Тема 11

«Стрессовое тестирование веб-ориентированных приложений»

Источники стрессовых ситуаций

Определение

Стрессовое тестирование (stress testing) – вид тестирования, направленный на проверку того, насколько приложение способно функционировать в условиях стресса и восстанавливаться после прекращения воздействия стресса.

Стрессом в данном контексте может быть повышение нагрузки (см. тему «Тестирование производительности и нагрузочное тестирование»), изменение конфигурации сервера и т.п.

Источники неприятностей

В большинстве источников стрессовое тестирование рассматривают как предельный случай нагрузочного, из чего логически вытекает, что первым источником стрессовых ситуаций для приложения является резкое превышение расчётной нагрузки.

Не будем дублировать материал темы про нагрузочное тестирование, а просто отметим примеры:

- ситуация «час пик»;
- (D)DoS-атака;
- проблемы с сетью.

Источники неприятностей

Второй важной причиной стрессов является изменение конфигурации среды исполнения приложения.

Чаще всего это связно с обновлением системного ПО или изменением настроек системного ПО.

Например:

- обновили веб-сервер;
- доработали настройки СУБД;
- приложение перенесли на другую ОС.

См. тему «Инсталляционное тестирование» (раздел про настройки, их проверку и реакцию на их изменение).

Источники неприятностей

Третьей причиной стресса является выход из строя, сбой или изменения (например, по факту обновления) в аппаратной конфигурации компьютера.

Да, веб-ориентированные приложения слабо привязаны к аппаратной части, но, во-первых, бывают исключения, в во-вторых, к объёму оперативной памяти или частоте процессора чувствительны любые приложения.

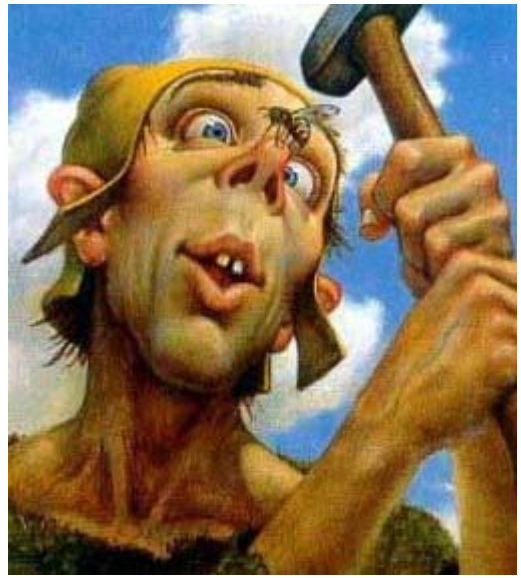
И уж точно любое приложение так или иначе «заметит» отключение питания, перезагрузку и т.п.

Источники неприятностей

Четвёртая причина – сбои или потенциально опасные ситуации, вызванные действиями пользователей.

Например:

- пользователь прерывает закачку файла;
- пользователь передаёт чрезмерно большой объём данных;
- пользователь передаёт данные в неверном формате.



Источники неприятностей

Пятая причина – попытки злоумышленников нарушить безопасность приложения.

Здесь сложно приводить примеры, не зависящие от конкретной ситуации, но в любом случае атака на приложение или среду его исполнения может и должна быть рассмотрена как стрессовая ситуация.

Источники неприятностей

Шестая причина – сбои и отказы в среде исполнения приложения.

Возможно, что-то «сломалось» в операционной системе, веб-сервере и т.п.

Теперь приложению нужно как-то на это отреагировать, а в идеале – продолжить работать «как ни в чём ни бывало».

Источники неприятностей

Седьмая причина – повреждения приложения, вызванные программными или аппаратными сбоями в файловой (дисковой) системе, воздействиями злоумышленника, вирусов и т.п.

Источники неприятностей

Восьмая причина – некомпетентные действия администратора, ставящие под угрозу безопасность, производительность и саму работоспособность приложения.

Источники неприятностей

Девятая причина – конфликты с другими приложениями, работающими в системе.

Это могут быть новые приложения, которых раньше не было.

Старые приложения, которые раньше не конфликтовали с нашим приложением, могут начать это делать в силу изменения настроек.

Также старые приложения могут выполнить те или иные (вредные для нашего приложения) действия, которых раньше не выполняли.

Источники неприятностей

Десятая причина – комбинация нескольких из ранее перечисленных причин.

Этот вариант вынесен в отдельный пункт, т.к. такие случаи достаточно тяжело диагностировать.

Способы эмуляции стрессовых ситуаций

Вместо введения

Стрессовые ситуации прекрасно возникают сами в процессе выполнения иных видов тестирования или эксплуатации приложения.

Однако, мы боремся за высокое качество ПО, а потому должны быть уверены, что предусмотрели всё или почти всё.

Способы эмуляции

Проблемы, связанные с нагрузкой на систему, хорошо эмулируются нагрузочным тестированием с заведомым превышением допустимой нагрузки (порой – в разы или десятки раз).

Способы эмуляции

Для проверки того, как приложение реагирует на изменение конфигурации среды исполнения, можно или действительно менять конфигурацию, или… переносить одно и то же приложение между разными системами (например, запускать под разными виртуальными машинами).

Способы эмуляции

Чтобы проэмулировать изменения или сбои в аппаратной части системы, можно также воспользоваться виртуальными машинами, или воспроизвести совершенно реальные сбои (выключить питание, выдернуть сетевой кабель, нажать reset и т.п.)

Способы эмуляции

Проверка реакции на потенциально опасные действия пользователей отчасти выполняется в рамках общего функционального тестирования сложными негативными тестами, но в рамках стрессового тестирования можно позволить себе «перейти все рамки разумного».

Здесь как никогда прекрасно работает правило: «конечный пользователь – либо подлец, либо идиот».

Способы эмуляции

Воспроизведением ситуаций, связанных с нарушением безопасности, в большей мере занимается «тестирование безопасности».

Но ничто не мешает нам использовать готовые автоматизированные средства тестирования безопасности, которые подвергнут наше приложение достаточно жёсткому, чтобы считаться стрессовым, воздействию.

Способы эмуляции

Проэмулировать сбои и отказы в среде исполнения – проще простого. Нужно останавливать сервисы, удалять библиотеки и компоненты и т.п.

Можно, опять же, использовать готовые «сбойные окружения» в виде образов виртуальных машин.

Способы эмуляции

Повреждения приложения также достаточно просто эмулировать – можно удалять и/или повреждать его файлы, повреждать записи в БД.

Единственное, о чём следует помнить, так это о том, что всегда есть некий предел, после которого выйдет из строя любое приложение.

Например, если вы полностью удалите приложение, оно само уже не восстановится.

Этот вариант тестирования чем-то похож на тестирование иммунитета и системы регенерации человека: от гриппа или перелома он может оправиться, а вот от пули в голову – уже нет.

Способы эмуляции

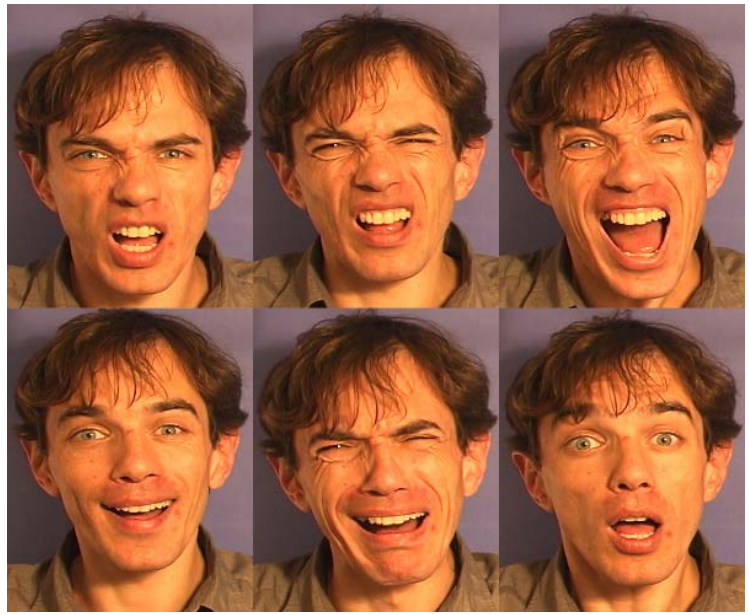
Некомпетентные действия администратора эмулировать одновременно и просто и сложно.

Просто потому, что можно действовать как «обезьяна с клавиатурой», выполняя хаотичные команды.

Сложно потому, что просто «безобразничать» – мало. Нужно продумать и воспроизвести максимально разрушительные действия.

Способы эмуляции

Для проверки реакции приложения на конфликты с другим ПО, следует, во-первых, обратиться к практикам тестирования совместимости, а во-вторых, сформировать перечень потенциально опасного ПО и **проверить реакцию** нашего приложения на самые неприятные варианты развития событий.



Способы эмуляции

И, наконец, да – можно создавать несколько стрессовых ситуаций одновременно.

Это тоже интересный тест.

Реакция приложения

Какой реакции приложения мы ожидаем в ответ на стрессовые ситуации?

В самом идеальном случае, приложение должно полностью сохранить все показатели качества и отправить администратору уведомление о том, что «что-то непонятное вокруг творилось».

В хорошем случае – приложение должно сохранить ключевые показатели качества (как правило, они относятся к безопасности и живучести), и подать сигнал тревоги.

Реакция приложения

В самом худшем случае приложение должно перейти в режим работы, в котором пользователям отображается сообщение вида «Извините, зайдите попозже», и при этом не производить никаких деструктивных действий в плане работы с данными или обеспечением безопасности.

Реакция приложения

Недопустимыми вариантами считаются:

- приложение «подвисает», «падает», «перестаёт запускаться»;
- приложение повреждает свои или пользовательские данные;
- приложение повреждает среду исполнения;
- приложение перестаёт контролировать параметры безопасности, открывая доступ злоумышленнику;
- и т.п.

Использование виртуальных машин

Введение

В предыдущих разделах данной темы мы неоднократно упоминали о том, что эмуляцию некоторых условий удобно реализовывать с применением виртуальных машин.

Сейчас мы немного поговорим о VMWare – одной из самых распространённых виртуальных машин.

Определение

Виртуальная машина (virtual machine) – (в нашем случае) комплекс программных средств, эмулирующий аппаратную среду для выполнения операционной системы и программ.

Если «по-простому» – это программа, позволяющая сделать «компьютер внутри компьютера».

Логика такая: на компьютер ставится операционная система, на ней запускается виртуальная машина, а на ней… получается «чистый компьютер», на который можно установить другую операционную систему, программы и т.п.

Зачем это нужно

Устанавливать на один компьютер десятки операционных систем – нерационально (а иногда и невозможно).

А тестировщикам часто приходится работать с разными конфигурациями среды исполнения приложений, разными операционными системами и т.п.

Вывод очевиден: устанавливается одна операционная система, на неё ставится виртуальная машина, под которой можно настроить десятки образов нужных нам операционных систем и загружать их по мере надобности.

Принцип использования

Принцип использования виртуальных машин следует из их определения.

Вы устанавливаете на свой компьютер специальное ПО, настраиваете под ним виртуальный компьютер, на который ставите и настраиваете нужным образом нужную вам операционную систему.

Результат сохраняется в виде образа, который можно в любой момент загрузить.

Ещё одно важное преимущество

Как несложно догадаться, образ виртуального компьютера и операционной системы можно сохранить отдельно.

После этого вы можете выполнять даже самые деструктивные действия с виртуальной операционной системой и приложениями под ней – у вас всегда есть полностью готовая к работе резервная копия, которую можно использовать для восстановления «разрушений» в течение единиц секунд.

И ещё одно важное преимущество

Иногда для работы того или иного приложения нужна специфическая операционная система. Под вашей основной ОС это приложение не работает.

Тогда, опять же, можно установить нужную ОС на виртуальную машину.

Как видите, преимуществ много.

Тест для проверки изученного

1. Что такое «стрессовое тестирование»?
2. Какие вы запомнили основные источники неприятностей для веб-ориентированных приложений в контексте стрессового тестирования?
3. Какую опасность может представлять для веб-ориентированного приложения изменение аппаратной конфигурации компьютера?
4. Что такое «виртуальная машина»? Зачем она нужна тестировщикам?
5. Какие проблемы, изучаемые стрессовым тестированием, могут быть вызваны человеческим фактором?
6. Как стрессовое тестирование связано с нагрузочным тестированием?
7. Нехватка каких аппаратных ресурсов в первую очередь приведёт к возникновению стрессовой ситуации?
8. Можно ли заменить стрессовое тестирование более глубоким исследованием в других направлениях тестирования?
9. На какой стадии развития проекта следует приступать к стрессовому тестированию приложения?
10. Можно ли считать качество приложения низким, если приложение не прошло некоторые стрессовые тесты?