Distributed Intrusion Alert Aggregation with Data Stream Modeling

Rupali Ramdas Shevale Yogadhar Pandey Maheshkumar A. Sali

Abstract- Intrusion Detection System (IDS) technology is an important component in designing a secure environment. Alert aggregation is an important subtask of intrusion detection. The goal is to identify and to cluster different alerts produced by low-level intrusion detection systems, firewalls, etc. belonging to a specific attack instance which has been initiated by an attacker at a certain point in time. Thus, meta-alerts can be generated for the clusters that contain all the relevant information whereas the amount of data (i.e., alerts) can be reduced substantially. Distributed IDS systems are the next logical level for IDS systems to move to. A distributed IDS (dIDS) consists of multiple Intrusion Detection Systems (IDS) over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring, incident analysis, and instant attack data. A dIDS also allows to identify threats to the network across multiple network segments.

In Network monitoring client will receive data & filter the data contents as per queue signature or algorithm & it will generate alerts & it will transmit alerts to server end.

Key Words- Distributed Intrusion detection, Attack alert aggregation, data stream, data se0074, Monitoring, Data transmission, Alert UI.

I. INTRODUCTION

Intrusion Detection is a crucial technique in network Security, which not only detects outside intrusions but also monitors unauthorized activities inside network. However, there are some limitations on Intrusion Detection System (IDS). First, IDS is prone to producing a large number of alerts, which is difficult for experts to analyze and discover causal relationships in alert streams.

Second, false positives and false negative of IDS are inevitable. Third, IDS can only detect single attack but not multi-step attacks, to detect which network security experts need to analyze manually. Finally, it is hard to deploy IDS in large scale network. A distributed IDS (dIDS) consists of multiple Intrusion Detection Systems (IDS) over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring, incident analysis, and instant attack data. By having these co-operative agents distributed across a network, incident analysts, network operations and security personnel are able to get a broader view of what is occurring on their network as a whole. A dIDS also allows a company to efficiently manage its incident analysis resources by centralizing its attack records and by giving the analyst a quick and easy way to spot new trends and patterns and to identify threats to the network across multiple network segments. Distributed intrusion detection systems including the general setup of a dIDS and a fictional case study to demonstrate the distributed analysis abilities.

II. RELATED WORKS

Present, most IDS are quite reliable in detecting the intrusive actions caused by a single attack instance which is the occurrence of an attack of a particular type that has been launched by a specific attacker at a certain point in time and that are often spread over many network connections or log file entries, a single attack instance often results in hundreds or even thousands of alerts. IDS usually focus on detecting attack types, but not on distinguishing between different attack instances. In addition, even low rates of false alerts. As a consequence, the IDS create many alerts at a low level of abstractions. In our opinion, a "perfect" IDS should be situation-aware in the sense that at any point in time it should "know" what is going on in its environment regarding attack instances (of various types) and attackers. In this paper, we make an important step toward this goal by introducing and evaluating a new technique for alert aggregation. Alerts may originate from low-level IDS from firewalls (FW). Alerts that belong to one attack instance must be clustered together and meta-alerts must be generated for these clusters. The main goal is to reduce the amount of alerts substantially without losing any important information which is necessary to identify on-going attacks. Due to the greater view the agent allows the analyst to achieve, the dIDS offers the incident analyst many advantages over other single mode IDS systems. One of these advantages is the ability to detect attack patterns across an entire corporate network, with geographic locations separating segments by time zones or even continents. This could allow for the early detection of a well-planned and coordinated attack against the organization in question, which would allow the security people to ensure that targeted systems are secured and offending IPs are disallowed any access. Another proven advantage is to allow early detection of an Internet worm making its way through a corporate network. This information could then be used to identify and clean systems that have been infected by the worm, and prevent further spread of the worm into the network, therefore lowering any financial losses that would otherwise have been incurred.

The second major advantage is that a single analysis team can now do what previously required several incident analysis teams due to physical distance. This obviates the need to pay for distinct incident analysis teams for each separate geographic location of the organization's offices. Another issue that it addresses is attacks from within the

corporation's network by angry, upset, or bored employees. By tying the central analysis server in with the companies DHCP or RADIUS servers, the incident analysts can track down people launching attacks from within the company, and track what they have attempted to do, as well as provide evidence against the perpetrators.

A. Distributed Intrusion Detection System

Intrusion Detection System (IDS) technology is an important component in designing a secure environment of IDS. Using our agent's platform we have designed a distributed intrusion detection system. (DIDS). A specialized IDS Agent is running on the MonALISA service and in case of an alert it takes custom reactive actions (e.g. adding a blocking rule in firewall) and also broadcasts the alert in its communication group. In this way the other services can prevent possible future attacks from the same host. The attacking hosts are dynamically moved in a black-list based on the attacks level and the frequencies of them. A periodical report containing the intrusion alerts is generated and sent to the farm administrator. The dIDS system gives the analyst a quicker, easier, more efficient method to identify coordinated attacks across multiple network segments, and to trace back the activities of the attackers. The system also, ultimately, saves the corporation whose networks it is deployed on money by reducing the number of Incident Analysts needed, as well as the amount of time required to gather logs from the various IDS systems setup in a large corporate network. By having all of these attack records stored in a single place, it allows the analyst much more flexibility in discovering attack patterns, and other attack issues which may have otherwise gone unnoticed. As attackers, and attack methods become increasingly complex, the need for a dIDS system in large corporate and military networks increases drastically. With the increased complexity of these attacks, analysts are leaving themselves open to the problems of communications breakdowns, where one analyst sees a single attack on his segment, and dismisses it as nothing. While several other segments receiving the same attacks in a coordinated manner, their analysts may be dismissing the seriousness of the attack. However, when all the attack data is viewed together, a dramatically different perspective the attack may emerge.

The central analysis server is really the heart and soul of the operation. This server would ideally consist of a database and Web server. This allows the interactive querying of attack data for analysis as well as a useful Web interface to allow the corporate guys upstairs to see the current attack status of your network. It also allows analysts to perform pre-programmed queries, such as attack aggregation, statistics gathering, to identify attack patterns and to perform rudimentary incident analysis, all from a Web interface. It will monitor all transaction or data activity that occurs in the client end.

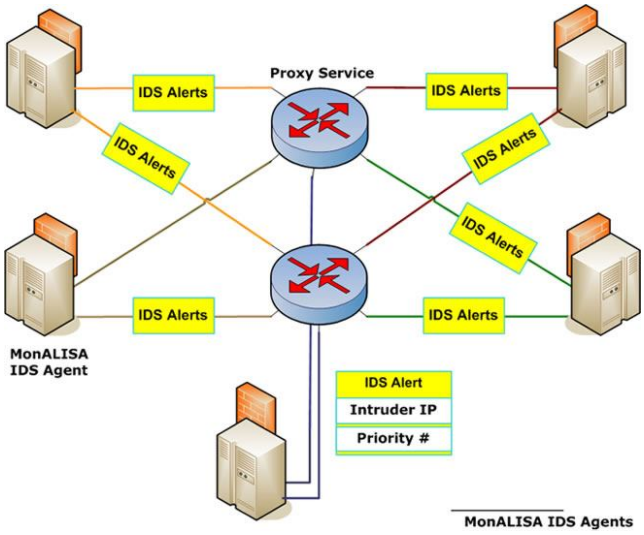

Fig.1. The Central Analysis Server

B. The Co-operative Agent Network

The co-operative agent network is one of the most important components of the dIDS. An agent is a piece of software that reports attack information to the central analysis server. The use of multiple agents across a network allows the incident analysis team a broader view of the network than can be achieved with single IDS systems. Ideally these agents will be located on separate network segments, and geographical locations (See diagram below.) The agents can also be distributed across multiple physical locations, allowing for a single incident analysis team to view attack data across multiple corporate locations.

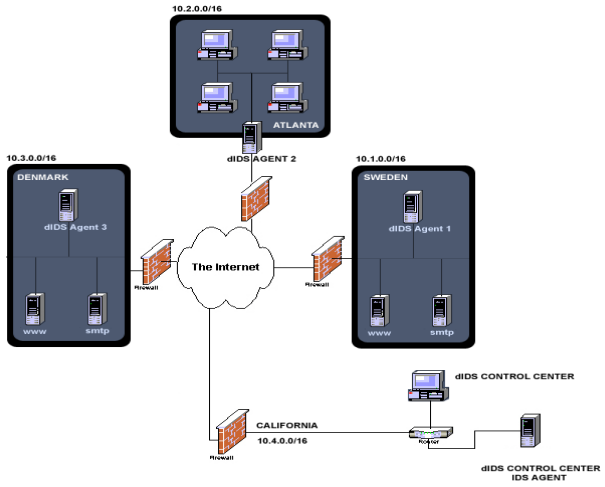

Fig. 2. A distributed intrusion detection system

The local daemon- a deamon running on the user machine that intermediates the user with the distributed agents. The search and storage distributed agents - agents loaded on

MonALISA distributed services that cooperate to accomplish user requests.

C. Distributed IDS Architecture

As networks become faster there is a need for security analysis techniques that can keep up with the increased network throughput. Traditional centralized approaches to traffic analysis cannot scale with the increase of bandwidth advances mainly due to their memory and computational requirements. In the last few years a number of distributed architectures have already been proposed for dedicated network monitoring tasks but they are not scalable in the context of high speed networks. In this paper we present an optimized scalable distributed architecture which is about 10 times quicker than the centralized architecture. The solution is based on switch-based splitting approach that supports intrusion detection on high-speed links by balancing the traffic load among different sensors running Snort. Networks began a potentially hostile environment, where intruders are passively or actively trying to breach network security. Passive intruders may browse through sensitive data files, monitor private conversations, or intercept e-mail messages. Active intruders, on the other hand, are malicious and seek to destroy information, deny others access to network resources, and introduce false data or unauthenticated messages onto the network. This type of intruder may even seek to destroy programs and applications by introducing viruses or worms into the network. The co-operative agent network is one of the most important components of the dIDS. An agent is a piece of software that reports attack information to the central analysis server. The use of multiple agents across a network allows the incident analysis team a broader view of the network than can be achieved with single. IDS systems. Ideally these agents will be located on separate network segments, and geographical locations (See diagram below.) The agents can also be distributed across multiple physical locations, allowing for a single incident analysis team to view attack data across multiple although any IDS could be used on the agent machines, it is highly suggested that SNORT be used. It has been demonstrated, however, that any attack logging system can be incorporated into this agent network. This can range from router attack logs, firewalls, and even Windows personal firewall systems. Analysis Using Aggregation is the main component used to facilitate this advanced method of analysis across a networks multiple segments. By aggregating similar or related data, the analyst is able to easily see how an attack progressed through the different stages: from active network reconnaissance, to the final attack. It is possible for the incident analyst to see what kind of time frame the attacker was working within and to correlate other attack attempts against the networks to determine if there were multiple co-operative attackers. The most common methods of aggregation are according to attacker IP, destination port, agent ID, date, time, protocol, or attack type. Aggregating by attacker IP allows the analyst to view the steps of an attacker's attempt from start to finish across the multiple network segments. Aggregating by destination port allows an analyst to view new trends in attack types, and to be able to identify new attack methods, or exploits being used. Aggregating by agent ID allows an analyst to see what variety of attacks and attackers have made attempts on the specific network segment the agent is on. Consequently, the analyst can determine if there are multiple attackers working in conjunction, or if there are network segments that are of more interest to attackers.

III. Alert Aggregation

Offline Alert Aggregation- Off-line algorithm1 for alert aggregation will be extended to a data stream algorithm for on-line aggregation one or several attackers launch several attack instances belonging to various attack types. The attack instances each cause a number of alerts with various attribute values. The task of the alert aggregation is to estimate the assignment to instances by using the unlabeled observations only and by analyzing the cluster structure in the attribute space. Reconstruct the attack situation. Meta-alerts generated are basically an abstract description of the cluster of alerts assumed to originate from one attack instance. The amount of data is reduced substantially without losing important information. Different potentially problematic situations: False alerts are not recognized as such and wrongly assigned to clusters. Acceptable as long as the number of false alerts is comparably low. True alerts are wrongly assigned to clusters (not really problematic as long as the majority of alerts belonging to that Cluster is correctly assigned, no attack instance is missed).Off-line algorithm for alert aggregation will be extended to a data stream algorithm for on-line aggregation one or several attackers launch several attack instances belonging to various attack types. The attack instances each cause a number of alerts with various attribute values. The task of the alert aggregation is to estimate the assignment to instances by using the unlabeled observations only and by analyzing the cluster structure in the attribute space. Reconstruct the attack situation. Meta-alerts generated are basically an abstract description of the cluster of alerts assumed to originate from one attack instance. The amount of data is reduced substantially without losing important information. Different potentially problematic situations: False alerts are not recognized as such and wrongly assigned to clusters. Acceptable as long as the number of false alerts is comparably low. True alerts are wrongly assigned to clusters (not really problematic as long as the majority of alerts belonging to that cluster is correctly assigned, no attack instance is missed).

Online Alert aggregation data stream modeling Algorithm- On line intrusion attack - A "perfect" IDS should be situation aware in the sense that at any point in time it should "know" what is going on in its environment regarding attack instances (of various types) and attackers. In this article we make an important step towards this goal

by introducing and evaluating a new technique for alert aggregation. Alerts may originate from low-level IDS such as those mentioned above, from firewalls, etc.

Algorithm 2: ON-LINE ALERT AGGREGATION (DATA STREAM MODELING)

```
   // initialize alert buffer
1  B := ∅
2  while new alert a is received do
3      if C = ∅ then
           // create first component
4          C_1 := {a}
5          C := {C_1}
6          initialize parameters μ_1, σ_1^2, and ρ_1.
7      else
8          C' := C
           // E step: assign alert to most
              likely component
           j* := argmax_{j∈{1,...,|C|}} H(a|μ_j, σ_j^2, ρ_j)
9          C_{j*} := C_{j*} ∪ {a}
10         // M step: update component
              parameters
11         N_{j*} := |C_{j*}|
12         for all attributes d ∈ {1,...,D_m} do
13             ρ_{j*_d} := (1/N_{j*}) · Σ_{a∈C_{j*}} a_d
14         for all attributes d ∈ {D_{m+1},...,D} do
15             μ_{j*_d} := (1/N_{j*}) · Σ_{a∈C_{j*}} a_d
               σ^2_{j*_d} := (1/N_{j*}) · Σ_{a∈C_{j*}} (a_d − μ_{j*_d})^2
16         // discard changes if degradation is
              too large (cf. Eq. 13)
17         if Ω(C)/Ω(C') < θ then
18             C := C'
19             B := B ∪ {a}
           // initiate novelty handling with Eq.
              14; call algorithm 3
20         if novelty(a) then
21             C := ALG3(C, j*, B)
22             B := ∅
           // initiate obsoleteness handling
              with Eq. 15
23         for j ∈ {1,...,|C|} do
24             if obsoleteness(C_j) then
25                 C := C\C_j
       // now we have a model of the current
          attack situation
```

Meta Alert Intrusion Detection Architecture: The metal alert intrusion detection architecture comprises of the following phases collaborating Intrusion Detection Agents, alert Alerts that belong to one attack instance must be clustered together and meta-alerts must be generated for these clusters. The main goal is to reduce the amount of alerts substantially without losing any important information which is necessary to identify ongoing attack instances. We want to have no missing meta-alerts, but in turn we accept false or redundant meta-alerts to a certain degree. This problem is not new, but current solutions are typically based on a quite simple sorting of alerts. e.g., according to their source, destination, and attack type. Under real conditions such as the presence of classification errors of the low-level IDS (e.g., false alerts), uncertainty with respect to the source of the attack due to spoofed IP addresses, or wrongly adjusted time windows, for instance, such an approach fails quite often.

IV. Collaborating Intrusion Detection Agents

Intrusion Detection agents through self-organized collaboration form a distributed intrusion detection system. The sensor layer provides the interface to the network and the host on which the agent resides. Sensors acquire raw data from both the network and the host, Filter incoming data, and extract Interesting and potentially valuable (e.g., statistical) information needed to construct an appropriate Event. At the detection layer, different detectors assess the attack events and search for known attack Signatures (misuse detection) and suspicious behavior (anomaly detection).In case of attack suspicion, they create alerts and forwarded to the alert processing layer. Alerts may also be produced by firewalls (FW). At the alert processing layer, the alert aggregation combine alerts assumed to belong to a specific attack instance. Meta alerts are generated.

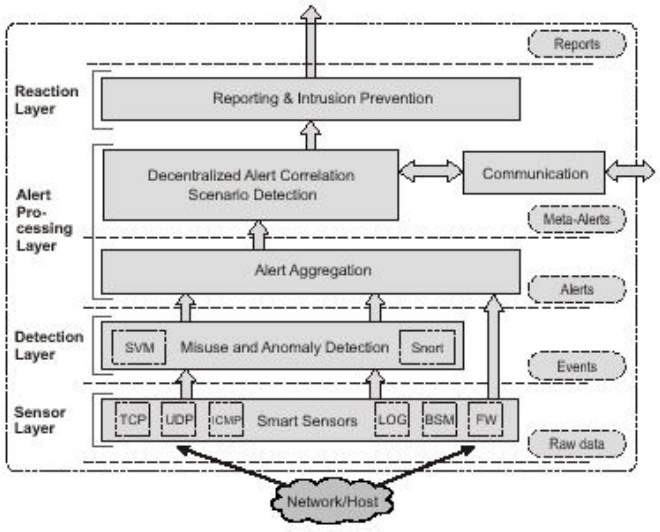

Fig. 3. Architecture of an Intrusion Detection Agent

A completely different clustering approach is presented; the reconstruction error of an auto associator neural network (AA-NN) is used to distinguish different types of alerts. Alerts that yield the same (or a similar) reconstruction error

are put into the same cluster. The approach can be applied online, but an offline training phase and training data are needed to train the AA-NN and also to manually adjust intervals for the reconstruction error that determine which alerts are clustered together. In addition, it turned out that due to the dimensionality reduction by the AA-NN, alerts of different types can have the same reconstruction error which leads to erroneous clustering. We applied the well-known c-means clustering algorithm in order to identify attack instances However, this algorithm also works in a purely offline manner. We describe our new alert aggregation approach which is at each point in time based on a probabilistic model of the current situation. To outline the preconditions and objectives of alert aggregation, we use intrusion framework. Then, we briefly describe the generation of alerts and the alert format. We continue with a new clustering algorithm for offline alert aggregation which is basically a parameter estimation technique for the probabilistic model. After that, we extend this offline method to an algorithm for data stream clustering can be applied to online alert aggregation.

Performance Measure of Intrusion Alert Aggregation

Percentage of Detected Instances (*p*)

An attack instance is being detected if there is at least one meta-alert that predominantly contains alerts of that particular instance. The percentage of detected attack instances *p* can thus be determined by dividing the number of instances that are detected by the total number of instances in the data set. The measure is computed with respect to the instances covered by the output of the detection layer, i.e., instances missed by the detectors are not considered. Data Stream Intrusion Alert Aggregation for Distributed Heterogeneous Sources 380.

Number of Meta-Alerts (*MA*) and Reduction Rate (*r*) The number of meta-alerts (*MA*) is further divided into the the number of attacks. Predominantly it contain true alerts and the number of non-attack meta-alerts *MA*non-attack which predominantly contain false alerts. The reduction rate *r* is 1 minus the number of created meta-alerts *MA* divided by the total number of alerts *N*.

Average Run-Time (*T*avg) and Worst Case Run-Time (*T*worst)The average run-time is measured in milliseconds per alert. Assuming up to several hundred thousand alerts a day, *t*avg should stay clearly below 100 ms per alert. The worst case run-time *t*worse, which is measured in seconds, states how long it takes at most to execute the *while* loop.

Meta-Alert Creation Delay (*d*)

It is obvious that there is a certain delay until a meta-alert is created for a new attack instance. The meta-alert creation delay *d* measures the delay between the actual beginning of the instance (i.e., the creation time of the first alert) and the creation of the first meta-alert for that instance. In the following, the results for the alert aggregation are presented. For all experiments, the same parameter settings are used. We set the threshold θ that decides whether to add a new alert to an existing component or not to 5%, and the value for the threshold γ that specifies the allowed temporal spread of the alert buffer to 180 seconds. Θ was set that low value in order to ensure that even a quite small degrades of the cluster quality, which could indicate a new attack instance, results in a new component.

There were two parts to the 1998 DARPA Intrusion Detection Evaluation: an off-line evaluation and a real-time evaluation. Intrusion detection systems were tested in the off-line evaluation using network traffic and audit logs collected on a simulation network. The systems processed this data in batch mode and attempted to identify attack sessions in the midst of normal activities. Intrusion detection systems were delivered to AFRL for the real-time evaluation. These systems were inserted into the AFRL network test bed and attempted to identify attack sessions in the midst of normal activities, in real-time. Intrusion detection systems were tested as part of the off-line evaluation, the real-time evaluation or both. Detection and probability of false alarm for each system under test. These evaluations contributed significantly to the intrusion detection research field by providing direction for research efforts and an objective calibration of the technical state of the art. They are of interest to all researchers working on the general problem of workstation and network intrusion detection. The evaluation was designed to be simple, to focus on core technology issues, and to encourage the widest possible participation by eliminating security and privacy concerns, and by providing data types that were used commonly by the majority of intrusion detection systems.

ACKNOWLEDGMENT

This work was partly supported by the Authors and they have sincerely thanks to improve the quality of the Paper.

REFERENCES

[1] S. Axelsson, *"Intrusion Detection Systems: A Survey and Taxonomy,"* Technical Report 99-15, Dept. of Computer Eng., Chalmers Univ. of Technology, 2000.

[2] M.R. Endsley, "Theoretical and erpinnings of Situation Awareness:A Critical Review," Situation Awareness Analysis and Measurement, M.R. Endsley and D.J. Garland, eds., chapter 1,pp. 3-32, Lawrence Erlbaum Assoc., 2000.

[3] Online Intrusion Alert Aggregation with Generative Data Stream Modeling Alexander Hofmann, Member, IEEE, and Bernhard Sick, Member, IEEE. [4] M.R. Henzinger, P. Raghavan, and S. Rajagopalan, Computing on Data Streams. Am. Math. Soc., 1999.

[4] A. Allen, "Intrusion Detection Systems: Perspective," Technical Report DPRO-95367, Gartner, Inc., 2003.Sourcefire, Inc., http://www.snort.org/, 2009.

[5] CISCO Systems, Inc., "Cisco PIX Firewall System Log Messages, Version6.3,"http://www.cisco.com/en/US/docs/security/pix pix63/system/message/pixemsgs.html, 2009.

[6] Organic Computing, R.P. Wu¨ rtz, ed. Springer, 2008.S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Chalmers University of Technology, Department of Computer Engineering, Tech. Rep. 99-15, 2000.

[7] M. R. Endsley, "Theoretical underpinnings of situation awareness: A critical review," in Situation Awareness Analysis and Measurement, M. R.

Endsley and D. J. Garland, Eds. Mahwah, NJ: Lawrence Erlbaum Associates, 2000, ch. 1, pp. 3–32.
[8] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY: Springer, 2006.
[9] M. R. Henzinger, P. Raghavan, and S. Rajagopalan, Computing on data streams. Boston, MA:American Mathematical Society, 1999.
[10] A. Allen, "Intrusion detection systems: Perspective," Gartner Inc., London, UK, Tech. Rep. DPRO-95367, 2003.
[11] F. Valeur, G. Vigna, C. Kr¨ ugel, and R. A. Kemmerer, "A comprehensive approach to intrusion detection alert correlation," in IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 3, 2004, pp. 146–169.
12] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in Recent Advances in Intrusion Detection, ser. LNCS, W. Lee, L. Me, and A. Wespi, Eds., vol. 2212. Berlin, Germany: Springer, 2001, pp. 85–103.
[13]C.M. Bishop, Pattern Recognition and Machine Learning. Springer,2006.

AUTHOR'S PROFILE

Rupali Ramdas Shevale
M.TECH, Computer Science and Engineering
Sagar Institute of Research and Technology, Bhopal (India)
E-mail: rupalishewale2003@yahoo.co.in

Passport Size Latest Color Photo

Yogadhar Pandey
M.TECH, Computer Science and Engineering
Sagar Institute of Research and Technology, Bhopal (India)
E-mail: p_yogadhar@yahoo.co.in

Passport Size Latest Color Photo

Maheshkumar A. Sali
M.TECH, Computer Science and Engineering
Sagar Institute of Research and Technology, Bhopal (India)
E-mail: Maheshsali@gmail.com