Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б11-мм

Добавление в ОСРВ Embox поддержки библиотеки OpenSSL

Пилюк Дмитрий Алексеевич

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., Д. В. Луцив

Консультант:
ген. директор ООО «Ембокс» А. В. Бондарев

Санкт-Петербург
2024

Оглавление

Введение

С давних времен люди сталкиваются с задачей безопасного хранения и передачи информации. В современном мире безопасность данных становится все более актуальной темой. Криптографические методы являются одним из основных инструментов для защиты информации. Различные алгоритмы используются для шифрования паролей, защиты файлов и директорий, безопасной передачи данных, создания цифровых подписей и многих других задач. Криптография в операционных системах играет важную роль в обеспечении безопасности пользовательских данных.

Не является исключением и операционная система реального времени(ОСРВ) Embox. В ней реализованы и используются некоторые методы шифрования.

Целью данной работы является добавление поддержки библиотеки OpenSSL и проверка ее работоспособности.

1. Постановка задачи

Целью работы является расширение криптографических возможностей в ОСРВ Embox путём портирования библиотеки OpenSSL. Для её выполнения были поставлены следующие задачи:

1. Рассмотреть реализованные алгоритмы шифрования в Embox.
2. Портировать OpenSSL.
3. Провести апробацию.

2. Обзор

2.1. ОСРВ Embox

Embox [1] — это конфигурируемая операционная система реального времени, предназначенная для встраиваемых систем с ограниченными ресурсами. Разработка начиналась в 2010 году на кафедре системного программирования Математико-механического факультета СПбГУ как студенческий проект.

Исходный код ОСРВ Embox представляет собой модульную систему, что позволяет гибко настраивать конечную сборку под конкретную задачу, с возможностью включить только минимально необходимые компоненты, либо получить ОС с богатой функциональностью. Эта модульность достигается за счёт использования системы сборки Mybuild.

Embox предоставляет POSIX-совместимость, благодаря чему позволяет портировать многие Linux приложения. Так же в состав проекта входит стандартная библиотека языка C.

Embox может применяться в робототехнике, благодаря тому, что совмещает в себе возможность выполнять сложные функциональные задачи и является системой реального времени. Кроме того ОС имеет развитый сетевой стек и набор различных приложений и библиотек и при этом низкие аппаратные требования, поэтому хорошо подходит для использования в устройствах интернета вещей.

2.2. Задачи криптографии в ОС

В книге Э. Таненбаума «Современные операционные системы» [5] приводятся несколько видов шифрования:

1. Симметричное шифрование — алгоритмы при котором используется секретный ключ для шифрования и дешифрования.

2. Асимметричное шифрование — алгоритмы при котором используется открытый и закрытый ключи, является более надежным, но,

обычно, более ресурсозатратным. Открытый ключ используется для шифрования, а закрытый для дешифрования.

3. Хэш-функция - алгоритм шифрования, при котором данные, превращаются в какой-то код. Нет возможности из кода получить данные обратно.

Алгоритмы симметричного и асимметричного шифрования используются для безопасной передачи информации.

Хэш-функции используются для шифрования и хранения каких-то данных, например, чтобы не хранить сами пароли в системе, хранят их хэши, и даже если злоумышленник узнает хэш, он не сможет получить пароль. Так же хэш-функции используются для проверки целостности информации.

Комбинации асимметричного шифрования и хэш-функций используются для создания электронной подписи.

2.3. Алгоритмы в ОСРВ Embox

В Embox реализованы:

- MD5 — алгоритм хэщирования. Может использоваться для проверки целостности информации и для хранения хэшей паролей. Сейчас не рекомендуется использования для хранения паролей, так как существуют способы нахождения коллизий.

- CRC16 и CRC32 — хэш функции общего назначения, применяются для проверки целостности данных.

- DES — алгоритм симметричного шифрования. На сегодняшний день считается недостаточно безопасным, рекомендуется использовать Triple DES или AES.

- Base64 — алгоритм кодирования двоичных данных с помощью 64 символов ASCII.

В проекте присутствует набор криптографических функций, но некоторые из функций считаются не такими устойчивыми на сегодняшний день.

2.4. Криптографические решения в Open Source

Можно реализовывать необходимые криптографические функции с нуля. Но для этого необходимо точно определиться каких задач необходимо достич и такая реализация, возможно, будет слишком узконаправленной. Кроме того, существует большое количество библиотек с открытым кодом, где реализованы алгоритмы шифрования под разные задачи.

Одна из наиболее распространенных библиотек — это OpenSSL [3], криптографичесская библиотека, которая поддерживает большое количество криптографических функций и расширение SSL/TLS, необходимое для общения по протоколу HTTPS.

Другая библиотека с алгоритмами шифрования — это Mbed TLS [2]. Основная цель данной библиотеки — поддержка протокала TLS. При этом библиотека является легковесной и позиционируется как программное обеспечение для встраиваемых систем. Кроме того Mbed TLS имеет хорошую документацию и гибко кофигурируется, это будет полезным для портирования на новую ОС.

2.5. OpenSSL

Набор инструментов OpenSSL включает в себя [4]:

- libssl — реализация протокола TLS до версии TLSv1.3, DTLS до версии DTLSv1.2 и протокола QUIC.
- libcrypto — полноценная криптографическая библиотека общего назначения. Также содержит основы протокола TLS.
- openssl — консольное инструмент. Может использоваться для:
  - создания ключевых параметров;

– создания сертификатов X.509, CSR и CRL;

– шифрования и дешифрования;

– SSL/TLS/DTLS тестов клиента и сервера

– QUIC-клиентских тестов;

– обработки подписанной или зашифрованной почты S/MIME;

– и другого.

3. Реализация

3.1. Добавление стронней библиотеки

В ОСРВ Embox сторонние библиотеки добавляются в директорию embox/third-party/lib. Создадим в ней модуль openssl3, где будет располагаться Mybuild и прочие файлы необходимые для сборки.

Листинг 1: Mybuild файл для OpenSSL.

```
package third_party.lib

@BuildArtifactPath(
cppflags= \
    "-I$(EXTERNAL_BUILD_DIR)/third_party/lib/openssl3/install/include")
@Build(stage=2,script="$(EXTERNAL_MAKE)")
module openssl3 {
        source "localtime.c"

        @AddPrefix("^BUILD/extbld/^MOD_PATH/install/lib")
        source "libssl.a", "libcrypto.a"

    depends embox.compat.posix.stubs
}
```

Для модуля OpenSSL указываем путь, где будут располагаться header файлы. Указываем скомпилированные файлы, которые понадобятся в сборке, и добавляем необходимую для зависимость. Также указываем, что для сборки вызывается сторонний Makefile.

3.2. Makefile

Для установки OpenSSL добавим дополнительный Makefile, служащий "оберткой" для вызова скрипта сборки поставляемого в исходниках. Весь процесс условно поделим на 4 части:

- загрузка исходников;

- настройка конфигурации;
- сборка исходников;
- установка.

Для загрузки исходников указываем URL путь. Также применяем исправления исходников необходимые для сборки и подключаем модуль EXTBLD_LIB.

Листинг 2: Makefile. Загрузка исходников.

```
PKG_NAME := openssl
PKG_VER := 3.3.1
PKG_SOURCES := \
https://github.com/$(PKG_NAME)/$(PKG_NAME)/releases/download\
/$(PKG_NAME)-$(PKG_VER)/$(PKG_NAME)-$(PKG_VER).tar.gz
PKG_PATCHES := embox.patch
PKG_MD5     := 8a4342b399c18f870ca6186299195984
include $(EXTBLD_LIB)
```

Первая цель $(CONFIGURE). Копируем файл embox.conf в директорию Configure. Затем запускаем скрипт конфигурации под платформу embox, дополнительно указывая аргументы, необходимые для сборки.

Листинг 3: Makefile. $(CONFIGURE).

```
$(CONFIGURE) :
        export EMBOX_GCC_LINK=full; \
        cp $(THIRDPARTY_DIR)/lib/openssl3/embox.conf \
        $(PKG_SOURCE_DIR)/Configurations
        cd $(PKG_SOURCE_DIR) && ( \
                ./Configure \
                embox \
                --cross-compile-prefix=\
                $(ROOT_DIR)/mk/extbld/arch-embox- \
                --prefix=$(PKG_INSTALL_DIR) \
```

```
                -DPEDANTIC \
                -I$(THIRDPARTY_DIR)/lib/openssl3 \
                "$(EMBOX_CFLAGS)" \
        )
        touch $@
```

Цели $(BUILD) и $(INSTALL) представляют из себя вызов цели build и install для OpenSSL, соответственно.

Листинг 4: Makefile. $(BUILD) и $(INSTALL)

```
$(BUILD) :
        cd $(PKG_SOURCE_DIR) && ( \
                $(MAKE) build_sw \
                    MAKEFLAGS='$(EMBOX_IMPORTED_MAKEFLAGS)'; \
        )
        touch $@

$(INSTALL) :
        cd $(PKG_SOURCE_DIR) && ( \
                $(MAKE) install_sw \
                    MAKEFLAGS='$(EMBOX_IMPORTED_MAKEFLAGS)'; \
        )
        touch $@
```

3.3. Конфигурация OpenSSL

OpenSSL содержит большое количество готовых конфигураций и для разных операционных систем и платформ. Туда входит выбор компилятора, его ключей, списки для включения и выключения отдельных модулей. Для этого используется язык Pearl. Создадим файл Embox.conf, который мы используем для конфигурирования.

Листинг 5: Embox.conf. Файл конфигурации для сборки OpenSSL.

```
my %taragets = (
        "embox" => {
        inherit_from     => [ "gcc" ],
        disable          => ["asm", "dso", "async", "shared",
        "apps", "afalgeng"],
        bn_ops           => "THIRTY_TWO_BIT",
    },
)
```

Модули dso и shared отключаются, так как в Embox используется статическая линковка. Модуль afalgeng отключем, т.к. это движок поддерживается в Linux.

3.4. Сборка

Остается добавить небольшие патчи к исходному коду, для совместимости. Полную реализацию можно посмотреть на GitHub[1]. В результате проект с OpenSSL собирается без ошибок и запускается на Qemu.

[1]Репозиторий Embox https://github.com/embox/embox, форк репозитория https://github.com/DmitryPilyuk/embox/tree/openssl

4. Апробация

4.1. Условия

В проверки воспроизведём 3 сценария использования OpenSSL:

- функция RAND_bytes.;
- алгоритм RSA;
- алгоритм шифрования AES.

Реализуем для каждого случая консольную команду, которая будет вызывать необходимую функцию. Создадим под эту задачу проект openssl3 в директории project.

4.2. RAND_bytes

Для проверки используем следующий код

Листинг 6: openssl_rand_demo.c

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <openssl/rand.h>

int main(int argc, char **argv) {
    int outf;
    unsigned char buf[5] = { 'a', 'b', 'c', 'd', 'e' };

    if (RAND_bytes(buf, sizeof(buf))) { /* 1 success, 0 otherwise */
        outf = open("/mnt/rnd_bytes", O_CREAT|O_TRUNC|O_RDWR, 0600);
        if (outf < 0) {
            perror("open");
            return 1;
        }
```

```
        if (write(outf, buf, sizeof(buf)) < 0) {
            perror("write");
            close(outf);
            return 1;
        }
        close(outf);
    } else {
        printf("-ERR: RAND_bytes\n");
    }
    return 0;
}
```

Вынесем это в отдельный модуль и создадим Mybuild файл. В нем укажем аннотации @AutoCmd и @Cmd, также добавим зависимость depends third_party.lib.openssl3.

Листинг 7: Mybuild файл для openssl_rand_demo

```
package embox.project.openssl3.cmd

@AutoCmd
@Cmd(name = "openssl_rand_demo", help="Openssl rand demo")
@BuildDepends(third_party.lib.openssl3)
@Build(stage=2)
module openssl_rand_demo {
    source "openssl_rand_demo.c"
    depends third_party.lib.openssl3
}
```

4.3. RSA и AES

В качестве теста алгоритма RSA и AES возьмем демо, расположенные в исхидниках OpenSSL в директроии demos. Для каждого случая добавим Mybuild файл.

Листинг 8: Mybuild файл для rsa_demo

```
package embox.project.openssl3.cmd

@AutoCmd
@Cmd(name = "rsa_demo", help="Openssl rsa_demo")
@BuildDepends(third_party.lib.openssl3)
@Build(stage=2)
module rsa_demo {
    source "rsa_encrypt.c"
    depends third_party.lib.openssl3
}
```

Листинг 9: MYBUILD файл для aesgcm_demo

```
package embox.project.openssl3.cmd

@AutoCmd
@Cmd(name = "aesgcm_demo", help="aesgcm demo")
@BuildDepends(third_party.lib.openssl3)
@Build(stage=2)
module aesgcm_demo {
    source "aesgcm.c"
    depends third_party.lib.openssl3
}
```

4.4. Сборка и запуск

Скопируем в данную директорию стандартный темплейт под архитектуру x86 и для запуска на QEMU и подключим в mods.conf созданые ранее демо, добавив следующие строки.

Листинг 10: mods.conf. Подключение команд.

```
        include embox.project.openssl3.cmd.openssl_rand_demo
        include embox.project.openssl3.cmd.rsa_demo
        include embox.project.openssl3.cmd.aesgcm_demo
}
```

Так же отредактируем файл system_start.inc, чтобы реализованные демо запускались при старте.

Листинг 11: system_start.inc.

```
"export PWD=/",
"export HOME=/",
"partition",
"service net_service",
"service telnetd",
"service httpd",
"source time_setup.sh",
"mkdir -v /mnt",
"mount -t ramfs /dev/static_ramdisk /mnt",
"openssl_rand_demo",
"rsa_demo",
"aesgcm_demo",
"tish",
}
```

Код демо расположен на GitHub[2]. Теперь соберем проект, для этого вызываем:

make confload-project/openssl3/x86_qemu

make

И для запуска:

./scripts/qemu/auto_qemu

Результат запуска демо сценариев представлен на рисунке 1. Шифрование и дешифрование корректно завершилось в обоих случаях, а RAND_bytes демо сгенерировало и записало свой результат в файл по пути /mnt/rnd_bytes.

[2] https://github.com/embox/embox/tree/master/project/openssl3

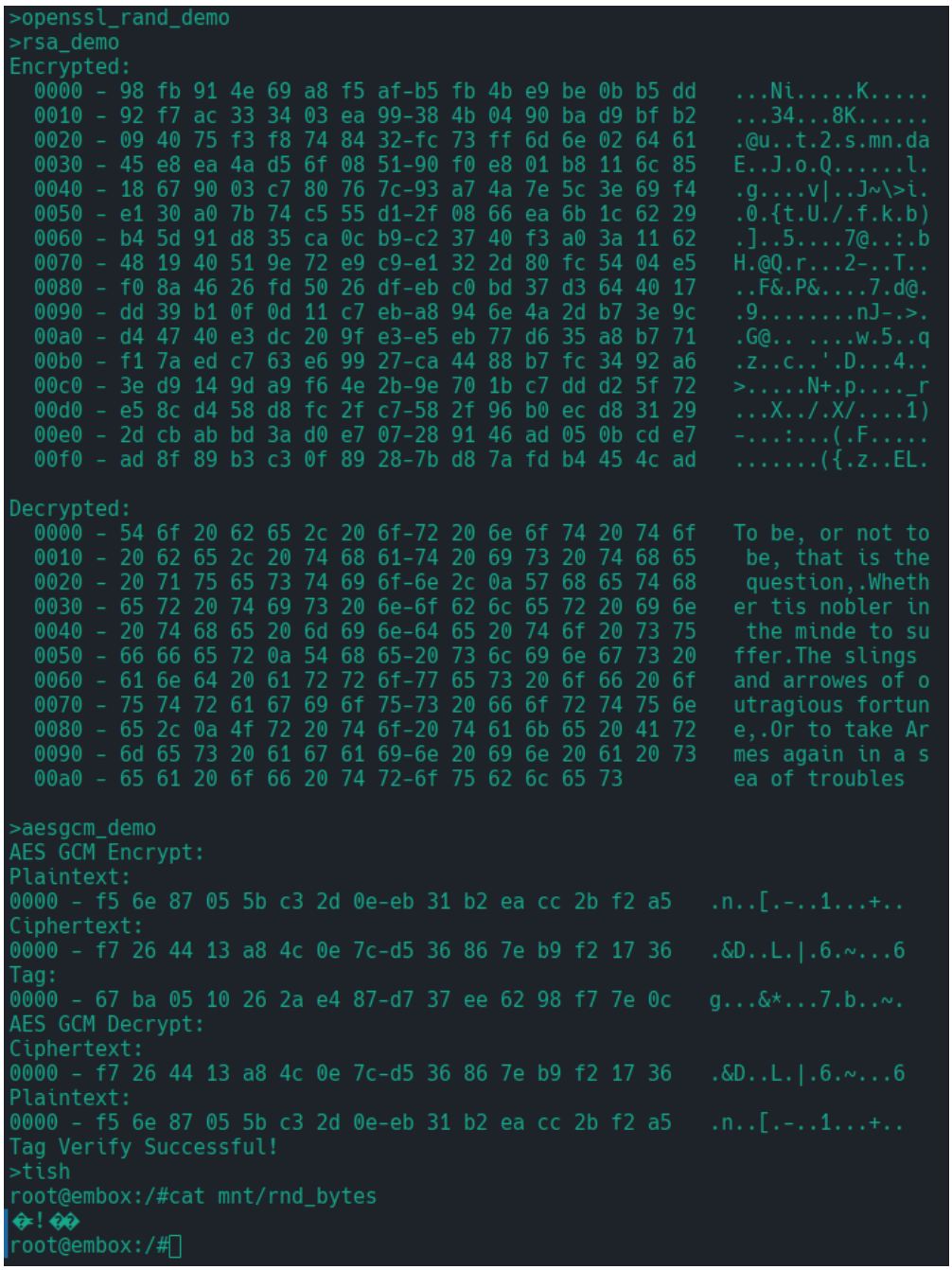

Рис. 1: Результат запуска ОСРВ Embox с демо.

Заключение

Данная работа направлена на расширение криптографических возможностей ОСРВ Embox путем портирования OpenSSL.

Для этого было сделано:

1. Рассмотрены криптографические алгоритмы уже реализованные в проекте.
2. Портирована библиотека OpenSSL.
3. Протестирована работоспособность библиотеки.

Список литературы

[1] Embox. Wiki. — URL: https://github.com/embox/embox/wiki (дата обращения: 30 августа 2024 г.).

[2] Mbed-TLS. Documentation. — URL: https://mbed-tls.readthedocs.io/en/latest/ (дата обращения: 25 августа 2024 г.).

[3] OpenSSL. OpenSSL. — URL: https://www.openssl.org/ (дата обращения: 4 сентября 2024 г.).

[4] OpenSSL. OpenSSL GitHub. — URL: https://github.com/openssl/openssl (дата обращения: 14 сентября 2024 г.).

[5] Tanenbaum A. Bos H. Modern Operating Systems. — Boston, MA, USA : Pearson, 2015. — ISBN: 9780133591620.