Лабораторная работа №3

Анализ безопасности кода

Часть 2.
Методика оценки безопасности кода

Теоретические основы

В предыдущих лабораторных работах были рассмотрена и проанализирована безопасность программных систем по принципу «чёрного ящика». При создании программного обеспечения необходимо помнить об особенностях языка, которые позволяют создавать безопасный код. Безопасный код – это код, позволяющий минимизировать риски нарушения конфиденциальности и/или доступности данных при работе программного обеспечения в любых ситуациях.

При создании программных продуктов необходимо учитывать

– поведение приложения при вводе невалидных данных (ввод длинных строк, ввод некорректных данных, ввод данных некорректных типов),

– поведение приложения в аварийных ситуациях (при сбоях программного или аппаратного обеспечения),

– поведение приложения при больших нагрузках (сетевых, процессорного времени, оперативной памяти) и т. д.

Одним из решений обработки невалидных данных являются валидационные сообщения. Для корректной работы программы при больших нагрузках или в аварийных ситуациях необходимо:

– писать код таким образом, чтобы конфиденциальные данные из оперативной памяти не могли попасть на жёсткий диск или быть переданными по сети (кэш, дамп),

– корректная настройка окружения и условий запуска приложения.

Существует большое количество программ, позволяющих анализировать текущее состояние оперативной памяти, создавать дампы из оперативной памяти для выбранных процессов, анализировать дампы, созданные ранее, в том числе и после сбоев.

Среди таких программ можно выделить, например, средства, поставляемые с различными SDK. Например, для JDK такими средствами являются jvisualvm, jstack, jhat и т. д. Также есть ряд ПО, позволяющих работать с оперативной памятью напрямую. Примером таких программ могут быть Heap Inspector, Memoryze и т. д. Также с помощью различных профайлеров можно проанализировать состояние аппаратных ресурсов, в том числе и оперативной памяти, для выбранного процесса. Примером таких программ могут быть Jvisualvm, Jprofiler, cProfile и многие другие.

Инструментальные и вспомогательные средства

1. Виртуальная машина с установленной ОС (Windows, Linux).
2. IDE для создания приложения (VS, IntelliJ Idea, NetBeans, Eclipse).
3. Средства для анализа оперативной памяти (Heap Inspector, Memoryze, средства JDK и т. д.).
4. Предоставленный вариант предыдущей части лабораторной работы.

Задание для самостоятельной работы

1. На основе полученных результатов предыдущей части лабораторной работы создать методику оценки безопасности кода.
2. По созданной методике сравнить предоставленный алгоритм со своим из предыдущей части лабораторной работы.
3. Оформить отчёт о проделанной работе.

Отчётность

В качестве результата ожидается:

1. обоснование корректности разработанной методики;
2. демонстрация работы разработанного алгоритма предыдущей части лабораторной работы и предоставленного алгоритма;
3. анализ безопасности предоставленного алгоритма с точки зрения кода;
4. анализ безопасности работы предоставленного алгоритма в оперативной памяти;
5. сравнение своего варианта с предоставленным на основании созданной методики;
6. отчёт о проделанной работе.

Отчёт должен быть оформлен в соответствии с http://www.bsuir.by/m/12_100229_1_80040.pdf.

Отчёт должен содержать следующую информацию.

1. Постановка задачи.
2. Описание реализованного алгоритма в лабораторной работе №3.1 и предоставленного алгоритма.
3. Анализ безопасности предоставленного алгоритма.
4. Описание созданной методики.
5. Сравнение своего варианта реализации программы в лабораторной работе

№3.1 с предоставленным по созданной методике.

6. Вывод. В выводе должны быть представлена семантическая и прагматическая составляющие проделанной работы.

Список рекомендуемых источников

1. Ховард, М. Защищённый код // М. Ховард, Д. Лебланк — М.: Русская редакция, 2004. — 704 с.
2. Ховард, М. 24 смертных греха компьютерной безопасности. Библиотека программиста // М. Ховард, Д. Лебланк, Дж. Вьега — СПБ.: Питер, 2010. — 400 с.
3. Java Virtual Machine Monitoring, Troubleshooting, and Profiling Tool [Электронный ресурс]. – Электронные данные. – Режим доступа: http://docs.oracle.com/javase/6/docs/technotes/tools/share/jvisualvm.html Дата доступа: 20.08.2014.
4. Find Evil in Live Memory [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.mandiant.com/resources/download/memoryze Дата доступа: 20.08.2014.