ПРОБЛЕМЫ АНАЛИЗА И ТЕСТИРОВАНИЯ WEB-СИСТЕМ

К.Г. Колодий (г. Иркутск ИрГУПС)

THE PROBLEMS OF WEB-SYSTEM ANALYSIS AND TESTING

K.G. Kolodiy (Irkutsk ISURE)

This article contains possible problems, which appeared during web-system development. It also contains review of tools for automated testing. Moreover there are description of problems of automation of testing.

Сейчас web-сайты представляют собой полноценные приложения – электронный магазин, форум, почтовый сервис, специализированные системы, используемые в локальных сетях компаний с возможностью удаленного доступа с помощью Интернет, и многое другое – все эти интерактивные сервисы определенным образом взаимодействуют с пользователями, с другими серверами. Это роднит их с обычными приложениями, называемыми десктопными – в некоторых случаях между ними разница только в месте выполнения кода.

При создании web-систем приходиться решать задачи: обеспечения безопасности, стабильности работы при больших нагрузках, утечки памяти, проблемы синхронизации процессов (потоков).

Так как от безопасности сервера зависит практически все, необходимо решить задачу обеспечения безопасности создаваемой web-системы. Тестирование безопасности следует проводить регулярно. Кроме того, тестированию подвергается не только сам сайт или web-приложение, а весь сервер полностью – и web-сервер, и операционная система, и все сетевые сервисы. Программа тестирования безопасности имитирует действия реального пользователя-взломщика и пытается применить к серверу все известные ей методы атаки и проверяет все уязвимости. Результатом работы будет отчет о найденных уязвимостях и рекомендации по их устранению. Обычно такие сканеры безопасности продаются как самостоятельный продукт – к примеру, один из лучших сканеров, Xspider компании Positive Technologies [1]. Также можно воспользоваться сервисами для тестирования безопасности Web-сайтов, такие как XSpider Online компании Positive Technologies и QualysGuard компании Qualys [2]. Основные возможности сервисов:

- сканирование сети — применяется для обнаружения компьютеров за сетевым экраном. Генерируется графическая карта всех компьютеров (возможно, доступных) во внешнем окружении или в диапазоне IP-адресов;
- сканирование инфраструктуры на наличие уязвимых мест;

– моделирование DoS-атак («отказ от обслуживания»). Примеры DoS-сценариев – DNS-атаки, SYN-переполнения, FIN-переполнения и др.;
– сканирование систем обнаружения вторжений на их надежность и пределы работоспособности при плотном трафике.

Еще одной проблемной задачей является обеспечение устойчивости к большим нагрузкам. Для это используют так называемое тестирование нагрузки – load-testing, stress-test или performance test. Такой тест имитирует одновременную работу нескольких сотен или тысяч посетителей, проверяя, будет ли устойчивой работа сайта под большой нагрузкой. Схема проведения нагрузочного тестирования представлена на рисунке 1.

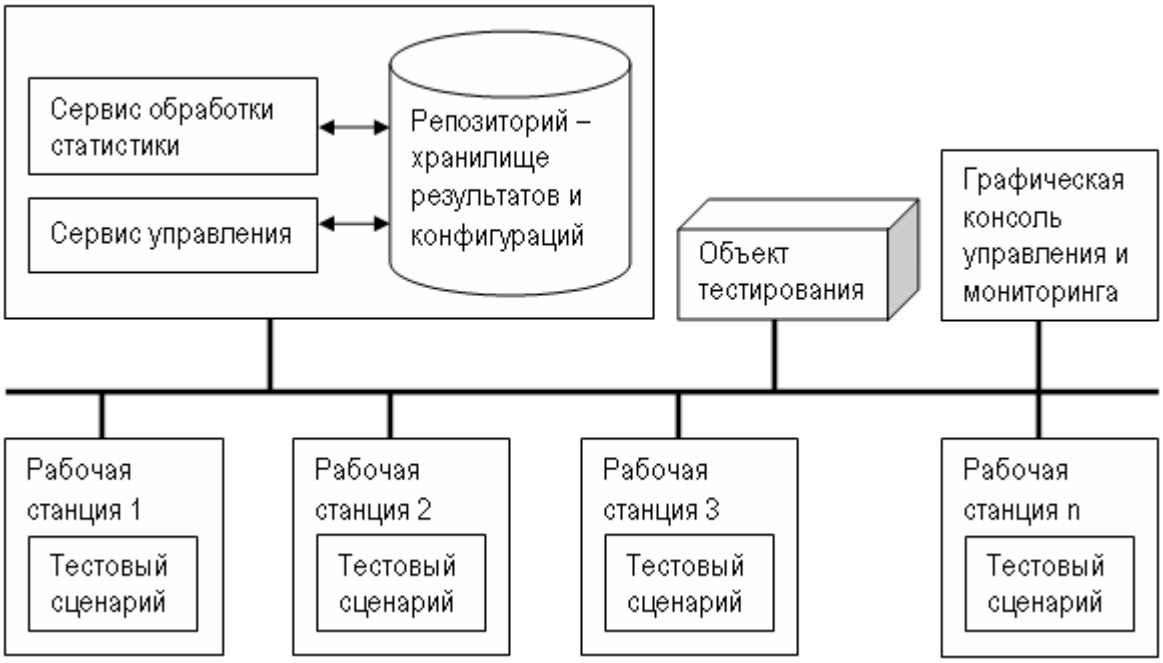

Рисунок 1 – Схема проведения нагрузочного тестирования

Кроме этого, можно имитировать кратковременные пики нагрузки, когда количество посетителей скачкообразно увеличивается – это очень актуально для новостных ресурсов и других сайтов с неравномерной аудиторией. В таком тесте проверяется совместная слаженная работа всего комплекса – аппаратной части сервера, web-сервера, программного ядра и других компонентов сайта. К подобным решениям можно отнести утилиту WAPT [3], а так-

же ряд других – Microsoft Web Application Stress Tool (WAS), Segue SilkPerformer, Webserver Stress Tool и другие [4].

Ошибки, связанные с утечкой памяти, являются одними из самых трудоемких в обнаружении. Современные языки программирования (например Java, семейство языков .NET) предоставляют средства, позволяющие автоматически освобождать неиспользуемую память (garbage collector, «сборщик мусора»). Тем не менее, сборщик мусора не может полностью справиться с утечками памяти по причинам крайней трудности контроля. Также за использование подобных средств приходится расплачиваться быстродействием системы.

Установить наличие утечек памяти в приложении можно по следующим признакам:

- существенное замедление отклика из-за того, что происходит обмен с виртуальной памятью, расположенной на диске;
- медленное (или не очень медленное) увеличение значения объема используемой памяти.

Для обнаружения утечек памяти можно использовать специальные средства такие как IBM Rational Purify, QNX Momentics [5].

Такие средства позволяют обнаруживать такие ошибки программирования как:

- использование неинициализированной памяти;
- утечки памяти и потенциальные утечки;
- выходы за пределы массивов при записи и чтении;
- ошибки при работе с указателями.

IBM Rational Purify помогает командам разработчиков находить дефекты с самого начала процесса разработки, достигая необходимого уровня качества к моменту передачи кода тестировщикам [6].

QNX Momentics разработанный компанией QNX Software Systems позволяет локализовать утечки памяти, переполнение буферов, недопустимые указатели, двойные операции освобождения памяти и другие распространенные ошибки. Во многих случаях разработчику нужно просто щелкнуть мышью по отображенной ошибке, чтобы найти проблемный исходный код. Кроме того, в инструменте применяется подход по оптимизации использования памяти, заключающийся в использовании базы данных, в которой регистрируются выполняемые приложениями операции выделения и освобождения памяти. Этот подход позволяет сохранять и воспроизводить процесс выделения памяти за длительный период времени, давая возможность разработчику обнаруживать скрытые проблемы непроизводительного использования памяти, которые приводят к ее исчерпыванию и не могут быть выявлены при помощи обычных инструментов анализа.

Web-системы – это клиент-серверные или распределенные системы, в которых вычис-

ления производятся на различных территориально-разнесенных компьютерах. В связи с этим возникают проблемы взаимодействия процессов. В многопроцессорных (многопоточных) системах могут возникать следующие виды процессов [7]:

- если родительский процесс по какой-то причине завершится раньше дочернего, последний становится «сиротой». Процесс-сирота усыновляется системным процессом, но при этом его всё равно считают "осиротевшим", поскольку первоначально создавший его процесс более не существует (рисунок 2);

Рисунок 2 – Образование процесса-сироты

- если же потомок уже завершил работу, а предок не готов принять от системы сигнал об этом событии, то потомок не исчезает полностью, а превращается в «зомби». Зомби не занимает процессорного времени, но строка в таблице процессов остается, и соответствующие структуры ядра не освобождаются;
- наконец, процесс может надолго впасть в «сон», который не удается прервать. Процесс, находящийся в таком состоянии, не реагирует на системные запросы и может быть уничтожен только перезагрузкой системы.

В Unix системах для обнаружения вышеописанных процессов можно использовать программу ps. Она выдает "мгновенный снимок" таблицы процессов, которые запущены в системе.

В Windows системах для обнаружения ошибок взаимодействия потоков можно воспользоваться средством Intel® Thread Checker [8]:

- определяет трудновоспроизводимые ошибки, используя продвинутый механизм их обнаружения (для обнаружения ошибок им необязательно срабатывать);
- автоматизирует обнаружение всех типов ошибок многопоточных приложений (гонка данных, взаимоблокировка потоков, блокировки потоков);
- поддерживает все широко распространённые многопоточные модели (Win32* и OpenMP*).

Также существует версия Thread Checker для операционной системы Linux.

Кроме решения вышеописанных задач, web-система, как любой программный продукт, должна выполнять действия соответствующие ожидаемому поведению. Целью про-

верки правильности функционирования является проверка реализации в программной системе всех функциональных и поведенческих требований, а также требований эффективности. На сегодняшний день существует большое количество средств автоматизированного тестирования функциональности Automated QA TestCommplete, Mercury WinRunner, IBM Rational Robot, Segue SilkTest, AdventNet QEngine и др.

Все эти средства основаны на записи действий пользователя и возможностью автоматического запуска различных сценариев действий пользователя. При выборе автоматизированного средства тестирования необходимо учитывать стоимость, поддерживаемые технологии, возможность интеграции с системами разработки и поддержку, предоставляемая разработчиком средства автоматизированного тестирования.

Ручное тестирование подвержено ошибкам и не обеспечивает тот же уровень проверки качества, который возможен при использовании автоматизированных средств тестирования. Однако не всегда ожидания в области автоматизированного тестирования оказываются реалистическими. Первичное внедрение автоматизированного тестирования требует тщательного анализа целевого приложения для определения того, какие разделы приложения могут тестироваться автоматически. По началу использования средств автоматизированного тестирования может даже повысить трудоемкость из-за того, что нужно будет выполнить задачи по установке. Тем не менее инвестиции в области применения средств тестирования начнут окупаться после первой итерации тестирования вследствие роста производительности команды тестировщиков.

Сравнение производительности, проведенное Институтом по проблемам обеспечения качества, показало специфическую разницу в объемах работ, измеренных в человеко-часах, при выполнении тестирования с применением ручных и автоматизированных методов. Анализ продемонстрировал, что суммарный объем работ при использовании автоматизированного тестирования составляет только 25% от количества человеко-часов, требуемых для проведения тестирования ручными методами. Эталонное исследование показало, что ручное тестирование требует почти в семь раз больших трудозатрат, чем автоматизированное тестирование [9].

В связи с повышением требований к качеству и надежности web-систем возникает необходимость решения следующих задач:

- обеспечение безопасности,
- стабильность работы при больших нагрузках,
- утечка памяти,
- проблема синхронизации процессов (потоков).

Для более эффективного тестирования необходимо проанализировать возможность

автоматизированного тестирования и тщательно разработать план тестирования. Данного плана необходимо придерживаться в течении всего процесса тестирования. При разработке плана тестирования стоит учесть имеющийся опыт по тестированию в предыдущих проектах. Более детально некоторые из существующих проблем тестирования web-систем будут рассмотрены в дальнейших работах.

ЛИТЕРАТУРА

1. Официальный сайт компании Positive Technologies –http://www.ptsecurity.ru
2. Официальный сайт компании Qualys – http://www.qualys.com
3. Брод М. Тестирование сайта [Электронный ресурс] /М. Брод 2004. — Режим доступа: http://hostinfo.ru/articles/439
4. Лозовюк А. Обзор решений для тестирования сайтов [Электронный ресурс] /А. Лозовюк 2004. — Режим доступа: http://hostinfo.ru/articles/442
5. Саватеев И. QNX выпустила средство обнаружения утечек памяти /И. Саватеев // PCWeek online [Электронный ресурс]. – Электрон. журн. – 2007. – Режим доступа: http://www.pcweek.ru/?ID=624484
6. Новичков А. Инструментальные средства поддержки процесса тестирования / А. Новичков // Тестирование и качество. – 2005. – №5. – С.15-22.
7. Хименко В. Процессы, задачи, потоки и нити /В. Хименко // Мир ПК [Электронный ресурс]. – 2000. – Режим доступа: http://www.citforum.ru/operating_systems/articles/process.shtml
8. Intel Threading Analysis Tools // Официальный сайт компании Intel – http://www3.intel.com/cd/software/products/asmo-na/eng/threading/219785.htm
9. Элфрид Дастин Автоматизированное тестирование программного обеспечения / Элфрид Дастин, Джефф Рэшка, Джон Пол – М.:ЛОРИ, 2003. – 567 с.