"Построение системы управления сетевыми соединениями на базе протокола RADIUS"

Хрущев Сергей Анатольевич, Черенкова Светлана Юрьевна
Омский Филиал НИУ Институт математики имени С.Л. Соболева СО РАН
hrushev@okno.ru, scher@omsk.net.ru

В настоящее время все большую значимость приобретает обеспечение защиты сетевых коммуникаций. Данная проблема является достаточно объемной и включает в себя множество задач более локального характера. Одна из таких задач - построение программной среды для идентификации удаленных сетевых подключений к серверам доступа (NAS). Так как получение неавторизованного доступа к сетевым ресурсам зачастую является первым шагом к различным действиям разрушительного характера, создание качественной системы идентификации удаленных сетевых подключений представляется очень важным.

На данный момент существует достаточно много различных систем сетевой идентификации на базе протокола RADIUS, например Livingston RADIUS (сейчас Lucent RADIUS), Cistron RADIUS, YARD RADIUS, KISS RADIUS, FreeRADIUS, Merit, RAM (RADIUS authentication manager), XtRadius, Ascend Radius и другие. Главной отличительной особенностью отмеченных систем контроля доступа является их ориентация преимущественно на работу под управлением различных версий ОС Unix. Их настройка также обычно осуществляется по принятой в Unix схеме, т.е. с использованием множества текстовых конфигурационных файлов. Использование данного программного обеспечения, однако, предполагает его тесную интеграцию как минимум с системами учета сетевых ресурсов и системами внутрисетевой авторизации в корпоративных сетях. Так как базовые версии демонов RADIUS редко бывают ориентированы на подобную интеграцию, их использование обычно подразумевает установку так называемых “патчей”, работа которых, как правило, заключается в подмене стандартных модулей чтения конфигурации пользователей и сохранения статистики. Множество примеров подобных модификаций можно встретить, например, на русскоязычном сайте OpenNet (http://www.opennet.ru/prog/sml/53.shtml). Такая доработка, естественно, не ведет к повышению надежности работы программ, так как авторская документация на исходные тексты в большинстве случаев отсутствует, и модификация делается “методом проб и ошибок”. Дополнительной проблемой для ориентированных на ОС Windows пользователей является перекомпиляция оригинальных исходных текстов (в большинстве случаев написанных на языке С, и использующих вызовы системных библиотек Unix). Подобные сложности могут возникнуть и при переносе исходных текстов между разными версиями Unix, например, когда программа использует специфичные для некоторой системы возможности.

Одной из серьезных проблем безопасности существующих систем является хранение зашифрованных алгоритмом DES паролей пользователей в текстовых файлах. Так как в настоящее время широко известен факт недостаточной криптостойкости стандартного алгоритма DES, весьма желательным было бы изменение формата хранения паролей. Однако практически во всех широко используемых демонах RADIUS до сих пор продолжает использоваться старый метод шифрования.

Устранением большинства существующих проблем могло бы стать создание сервера RADIUS, построенного на базе современных стандартов программирования, обеспечивающего портирование на различные операционные системы, и изначально допускающего подключение внешних модулей авторизации и статистики.

Именно такое решение и было принято в нашей организации. В качестве языка программирования был выбран язык Java, как наиболее полно удовлетворяющий предъявленным требованиям. Прежде всего, он обеспечивает многоплатформенность системы (прямая поддержка JRE в настоящее время присутствует в Solaris, Linux и Windows, JRE также портирована на FreeBSD). Кроме того, в Java на современном уровне реализована поддержка сетевых возможностей. Дополнительным плюсом этого языка является наличие поддержки доступа к базам данных. Помимо достоинств, у данного решения существуют и некоторые недостатки, например, несколько завышенные требования к объему оперативной памяти на сервере. Однако в

современных условиях использование программой десятка лишних мегабайт уже не кажется чем-то чрезмерным. Так как время обработки запроса от NAS является критичной величиной, принимая решение о выборе языка Java, мы изучали вопросы, связанные с быстродействием разработанных на нем приложений. Выводы делались на основе изучения работы механизма JSP, который, будучи написанным на Java, по существу является интерпретатором некоторого языка. Опытным путем было обнаружено, что существенное падение производительности происходит только в момент компиляции кода сервлета. В то время как интерпретация кода страницы происходит без видимых задержек. Так как RADIUS-сервер не требует больших вычислительных ресурсов, мы посчитали такие временные характеристики удовлетворительными и для нашего случая, что и было впоследствии подтверждено испытаниями RADIUS-сервера.

В настоящее время нами создана тестовая версия RADIUS-сервера, включающая в себя модули авторизации и статистики, поддерживающие обмен информацией с сервером баз данных Oracle. Модули авторизации и статистики реализованы с использованием технологии JDBC. Данный подход обеспечил возможность хранения пользовательских учетных записей в таблицах базы данных, благодаря чему получить информацию о паролях становится значительно труднее, чем из текстовых файлов, т.к. для этого необходимо пройти процедуру авторизации сервера Oracle. Используя технологию Java Cryptography Extensions, появляется возможность шифрования паролей с помощью множества современных эффективных методов шифрования, например Blowfish.

Разработанный сервер RADIUS представляет собой многопоточное приложение. Сервер имеет следующую структуру:

- основной поток;
- поток поддержки авторизации;
- поток поддержки статистики.

Основной поток отвечает за запуск, управление работой, и останов сервера. Внутри него создаются и стартуют два вспомогательных потока. Когда сервер находится в рабочем состоянии, основной поток служит для управления сервером. В настоящее время реализовано простейшее управление с передачей команд через сокеты. Как команды, так и ответы сервера представляют собой текстовые строки, поэтому управление может осуществляться как путем запуска второй его копии в режиме клиента, так и с помощью клиента TELNET.

Как и следует из контекста, поток поддержки авторизации отвечает за авторизацию клиентов NAS. Этот поток получает запросы на авторизацию клиентов и отправляет их менеджеру авторизации, который после обработки запроса возвращает один из ответных пакетов Accept, Challenge или Reject. Таким образом, достигается разделение обязанностей между собственно RADIUS- сервером и механизмом авторизации.

Аналогично поток поддержки статистики отвечает за получение пакетов статистики и отправку ответных пакетов NAS. В данном случае запросы также обрабатываются специализированным менеджером статистики.

Благодаря разделению процессов обработки и передачи информации ядро сервера становится достаточно универсальным. Такой подход избавляет от необходимости вмешательства в код ядра сервера при смене механизмов обеспечения авторизации или обработки статистики. После проведения тестирования и уточнения функций всех частей системы планируется разработать стандартный интерфейс для взаимодействия ядра с внешними модулями.

Установка и настройка RADIUS-сервера практически не зависят от операционной системы. Достаточно просто скопировать JAR-архивы в нужный каталог и установить параметры подключения к NAS и к серверу баз данных. Все настройки, касающиеся параметров подключения пользователей, зависят только от используемых модулей авторизации и статистики. В нашем случае внешние модули пользуются информацией, хранящейся на сервере баз данных Oracle, для чего был создан набор соответствующих таблиц. Для каждого пользователя определяется необходимый набор RADIUS-атрибутов, которые упаковываются в пакет и передаются ядру RADIUS-сервера.

Безопасность системы управления сетевыми соединениями в данном случае определяется степенью защищенности механизма хранения паролей и степенью защищенности канала, по которому выполняется передача информации между RADIUS-сервером и NAS.

Защищенность коммуникаций по протоколу RADIUS обеспечивается несколькими составляющими:

1. Установкой на NAS и сервере RADIUS закрытого ключа достаточной длины;
2. Генерацией на NAS удостоверений (authenticators) пакетов с использованием “хорошего” генератора случайных чисел, а также отсутствием повторений ранее сгенерированных значений;
3. Стойкостью алгоритма хеширования MD5 [4].

При этом генерация удостоверений полностью зависит от используемого NAS. Под “хорошим” генератором случайных чисел здесь понимается генератор, обеспечивающий равномерное распределение значений. Тем самым обеспечивается равновероятное (и непредсказуемое) появление удостоверений пакетов. Отсутствие повторений обеспечивает невозможность подделки злоумышленником Response-пакета путем повторного использования Response-удостоверения. Не имея алгоритма генерации, используемого NAS, тяжело судить о выполнении данного требования. Однако в силу того, что число возможных удостоверений равняется 2^{128}, можно предположить, что даже простая генерация удостоверений, дающая равновероятные исходы, обеспечит приемлемый результат.

Установка закрытых ключей является, пожалуй, наиболее важной из трех отмеченных составляющих, так как выбор таких ключей осуществляется администратором NAS. Установка пустого ключа сильно ослабляет защиту, поэтому программное обеспечение некоторых NAS попросту не позволяет выполнять запросы по протоколу RADIUS в данном случае. В то же время, отдельные NAS продолжают работать и с пустым ключом. Примером тому может служить сервер доступа Cisco 2511 с различными вариантами IOS.

За последние несколько месяцев авторы неоднократно встречали в Internet сообщения, касающиеся обнаружения уязвимостей в механизме обеспечения защиты протокола RADIUS. В силу неофициальности подобной информации, в настоящее время трудно судить о ее достоверности. Однако приведенные в сообщениях данные свидетельствуют о наличии методов, позволяющих проводить анализ хеш-значений MD5 и взламывать, таким образом, закрытый ключ, зная удостоверение пакета. Для противодействия подобным методам взлома авторами сообщений предлагается использовать закрытые ключи длиной свыше 16 или даже свыше 32 символов.

В общем случае, для проведения атаки на систему, использующую протокол RADIUS, злоумышленнику необходимо обеспечить:

1. Возможность перехвата UDP пакетов, идущих от NAS к серверу RADIUS и обратно.
2. Возможность подмена UDP пакетов, идущих от сервера RADIUS к NAS.
3. Знание секретного ключа, либо эффективный алгоритм его восстановления.

Невозможность выполнения отмеченных действий в большинстве случаев обеспечит безопасное управление сетевыми соединениями.

Мы предполагаем развивать RADIUS-сервер в нескольких направлениях. Прежде всего – это интеграция с биллинговой системой. Второе направление – обеспечение поддержки IPv6.

Литература:

1. Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
2. Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
3. Aboba, B., Zorn, G. and D. Mitton, "RADIUS and IPv6", RFC 2869, August 2001.
4. Rivest, R. and S. Dusse, "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.