Новые возможности Heimdal - объединение открытых ключей и Kerberos.

mkondrin
из hppi.troitsk.ru

7 мая 2008 г.

Аннотация

Проект Heimdal-Kerberos, в прошлом году отметивший 10-летний юбилей выпуском «круглой» версии 1.0 и переездом на новый сайт, позволяет объединить возможности инфраструктуры публичных ключей (Public Key Infrastructure - PKI) и Kerberos в рамках единой регистрационной системы. Об этом, а также других новинках этого программного пакета, будет рассказано в этой статье.

1 Центральное правительство или групповая солидарность?

Прежде, чем заняться практической демонстрацией новых возможностей Heimdal-Kerberos, хотелось бы напомнить историю возникновения Keberos и систем PKI, в частности стандарта X.509, используемого в том числе и в таком популярном и известном в мире Linux продукте как OpenSSL. Эта история поможет понять достоинства и недостатки каждого из решений.

Протокол Kerberos и идея публичных ключей, представленная в протоколе X.509, появились примерно одновременно в середине 80-х годов, и оба пытались решить одну и ту же задачу обеспечения сетевой безопасности. Под сетевой безопасностью нужно понимать не только шифрование передачи данных и дистанционной проверки паролей, но и возможность как серверу так и клиенту удостовериться в идентичности друг друга. Оба проекта решали эту проблему примерно одинаковым способом - введением третьей trusted стороны, авторитету которой доверяли бы обе стороны сетевого взаимодействия.

На этом их сходство и заканчивались. Kerberos возник всё же несколько ранее и идеологически он был построен более просто. Использование алгоритма симметричного шифрования, когда третья сторона (KDC - контроллер сервера Kerberos) владеет паролями как клиента так и сервера, поэтому может обеспечить обе стороны общим ключом, пригодным как для идентификации друг друга, так и для шифрования пересылаемых данных, позволяет более экономно использовать вычислительные ресурсы. В то же время протокол работал в онлайновом режиме - выпуск сессионных ключей (билетиков, tickets) всегда выполняется сервером KDC и по специальному протоколу пересылаются клиенту. Не вдаваясь в подробности (которые, однако, можно найти в статье [1]), идентичность клиента и сервера удостоверяется тем, что им удалось расшифровать сессионный ключ, который получен от KDC, будучи зашифрованным паролями клиента(пользователя) и сервера. А успешность расшифровки сессионного ключа проверяется тем, что обеим сторонам удалось установить сетевой обмен, который они при желании могут шифровать, используя все тот же сессионный ключ. Относительная слабость симметричных шифровальных алгоритмов компенсируется тем, что все пароли, пересылаемые по сети, короткоживущие, и срок их действия истекает прежде, чем их бы удалось сломать. Авторы Kerberos не особо заботились о масштабируемости своего протокола, поскольку их целью являлись университетские сети с довольно большим числом не слишком мощных компьютеров (поскольку в эти сети входили и компьютеры студенческих кампусов), но все же не сопоставимых по размеру с

сетями масштаба целой страны. Но по мере развития этого протокола (переходу от версии 4 к Kerberos5) происходило одновременно и добавление в него дополнительной функциональности, позволяющей с помощью трастов объединять несколько секторов (realms) Kerberos в общую сеть.

История PKI шла совсем иным путём. Хотя постепенно она эволюционировала примерно к тому же, чем сейчас является Kerberos, но изначально она задумывалась как глобальная централизованная система. X.509 возник как стандарт OSI, организации, которая первой попыталась стандартизовать сетевые протоколы (ей же принадлежит стандарт на знаменитый 8-уровневый стек сетевых протоколов OSI, в настоящее время вытесненный более простым TCP). В частности X.509 возник, как стандарт аутентификации и авторизации для доступа к X.500 (Directory Access Protocol/DAP), интересного проекта, но тоже в настоящее время вытесненного своим более лёгким потомком - LDAP. X.500 предполагал создание глобального каталога, где идентичность двух узлов каталога удостоверялась бы специальными сертификатами, «подписанными» Certification Authority (CA), привязанной к узлу, который является для них, во-первых, общим предком, а, во-вторых, доверенной третьей стороной. Собственно, отголоски этой идеи до сих пор наблюдаются в OpenSSL, например в виде полей subject(субъект) и issuer(гарант) сертификатов, которые представляют по своему формату типичный Distinguished Name (DN), хорошо известный сетевым администраторам, знакомым с LDAP. Главным достоинством PKI является использование более сильных (хотя, одновременно, и более требовательных к вычислительным ресурсам) алгоритмов несимметричного шифрования, что позволяет иметь более долгоживущие сертификаты и ключи. Подробнее с форматом сертификатов можно познакомится из многочисленных руководств по OpenSSL/SSL/TLS, например из статей [2, 3]. В то же самое время большинство недостатков X.509 (в первую очередь претензия на универсальность - «один инструмент на все случаи жизни») связаны именно с тем, что стандарт X.500 никогда не был реализован в полном объёме[1].

Подробное обсуждение недостатков PKI можно найти в статье Петера Гутмана [4], а также в наборе слайдов его же авторства [5] с более язвительной, но одновременно более смешной критикой PKI (эта подборка, кстати, цитируется в английской Википедии в статье [6]). Читать слайды довольно трудно, но если вы восстановите канву его аргументов, используя статью [4], то пара часов непрерывного смеха вам гарантирована. В самом деле, единственным условием, позволяющим клиенту и серверу удостовериться в идентичности друг друга - это наличие ключей, сертифицированных (подписанных) третьей стороной. Но вне привязки к глобальному иерархическому дереву возникают естественные вопросы - «Кто такая это третья сторона и чью идентичность она сертифицирует?». Т.е. в рамках директории DAP сертификат X.509 имеет прозрачный смысл - это средство авторизации доступа к какому-то из узлов каталога, а иерархия CA естественным образом следует из иерархии самого каталога. В отсутствие же глобального каталога, поля subject и issuer никакого смысла не несут, а выдумывание смешных и фантастических значений для этих полей уже превратилось в особый жанр компьютерного фольклора. С сертификацией ситуация точно такая же - внутри организации используются или самоподписанные сертификаты, или же (при необходимости выйти во внешний мир) покупаются подписанные сертификаты у специализированных компаний, которые являются своего рода нотариальными конторами цифрового мира. Но, во-первых, удовольствие это не из дешёвых, подписанный сертификат считается одним из самых дорогих последовательностей битов (компания Thawte, которая принесла миллиарды Марку Шаттлворту, просила за 2кБ сертификат для Web сервера сроком на год до 160 долларов), а во-вторых, у компании его подписавшей нет технических средств гарантировать его достоверность (в отличии от обычных нотариусов, у которых такое средство есть - суд).

Это вытекает из другого основного недостатка X.509 - это стандарт, а не протокол. Имен-

[1]Несмотря на свою утопичность, идея глобальной информационной службы имеет право на существование. В конце концов существует же глобальная система доменных имён, и кто знает, если бы стек протоколов OSI не был бы вытеснен стеком TCP, может быть и стандарт X.500 получил бы практическое воплощение

но по этому компания Verisign (фактически монополист на рынке цифровых сертификатов) может продать вам последовательность бит, но она не может забрать у вас эти данные, если срок их действия истёк или у вас их украли злоумышленники. В самом деле, стандарт регламентирует только ведение чёрных списков CRL (certificate revocation list), поэтому если вы обратитесь в компанию, с просьбой аннулировать утерянный вами сертификат, то она, конечно же, внесёт этот сертификат в CRL. Однако это не гарантирует, что какой-нибудь из сетевых сервисов будет консультироваться с этим списком, когда он получит от клиента украденный у вас сертификат. Даже при условии добросовестности поставщика сетевых услуг, у злоумышленника есть довольно большой временной интервал, прежде чем запрос на аннулирование будет обработан, CRL пополнен, а пополненные CRL дойдёт до сетевого поставщика. Не говоря уже о том, что сверка сервером каждого полученного сертификата с многомиллионными «чёрным» списком выглядит достаточно устрашающей операцией.

Таким образом, всё что нам нужно от PKI - это ключ («Bring me the keys of Alfredo Garcia!», как это сформулировано в статье [4]). Семантика же этого ключа определяется уже внутренней политикой организации, т.е., например, привязка ключа и CA может осуществляться к внутреннему LDAP каталогу организации. В этом смысле интересно решение предложенное Heimdal - привязка ключей к принципалам Kerberos, а Certification Authority к KDC. В этом случае проблема аннулирования ключей решается простыми средствами - ликвидацией соответствующего принципала. Т.е. сертификат (или же smart-card) в данном случае выступает в роли ключа, который открывает пользователю доступ к супербилету (TGT), а дальнейшее обеспечение сетевой безопасности решается уже средствами Kerberos. Собственно это техническое решение мы и будем рассматривать в этой статье.

2 hxtool

Установка и конфигурирование сектора Kerberos описаны в статье [7]. Лучше всего брать последнюю версию Heimdal-Kerberos с их нового узла [8], или же использовать пакет уже имеющийся в вашем дистрибутиве (номер версии должен быть больше 0.8). Также как и в статье [7], предполагается, что сектор Kebreros называется MYREALM.RU, а контроллер сектора расположен на компьютере с именем kdc.myrealm.ru. Теперь можно приступать к изготовлению PKI сертификатов.

Как вы уже поняли из предыдущих аргументов, нас интересуют только самоподписанные сертификаты, которые предполагается использовать исключительно внутри сетевой структуры организации. Их можно изготовить с помощью консольной утилиты `openssl` (можно посмотреть, как это сделано в пакете heimdal, с помощью скрипта `lib/hx509/data/gen-req.sh` в исходниках heimdal). Но для решения поставленной нами задачи проще использовать аналогичную утилиту `hxtool`, реализованную средствами самого Heimdal. OpenSSL строилась с расчётом на использование сертификатов, подписанными внешним центром сертификации, поэтому, скажем, операция изготовления своего сертификата распадается на три этапа - изготовление частного ключа, оформление запроса на подпись сертификата (который может быть адресован внешней доверенной компании) и собственно подписывание сертификата.

Понятно, что все эти сложности не нужны, если вы собираетесь изготовлять самоподписанный сертификаты. Как, например, изготовление своего CA:

```
hxtool issue-certificate \
       --self-signed \
       --issue-ca \
       --subject="CN=CA,DC=myrealm,DC=ru" \
       --lifetime=10years \
       --generate-key=rsa \
       --certificate=FILE:ca.pem
```

Справку по программе `hxtool` и её командам можно получить с помощью

hxtool help и hxtool <command> -help. В данном случае выдаётся самоподписанный Certificate Authority сертификат сроком на 10 лет, выданный пользователю с DN CN=CA,DC=myrealm,DC=ru с использованием алгоритма шифрования RSA. Минимальную информацию о сертификате можно получить с помощью команды:

```
#hxtool print FILE:ca.pem
cert: 0
    private key: yes
    issuer:  "CN=CA,DC=myrealm,DC=ru"
    subject: "CN=CA,DC=myrealm,DC=ru"
    serial: 2460E08A177846F035AE02841F995F341D0E4557
    keyusage: cRLSign, keyCertSign, keyEncipherment, digitalSignature
```

Если открыть сертификат с помощью текстового редактора, легко заметить, что он состоит из двух Base64-encoded частей: собственно сертификата и приватного ключа. Из-за наличия в нём этого ключа его нужно хранить как Кащееву смерть, поскольку если он попадёт в руки злоумышленникам, то ничего хорошего вас не ждёт. Так что лучше всего создать каталог рядом с базами Heimdal /var/heimdal/secure, ограничив к нему доступ насколько это возможно, и поместить его туда. Все дальнейшие сертификаты будут генерироваться с помощью этого файла.

Чтобы клиент и сервер могли удостовериться в идентичности друг друга, оба они должны иметь копию сертификата CA, который, однако, уже сгенерирован и закодирован в первой секции сертификата. Так что этот сертификат нужно или просто вырезать из файла и сохранить с помощью текстового редактора в отдельном файле или же воспользоваться командой openssl x509 -in ca.pem -text > ca.crt. В дальнейшем этот файл можно будет выдавать по мере надобности службам и пользователям, заинтересованным в подключении к вашему PKI.

Есть определённые соглашения по расширениям, приписываемым сертификатам различного типа. Расширение .pem означает, что сертификат закодирован в формате Base64, и как правило состоит из двух частей, включающих в себя приватный ключ и собственно сертификат. Файлы с расширением .crt содержат только сертификат с публичным ключом, но без приватного, а .key – только приватный ключ без сертификата. Кроме того встречаются ещё .csr (Certificate Signing Request), но в данной статье они не используются.

Например, для создания защищённых страниц на Web сервере Apache вам понадобится сгенерировать ключ для сервера и клиента, одновременно подписав их с помощью приватного CA ключа:

```
hxtool issue-certificate \
       --subject="CN=www.myrealm.ru,DC=myrealm,DC=ru" \
       --type="https-server" \
       --hostname="www.myrealm.ru" \
       --generate-key=rsa \
       --ca-certificate=FILE:ca.pem \
       --certificate=FILE:https.pem
hxtool issue-certificate \
       --subject="CN=www.myrealm.ru,DC=myrealm,DC=ru" \
       --type="https-client" \
       --generate-key=rsa \
       --ca-certificate=FILE:ca.pem \
       --certificate=FILE:https-client.pem
```

Чтобы сервер заработал, необходимо будет переместить ключи https.pem и ca.crt подальше от посторонних глаз, но так, чтобы httpd-сервер мог найти их, и отредактировать конфигурационный файл /etc/httpd/extra/httpd-ssl.conf, указав там их расположение:

```
SSLCertificateFile "/etc/httpd/ssl/https.pem"
```

```
SSLCertificateChainFile "/etc/httpd/ssl/ca.crt"
SSLCACertificateFile "/etc/httpd/ssl/ca.crt"
```

и импортировать в браузер CA сертификат, что позволяет избавиться от надоедливых вопросов о неизвестной идентичности защищенного узла. Для Firefox это делается выбором пункта меню Edit->Prefernces, затем в диалоге настроек выбором закладки Advanced->Encryption, а затем уже в следующем диалоговом окне с помощью кнопки Import на закладке Authorities осуществляется загрузка файла сертификата. Такая процедура позволяет устанавливать шифрованное соединение (https) между сервером и клиентом.

Клиентский сертификат требуется, если сервер содержит страницы, специальным образом защищенные с помощью директив вида:

```
<Directory /var/www/StrongerSSL>
SSLVerifyClient require
SSLVerifyDepth 1
</Directory>
```

Для доступа к таким страницам браузерам нужно предъявить клиентский сертификат, наподобии того, что был нами создан ранее. Однако, не всё так просто. Большинство браузеров требуют, чтобы сертификаты этого типа были в бинарном PKCS12 формате, и защищены с помощью пароля, что делает их хранение на диске более безопасным. Хотя утилита `hxtool` поддерживает и этот формат, но не вполне, поскольку ею нельзя зашифровать сертификат. Поэтому придется прибегнуть к команде `openssl`:

```
openssl pkcs12 -export \
               -in https-client.pem \
               -name HTTPS-Client \
               -out https-client.p12
```

В процессе выполнения команда попросит вас ввести и подтвердить пароль, который вам потом будет использован при импорте сертификата `https-client.p12` в браузер. Делается это в том же диалоговом окне, что и выше, но только в этот раз кнопка Import будет располагаться на закладке «Your Certificates». Снимок с экрана как раз и демонстрирует эту закладку.

Можно сравнить как та же процедура осуществляется средствами только команды `openssl` в статье [3] или в известном mini-howto [9]. Как видите, использование `hxtool` существенно сокращает дистанцию.

Таким образом, если не выдвигать слишком высоких требований, типа защищённых паролем сертификатов и ключей, или использования алгоритма DSA для шифрования, то утилиты `hxtool` достаточно, чтобы управлять PKI предприятия.

3 pkinit

Как вы могли заметить, пока речь не шла о Kerberos. Объединение PKI и Kerberos обычно обозначают термином pkinit (это слово можно расшифровать как public key init), что позволяет получать доступ к сектору Kerberos с помощью открытых ключей, smartcard, биометрических средств и др. Следует иметь в виду, однако, что стандарт в настоящее время не устоялся, и возможно будет меняться в будущем. Спецификации на него находятся в разработке у Kerberos Working Group [10].

Посмотрим как можно организовать такой доступ с помощью нашего самодельного Certification Authority. Наподобие того как в предыдущей части это было проделано для httpd-сервера и клиента, сгенерируем сертификат для KDC, точнее сервиса, ответственного за выдачу супербилетов, и для любого принципала из базы Heimdal:

```
hxtool issue-certificate \
        --type="pkinit-kdc" \
```

```
        --pk-init-principal="krbtgt/MYREALM.RU@MYREALM.RU" \
        --subject="uid=kdc,DC=myrealm,DC=ru" \
        --generate-key=rsa \
        --ca-certificate=FILE:ca.pem \
        --certificate=FILE:kdc.pem
hxtool issue-certificate \
       --type="pkinit-client" \
       --pk-init-principal="mike@MYREALM.RU" \
       --subject="uid=mike,DC=myrealm,DC=ru" \
       --generate-key=rsa \
       --ca-certificate=FILE:ca.pem \
       --certificate=FILE:mike.pem
```

Поскольку все сгенерированные файлы содержат приватные ключи, то не стоит их разбрасывать, где попало. Ключ `kdc.pem` прячем в уже имеющейся директории `/var/heimdal/secure`, а `mike.pem` выдаём пользователю, соответствующему принципалу `mike@MYREALM.RU`, и строго предупреждаем его, чтобы он спрятал этот ключ в своём домашнем каталоге (например, в подкаталоге `/home/mike/secure`) и никому его не показывал (`chmod -R 600 /home/mike/secure`).

Собственно, после этого можно приступать к конфигурированию Heimdal, что включает в себя правку двух файлов - `/etc/krb5.conf`, ответственного за настройку клиентской библиотеки и приложений, и `/var/heimdal/kdc.conf`, который отвечает за работу сервера KDC. Начнём с последнего, где необходимо поправить секцию `[kdc]`, указав там расположение приватного ключа kdc и сертификата CA:

```
[kdc]
        enable-pkinit=yes
        pkinit_identity = FILE:/var/heimdal/secure/kdc.pem
        pkinit_anchors=FILE:/etc/ssl/ca.crt
```

Сертификат CA должен быть доступен и клиентской части Kerberos, поэтому его лучше выложить в общедоступный каталог (убедившись предварительно, что из файла удалена секция приватного ключа) и отредактировать `/etc/krb5.conf`

```
[appdefaults]
        pkinit_anchors = FILE:/etc/ssl/ca.crt
[kinit]
 MYREALM.RU = {
        pkinit_anchors =  FILE:/etc/ssl/ca.crt
 }
```

Кроме того, можно создать дополнительный файл `/var/heimdal/pki-mapping`, где хранится таблица соответствий между принципалами Kerberos и субъектами (поле `subject`) PKI сертификатов. На самом деле, в данном случае эта таблица нам не нужна, поскольку соответствующий принципал записан в поле `subjectAltName` выданных нами сертификатов, благодаря использованию ключа `--pkinit-principal` при их создании. В этом можно легко убедиться:

```
#hxtool print --content FILE:mike.pem
cert: 0
    private key: yes
    issuer:  "CN=CA,DC=myrealm,DC=ru"
    subject: "UID=mike,DC=myrealm,DC=ru"
    serial: 51DBAEA05A82924AAA2DFFEE173B922872AA7E48
    keyusage: keyEncipherment, digitalSignature
subject name: UID=mike,DC=myrealm,DC=ru
```

```
issuer name: CN=CA,DC=myrealm,DC=ru
Validity:
        notBefore 2008-04-11 20:29:22
        notAfter  2009-04-12 20:29:22
checking extention: keyUsage
checking extention: extKeyUsage
checking extention: subjectAltName
subjectAltName otherName pk-init: mike@MYREALM.RU
checking extention: authorityKeyIdentifier
        authority key id: B53226CCEA74AC84A4F88624E09D77EC270FEA15
checking extention: subjectKeyIdentifier
        subject key id: 9F096F50AC6000E11AB82E075A3434FC951B7069
checking extention: basicConstraints
        is NOT a CA
Not a CA nor PROXY and doesn't haveCRL Dist Point
```

Но файл `/var/heimdal/pki-mapping` может оказаться полезным для «нестандартных» [2] сертификатов. Его формат выглядит примерно так (обратите внимание на обратный порядок полей в DN):

```
mike@MUREALM.RU:DC=ru,DC=myrealm,UID=mike
```

Теперь осталось только перезапустить KDC, чтобы пользователь mike мог получать свой супербилет командой:

```
#kinit  --pk-user=FILE:mike.pem mike
#klist -v
Credentials cache: FILE:/tmp/krb5cc_P23967
        Principal: mike@MYREALM.RU
    Cache version: 4

Server: krbtgt/MYREALM.RU@MYREALM.RU
Client: mike@MYREALM.RU
Ticket etype: des3-cbc-sha1, kvno 1
Ticket length: 364
Auth time:  Apr 16 13:00:29 2008
End time:   Apr 17 13:00:29 2008
Ticket flags: forwardable, initial, pre-authenticated
Addresses: addressless
```

Heimdal также позволяет использовать сертификаты в формате PKCS11 – одного из стандартов, получившим распространение при работе со smart-card. Его практическое применение зависит от оборудования для чтения пластиковых карт, установленном на вашем компьютере, но в демонстрационных целях можно поэкспериментировать с программным модулем `soft-pkcs11`, разработанным одним из основных авторов Heimdal – Лав Эрнквист Острандом [11]. В данном случае модуль позволяет просто более длинным путём добраться до того же самого сертификата. В последних версиях Heimdal (выше 1.0) этот модуль входит в состав Heimdal в виде библиотеки `libhx509.so`, так что можно поэкспериментировать с ним не устанавливая дополнительных библиотек. Конфигурация модуля осуществляется с помощью rc-файла, на который указывает переменная окружения `SOFTPKCS11RC`. Для наших целей этот файл должен выглядеть таким образом (заметьте, что разделителем полей является табуляция!):

```
certificate     Certificate     Mike       FILE:/home/mike/secure/mike.pem
```

[2] Хотя что тут считать стандартом?

Т.е. в нем просто указано расположение на жёстком диске pem-сертификата. После этого запуск:

```
kinit --pk-user=PKCS11:libhx509.so mike
```

позволяет получить супербилет. Понятно, что это имеет смысл в тестовых целях, в реальных «полевых» условиях вместо динамической библиотеки необходимо указать модуль, который управляет устройством для считывания информации со smart-card, установленном в компьютере.

В качестве более или менее практического примера применения этих возможностей Heimdal, попробуем открыть доступ к некоей рабочей станции под управлением Linux только для обладателей pem-сертификатов, записанных на сменный носитель (USB флешку). Т.е. сертификат сбрасывается на неформатированный usb-носитель обычным cat-ом `cat mike.pem > /dev/usb/sda1`, после чего сертификат на жёстком диске стирается, и в дальнейшем доступ к компьютеру разрешается только пользователям, которые правильно указали имя принципала, находящегося в базе Heimdal, и предъявили на флешке соответствующий сертификат.

Хотя возможности pkinit ещё не задействованы в программу `login` из состава Heimdal, но эту задачу можно решить либо с помощью PAM-модуля pam-krb5 [12], либо в тестовых целях можно обойтись простым bash-скриптом `login.sh`:

```
#!/usr/bin/bash
echo -n "Insert pem-certificate and hit enter when ready!"
read
/usr/bin/kinit -C FILE:/dev/sda1 $2 && \
/usr/bin/kgetcred host/$(hostname -f) && \
/usr/bin/login -f $2
```

который запускается из `/etc/inittab`, сконфигурированного следующим образом:

```
...
c1:1235:respawn:/sbin/agetty -l /usr/local/bin/login.sh 38400 tty6 linux
c2:1235:respawn:/sbin/agetty -l /usr/local/bin/login.sh 38400 tty6 linux
...
```

Этот скрипт просит пользователя вставит сменный носитель в usb-slot, после чего запускает kinit. При безошибочном получении супербилет он пытается получить билет для доступа к рабочей станции, как к принципалу `host/<имя компьютера>` [3], а после этого уже запускается программа login из пакета Heimdal в «пользователь-уже-зарегистрирован» режиме (ключ `-f` в вызове `login`). Дёшево, но сердито!

Таким образом, новые версии Heimdal позволяют объединить сильные криптографические алгоритмы, используемые инфраструктуре публичных ключей, и гибкость настроек Kerberos для контроля доступа к сетевым ресурсам.

Список литературы

[1] М.Кондрин. Изучаем принципы работы heimdal kerberos. *Системный Администратор*, 6, 2005.

[2] Всеволод Стахов. Теория и практика OpenSSL. *Системный Администратор*, 1:17–26, Jan 2003.

[3] Всеволод Стахов. Практика OpenSSL. *Системный Администратор*, 1:32–36, Feb 2003.

[4] Peter Gutman. PKI: It's Not Dead, Just Resting. *Computer*, 35(8):41–49, Aug 2002.

[3] Хотя это не обязательно, но именно таким образом работает стандартный Kerberized login

[5] Peter Gutman. Everything you never wanted to know about pki but were forced to find out. external reference on `http://en.wikipedia.org/wiki/Public_key_infrastructure`.

[6] Статья в wikipedia о pki.

[7] М.Кондрин. Развертываем heimdal kerberos. *Системный Администратор*, 7, 2005.

[8] Последняя версия heimdal-kerberos.

[9] Van's Apache SSL/TLS mini-HOWTO [online].

[10] Спецификации pkinit - draft-zhu-pkinit-ecc-03.txt.

[11] Лав Эрнквист Остранд pkcs11 модуль.

[12] Kerberized pam-модуль.