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

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ

1.1. Формулировка

Требуется разработать средство обеспечения двухфакторной аутентификации (в формате PAM модуля) пользователя в операционной системе МСВС 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 PAM 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 6.1.

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*. Были реализованы следующие функции:

1. `md5_hash` — функция, вычисляющая *md5*-хер строки;
2. `aes_encode` — функция, шифрующая ключ с помощью пароля алгоритмом шифрования AES;
3. `aes_decode` — функция, расшифровывающая ключ с помощью пароля алгоритмом шифрования AES;
4. `base64_encode` — функция, переводящая массив байт в base64-строку;
5. `base64_decode` — функция, переводяшая base64-строку в массив байт.

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 приводится аргументы по выбору одного их перечисленных решений. Для обоснования могут использоваться сравнение с аналогом (при наличии).

ГЛАВА 6. АЛГОРИТМ ФУНКЦИОНИРОВАНИЯ

В этой главе подробно описаны алгоритмы работы PAM модуля *pam_twofactor_auth* и программы *aes_generate*.

6.1. Алгоритм работы PAM модуля *pam_twofactor_auth*

Представленный алгоритм является реализацией функции `pam_sm_authenticate` (см. раздел 2.2.1).

Ниже приведены шаги алгоритма:

1. Получить имя пользователя функцией `pam_get_user`.
2. Узнать, используя файл `/etc/u2sn`, какому серийному номеру USB устройства соответствует имя пользователя. Если соответствие для пользователя не задано или файл `/etc/u2sn` не существует, выдать об этом сообщение и перейти к шагу 1.
3. По серийному номеру USB устройства, вычислить какому блочному устройству оно соответствует (например, это может быть `/dev/sdc`). Если USB устройство с требуемым серийным номером не обнаружено, выдать об этом сообщение и перейти к шагу 1.
4. Проверить, есть ли у полученного блочного устройства разделы (например, `/dev/sdc1`, `/dev/sdc2`). Если нет, то считать разделом `/dev/sdc`.
5. Проверить в файле `/etc/mtab` примонтированы ли какие-нибудь разделы USB устройства. Если да, то на тех, которые примонтированы проверить наличие файла `ptfa.key` в корне. Если этот файл будет обнаружен, прервать проверку и перейти к шагу 7. Иначе, перейти к шагу 6.
6. По очереди монтировать непримонтированные разделы

USB устройства, проверять наличие файла `ptfa.key` и отмонтировать.

7. Если на шаге 5 или на шаге 6 был обнаружен файл `ptfa.key`, прочитать его содержимое. Иначе сообщить об ошибке аутентификации.
8. Запросить пароль (средствами PAM). Если введенным паролем не удастся расшифровать содержимое ключа, повторить шаг 8. Максимальное число повторений ограничено константой. После того, как попытки будут потрачены, заблокировать аутентификацию и завершить выполнение алгоритма. Если же содержимое ключа удастся расшифровать паролем, то перейти к следующему шагу.
9. Задать расшифрованный паролем ключ в качестве `PAM_AUTHTOK` (таким образом, следующий модуль аутентификации сможет использовать его в качестве пароля). Разрешить аутентификацию.

6.1.1. Детали реализации шага 3

Реализация данного шага для операционных систем МСВС 3.0 и Ubuntu различна.

6.1.1.1. Детали реализации шага 3: МСВС 3.0

Реализация шага 3 алгоритма работы модуля *pam_twofactor_auth* представлена отдельной функцией `getDeviceBySerialNumber_MSVS30`. На вход она принимает серийный номер устройства, а на выход выдаёт либо ошибку «устройство не найдено», либо блочное устройство, которому соответствует USB устройство с требуемым серийным номером.

Алгоритм работы этой функции следующий:

1. Прочитать содержимое файла /proc/bus/usb/devices. Если в нем отсутствует строка, имеющая вид S: serialnumber=<серийный_номер>, выдать ошибку «устройство не найдено» и завершить выполнение функции.
2. Перебирать все файлы, соответствующие маске /proc/scsi/usb-storage-*/*, и искать внутри них строку, имеющую вид serial number: <серийный_номер>. Как только файл с такой строкой будет обнаружен, прочитать в нем значение параметра Host scsi (оно может быть, например, scsi1) и перейти к следующему шагу.
3. Читать файл /var/log/messages с конца построчно (для этого была разработана специальная функция getLineFromBack) до тех пор, пока не будет найдена строка, соответствующая маске:

```
Attached scsi removable disk * at scsi <значение_параметра_из_шага_2>
```

4. Вернуть в качестве блочного устройства конкатенацию /dev/ и того значения, которое будет соответствовать звездочке в приведенной выше маске.

6.1.1.2. Детали реализации шага 3: Ubuntu

Реализация шага 3 алгоритма работы модуля *pam_twofactor_auth* представлена отдельной функцией getDeviceBySerialNumber. На вход она принимает серийный номер устройства, а на выход выдаёт либо ошибку «устройство не найдено», либо блочное устройство, которому соответствует USB устройство с требуемым серийным номером.

Алгоритм работы функции getDeviceBySerialNumber:

1. Перебирать все файлы, содержащиеся в папке /dev/disk/by-id, и искать файл с названием

`usb-<serial number>`.

2. Если такой файл не был найден, то выдать ошибку «устройство не найдено» и завершить выполнение функции.
3. Если такой файл был найден, то вернуть в качестве блочного устройства `/dev/`, результат выполнения функции `readlink`.

6.2. Алгоритм работы программы *aes_generate*

Приложение *aes_generate* принимает на вход пароль(*passphrase*) и генерирует на выходе строку вида *md5(<passphrase>+<key>)+aes(<key>)* в *base64*-формате, где «+» — конкатенация строк, *md5* – *md5*-хеш от строки, *aes* — зашифрованный алгоритмом *AES* с помощью введенного пароля случайно сгенерированный ключ. Полученная строка сохраняется на USB устройство. Алгоритм аутентификации:

1. Строка, сохраненная на USB, разбивается на две: *md5*-хеш и зашифрованный ключ.
2. Пользователь вводит пароль.
3. Ключ расшифровывается с помощью пароля.
4. Вычислить *md5*-хеш от конкатенации пароля и расшифрованного ключа. Если хеш совпадает с ожидаемым, то аутентификация успешна, иначе не успешна.

В приложении были использованы алгоритмы шифрования библиотеки *openssl*.

6.3. To be erased

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 PAM 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:

```
<имя_пользователя> <серийный_номер_USB_устройства>
```

Пользователи, которые не будут указаны в этом файле, **не смогут** пройти аутентификацию после подключения модуля *pam_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 МСВС 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/u2sn`;
- 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 PAM 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 `ccmake ..`;
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:

- МСВС 3.0;
- Ubuntu 12.04 и новее.

Следует обратить внимание на то, что в операционной системе Ubuntu изначально нет утилиты *ccmake*, 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 PAM модуля *pam_twofactor_auth*.
https://github.com/Tsar/twofactor_auth.