhou et al. (2009)	2009	Multi- dimensional alert clustering algorithm	Significant reduction in number of alert messages generated	Similarity between alert attributes and filtering algorithm	Anomaly	Alarm correlation
Hoang et al. (2009)	2009	Hybrid fuzzy-based anomaly IDS utilizing hidden Markov model (HMM) detection engine and a normal database detection engine	Reduced FPR by 48%	Similarity between alert attributes	Anomaly	Alarm correlation
Anuar et al. (2008)	2008	Statistical analysis of both attack and normal traffics based on hybrid statistical approach using Data Mining and Decision Tree Classification	Different accuracy results for two models of decision tree and rule-based data mining	Pre-conditions and post conditions of attacks	Unspecified	Alarm correlation
Pietraszek and Tanner (2005)	2005	Data mining (clustering), machine learning (feedback)	More than 50% reduction in FPR	Filtering algorithm	Misuse	Alarm quality

Table 1: Classification of alarm management techniques [3].

2.1.1.2 Structural layer

The design/framework of IDS can be either collaborative or individual. An individual arrangement of IDS can be attained by physically integrating it with a firewall. A collaborative IDS involves multiple IDS over a huge network where each and every IDS communicates with each other. There are two main functional components of IDS: correlation handler and detection element. Detection element is composed of various detection components which observe their own host or sub-network individually and trigger low level alerts. Then the correlation handler renovates the low level alerts into a high level. Furthermore as shown in figure 5, collaborative IDS can be categorized into three main categories [9]:

2.1.1.2.1 Central: each IDS behave as a detection element where it generates alerts locally. The alerts generated are forwarded to a central server that plays the character of a correlation handler to examine them. Detection decision can be devised through a centralized management control, based on all the available data. The major limitation of this approach is that any failure to the central server will result in deactivating the entire process of correlation.

2.1.1.2.2 Hierarchical: the entire system is separated into several small groups based on their features such as: administrative control, geography and similar software platforms. The IDS at the lower level are employed as detection elements, while the IDS at the higher level are employed with both the correlation handler and the detection element and correlate alert from the lower level as well as from their own level. The correlated alerts are then promoted to a higher level for further analysis.

2.1.1.2.3 Fully distributed: the information is not processed by a centralized coordinator, it comprises of a fully independent system with a distributed management control. Each participating IDS has two main functional components that communicate with each other. Some of the advantages of a fully distributed system includes: there in no compulsion for network entities to have entire information of the network topology, since there is no central entity that is liable for doing all the correlation work, it is feasible to have more scalable design.

2.1.2 Some of the desired characteristics that are being identified for ideal IDS to have maximum protection, optimized performance and minimum error [11]:

DC1: Run continuously with least or no human supervision.

DC2: Must be able to tolerate faults and should be able to recover when system crashes.

DC3: Be simply attached to a specific network.

DC4: With time, it should adapt to changes depending upon user and system behaviour.

DC5: Work in real time.

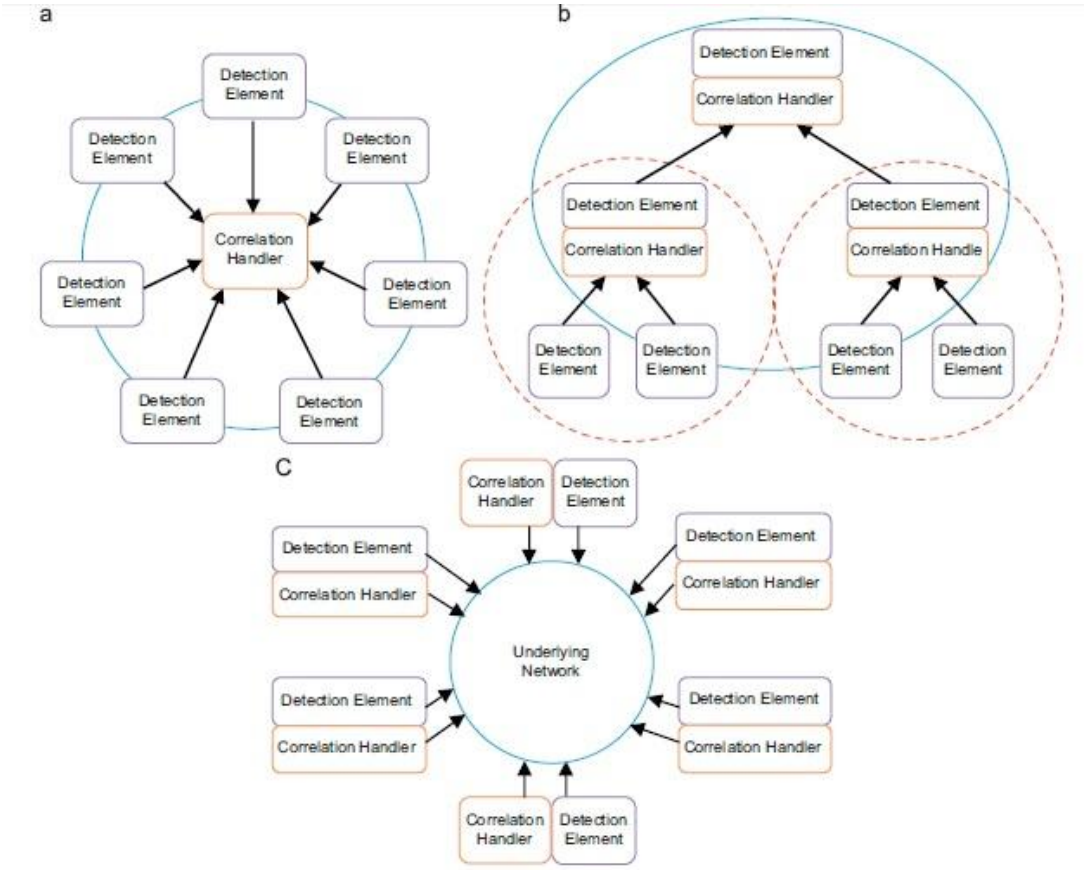


Fig.5: Different management structure of collaborative IDS [3]. (a)Central (b) Hierarchical (c) Fully Distributed.

DC6: Should be able to recognize maximum number of intrusions with few false-positive alarms.

DC7: Be self-protected and self-monitored, in case it is tempered by an attacker.

DC8: It should be able to configure itself in accordance with the changing security policies of the machine under observation.

DC9: Must operate with minimal overhead.

To the best of our knowledge, most of the developed IDS fulfil only 3 characteristics that is DC2, DC5, DC9. DC1 which is the first characteristic totally depends upon the human supervision. Traditional IDS those are present today are not being able to satisfy DC3 and DC4 as they cannot be fitted into a new network scenario and are not compatible new network environment. IDS that are running in most of the organizations are able to recognize mostly all the intrusion (DC6), on the cost of high false alarm rate. Since the current developed IDS are not self-managed system thus they are far from achieving DC7 and DC8.

2.1.3 Evolving challenges that limit IDS development:

a. Traditional IDSes are not compatible with the new networking environment which includes wireless and mobile network technology. They also failed to satisfy the high speed networking requirements [12] due to their scalability limitation.
b. Due to some counter positive factors like noise in the audit data, the traffic profile constantly changes which creates difficulty in building a normal traffic profile in an enormous amount of network traffic.
c. High false alarm rate is one of the most serious limiting factors blocking the widespread usage of IDS.
d. There is no globally accepted metric or standard to judge the efficiency and effectiveness of IDS.
e. Increasing number of 'insider' threats, which are very much difficult to detect is also a limiting factor for an IDS.
f. Provision of appropriate policies and rule sets for internal intruders and the proper configuration of the system is also a very challenging task.

2.2 Grid and cloud computing

Grid computing came to existence in the early 1990s, where high performance machines were inter-connected via speedy data communication link, with the goal of processing data-intensive scientific applications and complex calculation. Grid computing can be explained as '' a software and hardware infrastructure that deliver reliable steady, prevalent, and affordable access to high-end computational capabilities". Cloud computing has emerged from the convergence of utility computing, grid computing, and SaaS, and basically represents the increasing drift towards the outer deployment of IT resources, such as business or storage application or computational power and obtaining them as a service [13]. Cloud computing is a framework for enabling suitable on-demand network access, to a shared pool of configurable computing resources such as servers, network, applications, storage and services, that can be quickly supplied and released with least management effort [14].

The key word 'cloud computing' was inspired by the symbol of the cloud that is mostly used to represent the flow of internet in charts and diagram. Over recent years, a distinct relocation to the cloud has been taking place 'bit by bit' maintaining a increasing number of personal data, including photographs, bookmarks, music files and much more, on distant server reachable/attainable via network. Virtualization is the key technology that empowers the cloud computing. In layman term, a host machine runs a process known as

hypervisor; hypervisor can be used to generate one or more virtual machines, which feign physical computer so obediently, that the simulation can execute any software [15]. If we talk about hardware level, a number of physical devices are located in data centres that include hard drives, processors, and network devices which are independent from geographical position and are responsible for processing and storage needs. Above this, the amalgamation of the virtualization layer, software layer and the management layer, authorize for all effectual management of servers. The most crucial element of cloud implementation is the virtualization which provides the essential cloud attributes/features of resource pooling, location independence and rapid elasticity. In contrast with the traditional network topologies, cloud computing is capable of reducing the network traffic overcrowding issues and it also offer robustness. The management layer is capable of observing the traffic and respond to high and low with the devastation of the non necessary servers or the formation of the new ones. The management layer has the supplementary capability of being able to execute security monitoring and rules all over the cloud environment. According to Merrill Lynch, virtualization is the only key feature that differentiates cloud computing from the grid computing. In grid computing, high utilization is achieved by allocating a multiple servers to a unique task while the virtualization of servers in cloud attains high utilization by permitting one server to process several task simultaneously [16]. Many authors states and admit that cloud computing and grid computing have many similarities; along with the statement that states cloud computing has developed from the grid computing and grid computing is the base for cloud computing.

2.2.1 In cloud computing, the available service models are:

2.2.1.1 Infrastructure as a Service (IaaS): provides end users with the ability to provision storage, processing, networks and other essential computing resources and permits users to install and run different software's that include operating system and software's related to different application. The customer has a proper control over deployed applications, storage, and operating system and possibly less command over the few networking components.

2.2.1.2 Platform as a Service (PaaS): End user has the power to implement onto the cloud infrastructure; consumer acquired or created applications, constructed using different programming languages and tools that are compatible with the cloud infrastructure. The end user does not supervise or manage the fundamental cloud

infrastructure including operating system, servers, network or storage, but has proper control over the deployed application.

2.2.1.3 Software as a Service (SaaS): Lets the end user with the ability to utilize the provider's application rolling on the cloud infrastructure. The distinct application can be accessed from various client devices, by means of thin client interface, like web browser. The end user does not supervise or manage the fundamental cloud infrastructure including operating system, servers, network or storage, with the limited exception of user particular application configuration settings.

2.2.2 Cloud architecture has four deployment models:

2.2.2.1 Private cloud: The infrastructure of the cloud is run and managed for a private organization. It can be supervised by the concerning organization or by a third party, and may exist off premise or on premise.

2.2.2.2 Community cloud: The infrastructure of the cloud is shared by several organizations and is supported by a specific community that has shared concerns like mission, policy, requirements, security and compliance consideration. It can be supervised by the concerning organization or by a third party, and may exist off premise or on premise.

2.2.2.3 Public cloud: The infrastructure of the cloud is made available to the general public and industry group and is managed and owned by an organization vending cloud services**.**

2.2.2.4 Hybrid Cloud: The infrastructure of the cloud is composed of two or more cloud deployment models (community, private or public) that remains sole entities, but are tied together by proprietary and standardized technology that permits application and data portability [14].

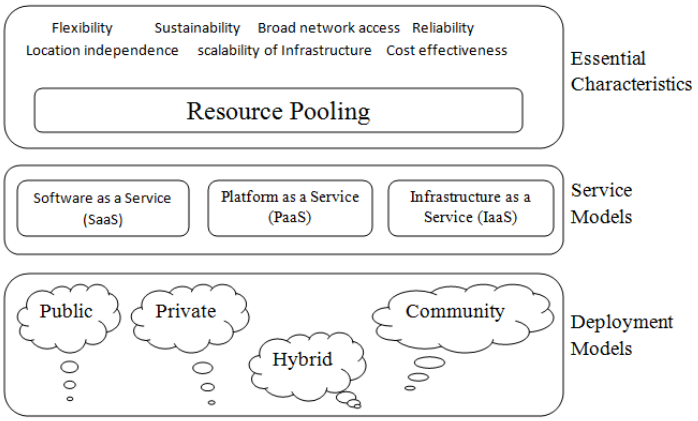


Fig.6: Virtual model of cloud computing definition.

2.2.3 In computing world, cloud computing can be viewed as one of the most auspicious technologies, which inherently describes a number of issues. Some of the "key characteristics of cloud computing" has been identified [17, 18]:

1. **Flexibility:** End user can quickly provision computing assets, as required, without the need of any human interaction. Capabilities can be elastically and rapidly supplied, in few cases automatically, to quickly scale up or down.
2. **Broad network access:** Capabilities are accessible over the network and are available through quality procedure that promotes use by heterogeneous platforms like laptops, mobile phones, and PDA.
3. **Scalability of infrastructure:** fresh nodes can be added up or dropped down from the network as can physical server, with slight alteration to software or infrastructure set up. According to demand, cloud architecture can be scaled up horizontally and vertically.
4. **Reliability:** enhances through the utilization of numerous redundant sites, which lets cloud computing acceptable for the flow of business and disaster recuperation.
5. **Location independence:** There is a sense of location, in that the end-user normally has no control or knowledge over the accurate location of the available resources, but may be able to identify location at a higher level of abstraction.
6. **Economics of scale and cost effectiveness:** The implementation of the cloud, regardless of the deployment models, tends out to be as vast as possible in order to take benefit of economies of scale. Most of the time clouds are deployed close to low-cost power station and in low-priced real estate, to reduce cost.
7. **Sustainability:** comes about through ameliorate resource utilization and carbon neutrality.

2.2.4 Main aspects of cloud computing:

In figure 7, we have classified cloud computing into eight main categories. These include service delivery model, comparison, layers, roles, deployment models, features, locality, and gaps which is the latest addition. Each of the listed features has at least three sub aspects. All the sub aspects are attached to relevant main aspects. Among the eight main features, 'gaps' is one of the major aspects because it is very much important to be taken into consideration and cannot be ignored. Moreover, our proclamation is that security threats, trust issues, security risk and some other specific cloud security related issues are the major gaps of cloud computing.

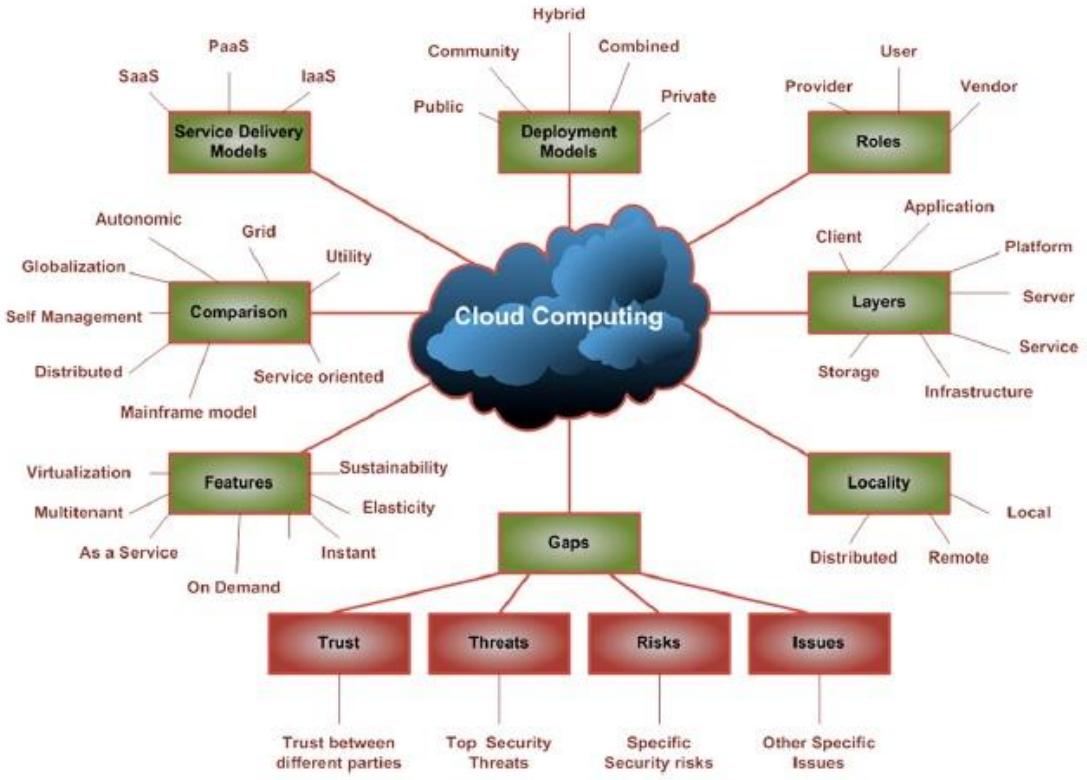


Fig.7: Understanding cloud computing [1].

2.2.5 Major gaps of cloud computing, factors that are interrupting migration to cloud computing from existing system: Cloud computing takeover all the security problem from the existing system, for example grid computing, plus the issues of security that has been established due to its unique architecture and characteristics.

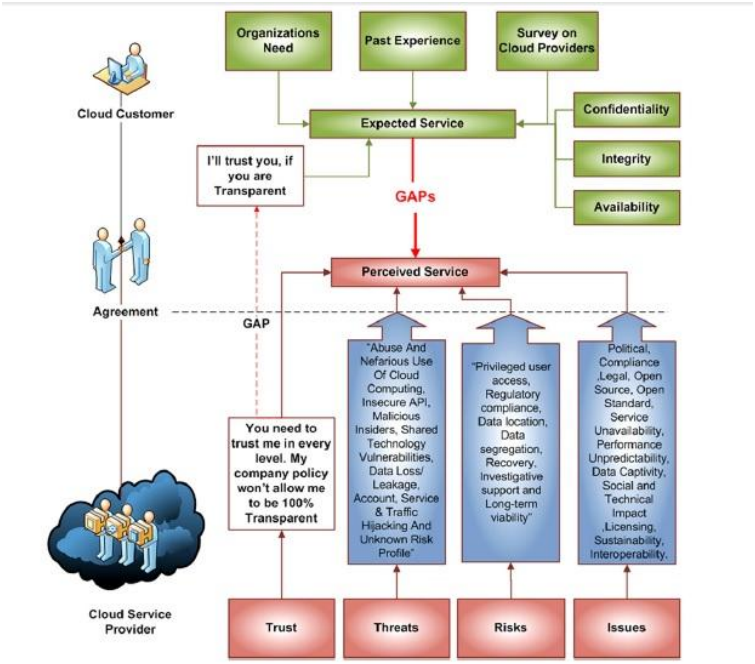


Fig.8: Understanding cloud computing gaps [1].

2.2.6 Some of the major barriers that restrict the broader adoption of cloud:

According to National Institute of Standards and Technology [19]:

1. Security.
2. Portability.
3. Interoperability.

Ness [20] identifies three major barriers:

1. Cloud depends on fresh approaches to security.
2. Cloud can break down the static networks.
3. Network automation is critical.

2.2.7 10 obstacles to cloud computing [21]:

1. Data lock in.
2. Data transfer bottlenecks.
3. Availability of services.
4. Data confidentiality and auditability.
5. Scalable storage.
6. Scaling quickly.
7. Bugs in large distributed system.
8. Performance unpredictability.
9. Software licensing.
10. Reputation fate sharing.

2.2.8 Review of cloud computing security:

There are three sections in which cloud computing security has been organized, shown in figure 9:

1. Security in service delivery models.
2. Security Categories.
3. Security dimensions.

2.2.9 Crucial security concern in cloud computing: this encourages new customers to ask difficult questions and consider getting evaluation from a impartial third party before pledging to a cloud provider.

The security farm Gartner in June 2008 issued a report entitled 'Assessing the Security risk of cloud computing' [22]. In it, they pointed out seven distinct security issues that the end users should raise with vendors prior to selecting a cloud provider. Some of the specific issues are:

1. Data location.
2. Recovery.
3. Long-term viability.
4. Privileged user access.
5. Investigative support.

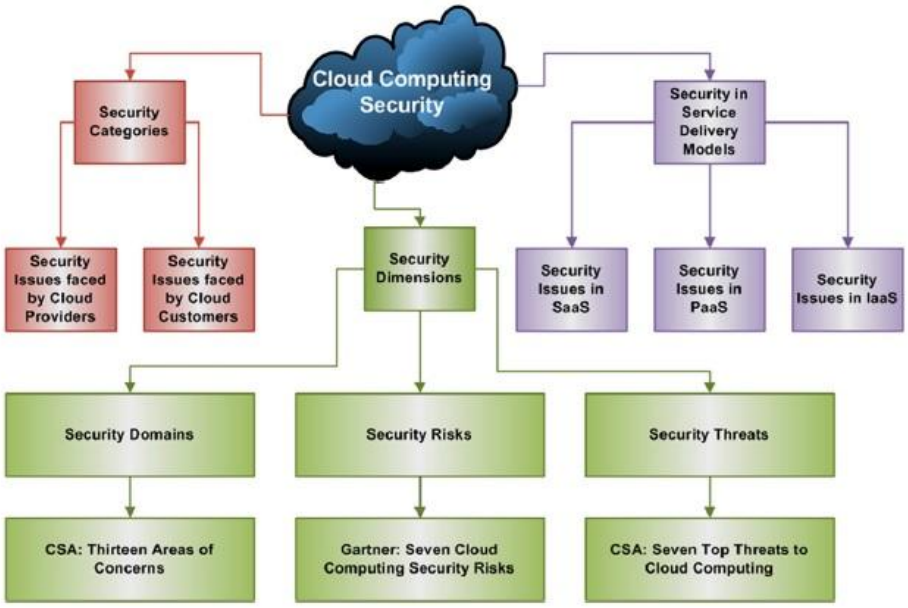


Fig.9: Graphical view of cloud computing security [1].

6. Regulatory compliance.
7. Data segregation.

The European Network and Information Security Agency (ENISA) in November 2009, issued additional research report entitled 'Cloud Computing: Risks, Benefits and Recommendation for Information Security' [23]. The record lists eight crucial cloud-specific risks which include:

1. Malicious insider.
2. Data protection.
3. Compliance risk.
4. Insecure or incomplete data deletion.
5. Management interface compromise.
6. Isolation failure.

7. Lock-in.
8. Loss of governance.

'Security Guidance for Critical Areas of Focus in Cloud Computing', a document released by Cloud Security Alliance in December 2009 [24], in which they have identified three major sections with thirteen areas of concern:

Section 1: Governing in the cloud.

Domain 1: Legal and electronic discovery.

Domain 2: Information lifecycle management.

Domain 3: Compliance and audit.

Domain 4: Portability and interoperability.

Domain 5: Governance and enterprise risk management.

Section 2: Cloud architecture.

Domain 6: Cloud computing architectural framework.

Section 3: Operating in the cloud.

Domain 7: Data centre operations.

Domain 8: Application Security.

Domain 9: Identity and access management.

Domain 10: Traditional security, disaster recovery and business continuity.

Domain 11: Incident response, notification and remediation.

Domain 12: Encryption and key management.

Domain 13: Virtualization.

In March 2010, research discovery on the cloud computing has been published by the CSA. The motive of the research was to facilitate cloud providers and also their potential customers in identifying the important risks and to help them determine whether or not to link with cloud infrastructure, and also how to actively guard them from these risks. Seven major threats to cloud computing are mentioned below [25]:

1. Insecure application programming interface.
2. Shared technology vulnerabilities.
3. Account, service and traffic hijacking.
4. Abuse and nefarious use of cloud computing.
5. Malicious insiders.
6. Data loss/leakage.
7. Unknown risk profile.

2.2.10 Before committing to any cloud provider, there are some specific risks that a user should raise:

1. A costumer/end user must validate if the infrastructure meet the specified standards with some regulatory security needs.
2. Sensitive information must be processed outside the perimeter of an organization with the confidence that they are only available and communicated to privileged users.
3. The data of one customer must be perfectly shielded from those of another customer
4. The cloud vendor must provide replication and disaster recovery mechanism.
5. Information must be available even when the service provider is acquired by some other organization of if the user shifts to another provider.
6. The cloud provider must pledge to store and process data in particular jurisdiction and adhere to local privacy requirements on behalf of the end-user, who don't know where data is banked.
7. Investigative back-up needs to be ensured.

2.3 Intrusion detection system in cloud computing

Although IDS have been evaluated, having the ability of protecting securely in large scale networks, but the deployment and utilization in cloud computing faces numerous difficulties [26]. The heterogeneity of cloud services customers and the complexity of its architecture results in different needs and prospects for being secured by IDS. Cloud computing lifts all the existing network security issues, in addition to security issues generated by its peculiar features and architecture [1]. In order to tackle the needs of IDS for cloud computing, first of all we focus at the unique characteristics of cloud computing systems and facing summons of IDS development in cloud computing. Then the present available systems are examined on the basis of their effectiveness and efficiency in order to be implemented on cloud computing environment. At last, the requirements are developed corresponding to cloud computing systems' attribute and the required characteristics of IDS.

2.3.1 Finding out the exact characteristics of a particular environment helps in developing the system and establishing the system requirements:

CC1: Elasticity is one of the most crucial feature for cloud computing, which limits the underlying infrastructure potential to habituate to changing needs such as size and amount of data used in application. Cloud computing deals with two types of scalability: horizontal and vertical. The vertical scalability is concerned with the size of the instances and implied to the

quantity of resources which are needed for maintaining the size. While, the horizontal scalability stands for the amount of instances to gratify changing amount of request.

CC2: Reliability is the proficiency of confirming the progression of the system operation without disruption such as code reset at the time of execution or the loss of data. Reliability is basically attained through utilizing unemployed resources. There is a strong link between reliability and availability. However, reliability is concerned in particular on prevention of loss of data or the execution progress.

CC3: Quality of Service support is essentially significant for specific needs which should be met through the provided resources or services. In order to verify that the acknowledged service quality of the cloud customer in Service Level Agreement is met the primary metrics of QoS such as response time, safety and throughput must be promised.

CC4: Adaptability and agility are the two main features of great importance to cloud systems that are closely connected to the elastics capabilities. They deal with the on-time reaction to modification in the size of resources and the number of requests as well as adjustment to changes according to the state of the environment. This adoption may demand distinct types of resources, contrasting routes or even different qualities. Precisely, adaptability and agility demands for the management of assets to be autonomic.

CC5: Availability of services depends upon the ability of providing redundant services and data to disguise failures transparently. Fault tolerance also requires this potential to introduce novel redundancy such as fresh or earlier failed nodes, in an online vague without or with minimal performance penalty. With this rise concurrent access, availability is achieved through duplication of services or data spread them across various resources. This account to the primary quintessence of scalability in cloud systems and services.

In brief, it can be pre-assumed that the present systems are not efficient and effective to be deployed on cloud computing environments which have their own unique essence and requirement. There is no traditional IDS to meet these features systematically.

2.3.2 Challenges of IDS deployment in cloud computing environment

It is of great concern to spot the challenges which have its origin from cloud computing phenomena before deploying IDS. Some of the unique challenges that developers face while developing IDS for cloud computing environment:

1. In conventional IDS, due to static essence of the observed systems, the policies seem to be static since the node groups have fixed requirements which have been discovered over time. Differing from the traditional mode, the observed virtual

systems are dynamically removed and added. Moreover the security needs of each virtual machine tend to be varied [27].

2. The policies of the security are usually managed and established by a system administrator, who is responsible for the security of the complete system. Having several system security administrators in cloud, poses negative effects on intrusion response time. The human interference would slow down the response time.
3. Vulnerabilities in cloud infrastructure increases because of shared infrastructure and virtualization technology. Any flaw in hypervisor makes it vulnerable to inappropriate access, which enable/permit creating virtual machines and running different operating systems [28].
4. Data transfer cost is one of the most important issue in cloud computing. For example, the data transfer cost in Amazon cloud is about $100 to $500 per terabyte. Hence, new researchers should make an effort to come up with data cost effective solutions for IDS in cloud environment which helps in lowering down the network bandwidth [29].
5. New issues deal with visibility into the inter virtual machine congestion on a virtual host platform, because the switch is also virtualized, thus common explication for physical monitoring are not being able to scrutinize this network traffic. The new platform of virtualization should be monitored and checked for configuration errors as they would have vulnerabilities that may lead to huge compromise [30].
6. To provide a risk profile, each company maintains the security strategy. But, the providers of cloud services are not willing to provide the audit data, security logs and security practices [31]. Due to lack of transparency on the security management policies such as logging, vulnerabilities, auditing, security policies and incident response results in inefficiency of classical risk management approach in the truancy of customer awareness, Cloud-Security-Alliance, 2010.

Table 2, displays the latest reviewed papers that are related to detection of intrusion in cloud. Their working features are very much similar to each other; the researches that have been conducted recently along with their proposed solution are still very far from ideal IDS for cloud computing, as they not only fall short in required IDS characteristics (DCs), but they also failed to achieve cloud systems characteristics (CCs). They have been able to satisfied CC5 and CC1, CC4, CC2 partially, but they are not being able to address all the features which should be addressed in their systems. This incompetency is the evidence to tha lack of knowledge of suitable requirements which should be identified before starting any

development.

Reference	Year	Detection technique	Technology layout	Time of detect	Response type	Audit source location	Management structure	Data diffusion	Prevention capability
Vieira et al. (2010)	2009	Hybrid (signature & anomaly)	N/A	Real time	Active	Host & Network	Collaborative	Distributed	No
Dastjerdi et al. (2009)	2010	N/A	Wireless (mobile-agent)	Real time	N/A	Network	Collaborative	Distributed	Yes
Tupakula et al. (2011)	2011	Hybrid (signature & anomaly)	N/A	Real time	Active	Network	Individual	Distributed	Yes
Gustavo and Miguel (2011)	2011	Anomaly	N/A	Real time	Active	Network	N/A	Distributed	No
Smith et al. (2010)	2010	Anomaly	N/A	Real time	Active	N/A	N/A	Distributed	No
Lee et al. (2011)	2011	Anomaly	N/A	Real time	Active	Host & Network	Individual	Distributed	No
Kholidy and Baiardi (2012)	2012	Hybrid (signature & anomaly)	Wireless	Real time	Active	Host & Network	Collaborative	Distributed	No
Dhage et al. (2011)	2011	Anomaly	N/A	Real time	Active	Host	Individual	Distributed	No
Takahashi et al. (2010)	2010	Anomaly	N/A	Real time	Active	Network	Collaborative	Distributed	Yes
Jin et al. (2011)	2011	Anomaly	N/A	Real time	Active	Network	Collaborative	Distributed	Yes

N/A=Not applicable.

Table 2: Proposed CIDS for cloud computing [3].

2.3.3 Taking the characteristics of an ideal IDS and that of cloud computing into account, the requirement of CIDS are identified from high-level point of view:

R1: Control of large scale data processing environments and dynamic multi-tiered autonomous computing

Cloud can be defined as large scale virtual machine dependent systems which are created automatically, deleted and migrated on request of a user at runtime. Most of the time, it is expected that the middleware controller initially informed from the modification in the resources, on the other hand in cloud computing which require large scale networks and systems it is essential to maintain these modification automatically without human involvement. To conquer the complexity of its dynamic nature, the intrusion detection system of cloud should be able to guide itself with minimal or no human meddling which accelerate the control and monitoring of network elements in real-time. This requirement looks after CC2 and CC3.

R2: Should identify variety of attacks with minimum False Positive Rates

Due to increase in the growth of complexity, attacks and unpredictability, it is obligatory for the system to identify the recently developed attacks and their vulnerable aim to choose the best acknowledgement according to the risk acuteness and proper prevention strategy. The system must be learnable and should be able to brush-up its detection capability over time to support DC6. It also demands to be organised to sustain a required level of performance and security at the same time with minimal computation resources as the efficiency of cloud services depends on its computation capability. Therefore, logical approach should be utilied to control false positive alarms retaining detection performance. It covers CC3.

R3: Superfast detection and prevention

Intense detection and prevention is a very critical enabling factor for CIDS since it influence the entire system performance and is very important to deliver the promised Q0S. It refers to CC3 and CC2. A cloud based system with a number of administrators should have very less or no human involvement to evade wasting time for administration responses. The system must work in real-time and deliver automated responses to doubtful activities. This satisfies DC5 and DC1.

R4: Self-Adaptive Autonomic ally

The property of easy to adjust to the cloud conditions in area which it is supposed to work is very crucial. A CIDS should configure itself and be reconcilable to configuration amendments as computing nodes are dynamically removed and added. Mapping a acceptable architecture of collaborative IDS would dictate how the alerts should be shared processed from sole detection components with preserving a topological model of cloud computing. This also assists controlling and monitoring network components. Blueprint of such systems be pliable enough to be able to incorporate future deployed standards. It supports DC4, DC3, DC8, CC5 and CC4.

R5: CIDS Scalability

A CIDS must be scalable in order to systematically handle the huge number of network nodes present in cloud and their computational and communication burden. It should be able to scale itself horizontally or vertically as the nodes added to fit into large cloud perimeter. The positioning of detection and correlation handler also influences the scalability and potential of CIDS. It refers to CC1.

R6: Deterministic

Cloud computing offers pivotal and critical operational services which have certain performance prerequisite, in terms of reliability, latency and resilience. Supporting DC9, DC2, CC3, CC2 and CC5, a CIDS must be able to offer and maintain satisfactory level of assistance in the face of faults, be highly dedicated and must deliver very high uptime services with imposing least overhead. A CIDS should not only establish real time performance, but also guarantee that the optimistic nature of network is not negatively affected. On the other hand, the functioning of the monitored systems must not be influenced by excessive burden of the CIDS. It is essential to verify that the CIDS has enough ability to process all the information. Dividing the computational load and diagnostic capabilities to autonomous agents over the network would raise the level of fault tolerance.

R7: Synchronization of autonomous CIDS

A CIDS is a collaborative IDS consisting of a huge number of autonomous IDS. While each and every system operates and detects anomalies and intrusions independently, their information and activities need to be synchronized in order to identify concurrent and distributed attacks, apply suitable response and alter o specific component system or entore network configuration, and embrace proper prevention scheme.

R8: Resistance to Compromise

Referring to DC7, a CIDS must secure itself from unauthorized attacks and access. A CIDS must be capable of authenticating IDS and network devices mutually, authenticating the security administrator and examine his activity, protecting its information and blocking any loopholes which may generate additional vulnerabilities.

All the new solutions for establishment of a CIDS must be taken into consideration in order to surpass the complexities of cloud computing and met the real operational aims of the cloud computing world.

Table 3, shows the developed CIDS which met our proposed requirements.

References requirements	R1	R2	R3	R4	R5	R6	R7	R8
Vieira et al. (2010)	×	√	×	√	√	√	×	√
Dastjerdi et al. (2009)	×	N/A	N/A	√	√	√	×	N/A
Tupakula et al. (2011)	×	P	×	√	√	×	N/A	√
Gustavo and Miguel (2011)	×	P	×	N/A	N/A	×	×	√
Smith et al. (2010)	×	√	P	N/A	N/A	P	N/A	P
Lee et al. (2011)	×	√	×	N/A	N/A	√	N/A	×
Kholidy and Baiardi (2012)	P	√	×	√	√	√	×	√
Dhage et al. (2011)	×	√	N/A	√	√	×	N/A	√
Takahashi et al. (2010)	√	√	√	√	N/A	P	×	√
Jin et al. (2011)	√	P	×	√	√	P	N/A	N/A

P=Partially X=Does not meet requirement √=meet requirement N/A=Not applicable.

Table 3: Developed CIDS which met our proposed requirement [3].

2.3.4 A proper defence strategy for cloud systems:

1. Must avoid any single point of failure.
2. Should have a flexible architecture that can be applied to several cloud architecture.
3. Consider different user requirements and service models.
4. Be scalable and distribute to adapt to the cloud characteristics.
5. Must be able to protect and isolate the IDS from the vulnerabilities of the host machine.
6. Integrate both knowledge and behaviour based techniques.

Chapter 3: Research Motivation and Problem Statement

3.1 Research Motivation

The open and utterly distributed architecture of cloud computing and services becomes even more appealing target of potential attackers/intruders. It involves service oriented and multi-mesh distributed paradigm, multi-domains, multi-tenancies, and multi-user self-governing administrative infrastructure which are vulnerable and susceptible to security risks. The service architecture of cloud computing combines three layers of inter-dependent application, platform and infrastructure; where each layer may have certain vulnerabilities which are put forward by different configuration or programming errors of the service provider or the user. A cloud computing architecture can be vulnerable to several threats; these include threats to integrity, availability and confidentiality of its data, resources and the virtualized infrastructure that can be used as a launching pad for modern attacks (Cloud-Security-Alliance, 2010). The situation becomes even more complex when a cloud with huge storage capacity and computing power is exploited by an insider, thus for cloud services' customer the lack of full control over the infrastructure is a supreme concern. It signifies the role and importance of IDS in securing the customers' information assets in cloud environment.

3.2 Problem Statement

Conventional IDS when implemented on a cloud environment generates a large volume of alerts regardless of all the progress made over the last few years. The investigation of such a huge number of raw alerts from a large cloud environment is usually very much time consuming and requires a lot of efforts because the important or relevant alerts are usually buried under the irrelevant alerts. Thus the approaches related to alert management have received a considerable attention and appears very much promising in improving the quality of alerts. Hence we need to design a smart IDS which when installed on a cloud hypervisor can filter out all those alerts that does not have any relation to any anomalous activities or vulnerability, and also summarizes and parses the alerts with similar attributes (alerts with same destination IP, signature, timestamp), which enables the security administrator to concentrate on the important alerts and thus can take action in real time.

Chapter 4: Implementation and Results

Consider that a mysterious man is standing\walking in front of your home. He looks around, observing the surroundings, and then steps towards the front door and starts moving the knob, he found that the door is locked. He walks towards a nearby window and smoothly tries to open it, that to was locked. It can be considered that your house is secure. So why to install an alarm? This query is often asked to intrusion detection advocates. Why to concern about detecting intrusions if you have already installed firewalls, patched operating systems, etc…? The reason is simple: as intrusions still occur. Just as people sometimes forget to lock a window, for example, they sometimes miss to properly update a firewall's rule set. Even with the availability of different advanced protection mechanism, computer machines are still not completely secure. Hence, the above stated philosophy signifies the importance of implementing IDS in any organization to protect its technology and information.

4.1 Operating System and Configuration setup

4.1.1 Configuration of Main Machine:

1. Host Operating System Version: Windows 7 Home Basic, 64-bit 6.1.7601, Service Pack 1.
2. Processor: Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz.
3. Installed Memory (RAM): 3.0 GB.
4. System Type: 64-bit Operating System.
5. HDD: 320GB.
6. Hypervisor Information.
    a. Product: VMware® Workstation.
    b. Version: 7.1.2 build-301548.

4.1.2 CIDS Node setup:

1. Operating System Version: Microsoft Windows XP Professional Version 2002 SP3.
2. Processor: Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz.
3. Installed Memory (RAM): 1.0 GB.
4. System Type: 32-bit Operating System.
5. HDD: 40GB (C: (System) 20GB, E: (CIDS) 20GB).

4.2 Downloading core 'Cloud Intrusion Detection Systems (CIDS) Software (Save all the downloaded files to *e:\temp folder*):

1. adodb518a.zip.
2. Barnyard2-1.13-build327.zip.
3. Base-1.4.5.zip.
4. dotNetFx40_Full_x86_x64.exe.
5. npp.5.7.Installer_3.exe.
6. Snort_2_9_5_6_Installer.exe.
7. Snortrules-snapshot-2956.tar.gz.
8. Strawberry-perl-5.14.2.1-32bit.msi.
9. Httpd-2.4.9-win32.zip.
10. Mod_fcgid-2.3.9-win32.zip.
11. Mysql-installer-community-5.6.17.0.msi.
12. Mysql-gui-tools-5.0-r17-win32.msi.
13. Php-5.4.26-Win32-VC9-x86.zip.

4.3 Environment:

1. The detection engine of snort will be running in passive mode, logging event to a unified2 log files.
2. Barnyard2 will be processing Cloud Intrusion Detection System (CIDS) unified2 log files.
3. A database driven by MySQL will store processed events/logs for further analysis.
4. Apache2 will drive the Cloud Intrusion Detection Systems (CIDS) web based GUI security console.
5. BASE will serve as the Cloud Intrusion Detection Systems (CIDS) web based GUI security console.

4.4 Steps for Execution:

1. Run VMware Workstation with administrative privileges.
2. Setup the Cloud environment by creating different nodes having different operating systems, utilizing the resources of the main machine.
3. Create a host IDS machine running over hypervisor (VMware Workstation), with Windows XP Pro SP3 as running OS.

4. Initially there is only one HDD i.e. ‘C:’ with 40 GB of space, thus partition the HDD into two parts ‘C:’ and ‘E:’ each with 20 GB of space, using the software PAssist_Std.

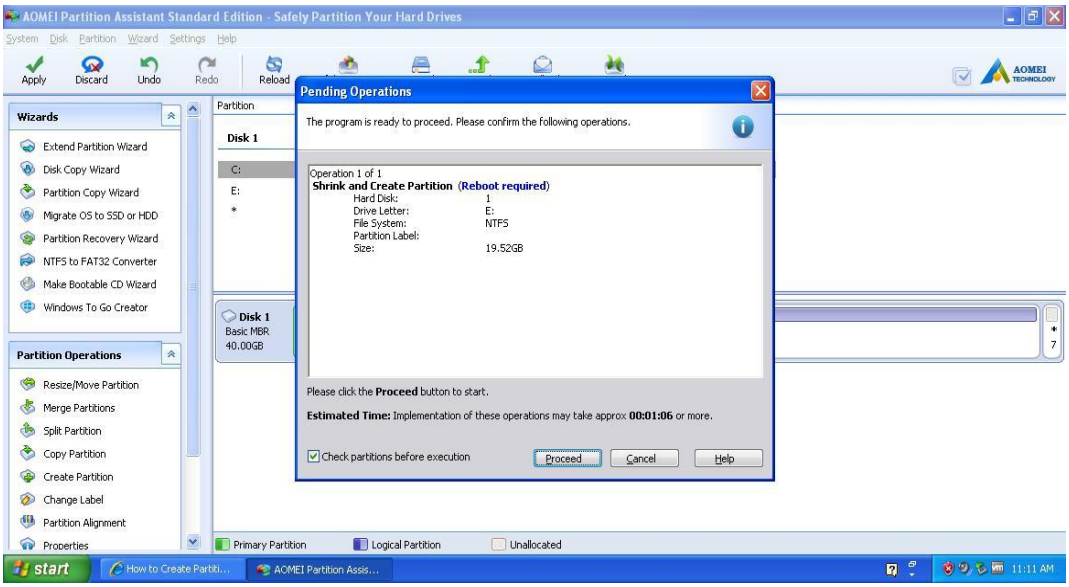


Fig 10: Partitioning HDD into two drives, ‘C’ and ‘E’.

5. Create folders named ‘temp’ and ‘winids’ inside ‘E:\’, all the software and scripts that we will download are saved to ‘E:\temp’ folder while the complete installation of CIDS will be done into ‘E:\winids’ folder.
6. Installing WinPcap: In a CMD window type the command “e:\temp\WinPcap_4_1_3.exe” and press enter key.
7. Installing Snort: type ‘e:\temp\ Snort_2_9_5_6_Installer.exe in the CMD window and tap enter. Note: in the 'Destination Folder' dialog box, type 'e: \winids\snort'.

Fig.11: Successful installation of Snort.

8. Testing the Snort for network traffic: In the CMD window type 'e:\winids\snort\bin\snort -W' and press enter.

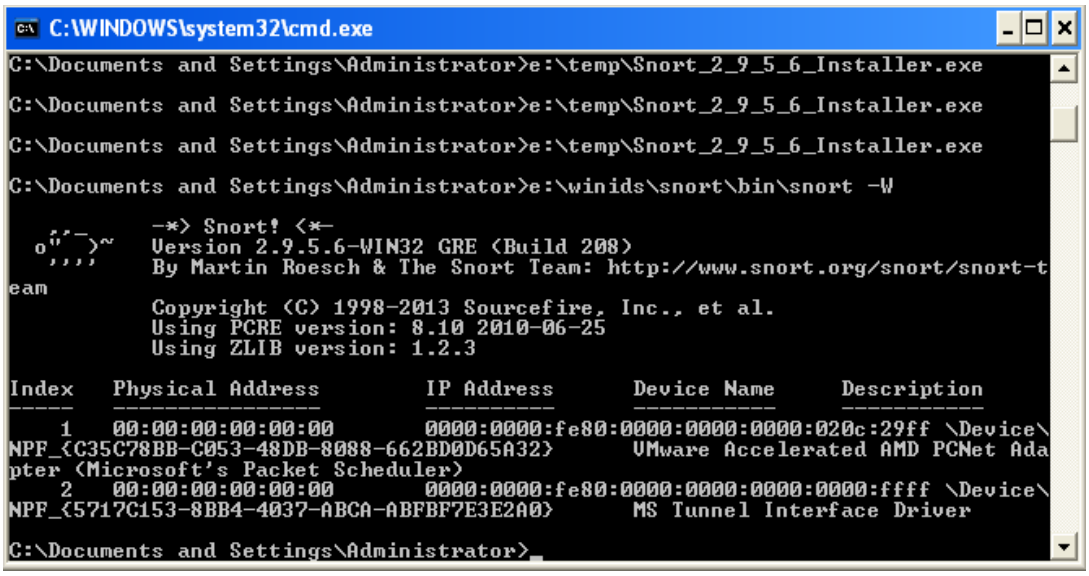


Fig.12: Testing Snort for network traffic.

9. To verify that there is network traffic at interface '1', type the command on CMD window 'e:\winids\snort\bin\snort -v –i1'.

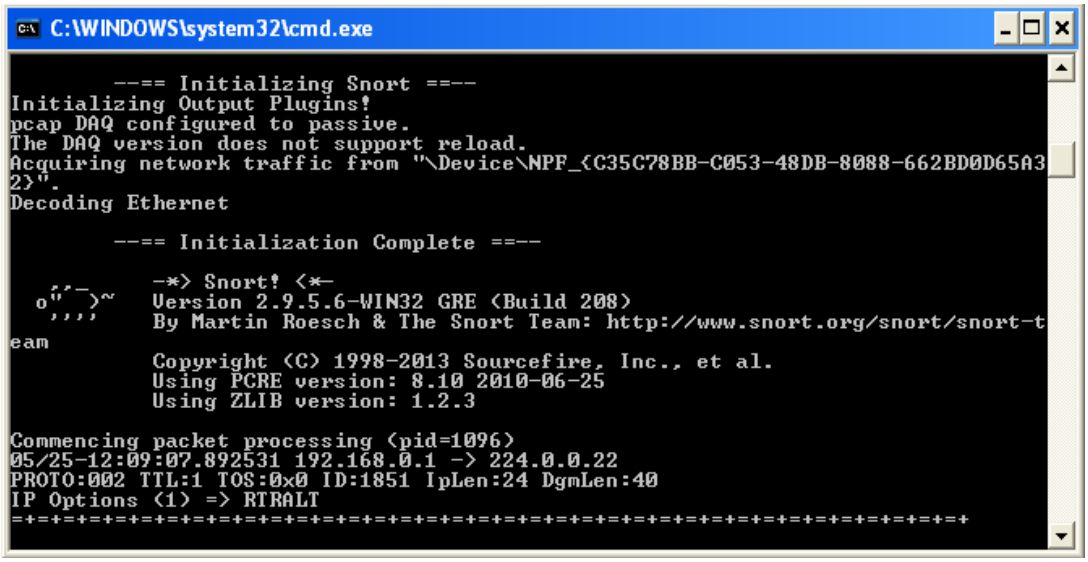


Fig.13: Initialization Snort.

10. Register on the Snort website https://www.snort.org/signup in order to download Snort Rules. After registration completes, you will automatically redirected to the download page.
11. Click the Get Rules button to download the latest rules. In this project we have downloaded snortrules-snapshot-2956.tar.gz.

12. Extract the downloaded rules and copy the following folders named rules, so_rules, and preproc_rules to 'E: \winids\snort'.
13. Installing Strawberry Perl: type 'e: \temp\strawberry-perl-5.14.2.1-32bit.msi', on the CMD window and press the Enter key. Note: In the 'Install Strawberry Perl to:' dialog box type 'e: \winids\strawberry\'.
14. Installing the Apache2 Web-Server: type the command 'unzip -oqq e:\temp\httpd-2.4.9-win32.zip -d e: \winids' and press Enter key.
15. Installing of FastCGI ASF support module for Apache2: type 'unzip -joqq e:\temp\mod_fcgid-2.3.9-win32.zip"mod_fcgid*.so"-de: \winids\Apache24\modules' in the CMD window and press the Enter key.
16. Installing the security console for CIDS, BASE: type 'unzip -oqq d:\temp\base-1.4.5.zip -d d:\winids\apache24\htdocs\base' in the CMD window and press the Enter key.
17. Installing Barnyard2: Barnyard2 is responsible for processing and parsing Snort's unified2 log files sending them to a database server where they will be used for security analysis. To install type 'unzip -oqq e:\temp\barnyard2-1.13-build327.zip -d e: \winids\barnyard2' on CMD window and press the Enter key.
18. Installing MySQL Database Server: Double click on 'mysql-installer-community-5.6.17.0.msi' present inside 'E: \temp'. Note: In the 'Installation Path:' dialog box type 'e\winids\mysql\' and in the 'Data Path:' dialog box type 'E: \winids\mysql\'. MySQL Root Password is 'd1ngd0ng'.
19. After the successful installation of MySQL go to CMD window and type 'copy "e:\winids\mysql\MySQL Server 5.6\lib\libmysql.dll" c: \windows\system32'.
20. Installing ADOBD: to install ADOBD type the following command in CMD window 'unzip -oqq e:\temp\adodb518a.zip -d e: \winids' and tap the Enter key.
21. Installing PHP: At the CMD prompt type 'unzip -oqq e:\temp\php-5.4.26-Win32-VC9-x86.zip -d e: \winids\php', and tap the 'Enter' key.
22. Updating 'sid-msg.map' file: The file maps the Rule MSG alert name to the sid number allocated to the rule. Go to CMD window and type 'unzip -oqq e:\temp\activators.zip -d e:\winids\activators'. Now to update the file type the following command 'unzip -oqq e:\temp\create-sidmap.zip -d e: \winids\create-sidmap' and 'e:\winids\create-sidmap\create-sidmap.pl e:\winids\snort\rules\ > e: \winids\snort\etc\sid-msg.map'.
23. The Heart of CIDS-Snort Configuration:

a. Type the following commands in the CMD window.
   i. type NUL > e:\winids\snort\rules\white_list.rules
   ii. type NUL > e:\winids\snort\rules\black_list.rules
b. Now it's time to configure snort.conf file located at E:\winids\snort\etc.
c. Open the snort.conf file with Notepad++.
d. Scroll down to the Step #1: Set the network variables section of snort.conf file. In the HOME_NET line, replace any with the 192.168.0.0/24.

```
##################################################
# Step #1: Set the network variables.  For more informa
##################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.0.0/24
```

Fig.14: Configuring the network variables.

e. Scroll down to RULE_PATH, replace '../rules' with 'e:\winids\snort\rules', 'var SO_RULE_PATH ../so_rules' with '# var SO_RULE_PATH ../so_rules' and '../preproc_rules' with 'e:\winids\snort\preproc_rules', replace 'var WHITE_LIST_PATH ../rules' with 'var WHITE_LIST_PATH e: \winids\snort\rules' and 'var BLACK_LIST_PATH ../rules' with 'var BLACK_LIST_PATH e: \winids\snort\rules'.

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH e:\winids\snort\rules
# var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH e:\winids\snort\preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH e:\winids\snort\rules
var BLACK_LIST_PATH e:\winids\snort\rules
```

Fig.15: Configuring the RULE_PATH.

f. Scroll down to Step #4: Configure dynamic loaded libraries section. Configure dynamic loaded libraries in the section, modify path to dynamic pre-processor libraries and base pre-processor engine.

```
##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory e:\winids\snort\lib\snort_dynamicpreprocessor

# path to base preprocessor engine
dynamicengine e:\winids\snort\lib\snort_dynamicengine\sf_engine.dll

# path to dynamic rules libraries
# dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

Fig.16: Configuration of dynamic loaded libraries.

g. Scroll down to Step #5: Configure Pre-processors section, comment all the pre-processors listed in this section by adding # before each pre-processors and replace '# preprocessor sfportscan: proto { all } memcap { 10000000 } sense_level { low }' with 'preprocessor sfportscan: proto { all } memcap { 10000000 } sense_level { low }'.

h. Scroll down to Step #6: Configure output plugins. In this step, provide the location of the classification.config and reference.config files.

```
##################################################
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring
##################################################

# unified2
# Recommended for most installs
output unified2: filename merged.log, limit 128
```

```
# metadata reference data.  do not modify these lines
include e:\winids\snort\etc\classification.config
include e:\winids\snort\etc\reference.config
```

Fig.17: Configure output plugins.

i. In Step #8: Customize your preprocessor and decoder alerts; remove the '#' symbol from all the 3 decoder and preprocessor event rules,

```
##################################################
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
##################################################

# decoder and preprocessor event rules
include $PREPROC_RULE_PATH/preprocessor.rules
include $PREPROC_RULE_PATH/decoder.rules
include $PREPROC_RULE_PATH/sensitive-data.rules
```

Fig.18: Customise preprocessor.

j. In Step #9: Customize your Shared Object Snort Rules; replace 'include threshold.conf' with 'include d:\winids\snort\etc\threshold.conf'.

k. Save the Snort.conf file and exit Notepad++.

24. In order to check that the Snort.conf file is properly configured type in CMD window the following command: 'e:\winids\snort\bin\snort -c e:\winids\snort\etc\snort.conf -l e:\winids\snort\log –i1 -T'.

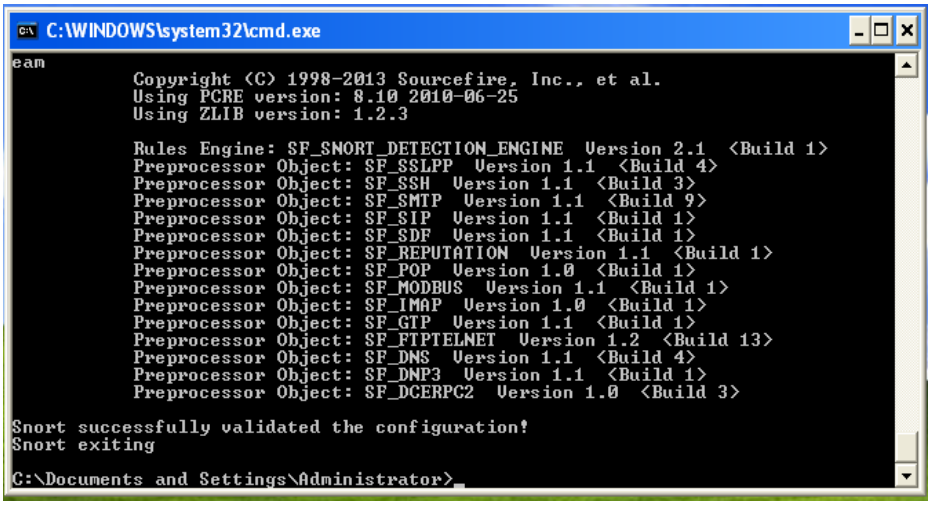


Fig.19: Validating the configuration of Snort.

25. Configuring the 'php.ini' file present inside 'e: \winids\php\' folder. To configure 'php.ini' open the file using Notepad++ and do the following changes:

a. Go to line no. 375 and replace max_execution_time from 30 to 60.

b. Go to line no. 452 and place ';' in front of the line.

c. Go to line no. 699 and set include_path to 'e:\winids\php; e: \winids\php\pear'.

d. Go to line no. 721 and set extension_dir to "e:\winids\php\ext".

e. Go to line no. 862 and remove ';' from the line '; extension=php_gd2.dll'.

f. Go to line no. 871 and remove ';' from the line '; extension=php_mysql.dl'.

g. Go to line no. 909 and set date.timezone to Asia/Calcutta.

h. Go to line no. 1386 and set session.save_path to "c:\windows\temp".

i. Save the file and exit.

26. Configuring the Apache2 Web-Server: open the 'httpd.conf' file present inside E:\winids\Apache24\conf in Notepad++.
    a. Go to line no. 37 and make change the ServerRoot directory to "e: /winids/Apache24", as 'ServerRoot "e: /winids/Apache24"'.
    b. Add line 'LoadModule fcgid_module modules/mod_fcgid.so', just below the line '#LoadModule xml2enc_module modules/mod_xml2enc.so'.
    c. At line no. 217 point ServerName to winids:80 as 'ServerName winids:80'.
    d. At line no. 241 modify line 'DocumentRoot "c:/Apache24/htdocs"' with 'DocumentRoot "e:/winids/apache24/htdocs'.
    e. At line 255 replace 'Options Indexes FollowSymLinks' with 'Options - Indexes'.
    f. At line 358 replace 'ScriptAlias /cgi-bin/ "c:/Apache24/cgi-bin/"' with 'ScriptAlias /cgi-bin/ "e:/winids/Apache24/cgi-bin/"'.
    h. Scroll all the way to the bottom of the file and insert the next 26 lines:

```
<IfModule fcgid_module>
    FcgidInitialEnv PHPRC "e:\winids\php"
    FcgidInitialEnv PATH "e:\winids\php;c:\Windows\system32;c:\Windows;c:Windows\System32\Wbem;"
    FcgidInitialEnv SystemRoot "c:\Windows"
    FcgidInitialEnv SystemDrive "c:"
    FcgidInitialEnv TEMP "c:\Windows\Temp"
    FcgidInitialEnv TMP "c:\Windows\Temp"
    FcgidInitialEnv windir "c:\Windows"
    FcgidIOTimeout 40
    FcgidConnectTimeout 10
    FcgidMaxProcesses 8
    FcgidOutputBufferSize 64
    ProcessLifeTime 0
    FcgidMaxRequestsPerProcess 0
    FcgidMinProcessesPerClass 0
    FcgidMaxProcesses 50
    FcgidFixPathinfo 1
```

```
FcgidZombieScanInterval 20
FcgidMaxRequestLen 536870912
FcgidIOTimeout 120

<Files ~ "\.php$">
  Options Indexes FollowSymLinks ExecCGI
  AddHandler fcgid-script .php
  FcgidWrapper "e:/winids/php/php-cgi.exe" .php
</Files>
</IfModule>
```

27. Save the 'httpd.conf' file and close the Notepad++.
28. Command to add Apache2 to Windows Services Database: 'e:\winids\apache24\bin\httpd.exe -k install', as soon as you press 'Enter' key it will display the following result.

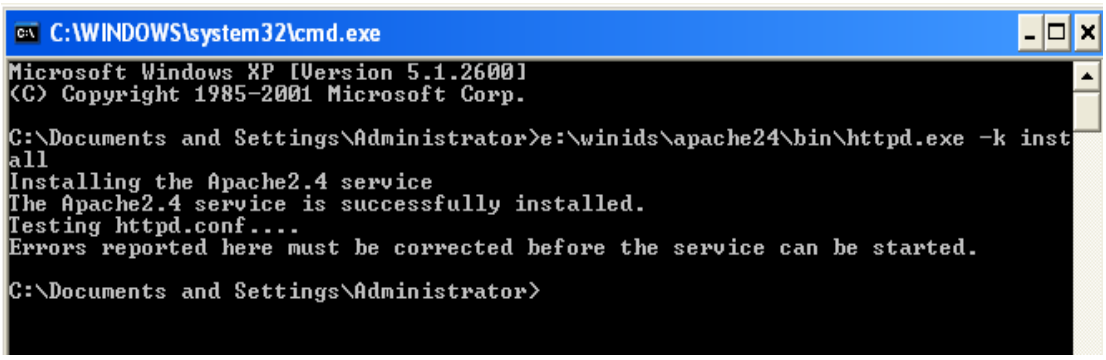


Fig.20: Installing the Apache2.4 service.

29. Now go to CMD window and type the following command 'net start apache2.4' and press the 'Enter' key.

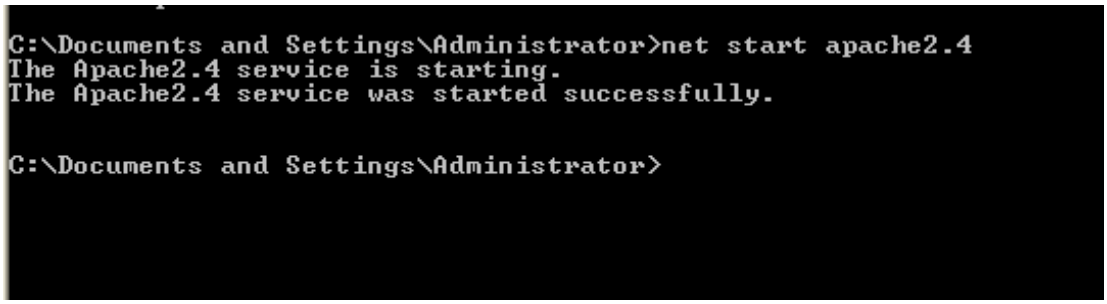


Fig.21: Starting Apache2.4

30. Testing the working of Apache2, and the PHP: in a web browser type 'http://winids/test.php' in the URL and press enter. It will show you the following screen.

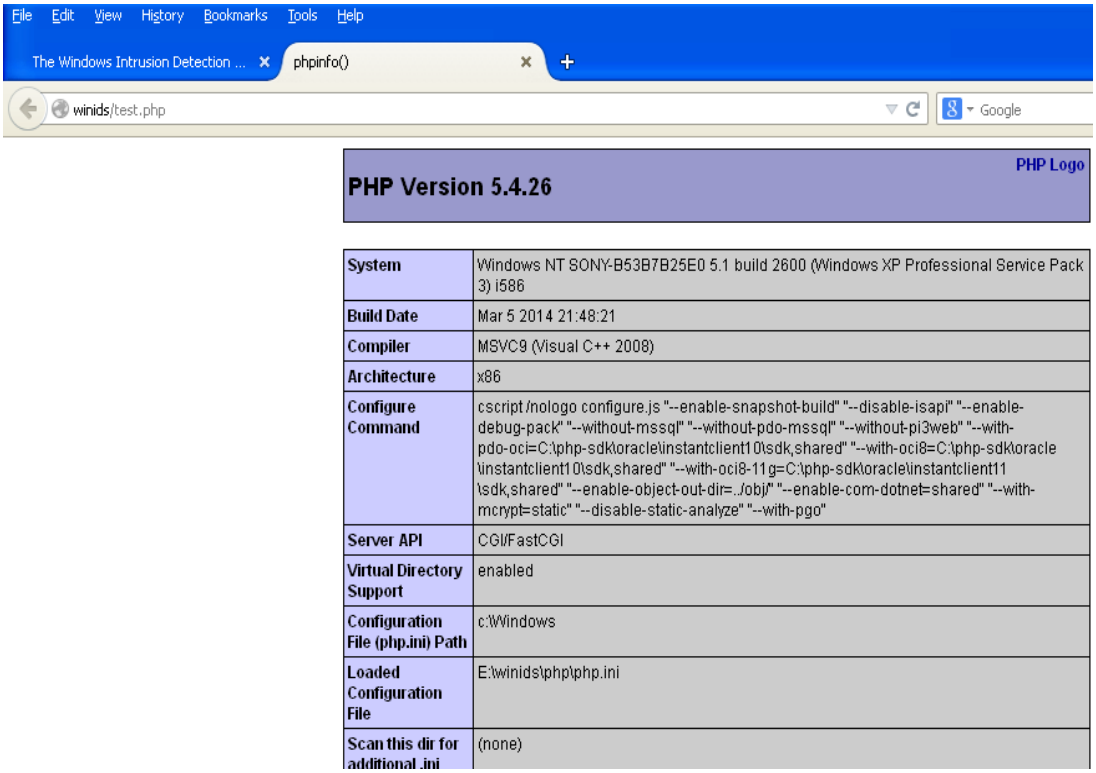


Fig.22: Testing the working of Apache2 and PHP.

31. Adding Snort to the Windows Services Database:

a. In the CMD window, change the dir to E:\winids\snort\bin and then type the following command 'snort /SERVICE /INSTALL -c e:\winids\snort\etc\snort.conf -l e:\winids\snort\log –il ' and press 'Enter' key.

b. Now type 'sc config snortsvc start= auto'. As you press 'Enter' key it will give you the following result: "[SC] ChangeServiceConfig SUCCESS".

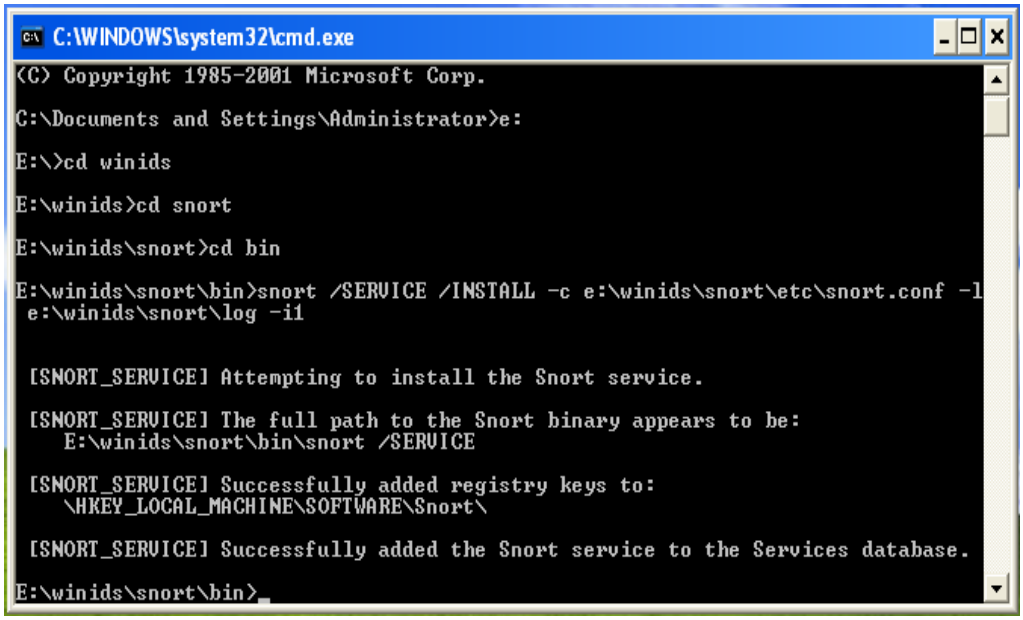


Fig.23: Adding Snort to windows service database.

32. Configuring the MySQL Database Server:

a. Open ‘my.ini’ file present inside ‘E:\winids\mysql’ folder and add the line ‘bind-address=127.0.0.1’ just below ‘[mysqld]’.

b. Creating the Cloud Intrusion Detection System Databases: In the CMD window change the directory to ‘E:\winids\mysql\MySQL Server 5.6\bin’ and type the following commands and press ‘Enter’ key after each command:

i. mysql -u root -pd1ngd0ng

ii. drop database test;

iii. create database snort;

iv. create database archive;

v. show databases;

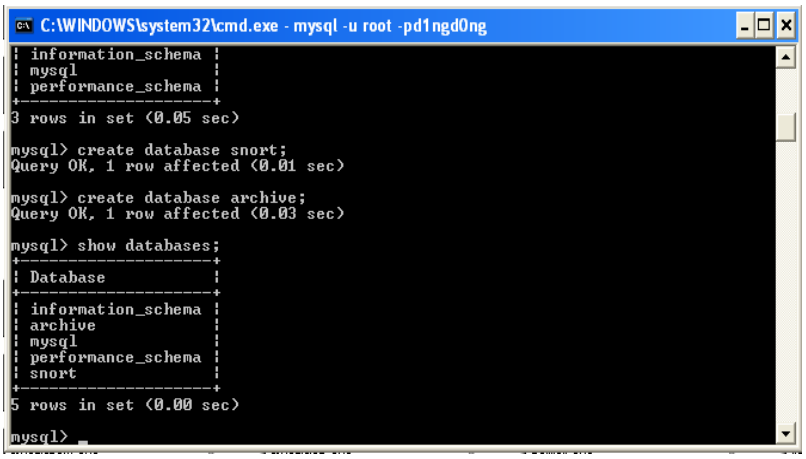


Fig.24: Creating CIDS database.

c. Creating the Cloud Intrusion Detection System Database Tables: following commands are used to create CIDS database tables:

i. connect snort;

ii. source d:/winids/barnyard2/schemas/create_mysql

iii. source d:\winids\apache24\htdocs\base\sql\create_base_tbls_mysql.sql

iv. show tables;

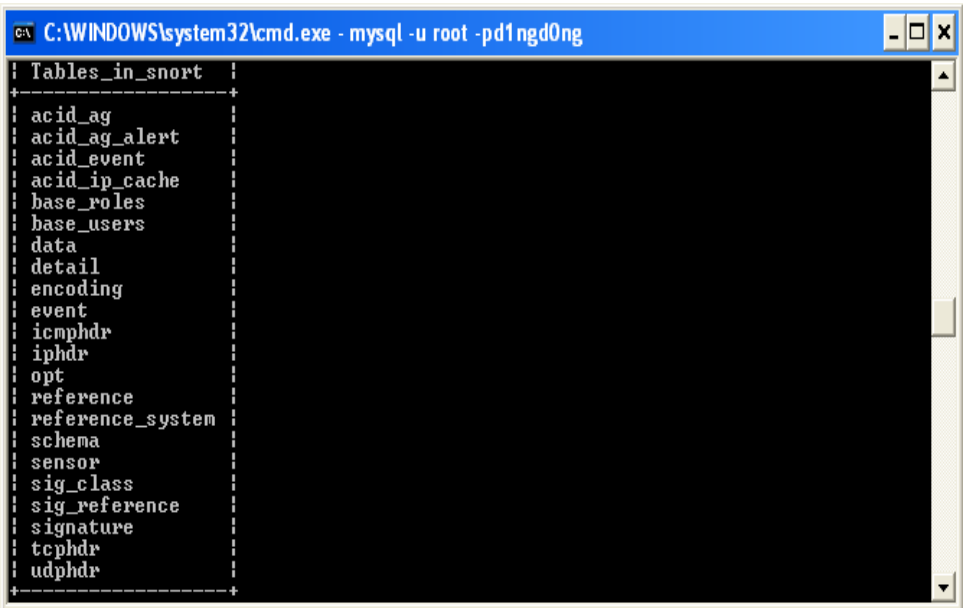


Fig.25: Creating CIDS database tables.

v. connect archive;

vi. source d:/winids/barnyard2/schemas/create_mysql

vii. source d:\winids\apache24\htdocs\base\sql\create_base_tbls_mysql.sql

viii. show tables;

d. Creating the CIDS Database Access, and Authenticated User: type the following command in the CMD window:

i. grant INSERT,SELECT,UPDATE on snort.* to snort identified by 'l0gg3r';

ii. grant INSERT,SELECT,UPDATE on snort.* to snort@localhost identified by 'l0gg3r';

iii. grant INSERT,SELECT,UPDATE,DELETE,CREATE on snort.* to base identified by 'an@l1st';

iv. grant INSERT,SELECT,UPDATE,DELETE,CREATE on snort.* to base@localhost identified by 'an@l1st';

v. grant INSERT,SELECT,UPDATE,DELETE,CREATE on archive.* to base identified by 'an@l1st';

vi. grant INSERT,SELECT,UPDATE,DELETE,CREATE on archive.* to base@localhost identified by 'an@l1st';

vii. Now insert command ‘use mysql;’ and then ‘select * from user;’ and tap the ‘Enter’ key.

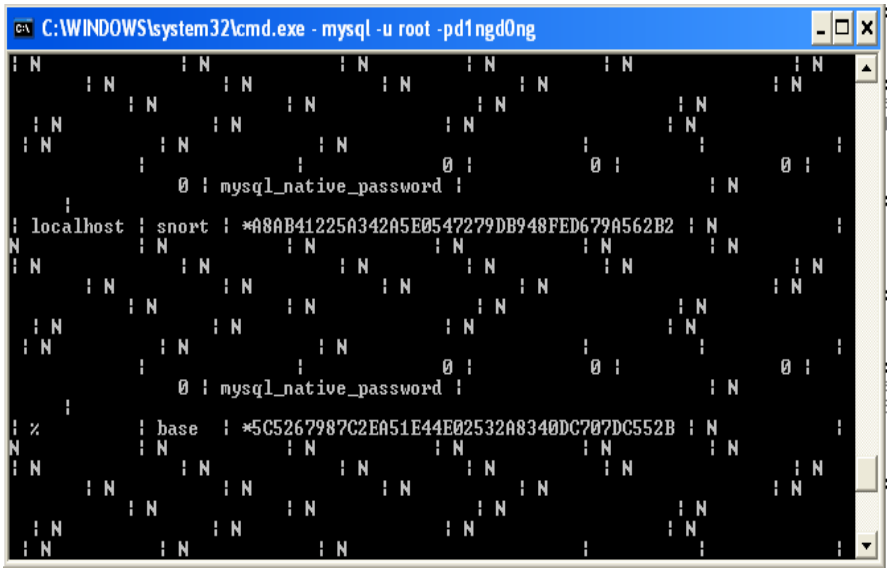


Fig.26: Creating the CIDS database access, and authenticated user.

33. Configuring the CIDS Security Consol.

a. First of all copy ‘base_conf.php.dist’ to ‘base_conf.php’ present inside ‘E:\winids\Apache24\htdocs\base’.

b. Extract ‘opensource.gz’ present inside ‘E:\temp’ to ‘E:\winids\Apache24\htdocs\base\signatures’.

c. Open ‘base_conf.php’ file, present inside ‘E:\winids\Apache24\htdocs\base’ in Notepad++.

i. Go to line no. 50 and replace ‘$BASE_urlpath = '';’ with ‘$BASE_urlpath = 'http://winids';’.

ii. Go to line no. 80 and replace ‘$DBlib_path = '';’ with ‘$DBlib_path = 'e:\winids\adodb5';’.

iii. Go to line no. 89 and replace ‘$DBtype = '?????';’ with ‘$DBtype = 'mysql';’.

iv. At line no. 101 set $alert_dbname to ‘snort’;

v. At line no. 102 set $alert_host to ‘winids’;

vi. At line no. 103 set $alert_port to ‘’;

vii. At line no. 104 set $alert_user to ‘base’;

viii. At line no. 105 set $alert_password to ‘an@l1st’;

ix. At line no. 108 set $archive_exists to ‘1’;

x. At line no. 109 set $archive_dbname to ‘archive’;

xi. At line no. 110 set $archive_host to ‘winids’;

xii. At line no. 111 set $archive_port to ‘’;

xiii. At line no. 112 set $archive_user to ‘base’;
xiv. At line no. 113 set $archive_password to ‘an@ll st’;
xv. At line no. 188 set $show_rows to 90;
xvi. At line no. 276 set $show_expanded_query to 1;
xvii. At line no. 430 set $colored_alerts to 1;
xviii. At line no. 433 set $priority_colors to array('000000','FF0000','FF9900','FFFF00','999999');
xix. At line no. 455 remove ‘//’ from the line ‘$graph_font_name = "Verdana";’
xx. At line no. 456 mark ’//’ in front of $graph_font_name = "DejaVuSans";
xxi. At line no. 474 replace ‘//$Geo_IPfree_file_ascii = "/var/www/html/ips-ascii.txt";’ with ‘$Geo_IPfree_file_ascii = "e:\winids\apache24\htdocs\base\ips-ascii.txt";’
xxii. Save the file and exit.

34. Configuring graphing for the CIDS Security Console: to configure, you need to do the following steps:
    a. Copy ‘go-pear.phar’ present inside ‘e:\temp’ folder to ‘e:\winids\php’ folder.
    b. Open the CMD window and change the directory to ‘e:\winids\php’ and than type the following commands and press ‘Enter’ after each command:
        i. php go-pear.phar
        ii. pear install Image_Color-alpha
        iii. pear install Image_Canvas-alpha
        iv. pear install Image_Graph-alpha
        v. pear install Log-alpha
        vi. pear install Math_BigInteger-alpha
        vii. pear install Numbers_Roman-alpha
        viii. pear install Numbers_Words-alpha
        ix. pear install Mail-alpha
        x. pear install Mail_Mime-alpha
        xi. copye:\winids\apache24\htdocs\base\world_map6.*e:\winids\php\pear\image\graph\images\maps
35. Configuring the Barnyard2: open barnyard2.conf file in Notepad++ and make the following changes:

a. Set the path of the files to respective destination:
   i. config reference_file: e:\winids\snort\etc\reference.config
   ii. config classification_file: e:\winids\snort\etc\classification.config
   iii. config gen_file: e:\winids\snort\etc\gen-msg.map
   iv. config sid_file: e:\winids\snort\etc\sid-msg.map
b. Change config event_cache_size to 32768.
c. Output database must be set as 'log, mysql, user=snort password=l0gg3r dbname=snort host=winids sensor_name=WinIDS-Home'
d. Save the file and exit.

36. To add the Barnyard2 to auto-run on user login, execute auto-local-barnyard2.reg from the CMD window and restart the system.
37. This completes the configuration of CIDS on windows machine. Now open the browser and type 'http://winids', the following screen will be loaded:

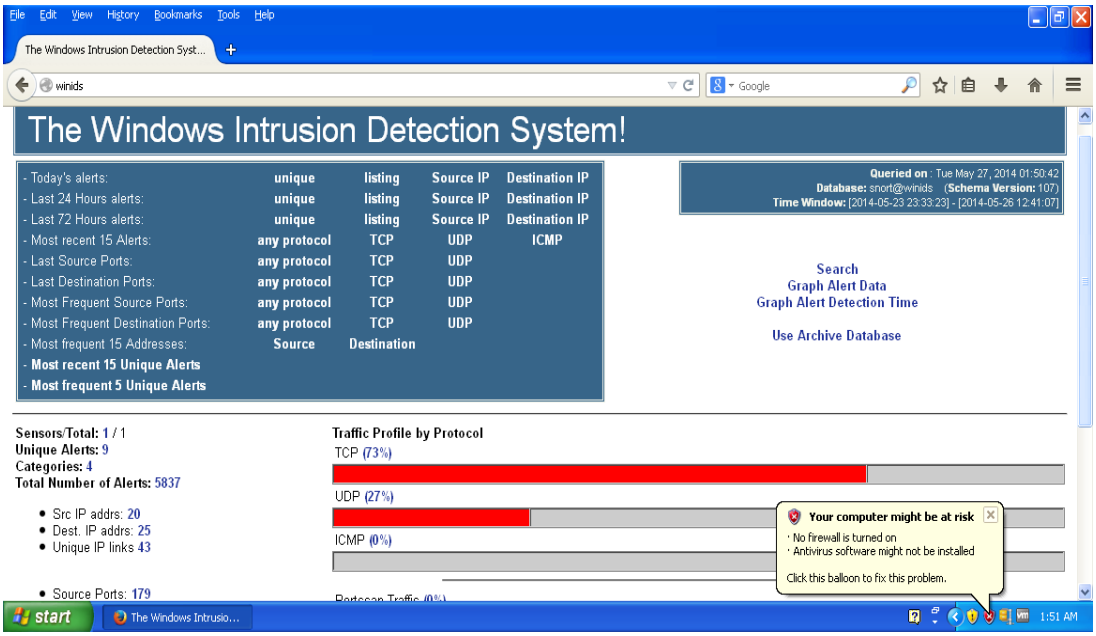
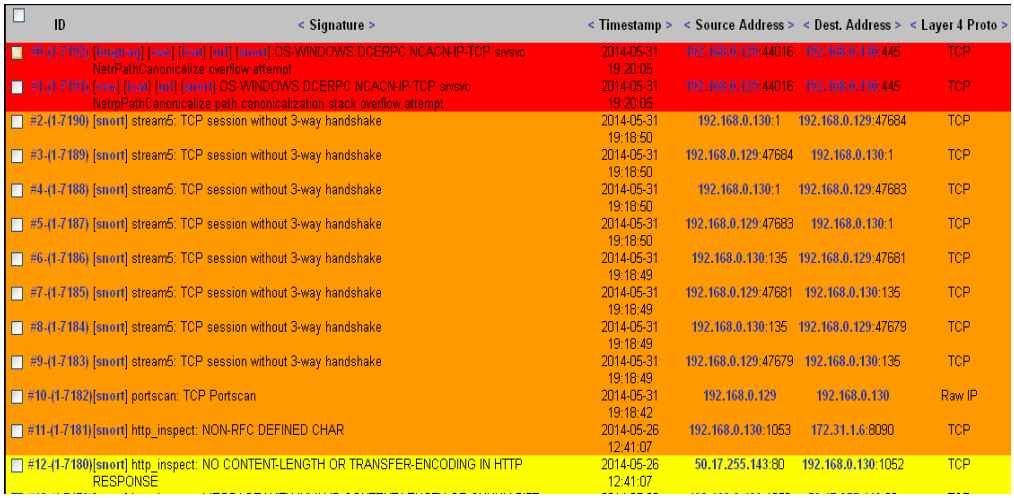


Fig.27: Main security console.

38. Now since all the process are properly running, the security administrator can check for different options that are available on the screen and then can take proper actions on the basis of data and information available to him.
39. Different options that are available are: Today's alerts, last 24 hours alerts, last 72 hours alerts, most recent 15 alerts and many more. On clicking any of the available option the administrator will have the proper information related to that particular option. Some example along with their snapshot are shown below:

a. If a security administrator clicks on 'any protocol' in front of 'Most recent 15 alerts', the following data as shown in Fig will be available to him. Similarly he can check for any particular protocol i.e. either TCP or UDP.



ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-7192) [snort] [cve] [bcat] [url] [snort]	OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrPathCanonicalize overflow attempt	2014-05-31 19:20:05	192.168.0.129:44016	192.168.0.130:445	TCP
#1-(1-7191) [cve] [bcat] [url] [snort]	OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrpPathCanonicalize path canonicalization stack overflow attempt	2014-05-31 19:20:05	192.168.0.129:44016	192.168.0.130:445	TCP
#2-(1-7190) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:50	192.168.0.130:1	192.168.0.129:47684	TCP
#3-(1-7189) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:50	192.168.0.129:47684	192.168.0.130:1	TCP
#4-(1-7188) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:50	192.168.0.130:1	192.168.0.129:47683	TCP
#5-(1-7187) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:50	192.168.0.129:47683	192.168.0.130:1	TCP
#6-(1-7186) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:49	192.168.0.130:135	192.168.0.129:47681	TCP
#7-(1-7185) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:49	192.168.0.129:47681	192.168.0.130:135	TCP
#8-(1-7184) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:49	192.168.0.130:135	192.168.0.129:47679	TCP
#9-(1-7183) [snort]	stream5: TCP session without 3-way handshake	2014-05-31 19:18:49	192.168.0.129:47679	192.168.0.130:135	TCP
#10-(1-7182)[snort]	portscan: TCP Portscan	2014-05-31 19:18:42	192.168.0.129	192.168.0.130	Raw IP
#11-(1-7181)[snort]	http_inspect: NON-RFC DEFINED CHAR	2014-05-26 12:41:07	192.168.0.130:1053	172.31.1.6:8090	TCP
#12-(1-7180)[snort]	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	2014-05-26 12:41:07	50.17.255.143:80	192.168.0.130:1052	TCP

Fig.28: Displaying 15 last attacks.

b. On clicking 'any protocol' of 'last source ports'.

< Port >	< Sensor >	< Occurrences >	< Unique Alerts >	< Src. Addr. >	< Dest. Addr. >	< First >	< Last >
44016 [sans] [tantalo] [sstats]	1	2	2	1	1	2014-05-31 19:20:05	2014-05-31 19:20:05
47683 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:50	2014-05-31 19:18:50
47684 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:50	2014-05-31 19:18:50
1 / tcp [sans] [tantalo] [sstats]	1	2	1	1	1	2014-05-31 19:18:50	2014-05-31 19:18:50
47679 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:49	2014-05-31 19:18:49
135 / tcp [sans] [tantalo] [sstats]	1	2	1	1	1	2014-05-31 19:18:49	2014-05-31 19:18:49
47681 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:49	2014-05-31 19:18:49
1052 / tcp [sans] [tantalo] [sstats]	1	29	3	1	1	2014-05-23 23:33:24	2014-05-26 12:41:07
8090 / tcp [sans] [tantalo] [sstats]	1	13	1	1	1	2014-05-24 18:09:05	2014-05-26 12:41:07
1053 / tcp [sans] [tantalo] [sstats]	1	39	3	1	2	2014-05-23 23:33:23	2014-05-26 12:41:07
80 / tcp [sans] [tantalo] [sstats]	1	1906	3	9	1	2014-05-23 23:33:23	2014-05-26 12:41:07
1050 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-26 12:41:07	2014-05-26 12:41:07
1051 / tcp [sans] [tantalo] [sstats]	1	9	3	1	2	2014-05-23 23:33:24	2014-05-26 12:41:07
1261 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-24 18:37:18	2014-05-24 18:37:18
1262 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-24 18:37:18	2014-05-24 18:37:18

Fig.29: Displaying 15 last source ports.

c. On clicking 'any protocol' of 'last destination ports'.

< Port >	< Sensor >	< Occurrences >	< Unique Alerts >	< Src. Addr. >	< Dest. Addr. >	< First >	< Last >
445 [sans] [tantalo] [sstats]	1	2	2	1	1	2014-05-31 19:20:05	2014-05-31 19:20:05
47683 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:50	2014-05-31 19:18:50
1 / tcp [sans] [tantalo] [sstats]	1	2	1	1	1	2014-05-31 19:18:50	2014-05-31 19:18:50
47684 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:50	2014-05-31 19:18:50
47679 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:49	2014-05-31 19:18:49
47681 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-31 19:18:49	2014-05-31 19:18:49
135 / tcp [sans] [tantalo] [sstats]	1	2	1	1	1	2014-05-31 19:18:49	2014-05-31 19:18:49
1052 / tcp [sans] [tantalo] [sstats]	1	32	3	1	1	2014-05-23 23:33:24	2014-05-26 12:41:07
80 / tcp [sans] [tantalo] [sstats]	1	2047	3	1	9	2014-05-23 23:33:23	2014-05-26 12:41:07
1050 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-26 12:41:07	2014-05-26 12:41:07
8090 / tcp [sans] [tantalo] [sstats]	1	15	2	1	1	2014-05-24 18:09:05	2014-05-26 12:41:07
1262 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-24 18:37:18	2014-05-24 18:37:18
1261 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-24 18:37:18	2014-05-24 18:37:18
1258 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-24 18:37:15	2014-05-24 18:37:15
1259 / tcp [sans] [tantalo] [sstats]	1	1	1	1	1	2014-05-24 18:37:15	2014-05-24 18:37:15

Fig.30: Displaying 15 last destination ports.

d. On clicking 'destination' option of the 'Most frequent 15 addresses'.

< Dst IP address >	Sensor #	< Total # >	< Unique Alerts >	< Src. Addr. >
192.168.0.130	1	2652	10	18
74.115.12.56	1	1538	2	1
192.168.0.2	1	760	4	1
50.17.255.143	1	283	3	1
192.168.0.255	1	216	4	2
207.171.189.80	1	100	2	1
74.125.236.192	1	42	2	1
74.125.236.183	1	32	2	1
103.245.222.175	1	30	2	1
74.125.236.166	1	28	2	1
63.245.217.44	1	16	2	1
172.31.1.6	1	15	2	1
224.0.0.22	1	14	1	2
74.125.236.191	1	14	2	1
23.57.219.27	1	14	2	1

Fig.31: Most frequent 15 address.

e. On clicking 'Most recent 15 unique alerts'.

< Signature >	< Classification >	< Total # >	Sensor #	< Source Address >	< Dest. Address >	< First >	< Last >
[snort] OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrPathCanonicalize overflow attempt	attempted-admin	1(0%)	1	1	1	2014-05-31 19:20:05	2014-05-31 19:20:05
[snort] OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrpPathCanonicalize path canonicalization stack overflow attempt	attempted-admin	1(0%)	1	1	1	2014-05-31 19:20:05	2014-05-31 19:20:05
[snort] stream5: TCP session without 3-way handshake	bad-unknown	8(0%)	1	2	2	2014-05-31 19:18:49	2014-05-31 19:18:50
[snort] portscan: TCP Portscan	attempted-recon	1(0%)	1	1	1	2014-05-31 19:18:42	2014-05-31 19:18:42
[snort] http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	16(0%)	1	2	1	2014-05-24 18:09:05	2014-05-26 12:41:07
[snort] http_inspect: NON-RFC DEFINED CHAR	bad-unknown	12(0%)	1	1	1	2014-05-24 18:09:05	2014-05-26 12:41:07
[snort] http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE	unknown	6(0%)	1	1	2	2014-05-24 18:09:05	2014-05-26 12:41:07
[snort] Snort Alert [1:4:1]	unknown	1494(26%)	1	4	24	2014-05-23 23:33:23	2014-05-23 23:48:40
[snort] Snort Alert [1:5:1]	misc-activity	392(7%)	1	4	5	2014-05-23	2014-05-23

Fig.32: Most recent 15 unique alerts.

f. On clicking 'Most frequent 5 unique alerts'.

< Signature >	< Classification >	< Total # >	Sensor #	< Source Address >	< Dest. Address >	< First >	< Last >
[snort] Snort Alert [1:3:1]	unknown	1816(31%)	1	19	5	2014-05-23 23:33:23	2014-05-23 23:48:39
[snort] Snort Alert [1:4:1]	unknown	1494(26%)	1	4	24	2014-05-23 23:33:23	2014-05-23 23:48:40
[snort] Snort Alert [1:2:1]	tcp-connection	1078(18%)	1	1	15	2014-05-23 23:33:23	2014-05-23 23:39:45
[snort] snort general alert	tcp-connection	621(11%)	1	14	1	2014-05-23 23:34:19	2014-05-23 23:39:45
[snort] Snort Alert [1:6:1]	misc-activity	402(7%)	1	4	8	2014-05-23 23:33:27	2014-05-23 23:48:39

Fig.33: Displaying 5 most frequent alerts.

Thus you can obtain any type of information by clicking on any of the option available.

40. Generating attack scenario on the cloud node and checking for the alerts generated by our CIDS.

    a. Start Backtrack in virtual machine; configure the network properties of the Backtrack, so that Windows XP and Backtrack are in the same network.

    b. Now click on Applications > Backtrack > Exploitation Tools > Network Exploitation Tools > Metasploit Framework > armitage.

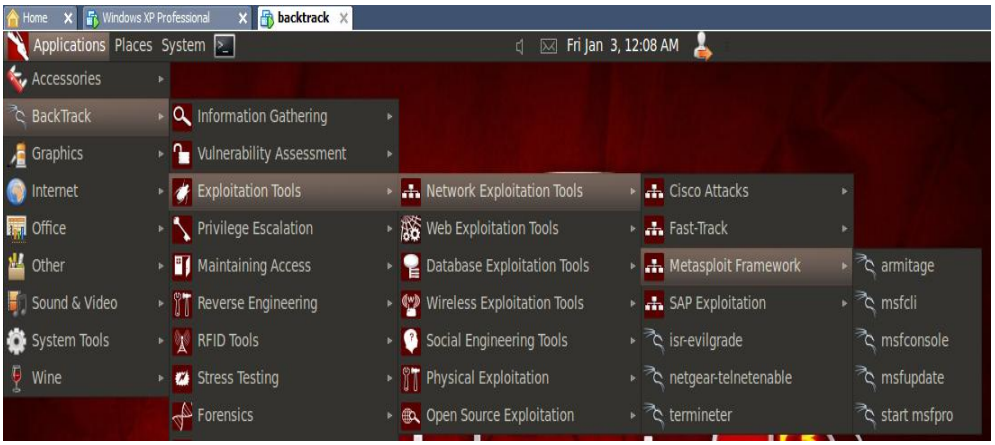


Fig.34: Path to Armitage.

c. Click on armitage option to open it. It will than launch Metasploit framework.

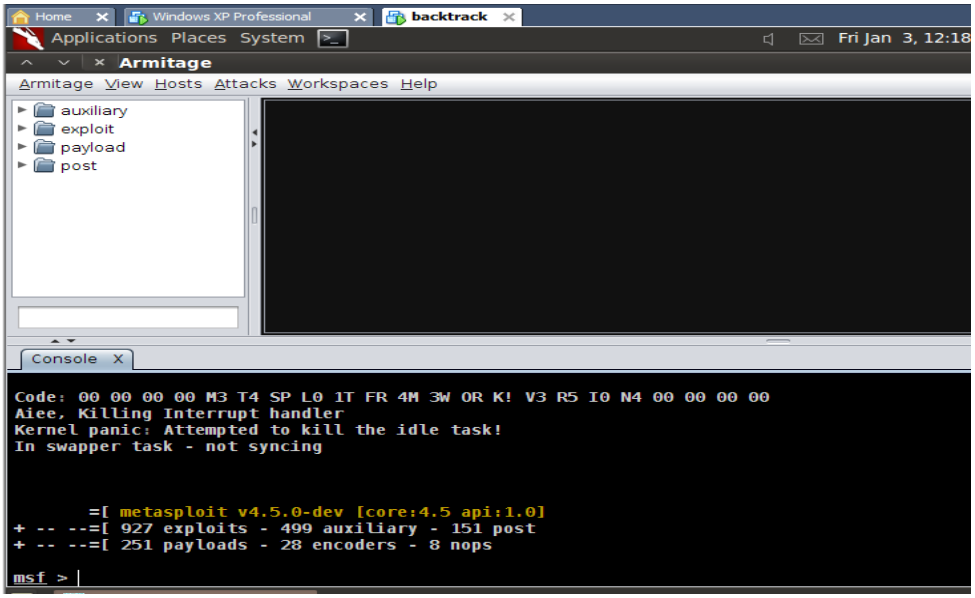


Fig.35: View of Armitage graphical interface.

d. Now go to Hosts option, and then click on Nmap scan, than on Ping scan, it will ask you to enter the network range, provide the network range.

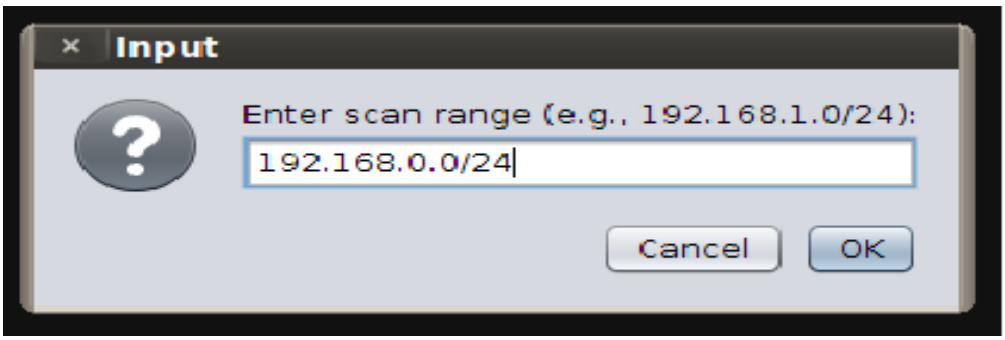


Fig.36: Providing Network Range for Ping Scan.

e. As soon u press ok, Metasploit launches its command and tells u which host are UP in the network. In our case we find that host with IP 192.168.0.128 is UP.

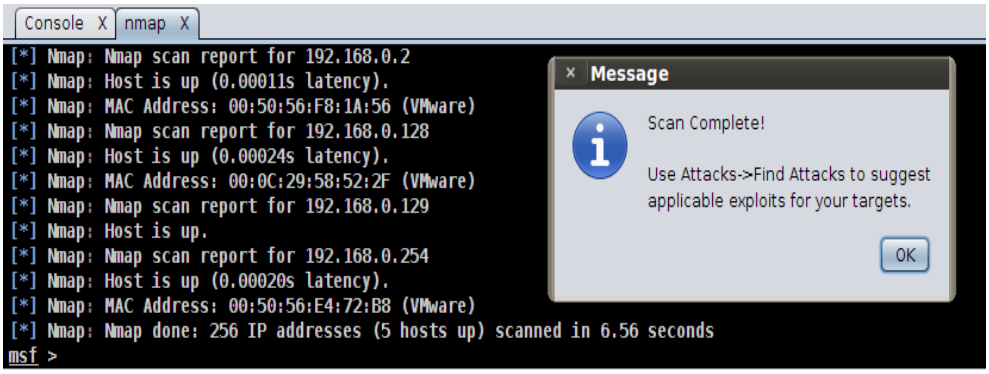


Fig.37: Result of Ping Scan.

f. Now click on Hosts than on Add Hosts (provide the IP of Windows Xp). A virtual desktop resembling as an XP machine will appear. Right click on it and then click on Scan, Armitage will tell u all the ports that are open and what services are running on those ports.

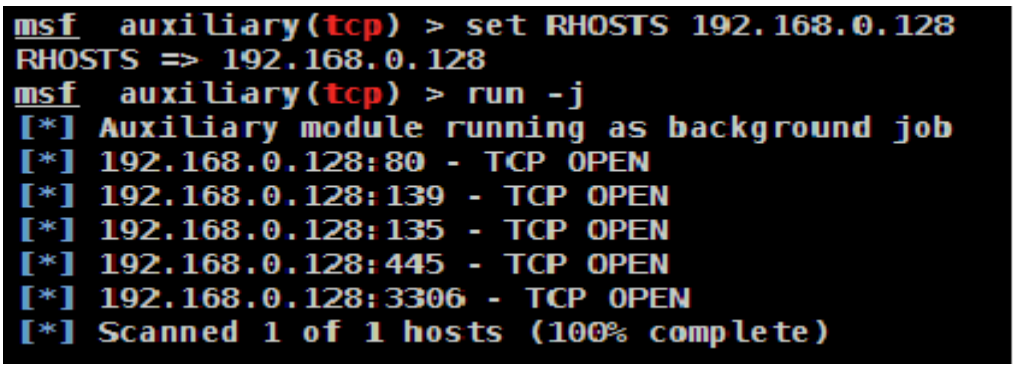


Fig.38: List of open ports on XP.

Fig.39: Services running on ports and OS running on host.

g. From the result we can see that the host is running Windows XP Service Pack 2, now we will find the vulnerabilities that can be used to exploit Windows Xp, and we found that Windows XP has a vulnerability named Microsoft Server Service Relative Path Stack Corruption. Thus we will take advantage of

MS08_067 vulnerability that uses the netapi module in the windows SMP protocol that may be used for arbitrary code execution, using this vulnerability we can obtain the shell of XP.

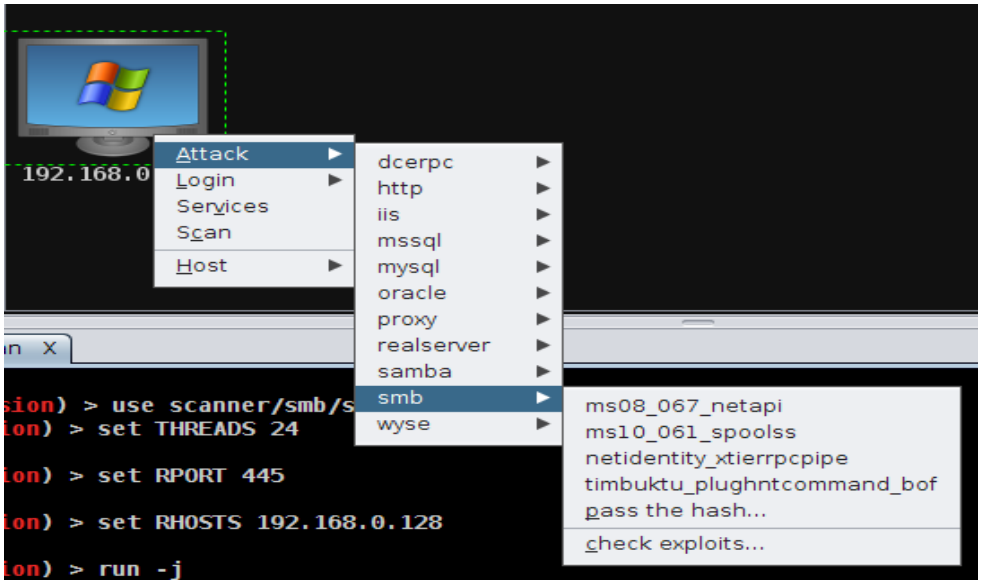


Fig.40: Exploiting netapi vulnerability.

h. As soon as you click on the ms08_067_netapi option, a screen will prompt and will ask u to check the required details, if all the details are correct than you can click on Launch button to launch the attack.

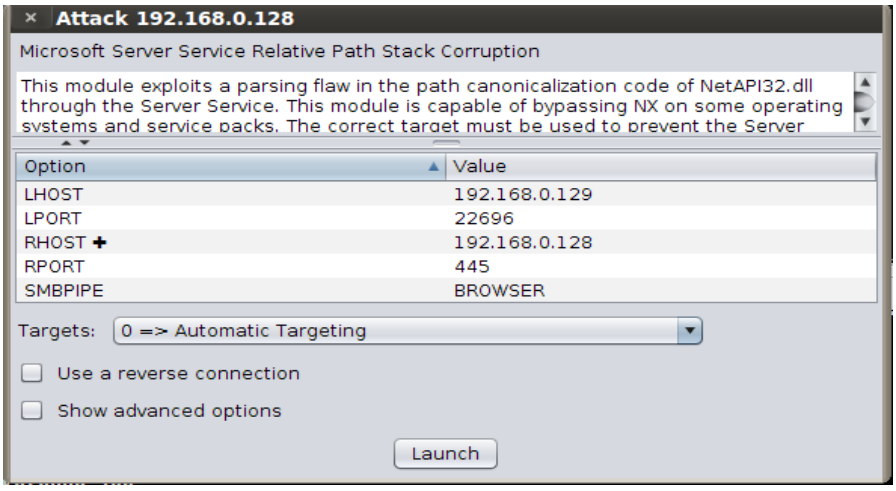


Fig.41: Launching attack.

i. Armitage will than ask the Metasploit framework to run desired script in order to exploit the vulnerability.

```
msf > use exploit/windows/smb/ms08_067_netapi
msf  exploit(ms08_067_netapi) > set LHOST 192.168.0.129
LHOST => 192.168.0.129
msf  exploit(ms08_067_netapi) > set RPORT 445
RPORT => 445
msf  exploit(ms08_067_netapi) > set LPORT 22696
LPORT => 22696
msf  exploit(ms08_067_netapi) > set RHOST 192.168.0.128
RHOST => 192.168.0.128
msf  exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf  exploit(ms08_067_netapi) > set SMBPIPE BROWSER
```

```
msf  exploit(ms08_067_netapi) > set TARGET 0
TARGET => 0
msf  exploit(ms08_067_netapi) > exploit -j
[*] Exploit running as background job.
[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.0.128
[*] Meterpreter session 1 opened (192.168.0.129:44782 -> 192.168.0.128:22696) at 2014-01-03 12:44:31 +0530
```

Fig.42: Metasploit framework exploiting the vulnerability.

j. As soon as the exploit is complete you will find the red dragons on the machine indicating that the machine with IP 192.168.0.128 has been compromised, and u will get the shell of the compromised machine.

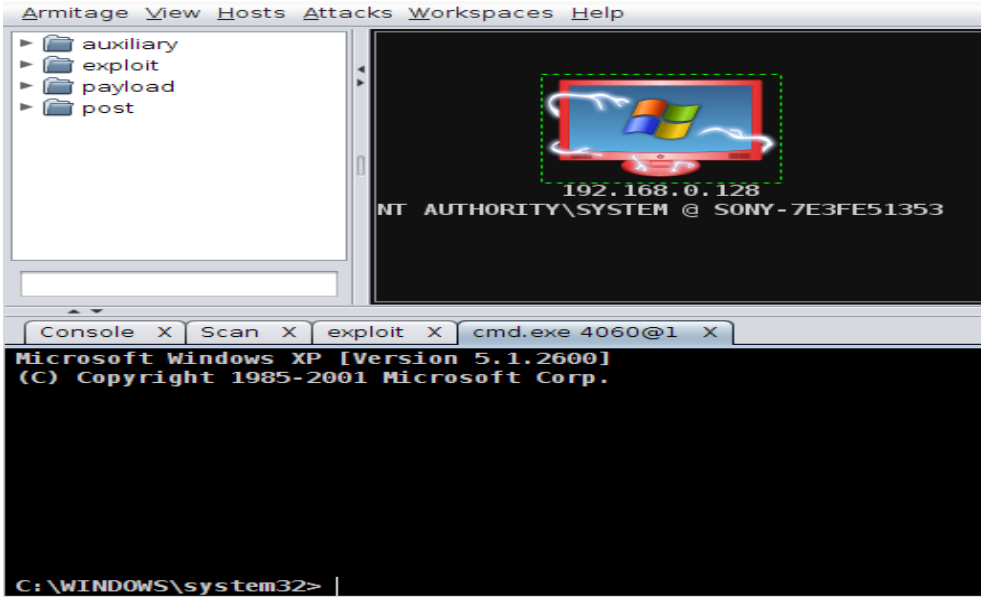
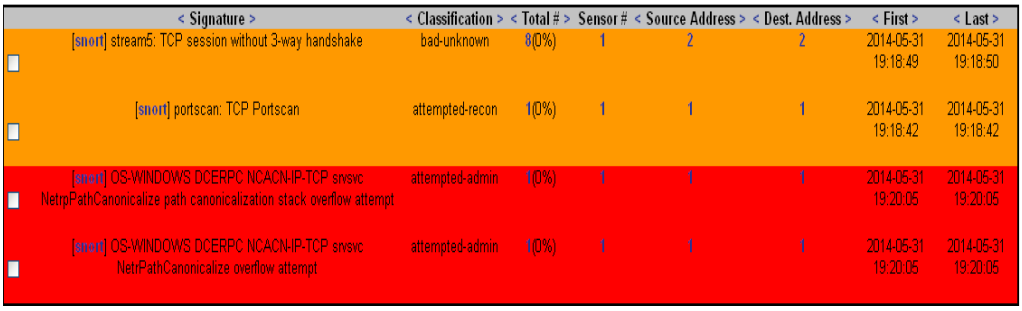


Fig.43: Victim machine get compromised indicated by red dragon.

k. Now the point to be noted is that, if we have not installed the snort in our network than the system administrator will never would have known that his/her machine has been compromised but since snort was installed and running in an NIDS mode that it will generate a log indicating that your machine has been compromised, because of the following vulnerability.



< Signature >	< Classification >	< Total # >	Sensor #	< Source Address >	< Dest. Address >	< First >	< Last >
[snort] stream5: TCP session without 3-way handshake	bad-unknown	8(0%)	1	2	2	2014-05-31 19:18:49	2014-05-31 19:18:50
[snort] portscan: TCP Portscan	attempted-recon	1(0%)	1	1	1	2014-05-31 19:18:42	2014-05-31 19:18:42
[snort] OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrpPathCanonicalize path canonicalization stack overflow attempt	attempted-admin	1(0%)	1	1	1	2014-05-31 19:20:05	2014-05-31 19:20:05
[snort] OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrPathCanonicalize overflow attempt	attempted-admin	1(0%)	1	1	1	2014-05-31 19:20:05	2014-05-31 19:20:05

Fig.44: Alerts generated with signatures.

41. **Parser and Summarizer Approach:** from the cloud administrator point of view, a proper readable and summarized alarm report is fundamental. IDS (Snort) by default produces an intensive number of alerts, because of the large scalability of the cloud network. Thus the executable code of our Parser and Summarizer algorithm when attached with the schema of 'snort' database will reduce the number of alerts for the cloud administrator. Our algorithm is based on an idea, if one machine is attacked by one or more machine using same attack signature and has a same timestamp of attack, than it is possible to reduce the number of alerts generated by the CIDS component. This can be executed by integrating all the alerts with the similar combination (destination IP, attack signature and timestamp) into a single alert only that also merges their attributes.

42. **Algorithm:**

```
while (database has more row) {
    //Fetch the current database row.
    r = currentRow();
    // Create an object of AlertId class. An instance of this class contains the destination machine IP Address and a signature ID along with the timestamp of the attack.
    AlertId id = new AlertId (r.getCol (destIp), r.getCol (signatureId), r.getCol (timestamp));
    if (hasmap.contains(id)) // If the hashmap already contains an instance of the class Alert Id
    {
        // Get the value(an instance of class AlertDesc) associated with the key (an instance of class AlertID).
        AlertDesc desc = hashmap.get (id);
        //append the IP Address to the list of source IP Address.
        desc.getSourceIp ().add (r.getCol (sourceIp));
        // append the port to the list of destination port.
        desc.getPort ().add (r.getCol (port));
```

```
}
else
{
        // Create an instance of class AlertDesc
        AlertDesc desc = new AlertDesc ();
        //set all the properties of the desc object with values from the //database;
        //add desc to the hashmap;
}
}
```

a. AlertId is a class with two properties: signatureId, timestamp and destinationId.
b. AlertDesc is a class with the following properties: signatureName, signatureClassId, and signaturePriority, a list of sourceIp, ipProtocol, sourcePort and list of destinationPort.
c. r is the current row.
d. hashmap is a map that will contain the summarised table.

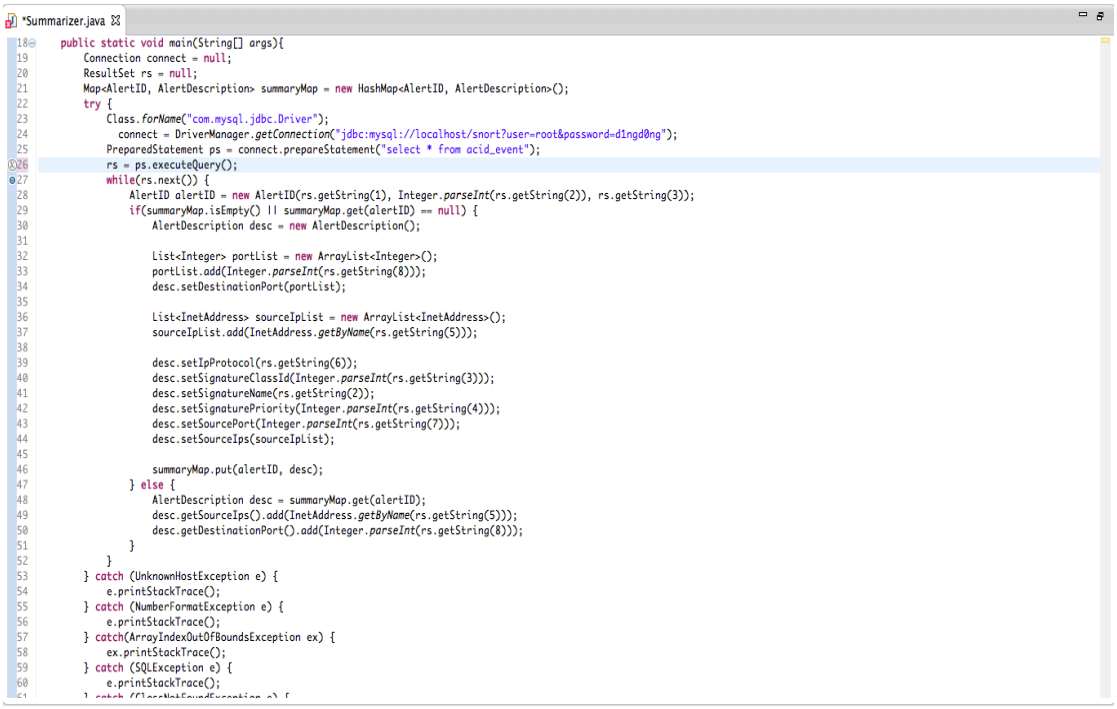
43. If we look at the number of entries in the 'acid_event' Schema Tables of 'Snort' Schema we will found that the number is very huge nearly to 5848.

signature	sig_name	sig_class_id	sig_priority	timestamp	ip_src	ip_dst
225	http_inspect: MESSAGE WITH INVALID CONTENT-LEN...	2	3	2014-05-26 12:41:07	3232235650	2887713030
230	http_inspect: NO CONTENT-LENGTH OR TRANSFER-E...	2	3	2014-05-26 12:41:07	2887713030	3232235650
253	http_inspect: NON-RFC DEFINED CHAR	3	2	2014-05-26 12:41:07	3232235650	2887713030
225	http_inspect: MESSAGE WITH INVALID CONTENT-LEN...	2	3	2014-05-26 12:41:07	3232235650	840040335
230	http_inspect: NO CONTENT-LENGTH OR TRANSFER-E...	2	3	2014-05-26 12:41:07	840040335	3232235650
253	http_inspect: NON-RFC DEFINED CHAR	3	2	2014-05-26 12:41:07	3232235650	2887713030
217	portscan: TCP Portscan	4	2	2014-05-31 19:18:42	3232235649	3232235650
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:49	3232235649	3232235650
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:49	3232235650	3232235649
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:49	3232235649	3232235650
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:49	3232235650	3232235649
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:50	3232235649	3232235650
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:50	3232235650	3232235649
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:50	3232235649	3232235650
126	stream5: TCP session without 3-way handshake	3	2	2014-05-31 19:18:50	3232235650	3232235649
506	OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrpPat...	12	1	2014-05-31 19:20:05	3232235649	3232235650
507	OS-WINDOWS DCERPC NCACN-IP-TCP srvsvc NetrPath...	12	1	2014-05-31 19:20:05	3232235649	3232235650

5848 rows fetched in 0.2879s (0.0329s) | Edit | Apply Changes | Discard Changes | First | Last | Search

Fig.45: Number of alerts before applying summarizer algorithm.

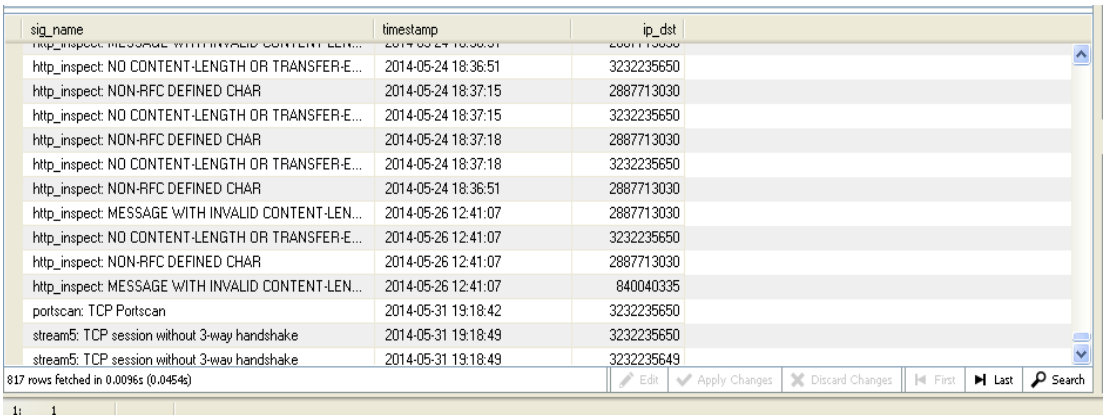
44. Write the java executable code implementing the algorithm and JDBC connectivity to 'snort' schema.



```
*Summarizer.java
18  public static void main(String[] args){
19      Connection connect = null;
20      ResultSet rs = null;
21      Map<AlertID, AlertDescription> summaryMap = new HashMap<AlertID, AlertDescription>();
22      try {
23          Class.forName("com.mysql.jdbc.Driver");
24            connect = DriverManager.getConnection("jdbc:mysql://localhost/snort?user=root&password=d1ngd0ng");
25          PreparedStatement ps = connect.prepareStatement("select * from acid_event");
26          rs = ps.executeQuery();
27          while(rs.next()) {
28              AlertID alertID = new AlertID(rs.getString(1), Integer.parseInt(rs.getString(2)), rs.getString(3));
29              if(summaryMap.isEmpty() || summaryMap.get(alertID) == null) {
30                  AlertDescription desc = new AlertDescription();
31
32                  List<Integer> portList = new ArrayList<Integer>();
33                  portList.add(Integer.parseInt(rs.getString(8)));
34                  desc.setDestinationPort(portList);
35
36                  List<InetAddress> sourceIpList = new ArrayList<InetAddress>();
37                  sourceIpList.add(InetAddress.getByName(rs.getString(5)));
38
39                  desc.setIpProtocol(rs.getString(6));
40                  desc.setSignatureClassId(Integer.parseInt(rs.getString(3)));
41                  desc.setSignatureName(rs.getString(2));
42                  desc.setSignaturePriority(Integer.parseInt(rs.getString(4)));
43                  desc.setSourcePort(Integer.parseInt(rs.getString(7)));
44                  desc.setSourceIps(sourceIpList);
45
46                  summaryMap.put(alertID, desc);
47              } else {
48                  AlertDescription desc = summaryMap.get(alertID);
49                  desc.getSourceIps().add(InetAddress.getByName(rs.getString(5)));
50                  desc.getDestinationPort().add(Integer.parseInt(rs.getString(8)));
51              }
52          }
53      } catch (UnknownHostException e) {
54          e.printStackTrace();
55      } catch (NumberFormatException e) {
56          e.printStackTrace();
57      } catch(ArrayIndexOutOfBoundsException ex) {
58          ex.printStackTrace();
59      } catch (SQLException e) {
60          e.printStackTrace();
```

Fig.46: Executable code of summarizer algorithm.

45. The code once executed parses and summarizes the alerts that can be validated by the number of alerts that are being generated after the execution of the algorithm.



sig_name	timestamp	ip_dst
http_inspect: NO CONTENT-LENGTH OR TRANSFER-E...	2014-05-24 18:36:51	3232235650
http_inspect: NON-RFC DEFINED CHAR	2014-05-24 18:37:15	2887713030
http_inspect: NO CONTENT-LENGTH OR TRANSFER-E...	2014-05-24 18:37:15	3232235650
http_inspect: NON-RFC DEFINED CHAR	2014-05-24 18:37:18	2887713030
http_inspect: NO CONTENT-LENGTH OR TRANSFER-E...	2014-05-24 18:37:18	3232235650
http_inspect: NON-RFC DEFINED CHAR	2014-05-24 18:36:51	2887713030
http_inspect: MESSAGE WITH INVALID CONTENT-LEN...	2014-05-26 12:41:07	2887713030
http_inspect: NO CONTENT-LENGTH OR TRANSFER-E...	2014-05-26 12:41:07	3232235650
http_inspect: NON-RFC DEFINED CHAR	2014-05-26 12:41:07	2887713030
http_inspect: MESSAGE WITH INVALID CONTENT-LEN...	2014-05-26 12:41:07	840040335
portscan: TCP Portscan	2014-05-31 19:18:42	3232235650
stream5: TCP session without 3-way handshake	2014-05-31 19:18:49	3232235650
stream5: TCP session without 3-way handshake	2014-05-31 19:18:49	3232235649

817 rows fetched in 0.0096s (0.0454s) | Edit | Apply Changes | Discard Changes | First | Last | Search

Fig.47: Number of alerts after the execution of summarizer algorithm.

From the above figure it is clear that the number of alerts reduces from 5848 to 817 that are reducing the alarm generation of same alert by **86%** which results in increasing the efficiency of the CIDS administrator.

46. In addition to the above implementation, you can filter the alerts according to your requirement by writing different SQL queries and executing it on the ‘acid_event’ table of ‘snort’ schema.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

The thesis presents a comprehensive literature that describes the characteristics and requirements for implementing efficient and effective IDS in cloud environment that generates relevant alarms only. The literature provides in-depth knowledge of the layered taxonomy of IDS and special attention was given to characteristics of cloud and the present challenges restricting the implementation of IDS in cloud environment. The thesis also provides the basic knowledge of cloud computing along with its various service and deployment models and its key characteristics. The work also points to the major barriers that restrict the broader adoption of cloud, crucial security concerns in cloud computing, specific risks that a user should raise to a cloud provider and the proper defence strategy that should be opted to protect the cloud environment from getting abused.

To rectify the problem presented in the thesis, Snort (an open source IDS) has been implemented on a cloud hypervisor to detect the anomalous activities. The implementation requires the installation and manual configuration of various support software that will help in the proper functioning of our IDS. A graphical console was created which helps the system administrator to view and analyse the alerts generated, the console has other security related option that can also be analysed to study and thus improving the security of cloud environment. All the events that are being captured by Snort are stored in a MySQL database which is attached to our 'summarizer and parser' algorithm which summarizes and combines the related events (depending upon various attributes/trust factors) thus reducing the number of alert triggered. After proper implementation of the algorithm on a Snort database it has been observed that **86%** of the irrelevant alerts have been deleted from generating alarm which gives extra time to the system administrator to work on relevant alarms.

5.2 Future Work

Summarizer and parser algorithm that has been implemented can be optimized further by changing or adding the trust attributes (destination IP, source IP, source port, destination port, timestamp, signature ID, etc.) thus obtaining the relevant alerts according to the requirement of the organization and eliminating the non necessary once.

Alerts can also be logged to a remote syslog server which lets the security administrator to check alerts at a remote location which can be outside the perimeter of an

organization. In addition to this, the implemented system can be made more advanced by sending the alerts through electronic mail or through short message services to the security analyst so that he can take relevant action in the real time, even if he is off from his work.

References

[1] Md. Tanzim Khorshed, A. B. M. Shawkat Ali, Saleh A. Wasimi, "A surveys on gaps, threat remediation challenge, and some thoughts for proactive attack detection in the cloud computing", Future Generation Computer Systems 28 (2012) 833-851.

[2] Dimitrios Zissis and Dimitrios Lekkas, "Addressing Cloud computing Security Issues", Future Generation Computer Systems 28 (2012) 583-592.

[3] Ahmed Patel, Mona Taghavi, Kaveh Bakhtiyari, Joaquim Celestino Junior, "An intrusion detection and prevention system in cloud computing: A systematic review", Journal of Network and Computer Applications.

[4] Whitman ME and Mattord HJ, "Principles of Information Security", ed.: Course Technology Ptr 2011:315.

[5] Herve Debar and Andreas Wespi, "Towards a taxonomy of Intrusion Detection System", Computer Networks, 31(8):805-822, April 2009, Special Issue on Computer Network Security.

[6] [Online], Available: http://www.snort.org/.

[7] Lucian E. Marin, Snort from scratch, Blog: http://www.linuxplanet.org/blogs/?cat=51

[8] Thatte G, Mitra U, Heidemann J, "Parametric methods for anomaly detection in aggregate traffic", IEEE/ACM Transactions on Networking (TON) 2011; 19: 512-525.

[9] Elshoush HT, Osman IM, "Alert correlation in collaborative intelligent intrusion detection system- a survey", Applied Soft Computing 2011; 11: 4349-4365.

[10] Pietraszek T, Tanner A, "Data mining and machine learning – towards reducing false positives in intrusion detection", Information Security Technical Report 2005; 10: 169-183.

[11] Sharma T, Sinha K, "Intrusion detection systems technology", International Journal of Engineering and Advanced Technology 2011; 1: 28-33.

[12] Patcha A, Park J-M, "An overview of anomaly detection techniques in information technology – a detailed analysis", European Journal of Scientific Research 2011; 65: 611-624.

[13] Wozniak T and Stanoevska-Slabeva, "Grid and Cloud Computing – A Business Perspective on Technology and Applications", Springer-Verlag, Berlin, Heidelberg, 2010.

[14] National Institute of Standards and Technology, "The NIST Definition of Cloud Computing", Information Technology Laboratory, 2009.

[15] Naone E, "Technology overview, conjuring clouds", MIT Technology Review, July-August, 2009.

[16] Harris D, "Why 'grid' doesn't sell", 2008.

[17] Reese G, "Cloud Application Architectures: Building infrastructure and Applications in the Cloud", in: Theory in Practice, O'Reilly Media, 2009.

[18] Rajkumar B, Yeo C, Venugopal S, Malpani S, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, 2009.

[19] NIST, 2011, NIST cloud computing program retrieved 21 May 2011, from http://www.nist.gov/itl/cloud/.

[20] Ness G, "3 Major barriers to cloud computing retrieved 22 May 2011", from http://www.infra20.com/post.cfm/3-major-barriers-to-cloud-computing .

[21] Armbrust M, Fox A, Griffith R, et al., "Above the clouds: a Berkeley view of cloud computing", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[22] Heiser J, Nicolett M, "Assessing the security risks of cloud computing", Gartner Report, 2009.

[23] Catteddu D, Hogben G, "Benefits, risks and recommendations for information security", European Network and Information Security Agency (ENISA), 2009.

[24] Brunette G, Mogull R, "Security guidance for critical areas of focus in cloud computing V2.1", http://www.cloudsecurityalliance.org/guidance/csaguide .v2.1, 1.

[25] Archer J, Boehme A, Cullinene D, Kurtz P, Reavis J, "Top threats to cloud computing, version 1.0. cloud security alliance retrieved 7 May 2011" from http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

[26] Roschke S, Cheng F and Meinel C, "Intrusion detection in the cloud", presents at the Eighth IEEE international conference on dependable, autonomic and secure computing, pp. 729-734, 2009.

[27] Arshad J, Townend P, Xu J, "A novel intrusion severity analysis approach for clouds", Future Generation Computer System, 2012.

[28] Grobauer B, Walloschek T, Stocker E, "Understanding cloud computing vulnerabilities", Security and Privacy, IEEE 2011; 9: 50-7.

[29] Dastjerdi AV, Bakar KA, and Tabatabaei SGH, ‘‘Distributed intrusion detection in clouds using mobile agents,’’ in Third International Conference on Advanced Engineering Computing and Applications in Sciences, Sliema.pp.175–180, 2009.

[30] Viega J, “Cloud computing and the common man”, Computer 2009; 42: 106-8.

[31] Wang C, Wang Q, Ren K, and Lou W, ‘‘Ensuring data storage security in cloud computing,’’ in 17th International Workshop on Quality of Service, 2009. IWQoS, Charleston, SC. pp. 1–9, 2009.

List of Publications

[1] L. Zuo, N. Kumar, H. Tu, **A. Singh**, N. Chilamkurti, S. Rho**,** "Detection and Analysis of Secure Intelligent Universal Designated Verifier Signature Scheme for Electronic Voting System**", The Journal of Supercomputing, Springer, SCIE, Impact Factor=0.91, DOI 0.1007/s11227-014-1149-2**.

[2**] A. Singh**, N. Kumar, "Design and Development of an Efficient Alert Summarization Technique for Cloud Environment to Detect Intrusions", **5th International Conference – Confluence 2014 (Theme – Cloud Security and Big Data)**, IEEE Conference # 33936, ISBN (XPLORE): 978-1-4799-4236-7, The Next Generation Information Technology Summit, Amity University, India.

[3 Invitation for the submission of the extended versions of my paper in **IEEE Transactions on Cloud Computing, special issue on "Cloud Networking"**