CYBR 510 Assignment 4

Mark Heckman

Contents

Introduction

Continuing the scenario from the Module 3 assignment, the client's software engineers have implemented the ABA using your security requirements. In this assignment, you will develop security test scenarios and cases, and conduct the tests, to make sure that the system satisfies some of the security requirements.

How to Install and Run the "Deliberately Bad" ABA

You are given an interface specification document that lists all of the ABA commands and a working (to some degree of working) ABA program (the "Deliberately Bad" ABA - DBABA) for testing. The program is written in Python and must be run on a Linux system that has a working Python 3 interpreter. Download the zip file "M4-dbaba-2024.zip" [link on Canvas] to the Linux system and extract the files. Further installation instructions are in the "ReadMe" file in the zip file. The interface specification document is "M4-ABA_CL_Spec" [link on Canvas].

The Security Requirements to Test

Not everyone will have created the same security requirements in the Module 3 assignment so, to standardize the assignment, I have provided a few security requirements here for which you are to develop test scenarios and test cases and carry out the tests.

Security Requirement	**Threats Mitigated**
SR-1 Login credential complexity - The system shall require that the login credentials are non-trivial and unpredictable, to mitigate password guessing, dictionary and other brute-force attacks.	T-2.1
SR-2 Account lockout - After three consecutive incorrect login attempts, the system shall generate an alarm and also lock-out the account for 1 minute, to mitigate dictionary and other brute-force attacks.	T-2.2
SR-3 Authentication – The system shall require users to authenticate themselves before granting access to system resources.	T-1.*
SR-4 Access rights - The system shall only allow access to system resources to authenticated users with the proper authorization.	T-1.*, T-4.*,T-5.1
SR-5 Sensitive Information Protection – The system shall protect sensitive information from unauthorized disclosure while it is stored or in transit.	T-7.*, T-11.*

Security Test Scenarios and Test Cases Spreadsheets

Use the spreadsheet format for scenarios and cases shown in the presentations, which cite the following examples and templates:

Test Scenarios: How to Review SRS Document and Create Test Scenarios – Software Testing Training Course on Live Project – Day 2 (softwaretestinghelp.com) and the template at Test-scenarios_Orange-HRM-3.0-SoftwareTestingHelp.xlsx (live.com).

Test Cases: Writing Test Cases From SRS Document: DOWNLOAD Sample Test Cases (softwaretestinghelp.com), Sample Test Case Template with Test Case Examples [Download]

(softwaretestinghelp.com), and an example Test-Cases-for-OrangeHRM-SoftwareTestingHelp.xlsx (live.com).

Assignment 4 Deliverables

Your deliverable for this project is a professional-quality report suitable for presentation to your client. The report must include the following sections:

1. Test scenarios and test cases for all of the security requirements listed above. Use the test scenario and test case spreadsheet formats that were mentioned in the presentations.
    a. Note: There is one serious, but not-immediately obvious bug in the LIN command that I know about that can give any user who has a valid login access to any other account, including "admin". That is a clear violation of SR-3. You can get 10 extra credit points if you include a test scenario/case that demonstrates that flaw.
2. Run the tests and create test execution spreadsheets (these are the test cases spreadsheets plus "status" and "actual result" columns).
3. A defect report:
    a. Note any deficiencies in the program where you find that it does not correctly implement the security requirements. For each such deficiency, determine if:
        i. The deficiency is in the interface specification (for example, does the specification fully support SR-1?), or
        ii. The interface specification is correct, but the program does not correctly implement the interface specification.
4. The output of a run of the Pylint static code checker (https://pylint.pycqa.org/en/latest/). Conduct a static code test on the "dbaba.py" file, save the output, and turn that in. (Unless you have programming experience, the output won't make a lot of sense to you, but I want you to know what a static checker is and to try one out.)

There is no minimum or maximum page limit on this document. I only hope that you will try to make it as professional as you can.

If you have questions, don't hesitate to ask your instructor for further clarification.

Hints

A few hints about how to go about creating test scenarios and cases:

1. Think like an attacker. What would an attacker try to do to violate the security objectives?
2. Start by testing obvious things. Don't assume that obvious things work correctly before you move on to more sophisticated testing.
3. Look for assumptions to violate. The specification writer made some assumptions about how the program would be used and so there are error cases that are not specified, which means that, even if the program correctly implements the specification, the program does not necessarily handle those cases in a secure way.
4. Focus the majority of testing on security-relevant commands. Many of the security requirements you are asked to test depend on the correct function of the "Login" command, for example.

ABA Implementation Details

Here are some implementation details about the ABA that might help you with your testing:

1. Configuration data for the ABA is stored in the "abacfg.py" file. Among other things, this file defines the file names for the various datafiles used by the ABA to store passwords, audit data, and the actual address database.
2. The password file is created the first time that the ABA is run, when the administrator chooses a password. Each line of the file contains 2 fields separated by a colon (":"). The first field is the user name and the second field is the salted and hashed password. (See https://en.wikipedia.org/wiki/Bcrypt for more information about the bcrypt function used to calculate the password hash.)
3. To completely reset the ABA (e.g., in case you forget the administrator password), delete all of the files that don't end with ".py". That will erase all of the stored data, including address book records, passwords, and audit data.