Journal of Artificial Intelligence
ISSN: 2229–3965 & E-ISSN: 2229–3973, Volume 3, Issue 1, 2012, pp.-42-48
Available online at http://www.bioinfo.in/contents.php?id=71

THE INTELLIGENT INTRUSION DETECTION TOOL FOR BIOMETRIC TEMPLATE STORAGE

MAITHILI ARJUNWADKAR[1]* AND KULKARNI R.V.[2]

[1]MCA, Modern College of Engineering, Pune, India.
[2]SIBER, Kolhapur, India.
*Corresponding Author: Email- maithili.arjunwadkar@gmail.com

Received: January 19, 2012; Accepted: February 13, 2012

Abstract- While there are various advantages of biometric authentication process, it is vulnerable to attacks, which can decline its security. To enhance the security of biometric process, Intrusion detection techniques are significantly useful. In this paper, we have designed intelligent agent as knowledge based Biometric Template storage Intrusion Detection tool. This intelligent agent can be located on the Biometric Template storage database. It performs intrusion detection using Operating System's audit trail, and RDBMS audit trail. The system consists of a user interface module, an inference engine, a knowledgebase of illegal transactions and audit trail of ORACLE database. Inference engine is implemented using JESS which is a Java based Expert System
Keywords- Biometric template, intelligent agent, multiagent system,Java Expert System Shell(JESS), database audit_trail.

Citation: Maithili Arjunwadkar and Kulkarni R.V. (2012) The Intelligent Intrusion Detection Tool for Biometric Template Storage. Journal of Artificial Intelligence, ISSN: 2229–3965 & E-ISSN: 2229–3973, Volume 3, Issue 1, 2012, pp.-42-48.

Introduction

Biometric authentication systems are used in order to verify the claimed identity of a user based on his/her biometric characteristics. A reliable identification system is a critical component in several applications that contribute their services specifically to genuine users. Examples of such applications include physical access control to a secure facility, e-commerce, access to computer networks, attendance mark etc.

A Biometric sample, commonly referred to as a "corpus" is the extraction of a unique set of data from an individual. Almost all biometric systems perform in the same basic manner. The biometric authentication systems are used either in centralized or distributed architecture. They mostly differ by how the processing steps for biometric authentication system are divided between different machines. A biometric system is essentially a pattern recognition system that operates by capturing a biometric image from an individual, extracting a unique feature set from the acquired data, and comparing this feature set against the template set in the database. A system database consisting of biometric templates must be created through a process of enrollment. The user then comes back to the same system that he has enrolled on and tries to authenticate through that stored template. The extracted components are run through an algorithm and stored as a template. Biometric Templates contain very sensitive information used to identify people which are bound to them. It is the template that is used to determine the user's rights and privileges to access that resource. Each individual's reference template must be stored in an accessible repository which can be compared to the user's biometric. It is commonly accepted that a biological template cannot be reverse engineered to create a complete biometric input of a user such as creating a complete fingerprint from the biometric input. In this paper we consider attacks on the biometric templates stored in the system database. Attacks on the template can lead to the vulnerabilities like insertion of a fake template, modification of an existing template, removal of an existing template, and replicate the template which can be replayed to the matcher to gain unauthorized access. We have made an attempt to develop intelligent tool which assists in detection of vulnerabilities.

In this paper we propose a Knowledge-based intrusion detection assistant for intrusions in biometric template storage. The detailed architecture of assistant for intrusions in biometric template storage and implementation of assistant is discussed. In this method we define the rules to detect attacks on biometric template storage and using back tracing, one can find source of intrusion.

Overview of Concept

A system that stores biometric template centrally is far more susceptible to massive disclosure than storage on a single PC or smart card, or device. One of the most potentially damaging attacks on a biometric system is against the biometric templates stored in the system database. The location of biometric storage is a key point in the consideration of controls. The biometric template store can be located remotely within a Central Repository, a Local Storage within the Biometric Reader Device, or on a portable token such as smart card. Each of these locations is appropriate for different systems, depending on the requirements. The privacy risks grow multifold for central storage locations. In this paper we consider biometric template stored within a Central Repository. Central repositories allow users to enroll at a central location and be recognized at any networked biometric device. Central repositories allow for easy auditing of authentication attempts [1].

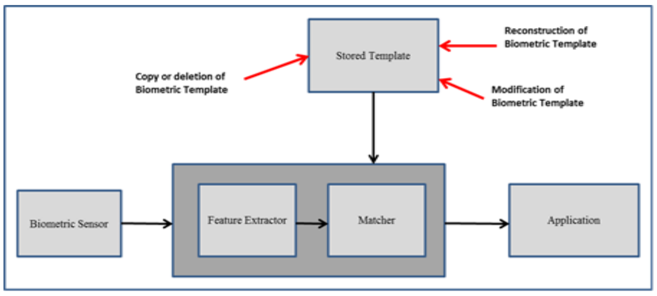

Reconstruction of Biometric Template
Stored Template
Copy or deletion of Biometric Template
Modification of Biometric Template
Biometric Sensor
Feature Extractor
Matcher
Application

Fig.1- Vulnerabilities in a Biometric Storage Template [2]

According to common criteria of biometric evaluation methodology supplement [3] it is particularly important to consider that attacks can be done on the direct input and output of a biometric template. The vulnerabilities in biometric storage are shown in (fig. 1). The Attacks can be done by authorized or unauthorized users. The users abuse of their rights and privileges to do unauthorized activities and to obtain unauthorized access. In this paper we consider two main categories of users those are normal user and user with DBA role that, intentionally or unintentionally damage the system [4]. One of the most potentially damaging attacks on a biometric system is against the biometric templates stored in the system database. As an example, a fingerprint template copied from a bank's database may be used to search a criminal fingerprint database or crosslink to person's health records. The imposter can access centrally stored template to avail unauthorized service whereas a lawful user will face Denial of service. To avoid this, smart cards are preferred. In that case, the template is stored in write once mode and erased or destroyed if altered. When this scenario is not an option, strong security controls or protection schemes must protect the template. The intrusion detection is an essential supplement of traditional security system. As per literature review [15], various Intrusion detection and prevention techniques are used in network systems, computer systems, web systems etc. But this type of technique is not available in biometric process. But detection of intrusion and prevention techniques to avoid such type of intrusion has become of paramount importance. This security system needs the robust automated auditing, intelligent reporting mechanism and robust detection and prevention techniques.

Every large scale biometric solution requires a RDBMS for efficient storage and access of data. A Biometric Template can be stored in a table column as:

- RAW data type
- Simple Object data type
- XML data type
- Full Common Biometric Exchange File Format-compliant (CBEFF) data type

In this paper we consider Biometric Template stored in the form of RAW data type.

Create a table to store employee data along with their fingerprint template

```
CREATE TABLE Employees
(name VARCHAR2(50),
employee_id VARCHAR2(10),
dept VARCHAR2(20),
fingerprint_template RAW(1024));
```

A security attack or intrusion can be defined like any action or set of actions which can violate the security of a system and tries to compromise the confidentiality. Intrusion detection is a general term which refers to the ability of a computer system to analyze the security-relevant events that violate security occurred in the past [5 -6].

Intrusion detection requires a number of security-relevant events which are collected and recorded in order to be analyzed. The role of an intrusion detection system (IDS) is to monitor system activities to detect malicious actions and to identify unauthorized and offensive uses [7].

It offers a defense when system vulnerabilities are exploited and does not require expensive equipments. The IDS is executed in background. When it detects suspicious or illegal activities, it notifies to the security administrator [8]. For detecting intrusive activities, IDS can use audit file data. In this paper we consider Distributed HOST-based IDS which are in charge of monitoring several hosts. It performs intrusion detection using Operating System's audit trail, RDBMS audit trail or information from multiple monitored hosts. In a distributed system, the audit data collection is ensured by several audit mechanisms which require a comparison of the audit records of the various components and a coordination of the analysis of the different hosts. In a distributed environment, the data collection and analysis can be done using a central repository, or distributed repositories with several analyzers. But, coordination between the various analyzers is necessary for producing systemwide audit information. The heterogeneity of a distributed network multiplies the vulnerabilities of the various systems, contrary to a centralized system.

The concept of Distributed Artificial intelligence (DAI) was defined, at the beginning of the Seventies, to find solutions to specific AI problems. Traditional AI concept deliberates intelligence within a single system. This involves some difficulties because of the need for integrating, within a same base of knowledge, expertise, competencies and knowledge of different individuals who, in reality, communicate and collaborate in the realization of a common goal [8]. The purpose of DAI is to extend the AI field in order to distribute the intelligence among several agents not subject to a centralized control. The multi-agent system is a system that consists of multiple agents that can interact together to learn or to exchange experiences jointly to take actions or to solve problems. The agent is a program module that functions continuously in a particular environment. It is able to carry out activities in a flexible and intelligent manner that is responsive to change in the environment (real or virtual). An agent is able to learn from its experiences. Different Agents deployed on different locations are shown in (fig.2).

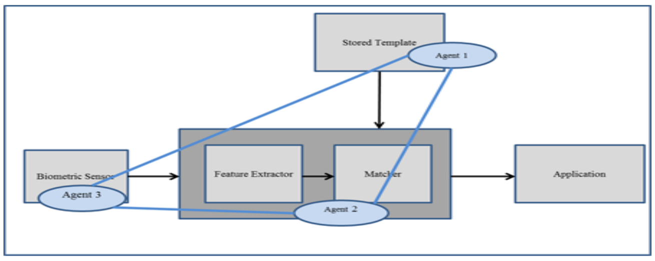


Fig.2- Different Agents deployed on different locations.

This autonomous agent takes actions based on its built-in knowledge and its past experiences. We consider the simple reflex agent distinguish the input from their environment i.e. DBA audit trail and interpret it to a state that matches the rules. This approach consists in detecting intrusions exploiting well-known system vulnerabilities. It is based on the fact that any known attack produces a specific trace in the audit trail or in the network data. It needs *a priori* information about the attacks it is able to detect. An alarm is raised when the trace of an attack is detected in the current audit trail or network data [9]. This approach works as follows:

- Attacking scenarios are collected,
- These scenarios are translated into facts using some predefined rules.
- Extracted knowledge is utilized to take some decision, an alarm can be raised.
- Using backward chaining approach source of intrusion can be found out.

For this approach we use Rule-based method (expert systems) which translates attacks that are collected through DBA audit trail of ORACLE. The current data are compared with the predefined rules. If a rule matches, knowledge is created and an alarm is raised. The construction of these rules depends entirely on the expertise of the security officer.

In this research we designed and developed one of the agents of multi-agent system called Biometric Template storage Intrusion Detection Assistant. Its customized dashboards deliver real-time compliance status and produce clear, easy-to-read reports for auditors and other stakeholders. Automatic security content updates target specific vulnerabilities and are acquainted with unknown exploits and take preventive action. This intelligent agent is located on the Biometric Template storage database. And user can see detail of illegal transaction like name of the user whose actions were audited (USERNAME),operating system login username of the user whose actions were audited (OS_USERNAME), client host machine name (USERHOST), Numeric ID of each ORACLE session (SESSIONID), Name of the object affected by action (OBJ_NAME),creator of the object affected by the action(OWNER), Timestamp of the creation of the audit trail entry in Coordinated Universal Time (UTC) zone (EXTENDED_TIMESTAMP) using backward chaining approach.

The biometric template storage Intrusion detection assistant is designed and implemented using JESS (Java Expert System Shell). JESS architecture shown in **(fig.3)**. Jess has been used to develop a broad range of commercial software, including:

- Expert systems that evaluate insurance claims and mortgage applications
- Agents that predict stock prices and buy and sell securities
- Network intrusion detectors and security auditors
- Design assistants that help mechanical engineers
- Smart network switches for telecommunications
- Servers to execute business rules
- Intelligent e-commerce sites
- Games

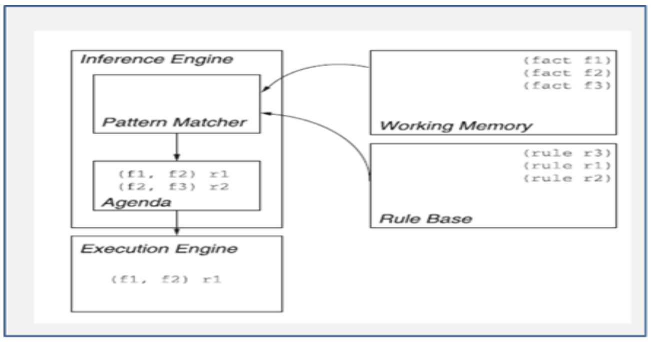


Fig.3- JESS architecture Diagram

The pattern matcher applies the rules in the rule-base to the facts in working memory to construct the agenda. The execution engine fires the rules from the agenda, which changes the contents of working memory and restarts the cycle.

JESS is a clone of the popular expert system shell CLIPS, rewritten entirely in Java which integrates Java Object manipulation with rule based inference. It is a rule engine - a special kind of program that very efficiently applies rules to data. A rule-based program can have different rules, and JESS will continually apply them to data in the form of a knowledge base (facts). Often the rules will represent the heuristic knowledge of a human expert in some domain, and the knowledge base will represent the state of an evolving situation (an interview, an emergency). JESS supports both forward and backward reasoning [10-11-12].

Proposed System

Architecture of Biometric Template storage Intrusion Detection Assistant is shown in (fig.4). Our architecture consists of a user interface module, an inference engine, a knowledgebase of illegal transactions and audit trail of ORACLE database. Auditing is the monitoring and recording of selected user database actions. It can be based on individual actions, such as the type of SQL statement executed, or on combinations of factors that can include user name, object, time, and so on.

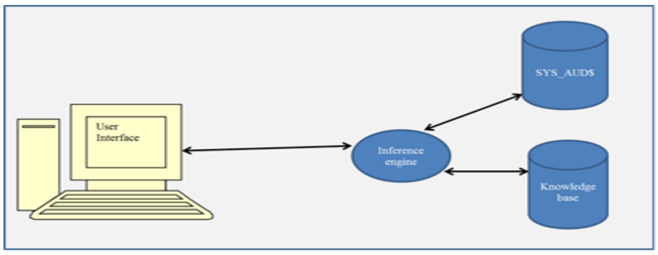


Fig.4- Architecture of Biometric Template storage Intrusion Detection Assistant

Security policies can trigger auditing when specified elements in an Oracle database are accessed or altered, including the contents within a specified object. The user interface module interacts with users. If user wants to find source of intrusion then user can select any illegal transaction, details of that transaction like, name of the user whose action were audited (USERNAME), operating system login username of the user whose actions were audited (OS_USERNAME), client host machine name (USERHOST), Numeric ID of each ORACLE session (SESSIONID), Name of the object affected by action (OBJ_NAME), Timestamp of the creation of the audit trail entry in Universal Time Coordinated (UTC) zone (EXTENDED_TIMESTAMP) will display on the screen. It exhibits five different tables which display information about DBA users, and other users, count of attacks from each user, count of attacks done from particular hosts and count of attacks by host who login as DBA. It also displays different graphs which depict how many times other users tried to insert a fake template, modify an existing template, remove an existing template, and copy the template which can be replayed to the matcher to gain unauthorized access based upon expert opinion. The Audit records include information such as the operation that was audited, the user performing the operation, and the date and time of the operation. Audit records can be stored in either a data dictionary table, called the database audit trail. The database audit trail is a single table named SYS.AUD$ in the SYS schema of each Oracle database's data dictionary. Several predefined views are provided to help you use the information in this table. The audit trail records can contain different types of information, depending on the events audited and the auditing options set. The information which is always included in each audit trail record regarding particular audit action is, the user name, session identifier, terminal identifier, name of the schema object accessed, and the operation performed or attempted so on. Database auditing is enabled and disabled by the AUDIT_TRAIL initialization parameter in the database's initialization parameter file. The parameter can be set by altering system using following parameters.

```
Audit_trail=DB   Scope=SPFILE
```

Audit_trail= DB

Enables database auditing and directs all audit records to the database audit trail (SYS.AUD$), except for records that are always written to the operating system audit trail.

Scope=SPFILE

uses the SCOPE clause because the database instance had been started using a server parameter file (SPFILE). Starting the database with a server parameter file is the preferred way of starting a database instance.

To control the Oracle auditing subsystem using system commands such as:

```
AUDIT SELECT,INSERT,UPDATE,DELETE ON
<object_name>
BY ACCESS WHENEVER SUCCESSFUL;
```

The AUDIT command only turns auditing options on; it does not enable auditing as a whole. To turn auditing on and control whether Oracle generates audit records based on the audit options currently set, set the parameter AUDIT_TRAIL in the database's parameter file. The database audit trail is a single table named SYS.AUD$ in the SYS schema of each Oracle database's data dictionary. Several predefined views are provided to help you use the information in this table, such as DBA_AUDIT_TRAIL. We use the view DBA_AUDIT_TRAIL, which displays all audit trail entries for suspicious database activities. The database is accessed using the JDBC (Java Database Connection). A suspicious knowledge is stored as a form of acts and rules in a JESS knowledge base. It is somewhat similar to a relational database, especially in that the facts must have a specific structure. A rule-based system maintains a collection of knowledge nuggets called facts. This collection is known as the knowledge base. It is somewhat akin to a relational database, especially in that the facts must have a specific structure. Similar to object-oriented languages, objects have named fields in which data appears; unordered facts offer this capability (although the fields are traditionally called slots.) We use unordered facts because they are structured in nature. We collect suspicious data from DBA_AUDIT_TRAIL and create different templates as per requirement. In our implementation, the suspicious data model has following template definitions.

```
(deftemplate Trans
(slot actmessage)
(slot action_name)
(slot username)
(slot userhost)
(slot timestamp)
…………………)
```

We fired SQL query on DBA_AUDIT_TRAIL and DBA_ROLE_PRIVS. The result set is asserted into facts. In addition to the facts, rules are defined. We design different rules to find out illegal transactions. The knowledge is represented as the following rule.

The above rule says that if the action name is insert, then modify action message and increase insert count and new value asserts

```
If action_name is Insert
Then  Modify  action message is Illegal Insertion AND
        Increase  count of  Insert actions AND
        Assert counted value into facts AND
        Assert username who did Insert action into facts AND
        Assert hostname from which Insert action take place
into facts
```

into facts. The Defrule can search knowledge base to find relationships between facts, and rules can take actions based on the contents of one or more facts. A JESS rule is something like if…then statement in a procedural language, but it is not used in a procedural way. While if…then statements are executed at a specific time and in a specific order, according to how the programmer writes those, JESS rules are executed whenever their if parts (their left-hand-sides or LHSs) are satisfied, given only that the rule engine is running. This makes JESS rules less deterministic than a typical procedural program. Rules are defined in JESS using the Defrule construct.
The following is the JESS language representation of the above rule.

```
(defrule insert_rule
 ?r1<-(Trans ( action_name  ?*actname*)(username ?un)….)
 ?r2<-(Cnt_action( …))
 ?c1<-(accumulate(bind ?*cnt* 0) (bind ?*cnt*(+ ?*cnt* 1))
 ?*cnt* (Trans(action_name  ?a&: (…..))))
  => (modify ?r1 (actmessage  \"Illegal-Insertion\" ))
    modify ?r2 (cnt ?*cnt*))
       (assert(Cnt_user(…. )))
       (assert(Cnt_host(…. ))));
```

Similarly we defined rules for suspicious transactions like modify, remove and copy the biometric template storage. In a backwards chaining system, rules are still if..then statements, but the engine seeks steps to activate rules whose preconditions are not met. This behavior is often called "goal seeking". JESS supports both forward and backward chaining. In this paper, we use back tracing for postmortem of the intrusion to find source of intrusion. We use Defquery construct for back tracing, which displays detail knowledge about OS username, username, object name, owner of object, time stamp, session-id and so on. The Defquery construct lets you create a special kind of rule with no right-hand-side. While rules act spontaneously, queries are used to search the knowledge base under direct program control. A rule is activated once for each matching set of facts, while a query gives you a java.util.Iterator of all the matches. It can be convenient to use queries as triggers for backward chaining. For this to be useful, Rete.run() must be called while the query is being evaluated, to allow the backward chaining to occur. Facts generated by rules fired during this run may appear as part of the query results. We use Defquery as follows:

```
defquery search-by-mess (declare(variables ?act )) (Trans
(actmessage ?act)(action_name ?an)(username ?un)
(timestamp ?ts)……….)
```

Similarly we backtrack for suspicious transactions like modify, remove and copy done by DBA role on the biometric template storage.

Result Screens

The (fig.5 and fig 6) show result screens as output of our system. The Biometric Template Storage Intrusion Detection Assistant which displays two tables namely User Intrusion which contains suspicious activities of normal users and DBA intrusion which contains suspicious activities of DBA. A text pane is used to display detail information of selected suspicious activity. Three tables which show top intruders, top suspicious hosts and top suspicious DBA hosts. These tables are used to find out most suspicious user or host and that knowledge is used for taking any preventive actions. One bar graph shows which transaction is done repeatedly as suspicious activity by normal user while another one that of DBA. User intrusion graph shows Insert transaction tried at 9 times, Delete transaction tried at 19 times, Update transaction tried at 18 times, Delete transaction tried at 20 times as suspicious activity. DBA intrusion graph shows Insert transaction tried at 6 times, Delete transaction tried at 7 times, Update transaction tried at 2 times, Delete transaction tried at 8 times as suspicious activity. If user selects any row from normal user suspicious activity table ,then details about name of the user whose action where audited(USERNAME),operating system login username of the user whose actions were audited(OS_USERNAME),client host machine name (USERHOST), Numeric ID of each ORACLE session (SESSIONID), Name of the object affected by action (OBJ_NAME),Timestamp of the creation of the audit trail entry in Universal Time Coordinated (UTC) zone (EXTENDED_TIMESTAMP) will display on the screen. Fig. 5 shows detail information about selected row of user Intrusion table which contains suspicious activity of normal user. Similarly if user selects any row from DBA suspicious activity table ,then details about name of the DBA whose action where audited (USERNAME),operating system login username of the user whose actions were audited(OS_USERNAME),client host machine name (USERHOST), Numeric ID of each ORACLE session (SESSIONID), Name of the object affected by action (OBJ_NAME), Timestamp of the creation of the audit trail entry in Universal Time Coordinated (UTC) zone (EXTENDED_TIMESTAMP) will display on the screen.fig. 6 shows detail information about selected row of DBA intrusion table which contains suspicious activity of DBA.

Conclusion

In this paper, a simple implementation of knowledge based Biometric Template storage Intrusion Detection assistant is portrayed. This intelligent agent is located on the Biometric Template storage database. The intrusion detection is executed in background. When it detects suspicious or illegal activities, it notifies the security administrator. For detecting intrusive activities, IDS can use audit file data. In this paper we consider Distributed HOST-based IDS which are in charge of monitoring several hosts. It performs intrusion detection using Operating System's audit trail, RDBMS audit trail or information from multiple monitored hosts. The system consists of a user interface module, an inference engine, a knowledgebase of illegal transactions and audit trail of ORACLE database. Inference engine is implemented using JESS which is a Java Expert System Shell.

Future Work

In this paper we design and implement intelligent agent for biometric template storage intrusion detection. In future the research will expand to design other agents and preventive actions for detected intrusion using different AI techniques.

References

[1] *Risk And Control of Biometric Technologies A Security, Audit And Control Primer at www.isaca.org*.

[2] Ratha N., Connell J.H. and Bolle R.M. (2001) *Audio and Video-based Biometric Person Authentication*, 223–228.

[3] Biometric Evaluation Methodology (BEM) (2002) *suppliment Produced by the common criteria Biometric evaluation methodology working group*.

[4] Anil K. Jain, Karthik Nandakumar and Abhishek Nagar (2008) *Journal on Advances in Signal Processing*, 17.

[5] Ford W. (1994) *Computer Communications Security: Principles, Standard Protocols and Techniques, Ed. Prentice Hall PTR*.

[6] Puketza M. Chung, Olsson R.A., Mukherjee B. (1997) *IEEE Software Journal*, 43-51.

[7] Price K. (1998) *Computer Sciences Computer, Purdue University*.

[8] Houda Labiod, Karima Boudaoud, and Jacques Labetoulle, *Towards a new approach for intrusion detection with intelligent agents*"

[9] Ernest Friedman –Hill, *JESS in Action*.

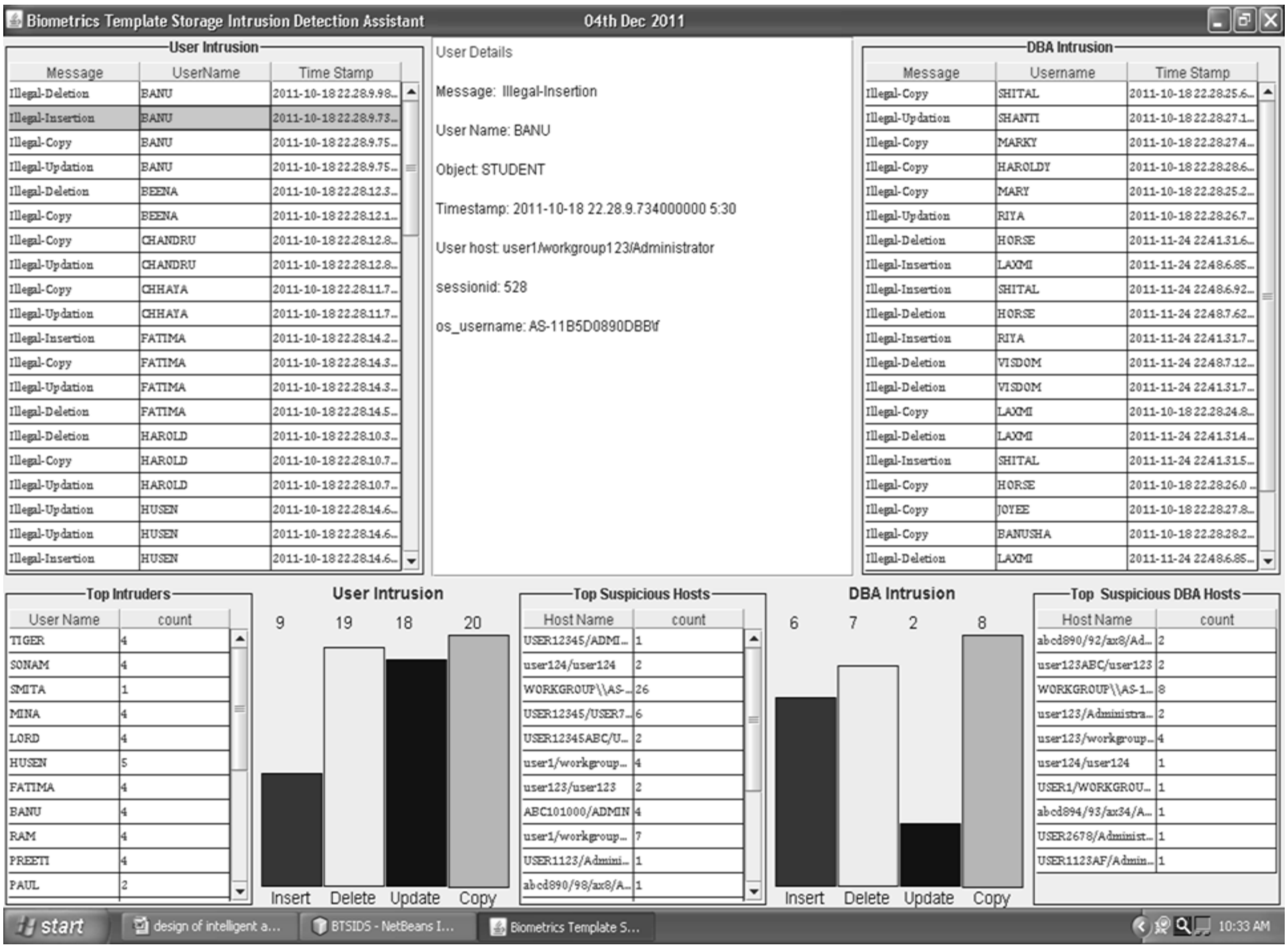
[10] Ernest Friedman –Hill, *Jess, The Expert System Shell for the Java Platform" http://herzberg.ca.sandia.gov/jess*.

[11] In-GookChun, In-Sik Hong (2001) *IEEE, ISIE, Pusan,Korea*.

[12] Maithili Arjunwadkar and Dr. R. V. Kulkarni (2010) *International Journal Of Computer Application*, 3(6) 10-12.

[13] Maithili Arjunwadkar and Kulkarni R.V. (2010) *Journal Of Emerging Trends In Computing And Information Science*,1(2), 117-120.

[14] Maithili Arjunwadkar and Kulkarni R.V. (2011) *International Journal of Computational Intelligence and Information Security*, 2(6), 50-60.



User Intrusion

Message	UserName	Time Stamp
Illegal-Deletion	BANU	2011-10-18 22.28.9.98...
Illegal-Insertion	BANU	2011-10-18 22.28.9.73...
Illegal-Copy	BANU	2011-10-18 22.28.9.75...
Illegal-Updation	BANU	2011-10-18 22.28.9.75...
Illegal-Deletion	BEENA	2011-10-18 22.28.12.3...
Illegal-Copy	BEENA	2011-10-18 22.28.12.1...
Illegal-Copy	CHANDRU	2011-10-18 22.28.12.8...
Illegal-Updation	CHANDRU	2011-10-18 22.28.12.8...
Illegal-Copy	CHHAYA	2011-10-18 22.28.11.7...
Illegal-Updation	CHHAYA	2011-10-18 22.28.11.7...
Illegal-Insertion	FATIMA	2011-10-18 22.28.14.2...
Illegal-Copy	FATIMA	2011-10-18 22.28.14.3...
Illegal-Updation	FATIMA	2011-10-18 22.28.14.3...
Illegal-Deletion	FATIMA	2011-10-18 22.28.14.5...
Illegal-Deletion	HAROLD	2011-10-18 22.28.10.3...
Illegal-Copy	HAROLD	2011-10-18 22.28.10.7...
Illegal-Updation	HAROLD	2011-10-18 22.28.10.7...
Illegal-Updation	HUSEN	2011-10-18 22.28.14.6...
Illegal-Updation	HUSEN	2011-10-18 22.28.14.6...
Illegal-Insertion	HUSEN	2011-10-18 22.28.14.6...

User Details

Message: Illegal-Insertion

User Name: BANU

Object: STUDENT

Timestamp: 2011-10-18 22.28.9.734000000 5:30

User host: user1/workgroup123/Administrator

sessionid: 528

os_username: AS-11B5D0890DBB\f

DBA Intrusion

Message	Username	Time Stamp
Illegal-Copy	SHITAL	2011-10-18 22.28.25.6...
Illegal-Updation	SHANTI	2011-10-18 22.28.27.1...
Illegal-Copy	MARKY	2011-10-18 22.28.27.4...
Illegal-Copy	HAROLDY	2011-10-18 22.28.28.6...
Illegal-Copy	MARY	2011-10-18 22.28.25.2...
Illegal-Updation	RIYA	2011-10-18 22.28.26.7...
Illegal-Deletion	HORSE	2011-11-24 22.41.31.6...
Illegal-Insertion	LAXMI	2011-11-24 22.48.6.85...
Illegal-Insertion	SHITAL	2011-11-24 22.48.6.92...
Illegal-Deletion	HORSE	2011-11-24 22.48.7.62...
Illegal-Insertion	RIYA	2011-11-24 22.41.31.7...
Illegal-Deletion	VISDOM	2011-11-24 22.48.7.12...
Illegal-Deletion	VISDOM	2011-11-24 22.41.31.7...
Illegal-Copy	LAXMI	2011-10-18 22.28.24.8...
Illegal-Deletion	LAXMI	2011-11-24 22.41.31.4...
Illegal-Insertion	SHITAL	2011-11-24 22.41.31.5...
Illegal-Copy	HORSE	2011-10-18 22.28.26.0 ...
Illegal-Copy	JOYEE	2011-10-18 22.28.27.8...
Illegal-Copy	BANUSHA	2011-10-18 22.28.28.2...
Illegal-Deletion	LAXMI	2011-11-24 22.48.6.85...

Top Intruders

User Name	count
TIGER	4
SONAM	4
SMITA	1
MINA	4
LORD	4
HUSEN	5
FATIMA	4
BANU	4
RAM	4
PREETI	4
PAUL	2

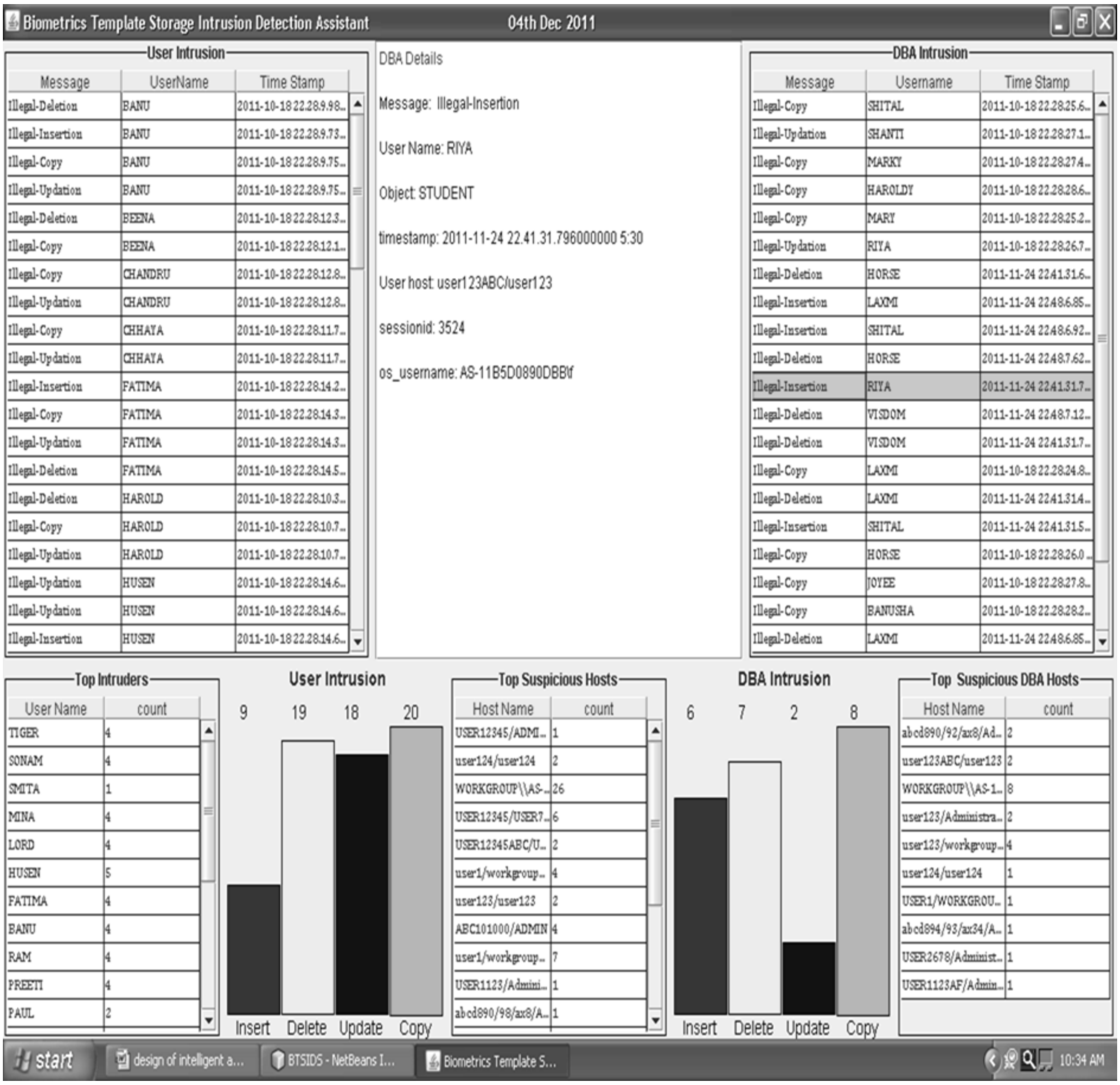
Top Suspicious Hosts

Host Name	count
USER12345/ADMI...	1
user124/user124	2
WORKGROUP\\AS-...	26
USER12345/USER7...	6
USER12345ABC/U...	2
user1/workgroup...	4
user123/user123	2
ABC101000/ADMIN	4
user1/workgroup...	7
USER1123/Admini...	1
abcd890/98/ax8/A...	1

Top Suspicious DBA Hosts

Host Name	count
abcd890/92/ax8/Ad...	2
user123ABC/user123	2
WORKGROUP\\AS-1...	8
user123/Administra...	2
user123/workgroup...	4
user124/user124	1
USER1/WORKGROU...	1
abcd894/93/ax34/A...	1
USER2678/Administ...	1
USER1123AF/Admin...	1

Fig.5- shows select any row from user Intrusion table and see the details of that transaction


Biometrics Template Storage Intrusion Detection Assistant
04th Dec 2011
start
design of intelligent a...
BTSIDS - NetBeans I...
Biometrics Template S...
10:34 AM

User Intrusion

Message	UserName	Time Stamp
Illegal-Deletion	BANU	2011-10-18 22.28.9.98...
Illegal-Insertion	BANU	2011-10-18 22.28.9.73...
Illegal-Copy	BANU	2011-10-18 22.28.9.75...
Illegal-Updation	BANU	2011-10-18 22.28.9.75...
Illegal-Deletion	BEENA	2011-10-18 22.28.12.3...
Illegal-Copy	BEENA	2011-10-18 22.28.12.1...
Illegal-Copy	CHANDRU	2011-10-18 22.28.12.8...
Illegal-Updation	CHANDRU	2011-10-18 22.28.12.8...
Illegal-Copy	CHHAYA	2011-10-18 22.28.11.7...
Illegal-Updation	CHHAYA	2011-10-18 22.28.11.7...
Illegal-Insertion	FATIMA	2011-10-18 22.28.14.2...
Illegal-Copy	FATIMA	2011-10-18 22.28.14.3...
Illegal-Updation	FATIMA	2011-10-18 22.28.14.3...
Illegal-Deletion	FATIMA	2011-10-18 22.28.14.5...
Illegal-Deletion	HAROLD	2011-10-18 22.28.10.3...
Illegal-Copy	HAROLD	2011-10-18 22.28.10.7...
Illegal-Updation	HAROLD	2011-10-18 22.28.10.7...
Illegal-Updation	HUSEN	2011-10-18 22.28.14.6...
Illegal-Updation	HUSEN	2011-10-18 22.28.14.6...
Illegal-Insertion	HUSEN	2011-10-18 22.28.14.6...

DBA Details

Message: Illegal-Insertion

User Name: RIYA

Object: STUDENT

timestamp: 2011-11-24 22.41.31.796000000 5:30

User host: user123ABC/user123

sessionid: 3524

os_username: AS-11B5D0890DBB\f

DBA Intrusion

Message	Username	Time Stamp
Illegal-Copy	SHITAL	2011-10-18 22.28.25.6...
Illegal-Updation	SHANTI	2011-10-18 22.28.27.1...
Illegal-Copy	MARKY	2011-10-18 22.28.27.4...
Illegal-Copy	HAROLDY	2011-10-18 22.28.28.6...
Illegal-Copy	MARY	2011-10-18 22.28.25.2...
Illegal-Updation	RIYA	2011-10-18 22.28.26.7...
Illegal-Deletion	HORSE	2011-11-24 22.41.31.6...
Illegal-Insertion	LAXMI	2011-11-24 22.48.6.85...
Illegal-Insertion	SHITAL	2011-11-24 22.48.6.92...
Illegal-Deletion	HORSE	2011-11-24 22.48.7.62...
Illegal-Insertion	RIYA	2011-11-24 22.41.31.7...
Illegal-Deletion	VISDOM	2011-11-24 22.48.7.12...
Illegal-Deletion	VISDOM	2011-11-24 22.41.31.7...
Illegal-Copy	LAXMI	2011-10-18 22.28.24.8...
Illegal-Deletion	LAXMI	2011-11-24 22.41.31.4...
Illegal-Insertion	SHITAL	2011-11-24 22.41.31.5...
Illegal-Copy	HORSE	2011-10-18 22.28.26.0 ...
Illegal-Copy	JOYEE	2011-10-18 22.28.27.8...
Illegal-Copy	BANUSHA	2011-10-18 22.28.28.2...
Illegal-Deletion	LAXMI	2011-11-24 22.48.6.85...

Top Intruders

User Name	count
TIGER	4
SONAM	4
SMITA	1
MINA	4
LORD	4
HUSEN	5
FATIMA	4
BANU	4
RAM	4
PREETI	4
PAUL	2

User Intrusion

Insert	Delete	Update	Copy
9	19	18	20

Top Suspicious Hosts

Host Name	count
USER12345/ADMI...	1
user124/user124	2
WORKGROUP\\AS-...	26
USER12345/USER7...	6
USER12345ABC/U...	2
user1/workgroup...	4
user123/user123	2
ABC101000/ADMIN	4
user1/workgroup...	7
USER1123/Admini...	1
abcd890/98/ax8/A...	1

DBA Intrusion

Insert	Delete	Update	Copy
6	7	2	8

Top Suspicious DBA Hosts

Host Name	count
abcd890/92/ax8/Ad...	2
user123ABC/user123	2
WORKGROUP\\AS-1...	8
user123/Administra...	2
user123/workgroup...	4
user124/user124	1
USER1/WORKGROU...	1
abcd894/93/ax34/A...	1
USER2678/Administ...	1
USER1123AF/Admin...	1

Fig.6- Shows select any row from DBA Intrusion table and see the details of that transaction