SASL – модуль сетевой аутентификации пользователей.

mkondrin из hppi.troitsk.ru

Аннотация

С помощью библиотеки SASL2 сетевые приложения могут автоматически договорить ся об использовании определенного механизма подтверждения идентичности пользователя. В статье описана настройка библиотеки SASL для работы совместно с Kerberos.

Общие свойства SASL.

Преимущества протокола Kerberos – сильная криптографическая защита паролей и данных, единая система управления паролями пользователей и возможность реализации single sign-on системы (система единой регистрации пользователей). Основной недостаток – сложная программная реализация механизма аутентификации. Для того чтобы протокол, точнее сервис и клиентское приложение, использовал аутентификацию пользователей через Kerberos соответствующий код должен быть реализован как в клиентской, так и серверной частях протокола. Поэтому от системного администратора, пытающегося организовать single sign-on систему требуются определенные усилия по анализу используемых программных пакетов и выбору тех из них, что поддерживают Kerberos.

Любой системный администратор в своей работе наверняка сталкивается с протоколами отправки и получения электронной почты. Электронная почта – старый и популярный протокол. И именно потому, что протокол старый (созданный в те времена, когда разработчики не слишком задумывались о безопасности своих продуктов) и популярный (широкий выбор потенциальных жертв) электронная почта является удобной целью для злоумышленников. При этом задача системного администратора состоит в организации безопасности нескольких сетевых протоколов – smtp, imap и/или pop3, их конфиденциальности и защиты от несанкционированного доступа. В принципе, способов добиться решения этой задачи может быть несколько (см., например, [1, 2, 3, 4]), но решение с помощью Kerberos представляется мне

наиболее простым и удобным. Удобство в данном случае подразумевает не только простоту администрирования, но и удобство пользователей, которые получают единообразный способ доступа к протоколам чтения и отправки почты на основании своего системного пароля как изнутри так и извне локальной сети (в последнем случае есть некоторые оговорки). Немаловажнымобстоятельством является и то, что Kerberos хорошо масштабируемый протокол, который разрабатывался в расчете на огромные университетские сети.

Ключевым элементом обеспечивающим взаимодействие Kerberos и почтовых службы является пакет SASL (Simple Secure Authentication Layer), разработанный в Университете Карнеги-Меллон. Помимо реализации от CMU (получившей название Cyrus-SASL) существует GNU-версия gsasl [5], разрабатываемая практически единолично шведским программистом Джеффсоном. Несмотря на то, что gsasl довольно быстро развивается, эта реализация все еще считается бета-версией и значительно уступает по надежности Cyrus-SASL. Всего университетом Карнеги-Меллон было выпущено две реализации библиотеки SASL, различающиеся между собой по API и формату хранения пользовательских данных, хотя и совместимые по протоколу. В настоящее время вторая версия SASL практически полностью вытеснила первую, и поэтому далее под SASL я буду понимать именно вторую версию (которую иногда обозначают как SASL2).

Хотя в этой статье меня будет интересовать настройка SASL применительно к её использованию в системе электронной почты, но, вообще говоря, функции SASL гораздо шире, чем просто интерфейс между почтовыми службами и Kerberos. SASL был задуман в качестве своего рода диспетчера, позволяющего сетевым клиентам и серверам договорится о механизме аутентификации, который клиент должен использовать, чтобы подключиться к серверу. Более того, библиотека, входящая в состав SASL, позволяет унифицированным образом провести эту аутентификацию независимо от выбранного механизма. Прозрачность процедуры аутентификации через SASL позволяет как клиенту, так и серверу одновременно поддерживать несколько различных типов аутентификации без необходимости программно реализовывать каждый из них - разработчик может положиться на библиотеку SASL, что та выберет наиболее подходящий из набора, предлагаемого клиентом или сервером.

Как это выглядит на практике? Клиент и сервер могут находится в разных сетях, которые построены и сконфигурированы разными администраторами, каждый из которых имеет свой взгляд на политику сетевой безопасности.Предположим, что один из них сконфигурировал сервис таким образом, что тот позволяет аутентифицировать пользователя через Kerberos5 (условное обозначение механизма – GSSAPI), с помощью пароля, закодированного пол алгоритму MD5 (CRAM-MD5) и простого текстового пароля (PLAIN). Если в клиенте настроена поддержка какого-

нибудь из этих механизмов, то SASL позволяет провести аутентификацию посредством именно этого механизма, причем в случае множественного пересечения будет выбран наиболее сильный. Программная реализация выглядит одинаково - разработчик должен инициализировать библиотеку (с помощью функций sasl_client/server_new, ..._init, ..._start) и затем войти в цикл аутентификации. На каждом шаге цикла программа должна прочитать блок данных полученный по сети (реплика клиента/сервера) и передать его функции sasl_client/server_step. На выходе функции контролируется возвращаемое значение и один из ссылочных параметров. Ссылочный параметров передается по сети без дополнительной обработки, а возвращаемое значение анализируется и в зависимости от его значение принимается решение о продолжении или остановки цикла аутентификации. Цикл прекращается когда возвращаемое значение станет равным SASL_OK, что означает успешную аутентификацию, или функция вернет сообщение об ошибке, что означает отказ в аутентификации. Хотя число шагов, которое требуется для подтверждения идентичности пользователя может быть различно для разных механизмов аутентификации, да и сам способ ввода пароля (или его отсутствие) в клиентском приложении может быть различным, но сам алгоритм аутентификации при этом остается тем же самым.

Теперь небольшой обзор возможностей SASL. Механизмы аутентификации, реализованные в Cyrus-SASL, условно можно разделить на три группы (см. рисунок). Первая группа, наиболее слабые механизмы – этот аутентификация посредством открытых текстовых паролей, PLAIN и LOGIN. Различие между ними только в формате, в котором имя и пароль пользователя передаются по сети от клиента к серверу. Это очень слабые механизмы (слабее них только ANONYMOUS открывающий беспарольный доступ любому пользователю), поскольку пароли могут быть перехвачены злоумышленником в момент передачи по сети. Если же системный администратор решит использовать эти механизмы, то ему нужно озаботиться поиском внешнего способа криптования канала передачи, например, с помощью SSL/TLS.

Вторая группа механизмов – это механизмы с общим ключом, CRAM-MD5, DIGEST-MD5 и несколько особняком стоящим от них механизмом одноразовых паролей – OTP. Слабость механизмов CRAM-MD5 и DIGEST-MD5 состоит в том, что хотя пароли и передаются по сети в зашифрованном виде, но они могут быть перехвачены и взломаны offline с помощью простого перебора. С другой стороны, при использовании этих механизмов требуется наличие незашифрованных паролей на почтовом сервере, где их безопасность бывает сложно гарантировать. Метод одноразовых паролей в этом смысле более безопасен. Идея этого метода состоит в создании последовательности легко запоминающихся фраз, сгенерированных из одного секретного пароля. Этот список хранится на сервере и время от времени пополняется. Для подключении к серверу пользователь использует очередную фразу

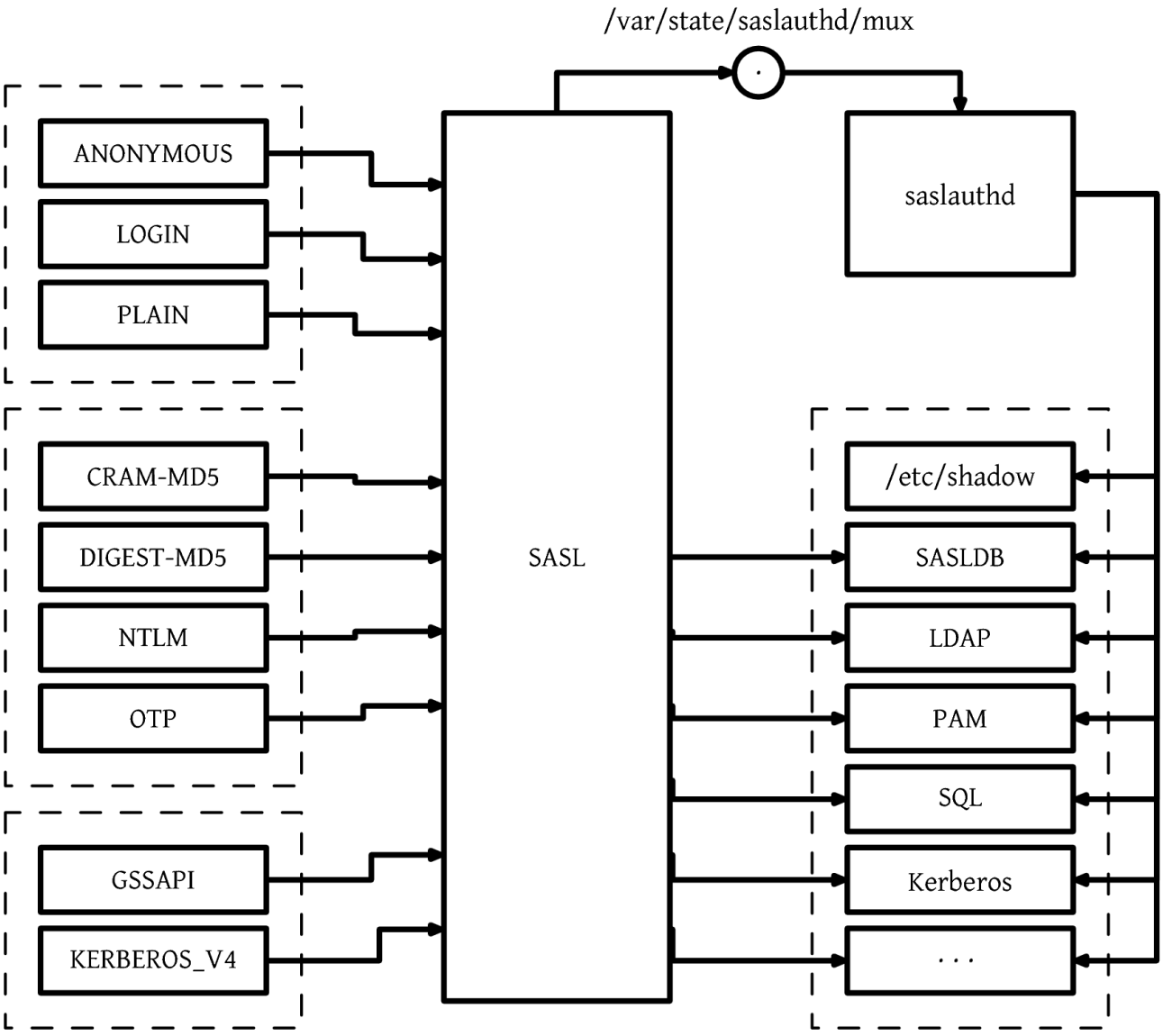

Рис. 1: Структура SASL.

из сгенерированного списка как свой пароль, после чего эта фраза из списка вычеркивается и для следующего подключения должна быть использована следующая в списке. Понятно, что перехват пароля ничего не дает злоумышленнику, хотя попрежнему остается опасность, что злоумышленник сумеет завладеть всем списком одноразовых паролей, хранящимся на сервере.

И, наконец, последняя группа – механизмы использующие Kerberos (GSSAPI и KEBEROS_V4). KEBEROS_V4 применяется для аутентификации по старому, не очень надежному, но тем не менее все еще используемому протоколу Kerberos4, а GSSAPI следует понимать как синоним Kerberos5, более современной версии Kerberos. В данной статье нас будет интересовать в основном этот последний механизм.

Как уже говорилось ранее, выбор механизма определяется процедурой “автоподстройки”, происходящей по сети между клиентской и серверной программой. Помимо разных механизмов SASL предлагает различные способы хранения паролей, которые в терминологии SASL называются методами. Системный администратор должен решить, какой из методов он будет использовать, причем можно подбирать метод индивидуально для каждого приложения. Возможные варианты: собственная база данных SASL (sasldb2), база данных LDAP, внешняя SQL-совместимая база данных, системные файлы с идентификационной информацией о пользователях (`/etc/passwd`, `/etc/shadow`), использование инфраструктуры PAM или Kerberos. В последнем случае не нужно путать Kerberos, как метод, с механизмами GSSAPI и KERBEROS_V4. Использование этого метода позволяет просто использовать базу данных Kerberos как хранилище паролей, без необходимости развертывания всей инфраструктуры Kerberos. Выбор метода во многом определяет набор доступных механизмов – бесполезно объявлять механизм GSSAPI как поддерживаемый приложением, если единственный доступный для него метод – это системные файлы. Библиотека SASL может использовать методы либо напрямую, либо через своего “агента” – демона saslauthd, с которым библиотека SASL общается через именованный сокет. Использование агента бывает необходимо для согласования прав доступа – сетевое приложение может быть запущено от непривелегированного системного пользователя, а для аутентификации, к примеру, через системные файлы нужен доступ на чтение `/etc/shadow`, что при разумной установке возможно только для root. Однако, можно запустить сервис saslauthd с привилегированными правами, тем самым открывая ему доступ к `/etc/shadow` и при этом не компрометируя безопасность системы, поскольку непосредственный доступ к этому сервису из сети невозможен. Результат процедуры аутентификации saslauthd сообщает сервису через именованный сокет, права на доступ к которому могут быть более либеральны, чем к `/etc/shadow`.

Установка библиотеки SASL.

После этого небольшого обзора SASL можно приступить к установке и тестированию самой библиотеки Cyrus-SASL. Исходный код доступен на сайте CMU (или на его зеркалах) [6]. Процедура сборки программного пакета стандартна – с помощью скрипта configure проводим предварительную настройку программы и компилируем командами make, make install. Помимо обычных опций (например, `--prefix=/usr` означающий, что программа будет установлены в каталог /usr) configure имеет еще большое число параметров, которые позволяют гибко настраивать Cyrus-SASL. Разберем некоторые из этих параметров. В первую очередь нас интересуют `--with-gssapi` и `--with-gss_impl=heimdal`, которые включают поддержку механизма GSSAPI и указывают какая реализация Kerberos будет использована для этого. Также как и в статье [7] предполагается, что в системе установлены библиотеки и заголовочные файлы Heimdal Kerberos, хотя возможно применение и более распространенной версии Kerberos от MIT. Другие механизмы SASL можно также включить (или выключить) при конфигурации, например:

```
--enable-login \
--enable-ntpl \
--disable-otp \
--disable-anon \
```

LOGIN и NTPL используются многими старыми почтовыми программами от Microsoft и их поддержку бывает полезно включить в Cyrus-SASL. Механизм ANONYMOUS для почтовых программ скорее вреден, OTP - на усмотрение администратора. Если вы решите использовать OTP, следует иметь ввиду, что Cyrus-SASL может либо использовать свою собственную версию OTP, либо уже установленные библиотеки OPIE (One Time Passwords in Everything [8], соответствующий параметр компиляции `--with-opie`). Хотя heimdal имеет свобю встроенную реализацию OTP, но у меня не получилось заставить Cyrus-SASL использовать именно эту библиотеку.

Поддержка метода SASLDB (или auxprop) обязательна для Cyrus-SASL, хотя в нашем случае она и не нужна. Другие методы можно выключить с помощью параметров:

```
--enable-ldapdb=no
--enable-sql=no
```

Можно также отключить агента saslauthd (опция `--with-saslauthd=no`), но мы это делать не будем.

SASLDB по умолчанию требует в качестве DB движка Berkley DB [9]. С этим связана одна небольшая проблема. Также как и SASL многие программы такие

как heimdal, Postfix и Cyrus-IMAP используют Berkley DB для хранения своих собственных данных, например, псевдонимов пользователей в Postfix. Многие дистрибутивы Линукс включают в себя несколько версий Berkley DB (от 2-ой до 4-ой), причём эти версии несовместимы между собой и линковка какой-нибудь программы с разными версиями Berkley-DB ни к чему хорошему не приведёт. К примеру, если попытаться скомпилировать SASL с 4-ой версией, в то время как heimdal, используемый SASL, слинкован с 3-ей, то в результате библиотека SASL будет неработоспособна. Грубо говоря, динамический загрузчик просто не разберётся из какой-именно библиотеки вызывать функцию, если она присутствует в обеих библиотеках. При конфигурации по-умолчанию интересующих нас программ автоматически выбирается самая новая версия BerkleyDB, которую удалось найти в системе. В то же время в этом случае могут возникнуть проблемы совместимости Cyrus-SASL и почтового сервера Postfix, который не поддерживает 4 версию BDB. Поэтому следует ограничиться только третьей версией и на время сборки программ деинсталлировать BerkleyDB4 (если она присутствует у вас в системе) или по крайней мере удалить ее заголовочные файлы. У меня все упомянутые выше программы были скомпилированы с BerkleyDB-3.3.

Взаимодействие с Kerberos через GSSAPI.

После сборки и установки Cyrus-SASL, можно проверить его работоспособность и взаимодействие с Kerberos с помощью тестовых программ client и server из папки cyrus-sasl-*/sample. Если они не скомпилировались вместе с библиотекой, то можно компилировать их и позже, выполнив make в папке sample.

Разумеется для проверки взаимодействия с Kerberos у вас должен быть настроен сектор Kerberos. Как это сделать описано в статье [7]. Поскольку в данном случае я хочу только продемонстрировать лишь принципы работы пакета Cyrus-SASL с Kerberos, будет достаточно, если ваш сектор Kerberos будет состоять только из одного компьютера. Это также на начальном этапе значительно упростит отладку. Также как и в статье [7] предполагаем, что сектор Kerberos называется MYREALM.RU, а машина, на которой мы разворачиваем “песочницу”, является также контроллером Kerberos и называется kdc.myrealm.ru. Чтобы не подвергать систему риску в процессе экспериментов с Kerberos и почтой, желательно использовать фиктивный сетевой интерфейс:

```
/sbin/ifconfig dummy0 192.168.10.1 netmask 255.255.255.0 up
echo ‘‘192.168.10.1     kdc.myrealm.ru’’ >> /etc/hosts
hostname kdc.myrealm.ru
```

Будем также считать, что в Kerberos уже созданы принципалы, соответствующие

машине host/kdc.myreal.ru@MYREALM.RU и пользователю mike@MYREALM.RU, а ключ принципала host/kdc.myrealm.ru записан в keytab-файл (/etc/krb5.keytab).

После того как все эти условия выполнены можно начать тестирование SASL. Вначале, войдя как root на одном из терминалов, запускаем серверную часть:

```
$./server -p 3001 -s host -m GSSAPI
```

Опция -p задает порт который будет прослушивать сервер. Имя сервиса определяется с помощью опции -s host, причем необходимо указать только первую часть имени принципала, к которой затем автоматически добавляется имя компьютера и имя сектора Kerberos. Можно выбрать любого другого принципала отличного от host/kdc.myrealm.ru, лишь бы его ключ присутствовал в keytab-файле. Список доступных механизмов определяется опцией -m, здесь мы ограничиваемся только GSSAPI механизмом.

Затем на другом терминале запускаете клиентскую часть. В первую очередь получим супербилет (TGT) для рядового пользователя (kinit mike) и от имени этого же пользователя запускаем клиента:

```
./client -p 3001 -s host -m GSSAPI kdc.myrealm.ru
receiving capability list... recv: {6}
GSSAPI
GSSAPI
please enter an authorization id: mike
...
successful authentication
closing connection
```

Все ключи означают тоже самое, что и в случае сервера; последняя опция – это имя компьютера, к которому производится подключение. В процессе программа клиент потребует ввести авторизационный идентификатор, т.е. имя пользователя, который должен быть аутентифицирован. Ввести нужно имя того принципала, для которого получен супербилет. После непродолжительного обмена закодированными сообшениями (в листинге они опущены) в обеих терминалах должно появится сообшение об успешной аутентификации. Если же появилось сообщение об ошибке, следует проверить, записан ли ключ принципала-сервиса в keytab-файле и правильно ли происходит преобразование имени компьютера в его адрес и наоборот. Просмотр логов контроллера Kerberos (`/var/log/krb5kdc.log`) также поможет в поиске неисправностей.

Проверка простых текстовых паролей в Kerberos.

Теперь протестируем взаимодействие клиента и сервера при аутентификации с помощью простых текстовых паролей, но так, чтобы пароль пользователя всё равно проверялся по базам данных Kerberos. Этот механизм требует использования демона saslauthd, который должен попытаться получить супербилет от KDC, используя имя и пароль пользователя. Если супербилет получен, то идентичность пользователя подтверждена. Следует понимать, что это не лучший способ использования Kerberos. Помимо риска перехвата паролей (а в случае паролей Kerberos ставки очень высоки), сама процедура генерации TGT для каждого подключения к сетевым службам достаточно ресурсоемка. Kerberos и не рассчитывали на такой режим его работы, эта идея целиком на совести разработчиков SASL. Заметьте также, что saslauthd не имеет собственного принципала в Kerberos, а сам пытается выступать от лица пользователя.

Перед запуском saslauthd нужно создать его рабочий каталог и выставить права доступа к нему. Как уже говорилось, эти права достаточно мягкие.

```
chown -R root.kerberos /var/state/saslauthd/
chmod -R 755 /var/state/saslauthd/
```

Теперь в ещё одном терминале запустим saslauthd:

```
/usr/sbin/saslauthd -a kerberos5 -d
```

С помощью ключа -a выбирается метод аутентификации, а ключ -d нужен только для отладки, чтобы все сообщения от saslauthd выводились на терминал, а не в log-файл. Как минимум, результатом работы этой программы будет создание именованного сокета `/var/state/saslauthd/mux`.

Следующим шагом нужно объяснить библиотеке SASL, что проверка паролей происходит опосредованно, через взаимодействие с saslauthd, причем эту настройку можно выполнить индивидуально для каждой из сетевых служб. При инициализации серверной части библиотеки SASL (функция sasl_server_init) один из параметров это так называемое имя приложения. Оно может отличаться от настоящего имени исполняемого файла и используется библиотекой для поиска файла настроек в каталоге /usr/lib/sasl2. Тестовый сервер выступает под именем “sample”, так что нам понадобится файл `/usr/lib/sasl2/sample.conf`, такого вида:

```
pwcheck_method:saslauthd
mech_list:gssapi plain
```

Нас в данном случае интересуют только два параметра – pwcheck_method и mech_list, а полный список параметров можно посмотреть в документации cyrus-sasl (doc/options.html). В этом конфигурационном файле указано, что помимо механизма GSSAPI, разрешено использовать также незашифрованные пароли (PLAIN),

и их проверка будет осуществляться агентом saslauthd. В конечном итоге выбор механизма аутентификации всегда остается за клиентом, сервер может только ограничить этот выбор, сообщив клиенту список поддерживаемых механизмов (GSSAPI и PLAIN в данном случае). Понятно, что исключение из mech_list механизма plain равносильно запрету использования простых текстовых паролей для данного приложения. Хотя в этом и есть свой резон, но пока в наши пляны это не входит.

Теперь презапустим server. Команда практически та же самая, что и использованная ранее, только без ключа -m. В этом случае список механизмов будет взят из файла `/usr/lib/sasl2/sample.conf`. Но при новом запуске клиента мы теперь должны указать механизм PLAIN:

```
./client -p 3001 -s host -m PLAIN kdc.myrealm.ru
receiving capability list... recv: {6}
GSSAPI PLAIN
PLAIN
please enter an authorization id: mike
...
successful authentication
closing connection
```

В отличии от предыдущего случая вам требуется ввести не только свой идентификатор, но и пароль – разумеется тот же самый, что и для команды kinit. Если всё же сервер вас не аутентифицировал, следует посмотреть на терминал, где запущен saslauthd и проверить есть ли там сообщения об ошибках. Если же там не появилось ввобще никаких новых сообщений, то скорее всего сервер не может найти именованный сокет для соединения с saslauthd или права доступа к этому сокету слишком строгие.

Заключение.

На этом настройку SASL можно считать завершенной и можно приступать к подключению этой библиотеки к сетевым службам. Несмотря на простоту разработки и программирования SASL используется не слишком широко. Главным образом это связано с довольно поздними появлением спецификации SASL (вторая половина 90х), когда большинство популярных сетвых протоколов уже имело свой собственный механизм идентификации пользователей. Поэтому SASL используется либо в протоколах, где существовавшая на тот момент защита была признана недостаточной, либо в новых протоколах. К первой категории можно отнести протоколы электронной почты и LDAP. Поддержка SASL включена в такие популярные продукты

как Sendmail, Postfix, Cyrus-IMAP (настройка последних двух программ будет рассмотрена в следующей статье). Протокол мгновенного обмена сообщениями jabber [10] - этот пример нового протокола, где SASL используется как на стороне сервера, так и поддерживается некоторыми клиентами (например, tkabber [11] собранный с библиотекой tcl SASL [12]). Интерес представляет также использование SASL для аутентификации пользователей сервера баз данных MySQL. Хотя эта поддержка не включена в основную ветку, но в интеренете можно найти патчи для 4 и 5 версий MySQL [13, 14]. Таким образом, соответствующая настройка библиотеки SASL позволяет интегрировать эти продукты в систему единой регистрации пользователей Kerberos.

Список литературы

[1] А.Бешков. Почтовая система для среднего и малого офиса. *Системный Администратор*, 5, 2003.
http://www.samag.ru/cgi-bin/go.pl?q=articles;n=05.2003;a=01.

[2] А.Мозговой. Smtp auth in da postfix + ... *Системный Администратор*, 10, 2003.
http://www.samag.ru/cgi-bin/go.pl?q=articles;n=10.2003;a=07.

[3] Е.Литвиненко. Настраиваем основные компоненты почтового сервера. *Системный Администратор*, 9, 2005.
http://www.samag.ru/cgi-bin/go.pl?q=articles;n=09.2005;a=08.

[4] Д.Шергин. Установка imap4-сервера на базе cyrus-imapd+sendmail. *Системный Администратор*, 10, 2003.
http://www.samag.ru/cgi-bin/go.pl?q=articles;n=10.2003;a=10.

[5] Реализация SASL, распространяемая под лицензией GNU/GPL. http://www.gnu.org/software/gsasl/.

[6] Каталог Cyrus-SASL и Cyrus-IMAP .
ftp://ftp.andrew.cmu.edu/pub/cyrus-mail.

[7] М.Кондрин. Развертываем heimdal kerberos. *Системный Администратор*, 7, 2005.
http://www.samag.ru/cgi-bin/go.pl?q=articles;n=07.2005;a=06.

[8] One-Time-Passwords in Everywhere.
http://www.inner.net/opie.

[9] BerkleyDB, ранее принадлежавщая компании SleepyCat, в настоящее время куплена Oracle-ом. Исходные тексты по-прежнему доступны на сайте Oracle. http://www.oracle.com/database/berkeley-db/index.html.

[10] IM сервер совместимый с Cyrus-SASL. http://www.jabber.org/software/jabberd2x.shtml.

[11] Jabber клиент с поддержкой SASL. http://tkabber.jabber.ru/en/.

[12] Интерфейс к библиотеке Cyrus-SASL для языка tcl. http://beepcore-tcl.sourceforge.net/tclsasl.html.

[13] Оригинальный патч с поддержкой Cyrus-SASL для MySQL-4.1.x. http://rc.vintela.com/topics/labs/mysql/.

[14] Патч и сборки для последних версий MySQL можно найти на этом сайте в указанном каталоге. ../SHlackware/.