

Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Кафедра компьютерных технологий

Волков Иоанн Русланович
Еремеев Андрей Игорьевич

Система строгой аутентификации для ОС МСВС 3.0

Курсовая работа

Санкт-Петербург
2014

ОГЛАВЛЕНИЕ

Оглавление	2
Глава 1. Постановка задачи	4
1.1 Формулировка	4
1.2 Реализуемость и наличие эффективных алгоритмов .	4
Глава 2. Функциональная архитектура	6
2.1 To be erased	6
2.2 Функции РАМ модуля	6
2.2.1 pam_sm_authenticate	6
2.2.2 pam_sm_setcred	6
2.3 Функции шифрования/дешифрования	7
Глава 3. Выбор критериев оптимальности	8
Глава 4. Оценка проектных решений	9
Глава 5. Обоснование выбора метода решения задачи	10
Глава 6. Алгоритм функционирования	11

Глава 7. Технология разработки программы	12
Глава 8. Порядок использования	13
8.1 Руководство для администратора	13
8.1.1 Подготовка модуля	13
8.1.2 Подготовка файла /etc/u2sn	13
8.1.3 Генерация ключей и паролей пользователей . . .	13
8.1.4 Подключение модуля к ОС	14
8.2 Руководство для пользователя	14
Глава 9. Методика тестирования	15
9.1 Самопроверка генератора ключей	15
9.2 Тестирование аутентификации вручную	15
Глава 10. Описание программы	17
10.1 Сборка	17
10.2 Поддержка разных ОС	17
Источники	19

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ

1.1. Формулировка

Требуется разработать средство обеспечения двухфакторной аутентификации (в формате РАМ модуля) пользователя в операционной системе MCBC 3.0 на основе предъявления им:

- внешнего USB устройства с заданным серийным номером;
- пароля;
- ключа пользователя (256 бит), зашифрованного на пароле, хранящегося на USB устройстве.

Алгоритм аутентификации должен включать следующую последовательность действий:

- USB устройство присоединяется к ПЭВМ;
- пользователь вводит локальное имя;
- проверяется соответствие имени серийному номеру устройства;
- пользователь вводит пароль;
- зашифрованный ключ пользователя считывается из USB устройства;
- с использованием пароля производится расшифровка ключа.

Далее использовать стандартную процедуру аутентификации, где в качестве пароля используется ключ. При неправильно введенном пароле запросить его повторный ввод. Если число ошибок превышает заданный порог, учетная запись должна заблокироваться.

1.2. Реализуемость и наличие эффективных алгоритмов

Для реализации РАМ модуля, описанного выше, необходимы следующие нетривиальные возможности:

1. создания PAM модуля аутентификации;
2. получение от PAM имени пользователя;
3. получение от PAM пароля к ключу;
4. определение наличия внешнего USB устройства с заданным серийным номером;
5. считывание зашифрованного ключа с USB устройства с заданным серийным номером;
6. расшифровка ключа паролем;
7. передача расшифрованного ключа в качестве пароля PAM;
8. подключение PAM модуля к системе.

Для выполнения пункта 1 требуется создать динамическую библиотеку, реализующую функции `pam_sm_authenticate` и `pam_sm_setcred`.

Пункты 2, 3 и 7 выполнимы с помощью таких функций PAM как `pam_get_user`, `pam_get_item`, `pam_set_item` и нескольких других.

Пункты 4 и 5 потребовали тщательного изучения того, как реализовано взаимодействие с USB устройствами в операционной системе МСВС 3.0, и описаны далее [привести ref?].

Пункт 6 решается использованием библиотеки *openssl* [1]. Был выбран алгоритм *AES*. Кроме того, необходимо создать утилиту, которая будет генерировать для пользователя ключ и шифровать его паролем.

Для выполнения пункта 8 требуется правильно прописать созданный PAM модуль в специальном конфигурационном файле и добавить модулю аутентификации *pam_unix* параметр `try_first_pass`. Подробности в разделе 8.1.

ГЛАВА 2. ФУНКЦИОНАЛЬНАЯ АРХИТЕКТУРА

2.1. To be erased

В разделе перечисляются основные функции, подлежащие реализации, приводится описание входных и выходных параметров, их локализации в системной архитектуре программы. Изложение данного раздела должно опираться на функциональность пользовательских и программных интерфейсов (при наличии).

2.2. Функции PAM модуля

Как уже упоминалось в разделе 1.2, требуется реализовать функции `pam_sm_authenticate` и `pam_sm_setcred`.

2.2.1. `pam_sm_authenticate`

Функция аутентификации пользователя. Объявление этой функции выглядит следующим образом:

```
PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags,  
                                   int argc, const char **argv);
```

Описание параметров функции:

- `pamh` — хэндл PAM;
- `flags` — флаги, переданные модулю;
- `argc`, `argv` — параметры, переданные модулю.

Возвращаемое значение отвечает за успешность аутентификации (это может быть тип ошибки).

Внутри этой функции реализован алгоритм аутентификации, описанный в разделе 1.1.

2.2.2. `pam_sm_setcred`

Функция, настраивающая права пользователя после аутентификации. Объявление этой функции выглядит следующим обра-

ЗОМ:

```
PAM_EXTERN int pam_sm_setcred(pam_handle_t *pamh, int flags,  
                             int argc, const char **argv);
```

Внутри этой функции написана заглушка, всегда возвращающая успешный результат, так как настройкой прав пользователя будут заниматься другие модули.

2.3. Функции шифрования/дешифрования

Было решено сделать удобные обертки над функциями библиотеки *openssl*.

Тут должен писать Андрей.

ГЛАВА 3. ВЫБОР КРИТЕРИЕВ ОПТИМАЛЬНОСТИ

В разделе подлежат обоснованному выбору требования, в соответствии с которым анализируются качества тех ли иных алгоритмов и решений. К числу критериев относятся: производительность, уровень защищенности от НСД, сложность разработки, переносимость (открытость) на другие системные платформы, возможность совмещения функций компонента с реализацией других, схожих по назначению функций.

ГЛАВА 4. ОЦЕНКА ПРОЕКТНЫХ РЕШЕНИЙ

В разделе должны быть перечислены методы решения задачи, различающиеся выбранными критериям (например, при реализации шифрования телекоммуникационных потоков соответствующий компонент может быть реализован в виде драйвера или DLL библиотеки).

ГЛАВА 5. ОБОСНОВАНИЕ ВЫБОРА МЕТОДА РЕШЕНИЯ ЗАДАЧИ

В разделе приводятся аргументы по выбору одного из перечисленных решений. Для обоснования могут использоваться сравнение с аналогом (при наличии).

ГЛАВА 6. АЛГОРИТМ ФУНКЦИОНИРОВАНИЯ

В разделе приводится укрупненная схема алгоритма, позволяющего понять работу программы.

ГЛАВА 7. ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММЫ

В разделе должны быть перечислены программные интерфейсы, переменные среды, требования к библиотекам и.т.д., а также приведен общий порядок разработки программного компонента.

ГЛАВА 8. ПОРЯДОК ИСПОЛЬЗОВАНИЯ

8.1. Руководство для администратора

В данном разделе приведены действия, которые необходимо произвести системному администратору для подключения и настройки разработанного РАМ модуля.

8.1.1. Подготовка модуля

Прежде всего требуется собрать бинарные файлы. Необходимые для сборки шаги описаны в разделе 10.1.

8.1.2. Подготовка файла `/etc/u2sn`

Требуется создать файл `/etc/u2sn` и заполнить его строками следующего вида:

<code><имя_пользователя> <серийный_номер_USB_устройства></code>

Пользователи, которые не будут указаны в этом файле, не смогут пройти аутентификацию после подключения модуля *ram_twofactor_auth* к системе!

8.1.3. Генерация ключей и паролей пользователей

Для создания ключа пользователя требуется запустить программу *aes_generate* и ввести пароль, которым будет зашифрован ключ. Программа выдаст 2 строки:

- расшифрованный ключ, который нужно установить в качестве пароля пользователя;
- зашифрованный ключ (с дополнительными данными для проверки корректности пароля к ключу при расшифровке), который нужно положить в файл с именем `ptfa.key` на USB устройство.

Если на USB устройстве несколько разделов, то можно положить файл `ptfa.key` на любой из них. Но при этом требуется проверить, что на других разделах нет такого файла с отличающимся содержимым, так как модуль будет использовать первый обнаруженный файл с именем `ptfa.key`. Порядок обнаружения зависит от того, какие разделы в данный момент примонтированы, поэтому неизвестен заранее.

8.1.4. Подключение модуля к ОС

Для подключения модуля к операционной системе требуется отредактировать следующий конфигурационный файл:

- в ОС MCBC 3.0: `/etc/pam.d/system-auth`;
- в Ubuntu 12.04 и новее: `/etc/pam.d/common-auth`.

Перед строкой, которая начинается с `auth` и содержит `pam_unix.so`, нужно написать строку:

```
auth    requisite    </полный/путь/к/файлу/pam_twofactor_auth.so>
```

В конце строки, которая начинается с `auth` и содержит `pam_unix.so`, требуется добавить параметр `try_first_pass` (если его в ней нет).

Пример того, как в итоге могут выглядеть добавленная и отредактированная строки вместе:

```
auth    requisite    /root/twofactor_auth/build/pam_twofactor_auth.so
auth    sufficient   pam_unix.so likeauth nullok try_first_pass
```

После этого действия PAM модуль будет использоваться системой для любой аутентификации.

8.2. Руководство для пользователя

Пользователь должен следовать алгоритму, описанному в формулировке задачи (см. раздел 1.1).

ГЛАВА 9. МЕТОДИКА ТЕСТИРОВАНИЯ

В этой главе описаны методики, которые были использованы для тестирования разработанного программного компонента.

9.1. Самопроверка генератора ключей

Утилита *aes_generate* после генерации ключа и шифрования его паролем запускает алгоритм расшифровки, реализованный в модуле аутентификации, и проверяет, что получился исходный ключ.

9.2. Тестирование аутентификации вручную

Тестирование вручную подразумевает настройку модуля аутентификации и подключение его к операционной системе (см. раздел 8.1), а затем проверку следующих случаев:

- корректная аутентификация (в соответствии с алгоритмом, описанным в разделе 1.1);
- отсутствие файла */etc/u2sn* (в этом случае аутентификация должна блокироваться);
- USB устройство с верным ключом и неверным серийным номером;
- USB устройство с верным серийным номером и неверным ключом;
- большое число USB устройств и среди них одно с верным серийным номером и верным ключом (т.е. корректная аутентификация при наличии большого числа лишних USB устройств);
- неправильный пароль к ключу;
- многократный ввод неправильных паролей к ключу;

- имя пользователя, отсутствующее в файле `/etc/passwd`;
- имя пользователя, не существующего в операционной системе.

Каждый случай был многократно проверен после разработки программного продукта. Не все результаты совпали с ожидаемыми. Ошибки были найдены и устранены. После этого многократное тестирование всех случаев было произведено повторно. Все результаты совпали с ожидаемыми.

ГЛАВА 10. ОПИСАНИЕ ПРОГРАММЫ

Результатами разработки являются РАМ модуль *pam_twofactor_auth* и программа *aes_generate*.

10.1. Сборка

Для сборки разработанного программного продукта необходимо выполнить следующие шаги:

1. клонировать репозиторий [2];
2. создать директорию для бинарных файлов (далее предполагается, что эта директория называется *build* и находится в корне репозитория);
3. запустить команду *cmake ..*;
4. прописать нужный тип сборки (например, *Debug* или *Release*), сконфигурировать *CMake* и сгенерировать им *Makefile*;
5. запустить команду *make*.

Если сборка выполнится успешно, то в директории *build* будут бинарные файлы *pam_twofactor_auth.so* и *aes_generate*.

10.2. Поддержка разных ОС

Разработанный продукт поддерживает 2 операционных системы:

- MSVC 3.0;
- Ubuntu 12.04 и новее.

Следует обратить внимание на то, что в операционной системе Ubuntu изначально нет утилиты *cmake*, поэтому для сборки проекта необходимо предварительно выполнить команду *sudo apt-get install cmake-curses-gui*.

Кроме того, для клонирования репозитория необходимо установить *git*. Альтернативный вариант: вместо первого шага сборки скачать архив с репозиторием и распаковать его.

ИСТОЧНИКИ

- [1] Библиотека криптографии *OpenSSL*.
<http://www.openssl.org/>.
- [2] Репозиторий РАМ модуля *pam_twofactor_auth*.
https://github.com/Tsar/twofactor_auth.