PROJECT SERENDIPITY

Ahmed Mahdy
Ahmed Tantawy
Dina Said
Mennat Mokhtar
Sherif Hashem

Design Document

Senior Project II

The American University in Cairo
Computer Science Department
Fall 2003

Supervised By: Dr. Sherif El Kassas
Serendipity Team

Project Serendipity

Design Document

Version: 3.0 (RC2)

Date: Thursday, October 23, 2003
Table of Contents

TABLE OF CONTENTS ........................................................................................................... 5
ACKNOWLEDGMENTS............................................................................................................. 7
GLOSSARY................................................................................................................................. 8
1. INTRODUCTION................................................................................................................... 9
  1.1 PURPOSE .......................................................................................................................... 9
  1.2 METHODOLOGY ............................................................................................................. 10
  1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS..................................................... 10
  1.4 REFERENCES.................................................................................................................. 10
  1.5 SPONSORSHIP AGREEMENT ....................................................................................... 11

2. OVERALL DESCRIPTION................................................................................................ 12
  2.1 PROBLEM DESCRIPTION ............................................................................................. 12
  2.2 PROPOSED SOLUTION .................................................................................................. 13
  2.3 PROJECT DESCRIPTION ............................................................................................... 14

3. UTILIZED COMPONENTS ............................................................................................. 16
  3.1 SENSORS ......................................................................................................................... 16
  3.1.1 Network Based............................................................................................................ 16
  3.1.1.1 Snort ......................................................................................................................... 17
  3.1.1.2 NetSTAT .................................................................................................................... 17
  3.1.2 Host Based................................................................................................................ 18
  3.1.2.1 LinSTAT .................................................................................................................... 18
  3.1.2.2 uSTAT ...................................................................................................................... 18
  3.1.2.3 WebSTAT ................................................................................................................ 18
  3.1.2.4 LogStat ...................................................................................................................... 19
  3.1.2.5 Tripwire .................................................................................................................... 19
  3.2 LIBRARIES ....................................................................................................................... 19
  3.2.1 Etheral Filter Language .............................................................................................. 21
  3.2.2 Java SNMP Library .................................................................................................... 21
  3.3 PACKAGES ........................................................................................................................ 23
  3.3.1 STAT framework ....................................................................................................... 23
  3.3.2 Stunnel ...................................................................................................................... 25
  3.3.3 Syslog Deamon .......................................................................................................... 26

4. DESIGN................................................................................................................................ 27
  4.1 DATAFLOW AND COMMUNICATION .......................................................................... 27
  4.2 CONNECTIVITY ................................................................................................................. 28
  4.3 USER INTERFACE ......................................................................................................... 29
  4.4 DATABASE ......................................................................................................................... 30
  4.5.1 Communication Interface Module ............................................................................ 32
  4.5.2 Analysis Engine ........................................................................................................ 33
  4.5.3 Serendipity RULES .................................................................................................... 33
  4.5.4 Statistics Engine ........................................................................................................ 34
  4.5.5 Database Driver ......................................................................................................... 34
  4.5.6 Web Interface Module .............................................................................................. 34
Acknowledgments

The Serendipity Team would like to thank:

**Dr. Sherif El Kassas** – Senior Project Supervisor

**Dr. Samy El Akabawi** – Associate Vice President for Computing

**Eng. Khaled Hilal** – Associate Director of University Network Services

**Eng. Hussein Moustafa** – Networking and Systems Security Officer

**Dr. Amir Zeid** – Senior Project Coordinator

**Eng. Wahib Basilious** - Assistant Director for User Support

**Eng. Emad Shaaban** - Senior Network Technician

**Eng. Emad Anwar** – Computer Lab Engineer

**Eng. Sherif El Kiki** – Technical Operations Manager

**Eng. Sinout Shenouda** – Senior Lab Engineer

**Mr. Ehab Mohsen** – Networking Lab Assistant

**Mr. Marmina Abdel-Malek** – Security Office Researcher

**Mr. Mohamed Ali** – Computer Technician

For their generous help and perpetual support.
Glossary

**Intrusion Detection System:** Intrusion Detection Systems are Software products monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, which are attempts to compromise the confidentiality, integrity, availability, or to alter the security Mechanisms of a computer or network.

**Log File:** A file that lists actions that have occurred. For example, Web servers maintain log files listing every request made to the server.

**Secure Socket Layer SSL:** The primary goal of the SSL Protocol is to provide privacy and reliability between two communicating applications. The protocol is composed of two layers. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP), is the SSL Record Protocol. The SSL Record Protocol is used for encapsulation of various higher level protocols. One such encapsulated protocol, the SSL Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. One advantage of SSL is that it is application protocol independent.

**Syslogd:** syslogd is a daemon (background application) that accepts messages from other applications and the network, and writes them out to system wide log files. Using these log files, the system administrator can see the "state" of the system at any point in time since the system was booted. syslogd reads and logs messages to the system console, log files, other machines and/or users as specified by its configuration file.

**SNMP:** Simple Network Management Protocol (SNMP) is the protocol governing network management and the monitoring of network devices and their functions. It is not necessarily limited to TCP/IP networks. SNMP is described formally in the Internet Engineering Task Force (IETF) Request for Comment (RFC) 1157 and in a number of other related RFCs.
1. Introduction

Serendipity is a Risk Management System that provides a solution to combine the collection, correlation and analysis of various computing network events. Serendipity integrates intrusion detection, response, vulnerability assessment, analysis and providing meaningful feedback into a single, cost-effective, managed solution.

In his book, “The Art of Deception”, Kevin Mitnik – The world’s most famous hacker and currently converted Security Consultant says:

“Having a single place to report security incidents will provide an effective early warning system that will make it clear when a coordinated attack is underway so that any damage can be controlled immediately.”

This best describes Serendipity’s foremost methodology.

1.1 Purpose

The purpose of this document is to communicate to the Faculty of the Computer Science Department at the American University in Cairo the design of the Serendipity System. This self contained document discusses the several blueprints that will make possible the implementation of the project
1.2 Methodology

It is crucial to state that Project Serendipity is a system integration project not a software engineering project. Serendipity heavily relies on currently existing software and projects. Software development is nominal compared to the integration aspect of the project.

To maintain the consistency of the document and the norm in the Computer Science department, this document will be structured as close as possible to a standard Design Document. Due to the integration aspect of this project, not all design is completed as this document goes to print. The development of this system follows more of a design as you integrate methodology. System Connectivity was designed then integrated, afterwards a networking solution was designed and implemented. This paper lacks the design of some few aspects of the system however, integration was taking place side by side of design, and as early as day 1 of the beginning of the semester and Senior Project II.

1.3 Definitions, Acronyms, and Abbreviations

The definition of all terms, acronyms and abbreviations will be explicitly stated throughout the document. A glossary section can be formerly located.

1.4 References

Where appropriate and needed, all citations and references will be properly footnoted and referred to in the latter bibliography section.
1.5 Sponsorship Agreement

The office of the Associate Vice President for Computing has agreed to sponsor our project. With the generous offer from Dr. Sami Akabawi, Associate Vice President for Computing, University Network Services and the Network and Systems Security Office will be providing us with technical help and support, offering us access to networking resources and providing us with network equipment and network data.

We have agreed that the office of the Associate Vice President for computing is entitled to a copy of all our work as well as the right to utilize and benefit from the outcome of our project.
2. Overall Description

2.1 Problem Description

With the increased attainment of computing assets and technology related infrastructure, securely networking such devices has become imperative. Acquisition of various systems that enable a secure networking environment has been rapidly increasing and has become an of the essence saddle to the aware network administrator to maintain an up and secure network. Firewall, Intrusion detection systems, Log Parsers, Traffic Analyzers and a bakers dozen more have catered to the prop up of the network.

Ample alerts, logs and events are generated by these devices, of which most are neglected or unattended to due to the size of data generated. Although in a small network such data might be more often than not examined, yet in a typical large production environment, only with the aid of an appropriate network and risk management system is this possible, and it becomes a tedious and labor extensive task for the network administrator. Opportune findings become as dreary as retrieving a needle from a haystack.

Commercially available Risk Management and Network Management systems hare feeble in several aspects. Some are not tailored to integrating various multi vendor systems, others are limited to components from the same vendor. Other limitations include but are not limited to closed source, inflexibility, and highly priced commercial systems.
2.2 Proposed Solution

Serendipity aims at integrating several network components and devices at the Computer Science Department network at the American University in Cairo. Various data, events and alerts are to be combined to a centralized location for automated examination and analysis, then providing a meaningful and suggestive output to the user. Serendipity is designed to be an open source system, not only does it share its code, but also relies on other peoples code and components and easily allows the interchange of its modules and subsystems as well be illustrated later. This gives serendipity a comparative advantage in flexibility.

Serendipity is a cost efficient solution as it is built out of readily available CS department resources, no additional acquisition of resources in both hardware nor software had to be made.

The system is tailored for the network components, critical assets, connectivity and computing resources of the CS department network. This provides local meaningful results and is implemented in a real-time production environment.
2.3 Project Description

Serendipity is composed of six major distinct subsystems. The Intrusion Detection Subsystem, Log Collection Subsystem and the Simple Network Management Protocol Subsystem which combined form the input to Serendipity. The Serendipity Core is the heart and brain of serendipity, which also controls all other subsystems and the communication amongst them. The Reporting Subsystem provides meaningful output and response. We are left
with the **Database Subsystem** which stores all serendipity data and information. It also contains a backup system.

As input, Serendipity relies three subsystems, Intrusion Detection Subsystem, Log Collector subsystem and the SNMP subsystem. The IDS subsystem is composed of a group of various Host Based and Network Based sensors that feed their alerts to the Core subsystem for analysis and processing. The Log collection subsystem collects and passes various logs to the core subsystem. Firewall logs, Web server logs, System logs, Switch Logs and Router logs. All are passed to the Core subsystem for parsing and analysis. Finally the SNMP subsystem provides network status and awareness to Serendipity.

The core subsystem provides the processing, parsing and analysis for the system which provides its output to the reporting subsystem. The database subsystem provides storage functionality to serendipity.

This was a brief overall description of the system. Subsequently we will be discussing a detailed and elaborate design of the system.
3. Utilized Components

3.1 Sensors

3.1.1 Network Based

Network based sensors monitor traffic on a network segment, detecting and identifying malicious activity and then generating an alert for each activity. They detect both internal and external attacks.

Network based sensors have several approaches in attempting to detect intrusion. The Intrusion detection subsystem combines of several types of sensors with different approaches for the sake of completeness and inclusive integration.

1. Misuse Detection

Misuse detectors analyze system activity, looking for events or sets of events that match a predefined pattern of events that describe a known attack.

2. Anomaly Detection

Anomaly detectors identify abnormal unusual behavior (anomalies) on a host or network. They function on the assumption that attacks are different from “normal” (legitimate) activity and can therefore be detected by systems that identify these differences.
3.1.1.1 Snort

Snort is a lightweight network IDS, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and more. Snort uses a flexible rules language to describe traffic that it should collect or pass, and includes a detection engine utilizing a modular plug-in architecture. Snort can also function as a "passive trap" to record the presence of traffic that should not be found on a network, such as file sharing (peer-to-peer) connections such as (Kazaa, Napster, etc.).

3.1.1.2 NetSTAT

NetSTAT is a network-based intrusion detection system that analyzes network traffic on a network segment looking for packets that represent evidence of malicious behavior.

The State Transition Analysis Technique is a method to describe computer penetrations as attack scenarios. Attack scenarios are represented as a sequence of transitions that characterize the evolution of the security state of a system. In an attack scenario states represent snapshots of a system's security-relevant properties and resources. A description of an attack has an "initial" starting state and at least one "compromised" ending state. States are characterized by means of assertions, which are predicates on some aspects of the security state of the system. For example, in an attack scenario describing an attempt to violate the security of an operating system, assertions would state properties such as file
ownership, user identification, or user authorization. Transitions between states are annotated with signature actions that represent the key actions that if omitted from the execution of an attack scenario would prevent the attack from completing successfully. For example, in an attack scenario describing a network port scanning attempt, a typical signature action would include the TCP segments used to test the TCP ports of a host.

3.1.2 Host Based

Host Based IDS detects threats aimed at critical hosts (servers). HIDS monitor system generated logs, changes to system hardware and software configurations, user privileges, modifications to a user account and successful/failed logins for indications of an attack.

3.1.2.1 LinSTAT

LinSTAT is a host-based intrusion detection system, that uses an event stream provided by a kernel auditing module as input.

3.1.2.2 uSTAT

uSTAT is a host-based intrusion detection system that interprets audit trails produced by Sun Microsystem's Basic Security Module (BSM)

3.1.2.3 WebSTAT

WebSTAT is an application-based intrusion detection system that uses as inputs the logs generated by the Apache web server.
3.1.2.4 LogStat

LogSTAT is a host-based intrusion detection system, that uses UNIX syslogs as input.

3.1.2.5 Tripwire

Tripwire is a tool that checks to see what has changed on your system. The program monitors key attributes of files that should not change, including binary signature, size, expected change of size, etc. The hard part is doing it the right way, balancing security, maintenance, and functionality. Tripwire is originally known as an intrusion detection tool, but can be used for many other purposes such as integrity assurance, change management, policy compliance and more.

3.2 Libraries

3.2.1 Ethereal Filter Language

Ethereal is a free network protocol analyzer for Unix and Windows. It allows you to examine data from a live network or from a capture file on disk. You can interactively browse the capture data, viewing summary and detail information for each packet. Ethereal has several powerful features, including a rich display filter language and the ability to view the reconstructed stream of a TCP session.

We will use this language to provide extended functionality to the system. Although it might seem to cater to the advanced user, which most Serendipity users should, and are expected to be, the filter language is quite easy and predictable compared to the more primitive libpcap filter language.
Example:

To find: Host sending or receiving web traffic (communication on port 80)

The filter to be applied should be:

```
tcp.port == 80 and ip.addr == hostname
```

Follows is an excerpt of the filter language:

```
ip.addr  Source or Destination Address
         IPv4 address

ip.dst   Destination
         IPv4 address

ip.src   Source
         IPv4 address

snmp.agent  Agent address
            IPv4 address

snmp.community  Community
                 String

syslog.level   Level
               Unsigned 8-bit integer
               Message level

syslog.msg     Message
               String
               Message Text

tcp.port      Source or Destination Port
              Unsigned 16-bit integer

arp.dst.hw    Target hardware address
              Byte array

arp.dst.hw_mac Target MAC address
                 6-Byte Hardware (MAC) Address
```
Not only is the filter applied to raw traffic, but also to alerts and events, being able to easily extract valuable information such as attacking IP addresses, ARP flooding, Illegal Traffic (P2P file sharing) etc…

2.2.2 Java SNMP Library

A package such as netsnmpj will be used, which is an open source Java library that allows java code to perform SNMP v1, v2c and v3 operations using the net-snmp library.
3.3 Packages

3.3.1 STAT framework

The MetaSTAT distribution is composed of a set of components whose tasks is the remote control and monitoring of a set of STAT-based applications.

CommSTAT  CommSTAT provides a means to create secure connections between STAT components. CommSTAT connections are used to exchange control messages and information, e.g., alerts. Data sent through CommSTAT connections is formatted according to the IDWG’s Intrusion Detection Exchange Message Format (IDMEF). The library uses SSL.

STAT Proxy  The STAT proxy acts as an intermediary between a STAT-based application and the MetaSTAT controller. A STAT proxy is responsible for maintaining a host-based repository of STAT modules.

Controller  The controller maintains connections to the deployed STAT proxies and provides a user interface that allows an operator to issue control messages to STAT-based applications.

Collector  The collector gathers IDMEF alerts from the deployed STAT-based sensors and stores them in an alert database.

Viewer  A Java application that provides a graphic user interface for browsing the alerts stored in the centralized alert database.
The STAT Framework distribution contains all the components that are needed to develop a STAT-based application.

STATL Parser  A tool, implemented in Java, that translates STATL scenarios into C++ scenario plugins. The scenario plugins can then be compiled and loaded into a STAT-based application.

STAT Core  The STAT core is the runtime of the STATL language, which implements the domain-independent characteristics of STATL. The STAT Core performs the actual runtime intrusion detection analysis process by matching an incoming stream of events against a number of scenario plugins. A running instance of the STAT Core is dynamically extended to build a STAT-based application.

STATed Editor  A graphical editor for STATL scenarios. STATed is written in Java.

xSTAT  A generic STAT-based application. xSTAT can be extended with other modules to create a complete STAT-based application without having to develop a single line of code.

Test Extension  An example STATL language extension. This extension can be used as a guideline to develop new language extensions.

Test Provider  An example STAT event provider module. This provider can be used as a guideline to develop new event providers.
Test Scenarios Some example STAT scenario plugin modules. These scenarios can be used as a guideline to understand how STATL scenarios are translated into executable code.

STAT Responses Some example STAT response modules that provide simple response functions, such as logging to a file or producing IDMEF alerts.

STAT sensors will be directly controlled and send their alerts to the STAT framework for processing and analysis rather than to the Serendipity CORE. The STAT reports via MetaSTAT will be integrated in the Serendipity Reports.

Other “non-STAT” alerts, events and feed will be processed by the serendipity core.

3.3.2 Stunnel

Stunnel is a program that allows you to encrypt arbitrary TCP connections inside SSL (Secure Sockets Layer) available on both Unix and Windows. Stunnel can allow you to secure non-SSL aware daemons and protocols (like POP, IMAP, LDAP, etc) by having Stunnel provide the encryption, requiring no changes to the daemon's code.

Stunnel will securely accommodate all communication between sensors, databases and subsystems.
3.3.3 Syslog Daemon

Syslogd is a daemon (background application) that accepts messages from other applications and the network, and writes them out to system wide log files. Using these log files, the system administrator can see the "state" of the system at any point in time since the system was booted.

Syslogd will be conveying switch log files, web server log files, PIX firewall alerts and other the real secure sensors to the machine running the serendipity core to be analyzed.
4. Design

4.1 Dataflow and Communication

The following diagram illustrates the dataflow and communication about among the detailed components and constituents of Serendipity.
4.2 Connectivity

The following diagram illustrates the equipment components of Serendipity as well as their connectivity and location in the CS department network.
4.3 User Interface

The following is the proposed site map for our web based user interface. Items in blue are applets that will be running when invoked.
4.4 Database

Entity Relationship Schema for Serendipity Core Database:
Entity Relationship schema for Snort + Mysql:
4.5 Core

This ingenious Subsystem is the aptitude spirit of Serendipity. It is composed of several interdependent modules. The core is the masterpiece of Serendipity. It contains several of the novel design features and innovative implementation techniques. The following diagram shows the modules of The Serendipity Core.

This part of serendipity has not yet been fully designed. Several integration steps had to be done first in order to deal with actual fields, alerts, logs etc… this is the next part of the project, the design, implementation and the integration of the core. Also it is highly dependant on the design of the databases and the imagination of the user interface and user functionality.

4.5.1 Communication Interface Module
This module is responsible for all the communication between all the subsystems. As seen in the main perspective diagram, all subsystem communication goes through the core. This design method was favored for several reasons:

1. Centralized Controlled Communication

2. To facilitate Local Event Logging

3. To control the amount of network traffic generated by the system as a total.

4.5.2 Analysis Engine

The Analysis Engine is responsible for all the scrutiny that is performed in Serendipity. It examines all the input from logs to sensor alerts correlating these events and trying to mine for relevant and valuable information. The main design of the engine has not been determined yet we are intrigued by a Smart engine that has an Intelligent aspect, combined with the rules defined by the Serendipity RULES.

4.5.3 Serendipity RULES

This module defines the set of rules for the Core Subsystem. These rules are mainly for the Analysis Engine to follow. It was designed as a separate engine to allow modifications such as adding new rules without having to modify the Analysis Engine. We are inclined in the Design of the Rules module to have an Application Programming Interface (API) that would allow users to write their own rules for the Analysis Engines. Also the Statistics Engine will be indirectly using these rules.
4.5.4 Statistics Engine

The role of this engine is to perform statistics based on the RULES and the output of the analysis Engine. It combines several filters with counting and reckoning techniques to provide useful numbers and figures.

4.5.5 Database Driver

This module provides the connectivity between the database and the core. We will use a standard communication implementation based on Open Database Connectivity (ODBC).

4.5.6 Web Interface Module

This module will provide a crossing point for Web Interface. It will deliver the output of serendipity to the user as well as accepting user input. It will be mostly interfacing with an intermediate Web server between serendipity and the end user’s client.
5. System Behavior

5.1 Static Behavior

5.1.1 Packages Diagram

The following package diagram aims on Group together functionally-similar components of our system. It also shows the data flow between the
5.1.2 Use Case Diagram

The following UML Use Case Diagram describes the functionality of our system in a horizontal way, by showing all of its available functionality.
<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Describes available system functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Priority</strong></td>
<td>Highest, as it is the only method of interaction with the system</td>
</tr>
<tr>
<td><strong>Actors</strong></td>
<td>User, Administrator</td>
</tr>
<tr>
<td><strong>Parent Use Case</strong></td>
<td>None (Standalone)</td>
</tr>
<tr>
<td><strong>Extending Use Cases</strong></td>
<td>None (Standalone)</td>
</tr>
<tr>
<td><strong>Included Use Cases</strong></td>
<td>None (Standalone)</td>
</tr>
<tr>
<td><strong>Pre-conditions</strong></td>
<td>A default administrator account must initially exist.</td>
</tr>
<tr>
<td><strong>Post-conditions</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Basic Flow</strong></td>
<td>A user can perform any of the existing services that are related to him. The Administrator, a special type of user has extended functionality.</td>
</tr>
<tr>
<td><strong>Related Objects</strong></td>
<td>User generalizes administrator</td>
</tr>
<tr>
<td><strong>Issues</strong></td>
<td>Both actors interact the system through the user interface, the only method to interact with the system.</td>
</tr>
</tbody>
</table>
5.2 Dynamic Behavior

5.2.1 SNMP Sequence Diagram
5.2.2 STAT Sequence Diagram
5.2.3 Web Server Sequence Diagram
5.2.4 Snort Sequence Diagram
6. System Testing

This part of the document deals with testing the various subsystems and components of serendipity. We will use a testing template for the various components.

6.1 Connectivity Testing

**System:**
Connectivity among subsystems

**Test Description:**
This test makes sure that each subsystem is operational and aware of the other subsystem it is required to communicate too.

**Severity:** High

**Instructions:**
1. Using SNMP, each subsystem should respond when polled. Pinging the device should also result in a live response.

2. Wiring connectivity must be checked to assure proper connectivity. A physical network connectivity tool is required to assure proper connectivity.

3. Each subsystem should send a test message to the subsystem it is communicating to.

**Expected Result:**
1. Subsystem should properly respond to polls.
2. Subsystem should respond to ping requests.
3. Subsystem should acknowledge the receive of test message.
4. Proper Physical connectivity must be maintained.

**Cleanup:**
1- During cable testing, cables have to be returned to their appropriate locations.
6.2 Sensor Testing

**System:** Various deployed Intrusion Detection Sensors.

**Test Description:**
This test will make sure that sensors have the latest rules files as well as properly respond to attacks.

**Severity:**
High

**Instructions:**
1. Sensors must be checked for the latest rules files.
2. Sensors must be simulated for an attack and response checked.

**Expected Result:**
1. Sensors must properly detect and alert for the attack.

**Cleanup:**
1. Attack must be recorded as a test attack and administrator notified before such a test.
6.3 Database Testing

**System:** The various databases

**Test Description:**
This test is to make sure databases accept incoming data, properly respond to queries appropriately.

**Priority:**
Medium-High

**Instructions:**
1. Send the database a test entry to be recorded in one of the tables
2. Query the database for that entry

**Expected Result:**
1. The query should return the appropriate test entry

**Cleanup:**
1. Test entries must be properly removed and discarded after inputting them
6.4 Core Testing

**System:** Serendipity Core

**Test Description:**
Testing the analysis and reporting functionality of the core

**Severity:**
High

**Instructions:**
1. Simulate several attacks and make sure that they are properly collected, analyzed and reported by the core.

**Expected Result:**
1. A report of the attack should be displayed

**Cleanup:**
1. Attack must be recorded as a test attack and administrator notified before such a test.
6.5 Security Testing

**System:**
Secure Communication between subsystems and the encryption of stored data

**Test Description:**
Since we are dealing with critical live information, confidentiality is mandatory. Stored data must be encrypted and all data communication must be secured.

**Severity:**
Highest

**Instructions:**
1. Sniffing to intercept inter-subsystem communication
2. Trying to break into databases.

**Expected Result:**
1. Communication should be encrypted properly with a proper algorithm, proper key control and password control.
2. The database should be unreadable as the data is encrypted following proper cryptography techniques.
7. Logistics

7.1 Detailed Time Plan

---

1. Setting up the machine
2. Choosing the utilization method
7.2 Division of Labor

7.2.1 Tasks Completed

<table>
<thead>
<tr>
<th>Name</th>
<th>Tasks Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sherif</td>
<td>Networking &amp; Connectivity</td>
</tr>
<tr>
<td></td>
<td>Systems Installation and Configuration</td>
</tr>
<tr>
<td>Mennat</td>
<td>Interface Design</td>
</tr>
<tr>
<td></td>
<td>Site Map</td>
</tr>
<tr>
<td></td>
<td>Time Plan</td>
</tr>
<tr>
<td>Dina</td>
<td>Sensors Installation</td>
</tr>
<tr>
<td></td>
<td>Sensor Configuration</td>
</tr>
<tr>
<td></td>
<td>Package Selection</td>
</tr>
<tr>
<td></td>
<td>Securing Connectivity</td>
</tr>
<tr>
<td>Ahmed Mahdy</td>
<td>Database Design and Implementation</td>
</tr>
<tr>
<td>Ahmed Tantawy</td>
<td>Logistics and Testing</td>
</tr>
</tbody>
</table>

7.2.2 Tasks Remaining

<table>
<thead>
<tr>
<th>Name</th>
<th>Tasks Remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sherif</td>
<td>Component Integration</td>
</tr>
<tr>
<td></td>
<td>Rules and Core Design</td>
</tr>
<tr>
<td>Mennat</td>
<td>Core Programming</td>
</tr>
<tr>
<td></td>
<td>Interface Implementation</td>
</tr>
<tr>
<td>Dina</td>
<td>Component Installation</td>
</tr>
<tr>
<td>Ahmed Mahdy</td>
<td>Database Design and Implementation</td>
</tr>
<tr>
<td>Ahmed Tantawy</td>
<td>Core Programming</td>
</tr>
<tr>
<td></td>
<td>Interface Implementation</td>
</tr>
</tbody>
</table>
Bibliography

http://www.snort.org

http://networking.webopedia.com/

http://www.cs.ucsb.edu/~rsg/STAT/

http://www.tripwire.org

http://net-snmp.sourceforge.net/

http://netsnmpj.sourceforge.net/

http://www.stunnel.org