

Distributed Network Forensics using JADE Mobile Agent Framework

Asha Nagesh, *Graduate Student, Arizona State University, Division of Computing Studies, Sutton Hall, Suite 140, 7001 E. Williams Field Rd, Mesa, AZ 85212. Email: Asha.Nagesh@asu.edu*

Abstract— Networked computers are being used more and more for committing crimes varying from financial fraud, identity theft, and resource violation to pornography. All activities on a network or the network traffic can be recorded in log files which form a veritable archive that can be used to analyze and reconstruct the human activity trail involved in the crime. This report presents a software tool based on mobile agent architecture that provides an efficient mechanism for evidence collection and visualization of network traffic to aid a human analyst's network forensics investigation. The existing architectures and software solutions suffer from centralized data collection and processing congestion because of the sheer volume of data, network congestion at the central system performing the forensics analysis and single point of failure. This paper presents a distributed architecture consisting of independent agent entities that travel to and from scattered systems to selectively gather network traffic logs, examine them and return results that will be displayed in a single combined user interface. This application of software agents in the development of network forensics software distributes both the processing load and storage requirements to multiple networked systems, relieves network congestion on a single system and with replication can also alleviate single point of failure problem. The specific design and implementation are done using the Java Agent DEvelopment Platform (JADE). The results obtained in the course of the project are explained.

Index Terms—Distributed software, JADE software agents, Network forensics, Network traffic log analysis, Mobile agent system.

I. INTRODUCTION

In the real world there will always be unscrupulous people who for personal gain and gratification commit crime against enterprises and people. With the advancement in computer technology both in terms of individual computer capability and collective Internet services, networked computers facilitate such crimes by providing an easy and readily available infrastructure. The same infrastructure also contains trail of evidence left behind by any activity and Computer forensics is the area that deals with collecting, analyzing and presenting these trails stored in the compromised infrastructure as digital

evidence for the purpose of proving and possibly convicting crime in cyber space. Network forensics is a part of this relatively new field where the network traffic trails are investigated in order to obtain digital evidence. The purpose of this project is to develop a network forensic tool using distributed mobile agent architecture to help an investigator efficiently examine, analyze network traffic logs and thus gather digital evidence.

A. Background

With the explosion of the Internet, more and more enterprises offer their services via the World Wide Web that consumers can make use of. But cyber space is also abused by both insider and hacker attacks featuring fraud, unauthorized access, denial of service, viruses, data manipulation and misuse of resources. Hence there is need for the investigation of charges related to computer crime in order to protect people and businesses [1]. Unlike traditional investigation, computer and network forensics have to produce original information and data taken from the storage media as evidence in order to persecute any kind of criminal case. Computer storage media contain the trail of actions performed on its various processes and files as record entries in system or log files. Typically a log file contains a timestamp and a short description of what happened relevant to the application or process writing the log. For example, mail logs contain a detailed list of which users sent e-mail to whom, and when. Router logs contain flow information of all the packets between two points. System and network log files are considered to be the primary information sources that record every action that occurs using the hardware and the software present on each individual computer [2].

Network forensics involves the capture of information from network traffic, analyzing and interpreting the traffic information stored in network information containing system logs like the packet sniffer logs, intrusion detection system logs, web server logs, router logs, firewall logs etc... These logs can be examined to determine if an email was received, if proprietary material was read without authorization, recreate events on a network to know what exactly transpired, connection details, source and destination addresses and so on. By this method it is thus

possible to extract and present concrete evidence of cyber attacks in any court of law and hold attackers responsible for their actions.

Traditionally, intrusion detection systems and firewalls exist to monitor network security. These tools are part of network security analysis but, network forensics tool have a distinct niche in the area of network security. Intrusion detection system (IDS) are primarily for learning, detecting and reporting attacks as they happen in real time and have no evidence gathering feature. Moreover IDS detect certain types of attacks that are already in their knowledge base but cannot detect new types of attacks. Firewalls control traffic that enters a network and leaves a network based on source and destination address and port numbers. It is easy to find loop holes in such a firewall settings to exploit the network. A network forensics tool helps achieve the better of the two worlds in a sense by allowing efficient analysis of specific traffic that is of interest after you know that an event has occurred [3]. In [4] the author expects “In the future to see increased support in the open source world for collecting traffic remotely and seamlessly presenting it to a local workstation” (p. 653) and the network forensics software design approach implemented and presented here is a step in that direction.

II. PROBLEM STATEMENT

Network forensics has a number of special challenges that make collection and analysis of data difficult. The evidence or activity trail is often distributed [5]; furthermore a cyber criminal can be at more than one place at the same time hence certain white collar crimes can be investigated only by analyzing the collective set of actions across multiple systems or networks captured in distributed logs because the action on a lone system seems to be innocent. For example: distributed denial of service attack. The log files contain a wealth of information but are not in a format that adapts well for forensics investigation. Also, each different log needs an enormous amount of storage much like the petabyte storage that Google maintains at each of its ISP data centers [6]. For example, an example in [5] speaks of a case wherein over an 8 month period tcpdump accumulated over 15 GB. The prohibitive amount of data makes the manual investigation extremely cumbersome involving more of the examiner’s experience and is open to human mistakes. The network forensics tools that are present today can be classified to follow two types of architecture: the centralized design that stores the network traffic information logs, does the forensics analysis and presentation all in one bulky system; the existing distributed design involves mostly homogenous systems following the client server paradigm with the servers gathering information in their logs and the clients establishing connections to the different servers to obtain

the data and analyze. These existing architectures have the disadvantage of creating network congestion near the network forensics system and suffer from single point of failure.

The network forensics software tool presented here automates the collection of network data from multiple distributed heterogeneous systems thus in effect reducing the data storage requirement of a single monolithic system. This also enables the presentation of a global view of the scattered traffic data helping investigation of distributed white collar crimes. The current approach has a different mobile agent architecture that reduces processing, bandwidth and communication overhead by using an agent as the unit of design of the distributed software system. This design also has greater scalability, addresses the single point of failure of the centralized architecture, better load balancing and can provide real time monitoring of network traffic since additional agents can be created and deployed at run time according to need.

A. Scope

Considering the time expectations on this project, the scope of the network forensics tool is limited to a small subset of network forensics investigation. Three network traffic logs; the packet sniffer log of tcpdump, web server log and intrusion detection system snort’s log are considered as the forensic information source. This is a reasonable scope definition since a majority of events can be investigated using these logs especially the full content data obtained by the tcpdump log [4]. For reasonable depth of scope a small representative number of events are focused on. Typical events considered are protocol specific packets, intrusion alerts classified by severity, list of URLs accessed, web server usage statistics. A simple graphical user interface for the forensics analyst to specify the query components of the event to be examined in the logs is developed. The query tab provides interface to specify data selection criteria the form of an SQL query statement with SELECT, FROM and WHERE clauses. The query can be specified either directly or can be assembled using the Query Builder interface where separate components for each clause can be selected by the forensic analyst. The user interface also allows configuration of each information source from which to gather evidence.

- Remote Tcpdump: Machine’s Domain Name, From Log or to use live capture agent.
- Remote Snort: Machine’s Domain Name, Database driver name, Database URL, Database port, Database username, Database password
- Remote Web Server: Machine’s Domain Name, Log Path, log file Name, log format string.

The tool automates the collection of data from information sources at remote sites either to be used for conversion to common format or for further refinement by

a human operator.

B. Assumptions

Legal issues form an important aspect in computer forensics. In this project since personal and departmental systems are primarily used; evidence gathering is assumed to respect the laws as far as this author's knowledge. In this project it is also assumed that the logs are not tampered with and no effort is made to ensure untampering by using encryption mechanisms. Nor does the tool handle originally encrypted data. All logs contain timestamp information and are essential for searching relevant records. Since logs can be distributed among multiple systems to correlate between different logs, all systems are assumed to have their clock time synchronized while a related project at a later time will consider time correlation problem as the main issue. All systems are assumed to have the Java and JADE environment installed. The software agents developed do not have any malicious intent and are assumed to be implicitly trusted by the distributed hosts containing the network traffic information sources to execute on their respective hardware and using their software resources with full permissions. The communication between the agents across the network is also assumed to be secure and no mobile agent security issues are considered during the course of this project.

III. RELATED WORK

Some projects similar to the current undertaking include Distributed Intrusion Detection Systems (DIDS), EMERALD, Java Agents for Meta-Learning (JAM), Distributed Intrusion Detection System Using Mobile Agents (DIDMA), NetIntercept. Except for NetIntercept, all the others are primarily Intrusion Detection Systems (IDS) in that they report that some event happened at real time but do not give user enough information to independently confirm it. But, they are considered here as similar work since they are distributed network security monitoring software and some even use agent architecture.

DIDS [7] uses distributed monitors and artificial intelligence algorithms to perform anomaly intrusion detection but the intelligence of detecting intrusions is purely centralized. EMERALD [8] focuses only on network intrusion detection using a centralized analysis of all the local network traffic monitored to generate correlated reports. JAM [9] uses the agent technology for intrusion detection but does not allow any kind of network forensics event investigation. JAM implements distributed machine learning algorithms for intrusion detection on a large centralized database of all individual system data. JAM agents are custom stationary agents implemented in Java and do not use any pre-existing agent/mobile agent technology. DIDMA [10] is most similar to the current project in that it uses attack specific mobile agents to move

from node to node to obtain intrusion related data and aggregate and correlate multi agent data to detect and report intrusions across the entire network and is implemented on the Voyager agent platform. But, DIDMA uses mobile agents to detect attacks and thus is an IDS, whereas the current work differs from DIDMA by using mobile agents to collect network traffic data from distributed sources and presenting the same in a single combined user interface for efficient traffic analysis by a human network forensic analyst. At an implementation level difference the current project differs from DIDMA by using the JADE mobile agent framework instead of Voyager.

NetIntercept [11], a forensics tool, has a completely centralized architecture for collection and analysis. It records all traffic onto a single hard drive from which the traffic is selected and analyzed in batch mode. Because of the centralized architecture, storage and processing capability of the system on which NetIntercept runs is enormous. As the network size increases with addition of more machines, the scalability of a single NetIntercept collection and analyzing system is small.

A. Technologies and Libraries Used

The network forensics tool is developed using the Java language version 1.4 on the Linux operating system. In addition the tool depends on pre existing tools and makes use of external libraries in order to implement its functionalities all of which are presented below.

1) *Tcpdump* [12]: Is a command line network traffic debugging tool. Tcpdump uses the libpcap packet capture library to capture all packets on the network by having the network interface card in the promiscuous mode and so requires root privileges to function. Expressions called Berkeley Packet Filters (BPF) can be used on the tcpdump captured data to filter out only the required packets which may be a small subset of the entire collected traffic data [4].

Output dump files of tcpdump forms the first of the primary information sources for data collection. The BPF filters are setup and applied by the mobile agents to retrieve the relevant packets as specified by the analyst's query.

2) *Snort* [13]: Is an open source IDS that can perform traffic analysis and can detect attacks such as buffer overflows, port scans etc... Snort can be used as a basic packet logger like tcpdump, but in this project it is used more as an intrusion detection system. Snort analyses the incoming network traffic in three phases to generate intrusion alerts. In the first pre processing phase, it identifies possible attack packets then in the second rules phase compares these packets to the signatures in the rule files for a possible match. If such a match is found then the post processing phase decides how to display the attack

alert.

In this project the snort's post processing phase is set to insert alerts to a database. The network forensics snort mobile agent then queries this database to obtain the alerts.

3) *Apache Web Server [14]*: Is an open source web server that provides HTTP services. In order to analyze the activity and performance of the server, Apache web server maintains a log of all access requests in the Access Log file and the errors in the Error Log file.

The current project uses the Access Log in the Common Log Format present in the remote system to gather log request records from and then display in the combined user interface. The actual log format of the log can be specified using the user interface.

4) *JADE [15]*: Is a middleware that enables development of peer to peer applications based on Agent paradigm wherein an Agent is a software concept that has a lifecycle that allows it to be autonomous, proactive and communicate with other agents [16]. Although agents can involve artificial intelligence algorithms to make them more intelligent, in this project it is used as a higher abstraction to the concept of objects in the development of distributed software. JADE is Foundation for Intelligent Physical Agents (FIPA) compliant and defines an agent platform with three mandatory agent services. The agent management system (AMS) agent can be used to control the lifecycles of other agents in the platform; the directory facilitator (DF) agent provides agent lookup service to all agents, the agent communication channel agent (ACC) provides default message oriented communication and IIOP for interoperability between different agent platforms [17]. An instance of JADE run-time is called a container which is a Remote Method Invocation (RMI) server object and can contain several agents. The agent platform has a GUI front end called Remote Monitoring Agent (RMA) and an associated main container and must always be started first. Every other container must register with the main container upon starting. A set of containers forming an agent platform can be distributed across a network and thus hides the underlying heterogeneity of the systems by emulating a single Java JVM [16].

Agents are identified by name and communicate using messages in Agent Communication Language (ACL). Agents communicate by formulating and sending individual messages to each other and can even have conversations using the interaction protocols that range from query-request protocols to negotiation protocols [17]. ACL message communication between agents in the same agent platform and within same agent container uses event signaling. Message communication between agents in the same agent platform but within different agent containers uses remote method invocation RMI. Message communication between agents in different platforms uses Internet Inter-ORB Protocol (IIOP).

Jade supports agent mobility feature by which agent code and execution state can be interrupted on a host where it is currently executing and asked to migrate to a different remote host and restart execution from that point in execution [16]. This is the important concept that enables distribution of processing load and scalability.

5) *JPCAP [18]*: Libpcap is an open source C library that provides an interface for real time packet capture of network traffic across the network. For Java applications, a similar interface is provided by jpcap library.

For capturing network traffic and later interpreting the stored network traffic information jpcap library is used in this project.

6) *Java (XML) Log Analyzer (JXLA) [19]*: Is a web log analyzer released under Apache-style license. The library is a command line tool that takes in an XML configuration file as input to parse the Access Log file and writes results to a custom XML file format.

During this project, the library was modified to take in a Document Object Model tree of configuration information to parse the Access Log and according to the specific forensic analyst's query and send back the result as Java objects. This modified library is used by the mobile agent to gather web server log information to be displayed on the user interface.

IV. SOFTWARE DESIGN AND ARCHITECTURE

A. Architecture

The software is developed using hybrid peer to peer agent based architecture [Fig. 1] where a special node acting as the server provides the service of lookup and discovery of other agents. This special node also hosts the stationary network forensics agent. Using the mobile agent technology, agents are dispatched from the special node to travel to each of the associated monitored systems. These mobile agents gather or process the relevant information according to the query at each individual system and return to the special node with the results.

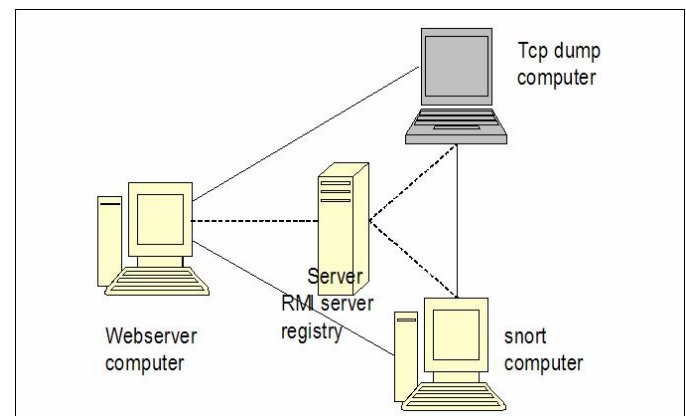


Fig. 1. Hybrid peer to peer architecture of the network forensics software.

The mobile agent platform used here is the JADE framework. The JADE RMA must be started first as the main container or the special node acting as the server for RMI server objects. The remote hosts containing tcpdump snort and web server software must each start a JADE container on each host and on start they register with the main container. Initially these host containers will be empty of any agents. On any of these hosts, the network forensics agent can be started in a container and on start they join the main container. Or while starting the main container the agent can also be started along with it. The agent level architecture for the application with the main container containing the network forensics GUI agent and the tcpdump, snort and web server hosts each with their own containers is given in Fig. 2.

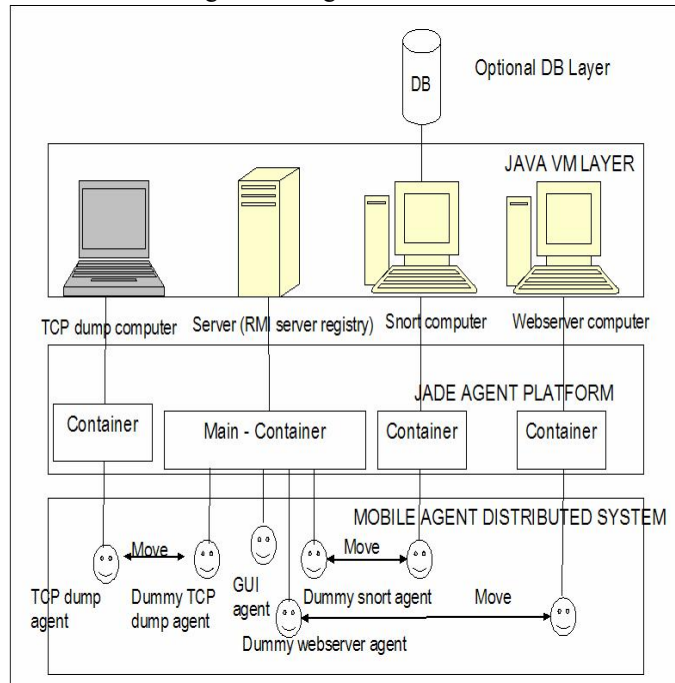


Fig. 2. Agent level network forensics software architecture.

If the network contains multiple web servers, snort systems and tcpdump configured to listen in promiscuous mode then this architecture can easily scale by simply creating as many multiple containers as there are multiple systems and multiple agents can be created to travel to remote systems and gather information to be displayed. This architecture does not overload a single system but distributes the processing load among the multiple machines. The results obtained from these multiple distributed systems can be correlated with one another in order to glean more intelligent analysis at a distributed global level and can help in analyzing attacks that involve multiple machines like the distributed denial of service attack.

B. Design

The design consists of the following different components.

- The user interface to specify the configuration of the remote sites with the forensics information sources.
- Query builder user interface to select the exact information to gather.
- The user interface to view the results obtained from the different sources.
- Individual agent design to move from the network forensics main GUI location to the target location to do the work of evidence gathering.
- Establish the message protocol for communication between the agents.

A high level view of the various components and their interaction is given in Fig. 3. The Display GUI is the special node with the network forensic agent where in the configuration; query and select criteria can be specified. The criteria are evaluated by the query processor to generate the query and start execution to obtain results. First the agents are created and then depending on the configuration information, the agents are dispatched to the respective host containers. Next the agents after moving to the remote location gather the results from the information source located in that host. Then the agent returns back to the host where the display GUI is present and updates the UI with the results that it brought back.

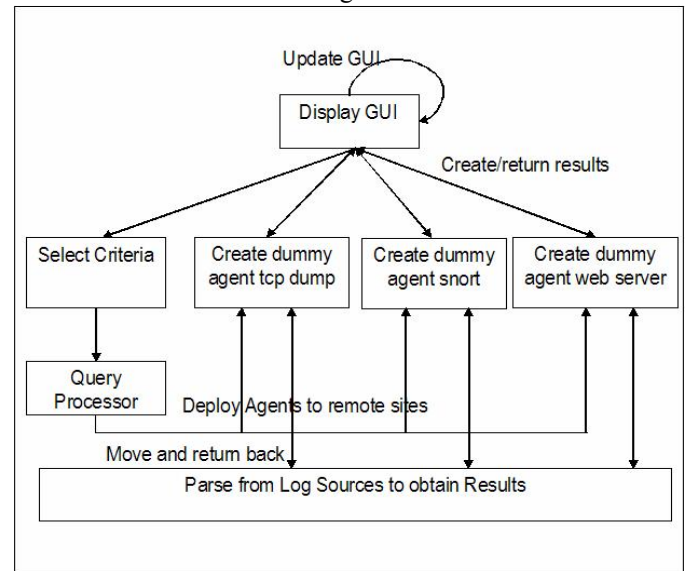


Fig. 3. High level design of network forensics software.

1) User Interface Design:

Java Swing is used to implement the user interface. The configuration, query specification and result form a separate tab in a tabbed pane. The configuration tab contains separate panels for each of Tcpdump, Snort and Web Server and Packet Capture. The query specification tab contains a query builder option which helps in specifying detailed criteria for selection of particular records.

The results are spread out in three tabs one each for Tcpdump packets, Snort alerts and Web Server access

records. The multiple entries matching the criteria given in the query are displayed in separate tabs panes. Each result displayed corresponds to one entry in the corresponding log. Navigation to the other result records is provided.

A menu for saving the results for future viewing is provided by serializing the result files into a suitable XML format.

2) Agent Design:

The agents in this application can be either stationary or mobile. The user interface is developed as a stationary agent. After the configuration information has been entered and the query is built, when the user presses the execute button, this GUI agent creates dummy agents responsible for each of the information sources called the Tcpdump Agent, Snort Agent and the Web Server Agent. Depending upon what agents the query calls for the GUI agent sends message to the particular agent. This message signals the agent to move to the remote system, gather results and return with results to update the result view in the user interface.

Each of the Tcpdump, Snort and Web Server Agent are mobile agents. With the receipt of the message request for gathering information from remote host systems, these individual agents know to which physical host they have to move. So the dummy agent now moves to the appropriate machine using the mobility feature of JADE to the target node and restarts execution at the remote node. After it gathers the correct information decided by the query, the individual agent moves back to the original location with the results. Once returned to the original location each individual agent sends a result message to the user interface agent to update the user interface to display the results asynchronously.

When the live network traffic capture option is selected, an agent that does the packet capture and stores to the file system is created. This agent is individually cloned to be started on the remote site for the entire time of packet capture. The mobile tcpdump agent then sends message to this traffic capture agent to retrieve the results. Packet capture technique is built on the libpcap component [20] and the Java packet capture agent is designed using the jpcap library for packet capture and storage of packet information to files in a tcpdump compatible format.

The method of obtaining the result from each information source varies depending on whether the source is in tcpdump libpcap format, snort database format or web server plain text format. Libpcap format file can be parsed using jpcap library. Snort when made to log to MySQL database sets up pre defined tables containing the alert information [21]. Using JDBC library and the MySQL database driver the database entries can be read and delivered as result. Web server log parser looks for specific regular expression patterns in the log file as given by the web server log format string.

3) Agent Communication:

Agents are identified by name; to communicate to one another, an agent sender sends a message to another agent receiver by specifying the message and receiver name. In this project, the agents communicate by sending messages which are objects and identify each agent by a predefined constant name and a variable instance local name. The various message exchanges amongst the agents are given in the following Fig. 4.

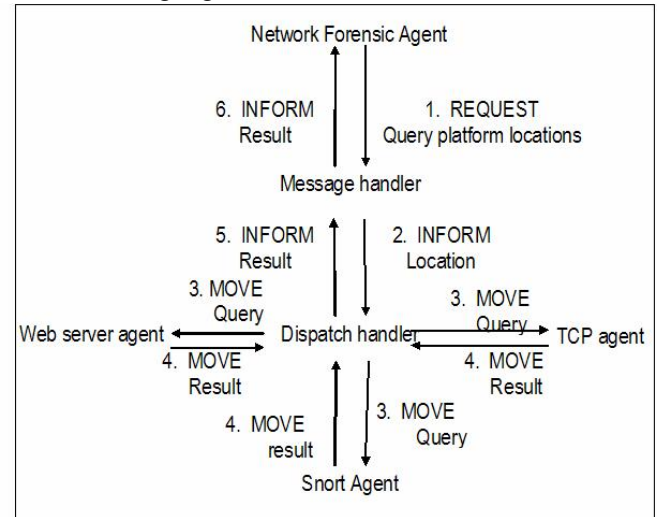


Fig. 4. Message Protocol for communication between agents.

Any request for container location is a REQUEST message and any result to a REQUEST message is a INFORM message according to ACL specifications. Message to request agents to move is the MOVE message. Notice that each message has an associated Result, Query or Location object.

C. Detailed Design

Agents are considered as a higher level of abstraction unit of software development. This tool is designed by emphasizing the software nature of an agent, communication between agents and uses the agent-based software development method, UML notation described in [22]. An agent which is an autonomous software unit is designed for each conceptual agent of the forensics tool. Behaviour of individual agent actions of sending a message, receiving a message is defined. Internal functionality that forms the set of activities and object manipulations performed by each agent are defined. This forms the mapping model of the business rules for the various functionalities of the distributed agent based network forensic tools.

1) Sequence Diagram Representation of Business Rules of network forensics tool

The business rules for this project can be represented by the sequence diagram in Fig. 5 and contains the lifecycle given below.

- Network forensics GUI executes a new query or asks to start packet capture

- If GUI agent has to execute a new query, it creates dummy agents for tcpdump, snort and web server; obtains the remote locations of the containers to which the dummy agents have to move and dispatches them. Else if GUI agent has to start packet capture then it creates a dummy packet capture agent and clones the agent to the remote system to start capturing there.
- The individual agents move to target locations, process the log information sources and only select the records that satisfy the query criteria. In case there is a packet capture agent on the remote system, the tcpdump agent

sends message to the packet capture agent requesting information and waits for the result reply from the packet capture agent. The agents return back with results.

- The individual agents present the results to the network forensics agent who updates the network forensics GUI.

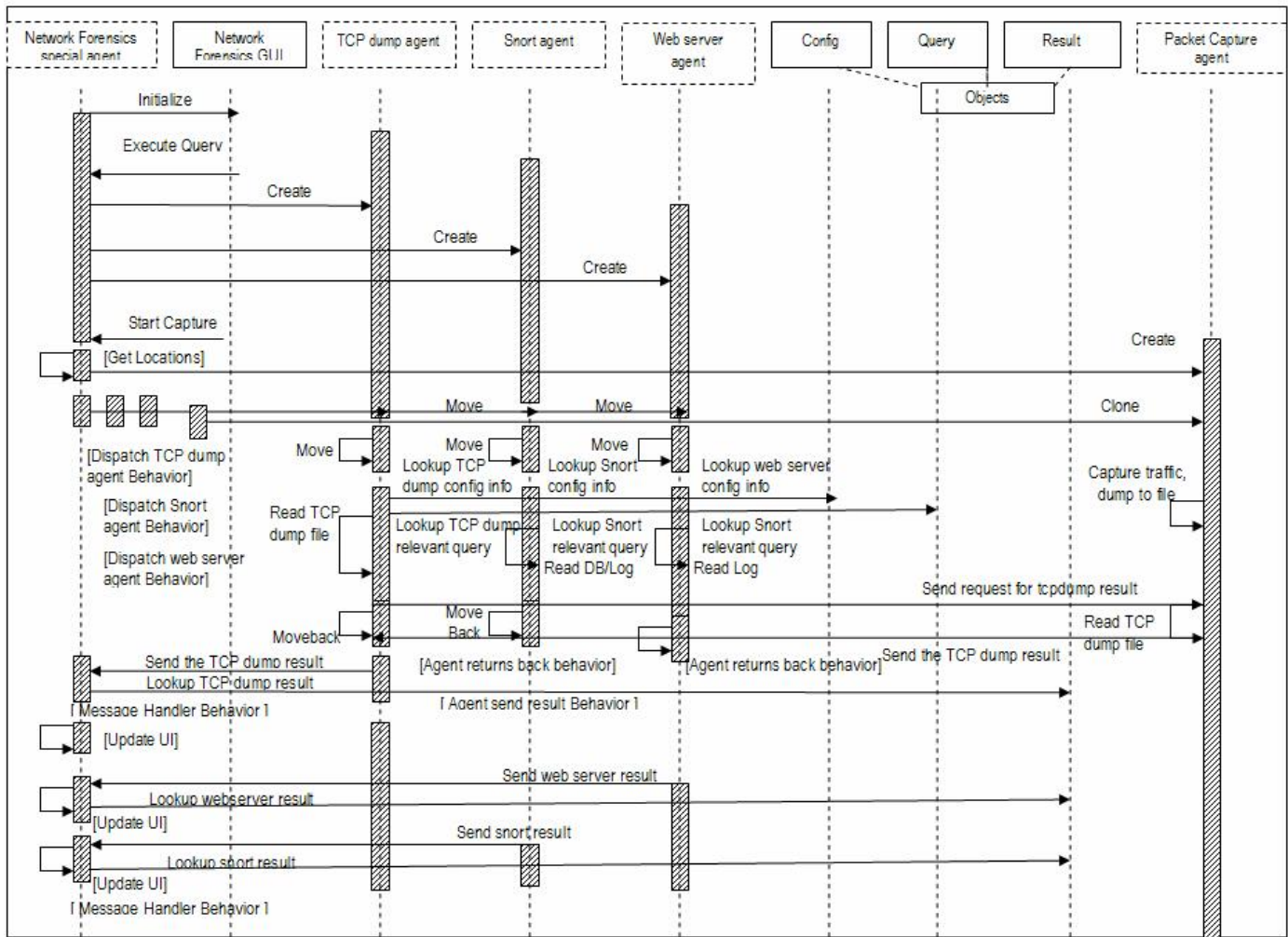


Fig. 5. Sequence Diagram.

2) Class Diagram

The class design using the business rules for this project is explained below.

- Network Forensics class is the stationary agent which is the combined GUI and the helper class for GUI event handlers is Network Forensics GUI class. Since an agent program runs on its own execution thread and handles its behaviours, when integrating a GUI with an agent writing custom GUI class is inefficient so uses the GuiAgent mechanism provided with Jade [23].

- The master slave pattern [24] is used such that the master agent Network Forensics class creates slave mobile agents that migrate.
- Mobile agents are designed as separate agent class named TcpdumpAgent, SnortAgent, and WebserverAgent and PacketCapture agent respectively.
- Agent functionality is defined in the Behaviour classes. MessageHandler is responsible for receiving messages to dispatch agents and update results. Dispatch Agent behaviour is responsible to send messages to agents to move to target locations. Update

Result behaviour is responsible for updating the GUI with results obtained. The SendResult Behaviour is implemented by each of the mobile agent to send the result obtained to the stationary agent. The packet capture agent functionality is in the Start Capture Behaviour which periodically invokes the PacketDumpToFile Behaviour to write the packets captured to a file. Packet Capture agent also has the

Message Handler Behaviour responsible for receiving tcpdump agent's message request for packets and delivering the result after reading the packets from the file.

The class design that make up all classes for the individual agents, agent behaviours, messages and plain objects is given in Fig. 6 using the notation in [24].

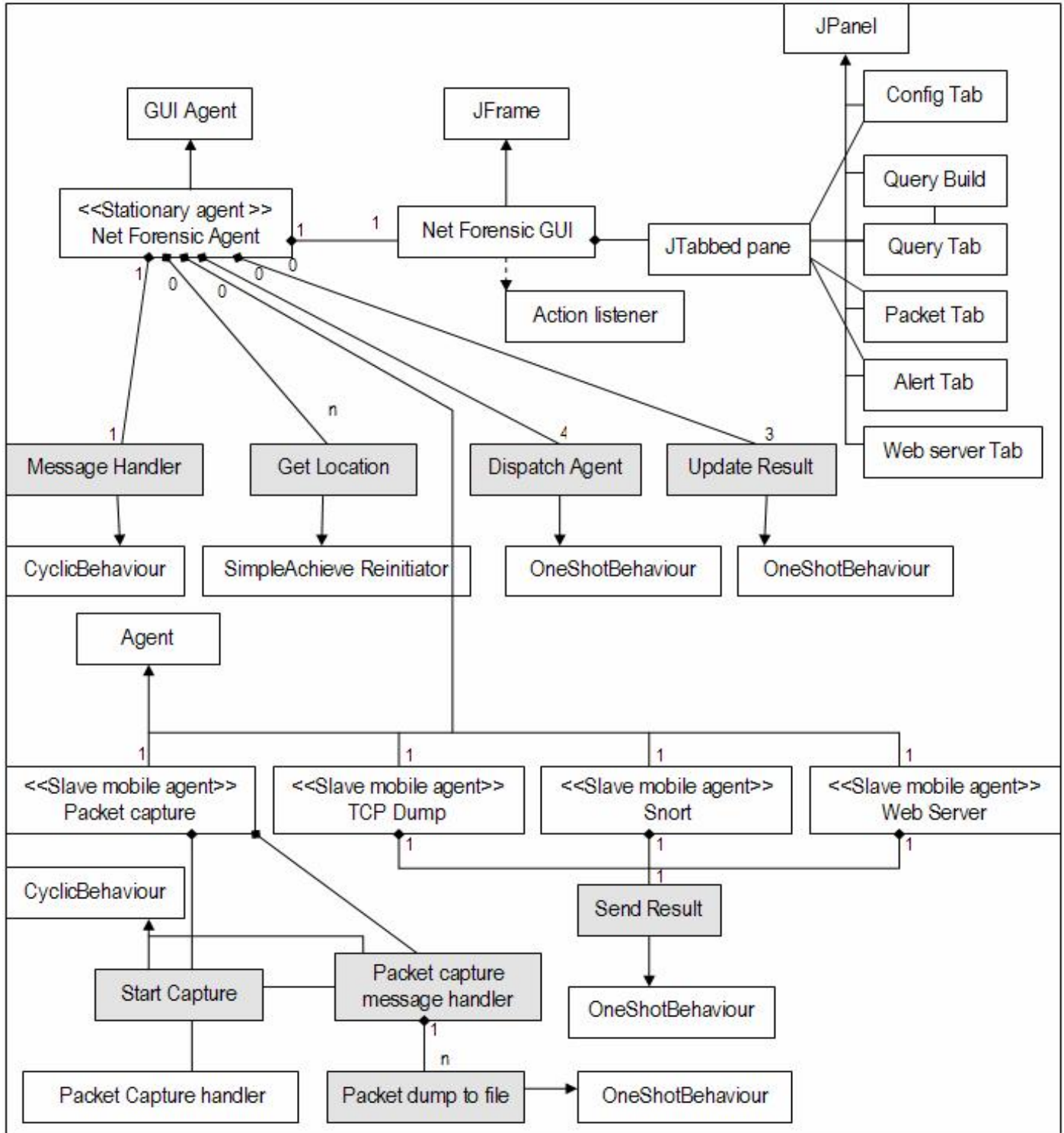


Fig. 6. Class Diagram.

V. VALIDATION AND RESULTS

A. Validation

The distributed network forensics software tool was developed considering validation at every stage of the software development cycle. During the requirements specification all the functionalities of the system, user interaction with the software, user interface look and feel, inputs and output expected, operating environment of the software were documented as a baseline requirements document. In the design phase, the requirement was translated into a high level representation of implementation by using object oriented analysis and design techniques by developing the data flow diagram [Appendix], class diagram and sequence diagram. This design was implemented during the coding phase and developer testing was done including both structural and functional testing to ensure that software functionality is correct. The different levels of testing carried out are presented below.

1) *Unit Testing*: During implementation new classes were created and procedures were implemented incrementally, each of them was tested by including print statements to standard output to test the input and output. Each component in the user interface classes was tested for correct action handling for each user input and click. The library JXLA was modified and tested for usability by this software.

2) *Integration Testing*: The software was developed incrementally by first developing a common query input and query component builder user interface (UI); then tcpdump related configuration UI, mobile agent and corresponding packet result display, followed by Snort related configuration UI, mobile agent and alert result display and finally web server related configuration UI, mobile agent and access result display. Finally the packet capture agent was developed and tcpdump agent was integrated with the packet capture agent to exchange messages in order to obtain the results. At every stage integration with the query specification and agent data collection was done and tested to ensure functionality correctness.

In addition, the software developed makes use of Apache-style license library modified by the author to fit the needs of the network forensics tool. The integration of the modified JXLA library to the network forensics software was tested.

3) *System Testing*: Using the requirements document as the baseline, all the functionality of the software was tested. The software was installed on the intended hardware systems with the required software. The agent special node agent that forms the server for all agents was

initially started on one of the systems. The network forensic tool's combined user interface was then invoked on one of the other systems, the configuration UI was used to fill in the configuration details of the system(s) from which the data has to be collected by the network forensic software along with the query of the actual content of data to be collected. All combinations of query specification and configuration specification were tested to see if the agents were dispatched to the configured systems correctly, if the data gathered agent return and finally data displayed were according to the requirement specification. The result saved as XML was validated.

4) *Exception Handling*: User input error(s), JADE framework specific error(s), configuration error(s), query specification error(s) and other exceptions during execution are properly caught and presented to the user. The same was tested by executing appropriate test cases to simulate the errors.

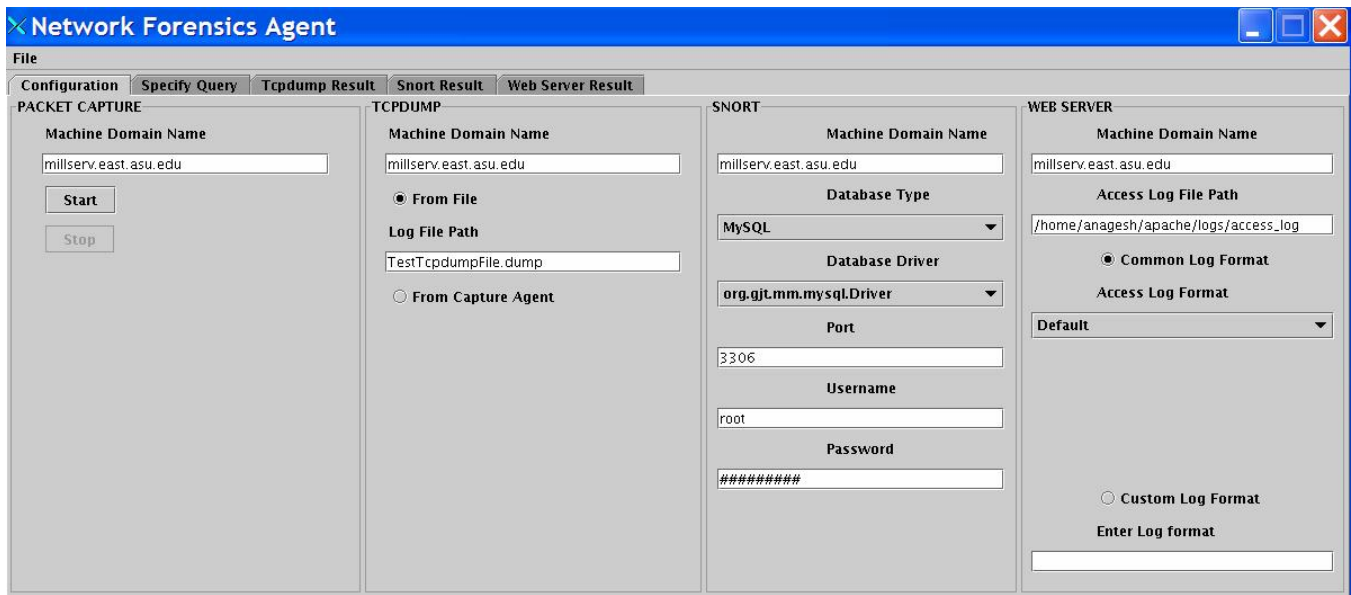
B. Results

1) Initial set up

The tool was used using two networked systems both with Linux operating system. The first system's network card was made to listen in the promiscuous mode and tcpdump, snort and web server were installed and configured to log in this system. Snort was configured to log to database by installing MySQL database. Jpcap library was installed and root permission was given to the account where Jpcap had to run in order to enable capture of packets. The main JADE container with the RMA was started on this system. Another machine was used to run the container with the user interface and create the dummy agents on. These agents move to the first machine to gather results and bring them back to be displayed.

2) Look and Feel

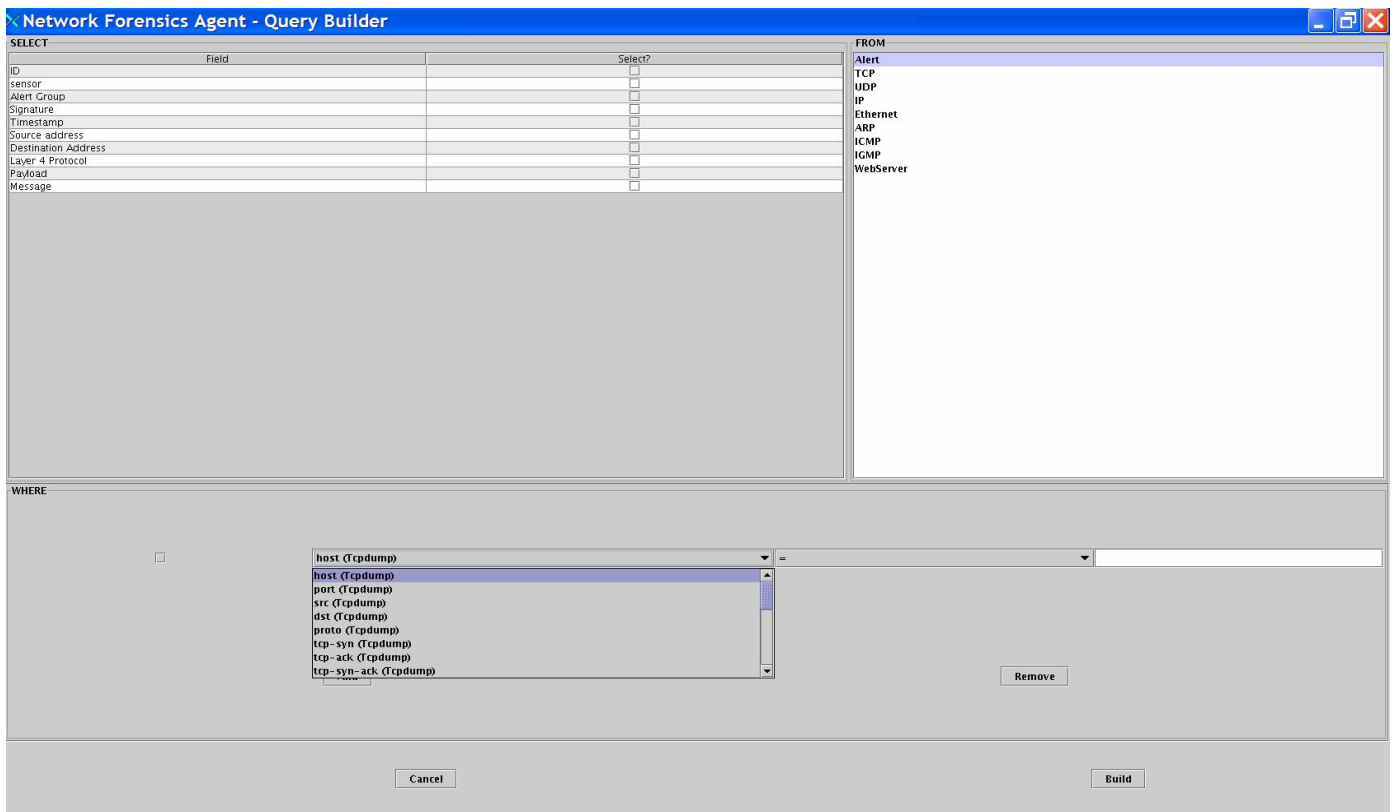
Given below are screenshots of the user interface of configuration and query builder of the network forensics software when executed [Fig. 7, Fig. 8]. The result views for each of tcpdump packet level [Fig. 9]; snort's alert level [Fig. 10] and web server's access level [Fig. 11] are shown. Also the query builder interface when only TCP packets are selected in the selection criteria [Fig. 12] and the packet level result obtained as a trimmed result of specific criteria in the query is shown in Fig. 13.



The configuration window is titled "Network Forensics Agent" and contains four main sections: PACKET CAPTURE, TCPDUMP, SNORT, and WEB SERVER.

- PACKET CAPTURE:** Includes a "Machine Domain Name" field with "millserv.east.asu.edu" and "Start" and "Stop" buttons.
- TCPDUMP:** Includes a "Machine Domain Name" field with "millserv.east.asu.edu", a radio button for "From File" (selected), a "Log File Path" field with "TestTcpdumpFile.dump", and a radio button for "From Capture Agent".
- SNORT:** Includes a "Machine Domain Name" field with "millserv.east.asu.edu", a "Database Type" dropdown set to "MySQL", a "Database Driver" dropdown set to "org.gjt.mm.mysql.Driver", a "Port" field with "3306", a "Username" field with "root", and a "Password" field with "#####".
- WEB SERVER:** Includes a "Machine Domain Name" field with "millserv.east.asu.edu", an "Access Log File Path" field with "/home/anagesh/apache/logs/access_log", a radio button for "Common Log Format" (selected), an "Access Log Format" dropdown set to "Default", and a "Custom Log Format" section with an "Enter Log format" field.

Fig. 7. Configuration user interface.



The query builder window is titled "Network Forensics Agent - Query Builder" and is divided into several sections for building SQL queries.

- SELECT:** A table with columns "Field" and "Select?". Fields include ID, sensor, Alert Group, Signature, Timestamp, Source address, Destination Address, Layer 4 Protocol, Payload, and Message.
- FROM:** A list of database tables including Alert, TCP, UDP, IP, Ethernet, ARP, ICMP, IGMP, and WebServer.
- WHERE:** A section for adding conditions. It shows a list of fields from the "host (Tcpdump)" table: host (Tcpdump), port (Tcpdump), src (Tcpdump), dst (Tcpdump), proto (Tcpdump), tcp-syn (Tcpdump), tcp-ack (Tcpdump), and tcp-syn-ack (Tcpdump). A "Remove" button is next to the list.
- Buttons:** "Cancel" and "Build" buttons are at the bottom.

Fig. 8. Query Builder User Interface.

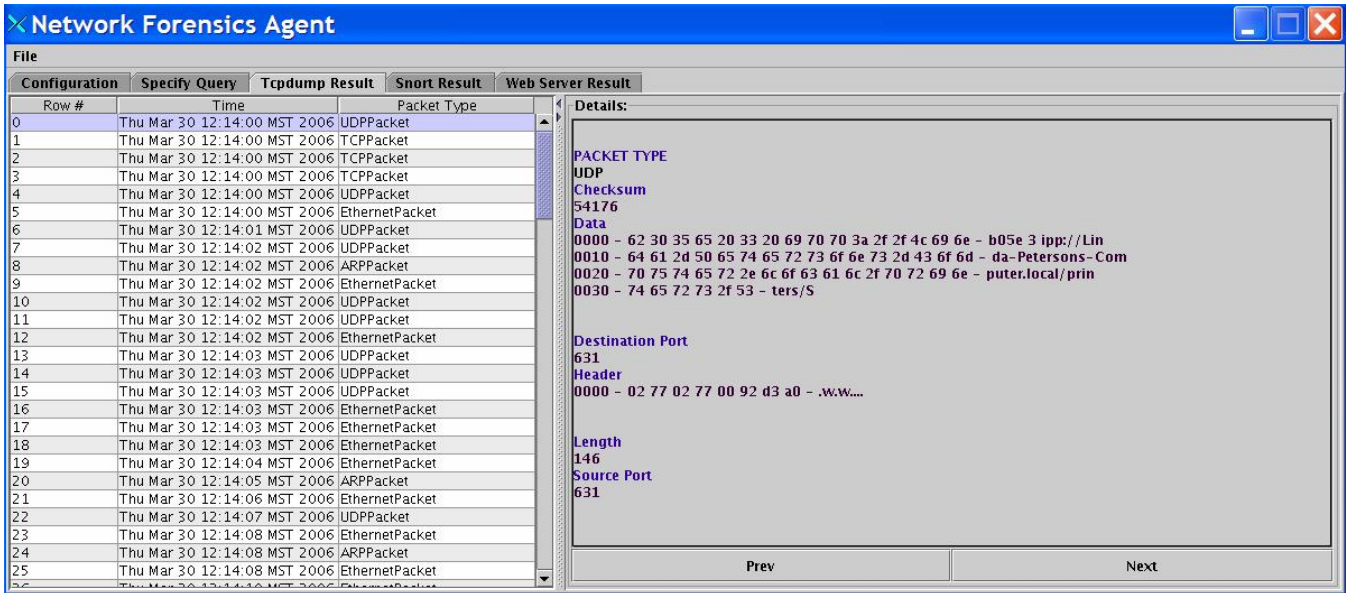


Fig. 9. Packet Level Result View.

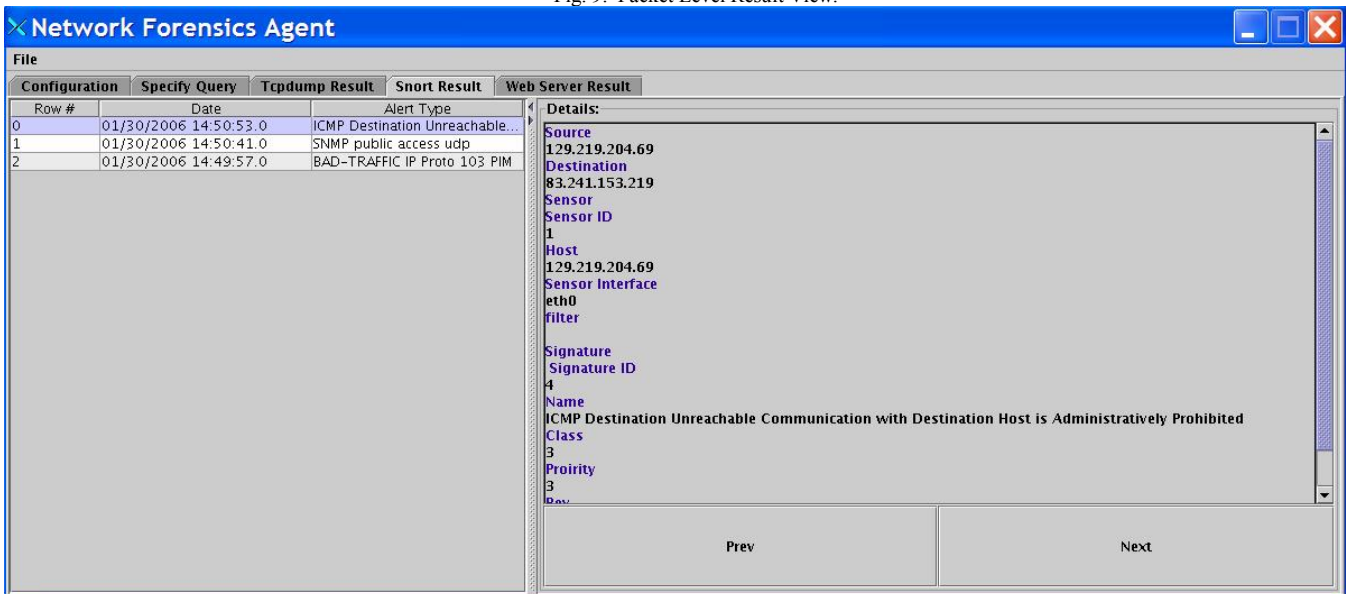


Fig. 10. Alert Level Result View

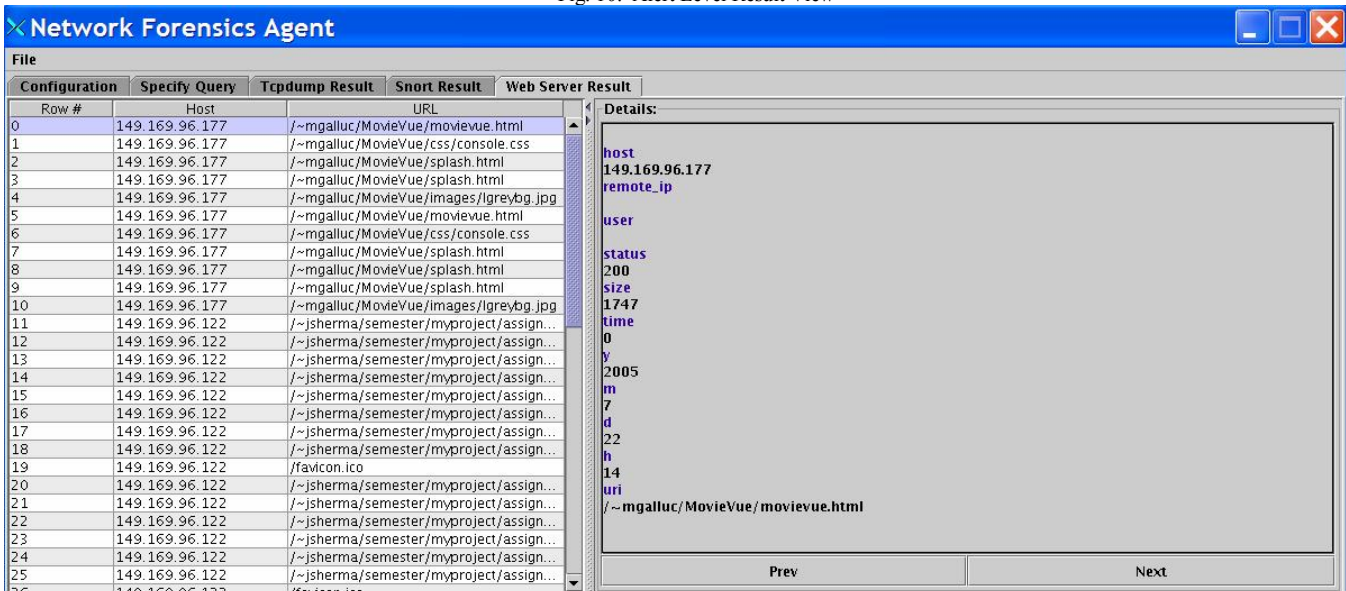


Fig. 11. Web server Access Level Result View.

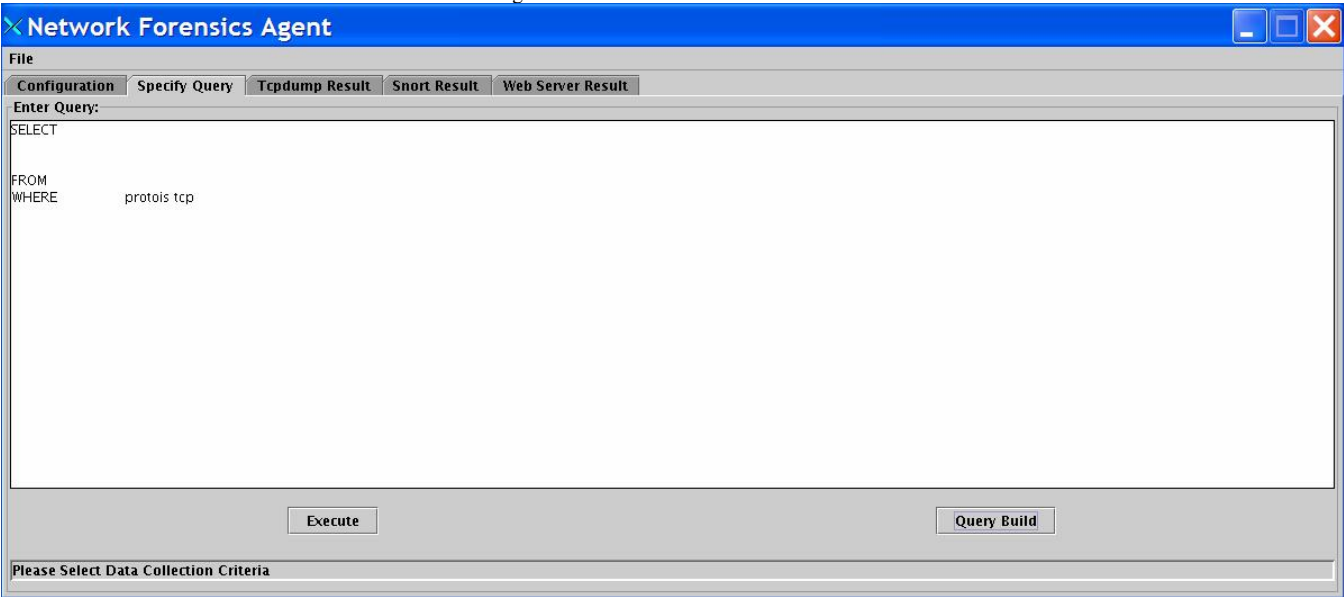


Fig. 12. Using the query builder, only TCP packets are requested to be collected from the tcpdump log.

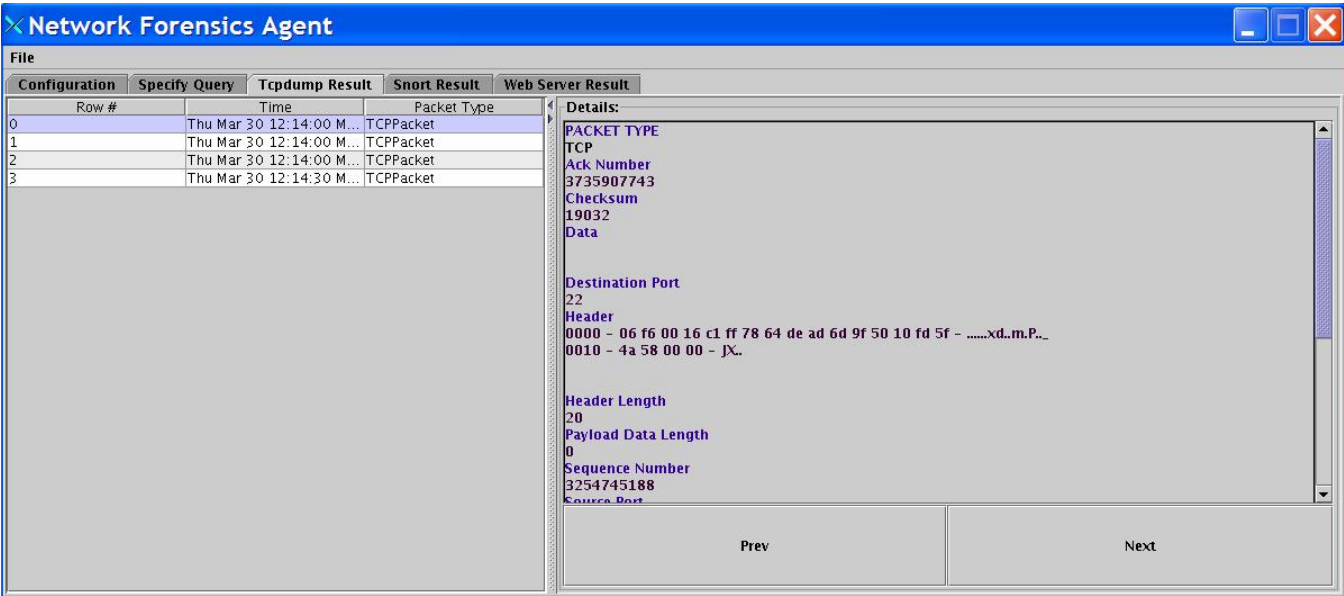


Fig. 13. The Packet Level result showing only TCP packets as a result of the above query with selection criteria applied.

3) Saving results

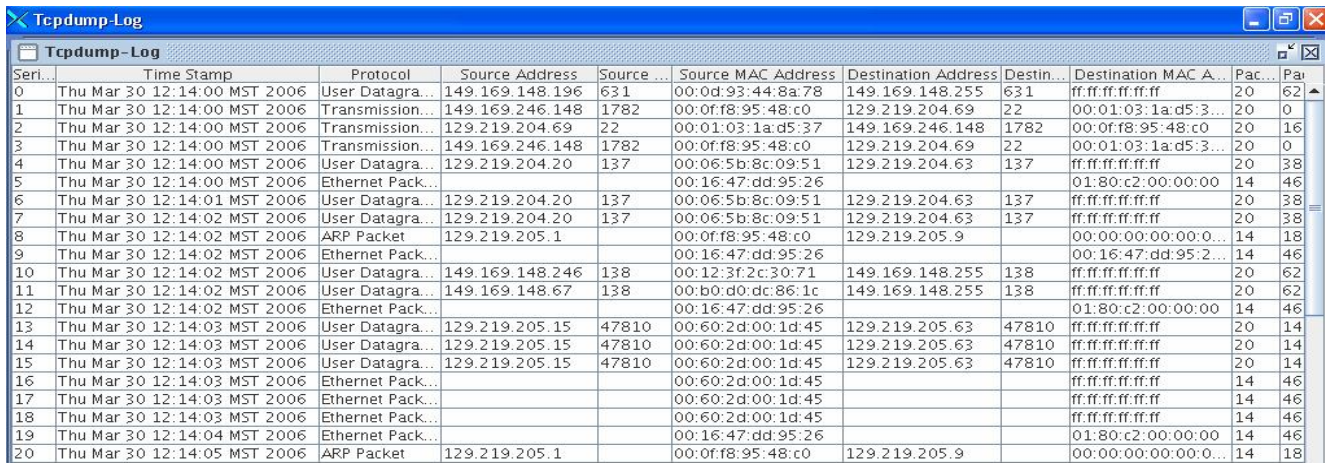
The results are saved as three separate XML files one each for tcpdump, Snort and Web server. The names of the files start with Tcpdump, AlertLog and AccessLog respectively appended with the year, month and day of that date. The XML format for Tcpdump AccessLog follow the format of LogTracker in order to enable further analysis and the AlertLog follows custom XML format.

4) Viewing saved XML result files using LogTracker

The tcpdump packet results and web server results are

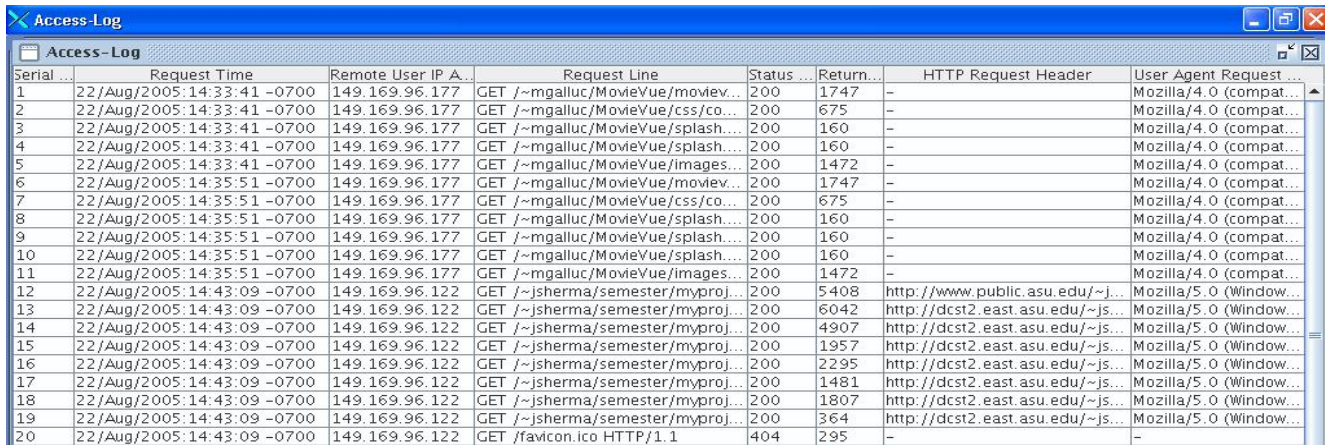
saved as XML and can be viewed using LogTracker [25]. The saved XML files should be copied to the LogTracker specific location and the LogTracker when invoked consumes the Tcpdump XML file and Web server XML file generated by the network forensics tool.

Screenshots of viewing the saved Tcpdump and Webserver result using the LogTracker shown in Fig. 14 and Fig. 15.



Serial...	Time Stamp	Protocol	Source Address	Source ...	Source MAC Address	Destination Address	Destin...	Destination MAC A...	Pac...	Pa...
0	Thu Mar 30 12:14:00 MST 2006	User Datagra...	149.169.148.196	631	00:0d:93:44:8a:78	149.169.148.255	631	ff:ff:ff:ff:ff:ff	20	62
1	Thu Mar 30 12:14:00 MST 2006	Transmission...	149.169.246.148	1782	00:0f:f8:95:48:c0	129.219.204.69	22	00:01:03:1a:d5:3...	20	0
2	Thu Mar 30 12:14:00 MST 2006	Transmission...	129.219.204.69	22	00:01:03:1a:d5:37	149.169.246.148	1782	00:0f:f8:95:48:c0	20	16
3	Thu Mar 30 12:14:00 MST 2006	Transmission...	149.169.246.148	1782	00:0f:f8:95:48:c0	129.219.204.69	22	00:01:03:1a:d5:3...	20	0
4	Thu Mar 30 12:14:00 MST 2006	User Datagra...	129.219.204.20	137	00:06:5b:8c:09:51	129.219.204.63	137	ff:ff:ff:ff:ff:ff	20	38
5	Thu Mar 30 12:14:00 MST 2006	Ethernet Pack...			00:16:47:dd:95:26			01:80:c2:00:00:00	14	46
6	Thu Mar 30 12:14:01 MST 2006	User Datagra...	129.219.204.20	137	00:06:5b:8c:09:51	129.219.204.63	137	ff:ff:ff:ff:ff:ff	20	38
7	Thu Mar 30 12:14:02 MST 2006	User Datagra...	129.219.204.20	137	00:06:5b:8c:09:51	129.219.204.63	137	ff:ff:ff:ff:ff:ff	20	38
8	Thu Mar 30 12:14:02 MST 2006	ARP Packet	129.219.205.1		00:0f:f8:95:48:c0	129.219.205.9		00:00:00:00:00:0...	14	18
9	Thu Mar 30 12:14:02 MST 2006	Ethernet Pack...			00:16:47:dd:95:26			00:16:47:dd:95:2...	14	46
10	Thu Mar 30 12:14:02 MST 2006	User Datagra...	149.169.148.246	138	00:12:3f:2c:30:71	149.169.148.255	138	ff:ff:ff:ff:ff:ff	20	62
11	Thu Mar 30 12:14:02 MST 2006	User Datagra...	149.169.148.67	138	00:b0:d0:dc:86:1c	149.169.148.255	138	ff:ff:ff:ff:ff:ff	20	62
12	Thu Mar 30 12:14:02 MST 2006	Ethernet Pack...			00:16:47:dd:95:26			01:80:c2:00:00:00	14	46
13	Thu Mar 30 12:14:03 MST 2006	User Datagra...	129.219.205.15	47810	00:60:2d:00:1d:45	129.219.205.63	47810	ff:ff:ff:ff:ff:ff	20	14
14	Thu Mar 30 12:14:03 MST 2006	User Datagra...	129.219.205.15	47810	00:60:2d:00:1d:45	129.219.205.63	47810	ff:ff:ff:ff:ff:ff	20	14
15	Thu Mar 30 12:14:03 MST 2006	User Datagra...	129.219.205.15	47810	00:60:2d:00:1d:45	129.219.205.63	47810	ff:ff:ff:ff:ff:ff	20	14
16	Thu Mar 30 12:14:03 MST 2006	Ethernet Pack...			00:60:2d:00:1d:45			ff:ff:ff:ff:ff:ff	14	46
17	Thu Mar 30 12:14:03 MST 2006	Ethernet Pack...			00:60:2d:00:1d:45			ff:ff:ff:ff:ff:ff	14	46
18	Thu Mar 30 12:14:03 MST 2006	Ethernet Pack...			00:60:2d:00:1d:45			ff:ff:ff:ff:ff:ff	14	46
19	Thu Mar 30 12:14:04 MST 2006	Ethernet Pack...			00:16:47:dd:95:26			01:80:c2:00:00:00	14	46
20	Thu Mar 30 12:14:05 MST 2006	ARP Packet	129.219.205.1		00:0f:f8:95:48:c0	129.219.205.9		00:00:00:00:00:0...	14	18

Fig. 14. Viewing the saved Packet Level result using Log Tracker.



Serial...	Request Time	Remote User IP A...	Request Line	Status...	Return...	HTTP Request Header	User Agent Request...
1	22/Aug/2005:14:33:41 -0700	149.169.96.177	GET /~mgalluc/MovieVue/moviev...	200	1747	-	Mozilla/4.0 (compat...
2	22/Aug/2005:14:33:41 -0700	149.169.96.177	GET /~mgalluc/MovieVue/css/co...	200	675	-	Mozilla/4.0 (compat...
3	22/Aug/2005:14:33:41 -0700	149.169.96.177	GET /~mgalluc/MovieVue/splash...	200	160	-	Mozilla/4.0 (compat...
4	22/Aug/2005:14:33:41 -0700	149.169.96.177	GET /~mgalluc/MovieVue/splash...	200	160	-	Mozilla/4.0 (compat...
5	22/Aug/2005:14:33:41 -0700	149.169.96.177	GET /~mgalluc/MovieVue/images...	200	1472	-	Mozilla/4.0 (compat...
6	22/Aug/2005:14:35:51 -0700	149.169.96.177	GET /~mgalluc/MovieVue/moviev...	200	1747	-	Mozilla/4.0 (compat...
7	22/Aug/2005:14:35:51 -0700	149.169.96.177	GET /~mgalluc/MovieVue/css/co...	200	675	-	Mozilla/4.0 (compat...
8	22/Aug/2005:14:35:51 -0700	149.169.96.177	GET /~mgalluc/MovieVue/splash...	200	160	-	Mozilla/4.0 (compat...
9	22/Aug/2005:14:35:51 -0700	149.169.96.177	GET /~mgalluc/MovieVue/splash...	200	160	-	Mozilla/4.0 (compat...
10	22/Aug/2005:14:35:51 -0700	149.169.96.177	GET /~mgalluc/MovieVue/splash...	200	160	-	Mozilla/4.0 (compat...
11	22/Aug/2005:14:35:51 -0700	149.169.96.177	GET /~mgalluc/MovieVue/images...	200	1472	-	Mozilla/4.0 (compat...
12	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	5408	http://www.public.asu.edu/~j...	Mozilla/5.0 (Window...
13	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	6042	http://dcst2.east.asu.edu/~js...	Mozilla/5.0 (Window...
14	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	4907	http://dcst2.east.asu.edu/~js...	Mozilla/5.0 (Window...
15	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	1957	http://dcst2.east.asu.edu/~js...	Mozilla/5.0 (Window...
16	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	2295	http://dcst2.east.asu.edu/~js...	Mozilla/5.0 (Window...
17	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	1481	http://dcst2.east.asu.edu/~js...	Mozilla/5.0 (Window...
18	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	1807	http://dcst2.east.asu.edu/~js...	Mozilla/5.0 (Window...
19	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /~jsherma/semester/myproj...	200	364	http://dcst2.east.asu.edu/~js...	Mozilla/5.0 (Window...
20	22/Aug/2005:14:43:09 -0700	149.169.96.122	GET /favicon.ico HTTP/1.1	404	295	-	-

Fig. 15. Viewing the saved Web server Level result using Log Tracker.

VI. CONCLUSION

Network forensics analysis requires tools that effectively look through massive amount of scattered data to gather relevant evidence. It is the author's belief that the software developed using the concepts and design of mobile agent distributed architecture described in this paper, when deployed, helps the forensic analysts achieve their goal. This project uses primarily the graphical interface, mobility, message communication and parallel behaviours of the agent platform to first, implement a user interface wherein the analyst can specify the data to be collected from distributed systems. Second, dispatch distributed agents to heterogeneous locations to gather data. Third, display the collected result in a format that is useful to analyze network events. Further the solution implemented during this project is scalable, portable, reduces network traffic and processing load on a single system, and is a novel way of using software agents for network security monitoring and analysis.

Some areas for future work would be to add more intelligence to each of the individual agents to achieve result correlation by using collaborative agent communication and provide more learned analysis using

graphical representations and reports. By utilizing all the system logs instead of only the network traffic logs a more global view of events can be obtained. Using data mining techniques, fuzzy logic or neural networks allow agents to be more autonomous and take initiative to proactively report or take corrective action to mitigate the attack. Mobile agents are compellingly applicable in distributed software development but production quality software that has to function reliably in an open network must meet very stringent security requirements. Hence an important extension to the current work would be to consider mobile agent security mechanisms such as authentication, authorization, encryption and others as in [26] to ensure that secure communication of agent, host security on which agent executes and agent protection against attacks such that the agent code and information sources themselves cannot be tampered. This can be done by developing security models or by evaluating the effectiveness of using the security plug-in JADE-S in JADE version 3.2 [15]. This project justifies the usage of agent oriented software development for distributed network security tools. In the future, a more production quality tool can be developed by considering the advantages and features made possible by software agent technology.

APPENDIX

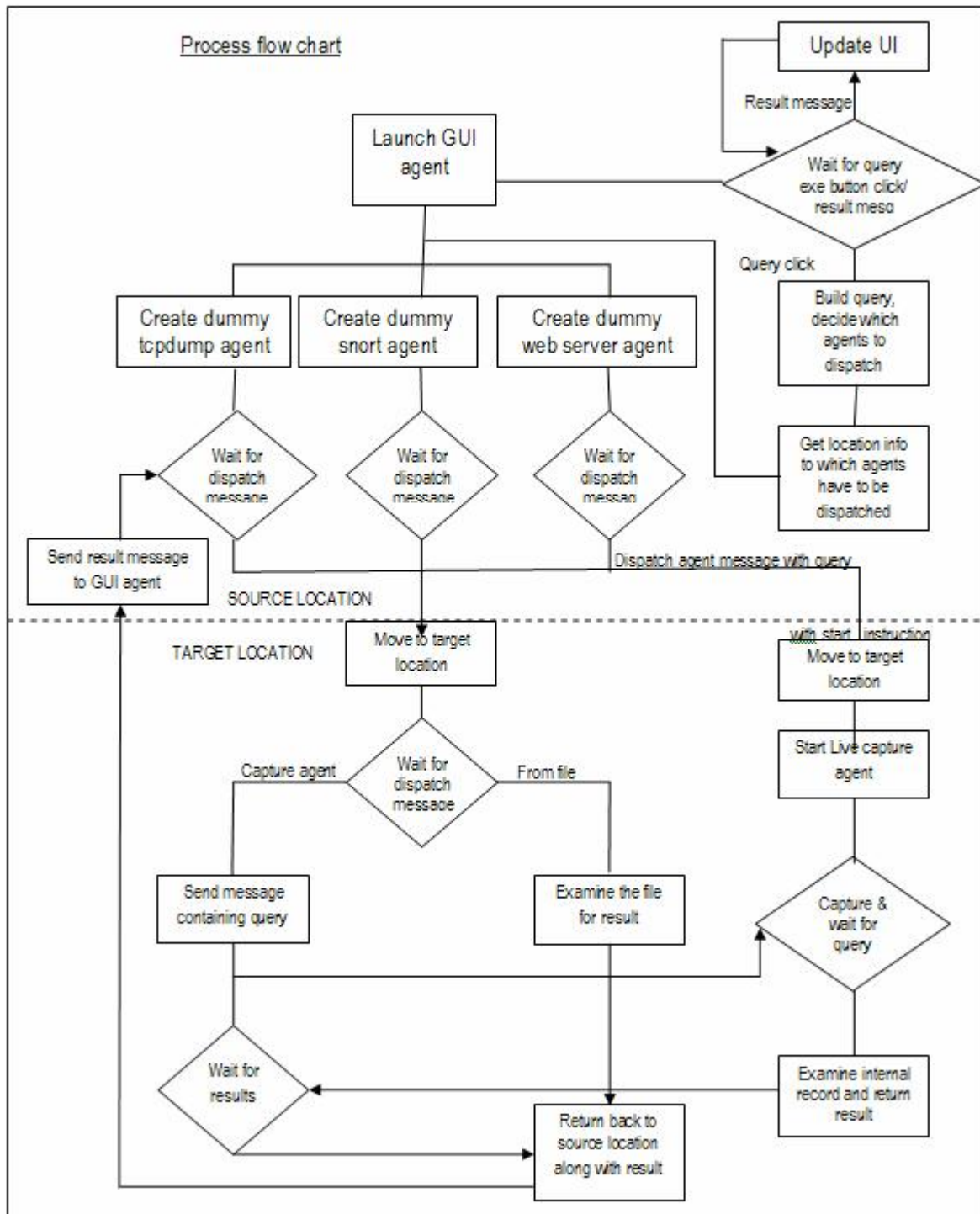


Fig. 16. Flow chart for validation

ACKNOWLEDGMENT

First the author would like to thank Dr. Bruce Millard the committee chair for his enthusiasm for the project idea, encouragement, support and useful insights. This project

would not have been possible without his patient guidance. Also, the author sincerely thanks Dr. Timothy Lindquist and Dr. Harry Koehnemann the committee members for their feedback and help with the project. I would also like to thank my loving husband, my parents and brother for

their constant support and encouragement.

REFERENCES

- [1] John D. Fernandez, Stephen Smith, Mario Garcia, Dulal Kar, "Computer forensics: a critical need in computer science programs", *Journal of Computing Sciences in Colleges*, Volume 20 Issue 4, April 2005.
- [2] Chris Prosise, Kevin Mandia, Matt Pepe, "Incident Response and Computer Forensic Second Edition", California, The McGrawHill Companies, 2003.
- [3] Corey, V., Peterman, C.; Shearin, S., Greenberg, M.S., Van Bokkelen, J., "Network forensics analysis", *Internet Computing, IEEE*, Volume 6, Issue 6, Nov.-Dec. 2002, Page(s): 60 - 66
- [4] Richard Bejtlich, "The Tao of Network Security Monitoring Beyond Intrusion Detection", Boston, MA, Pearson Education, Inc., August 2005.
- [5] Eoghan Casey, "Handbook of Computer Crime Investigation – Forensic Tools and Technology", San Diego, California, Academic Press, 2002.
- [6] Robert X. Cringely, (November 17, 2005), "Google-Mart", <http://www.pbs.org/cringely/pulpit/pulpit20051117.html>, (Jan 2006).
- [7] Biswanath L. Mukherjee, Todd Heberlein, and Karl N. Levitt, "Network Intrusion Detection", *IEEE Network*, vol. 8 no. 3, pp. 26-41, May 1994.
- [8] Phillip A. Porras and Alfonso Valdes, "Live Traffic Analysis of TCPIP Gateways," in *Networks and Distributed Systems Security Symposium*, March 1998.
- [9] S. Stolfo, A. Prodromidis, S. Tselepis, W. Lee, D. Fan, and P. Chan, "JAM: Java Agents for Meta learning over Distributed Databases", in *AAAI97 Workshop on AI Methods in Fraud and Risk Management*, 1997.
- [10] Kannadiga, P.; Zulkernine, M.; "DIDMA: a distributed intrusion detection system using mobile agents", *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference*, Page(s): 238 – 245, 23-25 May 2005.
- [11] Sandstorm Enterprises, Inc. (Jan 2006), "NetIntercept® 3.0 Data Sheet", <http://www.cv-data.com/pdf/ni-3-0-datasheet.pdf> (Jan 2003 - Jan 2004)
- [12] Jacobson, V., Leres, C., and McCanne, S. "TCPDUMP Documentation" 16 Dec 2005. <http://www.tcpdump.org/> (Jan 2006)
- [13] Snort, (Oct 2005), <http://www.snort.org/>, (Jan 2006).
- [14] Apache HTTP Server Project, 2005, <http://httpd.apache.org/>, (Jan 2006)
- [15] "Jade 3.2 Released", July 2004, <http://jade.tilab.com/news-art.php?id=27>, (Jan 2006)
- [16] Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa, "JADE - A FIPA-compliant agent framework", *Proceedings of PAAM'99*, London, April 1999, pp.97-108.
- [17] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, (January 2006), "Jade A White Paper", <http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf> (September 2003)
- [18] Jpcap network packet capture library, (September 2004), <http://jpcap.sourceforge.net/>, (January 2006)
- [19] Java (XML) Log Analyzer, April 6, 2005, http://sourceforge.net/project/showfiles.php?group_id=38386, (Jan 2006)
- [20] Mike D. Schiffman, "Building Open Source Network Security Tools – Components and Techniques", Indianapolis, Indiana, Wiley Publishing Inc., 2003.
- [21] Rehman, Rafiq Ur. "Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID", Upper Saddle River, N.J., Prentice Hall PTR, 2003.
- [22] Margus Oja, Boris Tamm, and Kuldar Taveter, "Agent-based Software Design", *Proceedings of the Estonian Academy of Sciences. Engineering*, Volume 50 No. 1 March 2001, pp 5-21
- [23] JADE Graphical Interfaces, April 2004, <http://www.iro.umontreal.ca/~vaucher/Agents/Jade/Gui.html>, (Jan 2006)
- [24] Emerson F. A. Lima, Patrícia D. L. Machado, Flávio R. Sampaio and Jorge C. A. Figueiredo, "An Approach to Modeling and Applying Mobile Agent Design Patterns", *ACM Software Engineering Notes*, May 2004 Volume 29 number 4.
- [25] Rupashree V, "Log Tracker", *Arizona State University, Division of Computing Studies*, MS Project, February 2006.
- [26] Yang Kun, Guo Xin, Liu Dayou, "Security in mobile agent system: problems and approaches", *ACM SIGOPS Operating Systems Review*, Volume 34 Issue 1, January 2000.