

Документ можно скачать в виде PDF по адресу: <http://bilbo.online.bg/~assen/radius.pdf>

## **ДИЗАЙН БИЛЛИНГ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ RADIUS И ВНЕШНЕЙ ВЕРИФИКАЦИИ**

### **ВВЕДЕНИЕ**

RADIUS (Remote Authentication Dial-In User Service) – стандартный протокол для верификации потребителей, разработанный голландской компанией Cistron. Протокол функционирует на базе диалога “клиент-сервер”.

На сегодняшний день практически все производители NAS (Network Access Servers) имеют встроенную поддержку этого протокола (встроенный RADIUS клиент). Поскольку протокол общедоступен, существует множество имплементаций сервера.

Исторически первым RADIUS сервером был Cistron RADIUS, хранящий всю конфигурацию, а также информацию от сессиях клиентов в текстовых файлах. В связи с развитием SQL как удобной платформы сохранения и обработки данных, был создан патч для Cistron RADIUS, позволяющий хранить данные в SQL массиве. Неудобство заключается в том, что структура SQL массива просто повторяет структуру текстовых файлов, не используя множество удобств реляционных баз данных, предлагаемые SQL.

Для использования полной мощи SQL требуется изменить структуру информационного массива. Новая структура, в свою очередь, требует новых средств для записи и извлечения данных, которыми Cistron RADIUS не располагает.

XtRADIUS был разработан на базе кода Cistron RADIUS, но с одним отличием – он использует внешнюю верификацию потребителей, что позволяет содержать всю поступающую информацию в удобном виде.

На нынешний день множество других имплементаций RADIUS сервера позволяют проводить внешнюю верификацию.

### **ДИЗАЙН СТРУКТУРЫ**

При использовании внешней верификации коммуникация проходит следующим образом:

1. Потребитель сети подключается к ней (напр., модемом).
2. NAS устанавливает протокол связи (чаще всего – Point-to-point Protocol, PPP), получает от потребителя его имя и пароль и передает их вместе со своим ключом и идентификатором обмена указанному RADIUS серверу.
3. RADIUS сервер выделяет имя потребителя, его пароль, а также другие атрибуты связи (например, имя или IP адрес NAS) и подает их внешнему слою для верификации.
4. Слой верификации делает справку в информационном массиве SQL и принимает решение верифицировать потребителя или нет. Сообщение передается RADIUS серверу при помощи exit code. Если необходимо, перед этим можно передать RADIUS серверу информацию для RADIUS-клиента.
5. В зависимости от решения верификационного слоя, RADIUS сервер дает команду RADIUS клиенту о допуске или сбрасывании потребителя.

## УСТАНОВКА RADIUS СЕРВЕРА

XtRADIUS можно скачать по следующему адресу: <http://xtradius.sourceforge.com>. XtRADIUS устанавливается обычным для C программ способом: `./configure && make && make install`.

Перед компиляцией не мешает проверить IP порты установки RADIUS – раньше пользовались 1645 и 1646, с некоторого времени – 1812 и 1813. Сервер может работать только на одной паре портов, для замены портов нужна перекомпиляция. Сервер и клиент должны работать на одной и той же паре портов.

Конфигурация XtRADIUS находится в директории `/etc/raddb`. Ниже показана конфигурация главных файлов для внешней верификации потребителей:

**`/etc/raddb/naslist`** – содержит список NAS

```
# NAS Name                Short Name      Type
#-----
nas1.somedomain.com      nas1           other
nas2.somedomain.com      nas2           other
nas3.somedomain.com      nas3           other
```

**`/etc/raddb/naslist`** – содержит список ключей NAS

```
# Client Name             Key
#-----
nas1.somedomain.com      Key1
nas2.somedomain.com      Key2
nas3.somedomain.com      Key3
```

**Н.В.:** Каждый клиент должен знать свой ключ (Key).

**`/etc/raddb/users`** – содержит список RADIUS атрибутов и их сокращенных обозначений, которые можно подавать внешним программам при верификации:

```
# Used letters: a b c d e f g h i k n o p s t u w x y z
# Free letters: b j l m q r v
```

```
NAS-IP-Address           n
Framed-MTU                t
User-Name                 u
Password                  w
Callback-Number           c
Framed-Protocol           a
Acct-Session-Time         e
Acct-Input-Octets         i
Acct-Output-Octets        o
NAS-Port-Type             y
Acct-Status-Type          d
Connect-Info              s
Calling-Station-Id        g
Called-Station-Id         h
NAS-Port-Id               p
Framed-IP-Address         f
Acct-Terminate-Cause      z
Acct-Session-Id           b
```

`/etc/raddb/execparams` – содержит список внешних программ для верификации и список параметров, подаваемых им. Слой для верификации использует два типа RADIUS пакетов: Alive и Stop. Start пакет не используется по той причине, что он не содержит в себе IP адрес, присвоенный потребителю. Этот адрес приходит лишь в Alive пакете. Разница во времени двух пакетов – не более нескольких секунд, что существенно на статистику потребителей не влияет. При самой верификации потребителя пароль поставляется внешнему слою не через command-line параметр, а через environment variable. Это делается с целью избежания появления пароли в log файлах.

```
DEFAULT Acct-Status-Type = "Alive"
Exec-Program-Account = "/usr/local/bin/alive.pl %u %n %p %y %g %b %f"

DEFAULT Acct-Status-Type = "Stop"
Exec-Program-Account = "/usr/local/bin/stop.pl %u %n %p %i %o %z %e %b"

# The password is not supplied via %w any more,
# but via "User-Password" environment variable.
# Thus we avoid having it logged in plain text in radius.log.

DEFAULT Auth-Type = "External"
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-Routing = Broadcast-Listen,
Framed-MTU = 1500,
Framed-Compression = Van-Jacobsen-TCP-IP,
Exec-Program-Wait = "/usr/local/bin/login.pl %u %n %p %y %g",
Fall-Through = Yes
```

## SQL МАССИВ

Чтобы полностью воспользоваться возможностями SQL, информационный массив был разработан в следующем виде:

- Таблица users, содержащая информацию о потребителях: имя, пароль, тип сервиса, последний день, когда клиенту разрешен доступ, остаточное время (если у клиента prepaaid сервис), а также другая информация – напр., телефон и email для связи и т.д. Для каждого потребителя делается одна запись в таблице.
- Таблица payments, содержащая информацию о поступивших взносах клиентов. Содержит в себе по одной записи для каждого взноса. Запись содержит в себе имя потребителя, день и час оплаты, сумма оплаты, тип сервиса, день начала сервиса и последний день сервиса.
- Таблица services, содержащая вспомогательную информацию о типах сервисов.
- Таблица access, содержащая по одной записи для каждой сессии потребителя. Запись отражает в себе имя потребителя, дня и час начала сессии, день и час конца сессии.
- Таблица rejects, в которой записываются все отброшенные попытки верификации. Содержит по одной записи для каждой попытки – день и час, имя и пароль, переданные потребителем, причина отказа.

В целях безопасности системы пароли потребителей не следует хранить в явном виде. Поскольку стандартная функция `crypt()` дает слишком слабое по современным меркам DES шифрование, я рекомендую использовать встроенную в SQL функцию `password()`. Хотя это шифрование не использует элемент рандомизации (т.е. `password()` от данной последовательности символов всегда возвращает одну и ту же последовательность символов), оно работает быстрее `crypt()` (из-за отсутствия необходимости выделять salt из зашифрованного при помощи DES пароля), к тому же этот механизм шифрования менее популярен и средств для bruteforce атак на зашифрованные таким образом пароли гораздо меньше.

## СЛОЙ ВЕРИФИКАЦИИ

Слой верификации (в данном случае, написанный на Perl), выполняет последовательность действий, которая при поступлении имени и пароля выглядит примерно так:

1. Пытается забрать зашифрованный пароль для данного потребителя, зашифровать поданный потребителем пароль, взять тип сервиса, остаток времени и – если есть – статический IP адрес (например, при помощи "SELECT PASSWORD('some\_pass') AS PASSWD, PASSWORD, EXPIRES, TIME\_LEFT, IP\_ADDR FROM users WHERE username='some\_name'").
2. Если в ответе PASSWORD пустой, то потребитель может быть сразу отброшен, а в таблицу rejects записать сообщение о несуществующем потребителе. Если PASSWORD и PASSWD не совпадают, то потребитель может быть сразу отброшен, а в таблицу rejects записать сообщение об ошибочном пароле.
3. Проверить, имеет ли потребитель право на доступ сегодня (сравнить EXPIRES с нынешней датой). Сравнение можно облегчить, если вместо EXPIRES в т.1 взять разницу в днях между EXPIRES и нынешним днем, например "TO\_DAYS(EXPIRES) – TO\_DAYS(CURDATE()) AS EXP". Если EXP меньше 1, то доступ потребителю можно отказать и таблицу rejects записать сообщение о неоплаченном сервисе.
4. Проверить сервис потребителя и если он prepaid, то нужно проверить, является ли остаток времени TIME\_LEFT положительным. Если нет, то доступ потребителю можно отказать и таблицу rejects записать сообщение о неоплаченном сервисе.
5. Проверить, не подключен ли уже к сети это потребитель – большинство ISP разрешают только по одной сессии на потребителя в каждом моменте времени – например, при помощи "SELECT COUNT(\*) FROM access WHERE username='some\_user' and TIME\_END IS NULL". Если число больше нуля, то доступ потребителю можно отказать и таблицу rejects записать сообщение о попытке повторного подключения. Исключение от правила – ISDN, если потребителю разрешено пользоваться обоими каналами.
6. Потребителю разрешается доступ. Если у потребителя статический IP адрес, то стоимость IP\_ADDR передается RADIUS серверу при помощи пары "Framed-IP-Address=xx.xx.xx.xx". Если такого нет, то RADIUS клиент должен сам выбрать IP адрес для потребителя из своего пула (который адрес появиться затем в Alive пакете). Само разрешение доступа дается тем, что программа верификации заканчивает свое действие нулевым exit code. Если exit code ненулевой, это и есть указание RADIUS серверу отказать в доступе потребителю.

При получении Alive пакета слой верификации делает запись в таблице access, в которой устанавливает имя потребителя, день и час начала сессии, а также дополнительно желанная информация (напр., имя NAS-а, порт и др.). При этом поле в таблице, где указываются день и час окончания сессии, остается со значением NULL, что используется для проверки существующего подключения к сети.

При получении Stop пакета слой верификации должен найти запись, соответствующую данной сессии, и дополнить ее днем и часом окончания сессии, а также – если сервис у потребителя prepaid - вычитать из TIME\_LIMIT продолжительность сессии. При наличии одного NAS, для опознавания сессии можно пользоваться параметром Session-ID, который передается NAS-ом в Alive и Stop пакетах. Если воспринять этот подход, при записи Alive пакета отмечается и Session-ID. Однако в большинстве NAS счетчик Session-ID обнуляется при рестартировании, так что если в сети несколько NAS одного типа, возможны ложные сигналы. В таком случае рекомендую опознавание сессии в базе данных вести по двум параметрам, которые тоже присутствуют в Alive и Stop пакетах – IP адрес NAS-а и порт потребителя (NAS-IP-Address и NAS-Port-ID). При получении Alive пакета, IP адрес и порт отмечаются в начале сессии.

## ИНТЕРФЕЙС УПРАВЛЕНИЯ

Конечной целью использования внешней верификации было иметь удобный интерфейс для управления биллингом. При описанной структуре можно запросто осуществлять управление потребителями, заменять пароли и т.д. Самый интересный на мой взгляд вопрос – ввод оплаты потребителей. В нашей системе два основных типа сервиса:

- Месячный сервис – потребитель платит фиксированную сумму и получает доступ без всяких ограничений на 1 месяц со дня оплаты.
- Prepaid сервис – потребитель покупает некое количество часов доступа, которыми может пользоваться в течение года.

Для эффективного управления оплатой мы ввели следующие критерии:

- Если у потребителя нет сервиса, это новый сервис становится активным в момент оплаты (т.е. в таблице users на сторке потребителя должны появиться сразу же новые EXPIRES и, если надо, TIME\_LIMIT стоимости).
- Если у потребителя есть сервис, то новый сервис будет иметь начальную дату в будущем. Для решения этого случая, каждую ночь в 00:00 выполняется небольшой скрипт, который проверяет есть ли сервисы, для которых первый день – только-что наступивший и если да, то скрипт делает соответствующие изменения в таблице users. Этот механизм позволяет также сделать перерыв в сервисе потребителя (напр., придти в офис 5-ого июля и заплатить с 12 июля по 11 августа и затем с 28-ого августа по 27-ое сентября).
- Если у потребителя текущий сервис – prepaid и новый тоже, то вновь закупленные часы доступа следует прибавить к текущей стоимости TIME\_LIMIT, а к стоимости EXPIRE прибавить год.

Использование SQL позволяет также удобно вывести в веб информацию о предыдущих оплатах каждого клиента, делать справки о сессиях, расходуемом и оставшемся времени (вкл. и предоставить возможность клиенту самому наводить справки в любое время суток), показывать в удобном виде когда и кому был отказан доступ и почему, печатать квитанции потребителям при самой оплате и т.д.