Что вы проверяете в White Box Testing?

Тестирование белого ящика включает тестирование программного кода для следующего:

- Внутренние дыры в безопасности
- Сломанные или плохо структурированные пути в процессах кодирования
- Поток конкретных входов через код
- Ожидаемый результат
- Функциональность условных циклов
- Тестирование каждого оператора, объекта и функции на индивидуальной основе

Тестирование может проводиться на системном, интеграционном и модульном уровнях разработки программного обеспечения. Одной из основных целей тестирования whitebox является проверка рабочего процесса для приложения. Он включает в себя тестирование ряда предопределенных входных данных по отношению к ожидаемым или желаемым выходным данным, так что когда конкретный ввод не приводит к ожидаемому выходному сигналу, вы столкнулись с ошибкой.

Нажмите здесь, если видео не доступно

Как вы проводите тестирование белого ящика?

Чтобы дать вам упрощенное объяснение тестирования белого ящика, мы разделили его на **два основных этапа** . Вот что делают тестеры при тестировании приложения с использованием техники тестирования белого ящика:

ШАГ 1) ПОНИМАТЬ ИСТОЧНИК КОД

Первое, что часто делает тестер, — это изучает и понимает исходный код приложения. Поскольку тестирование белого ящика включает в себя тестирование внутренней работы приложения, тестировщик должен быть очень хорошо осведомлен в языках программирования, используемых в тестируемых приложениях. Кроме того, тестирующий должен быть хорошо осведомлен о методах безопасного кодирования. Безопасность часто является одной из основных задач тестирования программного обеспечения. Тестер должен уметь обнаруживать проблемы с безопасностью и предотвращать атаки хакеров и наивных пользователей, которые могут вводить вредоносный код в приложение как сознательно, так и неосознанно.

Шаг 2) СОЗДАТЬ ИСПЫТАТЕЛЬНЫЕ ДЕЛА И ИСПОЛНИТЬ

Второй базовый шаг к тестированию белого ящика включает в себя тестирование исходного кода приложения на предмет правильной работы и структуры. Одним из способов является написание большего количества кода для проверки исходного кода приложения. Тестировщик разработает небольшие тесты для каждого процесса или серии процессов в приложении. Этот метод требует, чтобы тестировщик имел глубокие знания кода и часто выполнялся разработчиком. Другие методы включают ручное тестирование , тестирование и тестирование на ошибки, а также использование инструментов тестирования, как мы объясним далее в этой статье.

Пример тестирования WhiteBox

Рассмотрим следующий фрагмент кода

```
Printme (int a, int b) {                        ------------  Printme is a function
    int result = a+ b;
    If (result> 0)
        Print ("Positive", result)
    Else
        Print ("Negative", result)
    }                                          -----------   End of the source code
```

Целью тестирования WhiteBox является проверка всех ветвей решений, циклов, операторов в коде.

Чтобы выполнить утверждения в приведенном выше коде, тестовые случаи WhiteBox будут

- A = 1, B = 1
- A = -1, B = -3

Методы испытаний белой коробки

Основным методом тестирования Белого ящика является анализ покрытия кода. Анализ покрытия кода устраняет пробелы в наборе тестовых примеров. Он определяет области программы, которые не выполняются набором тестовых случаев. После выявления пробелов вы создаете контрольные примеры для проверки непроверенных частей кода, тем самым повышая качество программного продукта.

Доступны автоматизированные инструменты для анализа покрытия кода. Ниже приведены несколько методов анализа покрытия

Охват операторов: — Этот метод требует, чтобы каждое возможное утверждение в коде было проверено хотя бы один раз в процессе тестирования разработки программного обеспечения.

Покрытие ветвления — этот метод проверяет все возможные пути (если-еще и другие условные циклы) программного приложения.

Помимо вышесказанного, существует множество типов покрытия, таких как покрытие условий, покрытие нескольких условий, покрытие пути, покрытие функций и т. Д. Каждый метод имеет свои преимущества и пытается протестировать (охватить) все части программного кода. Используя покрытие Statement и Branch, вы обычно достигаете 80-90% покрытия кода, что является достаточным.

Чтобы узнать более подробно, обратитесь к этой статье https://www.guru99.com/code-coverage.html

Типы тестирования белого ящика

Тестирование белого ящика включает в себя несколько типов тестирования, используемых для оценки удобства использования приложения, блока кода или конкретного программного пакета. Там перечислены ниже —

-

Модульное тестирование: это часто первый тип тестирования, выполняемый в приложении. Модульное тестирование выполняется на каждом модуле или блоке кода по мере его разработки. Модульное тестирование по сути делается программистом. Как разработчик программного обеспечения, вы разрабатываете несколько строк кода, одну функцию или объект и тестируете их, чтобы убедиться, что они работают, прежде чем продолжить модульное тестирование, которое помогает выявить большинство ошибок на ранних этапах жизненного цикла разработки программного обеспечения. Ошибки, выявленные на этом этапе, дешевле и их легко исправить.

- **Тестирование на утечки памяти** : утечки памяти являются основными причинами медленной работы приложений. Специалист по обеспечению качества, имеющий опыт в обнаружении утечек памяти, необходим в тех случаях, когда у вас медленно работает программное приложение.

Помимо вышесказанного, несколько типов тестирования являются частью тестирования как черного ящика, так и белого ящика. Они перечислены ниже

- **Тестирование проникновения в White Box :** В этом тестировании тестировщик / разработчик имеет полную информацию об исходном коде приложения, подробную информацию о сети, задействованные IP-адреса и всю информацию о сервере, на котором работает приложение. Цель состоит в том, чтобы атаковать код с нескольких сторон, чтобы выявить угрозы безопасности.
- **Тестирование** мутации **White Box** : **Тестирование** мутации часто используется, чтобы обнаружить лучшие методы кодирования, используемые для расширения программного решения.

Инструменты тестирования White Box

Ниже приведен список лучших инструментов тестирования белого ящика.

- Parasoft Jtest
- EclEmma
- NUnit
- PyUnit
- HtmlUnit
- CppUnit

Преимущества тестирования белого ящика

- Оптимизация кода путем поиска скрытых ошибок.
- Тестовые случаи «белого ящика» могут быть легко автоматизированы.
- Тестирование является более тщательным, поскольку обычно покрываются все пути кода.
- Тестирование может начаться рано в SDLC, даже если GUI недоступен.

Недостатки тестирования WhiteBox

- Тестирование белого ящика может быть довольно сложным и дорогостоящим.
- Разработчики, которые обычно выполняют тесты «белого ящика», ненавидят это. Тестирование белого ящика разработчиками не детально может привести к производственным ошибкам.
- Тестирование белого ящика требует профессиональных ресурсов с подробным пониманием программирования и реализации.

- Тестирование белого ящика отнимает много времени, более крупные приложения для программирования требуют времени для полного тестирования.

Конечные заметки:

- Тестирование белого ящика может быть довольно сложным. Сложность связана с тестируемым приложением. Небольшое приложение, которое выполняет одну простую операцию, может быть протестировано в течение нескольких минут, в то время как более крупным программным приложениям требуются дни, недели и даже больше для полного тестирования.
- Тестирование белого ящика должно проводиться на программном приложении в том виде, в каком оно разрабатывается после его написания, и снова после каждой модификации.