CS547 Project update 1

Name	Roll
Ravi Kishan	1701CS39
Vivek Kumar Shaw	1701CS55
Shrish Chandra Sharma	1701CS58

Tools for vulnerability detection

We have done the following analysis of tools and techniques used for vulnerability detection

Static Analysis:

White box testing

Introduction :- White box testing techniques is one of the most important and prevalent software testing techniques, it is typically very effective in validating design, decision, assumptions and finding programming errors and implementation errors in software.White box testing based on analysis of internal workings and structure of a piece of software and it can uncover implementation errors such as poor key management.White box testing is performed based on the knowledge of how the system is implemented. White box testing is very effective and efficient in validating design decisions and assumptions.

Advantages :-

- it reveals error in hidden code
- it helps in removing extra lines of code

Disadvantages :-

- it is very expensive and some of the codes omitted in the code could be missed out.

Working Process of White Box Testing Technique:-

The general outlining of white box testing process is described below:

Step 1: **Input**

- Requirement
- Functional specification
- Detailed designing of documents
- Proper source code
- Security specifications are must

Step 2: **Processing Unit (Analyzing internal working only)**

- Perform risk analysis to guide whole testing process
- Proper test plan
- Execute test cases and communicate results

Step 3: **Output**

- Prepare final report

Some important types of white box testing are:-

- Control Flow Testing
- Branch Testing
- Basis Path Testing
- Data Flow Testing
- Loop Testing

We have analysed below testing techniques one by one :-

Branch Testing

It makes sure that each possible outcome from the condition is tested at least once. Branch testing however, has the objective to test every option (either true or false) on every control statement which also includes compound decision (when the second decision depends upon the previous decision).

In branch testing, test cases are designed to exercise control flow branches or decision points in a unit. All branches within the branch are tested at least once

Example of Branch Testing:

```
Read Length (l)
Read Breadth (b)
IF l > 4 THEN
Print “invalid length”
ENDIF
IF b > 5 THEN
Print “invalid breadth”
ENDIF
AREA= (l*b)
Print (“Area=”Area)
IF Area>12 THEN
Print “False Area”
ELSEIF Area<=12 and Area>=1 THEN
Print “True Area”
ELSEIF Area<1 THEN
Print “AreaNot Defined”
ENDIF
```

Loop Testing

Loop testing is another type of white box testing which exclusively focuses on the validity of loop construct. Loops are simple to test unless dependencies exist between the loops or among the loop and the code it contains. There are four classes of loops:

- Simple Loop
- Nested Loop
- Concatenated Loop
- Unstructured Loop

Data Flow Testing

Data flow testing is another type of white box testing which looks at how data moves within a program. In data flow testing the control flow graph is annotated with the information about how the program variables are defined and used. (Control Flow graph is a diagrammatic representation of a program and its execution)

Data flow testing picks enough paths to assure that:

- **Every data object has been initialized prior to its use**
- **All defined objects have been used at least once**

Some of the important points of data flow testing are:

- **All data flow anomalies are resolved.**
- **Avoid integration problems by doing all data flow operations on a variable within the same routine.**
- **When possible use explicit (rather than implicit) declaration of data.**

A number of data flow testing strategies have been studied and compared i.e. [NTA88], [FRA93].Data flow testing tends to uncover bugs like variable used but not initialized or declared but not used, so on

Conclusion:-

White box testing is verification and validation technique which software engineers can use to examine their code works as expected. I have described the working process and the various white box testing techniques such as Control Flow Testing, Data Flow Testing, Branch Testing and Loop Testing.

Using white box testing techniques , a software engineer can design test cases that

- **Exercise internal data structure to ensure their validity.**
- **Exercise logical decisions on both their true and false side.**
- **Exercise independent path within a module.**
- **Execute loops at their boundaries and within their operational bounds.**

Testers need to understand the white box techniques that are available to make educated decisions about their use for the specific system we are currently, and in future, will be testing.

DynamicAnalysis/Runtime Testing:

Black box testing

Introduction :-

Black box testing focuses on the functional requirement of the software. Black box testing is an integral part of correctness testing but its ideas are not limited to correctness testing only.

The tester, in black box testing only knows about the input (process by a system) and required output, or in the other word tester need not know the internal working of the system.

Black box testing occurs throughout the software development life cycle and software testing life cycle i.e. in regression testing, acceptance testing, unit testing, integration testing and system testing stage. The types of testing under this technique are totally focused on the testing for functionality of the software applications.

An ideal example of a BBT system would be a search engine, in which we enter text that we want to search for and get the result, we do not know or see the specific process that is being employed to obtain our result. (E.g. Input → search → Output) (No internal process)
Black box testing tools are mainly record and playback tools which record test cases in the form of some scripts like Perl, TSL, VB script, JAVAscript.
Factors which we should take into consideration at the time of selecting BBT Tool:

- Ease of use
- Cost Accuracy
- Test coverage
- Test completeness
- Reporting capability
- Capacity of vulnerability database

Advantages :-

- the designer and tester are independent of each other the test is unbiased
- the test is not done from the point of view of designer it is rather done from the point of view of user
- it is easier for tester to create test cases by simply working through the application, as would an end user
- the tester does not need knowledge of any specific programming language as they do not have to concern themselves with the inner working of an application
- more effective on larger units of code than white box testing
- quicker test case development as the tester only concerns them with the graphical user interface

Disadvantages :-

- script maintenance is very difficult as black box tools are relevant on the method of input being known
- test cases are difficult to design without clear and concise specification

Working Process of White Box Testing Technique:-
Below are the steps which explain the working process of Black Box Testing

Step 1 **Input:**

Requirement and functional specification of the system are examined. High level design documents and application block source code are also examined. Tester chooses valid input and rejects the invalid inputs

Step 2 **Processing Unit**:

Do not concern with the internal working of the system. In the processing unit tester constructs test cases with the selected input and executes them. Tester also performs load testing, stress testing, security review and globalization testing. If any defect is detected it will be fixed and re-tested

Step 3 **Output:**

After all these testing, tester gets desired output and prepares final report

Some important types of white box testing are:-

- Equivalence Partitioning
- Boundary Value Analysis
- Fuzzing
- Static transition technique
- All-Pair Testing

We have analysed all listed black box testing techniques one by one :-

Equivalence Partitioning:
Equivalence partitioning is a black box testing method that divides the input data of a software unit into partitions of data from which test cases can be derived. It reduces no. of test cases. In equivalence class partitioning an equivalence class is formed of the inputs for which the behavior of the system is specified or expected to be similar. An equivalence class represents a set of valid or invalid states for input conditions. Typically, an input condition is either a specific numeric value, array of values, a set of related values or Boolean condition. Once we select equivalence classes for each of the input, the next issue is to select the test cases suitably.
Now, let us consider an example of equivalence class partitioning:
For a program that takes two inputs, a string 'C' of length up to M and integer 'P'. The program is determine the top P highest occurring character in C

Input	Valid E C	Invalid E C
C	1:Contains numbers 2:Contains lower case letter 3:Contains upper case letter 4:Contains special character 5:String length between O - M	1:Non-ASCII character 2:String length>M
P	6:Integer in valid range	3:Integer out of range

Boundary Value Analysis:
Boundary value analysis is a testing technique that focuses more on testing at boundaries or where the extreme boundary values are chosen. Boundary values include maximum, minimum, just inside/outside boundaries, typical values and error values.
Suppose in boundary value analysis each input values has defined range, than there are six boundary values which are given below:

1. The extreme ends of the range
2. Just beyond the ends
3. Just before the ends

i.e. if an integer range is min to max, then six values are

- **Min-1**
- **Min**
- **Min+1**
- **Max-1**
- **Max**
- **Max+1**

Now, I will explain boundary value analysis with the help of an example:
Example 1: if input condition have a range from a to v (a=10 to b=60), create test cases

1. 1.Immediately below a (min-1)=9

2. 2.At a (min)=10
3. 3.Immediately above a (min+1)=11
4. 4.Immediately below b (max-1)=59
5. 5.At b (max)=60
6. 6.Immediately above b (max+1)=61

Example 2:

Suppose X1 and X2 are two variables where X1 lies between P & Q and X2 lies between R & S

$$P \leq X1 \leq Q$$

$$R \leq X2 \leq S$$

There are two ways to generate boundary value analysis technique

1)By the no. of variables (for n variables)

2)By the kinds of ranges

Static transition technique:

This is another black box testing which is useful when testing state machines and also navigation of graphical user interface. Delegates will generate state transition diagrams and will test various levels of coverage. Negative tests can also be generated using a different aspect called a state table (a grid showing the resulting transitions for each state combined with each possible event, showing both valid and invalid transition). If we have a manageable set of states for a system then we are able to build a state model which has four important impacts:

1. States → how the state of the system changes from one state to another.
2. Transition →how the state of the system changes from one state to another.
3. Events→input to the system
4. Actions→output for the events

Ex:-

An example of state transition testing for a machine M with corresponding state diagram and state transition table.

State Input	1	0
S1	S1	S2
S2	S2	S1

So we can define state transition testing as a black box test design in which test cases are designed to execute valid and invalid state transitions.

All-Pair Testing:

It is Black box test design technique in which test cases are designed to execute all possible discrete combinations of each pair of input parameters. In pair-wise testing we have to exercise all pairs of values during testing. As if there are ‘a’ parameters, each with ‘b’ values then between each two parameter we have a * b pair. The main objective of pairwise testing is to have a set of test cases that covers all the pairs. As there are ‘p’ parameters, then the set of test cases will cover **(p-1) + (p-2) + ------- = p (p-1)/2** pairs. Now let us take an example of all pair testing

E.g. suppose there are three parameters A (secondary memory), B (primary memory) and C (operating system). Each of them can have three values **(a1, a2, a3), (b1, b2, b3), (c1, c2, c3).** Then the total number of pairwise combination is **9*3=27,** these test cases are shown in table below:

Test cases of all pair testing

A	B	C	Pairs
a1	b1	c1	(a1, b1) (a1, c1) (b1, c1)
a1	b2	c2	(a1, b2) (a1, c2) (b2, c2)
a1	b3	c3	(a1, b3) (a1, c3) (b3, c3)
a2	b1	c2	(a2, b1) (a2, c2) (b1, c2)
a2	b2	c3	(a2, b2) (a2, c3) (b2, c3)
a2	b3	c1	(a2, b3) (a3, c1) (b3, c1)
a3	b1	c3	(a3, b1) (a3, c3) (b1, c3)
a3	b3	c1	(a3, b3) (a3, c1) (b3, c1)
a3	b3	c3	(a3, b3) (a3, c3) (b3, c3)

Conclusion:-

We conclude black box testing is a testing technique that ignores the internal mechanism or structure of a system and focuses on the outputs generated in response to selected inputs and execution conditions. Black box testing is conducted to evaluate the compliance of a system with specified functional requirements and corresponding predicted results. The various approaches to black box testing which I have described in my paper are

1.Equivalence partitioning: It divides the input data into partitions of data from which test cases can be derived.

2.Boundary value analysis: It focuses more on testing at boundaries or where the extreme boundary values are chosen.

3.Fuzzing: It is used for finding implementation bugs and also used to test for security problems in software.

4. State transition testing: in this test cases are designed to execute valid and invalid state transitions.

5.All pair testing: In this test cases are designed to execute all possible discrete combinations of each pair of input parameters