ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АППАРТНОГО МЕДИЦИНСКОГО КОМПЛЕКСА «АКУТЕСТ»

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Сухарев А.А.

Анисимов В.Я. – доцент, канд. физ.-мат. Наук

Аппаратный медицинской комплекс «Акутест» предназначен для проведение бесконтактной физиотерапии на основе воздействия электромагнитных импульсов на различные участки тела человека. Устройство выполнено в компактном корпусе, размеры которого не превышают размеры современного сотового телефона. Комплекс снабжен встроенным аккумулятором, что делает его портативным и автономным. Для заряда аккумулятора и связи с компьютером комплекс имеет один порт Mini-USB. Принцип действия медицинского комплекса заключается в генерации различной комбинации электромагнитных сигналов. Сигнал, имеющий фиксированную частоту и продолжительность действия называет *шагом*. Основными характеристиками шага являются частота воздействия и время выполнения шага. Последовательности из различных шагов формируют *программы*. Задачей аппаратного комплекса является генерация сигналов по составленным программам.

Программное обеспечение комплекса в первую очередь должно базироваться на работе с протоколом USB. USB - последовательный интерфейс передачи данных для среднескоростных и низкоскоростных периферийных устройств в вычислительной технике. Протокол требует от оконечного устройства поддержания достаточно сложного алгоритмического стека как для непосредственно обмена по шине USB, так и для поддержания сопутствующих функций типа инициализации или ответов на служебные сообщения. Несмотря на заявленную универсальность, USB-устройства, даже принадлежащие стандартным классам, большей частью требуют программной поддержки и отдельных драйверов на хосте USB. Так, современная операционная система Windows при подключении внешнего COM-порта или GPS-навигатора (которые относятся к одному стандартному классу коммуникационных устройств), требует для каждого из устройств отдельного драйвера.

Семейство операционных систем Windows, начиная с Windows XP Service Pack 2, имеют в своем комплекте поставки стандартный драйвер WinUSB для работы с любыми USB устройствами. WinUSB состоит из двух основных частей: WinUSB.sys и WinUSB.dll. WinUSB.sys представляет собой драйвер уровня ядра, который может быть установлен как фильтр или как функциональный драйвер в стек драйверов USB. WinUSB.dll представляет собой программный интерфейс для вызова процедур из WinUSB.sys, когда он установлен как функциональный драйвер устройств. Для подключение аппаратного комплекса «Акутест» необходима установка драйвера, зависящего от внутреннего контроллера USB. На данный момент используется два типа контроллеров: C201x компании Silicon Laboratories и контроллер FT232 производства Future Technology Devices International (FTDI). Каждый из них требует установки собственного проприетарного драйвера, который добавляется в стек WinUSB. USBXpress Driver представляет собой драйвер для работы с устройствами, которые оборудованы контроллерами от компании Silicon Laboratories. Последняя версия драйвера реализована на базе WinUSB и предоставляет разработчику упрощенное API для работы с устройствами, поддерживающими USBXpress. В отличие от USBXpress, программный интерфейс D2XX от компании FTDI базируется на базе собственного проприетарного драйвера.

Согласно требованиям, программное обеспечение для комплекса «Акутест» должно быть реализовано на платформе .NET Framework, которая имеет мощный встроенный механизм, позволяющий исполнять неуправляемый код из управляемого. Данный механизм называется P/Invoke (Platform Invocation Services). В него также входит механизм *маршалинга* (от англ. marshaling) – набор правил по передачи данных между управляемым и неуправляемым кодом. При помощи этих средств можно реализовать оболочку вокруг неуправляемого кода драйвера для вызова функций из управляемого кода**.** Любая работа с периферийными устройствами подразумевает собой операции ввода-вывода. .NET Framework имеет достаточно мощный и развитый механизм для выполнения таких операций. Платформа поддерживает бинарный, строковый, сжатый, шифрованный ввод-вывод, а также запись и чтение в файл, сеть, память. Однако, если устройство должно принимать определенные команды и посылать на них ответы (или генерировать ошибки), то использование стандартных средств ввода-вывода недостаточно без соответствующей надстройки. Для решения данной проблемы предлагается реализовать паттерн «запрос-ответ». Преимуществом данного подхода является то, что исполняемый код не будет зависеть от времени ожидания ответа устройства и то, что все обращения к устройству будут выполняться асинхронно. Отдельный программный поток будет выполнять все операции ввода-вывода и посылать в вызывающий программный поток ответы на соответствующие запросы.

В основе реализации программного комплекса «Акутест» лежит понятие *канала*. Канал – объект, который принимает запросы, обрабатывает их, посылает данные в устройство и обрабатывает результат. Одним из способов построения каналов является использование двух очередей. Одна очередь служит для приема сообщений (*входная очередь*), а вторая – для обработки ответов (*выходная очередь*). Использование такого подхода позволяет достигнуть достаточной гибкости при работе с устройством, гарантиро-

вать целостность системы, а также реализовать асинхронный ввод-вывод, что немаловажно для драйверов, поддерживающих только синхронные операции.

Список использованных источников:

1. Медицинский аппаратный комплекс «Акутест». Справочное руководство. - Москва, 2014 - 14 с.
2. Jeffrey Richter. CLR via C# (4th edition). - Microsoft Press, 2012 - 896 с.

ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ СОВРЕМЕННЫХ ПРОГРАММНЫХ ПЛАТФОРМ JAVA И .NET

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Марчук М. С., Саскевич А. В., Цыркунов Д. А.

Стройникова Е. Д. – ассистент кафедры информатики

В данной работе исследованы особенности платформ Java и .NET в сфере безопасности и защищенности по следующим критериям: 1) обеспеченность платформ собственными библиотеками цифровой подписи и шифрования данных; 2) защищенность скомпилированного кода от декомпиляции и дизассемблерирования, деобфускация кода и обратное восстановление; 3) безопасность кода во время выполнения; 4) возможность изменения скомпилированного кода; 5) внедрение в процесс, запущенный на платформе Java или .NET, и возможность изменения данных приложения во время работы,

6) цифровая подпись готовых сборок и приложений; 7) безопасность Android приложений; 8) наличие устранимых уязвимостей в платформах Java и .NET и возможные уязвимости в современных версиях Java 8 и .NET Framework 4.5. Исследование проводилось посредством реализации программных проектов на основе данных платформ.

1. Обеспеченность платформ собственными библиотеками цифровой подписи и шифрования данных

В языке Java существуют различные стандартные библиотеки шифрования и защиты данных, например, java.security.*. Библиотека предоставляет широкий набор интерфейсов и классов, предназначенных для настройки безопасности своего проекта. Данный пакет содержит такие классы, как, например: *AccessControlContext* – используется для принятия решения о предоставлении доступа к ресурсам системы; *KeyStore* – этот класс представляет хранилище для криптографических ключей; различные криптографические классы и алгоритмы шифрования. Также этот пакет поддерживает ряд классов для операций шифрования данных. Многие классы позволяют на своей основе реализовать собственные программные компоненты шифрования и защиты.

Платформа .NET располагает многими полезными классами и службами, которые облегчают написание защищенного программного кода и позволяют системным администраторам настраивать разрешения кода для доступа к защищенным ресурсам. Кроме того, среда выполнения и .NET Framework располагают полезными классами и службами, которые облегчают применение криптографии и системы безопасности, основанной на ролях. Платформа .NET предоставляет реализации многих стандартных криптографических алгоритмов. Эти алгоритмы просты в использовании и по умолчанию имеют наиболее безопасные из возможных значений свойств. Так, как и в Java, предоставляются интерфейсы классов, поэтому разработчик может написать свою реализацию методов.

2. Защищенность скомпилированного кода от декомпиляции и дизассемблерирования, деобфускация кода и обратное восстановление

Скомпилированный Java-код представляет собой набор класс-файлов. При запуске скомпилированный файл проходит так называемый процесс верификации. Скомпилированный байт-код доступен для декомпиляции, в результате чего может быть получен исходный код, аналогичный изначальному, однако из-за несовершенства современных инструментов в некоторых случаях может быть получен код со вставками байт-кода. Для защиты исходного кода могут быть применены обфускаторы – специальные инструменты, изменяющие названия методов, переменных, а также изменяющие сам код таким образом, что его практически невозможно расшифровать для последующей модификации. Так, например, инструмент *Zelix KlassMaster* изменяет байт-код таким образом, что впоследствии он не будет поддаваться декомпиляции.

Так как код платформы .NET, аналогично Java, работает в исполняемой среде, действующей в виртуальной машине, платформа .NET имеет некоторое общее с Java устройство. Однако .NET не производит верификации скомпилированного кода, все необходимые проверки осуществляются в процессе загрузки и исполнения. Скомпилированный код может быть преобразован в исходный код специальным инструментом *IL Disassambler*, что приводит к тому, что IL–код легко восстановить и модифицировать. В то же время существуют инструменты обфускации, изменяющие код и его структуру, при этом сохраняя его работоспособность.

3. Безопасность кода во время выполнения

В обоих случаях, для Java и .NET, исполняемый код проходит различные проверки, исключающие ошибочные или некорректные операции. Каждая из платформ предупредит пользователя о таких вещах, как выход за пределы разрядной сетки при выполнении арифметических операций, исключение при рабо-