Лабораторная работа:
Проверка подписи загружаемых модулей в Genode

Цель

Реализовать прототип модуля init с функцией контроля целостности загружаемых модулей, используя механизм электронной цифровой подписи (ЭЦП). Ознакомиться с процессом загрузки ОС, принципами формирования ЭЦП и интегрировать полученные инструменты в систему сборки Genode OS Framework.

1. Общая информация

Для начала разберемся, что представляет собой ЭЦП. Википедия дает следующее определение:

«Электронная подпись предназначена для идентификации лица, подписавшего электронный документ, и является полноценной заменой (аналогом) собственноручной подписи в случаях, предусмотренных законом.

Использование электронной подписи позволяет осуществить:

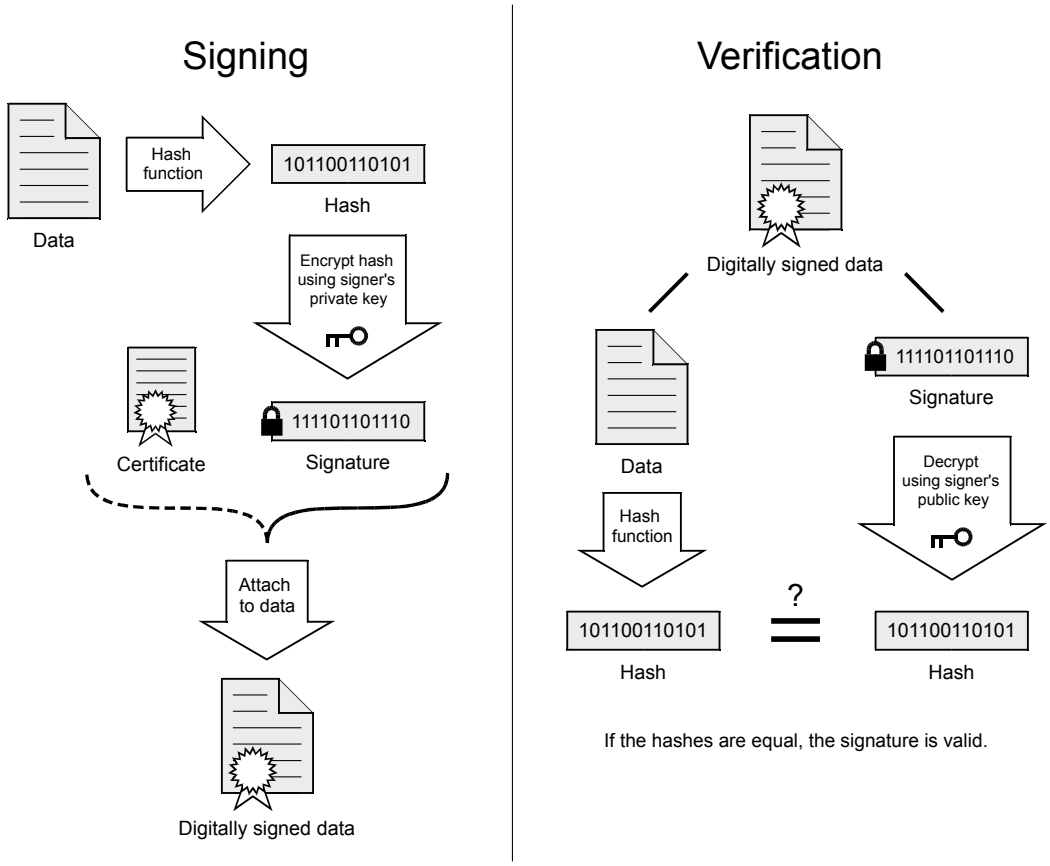
- Контроль целостности передаваемого документа: при любом случайном или преднамеренном изменении документа подпись станет недействительной, потому что вычислена она на основании исходного состояния документа и соответствует лишь ему.

- Защиту от изменений (подделки) документа: гарантия выявления подделки при контроле целостности делает подделывание нецелесообразным в большинстве случаев.

- Невозможность отказа от авторства. Так как создать корректную подпись можно, лишь зная закрытый ключ, а он известен только владельцу, он не может отказаться от своей подписи под документом.

- Доказательное подтверждение авторства документа: Так как создать корректную подпись можно, лишь зная закрытый ключ, а он известен только владельцу, он может доказать своё авторство подписи под документом. В зависимости от деталей определения документа могут быть подписаны такие поля, как «автор», «внесённые изменения», «метка времени» и т. д.» [1].

Для реализации ЭЦП наиболее часто используются асимметричные криптографические алгоритмы. Для подписи генерируется пара ключей: открытый и закрытый. Подпись и проверка выполняются в соответствии со следующей структурной схемой:



Исходя из этого к ключам предъявляются следующие требования:

- закрытый ключ должен надежно храниться у производителя ПО. Именно им производится подпись модулей;
- открытый ключ сохраняется в устройстве. Нет необходимости защищать этот ключ от чтения, но важно обеспечить невозможность его перезаписи.

В качестве алгоритмов, используемых для ЭЦП в России, рекомендуется применять ГОСТ Р 34.10-2012 [2] и ГОСТ Р 34/11-2012 [3] в качестве хеш-функции. В данной лабораторной работе рекомендованными алгоритмами являются ECDSA и SHA256 в качестве хеш-функции. В репозитории sss13 добавлены сторонние открытые библиотеки, реализующие данные алгоритмы micro-ecc [4] и SHA2 [5]. Открытый GOST engine из OpenSSL не использован по причине невозможности динамической линковки процесса init.

В качестве платформы будет использоваться Fiasco.OC (base-foc) для архитектуры x86_32. На самом деле не потребуется добавлять никакого платформо-специфичного кода, но формирование загрузочного образа в системе сборки является платформо-зависимым.

Рассмотрим загрузку образа, построенного на базе Genode+Fiasco.OC, более подробно:

1. Запускается загрузчик, в данном случае GRUB, все модули, указанные в конфигурационном файле загрузчика, загружаются в соответствии со спецификацией Multiboot. Управление передается модулю bootstrap.

2. bootstrap выполняет операции необходимые для старта ядра, такие как:

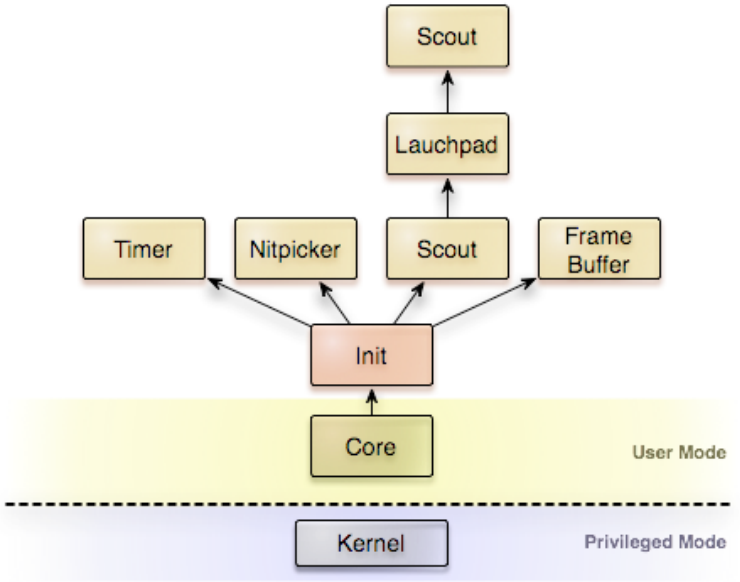
- сканирование доступной памяти;
- перемещение модулей в памяти (sigma0 и root task должны быть расположены в определенных регионах, чтобы ядро при старте могло их запустить);
- непосредственно запуск ядра.

3. Ядро выполняет инициализацию системы, запускает sigma0 и root task.

4. Запускается модуль core (являющийся root task в Genode), который инициализирует сервисы Genode и запускает модуль init.

5. init последовательно загружает указанные в конфигурации модули (файл config) и запускает их выполнение.

Процесс запуска изображен на рисунке [6]:



Более подробно модуль init и создание процессов в Genode были рассмотрены в лекции "Genode Architecture" [7] прошлогодней летней школы и в статье "Configuring the init process of Genode" [8], в разделе документации на сайте Genode.

2. Порядок выполнения и рекомендации

1. Для начала работы необходимо подготовить окружение:

- форкнуть наш репозиторий на GitHub: https://github.com/Ksys-labs/genode;
- переключиться на ветку **sss13**;
- подготовить библиотеки:

```
make prepare -C libport PKG=libc
make prepare -C sss13
```

В репозитории уже выполнена предварительная подготовка системы сборки и исходного кода. "Заготовка" утилиты для подписи с использованием рекомендованных библиотек находится в

sss13/tools/signtools, там же находятся утилиты для добавления значения подписи в конфигурационный файл и генерации ключей.

Оригинальный исходный код init находится в директориях *os/src/init* и *os/include/init*. Для данной работы заготовка исходного кода копирована в директории *sss13/src/initsig* и *sss13/include/initsig*. Этот исходный код будет использоваться в дальнейшем описании.

Замечание: при сборке Genode не используются стандартные библиотеки C и C++, они могут быть подключены только при условии динамической линковки модуля, что не возможно для init. Основные стандартные функции могут быть найдены в base/include/util и base/include/base.

2. Добавить расчет подписи для каждого исполняемого модуля и запись полученного хеша в конфигурацию на этапе создания образа.

Необходимо доработать утилиту для подписи *sss13/tool/signtools/signfile.cc*. Эта утилита должна получать на входе имя файла с данными для подписи, имя файла с содержимым закрытого ключа и выводить в stdout значение подписи в виде строки. Это значение при сборке с помощью утилиты *signfile.py* будет вставлено в конфигурационный файл.

Доступ к входящим в образ файлам в Genode осуществляется через ROM Session. Но для dataspace, относящегося к открытому файлу, невозможно получить реальный размер файла, размер dataspace всегда кратен размеру страницы памяти (4096 байт). Поэтому для получения хеша, содержимое файла должно быть дополнено нулями до размера, кратного размеру страницы и затем вычислен хеш.

Используя полученный хеш и закрытый ключ, получить подпись. Подпись добавляется в конфигурационный файл на этапе сборки в виде `<signature value="xxx">`. Пример:

```
        <start name="timer">
            <resource name="RAM" quantum="20M" />
            <provides><service name="Timer" /></provides>
            <signature
value="bc8d2e767f7da92b6216796eaff7ec3883597da0f751b5248baa1dae763a17bfc52b72280
b3a74c49ba9c0e459adc3d72d0af58622a60440018d3f0713028650" />
        </start>
```

3. Добавить проверку подписи в init

Модуль initsig (*sss13/src/initsig/main.cc*) выполняет следующие действия:

- инициализацию динамического линкера, если найден ld.lib.so (строки 154-157);
- регистрацию сигнала, используемого для уведомления init об изменении конфигурации (строки 167-169);
- определение роутинга сервисов (строки 178-186);
- и создание потомков (строки 188-219), заканчивающееся параллельным стартом всех

потомков;

- оставшийся код отвечает за перезапуск потомков в случае изменения конфигурации на лету.

В данный код добавлена обработка исключения `InitSig::Child::Signature_failed`, которое должно генерироваться классом `InitSig::Child` в случае каких-либо ошибок при проверке ЭЦП.

За создание потомка отвечает класс `InitSig::Child` (*sss13/include/initsig/child.h*), который непосредственно создает процесс, именно в его конструктор проще всего добавить проверку ЭЦП. Рассмотрим этот код и определим функционал, который необходимо реализовать:

- получение значения сигнатуры из конфигурационного файла для данного модуля (строки 486-502);
- конвертирование сигнатуры из строки в бинарный вид, пригодный для дальнейшего использования в проверке (**необходимо реализовать**, заготовка в строках 504-506). Можно использовать функцию:
  ```
  template <typename T> inline size_t ascii_to(const char *s, T *result, unsigned base = 0);
  ```
- получение указателя на данные модуля и его длину (**необходимо реализовать**, строки 508-510). Используйте Dataspace_client и RAM, RM сервисы;
- расчет хеша для данных загружаемого модуля (**необходимо реализовать**, строки 512-514);
- проверка ЭЦП (**необходимо реализовать**, строки 522-523);
- в случае успешной проверки ЭЦП производится создание нового модуля (строки 527-528).

Открытый ключ в рамках этой лабораторной работы может быть просто вкомпилен в init (*sss13/src/initsig/main.cc* строка 19).

3. Тестирование

- подготовить сборочную директорию для платформы foc_x86_32 и перейти в нее;
- включить репозитории **libport** и **sss13** в конфигурацию сборки;
- включить генерацию ЭЦП, используя спек **sign**:
  ```
  echo 'SPECS += sign' >> etc/specs.conf
  ```
- собрать утилиты signtools:
  ```
  make -C ../sss13/tool/signtools
  ```
- установить утилиты в директорию bin:
  ```
  PREFIX=`pwd`/bin make install -C ../sss13/tool/signtool
  ```

- сгенерировать пару ключей:

  ```
  bin/generate-keys bin/ > ../sss13/include/initsig
  ```

- запустить скрипт lab.run:

  ```
  make run/lab
  ```

- убедиться, что проверка подписей работает и подписи валидны;
- изменить содержимое файла hackme, запустить руками и убедиться, что подпись не сходится:

  ```
  cd var/run/
  ghex lab/genode/hackme
  ../../../tool/create_iso iso ISO=lab
  qemu-system-i386 -no-kvm -m 64 -nographic -cdrom lab.iso
  ```

- исправить подпись в конфигурации для измененного файла:

  ```
  ../../bin/signfile lab/genode/hackme ../../bin/private.key
  ```

- заменить полученным значением подпись в конфигурации lab/genode/config для модуля hackme
- убедиться, что проверка проходит:

  ```
  ../../../tool/create_iso iso ISO=lab
  qemu-system-i386 -no-kvm -m 64 -nographic -cdrom lab.iso
  ```

3. Выводы

Сделать выводы по работе, а именно:

- какие проблемы имеет эта реализация?

- что нужно доработать, чтобы использовать такую реализацию в реальных применениях, - какие еще могут быть варианты контроля целостности загружаемых модулей для Genode?

Ссылки

[1]: http://ru.wikipedia.org/wiki/Электронная_подпись

[2]: http://ru.wikipedia.org/wiki/ГОСТ_Р_34.10-2012

[3]: http://ru.wikipedia.org/wiki/ГОСТ_Р_34.11-2012

[4]: https://github.com/kmackay/micro-ecc

[5]: http://www.aarongifford.com/computers/sha.html

[6]: http://genode.org/documentation/developer-resources/img/setup.png

[7]: http://sss.ksyslabs.org/ru/prev/sss-12/genode-architecture

[8]: http://genode.org/documentation/developer-resources/init