Тема: Тестирование защиты программного обеспечения

Дисциплина МДК 04.02
Обеспечение качества функционирования компьютерных систем
Группа ИП-31

- ***Тестирование*** – процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом.

Уровень тестирования

Для каждого уровня тестирования может быть определено:

1. Цель
2. Объекты тестирования
3. Прослеживание связи с базисом тестирования (при наличии)
4. Критерии входа и выхода
5. Артефакты процесса тестирования, которые будет поставлять отдел тестирования - тестовые сценарии, протоколы тестирования, отчетность о результатах и другие
6. Тестовые методики
7. Измерения и метрики
8. Инструментарий

- ***Тестирование по требованиям безопасности*** - процесс выявления наличия или отсутствия уязвимостей в продукте в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом.

Тестирование защищенности

Тестирование защищенности: Тестирование с целью оценить защищенность программного продукта

Объекты тестирования:

- пароли
- шифрование
- аппаратные устройства доступа
- уровни доступа к информации
- авторизация
- скрытые каналы
- безопасность на физическом уровне

- Если при тестировании удалось обнаружить потенциально опасные участки кода, это еще не значит, что найдены уязвимости (поэтому такие сигнатуры и называются ПОТЕНЦИАЛЬНО опасными).
- Необходимо провести «экспериментальное тестирование», т.е. попробовать использовать опасные участки кода с целью компрометации приложения.
- На выходе необходимо добиться нарушения доступности, целостности, конфиденциальности, чтобы удостовериться в наличии уязвимости.
- В этом случае можно говорить, что обнаруженные сигнатуры влияют, либо не влияют на безопасность тестируемого программного обеспечения.

- При проведении тестирования по требованиям безопасности проводятся следующие испытания:
  - аудит безопасности кода (направленный на выявление уязвимостей);
  - функциональное тестирование (характеристик безопасности тестируемого программного обеспечения);
  - экспериментальное тестирование (проверяется возможность, или невозможность эксплуатации обнаруженных сигнатур).

- **ПРИЕМЫ ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ**
  - Ручной (экспертный анализ)
  - Статически анализ безопасности (по шаблону)
  - Динамический анализ безопасности

- При ручном подходе выявления уязвимостей применяется экспертный анализ, т.е. специалист, который проводит данное исследование, полагается на свои знания и опыт.
- Данный подход не подразумевает использования, каких либо автоматизированных средств.
- Данный прием имеет большие затраты по времени и предполагает наличие специалистов высокой квалификации.
- Данный подход считается самым эффективным с точки зрения точности и полноты покрытия проверок.

- Прием выявления уязвимостей «по шаблону» подразумевает использование материалов и наработок полученных исходя из опыта работы в данной области.
- При подходе выявления уязвимостей «по шаблону» часто применяется автоматизированный подход поиска уязвимостей по заданным шаблонам (спискам потенциально опасных сигнатур).
- Часто при данном подходе используется совмещение методов автоматизированного и ручного поиска уязвимостей.

- Используются средства автоматизированного поиска уязвимостей кода PREfix, FlawFinder, RATS, UCA, ITS4, АК-ВС и другие.
- Каждая испытательная лаборатория или даже отдельный департамент тестирования используют свои собственные базы сигнатур.
- Можно воспользоваться положительной практикой уже существующей в международных проектах CWE, либо известным ресурсом для Web-приложений OWASP .

- Динамический анализ является обязательным подходом при выявлении уязвимостей.
- Он позволяет проводить тестирование при непосредственном выполнении программного изделия.
- В процессе динамического тестирования программного комплекса реализуется составленный список тестов, направленный на достижение, либо провал нарушения функций безопасности продукта.
- Т.е. определяется возможность или невозможность эксплуатировать найденную потенциально опасную сигнатуру в рамках работающего программного продукта, при заданном тестовом окружении.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Уровни тестирования:

- Модульное тестирование (Unit testing)
  - Этот уровень тестирования позволяет проверить функционирование отдельно взятого элемента системы. Что считать элементом – модулем системы определяется контекстом.
- Наиболее полно данный вид тестов описан в стандарте IEEE 1008–87 “Standard for Software Unit Testing”, задающем интегрированную концепцию систематического и документированного подхода к модульному тестированию.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Уровни тестирования:

- Интеграционное тестирование (Integration testing)
  - Данный уровень тестирования является процессом проверки взаимодействия между программными компонентами/модулями.
  - Классические стратегии интеграционного тестирования – “сверху-вниз” и “снизу-вверх” – используются для традиционных, иерархически структурированных систем и их сложно применять, например, к тестированию слабосвязанных систем, построенных в сервисно-ориентированных архитектурах (SOA).

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Уровни тестирования:

- Системное тестирование (System testing)
  - Системное тестирование охватывает целиком всю систему.
  - Большинство функциональных сбоев должно быть идентифицировано еще на уровне модульных и интеграционных тестов.
  - В свою очередь, системное тестирование, обычно фокусируется на нефункциональных требованиях – безопасности, производительности, точности, надежности т.п.
  - На этом уровне также тестируются интерфейсы к внешним приложениям, аппаратному обеспечению, операционной среде и т.д.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Приёмочное тестирование (Acceptance/qualification testing)
  - Проверяет поведение системы на предмет удовлетворения требований заказчика.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Установочное тестирование (Installation testing)
  - Из названия следует, что данные тесты проводятся с целью проверки процедуры инсталляции системы в целевом окружении.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Альфа- и бета-тестирование (Alpha and beta testing)
  - Перед тем, как выпускается программное обеспечение, как минимум, оно должно проходить стадии альфа (внутреннее пробное использование) и бета (пробное использование с привлечением отобранных внешних пользователей) версий.
  - Отчеты об ошибках, поступающие от пользователей этих версий продукта, обрабатываются в соответствии с определенными процедурами, включающими подтверждающие тесты (любого уровня), проводимые специалистами группы разработки.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Функциональные тесты/тесты соответствия (Conformance testing/Functional testing/Correctness testing)
  - Эти тесты могут называться по разному, однако, их суть проста – проверка соответствия системы, предъявляемым к ней требованиям, описанным на уровне спецификации поведенческих характеристик.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Достижение и оценка надежности (Reliability achievement and evaluation)
  - Помогая идентифицировать причины сбоев, тестирование подразумевает и повышение надежности программных систем.
  - Случайно генерируемые сценарии тестирования могут применяться для статистической оценки надежности.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Регрессионное тестирование (Regression testing)
  - Определение успешности регрессионных тестов (IEEE 610–90 “Standard Glossary of Software Engineering Terminology”) гласит: “повторное выборочное тестирование системы или компонент для проверки сделанных модификаций не должно приводить к непредусмотренным эффектам”.
  - На практике это означает, что если система успешно проходила тесты до внесения модификаций, она должна их проходит и после внесения таковых.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Тестирование производительности (Performance testing)
  - Специализированные тесты проверки удовлетворения специфических требований, предъявляемых к параметрам производительности.
  - Существует особый подвид таких тестов, когда делается попытка достижения количественных пределов, обусловленных характеристиками самой системы и ее операционного окружения.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Нагрузочное тестирование (Stress testing)
  - Необходимо понимать отличия между рассмотренным выше тестированием производительности с целью достижения ее реальных (достижимых) возможностей производительности и выполнением программной системы с повышением нагрузки, вплоть до достижения запланированных характеристик и далее, с отслеживанием поведения на всем протяжении повышения загрузки системы.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Сравнительное тестирование (Back-to-back testing)
  - Единичный набор тестов, позволяющих сравнить две версии системы.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Восстановительные тесты (Recovery testing)
  - Цель – проверка возможностей рестарта системы в случае непредусмотренной катастрофы (disaster), влияющей на функционирование операционной среды, в которой выполняется система.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Конфигурационное тестирование (Configuration testing)
  - В случаях, если программное обеспечение создается для использования различными пользователями (в терминах “ролей”), данный вид тестирования направлен на проверку поведения и работоспособности системы в различных конфигурациях.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Виды тестирования:

- Тестирование удобства и простоты использования (Usability testing)
  - Цель – проверить, насколько легко конечный пользователь системы может ее освоить, включая не только функциональную составляющую – саму систему, но и ее документацию; насколько эффективно пользователь может выполнять задачи, автоматизация которых осуществляется с использованием данной системы; наконец, насколько хорошо система застрахована (с точки зрения потенциальных сбоев) от ошибок пользователя.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Техники, ориентированные на код (Code-based techniques):

- Тесты, базирующиеся на блок-схеме (Control-flow-based criteria)
  - Набор тестов строится исходя из покрытия всех условий и решений блок-схемы. В какой-то степени напоминает тесты на основе конечного автомата.
  - Отличие – в источнике набора тестов.
  - Максимальная отдача от тестов на основе блок-схемы получается когда тесты покрывают различные пути блок-схемы – по-сути, сценарии потоков работ (поведения) тестируемой системы.
  - Адекватность таких тестов оценивается как процент покрытия всех возможных путей блок-схемы.

ТРАДИЦИОННЫЕ МЕТОДЫ И ПОДХОДЫ ТЕСТИРОВАНИЯ

Техники, ориентированные на код (Code-based techniques):

- Тесты на основе потоков данных (Data-flow-based criteria)
  - В данных тестах отслеживается полный жизненный цикл величин (переменных) – с момента рождения (определения), на всем протяжении использования, вплоть до уничтожения (неопределенности).
  - В реальной практике используются нестрогое тестирование такого вида, ориентированное, например, только на проверку задания начальных значений всех переменных или всех вхождений переменных в код, с точки зрения их использования.

- Существует ряд средств автоматизации выявления уязвимостей, которые могут быть использованы при статическом анализе безопасности на уровне кода «по шаблону».

Список используемой литературы:

- 1. Microsoft Application Consulting and Engineering (ACE) Team. Performance Testing Microsoft. NET Web Application.
- 2. Роман Савин. Тестирование dot com.
- 3. Элфрид Дастин, Джефф Рэшка, Джон Пол. Автоматизированное тестирование программного обеспечения.
- 4. Alexey Kolosov. Using Static Analysis in Program Development.
- 5. Брайан Гетц. Избавьтесь от ошибок.
- 6. Криспин Кован. Безопасность систем с открытым кодом.
- 7. Алексей Марков, Сергей Миронов, Валентин Цирлов. Выявление уязвимостей в программном коде.
- 8. Алексей Марков, Валентин Цирлов. Аудит программного кода по требованиям безопасности.