Технологии контроля безопасности автоматизированных систем на основе структурного и поведенческого тестирования программного обеспечения

**Человек:** Предметом исследования являются основные методы и принципы тестирования программных комплексов, применяемые в интересах оценивания и контроля безопасности автоматизированных систем. В результате проведенных исследований приведена информация по способам выявления методами тестирования программного обеспечения наиболее распространенных угроз безопасности для подсистем: межсетевого экранирования; аудита; контроля доступа; контроля целостности; парольной и криптографической. При этом считали, что в изделии могут присутствовать следующие уязвимости: переполнение буфера, некорректная обработка форматных средств, гонки. Методы исследования включают методы теории программирования, теории надежности, программной инженерии, помехоустойчивого кодирования, информационной безопасности, системного анализа. Основным выводом проведенного исследования является то, что тестирование программного обеспечения является мощным средством для выявления как ошибок в работе программ, так и уязвимостей безопасности. Современные методы поведенческого тестирования позволяют выявлять уязвимости без исходных текстов программ и могут успешно применяться на российском рынке, для которого передача исходных текстов на тестирование задача почти невыполнимая.

**Key words:** структурное тестирование, программная инженерия, метод тестирования программы, подсистема безопасности, уязвимость программного обеспечения, поведенческое тестирование, тестирование программ, безопасность ифнормации, безопасность автоматизированной системы, угрозы безопасности программ

=================================

**FastText\_KMeans\_Clean:** Существуют инструментальные средства проведения структурного тестирования программного обеспечения, но, они, как правило, заточены на выполнение узких задач, например, выявление утечек памяти, определение участков кода, некорректно обрабатывающих массивы и переменные, а также средства анализа связности фрагментов кода и др. Подсистема контроля доступа. Способы структурного тестирования:. printf("Cannot open input file %s.n", argv[1]);.

**Key words part:** 0.5454545454545454

=================================

**FastText\_KMeans\_Raw/:** Существуют инструментальные средства проведения структурного тестирования программного обеспечения, но, они, как правило, заточены на выполнение узких задач, например, выявление утечек памяти, определение участков кода, некорректно обрабатывающих массивы и переменные, а также средства анализа связности фрагментов кода и др. Подсистема контроля целостности. Способы структурного тестирования:. CString sSQL = “SELECT money FROM zarpl WHERE id=” + ID;.

**Key words part:** 0.5454545454545454

=================================

**FastText\_PageRank\_Clean/:** int main (int argc, char\* argv[]). if ((in = fopen(argv[1], "rt")) == NULL). Cannot open file bad\_file234a078eab0. int main (int argc, char\* argv[]). const long MAX\_INPUT = 0x7fff;. len = strlen(argv[1]);. if(len < MAX\_INPUT). //основной функционал.

**Key words part:** 0.2727272727272727

=================================

**FastText\_PageRank\_Raw/:** int main (int argc, char\* argv[]). if ((in = fopen(argv[1], "rt")) == NULL). Cannot open file bad\_file234a078eab0. 3) Переполнение целых чисел. int main (int argc, char\* argv[]). len = strlen(argv[1]);. if(len < MAX\_INPUT). 4) Внедрение SQL-команд.

**Key words part:** 0.2727272727272727

=================================

**Mixed\_ML\_TR/:** В статье рассмотрены основные методы и принципы тестирования программных комплексов. К числу подсистем безопасности, требующих тестирования, как правило, относят следующие подсистемы: межсетевого экранирования; аудита; контроля доступа; контроля целостности; парольную; криптографическую. Считаем, что в изделии могут присутствовать следующие уязвимости: переполнение буфера, некорректная обработка форматных средств, гонки (эти уязвимости являются наиболее распространенными [17-26]). Способы структурного тестирования:. Сравнение эффективности структурного и поведенческого тестирования:. int main (int argc, char\* argv[]). Способы структурного тестирования:. Данная ошибка достаточно трудно выявляется. Как видно из приведенного анализа: тестирование программного обеспечения является мощным средством для выявления ошибок в работе программ, так и уязвимостей безопасности.

**Key words part:** 0.8484848484848485

=================================

**MultiLingual\_KMeans/:** В статье рассмотрены основные методы и принципы тестирования программных комплексов. К числу подсистем безопасности, требующих тестирования, как правило, относят следующие подсистемы: межсетевого экранирования; аудита; контроля доступа; контроля целостности; парольную; криптографическую. Считаем, что в изделии могут присутствовать следующие уязвимости: переполнение буфера, некорректная обработка форматных средств, гонки (эти уязвимости являются наиболее распространенными [17-26]). Способы структурного тестирования:. Сравнение эффективности структурного и поведенческого тестирования:. int main (int argc, char\* argv[]). Данная ошибка достаточно трудно выявляется.

**Key words part:** 0.7272727272727273

=================================

**Multilingual\_PageRank/:** Также применяется гибридная стратегия поведения, которая является комбинацией вышесказанных стратегий. int main (int argc, char\* argv[]). printf("Cannot open input file %s.n", argv[1]);. 3) Переполнение целых чисел. int main (int argc, char\* argv[]). const long MAX\_INPUT = 0x7fff;. len = strlen(argv[1]);. CString sSQL = “SELECT money FROM zarpl WHERE id=” + ID;.

**Key words part:** 0.2727272727272727

=================================

**RuBERT\_KMeans\_Without\_ST/:** Существуют инструментальные средства проведения структурного тестирования программного обеспечения, но, они, как правило, заточены на выполнение узких задач, например, выявление утечек памяти, определение участков кода, некорректно обрабатывающих массивы и переменные, а также средства анализа связности фрагментов кода и др. Способы структурного тестирования:. printf("Cannot open input file %s.n", argv[1]);. Комментарий: В случае, если в качестве аргумента будет введена строка, длина которой больше 32К, то переменная len станет отрицательной, поэтому будет выполнен "основной функционал" с неверными входными данными. Современные методы поведенческого тестирования позволяют выявлять уязвимости без исходных текстов программ и могут успешно применяться на российском рынке, для которого передача исходных текстов на тестирование задача почти невыполнимая.

**Key words part:** 0.696969696969697

=================================

**RuBERT\_KMeans\_With\_ST/:** Несмотря на большие потенциальные возможности по выявлению угроз безопасности тестирование не получило широкого распространения в России из-за нескольких предпосылок:. Например, для изделия, обрабатывающего поток поступающих транзакций, создать ситуацию, при которой в единицу времени на него будут поступать тысячи транзакций, либо понижать пропускную способность сети и многое другое [17, 18]. Способы поведенческого тестирования:. int main (int argc, char\* argv[]). Данная ошибка очень легко выявляется на этапе анализа кода, достаточно отследить все операции с операндами командой строки на предмет выявления процедур проверок.

**Key words part:** 0.5757575757575758

=================================

**RUBERT\_page\_rank\_Without\_ST/:** Все методы тестирования можно разделить на две большие группы: это функциональные тесты и тесты на безопасность, надежность и производительность. Этот вид тестирования обычно проводится самим разработчиком [13-16]. if(len < MAX\_INPUT). Эти программы могут блокировать друг друга, а также изменять один и тот же объект. Такая ошибка достаточно трудно выявляется.

**Key words part:** 0.6363636363636364

=================================

**RUBERT\_page\_rank\_With\_ST/:** if ((in = fopen(argv[1], "rt")) == NULL). printf("Cannot open input file %s.n", argv[1]);. const long MAX\_INPUT = 0x7fff;. len = strlen(argv[1]);. CString sSQL = “SELECT money FROM zarpl WHERE id=” + ID;.

**Key words part:** 0.2727272727272727

=================================

**RUSBERT\_KMeans\_Without\_ST/:** Тестирование программ можно рассматривать как в качестве самостоятельного подхода к анализу безопасности, так и использование его методов как в рамках сертификации, аттестации или аудита систем [7, 8]. Существует ряд причин, когда необходимо использовать тестирование программного обеспечения:. Способы структурного тестирования:. int main (int argc, char\* argv[]). Как видно из приведенного анализа: тестирование программного обеспечения является мощным средством для выявления ошибок в работе программ, так и уязвимостей безопасности.

**Key words part:** 0.8181818181818182

=================================

**RUSBERT\_KMeans\_With\_ST/:** Данное тестирование заключается в проверке соответствия функций изделия какой-либо документации, представленной разработчиком. Подсистема безопасности. Способы поведенческого тестирования:. Результатами могут быть: запись по произвольному адресу в памяти, обход защиты стека, исполнение произвольного кода и раскрытие информации. printf("Cannot open input file %s.n", argv[1]);. Можно использовать только рандомизированное тестирование. В результате реализации этой уязвимости, может быть осуществлено блокирование программы, аварийный останов программы. Такая ошибка достаточно трудно выявляется.

**Key words part:** 0.696969696969697

=================================

**RUSBERT\_page\_rank\_Without\_ST/:** 1) Несовершенство нормативной базы:. if ((in = fopen(argv[1], "rt")) == NULL). len = strlen(argv[1]);. if(len < MAX\_INPUT). //основной функционал.

**Key words part:** 0.2727272727272727

=================================

**RUSBERT\_page\_rank\_With\_ST/:** 1) Несовершенство нормативной базы:. Подсистема безопасности. len = strlen(argv[1]);. if(len < MAX\_INPUT). //основной функционал.

**Key words part:** 0.42424242424242425

=================================

**Simple\_PageRank/:** Существуют инструментальные средства проведения структурного тестирования программного обеспечения, но, они, как правило, заточены на выполнение узких задач, например, выявление утечек памяти, определение участков кода, некорректно обрабатывающих массивы и переменные, а также средства анализа связности фрагментов кода и др. Считаем, что в изделии могут присутствовать следующие уязвимости: переполнение буфера, некорректная обработка форматных средств, гонки (эти уязвимости являются наиболее распространенными [17-26]). Тестирование на проникновение, проверка системы фильтрации сообщений, сканирование портов и уязвимостей, попытка обхода системы, подмена сетевых пакетов, DoS-атака. Прежде всего, необходимо с использованием метода сигнатурного тестирования выявить все участки программы, где используются функции по работе со строками. Комментарий: пользователь может ввести любое значение идентификатора, либо вместо идентификатора ввести дополнительные операнды SQL-запроса, которые при конкатенации введенной строки со строкой из программы выведут недоступные ранее для пользователя данные. Как видно из приведенного анализа: тестирование программного обеспечения является мощным средством для выявления ошибок в работе программ, так и уязвимостей безопасности.

**Key words part:** 0.8181818181818182

=================================

**TextRank/:** Кроме этого будет приведена информация по способам выявления методами тестирования наиболее распространенных угроз безопасности. Существуют инструментальные средства проведения структурного тестирования программного обеспечения, но, они, как правило, заточены на выполнение узких задач, например, выявление утечек памяти, определение участков кода, некорректно обрабатывающих массивы и переменные, а также средства анализа связности фрагментов кода и др. При проведении тестировании на безопасность изделия необходимо выделить все подсистемы безопасности изделия и проверить реализацию существующих уязвимостей. Наиболее эффективным методом является рандомизированное тестирование, которое заключается в передачи анализируемому объекту полуслучайных данных произвольного объема и отслеживания работы программы при обработки этих данных. Поведенческое тестирование для выявления данной ошибки просто реализуемо и в более короткие сроки уязвимости могут быть выявлены. Как видно из приведенного анализа: тестирование программного обеспечения является мощным средством для выявления ошибок в работе программ, так и уязвимостей безопасности.

**Key words part:** 0.8787878787878788

=================================

**TF-IDF\_KMeans/:** 2) выявление ошибок в разработанном программном обеспечении;. Подсистема безопасности. 1) Переполнение буфера. Способы поведенческого тестирования:. Сравнение эффективности структурного и поведенческого тестирования:. int main (int argc, char\* argv[]). Можно использовать только рандомизированное тестирование. Для выявления ошибки используются метод сигнатурного анализа поиска операций взаимодействия с базой данных.

**Key words part:** 0.7272727272727273

=================================

**Текст:** Наиболее распространенными способами оценки и контроля безопасности автоматизированных систем являются: сертификация; аттестация; аудит; тестирование [4-6]. Несмотря на большие потенциальные возможности по выявлению угроз безопасности тестирование не получило широкого распространения в России из-за нескольких предпосылок:. 1) Несовершенство нормативной базы:. 2) Высокая стоимость и трудоемкость проведения тестирования программного обеспечения;. 3) Малое количество отечественных компаний, выполняющих независимое тестирование.. В статье рассмотрены основные методы и принципы тестирования программных комплексов. Кроме этого будет приведена информация по способам выявления методами тестирования наиболее распространенных угроз безопасности.. Тестирование программ можно рассматривать как в качестве самостоятельного подхода к анализу безопасности, так и использование его методов как в рамках сертификации, аттестации или аудита систем [7, 8]. Существует ряд причин, когда необходимо использовать тестирование программного обеспечения:. 1) предоставление разработчикам информации, необходимой для минимизации ошибок;. 2) выявление ошибок в разработанном программном обеспечении;. 3) оценка безопасности программного обеспечения;. 4) предоставление необходимой информации для разработчика, по созданию проекта, который будет легко в дальнейшем тестировать;. 5) попытка взлома программного обеспечения или проверка на уязвимости;. 6) проверка соответствия программного продукта, заявленной документации.. Существуют две основные стратегии тестирования программного обеспечения: стратегия поведенческого тестирования (которая также называется тестированием черного ящика или функциональным тестированием) и структурное тестирование (тестирование белого ящика). Также применяется гибридная стратегия поведения, которая является комбинацией вышесказанных стратегий.. Для проведения структурного тестирования необходимо наличие исходных текстов программ. Структурное тестирование предполагает составление программы тестирования на основании знаний о структуре и конфигурации объекта испытаний. Этот метод описан в стандарте ANSI/IEEE Std 1008-1987. Такой вид тестирования наиболее эффективен для выявления программных ошибок, однако, в тоже время, является самым трудоемким. Он используется в случае анализа модулей небольшого объема или отдельно взятых фрагментов кода, например, связанных с безопасностью изделия [9]. Метод структурного тестирования применяется в ходе проведения сертификационных испытаний программного обеспечения [10, 11].. Существуют инструментальные средства проведения структурного тестирования программного обеспечения, но, они, как правило, заточены на выполнение узких задач, например, выявление утечек памяти, определение участков кода, некорректно обрабатывающих массивы и переменные, а также средства анализа связности фрагментов кода и др. [10, 12]. В настоящее время распространена ситуация, когда разработчики не хотят предоставлять исходные тексты своих изделий, но в тоже время, хотят его протестировать. В данной ситуации возможно проведение только поведенческого тестирования. Потому остановимся на этом методе поподробнее.. Все методы тестирования можно разделить на две большие группы: это функциональные тесты и тесты на безопасность, надежность и производительность.. Функциональное тестирование имеет важное значение для разработчиков, т.к. это оно непосредственно влияет на качество программной продукции. Данное тестирование заключается в проверке соответствия функций изделия какой-либо документации, представленной разработчиком. В рамках функциональных проверок тестировщик выполняет все заявленные разработчиком возможности изделия. Как правило, этот вид тестирования выполняется с другими видами тестирования, например, нагрузочным или стрессовым тестированием. Названный вид тестирования описан стандартом ISO 09646-1-6:1991, регламентирующим проверку функциональных возможностей и поведения продукции относительно требований и рекомендаций ISO, а также заявлений и документации разработчиков о функциональных возможностей изделия. Этот вид тестирования обычно проводится самим разработчиком [13-16].. Тестирование на надежность и производительность заключается в создании стрессовых ситуаций для изделия и попыток заставить работать изделие в нештатном режиме. Например, для изделия, обрабатывающего поток поступающих транзакций, создать ситуацию, при которой в единицу времени на него будут поступать тысячи транзакций, либо понижать пропускную способность сети и многое другое [17, 18].. При проведении тестировании на безопасность изделия необходимо выделить все подсистемы безопасности изделия и проверить реализацию существующих уязвимостей. К числу подсистем безопасности, требующих тестирования, как правило, относят следующие подсистемы: межсетевого экранирования; аудита; контроля доступа; контроля целостности; парольную; криптографическую. Считаем, что в изделии могут присутствовать следующие уязвимости: переполнение буфера, некорректная обработка форматных средств, гонки (эти уязвимости являются наиболее распространенными [17-26]).. Соответствие указанных подсистем безопасности и методов тестирования представлено в таблице 1.. Таблица 1 - Соответствие подсистем безопасности и методов тестирования. Подсистема безопасности. Методы и направления тестирования. Подсистема межсетевого экранирования. Тестирование на проникновение, проверка системы фильтрации сообщений, сканирование портов и уязвимостей, попытка обхода системы, подмена сетевых пакетов, DoS-атака. Подсистема аудита. Генерация тестовых действий и отслеживание отображения их результатов в журнале, проверка защищенности журнала аудита, осуществление попыток его модификации, подмена пользователей. Подсистема контроля доступа. Осуществление попыток несанкционированного доступа к объектам защиты, отслеживание их фиксации и попытка изменения политики разграничения доступа,. Подсистема контроля целостности. Осуществление модификации контролируемых файлов и отслеживание реакции изделия на эти действия, подмена хешей контролируемых файлов. Парольная подсистема. Подбор паролей и имен пользователей, попытка взлома хранилища паролей и осуществление операции восстановления паролей, подмена пользователей. Криптографическая подсистема. Перехват засекреченной информации и попытка ее вскрытия, подбор ключей к криптосистеме и проверка уязвимостей их реализации. Еще одним распространенным методом тестирования является «аутсорсинг». Этот вид подразумевает тестирование продукта независимыми компаниями: за рубежом созданы специальные тестовые агентства, наиболее часто применяющие тестирование безопасности, надежности и производительности.. 1) Переполнение буфера. Переполнение буфера возникает из-за того, что данные и информация о потоке выполнения программы перемешаны, а также в низкоуровневых языках программирования разрешен прямой доступ к памяти. Пример возникновения переполнения буфера – запись программой данных в память, не принадлежащую выделенному буферу.. Результатами переполнения буфера может быть как крах программы, так получение противником контроля над положением или выполнение других программ с правами пользователя, запустившего программу.. Классический пример реализации переполнения буфера на языке С++:. #include <stdio.h>. void func1(). {. char buf[256];. gets(buf);. ….. }. Способы структурного тестирования:. Обнаружить места переполнения буфера можно как легко, так и очень сложно. Прежде всего, необходимо с использованием метода сигнатурного тестирования выявить все участки программы, где используются функции по работе со строками. Затем проанализировать все возможные параметры, а также размеры используемых строк и их проверку. Кроме этого, необходимо отследить все вводимые пользователями данные как напрямую, так и через конфигурационные файлы и базы данных, начиная с точки ввода в программу и далее по всем функциям.. Кроме этого можно использовать средства обнаружения ошибок при работе с памятью. Способы поведенческого тестирования:. Наиболее эффективным методом является рандомизированное тестирование, которое заключается в передачи анализируемому объекту полуслучайных данных произвольного объема и отслеживания работы программы при обработки этих данных. В ходе исследования во все места запроса от пользователей данные вводятся полуслучайные данные произвольного размера. Это может быть, ввод имени пользователя, либо имени файлов. Файлы настроек приложения, доступные для пользователей, либо записи из баз данных.. Кроме этого можно использовать средства обнаружения ошибок при работе с памятью и применять существующие эксплойты.. Сравнение эффективности структурного и поведенческого тестирования:. Поведенческое тестирование для выявления данной ошибки просто реализуемо и в более короткие сроки уязвимости могут быть выявлены. Однако нет 100% вероятности выявления как в случае структурного тестирования.. 2) Ошибки, связанные с форматной строкой. Такая ошибка является разновидностью переполнения буфера и заключается в отсутствии контроля вводимых от пользователя параметров.. Результатами могут быть: запись по произвольному адресу в памяти, обход защиты стека, исполнение произвольного кода и раскрытие информации.. Пример реализации ошибки:. Файл test.cpp. #include <stdio.h>. int main (int argc, char\* argv[]). {. FILE \*in;. if(argc > 1). {. if ((in = fopen(argv[1], "rt")) == NULL). {. printf("Cannot open input file %s.n", argv[1]);. return 1;. }. }. }. Вызов файла:. C:test.exe bad\_file%x%x.txt. В результате на экране выедено:. Cannot open file bad\_file234a078eab0.. Где, 234a07, 8eab0 – адреса из стека.. Способы структурного тестирования:. Данная ошибка очень легко выявляется на этапе анализа кода, достаточно отследить все операции с операндами командой строки на предмет выявления процедур проверок.. Способы поведенческого тестирования:. Наиболее эффективным методом также является рандомизированное тестирование. В ходе исследования при вызове программы в качестве параметров вызова передаются специальным образом сконфигурированные данные. Эти данные включают спецификаторы формата данных, длинные пути, применение существующих эксплойтов.. Сравнение эффективности структурного и поведенческого тестирования:. Ошибки, связанные с форматной строкой легко выявляются при использовании обоих методов.. 3) Переполнение целых чисел. Переполнение возникает из-за того, что существуют операции, при выполнении которых компьютер дает не тот же результат, что на бумаге. К данным операциям чаще всего относят операции приведения типов как явных так и неявным приведением.. Результат: Переполнение целых чисел чаще всего вызывает переполнение буфера с последующим выполнением кода.. Пример реализации:. #include <stdio.h>. int main (int argc, char\* argv[]). {. const long MAX\_INPUT = 0x7fff;. short len;. if(argc > 1). {. len = strlen(argv[1]);. if(len < MAX\_INPUT). {. //основной функционал. }. else. {. return 0;. }. }. }. Комментарий: В случае, если в качестве аргумента будет введена строка, длина которой больше 32К, то переменная len станет отрицательной, поэтому будет выполнен «основной функционал» с неверными входными данными.. . Способы структурного тестирования:. Такая ошибка очень трудно выявляется на этапе анализа кода, необходимо отследить все операции с переменными, арифметические операции и приведения типов.. Способы поведенческого тестирования:. Такая ошибка также очень трудно выявляется. Можно использовать только рандомизированное тестирование. В ходе исследования при вызове программы в качестве параметров передаются строки символов, чтобы вызвать ошибку. Часто ошибки возникают, если длина строк составляет,: 127,128, 255, 32К, 64К-1, 64К. Применение существующих эксплойтов.. . Сравнение эффективности структурного и поведенческого тестирования:. Ошибки, связанные с переполнением целых чисел сложно выявляются при использовании обоих методов.. 4) Внедрение SQL-команд. . Ошибка возникает в случае отсутствия проверки параметров, водимых пользователем, при создании SQL-запросов (их конкатенации).. В результате реализации уязвимости может быть скомпрометирована машина и осуществлено раскрытие секретных данных.. Пример реализации:. …. gets(ID);. CString sSQL = “SELECT money FROM zarpl WHERE id=” + ID;. ….. Комментарий: пользователь может ввести любое значение идентификатора, либо вместо идентификатора ввести дополнительные операнды SQL-запроса, которые при конкатенации введенной строки со строкой из программы выведут недоступные ранее для пользователя данные.. . Способы структурного тестирования:. Для выявления ошибки используются метод сигнатурного анализа поиска операций взаимодействия с базой данных. Определяются места формирования запросов и проверки вводимых данных.. Способы поведенческого тестирования:. Данная ошибка достаточно трудно выявляется. Можно использовать только рандомизированное тестирование. В ходе исследования при вызове программы в качестве параметров формирования запросов передаются частично некорректные данные.. . Сравнение эффективности структурного и поведенческого тестирования:. Структурное тестирование эффективней поведенческого тестирования за счет наглядности процедур формирования запросов.. 5) Гонки. Гонки возникают в случае существования двух и более программ, выполняемые в разных контекстах (процессах или потоках). Эти программы могут блокировать друг друга, а также изменять один и тот же объект.. В результате реализации этой уязвимости, может быть осуществлено блокирование программы, аварийный останов программы.. . Способы структурного тестирования:. Метод поиска не эффективен без использования динамического тестирования.. Способы поведенческого тестирования:. Такая ошибка достаточно трудно выявляется. Можно использовать только тесты на быстрых многопроцессорных машинах. А также выявление коллизий в ходе выполнения программы на высокоскоростных машинах.. . Сравнение эффективности структурного и поведенческого тестирования:. Структурное и поведенческое тестирование не очень эффективны при выявлении гонок, однако с некоторой вероятностью их можно выявить только при поведенческом тестировании.. Как видно из приведенного анализа: тестирование программного обеспечения является мощным средством для выявления ошибок в работе программ, так и уязвимостей безопасности. За рубежом тестированию уделяется значительное внимание, эффективно функционирует специальная отрасль «Software Testing & Quality Assurance». Современные методы поведенческого тестирования позволяют выявлять уязвимости без исходных текстов программ и могут успешно применяться на российском рынке, для которого передача исходных текстов на тестирование задача почти невыполнимая.