Computer Security

Bell-LaPadula Implementation

By: Trae Folkman, Alex Krentz, Ethan Picklesimer, Manahari Dahal, Mario Elias, Caleb Fullmer

Control

```
/*****************************************************************
 * COMPONENT:
 *    CONTROL
 * Author:
 *    Br. Helfrich, Manahari Dahal
 * Summary:
 *    This class stores the notion of Bell-LaPadula
 *****************************************************************/

#pragma once

enum Control
{
    PUBLIC, CONFIDENTIAL, PRIVILEGED, SECRET
};
```

This code is the implementation of the Control header file, which specifies the security levels used within the program as an enum.

```
/*********************************************************
 * USER
 * All the users currently in the system
 *********************************************************/
struct User
{
   const char *name;
   const char *password;
   const Control subjectControl;
};


/*********************************************************
 * USERS
 * All the users currently in the system
 *********************************************************/
const User users[] =
{
   { "AdmiralAbe",     "password", SECRET },
   { "CaptainCharlie", "password", PRIVILEGED },
   { "SeamanSam",      "password", CONFIDENTIAL },
   { "SeamanSue",      "password", CONFIDENTIAL },
   { "SeamanSly",      "password", CONFIDENTIAL }
};
```

This code shows the addition of the security levels defined in the Control header file into the User struct and users array. This gives each user a specific level of access which determines the documents they can read and write to. Any documents the users create will also be set to their access level.

Asset Control

```
private:
   int id;                    // the unique ID of this message
   static int idNext;         // the id of the next message created
   bool empty;                // is this message empty / cleared?
   std::string text;          // the textual content of this message
   std::string author;        // the author of this message
   std::string date;          // the date this message was created
   Control assetControl;      // the control level
```

This code creates the private member variable of assetControl and is implemented in the message header file.

```
/*************************************************
 * MESSAGE NON-DEFAULT CONSTRUCTOR
 * Create a message and fill it
 *************************************************/
Message::Message(const string & text,
                 const string & author,
                 const string & date,
                 const Control assetControl)
{
   this->text = text;
   this->author = author;
   this->date = date;
   this->id = idNext++;
   this->assetControl = assetControl;
   empty = false;
}
```

The assignment of the assetControl variable is implemented in the message non-default constructor (Where the message is created).

Subject Control

```
private:
   Messages * pMessages;
   std::string userName;
   Control subjectControl;
```

This code creates the private member variable of subjectControl and is located in the interact header file.

```
/*********************************************
 * INTERACT constructor
 * Authenticat ethe user and get him/her all set up
 *********************************************/
Interact::Interact(const string & userName,
                   const string & password,
                   Messages & messages)
{
   this->subjectControl = authenticate(userName, password);
   this->userName = userName;
   this->pMessages = &messages;
}
```

This code assigns the subjectControl variable to the authenticate function in the interact.cpp file. This variable is then used to verify file access in multiple functions including show, display, add, update, and remove.

Security Condition

```
/*********************************************
 * MESSAGE :: SECURITY CONDITION READ
 * Check if the subject has read access.
 *********************************************/
bool Message::securityConditionRead(const Control assetControl, const Control subjectControl) const
{
   return subjectControl >= assetControl;
}
```

securityConditionRead is contained in the Message class as a predicate function. If the subjectControl is greater than or equal to the assetControl, it returns true and the user will have read access to the asset.

```
/*********************************************
 * MESSAGE :: DISPLAY TEXT
 * Display the contents or the text of the message
 *********************************************/
void Message::displayText(const Control subjectControl) const
{
   // determine if credentials exist
   if (!securityConditionRead(this->assetControl, subjectControl))
      cout << "You have insufficient rights to read this message!\n";
   else
   {
      cout << "\tMessage: "
         << text
         << endl;
   }
}
```

The securityConditionRead function is present in the displayProperties and displayText functions. These functions read the messages.txt file on the Message trust boundary.

```
/*********************************************
 * MESSAGE :: SECURITY CONDITION WRITE
 * Check if the subject has write access.
 *********************************************/
bool Message::securityConditionWrite(const Control assetControl, const Control subjectControl) const
{
   return subjectControl <= assetControl;
}
```

The securityConditionWrite function is also contained in the Message class as another control check. If the subjectControl is less than or equal to the assetControl, it returns true and the user will have write access to the asset.

```
/*********************************************
 * MESSAGE :: UPDATE TEXT
 * Update the contents or text of the message
 *********************************************/
void Message::updateText(const Control subjectControl, const string & newText)
{
   // determine if credentials exist
   if (!securityConditionWrite(this->assetControl, subjectControl))
      cout << "You have insufficient rights to write to this message!\n";
   else
   {
      text = newText;
   }
}
```

The securityConditionWrite function is present in the updateText and clear functions. These functions write to the messages.txt file on the Message trust boundary.

Tests Cases

Each user was tested against all possible actions.

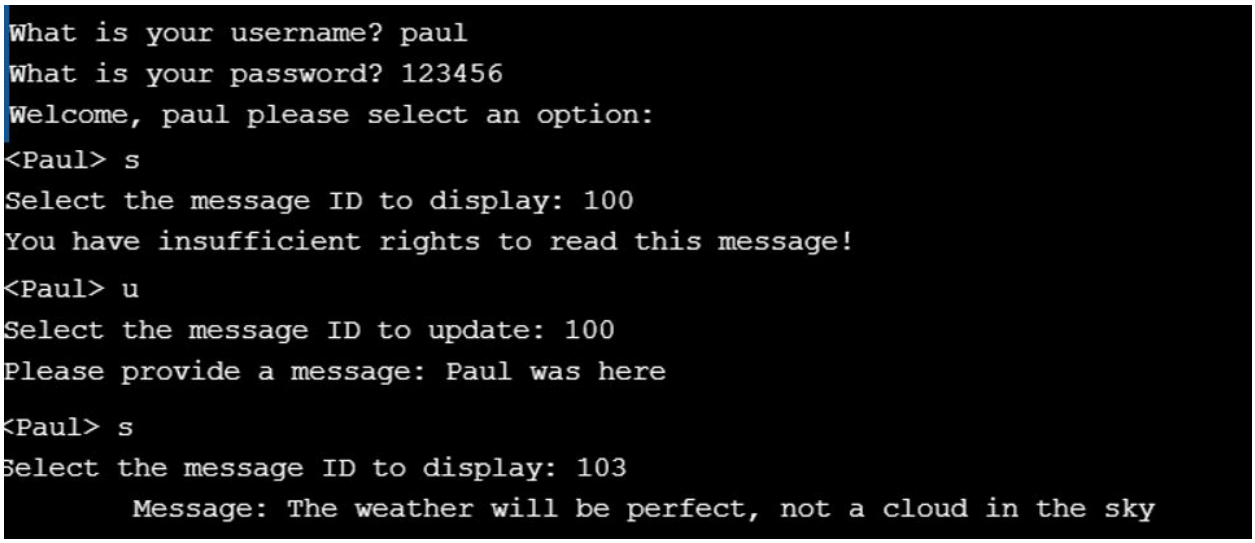
Read-down permitted: Users can view any document at or below their control level.

Write-down restricted: Users cannot write to any document below their control level.

Read-up restricted: Users cannot view any document above their control level.

Write-up permitted: Users can write to any document at or above their control level.

Paul's clearance is public: The below test cases show these privileges.



SeamanSly's clearance is confidential: The below test cases show these privileges.

```
What is your username? SeamanSly
What is your password? password
Welcome, SeamanSly please select an option:
<SeamanSly> s
Select the message ID to display: 103
        Message: The weather will be perfect, not a cloud in the sky
<SeamanSly> u
Select the message ID to update: 103
Please provide a message: SeamanSly was here
You have insufficient rights to write to this message!
<SeamanSly> s
Select the message ID to display: 106
You have insufficient rights to read this message!
<SeamanSly> u
Select the message ID to update: 106
Please provide a message: SeamanSly was here

<SeamanSly>
```

CaptainCharlie's clearance is privileged: The below test cases show these privileges.

```
What is your username? CaptainCharlie
What is your password? password
Welcome, CaptainCharlie please select an option:

<CaptainCharlie> s
Select the message ID to display: 103
        Message: The weather will be perfect, not a cloud in the sky

<CaptainCharlie> u
Select the message ID to update: 103
Please provide a message: CaptainCharlie was here
You have insufficient rights to write to this message!

<CaptainCharlie> s
Select the message ID to display: 109
You have insufficient rights to read this message!

<CaptainCharlie> u
Select the message ID to update: 109
Please provide a message: CaptainCharlie was here

<CaptainCharlie>
```

AdmiralAbe's clearance is secret: The below test cases show these privileges.

```
What is your username? AdmiralAbe
What is your password? password
Welcome, AdmiralAbe please select an option:
<AdmiralAbe> u
Select the message ID to update: 105
Please provide a message: Big Abe was here
You have insufficient rights to write to this message!
```