Assignment 4: Mini-project; Information Systems Management System

A Security Case Study submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Prepared By
Derek J. Sedlack
Division of Information Sciences
Graduate School of Computer and Information Sciences
Nova Southeastern University

Conducted for
DISS 799 Information Security
Dr. James Cannady, Professor

2005

Certification of Authorship of Doctoral Work

Submitted to:
Professor James Cannady, Ph.D.

Student's Name:
Derek J. Sedlack

Date of Submission:
December 9, 2005

Purpose and Title of Submission:
Partial fulfillment requirement for doctoral degree.
Assignment 4: Mini-project

Certification of Authorship: I hereby certify that I am the author of this document and that any assistance I received in its preparation is fully acknowledged and disclosed in the document. I have also cited all sources from which I obtained data, ideas, or words that are copied directly or paraphrased in the document. Sources are properly credited according to accepted standards for professional publications. I also certify that this paper was prepared by me for this purpose.

Student's Signature: 

Java Application
Conduct a thorough analysis of your course topic through the development of a Java-based application that demonstrates some aspect of the topic area. Your task is to design and develop a prototype system that utilizes information security techniques to solve problems in an appropriate domain.
You are required to submit a final report on your application that includes the following:

1. Identify the problem that you were trying to solve.

Many security experts agree (Patterson and Blue, 2004; Chivers and Fletcher, 2005; McCumber, 1991; Peltier, 2000; Peltier, 2004) that the idea of top-down security aspects should be augmented by analyzing the entire organization, especially using a designed methodology like ISO 17799 in an Information Security Management System (ISMS). Auditing and policy enforcement are vital to organizations maintaining current, viable information assurance security program. Peltier (2000) proposed a matrix for assessing information as deliverables based on predefined controls that are assessed both internally and externally, but this will be incorporated in phase 2.

2. Justify the need for an information security-based approach to solve the problem.

Creating an applet that provides electronic measures for auditing and enforcement will allow different organizational segments to compare/contrast information assurance measures. The first aspect should be automated and stored. While the application does not database the information yet, future revisions would allow comparisons over time or inter-corporate comparisons for multi-national conglomerates. Phase 2 will include automated Peltier (2000) deliverable recommendations in addition to the existing functionality.

3. Discuss the problem solving methodology in detail.

During phase 1 of this applet, a number of sections and some questions from various sources were utilized to create minimal coverage of auditing and enforcement information assurance security policies and procedures. Digitizing the responses would allow easy future comparisons and allow a programmatic approach to build in comparison algorithms based on actuarial tables or corporate standards for assessment based on the responses. The algorithm is currently experimental and only based on loose mathematics, but will be refined after several iterations of practical application. In phase 2, databasing the information will allow longitudinal contrast/comparisons from within- or inter-corporate analysis.

4. Provide the design specifications for the system

Access Control Lists
- Users:
  - Internal Employees
  - External Employees
  - Partner Employees
  - Patrons
- Creation and Maintenance
  - Defaults
- Privileged Users
- Conflicts
  - Lists
  - Rules
  - Exceptions
  - Escalations
- Revocations
  - Rights
    - Temporary
    - Permanant

Policies
- Creation and Maintenance
  - Defaults
  - Review Period (at least 1x / yr)
  - Review Committee
- Users
  - Internal Employees
  - External Employees
  - Partner Employees
  - Patrons
- Conflicts
  - Rules
  - Distribution
  - Exceptions
  - Escalations
- Mechanisms
  - Creation
  - Distribution

Auditing
- Base
  - State-Based
  - Transition-Based
- Logger
  - Mechanism Definition
  - Format (Binary/Interpretable)
  - Integrity Assurance
  - Sanitization
- Analyzer
  - Detect Events
  - Detect Nonevents
  - Detect Patterns/Symantics
- Notifier
  - Informants
  - Responses Required
  - Notification Facility

Deliverables
- Information accessed by personnel not intended to have access.
- Unclear or non-existent versioning of the information.
- Database could be corrupted by hardware failure, incorrect, or bad software.
- Data coul be corrupted by an incomplete transaction.
- Ability to change data in transit and then change it back in order to cover the activity.
- A failure to report integrity issues.
- Incompletely run process or failure to run a process that could corrupt the data.
- Lack of internal processes to create, control, manage data across functions.
- No notification of integrity problems.
- Information being used in the wrong context or environment.

- Third-party information may have integrity issues.
- Third-party access to information.

Each Yes answer = 0 points
Each No answer = 1 point
Each N/A answer = 2 points
i = each answer in section 4 (Organizational Policy)
Q = total number of questions in section 4
S = the security result of the system analysis

$$S = \Sigma_i / Q$$

The smaller number of total items required individual valuations that were conducted in the following manner:

	Inadequate protection	Minimal protection	Reasonable protection	Adequate protection
Access Control	S < .66	.66 >= S < .8	.8 >= S < 1.5	S >= 1.50
Policies	S < .7	.7 >= S < .9	.9 >= S < 1.9	S >= 1.90
Auditing	S < .7	.7 >= S < .9	.9 >= S < 1.9	S >= 1.90
Deliverables	S < .7	.7 >= S < .9	.9 >= S < 1.9	S >= 1.90

These values were arbitrarily assigned with no supporting evidence found during research, but based on the following logic: each N/A answer means the organization has not even considered this as a security risk; each No answer should be penalized; each Yes answer means that some form of security was implemented surrounding this question, or section. A high degree of N/A answers should countermand an equal number of Yes answers in that some number of areas were never identified for risk mitigation and could pose a catastrophic risk. A number of assessments should reveal a more accurate algorithm.

References

Chivers, H., & Fletcher, M. (2005). Applying Security Design Analysis to a Service-Based System. *Software—Practice & Experience, 35*(9), 873-897.

McCumber, J. (October 1991). Information Systems Security: A Comprehensive Model. *Proceedings 14th National Computer Security Conference.* National Institute of Standards and Technology, Baltimore, MD.

Patterson, T. and Blue, S. G. (2004). *Mapping Security: The Corporate Security Sourcebook for Today's Global Economy*. Addison-Wesley Professional.

Peltier, T. R. (2000). *Information Security Risk Analysis*. 1st. CRC Press, Inc.

Peltier, T. R. (September 2004). Risk Analysis and Risk Management. *The EDP Audit, Control, and Security Newsletter, 32*(3), 1-17.

Praxiom. (2005). *ISO IEC 17799 2000 Information Security Standard Translated into Plain English*. Retrieved from Praxiom research Group Limited on September 5, 2005 at: http://praxiom.com/iso-17799-2000.htm.

Appendix A: Java Source Code; Question.java

```
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
import javax.swing.BorderFactory.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeEvent;
import java.text.*;
import java.lang.Integer;

public class Question
{

 private String questionText;
 private JRadioButton yes;
 private JRadioButton no;
 private JRadioButton na;

 public Question( String text )
 {
   questionText = new String ( text );
   yes = new JRadioButton ( "Yes" );
   no  = new JRadioButton ( "No"  );
   na  = new JRadioButton ( "N/A", true );

 }

 public String getQuestionText()
 {
   return questionText;
 }

 public JRadioButton getYes()
 {
   return yes;
 }

 public JRadioButton getNo()
 {
   return no;
 }
```

```
 public JRadioButton getNa()
 {
   return na;
 }

 // return selected radio button
 // -1 = unknown state
 // 0  = yes
 // 1  = no
 // 2  = na
 public int getSelected()
 {
   if (yes.isSelected()) {
      return 0;
   }
   else if ( no.isSelected()) {
      return 1;
   }
   else if (na.isSelected()) {
     return 2;
   }
   else
     return -1;
 }


}
```

Appendix A: Java Source Code (cont.); Audit.java

```
/*************************************************************************
*****************/
/*                                                                       */
/* Created by Derek J. Sedlack                                           */
/* Address: 5043 Solar Point Drive                                       */
/*          Greenacres, Florida 33463                                    */
/* Email: Sedlack@Nova.edu                                               */
/* Web: scis.nova.edu/~sedlack                                           */
/*                                                                       */
/*************************************************************************
*****************/
/*                                                                       */
/* DISS 799 Information Security Management                              */
/* Dr. James Cannady                                                     */
/* Graduate School of Computer and Information Sciences                  */
/* Nova Southeastern University                                          */
```

```
/*                                                                      */
/************************************************************************
*****************/
/*                                                                      */
/* This program should help provide a measurement of ISO IEC 17799 2000 (BS 7799)
*/
/* adherence through a list of questions designed to ensure some coverage of          */
/* area predefined by the ISO initiative. The section covered by this phase is        */
/* policy and auditing. It will help determine if the organization has properly       */
/* considered methods of information assurance compliance and enforcement.            */
/*                                                                      */
/* Future consideration will include the matrix compiled Peltier for Deliverables     */
/* based on type (Int/Ext), Priority (A-D), and a list of predefined Controls.        */
/* The controls should be modified to appropriately conform to the organizations      */
/* specific requirements.                                               */
/*                                                                      */
/************************************************************************
*****************/
/*                                                                      */

/*                                                                      */
/************************************************************************
*****************/


import javax.swing.JOptionPane;
import javax.swing.JDialog;
import javax.swing.JButton;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.swing.BoxLayout;
import javax.swing.Box;
import javax.swing.border.Border;
import javax.swing.JTabbedPane;
import javax.swing.JPanel;
import javax.swing.JFrame;


import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
import javax.swing.BorderFactory.*;
```

```
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.text.*;
import java.lang.Integer;



public class Audit extends JPanel implements ActionListener
{

 // Arrays used for Access Control Lists
 private static String[] AccessControlList = { "1. Users.",
                                               "1.1 Internal Employees.",
                                               "1.2 External Employees.",
                                               "1.3 Partner Employees.",
                                               "1.4 Patrons.",
                                               "2. Creation and Maintenance.",
                                               "2.1 Defaults.",
                                               "3. Privileged Users.",
                                               "4. Conflicts.",
                                               "4.1 Rules.",
                                               "4.2 Distribution.",
                                               "4.3 Exceptions.",
                                               "4.4 Escalations.",
                                               "5. Mechanisms.",
                                               "5.1 Creation.",
                                               "5.2 Distribution." };

 // Arrays used for Policies
 private static String[] PoliciesList = {  "1. Creation and Maintenance.",
                                           "1.1 Defaults.",
                                           "1.2 Review Period.",
                                           "1.3 Review Committee.",
                                           "2. Users.",
                                           "2.1 Internal Employees.",
                                           "2.2 External Employees.",
                                           "2.3 Partner Employees.",
                                           "2.4 Patrons.",
                                           "3. Conflicts.",
                                           "3.1 Rules.",
                                           "3.2 Distribution.",
                                           "3.3 Exceptions.",
                                           "3.4 Escalations.",
                                           "4. Mechanisms.",
```

```
                                "4.1 Creation.",
                                "4.2 Distribution." };

  // Arrays used for Auditing
  private static String[] AuditingList = {  "1. Base.",
                                "1.1 State-based.",
                                "1.2 Transition-based.",
                                "2. Logger.",
                                "2.1 Mechanism Definition.",
                                "2.2 Format (Binary/Interpretable).",
                                "2.3 Integrity Assurance.",
                                "2.4 Sanitization.",
                                "3. Analyzer.",
                                "3.1 Detect Events.",
                                "3.2 Detect Non-events.",
                                "3.3 Detect Pattern/Symantics.",
                                "4. Notifier.",
                                "4.1 Response Required.",
                                "4.2 Notification Facility." };


  // Arrays used for Deliverables
  private static String[] DeliverablesList = {  "1. Information accessed by personnel not
intended to have access.",
                                "2. Unclear or non-existent versioning of the information.",
                                "3. Database could be corrupted by hardware failure,
incorrect, or bad software.",
                                "4. Data coul be corrupted by an incomplete transaction.",
                                "5. Ability to change data in transit and then change it back
in order to cover the activity.",
                                "6. A failure to report integrity issues.",
                                "7. Incompletely run process or failure to run a process that
could corrupt the data.",
                                "8. Lack of internal processes to create, control, manage
data across functions.",
                                "9. No notification of integrity problems.",
                                "10. Information being used in the wrong context or
environment.",
                                "11. Third-party information may have integrity issues.",
                                "12. Third-party access to information." };

  // Change the way the reporting chart worked
  // a little with only one item being used right now

  // The initial phase includes
  // Information Assurance Access Control
```

```
  private String bAccessControlX = "";
  private String bAccessControlXX = "";
  private String bAccessControlXXX = "";
  private String bAccessControlXXXX = "";

  // Information Assurance Policies / Compliance
  private String bPoliciesX = "";
  private String bPoliciesXX = "";
  private String bPoliciesXXX = "";
  private String bPoliciesXXXX = "";

  // Information Assurance Auditing
  private String bAuditingX = "";
  private String bAuditingXX = "";
  private String bAuditingXXX = "";
  private String bAuditingXXXX = "";

  // Information Assurance Deliverables
  private String bDeliverablesX = "";
  private String bDeliverablesXX = "";
  private String bDeliverablesXXX = "";
  private String bDeliverablesXXXX = "";

  // Wizard selection determinant
  private JTabbedPane tabbedPane;
  // Questions and option buttons
  private Question[] q1 = null;
  private Question[] q2 = null;
  private Question[] q3 = null;
  private Question[] q4 = null;

  private JLabel info = null;
  private boolean debug = true;


  public Audit()
  {

    super(new BorderLayout());

    JComponent panel0 = makeInformationPanel();
    tabbedPane = new JTabbedPane(JTabbedPane.TOP,
JTabbedPane.SCROLL_TAB_LAYOUT);


    // Information Panel
```

```
  tabbedPane.addTab("Information", null, panel0, "");
  tabbedPane.setToolTipTextAt(0, "");
  //ALT + I to jump to Information Page
  tabbedPane.setMnemonicAt(0, KeyEvent.VK_I);

  JComponent panel1= makeAccessControlPanel();
  tabbedPane.addTab("Access Control List", panel1);
  //ALT + V to jump to Valuation Page
  tabbedPane.setMnemonicAt(1, KeyEvent.VK_C);

  JComponent panel2 = makePoliciesPanel();
  tabbedPane.addTab("Policies List", panel2);
  //ALT + A to jump to Assessment Page
  tabbedPane.setMnemonicAt(2, KeyEvent.VK_P);

  JComponent panel3 = makeAuditingPanel();
  tabbedPane.addTab("Auditing List", panel3);
  //ALT + A to jump to Assessment Page
  tabbedPane.setMnemonicAt(3, KeyEvent.VK_A);

  JComponent panel4 = makeDeliverablesPanel();
  tabbedPane.addTab("Deliverables List", panel4);
  //ALT + A to jump to Assessment Page
  tabbedPane.setMnemonicAt(4, KeyEvent.VK_D);

  JComponent panel5 = makeFrameWorkPanel();
  tabbedPane.addTab("ISO 17799 FrameWork", panel5);
  //ALT + V to jump to Valuation Page
  tabbedPane.setMnemonicAt(5, KeyEvent.VK_F);

  tabbedPane.setPreferredSize(new Dimension(1000, 600));

  // Add the tabbed pane to this panel.
  add(tabbedPane);


}


public void actionPerformed(ActionEvent e) {
   if ( tabbedPane.getSelectedIndex() == 0 ) {
       tabbedPane.setSelectedIndex(1);
   }
   else if ( tabbedPane.getSelectedIndex() == 1) {
     tabbedPane.setSelectedIndex(2);
```

```
    }
    else if ( tabbedPane.getSelectedIndex() == 2) {
      tabbedPane.setSelectedIndex(3);
    }
    else if ( tabbedPane.getSelectedIndex() == 3) {
      tabbedPane.setSelectedIndex(4);
    }
    else if ( tabbedPane.getSelectedIndex() == 4) {
      tabbedPane.setSelectedIndex(5);
    }
      // was selected and how it all adds up
      // to build our final page output to the chart
      calculateRisk();

   }

  private JComponent makeInformationPanel()
  {
   JPanel panel = new JPanel(false);

   panel.setLayout(new BorderLayout());
   panel.setBorder(BorderFactory.createTitledBorder("Information Security Policy
Enforcement and Auditing"));

   JLabel info = new JLabel();

   String infoText = "<html><br>" +
                "<body bgcolor=e0e0e0>" +
                "Purpose:<br>" +
                "    The devise of this Framework is to help companies
validate and verify either<br>" +
                "    the existance of, or the accepted lack of policy
enforcement and auditing.<br>" +
                "    Future versions will include more precise questions
that reflect specific<br>" +
                "    functionality that overlays organizational
structure.<br>" +
                "    RESIZING IS BAD...it sometimes breaks the layout
for some reason and requires restart!<br>" +
                "<p><br>Usage:<br>" +
                "    From each Framework tab, select Yes, No, or N/A
depending on the appropriate<br>" +
                "    answer based on your organizations inclusion of
information security policies<br>" +
                "    that are enforced or audited.<br>" +
```

```
                "    Please answer: YES if you have implemented the item
questioned;<br>" +
                "    NO if you decided not to implement the item
questioned;<br>" +
                "    leave N/A if you had not considered the item
questioned.<br>" +
                "    Click on NEXT at the bottom and CALCULATE on
the final tab to show responding coverage.<br>" +
                "</body>";


    info.setText(infoText);
    panel.add(info, BorderLayout.NORTH);
    return panel;
  }

  private JComponent makeAccessControlPanel()
  {
//    JPanel panel = new JPanel(new GridLayout(2,1));
    JPanel panel = new JPanel(new BorderLayout());
    // create border around page with title
    panel.setBorder(BorderFactory.createTitledBorder("Access Control"));

    JPanel panel2 = new JPanel( new FlowLayout(FlowLayout.LEFT, 10, 10) );

    panel2.setLayout ( new GridLayout( 0,4 ) );
//    panel.setLayout(new FlowLayout(FlowLayout.LEFT,10,10));

    q1 = new Question[21];

    for (int x = 0; x < AccessControlList.length; x++) {
      q1[x] = new Question ( AccessControlList[x] );

      panel2.add ( new JLabel ( q1[x].getQuestionText() ) );

       //Create the radio buttons.
      ButtonGroup group = new ButtonGroup();
      group.add ( q1[x].getYes() );
      group.add ( q1[x].getNo() );
      group.add ( q1[x].getNa() );

      panel2.add ( q1[x].getYes() );
      panel2.add ( q1[x].getNo() );
      panel2.add ( q1[x].getNa() );
```

```
    }

    JButton next = new JButton ( "Next" );
    next.addActionListener(this);

    panel.add ( panel2 );
    panel.add ( next, BorderLayout.PAGE_END );

    return panel;
   }


  private JComponent makePoliciesPanel()
  {
    JPanel panel = new JPanel(new BorderLayout());
//    panel.setLayout(new BorderLayout());
    // create border around page with title
    panel.setBorder(BorderFactory.createTitledBorder("Corporate Policies"));

    JPanel panel2 = new JPanel( new FlowLayout(FlowLayout.LEFT, 10, 10) );

//    panel.setLayout(new FlowLayout(FlowLayout.LEFT,10,10));

    panel2.setLayout ( new GridLayout( 0,4 ) );

    q2 = new Question[21];

    for (int x = 0; x < PoliciesList.length; x++) {
      q2[x] = new Question ( PoliciesList[x] );

      panel2.add ( new JLabel (q2[x].getQuestionText()) );

       //Create the radio buttons.
      ButtonGroup group = new ButtonGroup();
      group.add( q2[x].getYes() );
      group.add( q2[x].getNo() );
      group.add( q2[x].getNa() );

      panel2.add (q2[x].getYes());
      panel2.add (q2[x].getNo());
      panel2.add (q2[x].getNa());


    }

    JButton next = new JButton ( "Next" );
```

```
    next.addActionListener(this);

    panel.add ( panel2 );
    panel.add ( next, BorderLayout.PAGE_END );
    return panel;
   }


   private JComponent makeAuditingPanel()
   {
    JPanel panel = new JPanel(new BorderLayout());
//    panel.setLayout(new BorderLayout());
    // create border around page with title
    panel.setBorder(BorderFactory.createTitledBorder("Auditing"));

    JPanel panel2 = new JPanel( new FlowLayout(FlowLayout.LEFT, 10, 10) );

    panel2.setLayout ( new GridLayout( 0,4 ) );
//    panel.setLayout(new FlowLayout(FlowLayout.LEFT,10,10));

    q3 = new Question[21];

    for (int x = 0; x < AuditingList.length; x++) {
      q3[x] = new Question ( AuditingList[x] );

      panel2.add ( new JLabel (q3[x].getQuestionText()) );

       //Create the radio buttons.
      ButtonGroup group = new ButtonGroup();
      group.add( q3[x].getYes() );
      group.add( q3[x].getNo() );
      group.add( q3[x].getNa() );

      panel2.add (q3[x].getYes());
      panel2.add (q3[x].getNo());
      panel2.add (q3[x].getNa());


    }

    JButton next = new JButton ( "Next" );
    next.addActionListener(this);

    panel.add ( panel2 );
    panel.add ( next, BorderLayout.PAGE_END );
```

```
    return panel;
  }


  private JComponent makeDeliverablesPanel()
  {
    JPanel panel = new JPanel(new BorderLayout());
//    panel.setLayout(new BorderLayout());
    // create border around page with title
    panel.setBorder(BorderFactory.createTitledBorder("Deliverables"));

    JPanel panel2 = new JPanel( new FlowLayout(FlowLayout.RIGHT, 70, 10) );

//    panel2.setLayout ( new GridLayout( 25,1 ) );
//    panel.setLayout(new FlowLayout(FlowLayout.LEFT,5,5));

    q4 = new Question[21];

    for (int x = 0; x < DeliverablesList.length; x++) {
      q4[x] = new Question ( DeliverablesList[x] );

      panel2.add ( new JLabel (q4[x].getQuestionText(), JLabel.LEFT) );

       //Create the radio buttons.
      ButtonGroup group = new ButtonGroup();
      group.add( q4[x].getYes() );
      group.add( q4[x].getNo() );
      group.add( q4[x].getNa() );

      panel2.add (q4[x].getYes());
      panel2.add (q4[x].getNo());
      panel2.add (q4[x].getNa());

    }

    JButton next = new JButton ( "Next" );
    next.addActionListener(this);

    panel.add ( panel2 );
    panel.add ( next, BorderLayout.PAGE_END );
    return panel;
  }


  private JComponent makeFrameWorkPanel()
  {
```

```
   JPanel panel = new JPanel(false);

   panel.setLayout(new BorderLayout());
   panel.setBorder(BorderFactory.createTitledBorder("Policy Enforcement and
Auditing"));

   JPanel panel2 = new JPanel( new FlowLayout(FlowLayout.LEFT, 10, 10) );

   // JLabel info = new JLabel();
   info = new JLabel();
   String infoText = getTableData();



   info.setText(infoText);
   panel.add(info, BorderLayout.NORTH);

   JButton calculate = new JButton ( "Calculate" );
   calculate.addActionListener(this);

   panel.add ( panel2 );
   panel.add ( calculate, BorderLayout.PAGE_END );
   return panel;
  }

  private String getTableData()
  {
    String infoText = "<html>" +
                      "<head>" +
                      "<Body>" +
                      "<Table align=center border=1 padding=0>" +
                      "<TD></TD>" +
                      "<TD colspan=4><Center>Protection Class</Center></TD>" +
                      "</TR>" +
                      "<TR>" +
                      "<TD><CENTER>ISO 17799 Framework </center></TD>" +
                      "<TD><CENTER> Class 1:   <br>" +
                      "     Inadequate <br>" +
                      "     Protection </CENTER></TD>" +
                      "" +
                      "<TD><CENTER> Class 2:   <br>" +
                      "     Minimal    <br>" +
                      "     Protection </CENTER></TD>" +
                      "" +
                      "<TD><CENTER> Class 3:   <br>" +
                      "     Reasonable <br>" +
```

```
                "     Protection </CENTER></TD>" +
                "" +
                "<TD><CENTER> Class 4:   <br>" +
                "     Adequate   <br>" +
                "     Protection </CENTER></TD>" +
                "</TR>" +
                "<TR>" +
                "<TD> Access Control </TD>" +
                "<TD> <center>" + bAccessControlX + "</center> </TD>" +
                "<TD> <center>" + bAccessControlXX + "</center> </TD>" +
                "<TD> <center>" + bAccessControlXXX + "</center> </TD>" +
                "<TD> <center>" + bAccessControlXXXX + "</center> </TD>" +
                "</TR>" +
                "<TR>" +
                "<TD> Policies </TD>" +
                "<TD> <center>" + bPoliciesX + "</center> </TD>" +
                "<TD> <center>" + bPoliciesXX + "</center> </TD>" +
                "<TD> <center>" + bPoliciesXXX + "</center> </TD>" +
                "<TD> <center>" + bPoliciesXXXX + "</center> </TD>" +
                "</TR>" +
                "<TR>" +
                "<TD> Auditing </TD>" +
                "<TD> <center>" + bAuditingX + "</center> </TD>" +
                "<TD> <center>" + bAuditingXX + "</center> </TD>" +
                "<TD> <center>" + bAuditingXXX + "</center> </TD>" +
                "<TD> <center>" + bAuditingXXXX + "</center> </TD>" +
                "</TR>" +
                "<TR>" +
                "<TD> Deliverables </TD>" +
                "<TD> <center>" + bDeliverablesX + "</center> </TD>" +
                "<TD> <center>" + bDeliverablesXX + "</center> </TD>" +
                "<TD> <center>" + bDeliverablesXXX + "</center> </TD>" +
                "<TD> <center>" + bDeliverablesXXXX + "</center> </TD>" +
                "</TR>" +
                "</Table>" +
/* this does not make sense still...
* will have to change appropriately when the
* measurements are solidified depending
* on reported coverage and specific items
* that require additional requirements
* Also have to consider the addition of the deliverables
* in a format that provides printing...
                "<TD>Legend</TD>" +
                "<TR>" +
                "<TD></TD>" +
                "<TD>no requirement</TD>" +
```

```
                "</TR>" +
                "<TR>" +
                "<TD>X</TD>" +
                "<TD>additional requirement based on implemented processes and
procedures</TD>" +
                "</TR>" +
                "<TR>" +
                "<TD>XX</TD>" +
                "<TD>additional requirement based on implementation of certified
products in at least half of the applicable product categories</TD>" +
                "</TR>" +
                "<TR>" +
                "<TD>XXX</TD>" +
                "<TD>additional requirement based on implementation of certified
products in all of the applicable product categories</TD>" +
                "</TR>" +
  */
                "<body>" +
                "</html>" +
                "</head>";
    return infoText;
  }

  /* Ok, this is truely hideous, but for this purpose it will work
   * If I don't change this for phase 2 I should be draw and shot
   * in public square in front of children in broad daylight ...
   */

  private void calculateRisk()
  {

    float totalAccessValues = 0, totalAccessItems = 0;
    float totalPoliciesValues = 0, totalPoliciesItems = 0;
    float totalAuditingValues = 0, totalAuditingItems = 0;
    float totalDeliverablesValues = 0, totalDeliverablesItems = 0;

//    System.out.println ("Real First totalValues" + totalValues);

    totalAccessValues = calculate(q1);
    totalPoliciesValues = calculate(q2);
    totalAuditingValues = calculate(q3);
    totalDeliverablesValues = calculate(q4);

//    System.out.println ("First totalValues" + totalValues);

    totalAccessItems = AccessControlList.length;
```

```
    totalPoliciesItems = PoliciesList.length;
    totalAuditingItems = AuditingList.length;
    totalDeliverablesItems = DeliverablesList.length;

    // See below for the twisted logic here
    totalAccessValues = totalAccessItems / totalAccessValues;
    totalPoliciesValues = totalPoliciesItems / totalPoliciesValues;
    totalAuditingValues = totalAuditingItems / totalAuditingValues;
    totalDeliverablesValues = totalDeliverablesItems / totalDeliverablesValues;

    System.out.println ("TotalAccessValues" + totalAccessValues + "totalAccessItems" +
totalAccessItems + "\n");
    System.out.println ("TotalPoliciesValues" + totalPoliciesValues + "totalPoliciesItems"
+ totalPoliciesItems + "\n");
    System.out.println ("TotalAuditingValues" + totalAuditingValues +
"totalAuditingItems" + totalAuditingItems + "\n");
    System.out.println ("TotalDeliverablesValues" + totalDeliverablesValues +
"totalDeliverablesItems" + totalDeliverablesItems + "\n");

    if ( totalAccessValues < .66 ) {
      bAccessControlX = "X";
      bAccessControlXX = "";
      bAccessControlXXX = "";
      bAccessControlXXXX = "";
    } else if ( totalAccessValues >= .66 && totalAccessValues < .8 ) {
      bAccessControlX = "";
      bAccessControlXX = "XX";
      bAccessControlXXX = "";
      bAccessControlXXXX = "";
    } else if ( totalAccessValues >= .8 && totalAccessValues < 1.5 ) {
      bAccessControlX = "";
      bAccessControlXX = "";
      bAccessControlXXX = "XXX";
      bAccessControlXXXX = "";
    } else {
      bAccessControlX = "";
      bAccessControlXX = "";
      bAccessControlXXX = "";
      bAccessControlXXXX = "XXXX";
    }

    if ( totalPoliciesValues < .7 ) {
      bPoliciesX = "X";
      bPoliciesXX = "";
      bPoliciesXXX = "";
      bPoliciesXXXX = "";
```

```
} else if ( totalPoliciesValues >= .7 && totalPoliciesValues < .9 ) {
  bPoliciesX = "";
  bPoliciesXX = "XX";
  bPoliciesXXX = "";
  bPoliciesXXXX = "";
} else if ( totalPoliciesValues >= .9 && totalPoliciesValues < 1.9 ) {
  bPoliciesX = "";
  bPoliciesXX = "";
  bPoliciesXXX = "XXX";
  bPoliciesXXXX = "";
} else {
  bPoliciesX = "";
  bPoliciesXX = "";
  bPoliciesXXX = "";
  bPoliciesXXXX = "XXXX";
}

if ( totalAuditingValues < .7 ) {
  bAuditingX = "X";
  bAuditingXX = "";
  bAuditingXXX = "";
  bAuditingXXXX = "";
} else if ( totalAuditingValues >= .7 && totalAuditingValues < .9 ) {
  bAuditingX = "";
  bAuditingXX = "XX";
  bAuditingXXX = "";
  bAuditingXXXX = "";
} else if ( totalAuditingValues >= .9 && totalAuditingValues < 1.9 ) {
  bAuditingX = "";
  bAuditingXX = "";
  bAuditingXXX = "XXX";
  bAuditingXXXX = "";
} else {
  bAuditingX = "";
  bAuditingXX = "";
  bAuditingXXX = "";
  bAuditingXXXX = "XXXX";
}

if ( totalDeliverablesValues < .7 ) {
  bDeliverablesX = "X";
  bDeliverablesXX = "";
  bDeliverablesXXX = "";
  bDeliverablesXXXX = "";
} else if ( totalDeliverablesValues >= .7 && totalDeliverablesValues < .9 ) {
  bDeliverablesX = "";
```

```
    bDeliverablesXX = "XX";
    bDeliverablesXXX = "";
    bDeliverablesXXXX = "";
  } else if ( totalDeliverablesValues >= .9 && totalDeliverablesValues < 1.9 ) {
    bDeliverablesX = "";
    bDeliverablesXX = "";
    bDeliverablesXXX = "XXX";
    bDeliverablesXXXX = "";
  } else {
    bDeliverablesX = "";
    bDeliverablesXX = "";
    bDeliverablesXXX = "";
    bDeliverablesXXXX = "XXXX";
  }

  info.setText(getTableData());

}
/**
 * Second Assignment with Arbitrary values!!!
 * They have to be though until context defines them!!!
 * Do calculations from all of the tabs (- the first and last)
 * to build our little chart
 * -1 means they didn't select anything (this is bad and they must be punished)
 * 0 means they have implemented the item
 * 1 means they did not implemented (at lease they are thinking about it, good for them!)
 * 2 means they are not even planning (again, punishment)
 *
 * Add up the values at this point for Dr. Cannady and determine
 * some meaningless arbitrary value that will later have real meaning
 * Well, make some algorithm anyway
 * n = number of questions for section
 * Ev = summation of the values for section
 * ex1. n = 10, 1/2 implemented = 0, 1/2 not = 5: 10/5 = 2
 * ex2. n = 10, 1/2 implemented = 0, 1/4 not = 2, 1/4 N/A = 6: 10/8 = 1.25
 * ex3. n = 10, 1/4 implemented = 0, 1/2 N/A = 10, 1/2 not = 3: 10/13 = 0.77
 *
 * I guess that lower numbers are a bad thing when determining using this method...
 *
 */
private int calculate(Question[] q)
{

  int totalValues = 0;

  if (q != null) {
```

```
    for (int i = 0; i < q.length; i++) {
      if (q[i] != null) {
        totalValues += q[i].getSelected();
      }
    }
  }


return totalValues;
}



/**
 * Create the GUI and show it. For thread safety, this method should be
 * invoked from the event-dispatching thread.
 */
private static void createAndShowGUI()
{
  // Make sure we have nice window decorations.
  JFrame.setDefaultLookAndFeelDecorated(true);

  // Create and set up the window.
  JFrame frame = new JFrame("17799 Framework");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  // Create and set up the content pane.
  JComponent newContentPane = new Audit();
  newContentPane.setOpaque(true); // content panes must be opaque

  frame.getContentPane().add(newContentPane, BorderLayout.CENTER);

  // Display the window.
  frame.pack();
  frame.setVisible(true);
}
public static void main(String[] args)
{
  // Schedule a job for the event-dispatching thread:
  // creating and showing this application's GUI.
  javax.swing.SwingUtilities.invokeLater(new Runnable()
  {
    public void run()
    {
      createAndShowGUI();
    }
```

```
    });
  }


}
```