Jeffrey M. Borkowski
13 Oct 2015
CEG-3110
Homework 4

Introduction

This program component provides a reusable segment of computer code to check the strength of a candidate password against a set of defined rules. The program component should output an indication of whether or not the candidate password is accepted or rejected. By convention, if a password is not rejected, then it will be accepted. If a candidate password is rejected, the program component should also indicate the reason why the password is not valid.

Requirements

1. The program shall reject any new password that has fewer than 2 upper-case English letters.

2. The program shall reject any new password that has fewer than 2 lower-case English letters.

3. The program shall reject any new password that has fewer than 2 numerals (0 through 9).

4. The program shall reject any new password that has fewer than 2 special characters.

    4.1 The program shall recognize and accept special characters from the inclusive list:
    ! @ # $ % & *) ( ] [} { < > ; : . , / | \ ~ ? _ - + =

5. The program shall reject any new password that has fewer than 9 total characters.

6. The program shall reject any new password that has more than 24 total characters.

7. The program shall reject any new password that contains blank spaces.

8. The program shall reject any new password that is similar to at least 1 of the two previous passwords. The quality “similar to” is defined as the new password containing any 5-character substrings that are replicated in a case insensitive manner, either forward or backward, in the previous passwords.

9. The program shall reject any new password that contains characters other than upper-case and lower-case English letters, numerals, the special characters listed in requirement 4.1.

10. The program shall accept any new password that is not rejected.

11. The program shall output an indication of whether the new password is accepted or rejected.

    11.1 The program shall, if rejecting a new password, output an indication of the reason why the password is invalid.

12. The program shall be structured in an encapsulated manner that supports reuse across multiple computer programs.

Jeffrey M. Borkowski
13 Oct 2015
CEG-3110
Homework 4

Orthogonal Arrays and Pairwise Coverage Testing

Introduction

This document describes how the concepts of orthogonal arrays and pairwise coverage testing may be used to generate test cases for the password strength checker program component. Then a set of test cases created using these techniques is presented in a standard ready-to-run format. Overall, these test cases are designed to help validate the ability of the program component to operate in satisfaction of the requirements listed above. During the execution of each test case, the tester will be required to input three predefined passwords: one as the candidate new password and two more as the user's previous passwords. The tester will then need to monitor the program output for proper indications of password acceptance or rejection.

Given a software component with multiple inputs, or multiple aspects of a single input, exhaustive testing quickly becomes infeasible and a new testing strategy needs to be formulated. One such candidate utilizes the notion that many of the errors in software projects arise as the result of a single faulty component or from the interaction between a pair of components. Known as pairwise testing, this candidate methodology seeks to test all pairs of inputs, or aspects of a single input, in order to recognize and isolate bugs in the software component. The question that remains is how to efficiently generate these pairings of inputs for use in testing the software project.

The mathematical concept of orthogonal arrays provides a concise framework for identifying the required input combinations. Fundamentally, an orthogonal array provides maximal pairwise coverage while minimizing the number of included permutations of the input. For testing, this means that each permutation provides a unique piece of information about the system's response and all pairings of inputs are tested at least once. Before generating an orthogonal array, the test developer must identify the various input factors and potential levels for each factor. The obvious choice for the input factors are the equivalence classes previously identified for the software component, and the partitions of the equivalence class become the potential levels for the factor.

After identifying these factors and levels, there are many open-source programs available for use to generate an orthogonal array. One particularly useful tool is the ALLPAIRS Test Case Generation Tool available from Satisfice at http://www.satisfice.com/tools.shtml. This tool formulates the orthogonal array and presents the set of cases, based on the factors and levels, which provide full pairwise coverage. The important thing to note is that, depending on the number of levels for each factor, certain pairs of inputs may be duplicated in order to facilitate full pairwise coverage of the input space. Once a set of test cases that provides full pairwise coverage is obtained, the semantics of the input factors can be applied. These semantics allow the test developer to remove or modify test cases that would be physically impossible. Finally, the specific test cases can be specified by selecting actual inputs that meet the conditions specified for particular combination of factors and levels.

In terms of applying the orthogonal array and pairwise coverage testing concepts to the password strength checker program component, the factors and levels are based on the equivalence classes shown in the diagram below.

Password Length

8 or fewer characters	Between 9 and 24 characters, inclusive	25 or more characters
Invalid	Valid	Invalid

Number of lower case letters

0 or 1 characters	2 or more characters
Invalid	Valid

Number of upper case letters

0 or 1 characters	2 or more characters
Invalid	Valid

Number of numerals

0 or 1 characters	2 or more characters
Invalid	Valid

Number of special characters

0 or 1 characters	2 or more characters
Invalid	Valid

Number of invalid symbols

Zero invalid characters	1 or more invalid characters
Valid	Invalid

Similar to Previous Passwords

1 or more matching 5-character substring found in at least one previous password	No matching 5-character substrings found in either previous password
Invalid	Valid

Based on these equivalence classes, the following table summarizes the input factors and levels that will be used by the orthogonal array tool. The column headers identify the factors and the rows in each column identify the potential levels for that factor. Note that the “similar to previous password” class is broken into two factors, in recognition of the fact that a new password may be similar to one, both, or neither of the previous two passwords. Provisions are not made to explicitly test for similarity between passwords when the case of letters is matched exactly and when the case of letters is not matched. This particular level of distinction will be covered by the process of selecting actual passwords that meet the specified conditions.

Length	Lower	Upper	Numerals	Specials	Invalids	Similar 1	Similar 2
Low	Low	Low	Low	Low	Sufficient	None	None
Sufficient	Sufficient	Sufficient	Sufficient	Sufficient	High	Forward	Forward
High						Reverse	Reverse

After running the AllPairs test case generation tool with this set of factors and levels, the following table specifies the setup of test cases that provides full pairwise coverage of the input space.

Case	Length	Lower	Upper	Numerals	Specials	Invalids	Similar 1	Similar 2
1	Low	Low	Low	Low	Low	Sufficient	Reverse	Reverse
2	Low	Sufficient	Sufficient	Sufficient	Sufficient	High	None	None
3	Sufficient	Sufficient	Low	Low	Sufficient	Sufficient	Reverse	None
4	Sufficient	Low	Sufficient	Sufficient	Low	High	None	Reverse
5	High	Sufficient	Low	Low	Low	High	Forward	Forward
6	High	Low	Sufficient	Sufficient	Sufficient	Sufficient	Forward	Forward
7	Low	Sufficient	Low	Sufficient	Sufficient	High	Reverse	Reverse
8	Low	Low	Sufficient	Low	Low	Sufficient	None	None
9	Sufficient	Low	Sufficient	Low	Sufficient	High	Reverse	Forward
10	High	Sufficient	Low	Sufficient	Low	Sufficient	None	Reverse
11	High	Low	Sufficient	Low	Sufficient	High	Forward	None
12	Low	Sufficient	Low	Sufficient	Low	Sufficient	Forward	Forward
13	Sufficient	Low	Low	Sufficient	Low	High	Forward	Reverse
14	High	Sufficient	Low	Low	Sufficient	Sufficient	Reverse	Reverse
15	Low	Sufficient	Sufficient	Low	Low	High	None	Forward

Note that cases 1 and 2, as currently written, represent impossible situations after applying the semantics of each factor-level combination. In case 1, since the count of each valid symbol type is 0 or 1 and no invalid symbols are used, the new password would have a maximum length of 4 characters, thus making it impossible to have a matching substring of at least 5 characters. On the other hand, the new password for case 2 has a minimum length of 9 characters, since it has at least 2 of each valid symbol type and at least 1 invalid symbol, thus making it impossible for the password to contain 8 or fewer characters. Also note that the generation of a full pairwise coverage test setup inadvertently omitted a case which tests an acceptable new password. By modifying cases 1 and 2, it is possible to test an acceptable new password, while still providing full pairwise coverage. The revised versions of cases 1 and 2 are given in the table below.

Case	Length	Lower	Upper	Numerals	Specials	Invalids	Similar 1	Similar 2
1*	Sufficient	Sufficient	Sufficient	Sufficient	Sufficient	Sufficient	None	None
2*	Low	Sufficient	Low	Sufficient	Low	Sufficient	Reverse	Reverse

The specific new passwords included in each test case are generated such that they satisfy the conditions contained in the 8-tuple of factor-level values obtained from the orthogonal array.

Test Cases

The following are the test cases created based on the orthogonal array described above which provides full pairwise coverage in terms of the characteristics of the new password. Each test case has an associated 8-tuple of factor-level values corresponding to entries in the orthogonal array. These 8-tuples all have the following format:

(Length, Lower, Upper, Numerals, Specials, Invalids, Similar 1, Similar 2)

where values qualitatively describe the overall password length, the counts of lower case, upper case, numerals, special characters, and invalid symbols, and the type of match with the previous two passwords for a given new password.

Test Case #1

This test case will verify the ability of the program component to accept a valid new password, based on the 8-tuple (sufficient, sufficient, sufficient, sufficient, sufficient, sufficient, none, none).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. AaBb12!@Z
2. BbCc23@#Y
3. CcDd34#$X

Expected output:

Password accepted.

Requirement(s) tested: 4.1, 10, 11, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #2

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (low, sufficient, low, sufficient, low, sufficient, reverse, reverse).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. !2Ccb34
2. BbCc2!3@#Y
3. CcDd34#$43bcC

Expected output:

Password rejected. Password length less than 9 characters. Fewer than 2 upper case letters. Fewer than 2 valid special characters. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 1, 4, 4.1, 5, 8, 11, 11.1, 12

Actual output:

Result: Pass ______ Fail ______

Test Case #3

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (sufficient, sufficient, low, low, sufficient, sufficient, reverse, none).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. 1bBaa&#g!
2. AaBb12!@Z
3. CcDd34#$X

Expected output:

Password rejected. Fewer than 2 upper case letters. Fewer than 2 numerals. Matching string found in first previous password.

Requirement(s) tested: 1, 3, 4.1, 8, 11, 11.1, 12

Actual output:

Result: Pass ______ Fail ______

Test Case #4

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (sufficient, low, sufficient, sufficient, low, high, none, reverse).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. @32CCB g5
2. CcDd34#$X

3. BbCc23@#Y

Expected output:

Password rejected. Fewer than 2 lower case letters. Fewer than 2 valid special characters. Password contains blank spaces or invalid characters. Matching string found in second previous password.

Requirement(s) tested: 2, 4, 4.1, 7, 8, 9, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #5

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (high, sufficient, low, low, low, high, forward, forward).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. Ccdd3b\r'j klmnorewitwtfht
2. CcDd34#$X
3. AcDd3b2!@Z

Expected output:

Password rejected. Password length greater than 24 characters. Fewer than 2 upper case letters. Fewer than 2 numerals. Fewer than 2 valid special characters. Password contains blank spaces or invalid characters. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 1, 3, 4, 4.1, 6, 7, 8, 9, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #6

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (high, low, sufficient, sufficient, sufficient, sufficient, forward, forward).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. HGAaBB123@#Y4UR%$T!NS3CUR
2. AaBb12!@Z
3. BbCc23@#Y

Expected output:

Password rejected. Password length greater than 24 characters. Fewer than 2 lower case letters. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 2, 4.1, 6, 8, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #7

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (low, sufficient, low, sufficient, sufficient, high, reverse, reverse).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. $#43dxX'
2. CcDd34#$X
3. Qw3$XXd34o8&Lk0(Cv4%Nb7!

Expected output:

Password rejected. Password length less than 9 characters. Fewer than 2 upper case letters. Password contains blank spaces or invalid characters. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 1, 4.1, 5, 7, 8, 9, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #8

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (low, low, sufficient, low, low, sufficient, none, none).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. 4&QWERTY
2. Qw3$As1@Po8&Lk0(Cv4%Nb7!
3. AaBb12!@Z

Expected output:

Password rejected. Password length less than 9 characters. Fewer than 2 lower case letters. Fewer than 2 numerals. Fewer than 2 valid special characters.

Requirement(s) tested: 2, 3, 4, 4.1, 5, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #9

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (sufficient, low, sufficient, low, sufficient, high, reverse, forward).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. QwUT=+4#$XY$AS@PO'LK(C%N
2. PQrs14+=TUvw25?_XYyz36~]
3. CcDd34#$XY

Expected output:

Password rejected. Fewer than 2 lower case letters. Fewer than 2 numerals. Password contains blank spaces or invalid characters. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 2, 3, 4.1, 7, 8, 9, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #10

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (high, sufficient, low, sufficient, low, sufficient, none, reverse).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. Qwut#43ddxy4as2po5lk9c5n8
2. PQrs14+=TUvw25?_XYyz36~]
3. CcDd34#$X

Expected output:

Password rejected. Password length greater than 24 characters. Fewer than 2 upper case letters. Fewer than 2 valid special characters. Matching string found in second previous password.

Requirement(s) tested: 1, 4, 4.1, 6, 8, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #11

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (high, low, sufficient, low, sufficient, high, forward, none).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. QrS1$$#DDXY$AS@PO%LK(%'*
2. PQrs1$+=TUvw25?_XYyz36~]
3. CcDd34#$X

Expected output:

Password rejected. Password length greater than 24 characters. Fewer than 2 lower case letters. Fewer than 2 numerals. Password contains blank spaces or invalid characters. Matching string found in first previous password.

Requirement(s) tested: 2, 3, 4.1, 6, 7, 8, 9, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #12

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (low, sufficient, low, sufficient, low, sufficient, forward, forward).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. Pqrs14#4
2. PQrs14+=TUvw25?_XYyz36~]
3. CcDs14#4$X

Expected output:

Password rejected. Password length less than 9 characters. Fewer than 2 upper case letters. Fewer than 2 valid special characters. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 1, 4, 4.1, 5, 8, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #13

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (sufficient, low, low, sufficient, low, high, forward, reverse).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. S14+0x8 7’9
2. PQrs14+0TUvw25?_XYyz36~]
3. CcDd8X0+4#$

Expected output:

Password rejected. Fewer than 2 lower case letters. Fewer than 2 upper case letters. Fewer than 2 valid special characters. Password contains blank spaces or invalid characters. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 1, 2, 4, 4.1, 7, 8, 9, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #14

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (high, sufficient, low, low, sufficient, sufficient, reverse, reverse).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. X$#4hzyyx_j)f#iu!w@[(cn]/
2. PQrs14+=TUvw25?_XYyz36~]
3. CcDd3h4#$X

Expected output:

Password rejected. Password length greater than 24 characters. Fewer than 2 upper case letters. Fewer than 2 numerals. Matching string found in first previous password. Matching string found in second previous password.

Requirement(s) tested: 1, 3, 4.1, 6, 8, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______

Test Case #15

This test case will verify the ability of the program component to reject a new password with multiple rejection reasons, based on the 8-tuple (low, sufficient, sufficient, low, low, high, none, forward).

Initial conditions: A new password strength checker object is available and ready to test a candidate password.

Required inputs:

1. CcDG3'!h
2. PQrs14+=TUvw25?_XYyz36~]
3. CcDG3d4#$X

Expected output:

Password rejected. Password length less than 9 characters. Fewer than 2 numerals. Fewer than 2 valid special characters. Password contains blank spaces or invalid characters. Matching string found in second previous password.

Requirement(s) tested: 3, 4, 4.1, 5, 7, 8, 9, 11, 11.1, 12

Actual output:

____

Result: Pass ______ Fail ______