УДК 004.223

ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ИГРОВОГО ПРИЛОЖЕНИЯ «CUBE» ПОД УПРАВЛЕНИЕМ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID

И.В. ЛАРИОНОВ
(Представлено: Ю.Н. КРАВЧЕНКО)

Рассматриваются способы хранения данных для мобильного игрового приложения под управлением операционной системы Android, а также способы их защиты.

Введение. Unity3D – кросплатформенный игровой движок от компании Unity Technologies. Unity позволяет создавать приложения, работающие под более чем 20 различными операционными системами. Движок поддерживает множество популярных форматов моделей, звуков, материалов и текстур.

Проект в Unity делится на сцены (уровни) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые игровые объекты — объекты, которые не имеют модели. Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты.

Все эти данные необходимо защищать от копирования, нелегального использования, пиратского распространения, профессионального анализа и взлома.

Основная часть. При компиляции проекта Unity создается исполняемый файл игры, а в отдельной папке – данные игры (включая все игровые уровни и динамически подключаемые библиотеки).

Способы хранения данных. Информационные объекты, взаимосвязи между ними, а также способы влияния их на систему должны быть представлены в конечном программном продукте с помощью модели данных. Отсюда следует, что модель данных - есть совокупность структур данных и операций по их обработке.

Разрабатываемое приложение будет включить большое количество заранее подготовленной для пользователя информации, и необходимо подобрать удобный и эффективный способ ее хранения и обработки.

Отметим тезисно основные категории данных, которые необходимо хранить для дальнейшей интерпретации игрой:

- информация об игровых уровнях;
- информация о ценах во внутри игровом магазине;
- заранее заданные настройки.

Для хранения приведенных выше данных подходят 2 варианта: хранение в формате XML и бинарная сериализация.

Идеальным вариантом будет не выбирать среди предложенных технологий, а использовать оба варианта, взяв лучшее из каждой и применив в идеально подходящей для текущей задачи ситуации.

Большим плюсом бинарной сериализации является небольшой размер выходных файлов. Данные очень компактно складываются в файлы и сжимаются самыми эффективными алгоритмами, но существует один недостаток: невозможность чтения данных без специального программного обеспечения.

Бинарная сериализация отлично подходит для хранения самых разнообразных настроек, цен и прочих конфигурационных параметров.

Способ хранения уровней требует прозрачности и простого доступа к данным. Данные должны с легкостью читаться при помощи выбранного языка программирования и при этом быть очевидными при простом просмотре документа. Именно поэтому для хранения игровых уровней было выбрано использование документов в формате XML.

Защита данных. Для Windows-сборок Unity компилирует и сохраняет исходный код всех игровых скриптов в директорию Managed. Находится код в библиотеках: Assembly-CSharp.dll, Assembly-CSharp-firstpass.dll и Assembly-UnityScript.dll. Для декомпиляции и просмотра managed-кода .NET библиотек существуют довольно удобные и при этом бесплатные утилиты: IlSpy и dotPeek.

Данных подход особенно эффективен для наших целей: Unity очень скупо оптимизирует исходный код игровых скриптов, практически не изменяя его структуру, а также не скрывает названия переменных. Это позволяет с легкостью читать и понимать декомпилированый материал.

В таких случаях разработчикам приходится самим беспокоиться о безопасности своего кода. Для таких целей обычно используют обфускаторы.

Обфускация - это процесс, в результате которого код программы приобретает вид, трудный для анализа. Обфускация, осуществляется с целью защиты программного кода и алгоритмов, которые он реализует, от чужих глаз. Но в большинстве случаев обфускация имеет массу побочных эффектов. В особо

неприятных обстоятельствах обфускатор может сделать программу совершенно непригодной к выполнению, в менее тяжёлых случаях в программе могут появиться новые ошибки. Поэтому обфускацию необходимо применять с максимальной осторожностью.

В AssetStore существует множество готовых решений, но большинство из них платные. Бесплатные версии, как правило, ограничены или имеют невысокую эффективность.

Большинство ресурсов Unity-проекта упаковываются в файлы проприетарного формата с расширениями .assets и .resources. Несмотря на закрытость форматов, существуют инструменты для распаковки таких файлов. Так, например, Unity Assets Explorer способен извлечь большинство текстур и шейдеров из игры. Полученные в результате текстуры будут иметь формат DDS, который можно открыть с помощью Windows Texture Viewer. Шейдеры извлекаются в уже скомпилированным виде и решений для их декомпиляции не существует. Тем не менее, это обстоятельство не мешает импортировать и использовать полученные шейдеры в другом Unity-проекте.

Трёхмерные модели в Unity-сборке расположены в различных ресурсах, а некоторые из них и вовсе могут генерироваться во время игры. Получить такие данные можно прямиком из памяти графического ускорителя. Когда игра запущена, вся информация о текстурах и моделях, видимых на экране, находится в памяти видеокарты. С помощью утилиты 3D Ripper DX можно извлечь всю эту информацию и сохранить в формате, понятном 3D-редакторам.

PlayerPrefs — это класс из стандартной библиотеки Unity, который позволяет сохранять данные в долговременную память устройства. Представляет собой пару ключ - значение. Он часто используется разработчиками для хранения различных настроек, достижений, прогресса игрока и другой информации о состоянии игры. На ОС Windows эти данные сохраняются в системном реестре. В других операционных системах данные сохраняются на устройстве в локальной папке приложения в специальном файле. В большинстве случаев к ним можно легко получить доступ и модифицировать с помощью текстового редактора. Например, в Windows достаточно использовать встроенную утилиту RegEdit чтобы модифицировать любые значения PlayerPrefs, изменяя тем самым конфигурацию и статус игры.

Простейшим способом противодействия является кодирование сохраняемых данных, например, base64. Данный способ не слишком эффективен, но может дать начальную защиту от просмотра. Проверить, изменялись ли данные без ведома помогут хеш-функции: сравнив контрольные суммы хранимых данных, мы сможем убедиться, что никто и ничто, кроме нашего кода эти данные не изменяло.

Для более надёжной защиты необходимо использовать шифрование или различные комбинации из перечисленных методов.

Также можно реализовать свой формат сохранений. Благодаря Mono Unity поддерживает работу с файловой системой. Таким образом можно сериализовать все необходимые данные, применить надёжное шифрование и сохранить в надёжном месте.

Заключение. В данной статье были рассмотрены способы хранения данных для игрового приложения под управлением операционной системы Android, а также способы их защиты.

К сожалению, существует не так уж много способов защитить игру от взлома. Будучи установленной на пользовательское устройство, она фактически раскрывает все текстуры, модели и исходный код. Если кто-то захочет декомпилировать приложение и украсть ресурсы — это лишь вопрос времени.

Невзирая на это, существуют действенные методы, которые позволят серьёзно усложнить жизнь злоумышленникам.

ЛИТЕРАТУРА

1. Guide to Storage Encryption Technologies for End User Devices / Karen Scarfone, Murugiah Souppaya, Matt Sexton. – November 2007. – 40 p.
2. Базы данных и модели данных [Электронный ресурс]. Режим доступа: http://edu.tltsu.ru. Дата доступа: 22.09.2018.
3. Unity - Scripting API: PlayerPrefs [Электронный ресурс]. Режим доступа: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html. Дата доступа: 22.09.2018.
4. A practical tutorial to hack (and protect) Unity games [Электронный ресурс]. Режим доступа: https://www.alanzucconi.com/2015/09/02/a-practical-tutorial-to-hack-and-protect-unity-games/. Дата доступа: 22.09.2018.