Chapter 24
Investigation of Security Issues in Distributed System Monitoring

Manjunath Kotari and Niranjan N. Chiplunkar

Abstract The distributed systems have a noteworthy role in today's information technology whether it is governmental or nongovernmental organization. Adaptive distributed systems (ADS) are distributed systems that can evolve their behaviors based on changes in their environments (Schlichting and Hiltunen, Designing and implementing adaptive distributed systems, 1998, http://www.cs.arizona.edu/adaptiveds/overview.html). For example, a constant monitoring is required in distributed system to dynamically balance the load using centralized approach (Sarma and Dasgupta, Int J Adv Res Ideas Innov Technol 2:5–10, 2014). A monitoring system or tool is used to identify the changes in the distributed systems and all the activities of the entire network systems. The monitoring of network may help to improve the efficiency of the overall network. However, the monitoring system may be compromised by the intruder by gathering the information from the distributed systems. The various secure and insecure monitoring mechanisms have been adopted by adaptive distributed systems. Most of the distributed systems nowadays use monitoring tools to monitor the various parameters of the networking system. The monitoring tool has been implemented to assess the performance overhead during monitoring. The Wireshark monitoring tool and JMonitor tool (Penteado and Trevelin, JMonitor: a monitoring tool for distributed systems. In Proceedings of international conference on systems, man, and cybernetics, COEX, Seoul, Korea, pp 1767–1772, 2012) have been used to monitor the communication between the various users and also to monitor the computational resources used in networked computers. The main concern of this chapter is to investigate the existing monitoring tools for finding the impacts of monitoring activities in the distributed network. The investigations result that, when the monitoring tool collects security-critical information, there is a high risk of information disclosure to unauthorized users. The second concern is that a secure communication channel

M. Kotari (✉)
AIET, Moodbidri, India

N. N. Chiplunkar
NMAMIT, Nitte, India
e-mail: nchiplunkar@nitte.edu.in

B. B. Gupta et al. (eds.), *Handbook of Computer Networks and Cyber Security*,
https://doi.org/10.1007/978-3-030-22277-2_24

can be implemented by using the Rivest, Shamir, and Adelman (RSA) algorithm to monitor the confidential information. This chapter illustrates the implementation and experimental results related to authors' research work and formulation of framework for security mechanisms in the context of adaptive distributed systems (Kotari et al., IOSR J Comput Eng 18:25–36, 2016).

Security issues for existing monitoring tool are investigated in detail here. In this connection, the chapter deals with the several security-related network scenarios experienced during monitoring with the help of Wireshark monitoring tool. The proper use of Wireshark monitoring tool helps to identify the possible security threats such as emerging threats of hackers, corporate data theft, and identifying threats due to viruses. The implementation of secure communication channel is discussed, which minimizes the above set of threats.

Keywords Distributed systems · Monitoring tool · Network management

1 Introduction

The monitoring may be exercised with the help of different monitoring tools available. These monitoring tools are indispensable for monitoring the entire network. In this chapter, monitoring security mechanism has been accomplished with the existing monitoring tool and various parameters over the distributed systems are studied. The foremost target of this chapter is to investigate the existing monitoring systems such as Wireshark monitoring tool and to develop a solution to secure communication channel to Wireshark monitoring tool [1] with improved security features. Thus, Wireshark monitoring tool is considered here and its advantages and disadvantages are discussed in detail.

1.1 Monitoring Systems

A system primarily used to keep track of other systems in a network is known as monitoring systems. The network monitoring expresses the usage of the system that continuously observes the networked computer in case of slow or failure and gives an alert message to network administrator [2]. The monitoring system usually comprises the following components such as monitoring module, event identifier, and monitoring hardware, detection of events, and processing of events. Specifically, monitoring is accomplished in two operations, namely detection of events that are relevant to program execution and storing or recording the collected data. The failure nodes [3] have been tracked by running monitoring application on network. Simple network management protocol (SNMP) agent accesses a monitor of another network to obtain current system status.

Monitoring system can be categorized into two types, namely internal monitoring system and external monitoring system [4]. In case of an internal monitoring system,

the monitoring system is integrated with each client. Each client's monitoring system keeps track of program executions of the same system and records the data here. The recorded information is transmitted to central monitoring system. Whereas in case of an external monitoring system, the monitoring system resides outside the existing network and monitors the target system. Here the target system is usually present in the distributed system. The monitoring system helps security administrators to keep track of the attacker in the system. So the assessment technique allows monitoring the current attacker position and forecast his (her) path in the network [5].

In Software Defined Security Architecture (SDSA) monitoring system [6], the run-time environment associated with the security execution control module contains the security engine for supporting dynamic monitoring of systems and a set of (sub) modules for supporting security status monitoring, security task executions, and software developer managements, which are all on the top of the security engine. The responsibility of the security status monitoring module is to watch and control the running status of various security programs such as the status of processes, the status of memory stacks, the status of the file system, and the status of resources scheduling.

1.1.1 Monitoring Tools

The network monitoring tool helps to monitor the utilization of traffic, bandwidth availability and utilization, latency, central processing unit (CPU) utilization, responsiveness of CPU, and fault identification. These monitored parameters help the network monitoring tool to estimate the performance of distributed systems easily. Normally, network monitoring tool is a combination of software and hardware products. It completely tracks the network activities and raises an alert if required.

In energy manufacturing plant [7], it is very difficult to decide about how much capacity is required to get desired output. There is a chance of capacity review and capacity measurement techniques, but it should be associated with new machine with many functions. One of the solutions to measure capacity is use of multiple sensors. This solution has been used to quantify variations in power input throughout monitoring. It determines the amount of power input reduction during manufacturing. A monitoring tool has been used to identify and validate energy use reduction system.

1.1.2 Purpose of Network Monitoring Tools

The network administrators use monitoring tools for various purposes [8], viz.

- To detect the faults in routers as well as in the switches.
- To supervise the internet services as well as resources.
- To supervise the operations of several host in the network.

- To supervise the natural processes of the client server computing system.
- To monitor and supervise the operation of broadcast systems for achieving scalability.
- To monitor the bandwidth level like the usage of the distributed network.
- To monitor the exchange of messages among the users of the distributed network.

1.1.3 Features of Network Monitoring Tools

The network monitoring tool acts as eyes and ears of an organization to solve the problems. These tools monitor the server or system crash, running application, utilization of bandwidth, and utilization of CPU as well as memory. Monitoring tool has been included with following features, viz. automatic discovery, inventorying devices, warnings, and web-based interface [9]. The monitoring tool [7] should be able to diagnose both IPv4 and IPv6 traffics of traditional network. Most of the monitoring tools use sensors to collect information for the purpose of analysis. Some monitoring tools use agents to collect information, but their use affects the overall performance of the system. Hence, an agent-less monitoring tool may be considered as a little efficient product. One more important feature of the monitoring tool is that monitoring tools are able to monitor all applications and services, which run across the network. They enable the network administrators to analyze performance issues from either the network or application itself. This feature of monitoring tool lets network administrators to track response time of application such as processing of server request and response of networks. The following list provides the various features of the monitoring tools.

- *Auto discovery:* It is very awkward for system administrators to insert each healthy device manually. Most of the monitoring tools perform auto discovery of system. It helps system administrators to get a glance of IT infrastructure catalogue.
- *Network traffic status:* Apart from monitoring CPU, memory and disk utilization, monitoring tools may be used to monitor network bandwidth usage. It helps the network administrators to get an insight into the ISP's bandwidth utilization.
- *Log monitoring:* Monitoring tools manage the activity logs created by operating systems. It verifies the file size of activity log and performs configuration actions. This feature helps the system administrator to control the entire network infrastructure.
- *Alert management:* The monitoring tool provides alert signal alongside mere network monitoring activity. For example, if a firewall system drops more number of packets or if the CPU of a server crosses the 95% of utilization. In all these cases, the tool should generate alert message to network administrator.
- *Customizable Web dashboard:* The monitoring tool must be customizable in such ways that user can decide about what should be on the dashboard. It helps to decide how much manpower should be utilized to address monitoring activities.
- *Security monitoring:* Monitoring tool should be secure during monitoring. The possible attack on data link layer, network layer, and application layer should be

encountered with this tool. The network administrators must decide what needs to be monitored, rather than monitoring all parameters.

1.1.4 How Network Monitoring Works

For observing the issues of network connectivity, ping utility is sufficient for simple network. The Microsoft network monitoring provides analysis of network packets mainly to resolve network issues. Normally open-source network monitoring tool provides accuracy of data based on metrics, but these require additional utilities like automatic alert signals. The open-source monitoring tools are inexpensive. However, these monitoring tools are little inefficient.

The Wireshark monitoring tool captures traffic data in real-time environment [10]. It captures one packet for every 10 msec of one time slot. So, a very large number of packets have been stored in 1 h time slot. For every such capture, the Wireshark monitoring tool saves the elapsed time, capture number, protocol used for capturing, size of the packet, source IP address, and destination IP addresses for future reference purpose.

In [11] the authors presented a solution for limiting the security breaches during monitoring service. An enough security protection against nasty achievement of the monitoring has been provided through the SELinux OS. The security policy has been applied on the basis of new configuration of OS.

Basically there are two ways of capturing packet in networks [12], one is Macro Capturing, which deals with capturing of large amount of information and its analysis. Another type is Micro Capturing, which deals only with specific information as mentioned by a user. Here, in the Wireshark monitoring tool, the Micro-Capturing method is used. The live network data can be captured and examined in this tool.

1.1.5 Passive Monitoring Framework

Jeswani et al. [13] proposed a manual as well as automatic based approach for system monitoring. The authors present frameworks with probes for adjusting monitor levels. A typical requirement for any monitoring systems is a good quality of monitoring data. This monitoring framework is deployed to capture the activities of communication, computation, and storage components. Monitoring metrics needs to be calculated for different collected parameters of nodes. Many of the monitoring tools are able to collect large amounts of metrics as per needs.

The traditional approach of monitoring data leads to collect small amounts of data and the modern approach of monitoring leads to a huge amount of information [14, 15]. The second approach is complex and difficult to store. In addition, the analysis of large volumes of information is a difficult task. The authors are focused on how to merge the two methods to develop proficient solutions for balancing the two techniques.

1.1.6 Customized Process Monitoring Tool

Comuzzi and Martinez [16] developed a customized tool for process monitoring. The users can specify customization options through Customization Interface (CI), which is a part of Monitoring Customization Console (MCC). The users usually get access of customized monitoring data through Monitoring Console (MC). The architectural framework of the tool contains the following components, viz. Query Engine (QE), Management Engine (ME), and Notification Engine (NE). The QE helps to capture monitoring parameters as per the user-desired option. The ME manages the monitoring data as per the user's logic. The NE gives notifications to users about monitoring results.

The network management framework [17] for distributed systems is modular and extensible. The modules have been built and customized according to needs of users. The framework is based on SNMP and is meant for monitoring. Framework has been used to manage distributed systems by the help of multiple points' entry.

1.1.7 Secure Monitoring Framework for Distributed System

Chen et al. [18] present a monitoring framework for distributed information management systems in a unique way. This monitoring framework makes use of Web services and message queue techniques to collect log data. The collected information is used for business process monitoring. This tool is implemented to assess the performance overhead due to monitoring. Evaluation was conducted with and without monitoring under various loads in users. The investigation reports show that the monitoring mechanism does not change the performance of the system significantly. However, the monitoring framework is a kind of passive monitoring because it uses only one message queue. This tool is unable to visualize two unrelated events that appear during concurrent process. Multiple queues may be applied, but there is a chance of monitoring overhead.

Fonseca et al. [19] proposed a framework for gathering events of certain profiles of social network users. It also periodically collects profile-related activities and profile-based information. The framework helps to compute the average events of certain profiles and compares the collected values with latest collected profile values. These values will help to detect the variation in profile activities. In addition, these variations indicate that illegal profile usage or account has been hijacked. The framework also notifies the users about cancellation friendships of each other. It also detects the abnormal activities like user added or removed many other users.

The authors presented a framework with two components. One of the components is called core component, which calculates the user profile interactions. Another component is called web interface component, which is responsible for users profile metrics by visualizing and interacting. The start component is responsible for getting the occurrence of activities from various social networks. Once the start component finishes its data collection, the statistics component analyzes the collected information and generates a report. Based on the report

generated by statistics module, the alert module processes the alert signals for appropriate suspicious activities. However, the proposed framework does not collect user-desired data from the social networks. Instead, the framework collects the bulk of data than it is necessary, in turn; it degrades the performance of the system.

Atighetchi and Adler [20] described new framework for remote monitoring. It supports largely security aspects by limiting unwanted users to access the monitoring elements. The framework also helps to cooperate in very congested distributed networks like temporary and low bandwidth networks. This remote monitoring resists spontaneous and passive attacks by the help of special sensor information.

In [21] a framework has been generated for specifying plans of monitoring parameters. Initially, the framework elaborates the process for stipulating plans and implementing them in both friendly and unfriendly environments. Then, it develops a model for execution of security policies.

1.1.8 Network Security Management

Agbogun and Ejiga [22] mentioned that different classes of intrusions, network tools, and procedures attackers are employed in networks hijack. The author presents how attacks prevented, minimized from the backend system. The following are the different steps used by intruders during exploitation of networks.

In scanning vulnerable systems, the intruders try to access network connection as well as scans IP address of a network. Once hackers are able to access IP address, then the hackers load Trojan viruses into the network. In addition, intruders use tools to identify vulnerabilities in the network and perform snooping or destroying operating system. The hacker tries to search the administrative passwords in computer system and thereby gain the access of the entire system.

The Access attack could be an external attacker. It uses different techniques to gain control of the network. An access attack may be classified into the following categories, namely Gaining Primary Access, Social Engineering; Password-based attacks, and IP Spoofing.

1.2 Problems in Monitoring Systems

According to Feyissa [23, 24] allowing the monitoring system to gather more data for the intention of adaptation may affect stern security problems. Because, sometimes, the monitoring system attacked by an intruder may misuse the information. The authors have also discussed the need to limit the monitoring system for the purpose of adaptation. They used security metrics for measuring the security criticality of data. These security metrics are helping in recognizing the malicious activities, which give security threats to distributed systems.

Aredo and Yildirim [4] discussed about the two main problems in adaptive distributed systems. Initially, system monitoring to collect information necessary for adaptation may cause security issues. Information about users, their activities, and message details gathered by the monitoring node typically take place outside the destination system. Such monitoring causes a significant security risk in the case, if an attacker overtakes the monitoring system. Finally, limiting the monitoring may hamper the capacity of the system to adapt to the varying environment and maintain the security mechanism. Aredo and Yildirim [4] do not address about how to achieve an adaptation through the minimal impact on its security mechanism. In addition, the authors do not discuss about what kind of data can be monitored and how to monitor without affecting the performance of the distributed monitoring architecture.

Aredo and Yildirim [4] presented the monitoring of adaptive distributed systems and security metrics for the adaptive distributed systems by using security metric functions. The basic mechanisms of ADS include monitoring, change detection, and reconfiguration in response to the changes in the environment [25]. A monitoring component is employed for collecting information on parameters, which are analyzed later to detect changes in the environment of the target-distributed system. The intruder may overtake the role of a monitoring system and misuse the information.

There are two scenarios for monitoring the target systems, one is monitoring of module that is a part of the system and another monitoring that is outside the system. Here authors considered the scenario of monitoring outside the system because of following reasons. One of the reasons is that, if monitoring is part of the system then it is difficult in directly controlling the entire distributed system from centralized server. It requires additional chronological methods of monitoring. Whereas external monitoring, system is depending upon the existence of a single thread of control. In addition, the problem lies in direct monitoring the system in its total from a lone point of surveillance [26], which needs the compilation of locally observed activities in order to build global observations. Second, there is no central point of decision making when monitoring is part of the system. Thus, the method of making decisions in a distributed system may itself be distributed. The third reason is that, the dependencies between different programs in a distributed system are such that any alteration in the activities of one program can alter the behavior of the whole distributed system.

Figure 24.1 depicts a scenario of an external server monitoring the distributed system. An attacker may have the right of entry to the insecure communication channels and interrupts the user activities by collecting confidential data information.

Intrusiveness [27] is the effect of monitoring the network because during the sharing of resources with the monitoring system intrusions are entering into the system. Intrusive may modify the activities in a random manner. The problems of intrusive monitoring include, degradation of system performance, incorrect results, delay in execution, and masking or creating deadlock situations. The intrusiveness can be measured by identifying events in the monitoring systems. The monitoring systems can be of three categories, viz. software monitors, hardware monitors, and hybrid monitors.

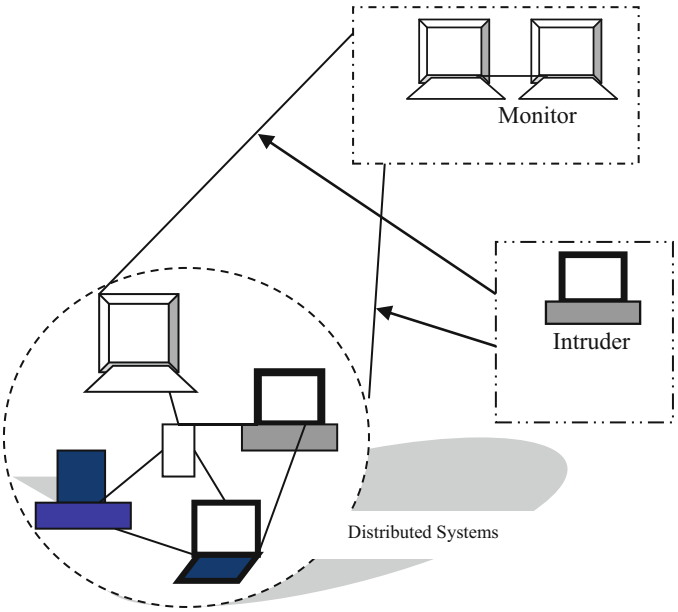

Fig. 24.1 External monitoring of distributed systems

1.3 Wireshark Monitoring Tool

Wireshark is a network packet analyzer tool, which has been used to capture the network packets [1]. The Wireshark tool could be used to monitor the online computers by using IP Address of the nodes.

1.3.1 Purposes of Wireshark

The Software developers, Network engineers, Network system administrators, and Researchers use Wireshark tool for various purposes, which include the following [4].

- Network System Administrators use this tool for troubleshooting the network problems.
- Network engineers use this tool for observing the security problems.
- Software developers are using it for debugging the protocol implementations.
- Researchers have been using this tool for studying the network protocol internals.

1.3.2 Characteristics of Wireshark

The following are the Characteristics of the Wireshark tool.

- By using the network interface, Wireshark tool captures live network packets.
- The complete protocol information can be displayed in each and every captured network packet.
- The captured packets could be saved and opened later. Also these captured packets could be filtered on the basis of some specific criteria and protocol.
- Wireshark tool colorizes the captured packets based on filtrations of packets.

1.3.3 Features Not Present in Wireshark Tool

The following features are not available in Wireshark tool when it is used in monitoring networked systems.

- Wireshark tool act as a packet analyzer, so it never detects any intrusions during monitoring. Hence it does not act as an Intrusion Detection System. For example, Wireshark tool does not provide any vigilant message if some other user changes the network bustle or performs unauthorized modifications in the network.
- By using Wireshark tool, it is very difficult to detect any kind of manipulation over the network.
- Wireshark tool gives the dump of information at a time. It is difficult to do the customization of data by using this tool.
- It is very difficult to implement and integrate with users connections, because Wireshark tool is an open source.
- It is very hard to keep track of the network activity of individuals, since it is not user-friendly.

However, the proper use of Wireshark tool helps to identify all of the above-mentioned activities, this will be discussed more in the section below.

1.3.4 Why Wireshark Tool?

In this chapter, Wireshark tool has been used to monitor the data. The Wireshark is better than other commercial tools because of the following reasons.

- It is an open source tool for monitoring.
- All other existing tools are not given attention to packet dissectors, which helps to count the packets and decoding it.
- In Wireshark tool, specialized hardware is not required to capture the packets and also, the packet capturing speed is high in Wireshark tool.
- The full documentation is available in Wireshark tool and protocols are not licensed like in other commercial tools.

The Wireshark tool has been selected for investigation of existing monitoring tool. When deploying the distributed nodes, captured network traffic [28] has been utilized. Captured network traffic may be utilized to ensure the good network connectivity. Also, the captured network traffic helps to locate the new nodes in a network. By passively capturing and analyzing packets facilitate deploying of new nodes, testing of existing nodes, and resolve the problems associated in distributed nodes. The system contains sniffer node and multiple user interfaces. The monitoring tool must have the following features, which includes

- Captured packets [29] should be stored according to its capture time.
- Captured packets should be interpreted as per human readable format in each protocol.
- Approximates the link qualities of the systems within the communication range.
- User interface shows only the selectively user required information.
- User interface shows the captured information in real time streams.
- Captured and analyzed information may be stored in file.

Apart from its good qualities, Wireshark tool has some drawbacks in terms of security breaches. The Wireshark tool is vulnerable to an attacker, which has been proved in this research work.

Wireshark tool identifies the viruses and worms traffic by looking at the raw data transmitted across the network. This has been viewed at the bottom of the Wireshark tool window. The suspicious packets are filtered using display filter. The display filter helps to identify traffics like, DCEPRC, NetBIOS, and ICMP. These traffics are not viewable under normal circumstances.

1.4 Algorithmic Procedure for Monitoring

A message that is transferring between the nodes of the distributed system as well as data related to that message could be monitored. The algorithmic procedure for monitoring of network using Wireshark tool has been explained in Fig. 24.2 [15].

As illustrated in Fig. 24.2, the monitoring node should start the Wireshark tool first. Select any one source node in the distributed system and create a new message for sending it to the destination node. Before sending this message to the target node, the source node has to select destination node IP Address. Upon selecting the IP Address of the target node, source node sends the created message. The message is received at the destination end. The monitoring node may capture these messages using the Wireshark tool and right click on the follow TCP stream to view the full message.

A simple chat application has been implemented to demonstrate the message passing between two users in a network. The following code snippet shows that, client1 and client2 should communicate through socket programming concepts.

client1 = New TcpClient (ComboBox1.Text, Port_No)
client2 = New TcpClient (TextBox2.Text, Port_No)

```
Algorithm_Monitor ()
{       Start Wireshark Software in Monitoring_Node
      Select Source_Node in developed application
       Source_Node creates a New_Message
       Source_Node locates the IPAddress of Destination_Node
       Send New_Message to Destination_Node
       Monitoring_Node capture the packet of New_Message for Analysis
       Right click on follow TCP Stream of captured packet
       Monitoring_Node now can monitor the full message transmitted over network.
}
```

Fig. 24.2 Algorithmic procedure for monitoring

The implementation of monitoring involves the Monitor_Node class. The Monitor_Node class consists of three constructors called Analyzer, Observer, and Capture. The Monitor_Node adds the requisite number of parameters and updates it for the intention of monitoring with a network.

The following code snippet demonstrates the working of Monitor_Node.

```
    Monitor_Node ()
     {
     //Add parameters that need to be monitored
     //Update parameters at each time
     //Call Analyser ()
     // Call Observer ()
     // Call Capture()
     }
     // The Analyzer class
class Analyzer extends Monitor
     {
     Analyzer ()
     {
super();// call the parent class for analysis
          // Adding "listener" parameter to the class
          addAttr(new_Attribute("listener", false));
     }
     }
     // The_Observer_class
Class_Observer_extends_Monitor
```

```
      {
      observer ()
      {
super();// call the constructor of the parent class extends
             Monitor
      // Adding read-only "result" parameter
      addAttr(new_Attribute("result"));
            }
              }
```

The functionalities of Observer, Analyzer, and Capture have been depending on the monitoring rules of the networks. In this implementation, the analyzer collects the information and applies the monitoring rules and generates the results.

The observer fetches the results from the database, normally in the form of alarm signals generated by analysis.

```
     // Capture class captures the port number
class Capture extends Monitor
      {
      Capture ()
      {
super();// call the constructor of the parent class
          // Capture port attribute
          capAttr(new Attribute("port_no"))
                 }
         }
```

Wireshark tool should be specified by promiscuous mode before the commencement of capturing, otherwise Wireshark tool captures all incoming and outgoing packets of the LAN environment. In the promiscuous mode, the following parameters are required to be considered while capturing the packets.

- Limiting every packet to "n" bytes permits the user to denote the highest number of information that can be captured for every packet. The default limit of each packet will be 65,535.
- The fixed buffer size is used to capture the packets and it temporarily keeps the packets before writing it to the permanent memory.

In the case of monitor mode capturing, Wireshark tool captures all incoming and outgoing traffic packets. During this mode, the network adapter has been disassociated from the network. The following parameters need to be set during the capturing of packets, which includes host name, port_number, void authentication, and password authentication. The host name or IP Address has been selected to capture the packets. The port number has been set to capture the remote packets, 2002 being the default port number. The null authentication is considered as insecure capturing of packets. Credentials are required for password authentication to capture packets.

The libpcap engine of Wireshark tool captures the data packets from the network card and keeps all data packets in a kernel buffer. The kernel buffer has been read by the Wireshark tool. When the Wireshark tool deals with large file of capture, it slows down the network speed. Hence, multiple file option has been used to save the captured file. The following different methods have been used while saving the files.

Algorithm_Capture_Packet ()
// Get the Network Device information from the network
NetworkInterface [] devices = JpcapCaptor.getDeviceList ();
// Obtain the available list of network interfaces
// for each network interfaces do the following
// Display the name and description of network interface
// Display the name of data link and corresponding description
// Display the network interface MAC address
// Display the its IP address, subnet mask and broadcast address
// Capture the packets by calling openDevice method and set its interface information
openDevice (NetworkInterface intrface, int snaplen, boolean promics, int to_ms);
JpcapCaptor captor=JpcapCaptor.openDevice(devices[index], 65535, false, 20);
captor.setFilter("icmp",true);
// Capture a single packet and display it.
End_Algorithm_Capture_Packet()

Fig. 24.3 Steps involved in packet capture procedure

A single temporary file has been created by default and saved. Multiple files with continuous mode have been used, if the file is more than threshold size. Multiple files with ring buffer, limit the maximum disk usage and keep only the latest captured packets.

The command ether_ [src|dst]_ host_ <ehost> allows monitoring tool to filter only Ethernet host addresses. The command [tcp|udp]_[src|dst]_ port_<port> allows the monitoring tool to filter only TCP and UDP port numbers with protocols.

The steps shown in Fig. 24.3 describe the process of packet capturing using Java API. The Jpcap is an API used in Java, it provides access to low level network information. Initially, jpcap.JpcapHandler interface class has been created and it allows processing the packets.

public class JpcapTip implements JpcapHandler.

```
{
    public void handlePacket(Packet packet)
    {
        system.out.println(packet);
    }
}
```

The Jpcap API provides a method jpcap.Jpcap.getDeviceList() to listen to the network device. This method returns an array strings as shown in the code snippet.

String[] devices = Jpcap.getDeviceList();

Once device names have been listed, then monitoring node must choose one device for listening.

String deviceName = devices[0];

Once device is selected and it has been open for listening by using method jcap,openDevice(). The method uses four parameters to open, which includes name of device, maximum number of bytes to read, status of promiscuous mode of the device, and timer value.

Jpcap jpcap = Jpcap.openDevice (deviceName, maxBytes, mode, timeout);

The openDevice method listens device packets by calling two possible methods such as processPacket() and loopPacket(). The processPacket method captures packet until it reaches the maximum number of specified.

Packets by user. The loopPacket method captures continuously.

jpcap.loopPacket(-1, new JpcapTip());

In this way, network packets have been captured by the monitoring node using Java API.

1.5 Implementation of Application for Message Exchange

In all kinds of networks like WAN or LAN users' exchange of messages between themselves with proper message formats. These message formats include IP Address as one of the key parameters. With the help of IP Address, any source systems or nodes present in the WAN or LAN can route the packets to the destination. The sending and receiving messages between two users' have been demonstrated in Figs. 24.4 and 24.5, respectively [15].

Initially, sender node locates DHCP server on the network. Then the sender node locates its IP Address by sending broadcast message to IP_Address 255.255.255.255. Upon receiving broadcast packet, the DHCP server sends a

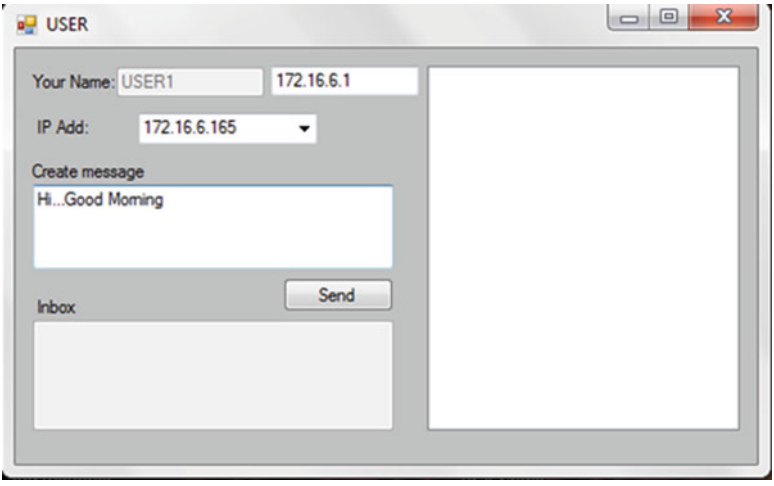

Fig. 24.4 Message sending process by Source_Node1

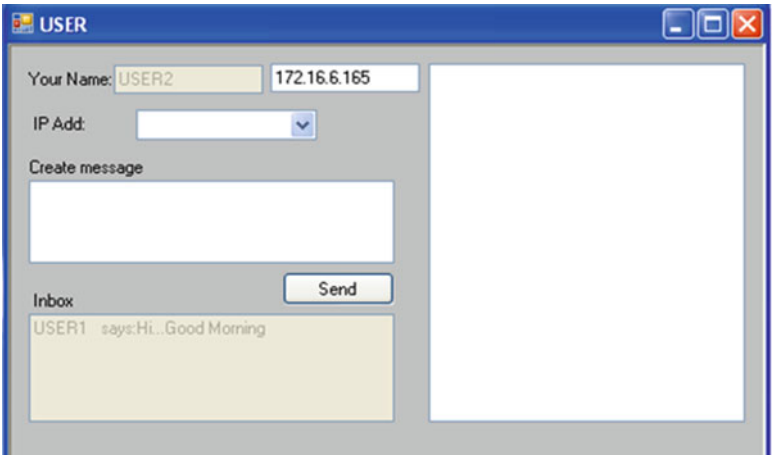

Fig. 24.5 Message receiving process by USER_Node2

response to the sender with packet containing IP address. After receiving this packet, the sender sends the request message to get the addressing information from DHCP server. In this way, sender locates the IP Address.

1.5.1 USER_Node1 is Sending a Message to USER_Node2

In this case, USER_Node1 is intended to send the message to USER_Node2, who may present in the same distributed network. Before transmitting the created message to USER_Node2, the USER_Node1 selects the destination IP Address (172.16.6.165) of the USER_Node2 as shown in Fig. 24.4.

1.5.2 USER_Node2 Received Message from USER_Node1

The USER_Node2, which is present in the same network, receives the message in its Inbox. In this way, USER_Node2 opens and reads the messages coming to its Inbox. This scenario has been illustrated in the Fig. 24.5. Similarly, USER_Node2 also creates a new message and sends it to USER_Node1 by selecting the IP Address (172.16.6.1) of USER_Node1.

Both cases are considered for exchanging of messages over distributed networks. These messages need to be monitored for the purpose of adaptation. The adaptive system helps to improve the quality of service of the message during sharing over distributed networking nodes. In this regard, one application is required for monitoring process. By the help of monitoring report, it is very easy to vary the bandwidth of the network. Also, this monitoring report helps to find existence of any node. The monitoring process has been explained in the next section.

1.6 Implementation of Monitoring Scenarios Using Wireshark Tool

The Sect. 1.5 describes the message passing procedure that uses IP_Address as a routing parameter. However, during this message transmission, IP_Address is disclosed to everyone. A third person who is present in the network may observe these messages with the service of monitoring tool with appropriate parameters. By using Wireshark monitoring tool, it is possible to view TCP packets. Instead of viewing in bunch of small chunks of data from client to server, the TCP stream sorts these chunks to make it easily viewable. Rather than taking in small packets and combining packets, the attacker may use follow TCP stream procedure to find the entire information. This scenario has been implemented in and presented in this chapter as shown in Fig. 24.6. As per the scenario, the user who is present in the network may right click on IP Address 176.16.6.165 and select follow TCP stream to get the entire information [15].

"Follow TCP stream" allows user to view all the packets on a TCP stream data between a pair of users. It is one of the most useful analyses in monitoring tool. The following TCP stream combines all the data pertaining to each packet. The TCP stream sorts all small packets and combines it for proper observation. The TCP-based method helps to view the data from the TCP stream, similar to what the application layer does. The type of data viewed may be passwords of Telnet stream or confidential messages communicated between the users. The size of the receiving TCP window decides the data transmission rates of the network. The TCP receive window updates about the packets that have been shared between the users during

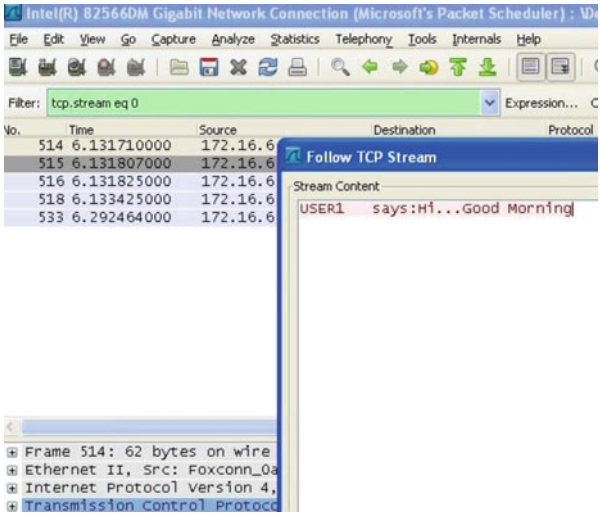

Fig. 24.6 Monitoring using follow TCP stream

transmission of data. Based on this size, the data transmission speeds up or slows down.

In situation of monitoring node, any intruder node can view the full message transmitted over the broadcast web. As illustrated in the Fig. 24.6, the intruder may right click on the packet number 515 on Destination_Node2 IP Address 172.16.6.165 by using follow TCP stream. In this way, the monitoring tool like Wireshark may be misused by intruder to view the confidential messages. In this regard, a secure communication mechanism is required during the monitoring activities.

1.7 Implementation of Secure Way of Monitoring

Wireshark tool has been used to monitor the activities of suspected employees who belong to the same network. The tool initially captures the suspected employee packets and deciphers it to view the contents of the packets. The display filter has been used to filter out TCP packets. The packets have been examined by identifying small bits of text information during transmission. Every packet data has been copied individually and combined to see entire message being transmitted. The Wireshark tool fetches entire information by right clicking on each packet. In this way, the TCP stream window displays the complete chat, which is communicated between two suspected employees. This feature allows the user to view the chat just as the application layer views it. By using this feature, anyone can view the passwords of other users in a Telnet stream.

A distributed system contains a group of nodes associated together by a computer network in order to switch the data. With this implementation mechanisms "N" numbers of nodes are connected all over distributed systems, among which one node is considered as a monitoring node. The developed application is run on web server to get its services over the distributed systems. Any user who is present in the distributed system can access this application. However, to apply the developed application some configuration of authentication is needed on both positions of the users.

The intruder, who may be present within the distributed networks, may try to access the data, which is transmitted between two organizations or users. Intruder also can utilize the same Wireshark monitoring tool to capture the information. The intruder may follow the same procedure like, clicks on a "Follow TCP stream" option of Wireshark monitoring tool to get the exact message.

Security mechanism has been employed in order to protect the data, which shifts between the distributed systems and Monitoring Node. Sender Node should need to encrypt the information in RSA algorithm or other equivalent algorithm while sending. In this chapter, we have discussed RSA 1024-bit encryption procedure. On the receiving end, information is decrypted and displayed over the inbox of the recipient. When the intruder tries to access this information through existing

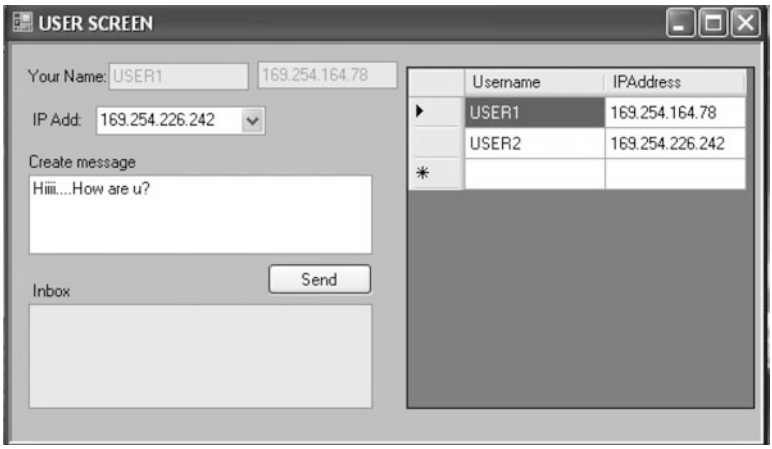

Fig. 24.7 GUI of the users in the network

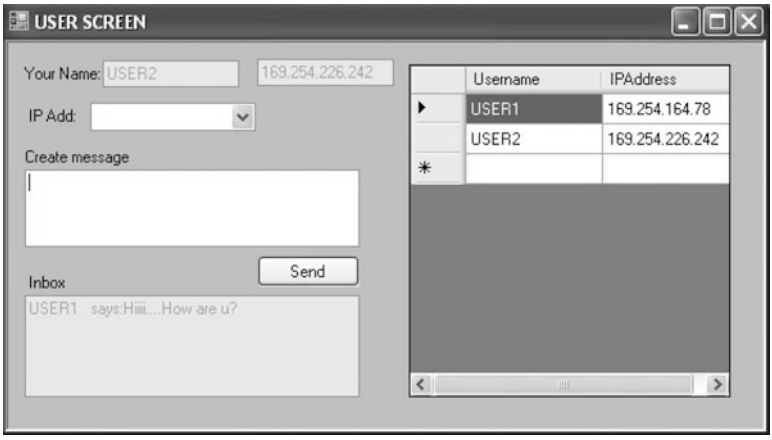

Fig. 24.8 Inbox of the User2

monitoring tool like Wireshark tool, only an encrypted message is displayed on the screen of the intruder.

Figure 24.7 shows the user screen of the sender process. In this, USER1 (169.254.164.78) has sent a created message to USER2 (169.254.226.242) by selecting an IP Address field. Username and its corresponding updated IP Addresses are displayed on the right side of the screen. Before broadcasting the message, the USER1 encrypts the message using RSA algorithm. The encryption process has been embedded along with send procedure, by clicking send button as shown in Fig. 24.8. The separate encryption button is avoided here to cut down the time delay of user interaction [15].

As evinced in Fig. 24.8, USER2 receives the message sent by USER1. The message is displayed over the inbox of USER2. On picking up the message, USER2 decrypts it and meets the original Message automatically. The decryption process has been embedded here to cut down the time delay of computation.

All the users' messages are encrypted and it provides a security to the monitored message. However, one important thing to be noticed here is that, all the parameters are encrypted; user is not having any option to bypass the encryption procedure.

1.8 Pseudo Code for Secure Transmission and Reception of Messages

Figure 24.9a shows the pseudo code of the sender process of the monitoring tool. While sending message to the user, sender process creates a string of message in line. These messages have been read line-by-line and encrypts entire message line-by-line using RSA algorithm. Also message has been displayed on the senders' screen.

Figure 24.9b depicts the receiver process. On the receiver side, the receiver process receives encrypted messages line by line and after reading it decrypts using RSA algorithm line by line. Also, original message has been displayed on screen of receiver.

2 Secure Way of Monitoring

As explained in the earlier section, massage can be supervised through the existing monitoring tool like Wireshark tool. Figure 24.10 shows the screen of Monitoring tool with an encrypted message. The Monitoring node may use "follow TCP stream" to look at the message. The coded message exposed on the cover of the Wireshark monitoring tool [15].

The implementation of secure way of monitoring has been done with RSA algorithm using Java programming. The pseudo code of RSA algorithm [30] is shown in Fig. 24.11. RSA is the widely used algorithm for secure encryption of data. In this algorithm, the USER_Node1 encrypts the information with the help of public key of USER_Node2 and the USER_Node2 decrypts the ciphertext with the help of private key of USER_Node2. The Java provides a BigInteger for the calculation of large prime numbers and uses 1024 bits of key length. Since 1024 bits provides more security for the messages, in terms of infeasibility for attackers to decrypt the messages.

The code snippet shown in the Fig. 24.11a explains about the generation of public key and private keys of RSA algorithm. In addition, code in the Fig. 24.11b explains how to encrypt or decrypt using these generated key pairs.

a

```
Sender_Process()
{
// User1 sends a message to User2
// while loop for sending messages to users
      while(true)
         try
         {
          pw.flush();
          String line = in.readLine();
                line =rsa.encrypt(line);
        pw.println(line);
         }
catch (Exception se) {//Connection Closes after sending otherwise gives exception}
}
```

b

```
Receiver_Process()
{
// open reader to receive messages from other users
// loop reading messages from server
          while(true)
          {
              String line = inFromUser.readLine();
              line= rsa.decrypt(line);
              System.out.println("From User" + line)
          }
}
```

Fig. 24.9 (**a**) Sender process of monitoring. (**b**) Receiver process of monitoring

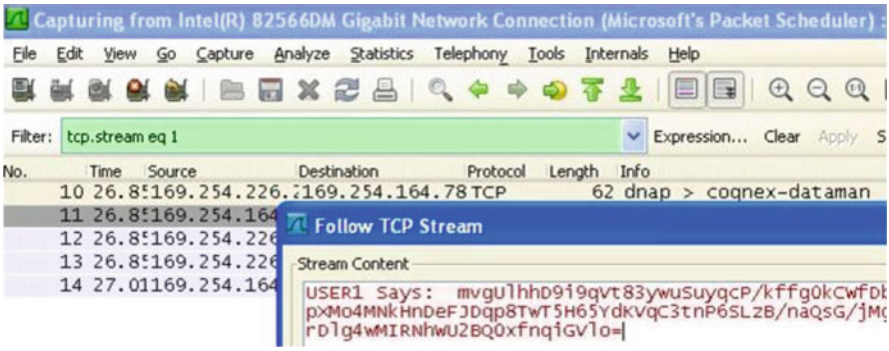

Fig. 24.10 Monitoring using Wireshark tool

a

```
RSA_Algorithm ()
{
//  Declaration of bit length of RSA using bit_length = 1024
//  Use of SecureRandom() function for getting random number
// Select any 2 large prime using function BigInteger(bit_length / 2, 100, random)
// Compute n by using   n = p.multiply(q)
//Calculate z
z = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));
//    Generation of key pairs, private and public keys
     en_key = new BigInteger("3");
    while (msg.gcd(en_key).intValue() > 1)
    {
          en_key = en_key.add(new BigInteger("2"));
    }
            dec_key = en_key.modInverse(z);
 }
```

b

```
// Encryption of given message
   public BigInteger encrypt(BigInteger msg)
{
   return msg.modPow(en_key, n);
 }

// Decryption of given message
   public BigInteger decrypt(BigInteger msg)
{
   return msg.modPow(dec_key, n);
 }
```

Fig. 24.11 (**a**) Pseudo code of RSA algorithm. (**b**) Pseudo code of RSA encryption and decryption

The RSA algorithm has been used mainly for two purposes, namely it is a factor-based algorithm and its computing power increases constantly. RSA-1024 is considered as safe enough for protecting most of the vital information in the web. However, unencrypted messages take comparatively less processing time. On the other hand, these unencrypted messages are not confidential while transmission. During the experimental process, Public Key Infrastructure (PKI) has been adopted between the users with pairs of RSA 1024 bits asymmetric keys. To enable secure communication between the involved parties during the monitoring, each party must receive a list of the public keys of the other users that they will communicate with. In that case, each user receives the other user's public key for encryption process.

The PKI needs to perform in order to provide trust and security to electronic communication. The following functions are involved in working of PKI-based key management [31].

- Generating public key and private pairs for creating and authenticating digital signatures.
- Providing authentication to control access to the private key.
- Creating and issuing certificates to authenticate users.
- Registering new users to authenticate them.
- Maintaining history of keys for future references.
- Revoking certificates that are not valid.
- Updating and recovering keys in case of key compromise.

Cryptosystem techniques are proven safe. In this regard, the only analysis can be made to outline is how to decrypt a message without knowing the decryption key. Brute force methods are very simple, but lengthy to crack a message for attacker. However, attackers need not to crack entire encryption scheme to get portion of the message. In spite of several attempts, no one has been succeeded with 1024 bits of RSA algorithm. Such a resistance to attack makes RSA secure in practice. In RSA, it has been proved that it is very difficult for factorizing large prime numbers. Suppose, if large prime numbers p and q are having 100-digit numbers, then resulting n would be approximately 200 digits. The factorization of above case would take far too long time for breaking the code. Similarly, methods for determination of d are also difficult. Factorization of algorithm is still an age-old mathematical problem, contributed by Fermat and Legendre.

The RSA has been used widely in most of the application for following reasons: (1) RSA provides privilege of key revocation; (2) RSA provides distribution of new key during revocation of existing key; (3) RSA supports the spreading of the revocation; (4) RSA helps recovery from the leaked key.

The Wireshark tool decrypts the encrypted packets of Internet Key Exchange version 2 only. All other packets like Internet Key Exchange version 1 and Encapsulation Security Payload are decrypted with the help of ISAKMP (Internet Security Association Key Management Protocol). The following fields of the ISAKMP protocol have been used for encryption and decryption of packets. Initially length of 16 hex characters has been created for Senders Security Protocol Index (SPI). Similarly, length of 16 hex characters has been created for Receivers SPI.

The IKEv2 packets of sender to receiver have been encrypted/decrypted by using the key en_key. Similarly, the IKEv2 packets of receiver to sender have been encrypted/decrypted by using the key dec_key. The Integrity Checksum for receiver to sender has been calculated by the key en_key. Similarly, the Integrity Checksum for sender to receiver has been calculated by the key dec_key.

3 Summary

The framework for security mechanisms has been discussed in two ways. In the first, investigation of existing security mechanisms during monitoring and in the second, implementation of secure communication channel for monitoring.

Initially, existing Wireshark monitoring tool has been used for monitoring process. In this regard, a chat application has been developed for transferring messages between two users. The algorithmic procedure for monitoring has been explained in detail. The packet capture algorithm also has been discussed here. The impacts of monitoring scenarios have been discussed with help of implementation results. Finally, a secure way of implementation of monitoring mechanisms has been discussed with the help of RSA algorithm.

References

1. Sharpe, R., & Warnicke, E. (2014). *Capturing live network of data, Wireshark user's guide: For Wireshark 1.99*. https://www.wireshark.org/docs/
2. Mittal, H., Jain, M., & Banda, L. (2013). Monitoring local area network using remote method invocation. *International Journal of Computer Science and Mobile Computing, 5*(2), 50–55.
3. Moraes, D. M., & Duarte, E. P. (2011). A failure detection service for internet-based multi-as distributed systems. In *Proceedings of IEEE 17th International Conference on Parallel and Distributed Systems* (pp. 260–267).
4. Aredo, D., & Yildirim, S. (2006). Security issues in adaptive distributed systems. In *Proceedings of the Fourteenth European Conference on Information Systems (ECIS)* (pp. 2206–2215).
5. Kotenko, I., & Doynikova, E. (2014). Evaluation of computer network security based on attack graphs and security event processing. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 3*(5), 14–29.
6. Liu, Y., Xingyu, L., Jian, Y., & Xiao, Y. (2016). A framework of a software defined security architecture. *China Communications, 13*, 178–188.
7. Wiczer, J., & Wiczer, M. B. (2015). Improving energy efficiency using customized monitoring tools. In *Proceedings of 117th Metalcasting Congress, Modern Casting, Vernon Hills, IL* (pp. 36–39).
8. Wireshark Tutorial (http://www.wireshark.org/docs/wsug_html_chunked/), man pages (http://www.wireshark.org/docs/man-pages/), and a detailed FAQ (http://www.wireshark.org/faq.html) Retrieved April 2015.
9. Fuginia, M., Hadjichristofib, G., & Teimourikiaa, M. (2015). *A web-based cooperative tool for risk management with adaptive security, future generation computer systems* (pp. 1–16). Nicosia/Limassol: Frederick University.

10. Hernandez, C., Pedraza, L. F., & Salgado, C. (2013). A proposal of traffic model that allows estimating throughput mean values. In *Proceedings of 27th International Conference on Advanced Information Networking and Applications Workshops* (pp. 517–522). IEEE Computer Society.
11. Pop, F., Arcalianu, A., Dobre, C., & Cristea, V. (2011). Enhanced security for monitoring services in large scale distributed systems. In *Proceedings of International Conference on Intelligent Computer Communication and Processing (ICCP)* (pp. 549–556). IEEE.
12. Murugan, M., Kant, K., Raghavan, A., & Du, D. H. C. (2014). FlexStore: A software defined, energy adaptive distributed storage framework. In *Proceedings of 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems* (pp. 81–90). IEEE Computer Society.
13. Jeswani, D., Natu, M., & Ghosh, R. K. (2012). Adaptive monitoring: A framework to adapt passive monitoring using probing. In *Proceedings of 8th International Conference on Network and Service Management (CNSM)* (pp. 350–356).
14. Penteado, M. G., & Trevelin, L. C. (2012). JMonitor: A monitoring tool for distributed systems. In *Proceedings of International Conference on Systems, Man, and Cybernetics, COEX, Seoul, Korea* (pp. 1767–1772).
15. Kotari, M., Chiplunkar, N. N., & Nagesh, H. R. (2016). Framework of security mechanisms for monitoring adaptive distributed systems. *IOSR Journal of Computer Engineering (IOSR-JCE), 18*(4), 25–36.
16. Comuzzi, M., & Martinez, R. I. R. (2014). Customized infrastructures for monitoring business processes. In *Proceedings of 8th International Symposium on Service Oriented System Engineering* (pp. 122–127). IEEE.
17. Oikonomou, G., & Apostolopoulos, T. (2007). A framework for the management of distributed systems based on SNMP. In *Proceedings of 22nd international symposium on Computer and information Sciences(ISCIS)* (pp. 78–83). IEEE.
18. Chen, S., Nepal, S., & Pandey, S. (2012). A unified monitoring framework for distributed information system management. In *Proceedings of 8th International Conference on Computing Technology and Information Management (ICCM)* (pp. 259–264). IEEE.
19. Fonseca, H., Rocha, E., Salvador, P., & Nogueira, A. (2014). Framework for collecting social network events. In *Proceedings of 16th International Conference on Telecommunications Network Strategy and Planning Symposium* (pp. 1–6). IEEE.
20. Atighetchi, M., & Adler, A. (2014). A framework for resilient remote monitoring. In *Proceedings of 7th International Symposium on Resilient Control Systems (ISRCS)* (pp. 1–8).
21. Jarraya, Y., Raya, S., Soeanua, A., Debbabia, M., Alloucheb, M., & Bergerb, J. (2013). Towards a distributed plan execution monitoring framework. In *Proceedings of 3rd International Symposium on Frontiers in Ambient and Mobile Systems (FAMS), Procedia Computer Science 19* (pp. 1034–1039). Elsevier.
22. Agbogun, J., & Ejiga, F. A. (2013). Network security management: solutions to network intrusion related problems. *International Journal of Computer and Information Technology, 4*(2), 617–625.
23. Feyissa, M. (2007). *Monitoring distributed systems for adaptive security*. Master thesis, Department of Computer Science, School of Graduate Studies of Addis Ababa University, Addis Ababa.
24. Zhou, Z. (2013). *Design and realization of distributed intelligent monitoring systems using power plant* (pp. 595–601). Berlin: Springer.
25. Schlichting, R. D., & Hiltunen, M. (1998). *Designing and implementing adaptive distributed systems*. University of Arizona, Arizona. Retrieved Feb, 2018, from http://www.cs.arizona.edu/adaptiveds/overview.html
26. Sarma, B., & Dasgupta, S. (2014). Dynamic load calculation in a distributed system using centralized approach. *International Journal of Advance Research, Ideas and Innovations in Technology, 2*(1), 5–10.
27. Falai, L. (2007). *Observing, monitoring and evaluating distributed systems*. Ph.D. Thesis, University of Lisboa, Portugal.

28. Hanninen, M., Suhonen, J., Hamalainen, T. D., & Hannikainen, M. (2011). Practical monitoring and analysis tool for WSN testing. In *Proceedings of International Conference on Design and Architectures for Signal and Image Processing (DASIP)* (pp. 23–32). IEEE.
29. Qadeer, M. A., & Zahid, M. (2010). Network traffic analysis and intrusion detection using packet sniffer. In *Proceedings of Second International Conference on Communication Software and Networks* (pp. 313–317). IEEE.
30. RSA elliptic curve cryptography. Retrieved November 30, 2017, from http://www.rsa.com/rsalabs/node.asp?id=2013
31. Choudhury, S., Bhatnagar, K., & Haque, W. (2002). *Public key infrastructure implementation and design*. New York: Hungry Minds.