CSE 453 Week 12 Lab Report

Bell-LaPadula

**Group 1** | Ryan Brower | Bryce Sanders | Lance Riley | Harrison Parrish | Everett Tsosie | Jared Harper

Table of Contents

[Introduction 3](#_Toc152352144)

[Components 3](#_Toc152352145)

[Control 3](#_Toc152352146)

[Asset Control 3](#_Toc152352147)

[Subject Control 3](#_Toc152352148)

[Security Condition 3](#_Toc152352149)

[Test Cases 4](#_Toc152352150)

[Tests 4](#_Toc152352151)

[Output 5](#_Toc152352152)

[Test Case #1 5](#_Toc152352153)

[Test Case #2 5](#_Toc152352154)

[Test Case #3 5](#_Toc152352155)

[Test Case #4 5](#_Toc152352156)

[Test Case #5 5](#_Toc152352157)

[Test Case #6 5](#_Toc152352158)

[Test Case #7 5](#_Toc152352159)

[Test Case #8 6](#_Toc152352160)

[Test Case #9 6](#_Toc152352161)

[Test Case # 10 6](#_Toc152352162)

[Test Case #11 6](#_Toc152352163)

[Test Case #12 6](#_Toc152352164)

[Summary 7](#_Toc152352165)

[Demonstration Video 7](#_Toc152352166)

# Introduction

The team must implement Confidentiality assurances in a Python-based messaging program to be used by the military. This will be done using the Bell-LaPadula system, which uses a linear scale to assign clearance levels to both assets and users. This leads to access only being granted to users who possess equal or greater clearance than the given asset.

# Components

## Control

In the Bell-LaPadula system, the Control is the “scale” of clearance levels that can be assigned to users and assets. These levels *must* be assigned integer values that rise with the level of clearance. The implementation of the Control in the program follows the scale shown below:

|  |
| --- |
| class Control(Enum):  PUBLIC = 0  CONFIDENTIAL = 1  PRIVILEGED = 2  SECRET = 3 |

Note: Python does not have native support for Enumerations, so the “Enum” module must be used to implement Bell-LaPadula properly.

## Asset Control

Asset Control is simply the act of applying the Control clearance levels to the assets of the program: the messages. Previously stored messages are listed in the “messages.txt” file and begin with the clearance level:

*Public|President Franklin D. Roosevelt|29 December 1940|No man can tame a tiger into a kitten by stroking it.*

The listed clearance level is converted into its proper integer value when Confidentiality assurance is performed. When new messages are created, the user must enter the Control level that should be assigned to their message. This is implemented in the add() function of the “interact.py” file and looks similar to the following:

|  |
| --- |
| level = ""  while level not in ("PUBLIC", "CONFIDENTIAL", "PRIVILEGED", "SECRET"):  print("Available Levels:")  print(" Public, Confidential, Privileged, Secret")  level = self.\_prompt\_for\_line("level of control").upper()  asset\_control = ct.Control[level] |

## Subject Control

Like Asset Control, Subject Control boils down to adding Control clearance levels to the users, or “subjects”, of the program. This is accomplished by adding Control levels to the user list in the “User” class in the “interact.py”. This implementation follows the format shown below:

*"SeamanSam", "password", control.CONFIDENTIAL*

## Security Condition

The Security Condition is what the previously listed components have been building up to. With each asset and each subject now assigned a Control level, validation can now be performed to confirm whether a user has an equal or higher level of clearance to access an asset. This is implanted in both the reading and writing functions of the program and are implemented as follows:

|  |
| --- |
| def securityConditionWrite(subject, asset):  return subject <= asset  def securityConditionRead(subject, asset):  return subject >= asset |

To properly implement Bell-LaPadula, these security checks are performed at all trust boundary junctions.

# Test Cases

Example of message stored in the “messages.txt” file:

*Secret|J. Robert Oppenheimer|16 July, 1945|Now I am become Death, the destroyer of worlds.*

## Tests

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Setup | Test | Result |
| 1 | AdmiralAbe : Secret  Message : Public | READ. Show message 101 (public message) | Success |
| 2 | AdmiralAbe : Secret  Message : Public | WRITE. Update message 101 (public message) | Failure |
| 3 | CaptainCharlie : Privileged  Message: Secret | READ. Show message 106 (secret message) | Failure |
| 4 | CaptainCharlie : Privileged  Message: Confidential | WRITE. Update message 102 (confidential message) | Failure |
| 5 | SeamanSam : Confidential  Message: Public | READ. Show message 101 (public message) | Success |
| 6 | SeamanSam : Confidential  Message: Public | WRITE. Update message 101 (public message) | Failure |
| 7 | SeamanSam : Confidential  Message: Privileged | READ. Show message 108 (privileged message) | Failure |
| 8 | SeamanSam : Confidential  Message: Privileged | WRITE. Update message 108 (privileged message) | Success |
| 9 | PublicUser : Public  Message: Public | READ. Show message 101 (public message) | Success |
| 10 | PublicUser : Public  Message: Secret | READ. Show message 109 (secret message) | Failure |
| 11 | SeamanSam : Confidential  Display list of messages | READ. Display list of messages | Display 6 public or confidential messages |
| 12 | SeamanSam : Confidential  Message : Secret | WRITE. Add a message with control level Secret.  READ. Display list of messages | Success, SeamanSam also can’t view the message he just sent. |

## Output

### Test Case #1

A black background with white text

Description automatically generated

### Test Case #2

A screenshot of a computer screen

Description automatically generated

### Test Case #3

A black background with white text

Description automatically generated

### Test Case #4

A screenshot of a computer screen

Description automatically generated

### Test Case #5

A black background with white text

Description automatically generated

### Test Case #6

A screenshot of a computer screen

Description automatically generated

### Test Case #7

A black background with white text

Description automatically generated

### Test Case #8

A screenshot of a computer screen

Description automatically generated

### Test Case #9

A black background with white text

Description automatically generated

### Test Case # 10

A black background with white text

Description automatically generated

### Test Case #11

A screenshot of a computer

Description automatically generated

### Test Case #12

A computer screen shot of a message

Description automatically generated

# Summary

Implementation of Confidentiality assurances in the program was successfully accomplished using the Bell-LaPadula system. The relatively simple scale of clearance levels offers a high level of Confidentiality, locking all users out of viewing assets above their assigned Control level. While implementation of Bell-LaPadula is trickier in Python due to a lack of native support for Enumerations, it is still possible using the “Enum” Python module.

While the level of Confidentiality that the Bell-LaPadula system offers is satisfactory, the level of Integrity it creates is non-existent. The use of this system raises concerns with public users being able to alter secret messages even though they cannot view them. Implementation of a system that offers higher levels of integrity would be best for a military-grade program such as this application.

# Demonstration Video

<https://youtu.be/8Z2w605flGo>