Содержание

Данное пособие рассчитано на людей, обладающими базовыми знаниями в области сетевых технологий:

- эталонная модель ISO/OSI;
- сети Ethernet, протоколы и технологии, используемые для их построения;
- сети Wi-Fi, протоколы и технологии, используемые для их построения
- стек TCP/IP.

I. Атаки

Общее название совокупности средств, разработанных для защиты данных и предотвращения действий злоумышленников, есть *компьютерная безопасность*. Термин *сетевая безопасность* предполагает наличие средств для защиты данных при их передаче. Но на самом деле нет четких границ между этими двумя терминами.

Атака – любое действие, которое компрометирует безопасность информации, принадлежащей организации.

1. Обобщенный сценарий атаки

Рассмотрим обобщенный сценарий атаки, который можно представить в виде следующих шагов:

- ✓ пассивная разведка;
- ✓ активная разведка;
- ✓ выбор (разработка) эксплойта;
- ✓ взлом целевой системы;
- ✓ загрузка «полезного груза» (которым, как правило, является вредоносная программа);
- ✓ сокрытие следов взлома.

Конечно, данная последовательность может быть нарушена или могут быть исключены отдельные шаги данного сценария. Кратко рассмотрим этапы.

Пассивная разведка

Данный этап называется пассивным, так как злоумышленник не входит в непосредственный контакт с целью или входит с соблюдением общепринятых правил (например, посещая сайт компании, куда входит целевая система).

Целью данного этапа является сбор информации о цели. В качестве цели может выступать как конкретный хост (сервер), так и целая сеть организации. Достаточно часто цель выбирается из условия простого поиска системы, содержащей известную уязвимость, к которой злоумышленник имеет эксплойт.

Активная разведка

Этап проведения активной разведки называют сканированием. Злоумышленник пытается определить структуру интересующей его сети, точки доступа, доступные узлы и хосты, расположение маршрутизаторов и межсетевых экранов, установленные операционные системы и их версии, открытые порты и работающие службы, версии установленных программных продуктов. На этом этапе злоумышленник входит в контакт с объектами интересующей его системы, поэтому его действия могут быть обнаружены средствами защиты целевой системы.

Для решения задач проведения активной разведки можно использовать большое количество разнообразных средств, многие из которых являются общедоступными (ping, traceroute, nmap, SuperScan).

Может использоваться «замедленное» сканирование, когда злоумышленник посылает отдельные запросы через большие интервалы времени. В этом случае достаточно высока вероятность того, что на целевой системе не обратят внимания на отдельные запросы.

Выбор эксплойта

На основании полученных данных злоумышленник выбирает, модифицирует имеющийся или разрабатывает новый эксплойт для проведения взлома.

Взлом целевой системы

Существует множество различных способов взлома. К их числу можно отнести получение доступа к системе, расширение имеющихся полномочий и отказ в обслуживании. Нападающий и защищающийся находятся в неравных условиях. Для защиты системы необходимо знание обо всех возможных атаках против данной системы, а для проведения атаки достаточно найти только одну уязвимость.

Загрузка «полезного груза»

Как правило, атака проводится не только ради самого процесса взлома. Обычно злоумышленник хочет иметь возможность возврата во взломанную систему или использования её в качестве плацдарма для атаки других систем. Кроме того его может интересовать получение конфиденциальных данных во взломанной системе. К действию по загрузке «полезного кода» можно отнести:

- ✓ установка снифферов на взломанную систему для получения информации, позволяющей осуществить взлом соседей;
- ✓ осуществление перенаправления портов, чтобы обеспечить передачу пакетов со взломанной системы на другой компьютер;
- ✓ установка «патчей» на использованную уязвимость или имеющиеся уязвимости, чтобы исключить взлом системы другими злоумышленниками;
- ✓ создание фальшивых учётных записей, наделенных привилегиями суперпользователя;
- ✓ создание или установка готового потайного входа во взломанную систему;
- ✓ установка «троянских коней» для сбора конфиденциальной информации, последующего входа в систему или для проведения атак других систем.

Эти и другие операции проводятся при отключении аудита на взломанной системе.

Сокрытие следов взлома

На данном этапе используются программы удаления записей из различных журналов, ведущихся в системе, скрытие установленных файлов, использование потоков файлов в файловой системе NTFS, троянизирование системных программ и утилит и замена программных компонент операционной системы.

Для реализации этих целей используются специальные наборы средств (root kit). Первые наборы таких средств были разработаны в начале 90-х годов для ОС Unix. В настоящее время они существуют практически для всех операционных систем. Первые такие наборы представляли собой троянские файлы, в которые были встроены потайные ходы. Они предназначались для подмены наиболее часто используемых программ типа netstat.

Различают два основных вида таких наборов средств: наборы уровня ядра и наборы уровня пользователя. Это различие вызвано теми компонентами программного обеспечения, которые перезаписываются (изменяются) в атакованной системе. Чтобы использовать набор средств, атакующий должен сначала получить права администратора в атакованной системе.

Наибольшую опасность представляют наборы средств уровня ядра. Поскольку они имеют высшие права, то могут быть полностью скрыты от другого ПО, запущенного в системе. Как правило, в набор входят файлы, которые заменяют программные модули в ядре опе-

рационной системы или отдельные библиотеки, различные вспомогательные средства для получения необходимой информации, а также скрипты инсталляции. Одной из главных задач атакующего является обеспечение выживаемости установленных средств после перезагрузки и обеспечение их «невидимости» для пользователя или администратора.

2. Классификация атак

В общем случае существует поток данных от источника информации к получателю. Нормальный поток отражен на рисунке 1.1а. Остальные части рисунка показывают четыре общие категории атак:

- *Прерывание*: передаваемая информация уничтожается или становится недоступной. Этот тип атак направлен на доступность информации. Примерами могут служить уничтожение части аппаратного обеспечения, обрыв линии или блокировка файловой системы.
- *Перехват*: неавторизованная сторона получает доступ к информации. Этот тип атак направлен на конфиденциальность информации. Неавторизованной стороной может быть человек или программа. Примерами могут служить перехват сообщений в сети и неавторизованное копирование файлов или программ.
- *Изменение*: неавторизованная сторона не только получает доступ, но и фальсифицирует информацию. Этот тип атак направлен на целостность информации. Примерами могут служить изменение данных в файле, изменение кода программы и модифицирование содержимого сообщения, передаваемого по сети.
- *Фальсификация*: неавторизованная сторона вставляет в поток информации поддельный объект. Этот тип атак направлен на подлинность информации. Примерами могут служить вставка случайных сообщений в сетевой поток информации или добавление записей в файл.

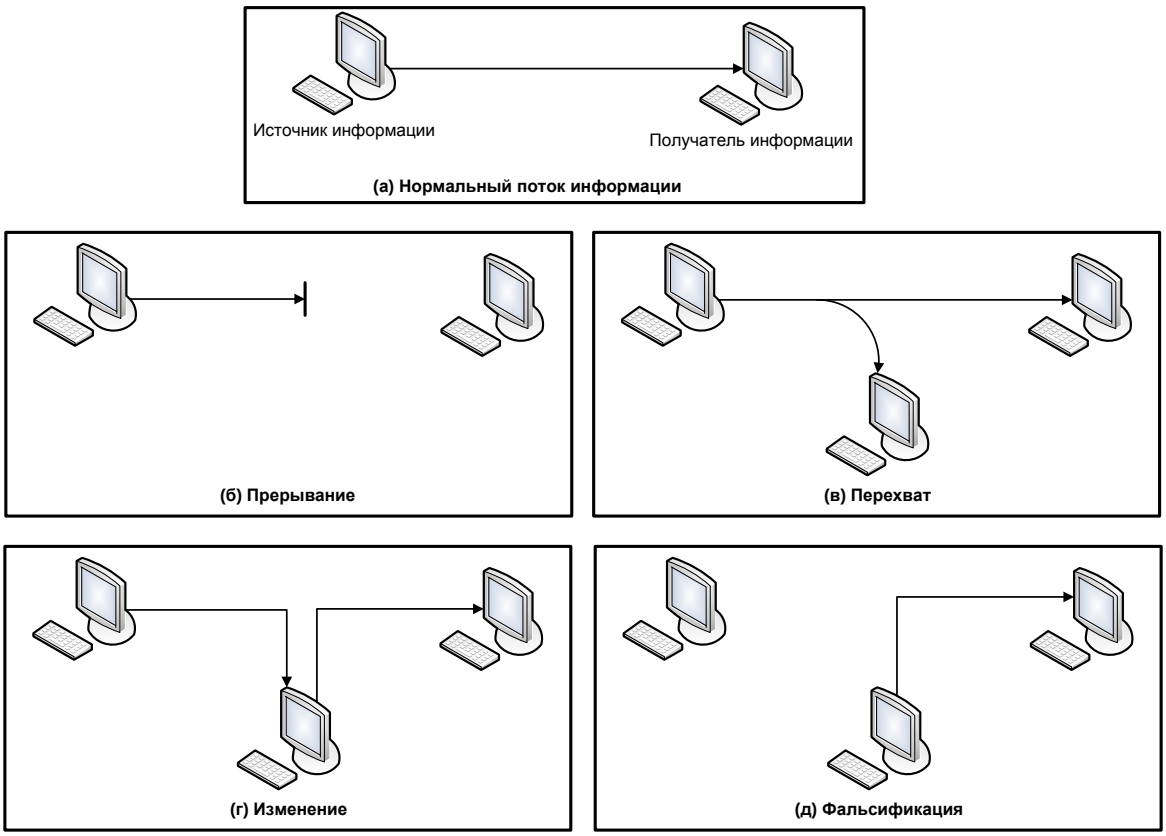


Рисунок 1.1. Классификация атак по Столингсу.

Также атаки делятся на пассивные и активные (рисунок 1.2).

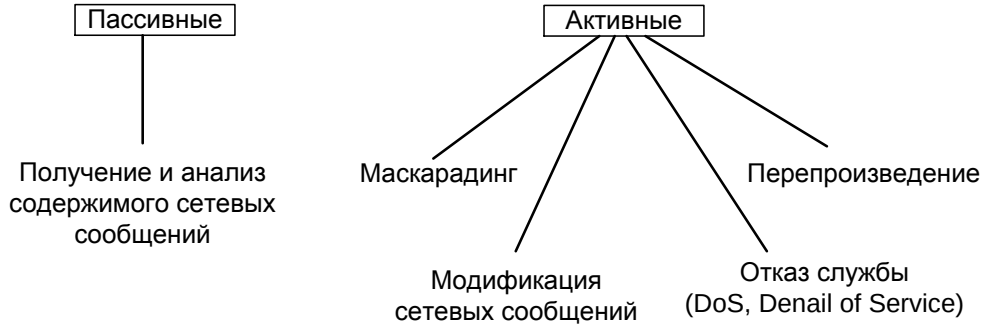


Рис. 1.2. Активные и пассивные атаки.

Пассивные атаки

Пассивные атаки по своей природе схожи с подслушиванием или слежкой. Целью оппонента является получение передаваемой информации. Существует два типа пассивных атак: получение содержимого сетевых сообщений и анализ сетевого трафика. Первый тип легок для понимания. Телефонная беседа, электронная почта, передаваемый файл и прочее могут содержать важную информацию.

Второй тип более тонок для понимания. Предположим, что мы можем маскировать содержимое сетевых сообщений таким образом, что даже если оппонент перехватит их, то все равно не сможет извлечь информацию из сообщения. Самый простой способ маскировки сообщений – это их шифрование. Тем не менее даже при наличии шифровки оппонент все равно в состоянии осуществить анализ перехваченного сообщения. Пассивные атаки очень трудны для обнаружения, так как они не изменяют содержимого сообщений. Тем не менее вполне реально предотвратить данный тип атак.

Активные атаки

Данные атаки производят модификацию потока данных или создают фальшивый поток. Делятся на четыре типа.

Маскарадинг имеет место когда одна сущность выдает себя за другую. Маскарадинг зачастую включает в себя какой-либо другой тип активных атак. Например, пакеты для прохождения процедуры аутентификации могут перехвачены и перепроизведены после того, как процедура аутентификации настоящего пользователя была успешно осуществлена. Таким образом, это даст злоумышленнику вход в систему под определенной учетной записью.

Перевоспроизведение включает пассивный перехват данных и их дальнейшую ретрансляцию. Модификация сообщений говорит само за себя.

Отказ службы (DoS) приводит к ненормальному функционированию какой-либо сетевой службы. Данный тип атак может также к неработоспособности всей сети в целом путем перегрузки ее сетевым «мусором».

II. Модель сетевой безопасности

Обобщенную модель сетевой безопасности можно представить в следующем виде (рисунок 2.1).

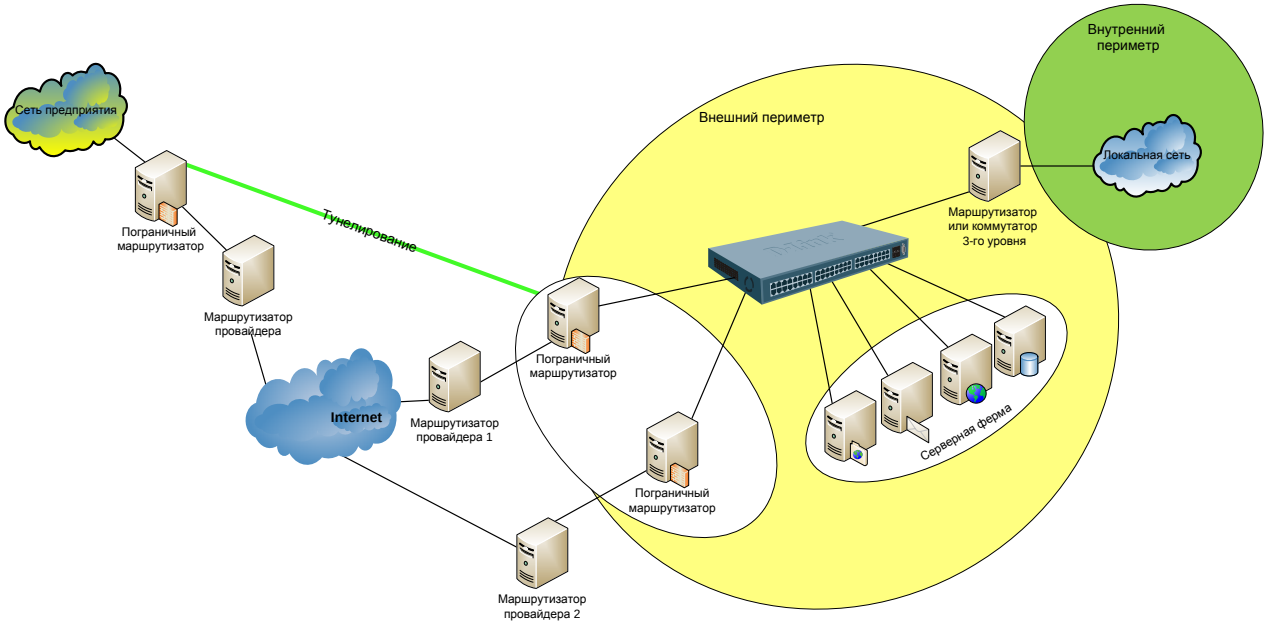


Рисунок 2.1. Обобщенная архитектура сетевой безопасной сети.

Грамотно построенная сеть (с точки зрения безопасности) должна содержать как минимум два периметра:

- ✓ Внутренний. В данном периметре располагаются только компьютеры пользователей, которые входят в сеть незарегистрированных IP-адресов. Также данная сеть может содержать беспроводные точки доступа для подключения мобильных пользователей. Во внутреннем периметре применяются механизмы внутрисетевой безопасности.
- ✓ Внешний периметр. Обычно содержит сервера предприятия (серверная ферма) и пограничные маршрутизаторы, обеспечивающие доступ во внешние сети. Все компьютеры во внешнем периметре обычно включаются в IP-сеть зарегистрированных адресов. В данном периметре применяются механизмы межсетевой безопасности, к которым относятся:
  - межсетевые экраны (МЭ);
  - системы трансляции адресов и портов;
  - системы обнаружения атак и вторжений (СОАВ);
  - системы туннелирования.

Между периметрами устанавливается маршрутизатор или коммутатор L3, на котором также могут применяться технологии межсетевой безопасности. Все маршрутизаторы могут как программным, так и аппаратными.

Данная концепция является расширяемой. Если необходимо повысить уровень безопасности сети, то можно увеличить количество внутренних периметров. Следующим этапом развития модели является добавление ещё одного внутреннего периметра. В этом случае во внешнем периметре устанавливаются сервера-ловушки, в первом внутреннем периметре – сервера предприятия и только во втором внутреннем периметре – компьютеры пользователей.

III. Криптография и системы шифрования

1. Стандартное шифрование

Стандартное шифрование (симметричное шифрование, шифрование с использованием одного ключа) было единственным используемым типом шифрования до изобретения шифрования на основе открытого ключа в конце 1970-х годов. До сих пор стандартное шифрование остается широко используемым.

Алгоритм работы системы стандартного шифрования состоит из пяти шагов (рисунок 3.1):

1. *Открытый текст* (в дальнейшем для простоты просто «текст»): это оригинальное сообщение или данные, которые являются входными для алгоритма;
2. *Алгоритм шифрования*: это алгоритм, который выполняет различные подстановки и изменения над исходным текстом;
3. *Секретный ключ*: также является входным элементом для алгоритма. Конкретные подстановки и изменения, выполняемые алгоритмом шифрования, зависят от ключа;
4. *Шифр-текст*: это зашифрованное сообщение, получаемое на выходе алгоритма. Его вид зависит от исходного текста и секретного ключа. Для одного и того же исходного сообщения два различных секретных ключа на выходе алгоритма дадут два различных шифр-текста;
5. *Алгоритм дешифрования*: это алгоритм шифрования, работающий в обратном порядке. В качестве входных данных он берет шифр-текст и такой же секретный ключ и производит исходный текст.

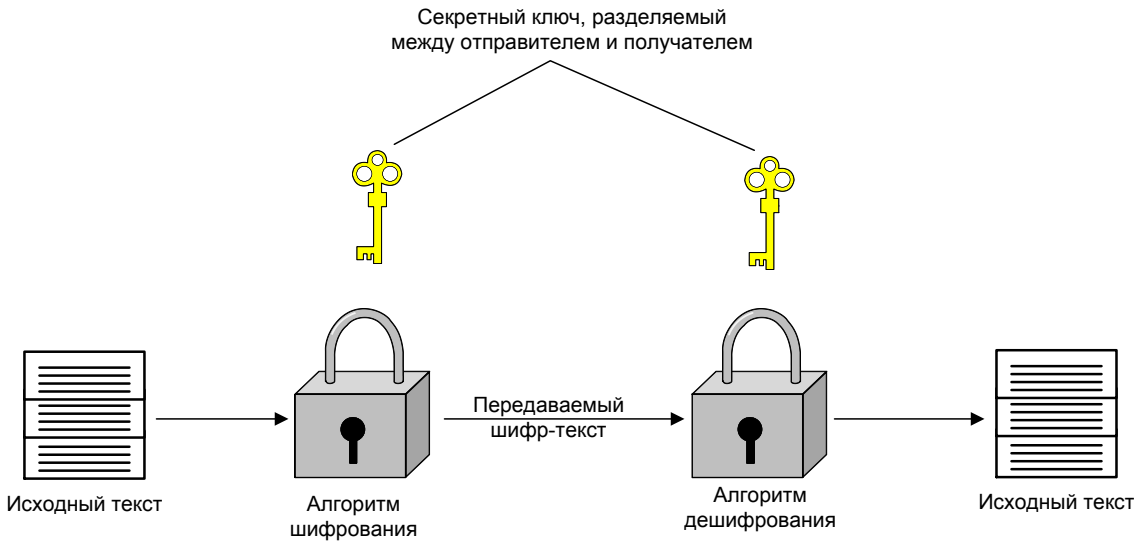


Рисунок 3.1. Упрощенная модель работы алгоритма стандартного шифрования.

Существует два требования для безопасного использования стандартного шифрования:

1. Необходим мощный алгоритм шифрования. Как минимум необходимо, чтобы алгоритм был таким, что злоумышленник, знающий этот алгоритм и имеющий доступ к одному или более шифр-текстам, не смог бы их расшифровать или вычислить ключ. Данное требование обычно звучит в следующей, более строгой форме: злоумышленник не должен расшифровать шифр-текст или обнаружить ключ даже в том случае, если он владеет несколькими шифр-текстами вместе с исходными текстами, из которых они были получены.
2. Отправитель и получатель должны получать копии секретного ключа и хранить его *безопасным образом*. Если кто-либо узнает ключ и знает алгоритм, то все данные, шифруемые с использованием этого ключа, станут читабельны.

Важно заметить, что безопасность стандартного шифрования зависит от секретности ключа, а не от секретности самого алгоритма шифрования. То есть подразумевается, что задача расшифровки сообщения на базе шифр-текста и знания алгоритма шифрования/дешифрования является практически невозможной. Другими словами, нет необходимости хранить в секретности сам алгоритм; необходимо хранить в секретности только ключ.

1.1. Криптография

Криптографические системы в общем разделяются на три независимых класса:

1. *Тип операций, используемых для преобразования исходного текста в шифр-текста*. Все алгоритмы шифрования базируются на двух основных принципах: подстановка, при которой каждый элемент в исходном тексте (бит, буква, группа бит или букв) представляются другим элементом, и перестановка, при которой элементы исходного текста меняются местами. Основное требование – не допустить потери информации, чтобы можно было в дальнейшем осуществить процесс расшифровки. Большинство систем включает не один, а несколько последовательных этапов замены и перестановки.
2. *Количество используемых ключей*. Если отправитель и получатель используют один и тот же ключ, система называется симметричной (с одним ключом, или с секретным ключом, или система стандартного шифрования). Если отправитель и получатель используют разные ключи, система называется ассиметричной (с двумя ключами или система шифрования на основе открытого ключа).
3. *Способ обработки исходного текста*. Блочный шифровщик обрабатывает на входе системы один блок элементов за раз, производя блок элементов на выходе для каждого входного блока. Поточный шифровщик обрабатывает входные элементы последовательно, производя на выходе один элемент за раз.

Криптоанализ

Процесс попытки раскрыть исходный текст или ключ называется криптоанализом. Стратегия, используемая шифроаналитиком, зависит от природы схемы шифрования и информации, доступной для шифроаналитика.

Таблица 3.1 – Типы атак на зашифрованные сообщения

Тип атаки	**Известно шифроаналитику**
Только шифр-текст	▪ Алгоритм шифрования ▪ Шифр-текст, который необходимо расшифровать
Известный исходный текст	▪ Алгоритм шифрования ▪ Шифр-текст, который необходимо расшифровать ▪ Один или более пар «исходный текст» – «шифр-текст»
Выбранный исходный текст	▪ Алгоритм шифрования ▪ Шифр-текст, который необходимо расшифровать ▪ Текстовое сообщение, выбранное шифроаналитиком совместно с соответствующим шифр-текстом.
Выбранный шифр-текст	▪ Алгоритм шифрования

	▪ Шифр-текст, который необходимо расшифровать ▪ Предполагаемый шифр-текст, выбранный шифроаналитиком, совместно с соответствующим дешифрованным исходным текстом
Выбранный текст	▪ Алгоритм шифрования ▪ Шифр-текст, который необходимо расшифровать ▪ Сообщение исходного текста, выбранное шифроаналитиком, совместно с соответствующим шифр-текстом ▪ Предполагаемый шифр-текст, выбранный шифроаналитиком, совместно с соответствующим дешифрованным исходным текстом

Таблица 3.1 показывает различные типы криптоаналитических атак, основанных на количестве информации, известной шифроаналитику. Наиболее сложная проблема – это, когда вся известная информация представлена только шифр-текстом. Тем не менее в большинстве случаев алгоритм шифрования все же известен. Одной из возможных атак в этом случае может быть «Брут-форс» атака («грубая сила»), которая заключается в переборе всех возможных ключей. Если множество ключей слишком большое, подобная задача становится практически невозможной. Таким образом, злоумышленник должен полагаться на анализ самого шифр-текста, применяя к нему различные статистические тесты. Для использования этого метода злоумышленник должен предполагать тип данных, содержащихся в исходном тексте (например, английский или французский текст, исполняемый файл какой-либо ОС, java-скрипт и т.д.).

Атаке под названием «только шифр-текст» легче всего противостоять, так как злоумышленник обладает минимальным количеством информации для своей работы. Тем не менее в большинстве случаев шифроаналитик обладает большим количеством информации. Аналитик может перехватить одно или более сообщений с исходным текстом также, как и их шифровки. Или же аналитик может знать, что конкретный кусок исходного появится в сообщении. Например, файл, представляемый в формате Postscript, всегда начинается с одного и того же сочетания символов. Все вышеприведенные случаи относятся к типу атак под названием «известный исходный текст». Зная эту информации, аналитик может алгоритмически вывести значение секретного ключа.

Тесно связана с атакой «известный исходный текст» атака под названием «вероятное слово». Если злоумышленник работает с шифровкой какого-либо типичного сообщения, то он может догадываться о содержимом части этого сообщения. Примером такого сообщения может служить исходный код программы, разработанный корпорацией, и этот исходный код может содержать копирайт в какой-то стандартной позиции.

Если аналитик в состоянии получить доступ к системе шифрования, чтобы затем подсунуть ей какой-либо сообщение, то подобный тип атак называется атака «выбранный исходный текст». В общем случае, если аналитик в состоянии выбрать сообщение для шифровки, то тогда аналитик может выборочно подбирать образцы текста для последующего анализа их шифровок с целью раскрытия секретного ключа.

Таблица 3.1 содержит два других типа атак: «выбранный шифр-текст» и «выбранный текст». Он возможны чисто теоретически, но не практически. Атаки с использованием только шифр-текста не сможет противостоять только очень слабый алгоритм шифрования. В общем случае алгоритмы шифрования разрабатываются таким образом, чтобы противостоять как минимум атакам «известный исходный текст».

Необходимо рассмотреть еще два определения. Схема шифрования является безопасной с точки зрения вычислений, если шифр-текст, сгенерированный этой схемой отвечает одному или более условиям:

- ✓ стоимость взлома шифра превышает ценность зашифрованной информации;
- ✓ время взлома превышает полезное время жизни информации.

Проблем заключается в том, что очень трудно вычислить количество попыток, требуемых для удачного взлома шифра. Тем не менее, если предположить, что алгоритм шифрования не имеет никаких математических изъянов и, следовательно, для его взлома применима только брут-форс атака, то тогда мы можем сделать некоторые предположения относительно времени и стоимости взлома.

Суть брут-форс атаки заключается в том, что при взломе шифра перебираются все возможные ключи до тех пор, пока не будет получен нужный результат. В среднем необходимо перебрать половину ключей. Таблица 3.2 показывает, как много времени требуется для перебора ключей различной длины.

Таблица 3.2 – Среднее время, требуемое для полного перебора всего множества ключей

Размер ключа (бит)	**Количество ключей**	**Затрачиваемое время при скорости 1 ключ/мкс**	**Затрачиваемое время при скорости 10^6 ключей/мкс**
32	$2^{32} = 4,3 * 10^9$	2^{31} мкс = 35,8 минут	2,15 миллисекунд
56	$2^{56} = 7,2 * 10^{16}$	2^{55} мкс = 1142 лет	10 часов
128	$2^{128} = 3,4 * 10^{38}$	2^{127} мкс = $5,4 * 10^{24}$ лет	$5,4 * 10^{18}$ лет
168	$2^{168} = 3,7 * 10^{50}$	2^{167} мкс = $5,9 * 10^{36}$ лет	$5,4 * 10^{30}$ лет

1.2. Структура шифрования Фейстеля

Все алгоритмы стандартного блочного шифрования имеют структуру, впервые описанную сотрудником IBM Хорстом Фейстелем в 1973 году. Данная структура показана на рисунке 3.2.

Входными данными для алгоритма шифрования являются блок текста длинной 2w бит и ключ K. Блок текста разбивается на две половины L_0 и R_0. Эти две половинки данных проходят обработку в течении n кругов (раундов), а затем объединяются для получения блока шифр-текста. Каждый i-ый раунд в качестве входных данных берет L_{i-1} и R_{i-1}, полученное с выхода предыдущего раунда, и ключ K_i, полученный из общего ключа K. В общем случае подключи K_i отличаются от K и друг от друга. Подключи K_i генерируются из ключа K алгоритмом генерации подключей.

Все раунды имеют одну и ту же структуру. Подстановка выполняется над левой половиной данных. Это процедура осуществляется следующим образом:

- ✓ выполняется функция F над правой половиной данных;

- ✓ выполняется побитовая операция исключающего ИЛИ (XOR) над выходными данными функции F и левой половиной данных.

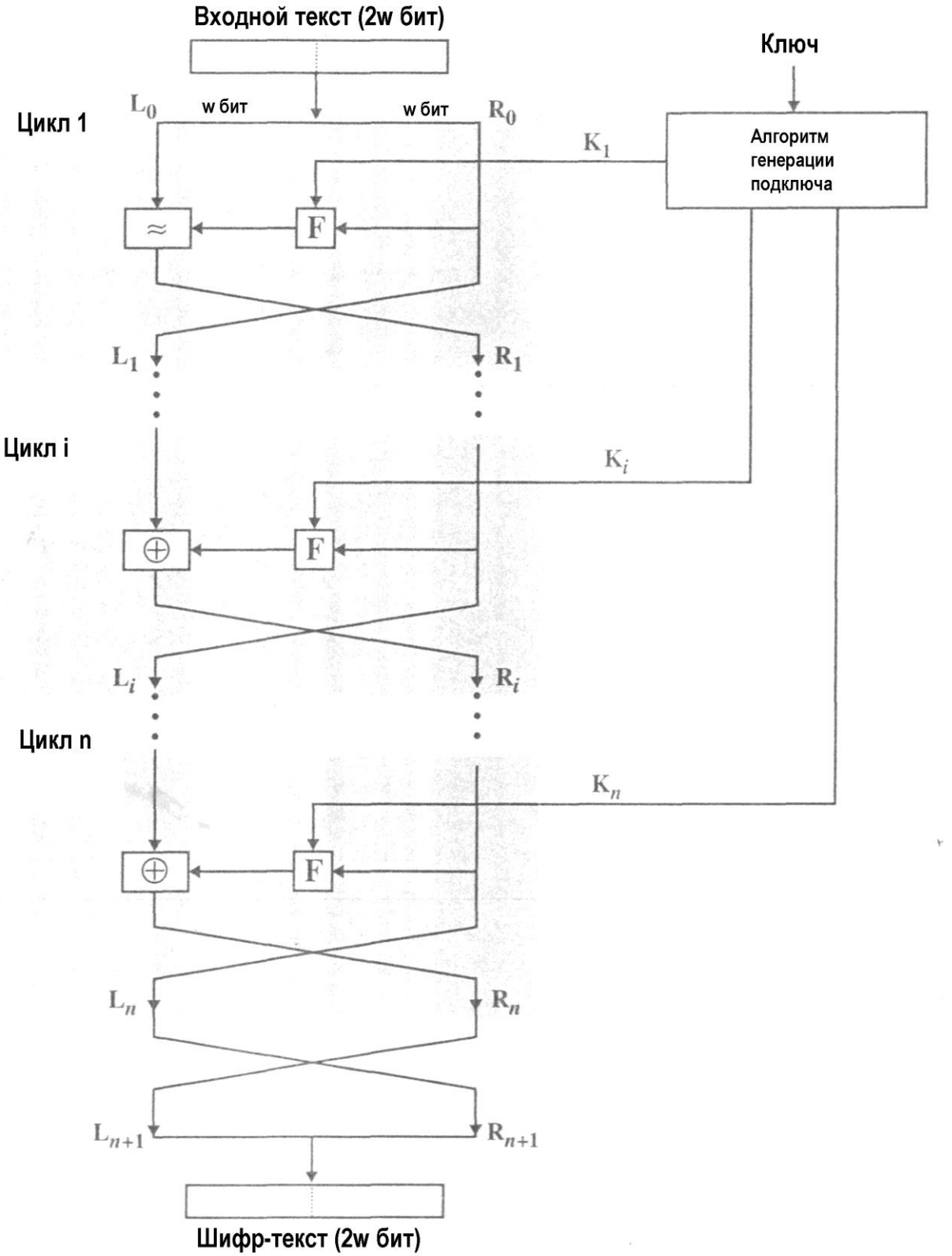


Рисунок 3.2. Классическая сеть Фейстеля.

Функция F имеет одну и ту же структуру для всех раундов, но разные входные подключи K_i. В конце каждого раунда происходит перестановка левой и правой половин данных. Конкретная реализация сети Фейстеля зависит от выбора следующих параметров:

- ✓ **размер блока данных.** Чем больше блок, тем больше безопасность и ниже скорость обработки. Стандартным и рекомендуемым считается блок в 64 бита.
- ✓ **размер ключа.** Чем больше ключ, тем больше безопасность и ниже скорость обработки. Наиболее распространенный размер ключа в современных алгоритмах – 128 бит.
- ✓ **количество раундов.** Один раунд не обеспечивает должной безопасности. Чем больше раундов, тем выше безопасность. Стандартным является 16 раундов.
- ✓ **функция F.** Чем сложнее функция, тем сложнее алгоритм шифрования более устойчив ко взлому.

Есть также два других соображения по поводу архитектуры сети Фейстеля:

- ✓ **скорость ПО шифрования/дешифрования.** В большинстве случаев шифрование алгоритм шифрования включается в ПО, дабы исключить его аппаратную реализацию. Соответственно скорость выполнения алгоритма становится важным аспектом.
- ✓ **легкость анализа.** Несмотря на то, что нам хотелось бы сделать наш алгоритм как можно более сложным для криптоанализа, существует некоторые плюсы, если алгоритм легко анализировать. То есть, если алгоритм может быть легко и кратко объяснен, в нем легче отыскать уязвимости и тем самым повысить уровень его защищенности.

Процесс дешифрования на основе сети Фейстеля в основном схож с процессом шифрования. В качестве входных данных используются блок шифр-текста и подключи K_i, но в обратном порядке. То есть необходимо использовать K_n в первом раунде и K_1 в последнем раунде. Это очень хорошо, так как нам не нужно использовать различные алгоритм для шифрования и дешифрования.

1.3. Алгоритмы стандартного шифрования

Наиболее распространенные алгоритмы стандартного шифрования являются блочными. Наиболее важными из них являются стандарт DES (Data Encryption Standard, Стандарт Шифрования Данных) и алгоритм TDEA (Triple Data Encryption Algorithm, Алгоритм Тройного Шифрования Данных).

Стандарт DES

Наиболее широко используемой схемой шифрования является стандарт DES, принятый в 1977 Национальным Бюро Стандартов США (сейчас Национальный Институт Стандартов и Технологий) в качестве Федерального Стандарта Обработки Информации. В 1994 году НИСТ утвердил DES для федерального использования в течении следующих пяти лет. Алгоритм, лежащий в основе стандарта, называется DEA.

Описание алгоритма

Общая схема шифрования DES представлена на рисунке 3.3.

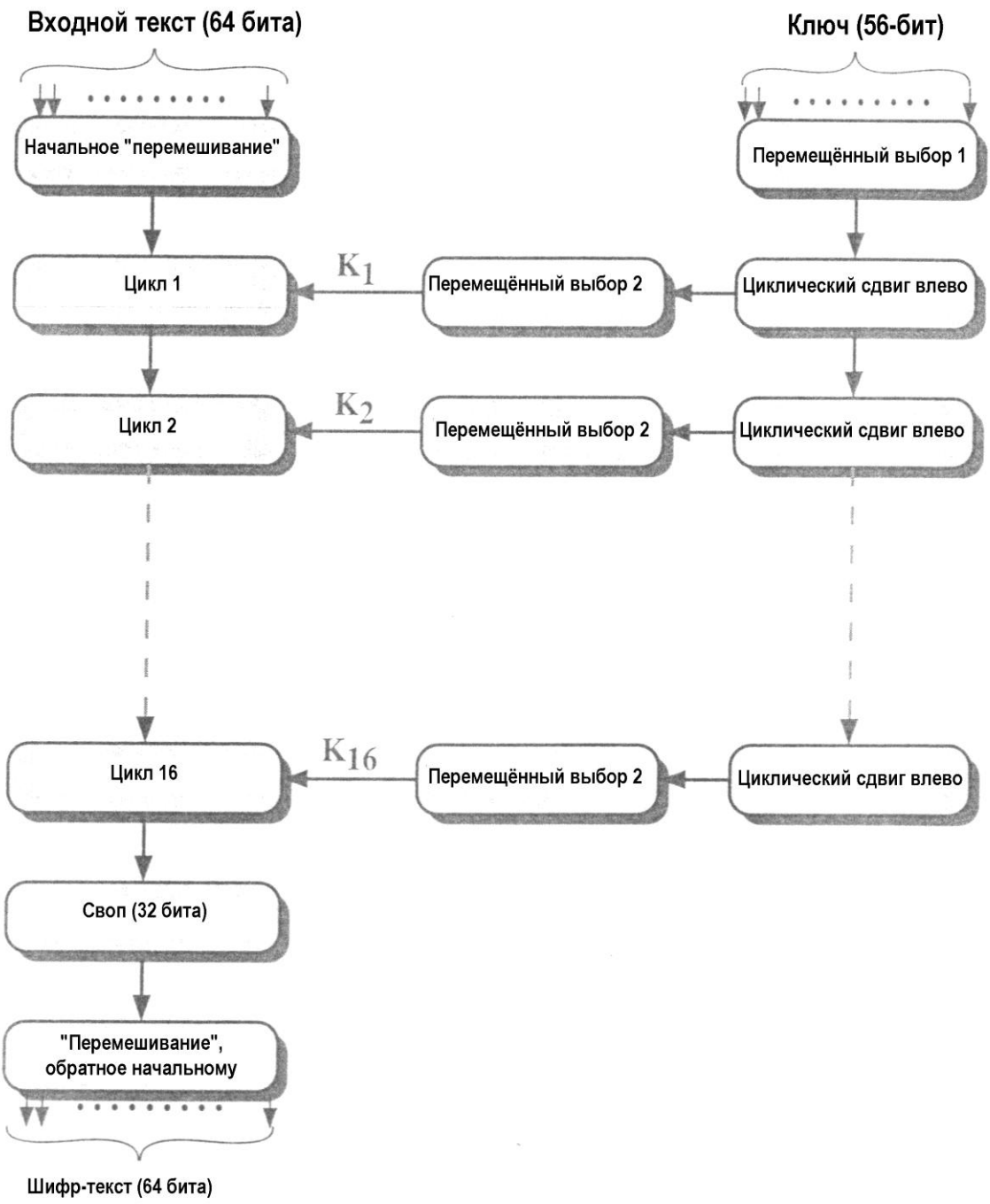


Рисунок 3.3. Общее представление алгоритма шифрования DES.

Исходный текст разбивается на блоки в 64 бита, а ключ имеет размер 56 бит. Левая часть рисунка показывает, что обработка текста происходит в три фазы. В первую очередь, 64-битный текст проходит начальную перестановку (НП), которая меняет местами биты. Затем следует фаза, состоящая из 16 итераций одной и той же функции. На выходе последней итерации производится перестановка местами левой и правой половин 64-битного текста. В конце производится конечная перестановка (КП), которая является инверсной относительно функции НП и выдает 64-битный шифр-текст.

Правая часть рисунка 3.3 показывает способ использования 56-бтового ключа. Изначально ключ проходит через процедуру перестановки бит. Далее для каждой из 16 итераций генерируется подключ K_i путем циклического сдвига влево, а затем перестановки бит.

Функция перестановки одна и та же для каждой итерации, но при этом генерируются различные подключи из-за того, что на каждом этапе производится циклический сдвиг влево.

Рисунок 3.4 более детально представляет алгоритм для единичной итерации. На входе каждой итерации 64-битовой текст разделяется на две 32-битовые половины, которые помечаются как L и R соответственно. Весь процесс для каждой итерации может быть выражен через следующие формулы:

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$

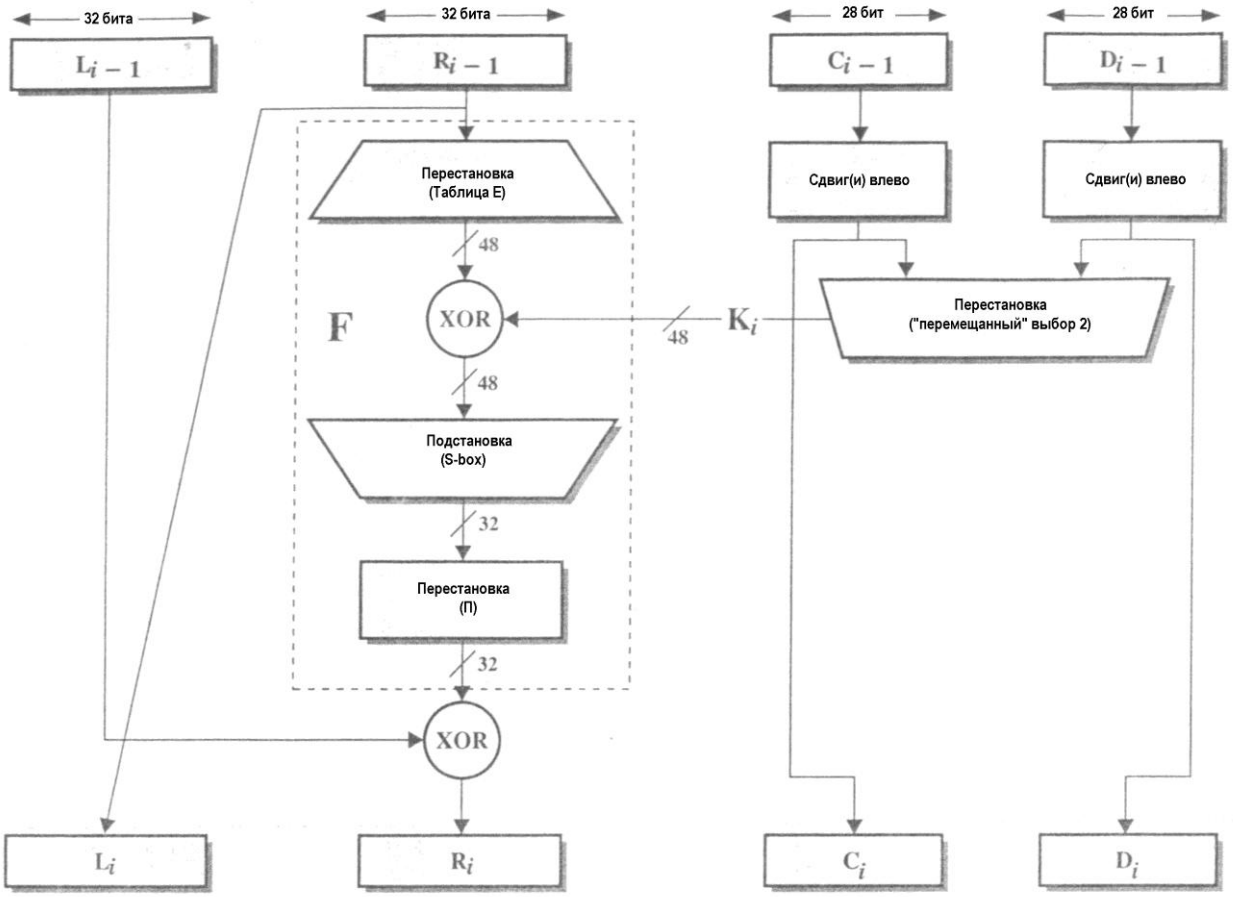


Рисунок 3.4. Алгоритм одной итерации DES.

Таким образом, левая часть данных на выходе итерации является идентичной правой части данных на входе итерации. Правая часть данных на выходе итерации является результатом операции побитового исключающего ИЛИ, произведенной над L_{i-1} и результатом комплексной функции F. Данная функция включает как операции перестановки, так и подстановки. Операция подстановки, представленная на блок-схеме вершиной «S-box», просто отображает комбинацию из 48 входных бит в соответствующий паттерн из 32 бит.

Возвращаясь к рисунку 3.3, видно, что 56-битовый ключ сначала проходит процедуру перестановки. Полученный 56-битовый ключ далее разделяется на две половины по 28 бит каждая, которые маркируются как C_0 и D_0. На каждой итерации половины C и D отдельно подвергаются операции циклического сдвига влево на 1 или 2 бита. Полученные путем сдвига величины служат входными половинами для следующей итерации, а также для функции перестановки.

Процесс дешифровки DES в основном тот же самый, что и процесс шифровки. При этом действуют следующие правила: использовать шифр-текст в качестве входных данных, и ключи K_i в обратном порядке.

Устойчивость DES

Споры относительно устойчивости алгоритма DES делятся на две области: относительно собственно самого алгоритма и относительно использования 56-битового ключа. Многие годы осуществлялось большое количество попыток найти и использовать слабые стороны этого алгоритма, что сделало DES наиболее изученным алгоритмом. Несмотря на это никто так и не нашел слабых мест в алгоритме DES, во всяком случае не опубликовал.
Более серьезным спором является проблемы длины ключа. Еще в начале 70-х годов эксперты предупреждали, что в плане безопасности дни алгоритма DES сочтены, это лишь вопрос времени роста скорости работы процессов и их стоимости. Это время наступило в июле 1998 года, когда фонд «Электронная Граница» объявил, что он взломал алгоритм DES, используя специальный компьютер «Взломщик DES», стоимость на создание которой оказались меньше $250000. Атака длилась менее трех дней. Фонд опубликовал детальное описание машины, тем самым предоставив другим людям возможность построить собственные взломщики. Цены на аппаратное обеспечение падают, а производительность процессоров растет, делая DES все более слабым.

Важно отметить, что для взлома была применена не чистая брут-форс атака – осуществлялся поиск нужного ключа, а не простой перебор всех возможных ключей. Для удачного взлома аналитик должен распознать исходный текст. Если сообщение представляет собой просто текст на каком-либо заранее известном языке, то процесс распознания можно автоматизировать. Если сообщение было сжато, распознание становится более сложным. Если же сообщение представлено каким-либо типом данных, затем сжато, проблема распознания и автоматизации становится довольно сложной. Таким образом, для успешного взлома необходимо предположение о типе исходного текста и некоторое средство его автоматического распознания. Фонд решил данную проблему и предложил несколько техник автоматического распознания, которые могут быть полезны во многих случаях.

Если бы единственной формой атаки была брут-форс атака, тогда способ противостояния данной атаки очевиден – использование более длинного ключа. При этом необходимо подумать о приемлемой длине ключа. Для этого воспользуемся взломщиком Фонда Электронная Граница. Данный взломщик был прототипом и следует предполагать, что на сегодняшний день возможно создать более быструю машину. Предположим, что взломщик может перебирать один миллион ключей за одну микросекунду, тогда алгоритм DES будет взломан не более, чем за 10 часов. Это примерно в 7 раз быстрее, чем время, затраченное Фондом. Используя эту скорость для взлома DES-подобного алгоритма с 128-битовым ключом понадобится примерно 10^{18} лет. Таким образом, 128-битовый ключ является гарантом, что алгоритм не будет взломан.

Triple DEA

Стандарт Triple DEA (TDEA) был впервые предложен Тучманом и впервые стандартизован в виде ANSI стандарта X9.17 в 1985 году для использования в финансовых приложениях. TDEA использует три ключа и три выполнения алгоритма DES: шифрование с первым ключом, дешифрование со вторым ключом, шифрование с третьим ключом:

$$C = E_{K3}[\,D_{K2}[\,E_{K1}[P]\,]\,]$$

где

C – шифр-текст
P – исходный текст
$E_K[X]$ – шифрование X с использованием ключа K
$D_K[Y]$ – дешифрование Y с использованием ключа K

Функция дешифрование является обратной относительно функции шифрования (рисунок 3.5).

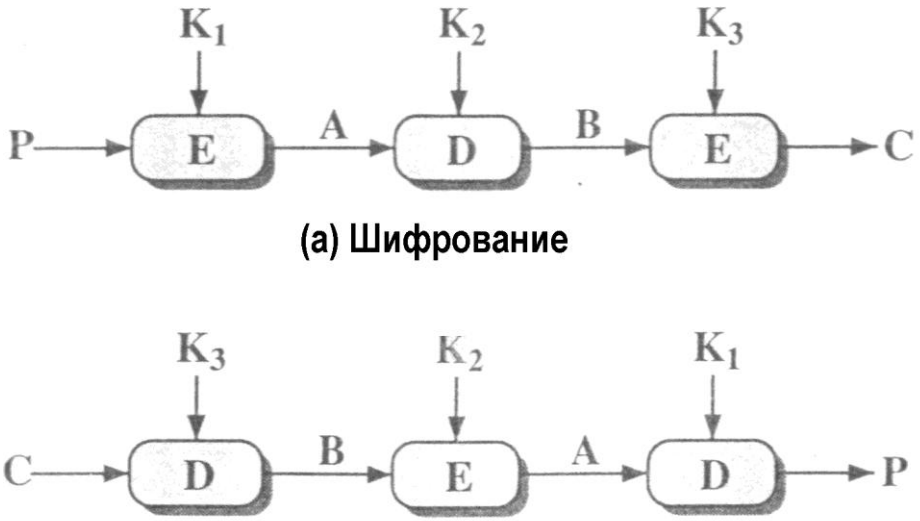


Рисунок 3.5. Triple DEA.

Особой криптографической значимости в том, что на второй стадии производится дешифрация, нет. Сделано это с целью совместимости с предыдущим алгоритмом DES, дабы пользователи TDEA могли дешифровать данные, закодированные DES:

$$C = E_{K1}[D_{K1}[E_{K1}[P]]] = E_{K1}[P]$$

При использовании трех ключей каждый из них имеет длину в 168 бит. Стандарт также позволяет использование двух ключей, при этом K1 = K3 и длина ключа равна 112 бит. Легко заметить, что TDEA является серьёзным алгоритмом. Из-за того, что в основе его лежит алгоритм DES, TDEA также устойчив ко взлому как и DES. А использование 168-битовых ключей делает атаки практически бессмысленными. Тем не менее, алгоритм является громоздким.

Стандарт Улучшенного Шифрования AES (Advanced Encryption Standard)

TDEA обладает двумя плюсами, которые гарантируют его широкое использование в течении следующих нескольких лет. Во-первых, длина ключа в 168 бит гарантирует устойчивость против брут-форс атак. Во-вторых, в основе TDEA лежит стандарт DEA. Данный алгоритм исследовался тщательнее любого другого в течении долгого периода времени и не было найдено ни одной эффективной криптоаналитической атаки (кроме брут-форс) против него. Соответственно и алгоритм TDEA очень устойчив к криптоаналитическим атакам. Если бы единственным фактором для выбора алгоритма являлась безопасность, тогда TDEA был бы приемлемым вариантом для ближайших десятилетий.

Принципиальным недостатком TDEA является то, что он очень громоздкий. Вторым недостатком является то, что как и DEA алгоритм TDEA использует блок размером в 64 би-

та. Как для безопасности, так и для производительности блок больших размеров более предпочтителен.

В качестве замены в 1997 году NIST выдвинул предложение для нового стандарта шифрования AES, который по устойчивости должен быть равен или лучше TDEA, но гораздо производительней его. В дополнении к данным основным требованиям NIST указал, что AES должен быть симметричным алгоритмом блочного шифрования с размером блока в 128 бит, а также должен поддерживать ключи длинной 128, 192 и 256 бит.

Другие симметричные алгоритмы блочного шифрования

Все современные алгоритмы симметричного блочного шифрования используют базовую структуру Фейстеля. Причина этого заключается в том, что структура давно хорошо изучена и это позволяет более легко определить криптографическую устойчивость нового алгоритма. Если бы в каждом новом случае использовалась новая структура, то она могла бы иметь уязвимости, не очевидные для разработчика. Сравнительные характеристики рассматриваемых алгоритмов приведены в таблице 3.3.

Таблица 3.3 – Алгоритмы стандартного шифрования

Алгоритм	Размер ключа, бит	Количество раундов	Математические операции	Приложения
DES	56	16	XOR, fixed S-boxes	SET, Kerberos
Triple DES	112, 168	48	XOR, fixed S-boxes	Финансовые, PGP, S/MIME
IDEA	128	8	XOR, сложение, умножение	PGP
Blowfish	До 448	16	XOR, variable S-boxes, сложение	
RC5	До 2048	До 255	Сложение, вычитание, XOR, ротация	
CAST-128	От 40 до 128	16	Сложение, вычитание, XOR, ротация, fixed S-boxes	PGP

IDEA

Международный Алгоритм Шифрования Данных (IDEA, International Data Encryption Algorithm) является алгоритмом блочного симметричного шифрования, разработанным Ксуеджия Лаи (Xuejia Lai) и Джеймсом Мэссей (James Massey) в Швейцарском федеральном институте технологий в 1991 году. IDEA использует 128-битовый ключ. IDEA заметно отличается от DES как функцией раунда, так и функцией генерации подключей. Для функции раунда IDEA не использует «S-box». Более того, IDEA основывается на трех различных математических операциях: исключающее ИЛИ, двоичное сложение 16-битовых целых и двоичное умножение 16-битовых целых. Эти функции комбинируются таким образом, чтобы полученный в результате шифр-текст было очень сложно анализировать. Алгоритм генерация подключей основывается исключительно на использовании циклических сдвигов, но использует их комплексным способом, чтобы сгенерировать сразу шесть подключей для каждого из восьми раундов алгоритма IDEA. Так как IDEA был одним из первых предложенных 128-битовых альтернатив алгоритму DES, то он подвергся сильному изучению и оказалось, что он очень стоек к криптоанализу. IDEA использует-

ся в PGP (как одна из альтернатив), а также используется в ряде коммерческих продуктов.

Blowfish

Blowfish был разработан в 1993 году независимым консультантом и криптологом Брюсом Шнейером (Bruce Schneier) и быстро стал одной из наиболее популярных альтернатив DES. Blowfish разрабатывался так, чтобы его было легко внедрить в ПО и чтобы он имел высокую скорость выполнения. Он также является очень компактным алгоритмом, что позволяет ему выполняться на 5 Кб оперативной памяти. Интересная особенность алгоритма Blowfish заключается в том, что длина ключа является переменной и максимально достигает 448 бит. На практике используются 128-битовые ключи. Blowfish использует 16 раундов.

Blowfish использует «S-box» и функцию исключающего ИЛИ, как и DES, но он также использует двоичное сложение. В отличие от DES, который использует фиксированные «S-box», Blowfish применяет динамические «S-box», которые генерируются как функция ключа. В алгоритме Blowfish подключи и «S-box» генерируются путем повторения самого алгоритма Blowfish применительно к исходному ключу. В общей сложности требуется 521 выполнение алгоритма Blowfish для генерации подключей и «S-box». Соответственно Blowfish не подходит для приложений, в которых секретный ключ часто меняется.

Blowfish является одним из наиболее мощных стандартных шифровальных алгоритмов. Обусловлено это тем, что подключи и «S-box» генерируются путем повторения самого алгоритма Blowfish, который тщательно «перемешивает» биты и делает таким образом криптоанализ практически невозможным. Было опубликовано несколько статей по поводу теоретических возможностей криптоанализа Blowfish, но на практике не было найдено ни одной слабости в алгоритме. Blowfish используется в ряде коммерческих приложений.

RC5

RC5 разработан в 1994 году Роном Ривестом, одним из разработчиков алгоритма на основе общих ключей RSA. RC5 определен в RFC-2040 и разрабатывался с учетом следующих положений:

- ✓ применим как для аппаратной, так и программной реализации. RC5 использует только примитивные вычислительные операции, имеющиеся в большинстве микропроцессоров.
- ✓ быстродействие. Для его достижения RC5 строился как простой алгоритм и работает с данными размером в слово.
- ✓ адаптируем к процессорам с различными длинами слов. Количество бит в слове является входным параметром алгоритма RC5.
- ✓ произвольное количество раундов. Количество раундов является вторым входным параметром алгоритма RC5. Этот параметр позволяет выбирать компромисс между скоростью и безопасностью.
- ✓ ключ произвольной длины. Длина ключа является третьим входным параметром алгоритма RC5. И опять же это позволяет выбирать компромисс между скоростью и безопасностью.
- ✓ простота. Простая структура RC5 легка для внедрения и упрощает задачу определения мощности алгоритма.
- ✓ низкие требования к памяти. Эта особенность делает RC5 подходящим для смарт-карт и прочих устройств с ограниченным объемом памяти.
- ✓ высокая безопасность.

- ✓ дата зависимая ротация. Количество ротаций (циклических битовых сдвигов) в RC5 зависит от характера данных.

RC5 используется в ряде продуктов от компании «RSA Data Security Inc».

CAST-128

CAST является основой для разработки симметричных алгоритмов шифрования и разработан в 1997 году Карлислом Адамсом и Стаффордом Таваресом в компании Entrust Technologies. Одним из алгоритмов, разработанных на основе CAST, является CAST-128, описанный в RFC 2144. CAST-128 использует ключи, длинной от 40 до 128 бит с шагом 8. CAST является результатом долгого процесса исследований и разработок. Он используется в ряде продуктов, включая PGP.

CAST-128 использует фиксированные S-boxes, но они гораздо больше тех, что используются в DES. Данные S-boxes очень нелинейны и устойчивы к криптоанализу. Процесс генерации подключей, использующийся в CAST-128, отличен от других процессов, использующихся в алгоритмах стандартного блочного шифрования. Разработчики CAST постарались сделать подключи как можно более устойчивыми к криптоанализу. Другая отличительная особенность CAST-128 заключается в том, что функция раунда меняется в каждом раунде.

1.4. Режимы работы блочных шифровальщиков

Симметричный блочный шифровальщик обрабатывает один блок битов за раз. В случае DEA и TDEA длина блока составляет 64 бита. Если исходный текст оказывается большего размера, то он разбивается на блоки в 64 бита (если необходимо, добавляя последний блок пробелами). Самым простым способом обработки текста в таком случае является режим электронной кодовой книги (electronic codebook, ECB) – исходный текст за раз разбивается на 64-битовые блоки и каждый блок кодируется, используя один и тот же ключ. Термин «кодовая книга» используется потому, что при заданном ключе получается уникальный шифр-текст для каждого 64-битового блока текста. Поэтому можно представить себе гигантскую «кодовую книгу», в которой для каждой возможной комбинации 64 бит существует готовый шифр-текст.

При использовании режима ECB один и тот же 64-битовый блок исходного текста выдает всегда один и тот же шифр-текст. В связи с этим использование режима ECB для длинных сообщений может быть небезопасным. Чтобы устранить данный недостаток режима ECB, необходимо при появлении одного и того же блока исходного текста выдавать различный шифр-текст. Рассмотрим две таковые альтернативы.

Режим цепочки (CBC)

В режиме цепочки (рисунок 3.6) входными данными для алгоритма шифрования является результат операции XOR текущего блока текста и предыдущего блока шифр-текста, при этом используется один и тот же ключ:

$$C_i = E_k[C_{i-1} \oplus P_i]$$

где Ek[X] – шифрование текста X ключом K.

Процесс дешифрования можно представить следующим образом:

$$D_k[C_i] = D_k[E_k(C_{i-1} \oplus P_i)]$$
$$D_k[C_i] = (C_{i-1} \oplus P_i)$$
$$P_i = C_{i-1} \oplus C_{i-1} \oplus P_i = C_{i-1} \oplus D_k[C_i]$$

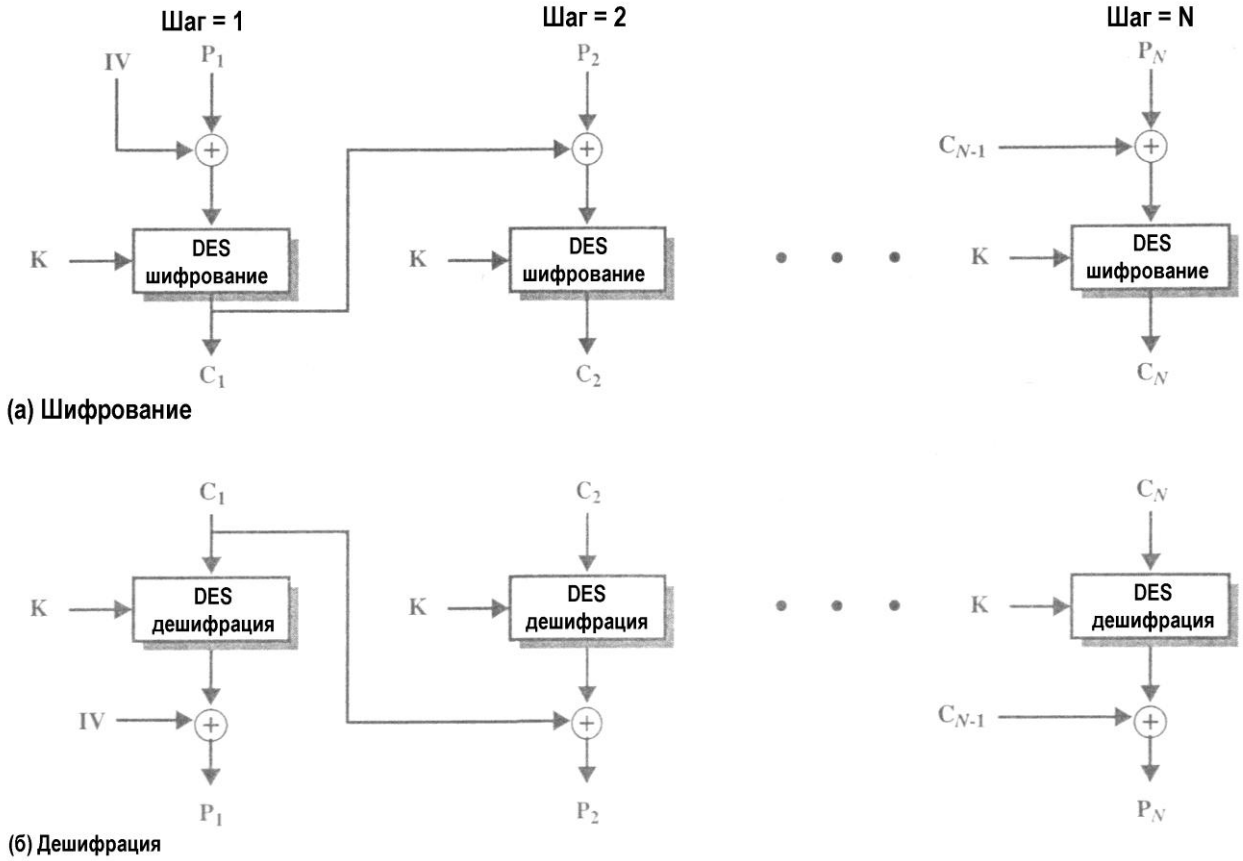


Рисунок 3.6. Режим цепочки.

Чтобы получить первый блок шифр-текста, вектор инициализации (IV) складывается по модулю 2 (XOR) с первым блоком исходного текста. При дешифровании IV складывается по модулю 2 с выходными данными алгоритма дешифрования для получения первого блока исходного текста.

Вектор инициализации должен быть известен как отправителю, так и получателю. Для большей безопасности вектор инициализации должен быть так же хорошо защищен как и секретный ключ. Это можно сделать путем посылки IV, используя режим ECB. Режим CBC широко используется в приложениях безопасности, как мы увидим позже.

Режим обратной связи (CFB)

Схема алгоритма DES применяет технику блочного шифрования, используя 64-битовые блоки. Тем не менее, возможно преобразовать DES к схеме поточного шифрования, используя режим обратной связи. Поточное шифрование не нуждается в разбиении исходного сообщения на целое число блоков. Также поточное шифрование может работать в реальном времени. Таким образом, если передается поток символов, то каждый символ может быть немедленно зашифрован и передан, используя символьно-ориентированный поточный шифровальщик.

Одним из желаемых свойств поточного шифровальщика является то, что шифр-текст будет той же длины, что и исходный текст. Рисунок 3.6 отображает схему CFB. Подразумевается, что передается единица данных размером в j бит; в общем случае j = 8. Как и в

случае схемы CBC единицы текста соединяются вместе таким образом, что шифр-текст является функцией всего предыдущего исходного текста.

Сначала рассмотрим процесс шифрования. На входе процедуры шифрования находится 64-битовый сдвиговый регистр, который изначально установлен в некоторое значение IV. Левые (наиболее значимые) j бит на выходе процедуры складываются по модулю 2 с первой единицей исходного текста P_1 для получения первой единицы шифр-текста C_1, который затем и передается. В добавку содержимое сдвигового регистра сдвигается влево на 8 бит и шифр-текст C_1 помещается на место правых (наименее значимых) бит. Этот процесс продолжается до тех пор, пока весь текст не будет зашифрован.

Для дешифровки используется аналогичная схема за исключением того, что полученные единицы шифр-текста складываются по модулю 2 с выходными данными функции шифрования для получения текста. Заметьте, что используется именно функции шифрования, а не дешифрования. Это легко объясняется. Обозначим $S_j(X)$ за наиболее значимые j бит величины X. Тогда:

$$C_1 = P_1 \oplus S_j(E(IV))$$

$$P_1 = C_1 \oplus S_j(E(IV))$$

1.5. Расположение устройств шифрования

Наиболее мощный и общий метод защиты сетевой безопасности – это шифрование. Используя его, нам необходимо решить, что шифровать и где должно располагаться устройство шифрования. Существует две основных альтернативы: шифрование канала и шифрование «точка-точка». Они проиллюстрированы на примере сети с пакетной коммутацией на рисунке 3.7.

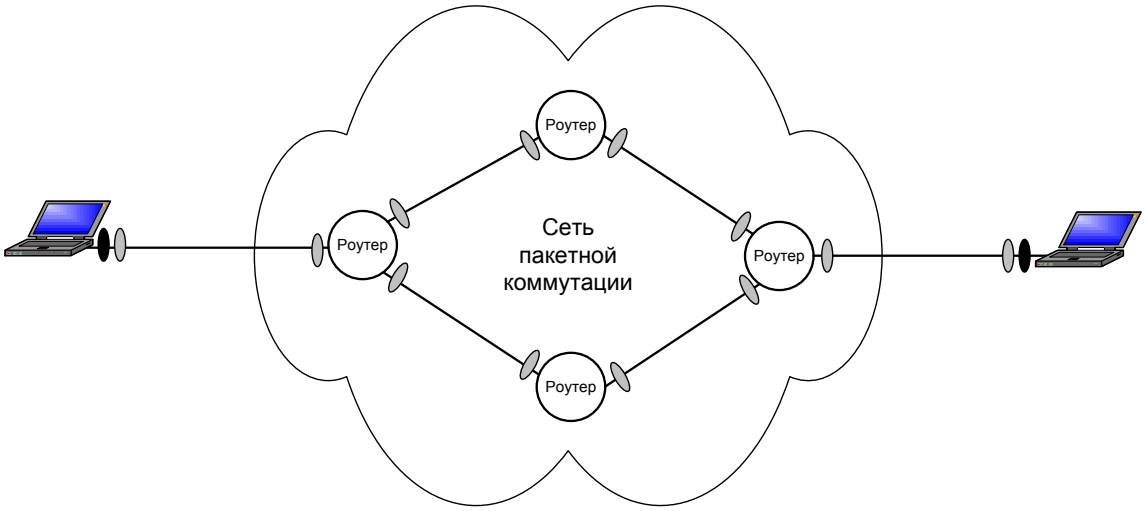


Рисунок 3.7. Расположение устройств шифрования.

Используя метод шифрования канала, каждый уязвимый канал передачи данных оборудуется на обоих концах устройством шифрования. Таким образом, весь трафик, передаваемый через канал, является защищенным. Несмотря на то, что данный метод требует большого количества шифровальных устройств в большой сети, он обеспечивает достаточно высокий уровень безопасности. Одним его недостатком является то, что сообщение должно декодироваться каждый раз по прибытию на маршрутизатор; это необходимо потому, что маршрутизатор должен считывать адрес назначения в заголовке пакета для определения следующего маршрута. Таким образом, сообщение является уязвимым на каждом маршрутизаторе. Если мы имеем дело с сетью общего назначения, то пользователь не имеет контроля над безопасностью в пределах узла.

Используя шифрование «точка-точка», процесс шифрования выносится на две оконечные системы. Отправитель шифрует данные. Затем данные передаются без изменений по сети до получателя. Получатель разделяет ключ с отправителем и, таким образом, в состоянии расшифровать данные. Этот метод может показаться безопасным для передачи данных. Но, тем не менее, и у него есть слабое место.

Рассмотрим следующую ситуацию. Узел подсоединен к сети X.25, устанавливает виртуальный канал с другим узлом и приготавливается передавать данные, используя шифрование «точка-точка». Данные передаются через сеть путем разбиения на пакеты, состоящих из заголовка и полезных данных. Какую часть каждого пакета необходимо шифровать узлу? Предположим, что узел шифрует весь пакет целиком, включая заголовок. Это не будет работать, потому что только получатель может осуществить дешифрацию. Следовательно, не возможно маршрутизировать пакет. Как вариант узел может шифровать только полезную часть данных и оставлять без изменений заголовок.

Таким образом, используя шифрование «точка-точка», можно обезопасить пользовательскую часть данных, но не весь трафик, потому что заголовки пакетов передаются в чистом виде. Для достижения полнейшей безопасности могут одновременно использоваться два рассмотренных типа шифрования.

1.6. Распределение ключей

Для того, чтобы стандартное шифрование работало, необходимо наличие одного и того же секретного ключа у отправителя и получателя информации. Более того желательна частая смена ключей. Поэтому надежность любой криптографической системы основывается на технике распределения ключей. Распределение ключей может быть осуществлено различными способами:

1. Ключ может быть выбран стороной А и физически доставлен на сторону Б.
2. Третий участник может выбрать ключ и физически доставить его сторонам А и Б.
3. Если А и Б уже использовали ключ, то одна из сторон может передать новый ключ, зашифрованный с использованием старого.
4. Если А и Б могут установить зашифрованное соединение к третьему участнику С, то С может передать ключ по этому соединению сторонам А и Б.

Первый и второй способы называются непосредственной доставкой ключа. Для шифрования канала это приемлемый способ. Но для шифрования «точка-точка» это неудобно. В распределенных системах любой узел может осуществлять обмен данными с множеством других систем, и чем больше сеть, тем больше участников обмена информацией.

Третий способ возможен как для шифрования канала, так и шифрования «точка-точка», но если злоумышленник уже получил старый ключ, то все последующие ключи будут взломаны. Четвертый способ является наиболее приемлемым для шифрования «точка-точка».

Рисунок 3.8 показывает решение, которое использует четвертый способ для шифрования «точка-точка». На рисунке 3.8 шифрование канала не применяется. Тем не менее, оно может быть добавлено. Для приведенной схемы определено два типа ключей:

- ✓ сессионный ключ. Когда две оконечные системы хотят установить соединение, они устанавливают логическое соединение. На время этого соединения все данные пользователя шифруются одноразовым сессионным ключом. По завершению соединения или сессии сессионный ключ разрушается;

- ✓ постоянный ключ. Постоянным является ключ, используемый для распространения сессионных ключей.

Конфигурация состоит из следующих элементов:

- ✓ центр распределения ключей. Определяет, каким системам разрешено взаимодействовать друг с другом. Если двум системам разрешено установить соединение, центр генерирует одноразовый сессионный ключ для этого соединения;
- ✓ внешний процессор. Выполняет шифрование «точка-точка» и получает сессионные ключи для своих узлов или терминалов.

Шаги, необходимые для установления соединения, показаны на рисунке 3.8. Когда один узел хочет установить соединение с другим узлом, он посылает пакет запроса на установление соединения (шаг 1). Внешний процессор сохраняет пакет и обращается к центру распределения ключей за разрешением на установление соединения (шаг 2). Обмен между процессором и центром шифруется с использованием мастер-ключа, известного только процессору и центру. Если центр подтверждает установление соединения, он генерирует сессионный ключ и передает его соответствующим двум внешним процессорам, используя уникальный постоянный ключ при обмене с каждым процессором (шаг 3). Все пользовательские данные, используемые при обмене, шифруются процессорами с использованием полученного одноразового сессионного ключа.

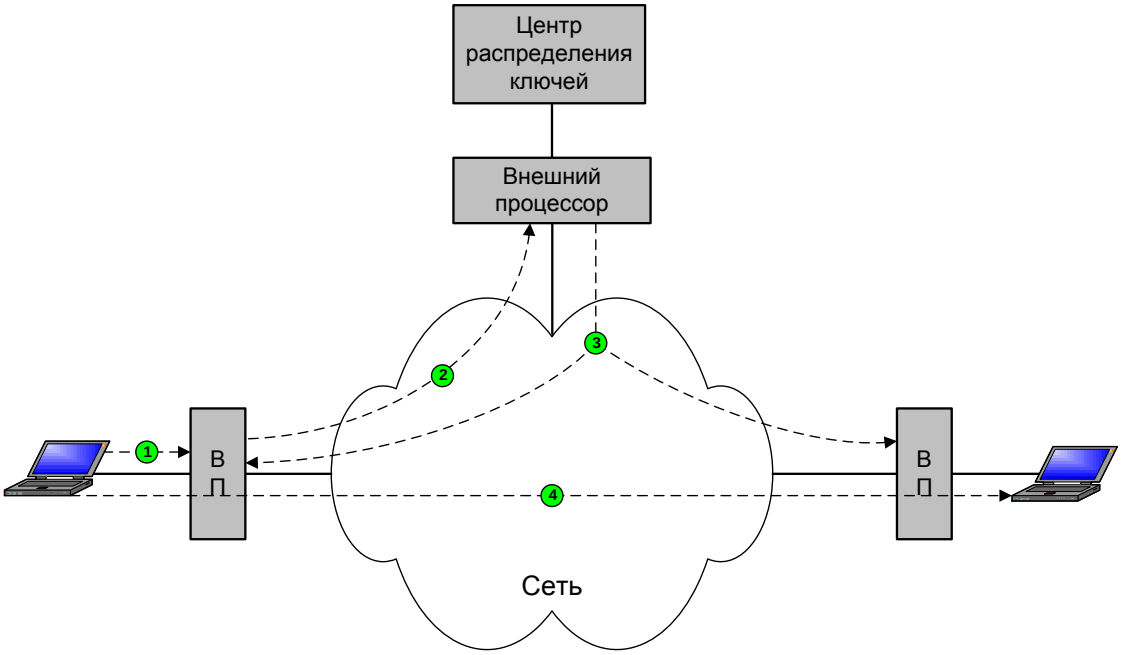


Рисунок 3.8. Автоматическое распределение ключей для протоколов с установлением соединения.

Другим способом распределения ключей является шифрование на основе общего ключа.

2. Криптография и аутентификация сообщений на основе общего ключа

КОК впервые была предложена в 1976 году. Особенности КОК:

- ✓ базируются на математических функциях, а не на простых операциях над битами (перестановка и сдвиг);
- ✓ КОК является асимметричной и использует два различных ключа (в отличие от стандартного шифрования, которое использует один ключ);

Схема КОК содержит 6 компонент (рисунок 3.9):

- ✓ открытый текст (текст) – это читабельное сообщение или данные, которые поступают на вход алгоритма шифрования;
- ✓ алгоритм шифрования – алгоритм, выполняющий различные преобразования над открытым текстом;
- ✓ открытый и приватный ключ – пара ключей, которые выбираются таким образом, что если один используется для шифрования, то другой для дешифрования;
- ✓ шифр-текст – это зашифрованное сообщение, получаемое на выходе алгоритма;
- ✓ алгоритм дешифрования – это алгоритм, получающий шифр-текст и ключ, а выдающий изначальный текст.

Как следует из названия, открытый ключ доступен для всех пользователей в сети, а приватный ключ доступен и известен только его хозяину. Основной принцип КОК заключается в том, что используется один ключ для шифрования и совсем другой (но связанный с первым) ключ для дешифрования. Основные шаги процедуры КОК следующие:

1. Каждый пользователь генерирует пару ключей (открытый и приватный), которая будет использоваться для шифрования и дешифрования сообщений.
2. Каждый пользователь выкладывает открытый ключ на общедоступный для других пользователей сети ресурс. Таким образом, каждый пользователь сети содержит набор открытых ключей других пользователей.
3. Если Алиса хочет послать приватное сообщение Михаилу, то данное сообщение шифруется на машине Алисы с использованием открытого ключа Михаила.
4. Когда Михаил получает сообщение, он дешифрует его с использованием собственного приватного ключа. Никакой другой пользователь в сети не сможет расшифровать сообщение для Михаила, потому что только Михаил обладает собственным приватным ключом.

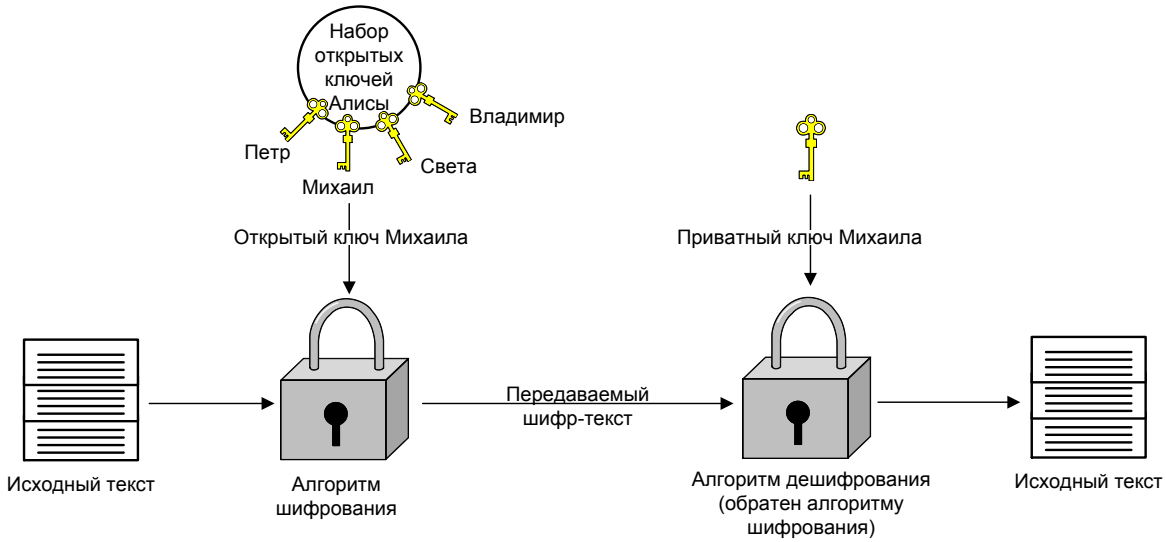


Рисунок 3.9. Упрощенная модель работы алгоритма шифрования с открытым ключом.

2.1. Области применения КОК

Системы на основе КОК делятся на следующие три категории:

- Собственно системы шифрования.
- Цифровые сертификаты.
- Цифровые подписи.
- Обмен ключами.

Цифровые сертификаты

В КОК открытый ключ является общедоступным. То есть любой пользователь сообщает всем остальным пользователям сети свой открытый ключ. При этом имеется следующий недостаток: если злоумышленник хочет выдать себя за пользователя А, то он может разослать свой открытый ключ по всей сети, выдав его за открытый ключ пользователя А. До тех пор, пока пользователь А не обнаружит обман и не сообщит о нем всем пользователям сети, злоумышленник сможет получать сообщения, предназначенные для А.

Решением данной проблемы являются цифровые сертификаты (ЦС) на основе открытого ключа. ЦС состоит из открытого ключа и идентификатора пользователя. ЦС «подписывается» надежной третьей стороной – центр управления сертификатами (ЦУС). Схема создания и распространения сертификатов следующая:

1. Пользователь генерирует открытый ключ и передает его безопасным образом в ЦУС.
2. ЦУС добавляет к ключу идентификатор пользователя и подписывает полученный сертификат (то есть шифрует сертификат своим приватным ключом).
3. ЦУС передает пользователю созданный сертификат.
4. Далее пользователь вместо передачи своего открытого ключа другим пользователям сети передает им собственный сертификат.
5. Любой пользователь может проверить подлинность сертификата, проверив его подпись. То есть подлинный сертификат может быть расшифрован только с использованием открытого ключа ЦУС.

Цифровые подписи

Отправитель «подписывает» сообщение своим приватным ключом. «Подпись» представляет использование криптографического алгоритма, который применяется либо к целому сообщению, либо к отдельной его части. Таким образом, дешифровать данное сообщение можно только с использованием открытого ключа отправителя и тем самым установить подлинность сообщения.

Обмен ключами с использованием КОК

В системах стандартного шифрования основным требованием для участников является сохранность секретного ключа. Естественно, возникает проблема обмена ключами между системами. Вторым наиболее применяемым способом решения данной проблемы является использования КОК. Когда пользователь А хочет послать сообщение пользователю Б он выполняет следующую последовательность шагов:

1. Подготавливает сообщение.
2. Используя стандартное шифрование, шифрует сообщение одноразовым сессионным ключом.
3. Используя КОК, шифрует сессионный ключ открытым ключом пользователя Б, который получен из сертификата пользователя Б.

4. Прикрепляет зашифрованный сессионный ключ к сообщению и посылает его пользователю Б.

Только пользователь Б в состоянии расшифровать сессионный ключ и, следовательно, сообщение пользователя А.

2.2. Методы аутентификации сообщений

Шифрование защищает от пассивных атак (прослушивание содержимого трафика сети). Совсем другие требования предъявляются к защите против активных атак (фальсификация данных и транзакций). Защита против подобных атак известна как аутентификация сообщений.

Сообщение, файл, документ или другое множество данных является подлинным (аутентичным), когда оно истинно и пришло из предполагаемого источника. Аутентификация сообщения есть процедура, которая позволяет взаимодействующим сторонам проверять, являются ли получаемые сообщения истинными. Двумя важными аспектами в этом вопросе являются проверка того, что содержимое сообщения не изменено и источник сообщения является подлинным. Также желательно проверять своевременность сообщения (доказательство того, что сообщение не воспроизведено и не имеет временных задержек) и последовательность сообщения в информационном потоке в целом.

Аутентификация с использованием стандартного шифрования

Возможно, выполнять аутентификацию, просто используя стандартное шифрование. Если только отправитель и получатель разделяют секретный ключ (как и должно быть), тогда только настоящий отправитель будет в состоянии зашифровать сообщение. Более того, если сообщение содержит код обнаружения ошибок и номер последовательности, то получатель будет уверен, что содержимое сообщения не менялось по пути следования и оно получено в правильной последовательности. А если сообщение к тому же включает временной штамп, то получатель уверен, что сообщение не задерживалось по пути следования.

Аутентификация сообщений без шифрования

Во всех подобных методах генерируется так называемый аутентификационный признак (тэг), который затем добавляется к каждому сообщению при его передачи. Сообщение само по себе не шифруется и может быть прочтено получателем независимо от его функции аутентификации.

Из-за отсутствия шифрования не обеспечивается конфиденциальность сообщения. Так как стандартное шифрование обеспечивает аутентификацию и широко используется в многих готовых программных продуктах, почему бы просто не использовать его. При этом мы получим и конфиденциальность, и функцию аутентификации. Существует как минимум три ситуации, когда предпочтительна аутентификация без конфиденциальности:

1. Существует ряд приложений, в которых одно и то же сообщение передается широковещательным образом нескольким получателям. Дешевле и более надежно иметь только одного получателя, который будет ответственен за мониторинг подлинности сообщения. Таким образом, подобные сообщения должны передаваться в открытом виде с добавлением аутентификационного тэга. Ответственная система выполняет аутентификацию. Если производится нарушение, то другие системы оповещаются о нем путем получения сигнала предупреждения.

2. Другим возможным сценарием является обмен, в котором одна сторона сильно загружена и не имеет времени на дешифровку всех входящих сообщений. Аутентификация производится выборочно.
3. Аутентификация компьютерной программы является привлекательной возможностью. Компьютерная программа может выполняться без необходимости быть расшифрованной, что разгружает процессорное время. Тем не менее, если аутентификационный признак добавлен к программе, то он может проверяться, когда необходимо проверить ее целостность.

Код аутентификации сообщения

Одна из аутентификационных техник предполагает использование секретного ключа для генерации маленького блока данных, называемого аутентификационным кодом. Данный код добавляется к сообщению. Данная техника предполагает, что две взаимодействующие стороны, А и Б, разделяют секретный ключ $К_{АБ}$. Когда А посылает сообщение Б, она вычисляет аутентификационный код как функцию сообщения и ключа. Затем сообщение и код передаются предполагаемому получателю. Получатель выполняет те же вычисления для генерации своего аутентификационного кода, а затем сравнивает его с полученным. Если только отправитель и получатель знают секретный ключ и если полученный код совпадает с вычисленным, то:

1. Получатель может быть уверен, что содержимое сообщения не изменено.
2. Получатель может быть уверен, что сообщение получено от предполагаемого источника.
3. Если сообщение включает номер последовательности, то получатель может быть уверен, что последовательность сообщений верна.

Для генерации аутентификационного кода может использоваться ряд алгоритмов. Национальное Бюро Стандартов США (публикация «Режимы операций алгоритма DES») рекомендует использовать DEA. DEA используется для генерации зашифрованной версии сообщения и только несколько последних бит этого шифра используются как код. Обычно это 16 или 32 битовый код.

Односторонняя кэш функция

Одним из вариантов создания аутентификационного кода является односторонняя кэш функция. Кэш функция в качестве входных параметров берет сообщение М переменного размера и производит слепок Н(М) сообщения, имеющий фиксированный размер. В отличие от предыдущей технологии (код аутентификации сообщения) кэш функция не использует секретный ключ. Для того, чтобы аутентифицировать сообщение, его слепок посылается вместе с самим сообщением получателю.

Рисунок 3.10 показывает три способа аутентификации сообщения. Слепок сообщения может быть зашифрован с использованием стандартного шифрования (a). Сообщение также может быть зашифровано с использованием шифрования на основе общего ключа (b). Данный способ шифрования имеет два преимущества: наряду с аутентификацией он обеспечивает цифровую подпись; не требуется применения техники распределения ключей.

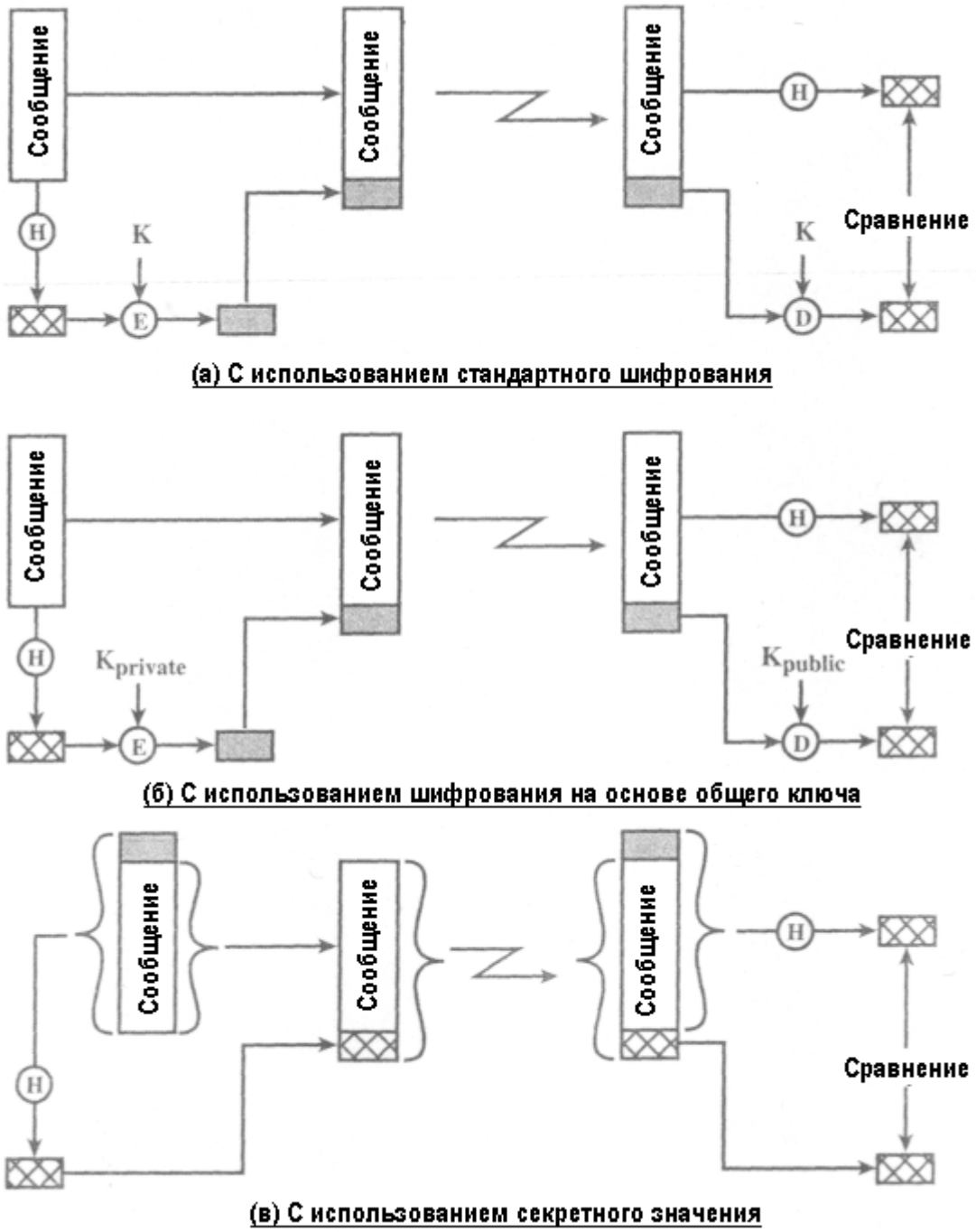


Рисунок 3.10. Аутентификация сообщения с использованием односторонней кэш функции.

Данные два метода имеют преимущество над теми, которые шифруют все сообщение целиком, – требуется проделывать меньше вычислений. Тем не менее, остается интерес в разработке техники аутентификации, которая бы не применяла шифрование вообще. На это существует ряд причин:

- ✓ шифровальное ПО достаточно медленно;
- ✓ шифровальное оборудование недешево. Существуют недорогие чипы для шифрования по DES, но стоимость растет, если необходимо внедрить подобные чипы во все узлы сети;
- ✓ шифровальное оборудование оптимизируется для обработки больших объемов данных. Для маленьких блоков данных затрачивается больше времени на инициализацию и вызов функций;
- ✓ алгоритмы шифрования могут быть защищены патентами;

- ✓ алгоритмы шифрования могут быть предметом контроля над экспортом. Это относится к DES.

Рисунок 3.10в показывает технику, которая использует кэш функцию, но не использует шифрования для аутентификации сообщения. Предполагается, что две взаимодействующие стороны, А и Б, разделяют между собой общее секретное значение S_{AB}. Когда А хочет послать сообщение Б, она вычисляет кэш функцию как конкатенацию секретной величины и сообщения: $MD_M = H(S_{AB} || M)$. Затем А посылает Б сообщение $[M || MD_M]$. Так как Б обладает величиной S_{AB}, то он может проверить пришедшее сообщение. Так как секретная величина не посылается, то злоумышленник не может модифицировать сообщение в пути.

2.3. Безопасные кэш функции и HMAC

Одностороння кэш функция, или безопасная кэш функция, полезна не только для аутентификации сообщений, но и в цифровых подписях.

Требования к кэш функциям

Основное назначение кэш функции заключается в производстве «отпечатков пальцев» файла, сообщения или другого блока данных. Для того, чтобы быть применимой для аутентификации сообщений, кэш функция H должна обладать следующими свойствами:

1. H может быть применена к блоку данных любого размера.
2. H производит выходные данные фиксированного размера.
3. H(x) легко посчитать для любого заданного x.
4. Для любого заданного кода h невозможно найти такое x, что H(x) = h.
5. Для любого заданного блока x невозможно найти y ≠ x так, что H(x) = H(y).
6. Невозможно найти любую пару (x, y) такую, что H(x) = H(y).

Первые три свойства являются требованиями для практического применения кэш-функции при аутентификации сообщений. Четвертое свойство является “односторонним”: легко сгенерировать код для заданного сообщения, но практически невозможно сгенерировать сообщение по заданному коду. Данное свойство является важным, если аутентификация включает использование секретного ключа. Сам секретный ключ не посылается. Тем не менее, если кэш-функция является не односторонней, то атакующий может легко раскрыть секретный ключ.

Пятое свойство гарантирует, что не возможно найти альтернативное сообщение с тем же самым кэш значением, что и для заданного сообщения. Это предохраняет от подделок.

Стойкость ко взлому криптографических систем

Схема шифрования является безопасной с точки зрения вычислений, если шифр-текст, сгенерированный этой схемой отвечает одному или более условиям:

- ✓ стоимость взлома шифра превышает ценность зашифрованной информации;
- ✓ время взлома превышает полезное время жизни информации.

Проблема заключается в том, что очень трудно вычислить количество попыток, требуемых для удачного взлома шифра. Тем не менее, если предположить, что алгоритм шифрования не имеет никаких математических изъянов и, следовательно, для его взлома применима только брут-форс атака, то тогда мы можем сделать некоторые предположения относительно времени и стоимости взлома.

Суть брут-форс атаки заключается в том, что при взломе шифра перебираются все возможные ключи до тех пор, пока не будет получен нужный результат. В среднем необходимо перебрать половину ключей. Если множество ключей слишком большое, подобная задача становится практически невозможной. Таблица 3.5 показывает, как много времени требуется для перебора ключей различной длины.

Таблица 3.5 – Среднее время, требуемое для полного перебора всего множества ключей

Размер ключа (бит)	**Количество ключей**	**Затрачиваемое время при скорости 1 ключ/мкс**	**Затрачиваемое время при скорости 10^6 ключей/мкс**
32	$2^{32} = 4{,}3 * 10^9$	2^{31} мкс = 35,8 минут	2,15 миллисекунд
56	$2^{56} = 7{,}2 * 10^{16}$	2^{55} мкс = 1142 лет	10 часов
128	$2^{128} = 3{,}4 * 10^{38}$	2^{127} мкс = $5{,}4 * 10^{24}$ лет	$5{,}4 * 10^{18}$ лет
168	$2^{168} = 3{,}7 * 10^{50}$	2^{167} мкс = $5{,}9 * 10^{36}$ лет	$5{,}4 * 10^{30}$ лет

IV. Механизмы обеспечения безопасности коммутируемых локальных сетей

1. Ограничение количества управляющих компьютеров

Современные коммутаторы позволяют задать конкретные компьютеры (IP-адреса), с которых будет происходить настройка данных коммутаторов по Web- или Telnet- инерфейсам или через протокол SNMP. Попытка прочих компьютеров подключиться к коммутаторам для управления ими будет отклонена.

2. Настройка безопасности индивидуального порта

Данная функция позволяет:

- заблокировать дальнейшее обновление таблицы коммутатора – если конфигурирование Вашей сети больше не изменятся, Вы блокируете таблицу коммутации, и коммутатор будет просто отбрасывать все пакеты, которые поступают с неизвестных адресов. Причём данное действие можно выбрать для конкретных портов. При этом записи в таблице коммутации не будут удаляться по тайм-ауту;
- задать максимальное количество MAC-адресов для привязки к конкретному порту.

3. Фильтрация MAC-адресов

Дополнительно можно настроить коммутатор так, чтобы он не принимал пакеты с определенным MAC-адресом (как получателя, так и отправителя). Для этого служит таблица фильтрации трафика (Filtering Table). Фактически данная таблица является «чёрным списком». После получения пакета коммутатор считывает оба аппаратных адреса, как получателя, так и отправителя. Если хотя бы один из них содержится в таблице фильтрации, то пакет отбрасывается.

4. Технология фильтрации IP-MAC Binding

Основное назначение данной технологии – это ограничить доступ к коммутатору определённого количества компьютеров. Для этого индивидуально для каждого желаемого порта создаётся таблица соответствия IP- и MAC-адресов. Далее коммутатор будет пропускать только пакеты с указанными MAC-адресами и только в том случае, если истинно соответствие IP- и MAC-адреса.

Существенные отличия данной технологии от технологии фильтрации MAC-адресов:

- фильтрация работает на всех портах, а в данной технологии можно настраивать каждый порт в отдельности;
- фильтрация содержит «чёрный» список, то есть работает по принципу: не пропускать те пакеты, MAC-адреса которых содержатся в таблице фильтрации. Данная технология, наоборот, содержит «белый» список, то есть пропускает только те пакеты, MAC-адреса которых содержатся в созданных списках;
- в отличие от фильтрации технология IP-MAC Binding проверяет не только MAC-адрес источника пакета, но и его IP-адрес.

5. Списки контроля доступа (Access Control Lists)

Списки контроля доступа обеспечивают ограничение прохождения трафика через коммутатор. Фактически технология ACL реализует на коммутаторах 3-го уровня полноценный фильтр пакетов. Приём пакетов или отказ в приёме основывается на определённых признаках, которые содержит пакет:

- ✓ IP- и MAC-адреса источника и приёмника;
- ✓ номер VLAN;
- ✓ номер порта TCP или UDP;
- ✓ тип ICMP-сообщения.

Ключевым понятием в данной технологии является понятие профиля доступа – это набор признаков, который содержит пакет, с определёнными значениями. Каждый профиль доступ имеет свой уникальный номер в пределах коммутатора. Пример профилей доступа:

1: <VLAN = Yes, Source MAC = 00-01-08-DE-FA-10, Destination MAC = 00-01-07-DE-FA-10>
2: <VLAN = Yes, Source IP = 192.168.3.1, Destination IP = 192.168.10.2, TCP port = Yes>

Помимо приведенных примеров существуют также контекстно-зависимые профили. Основное отличие контекстно-зависимых профилей от всех остальных заключается в том, что в них признаки задаются смещением относительно начала кадра Ethernet. Например, чтобы указать MAC-адрес источника пакета в контекстно-зависимом профиле необходимо указать смещение Offset = 6 байтам и далее значение MAC-адреса, так как данный адрес располагается в заголовке кадра Ethernet с 7 по 12 байт включительно.

Профиль является всего лишь образцом (некоторым эталоном), на основе которого создаются правила. Правила представляют собой профиль, заполненный конкретными значениями признаков и содержащий действие, которое необходимо выполнить над пакетом, если он попадает под эти признаки. Обычно действие – это удалить или продвинуть пакет. Для каждого профиля может быть создано более одного правила. Например, для вышеприведённого профиля с ID=2 могут быть созданы следующие правила:

Accept: <VLAN = Default, Source IP = 192.168.3.1, Destination IP = 192.168.10.2, TCP port = 514>
Deny: <VLAN = Marketing, Source IP = 192.168.3.1, Destination IP = 192.168.10.2, TCP port = 8080>

При поступлении пакета на коммутатор профили доступа применяются последовательно, в порядке возрастания их номеров. Пакет проверяется на соответствие условиям, указанным во всех правилах профиля, начиная с первого профиля. Если правило подходит, пакет в соответствие с действием обработки либо принимается, либо отбрасывается и дальше не проверяется. Если пакет не попадает ни под одно правило профиля, то пакет проверяется по правилам следующего профиля. Если не один профиль не подходит, то применяется политика по умолчанию – разрешающая или запрещающая прохождение трафика.

Очень важно правильно расставлять профили и правила внутри профилей потому, что если пакет попадает хотя бы под одно запрещающее правило, то он удаляется из коммутатора и дальше не проверяется на соответствие другим правилам, даже если среди них есть правила, которые могли бы его пропустить дальше.

6. Сегментация трафика (Traffic Segmentation)

Сегментация трафика служит для разграничения портов на Канальном уровне. Данная функция позволяет настраивать порты или группу портов таким образом, чтобы они были изолированы друг от друга, но в то же время имели доступ к разделяемым портам, используемым для подключения серверов или магистрали сети провайдера. Очень важной является возможность одновременного использования данной функции с виртуальными сетями (VLAN), что позволяет производить дальнейшее разграничение прав.

Данная технология схожа с технологией VLAN, но является более ограниченной по функциональности. К преимуществам технологии можно отнести:

- простота настройки;
- более низкая загрузка процессора коммутатора по сравнению с VLAN.

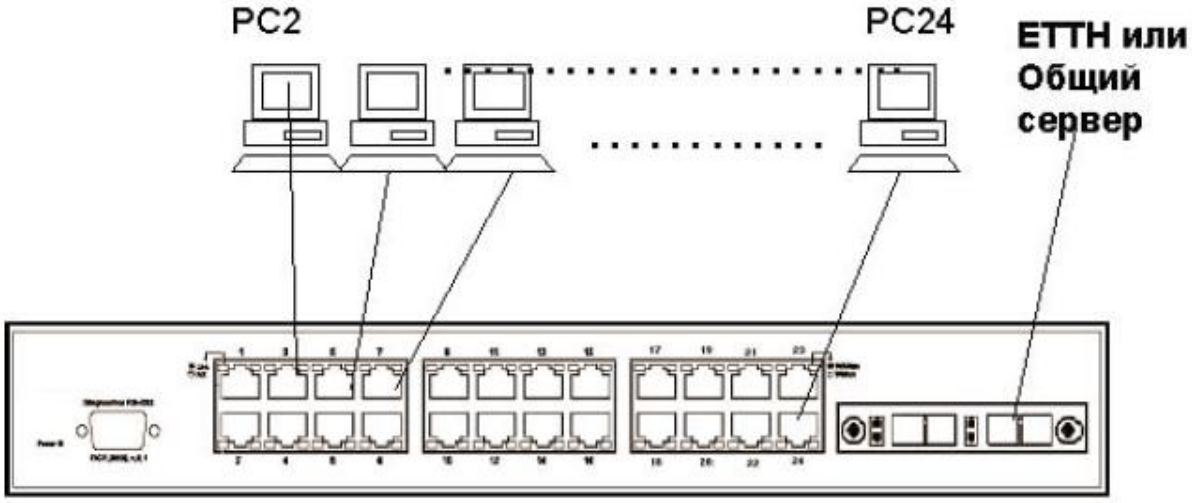


Рисунок 4.1.

Рассмотрим пример (рисунок 4.1). Все компьютеры имеют доступ к общему порту, но не имеют доступа к друг другу на Канальном уровне. Это решение может быть использовано в:

- в проектах ETTH (Ethernet To The Home, Ethernet до дома) для изоляции компьютеров конечных пользователей;
- для предоставления доступа к общему серверу.

7. Протокол IEEE 802.1x

Протокол IEEE 802.1x является механизмом безопасности, обеспечивающим аутентификацию и авторизацию пользователей и тем самым ограничивающим доступ проводных или беспроводных устройств к локальной сети. Работа протокола базируется на клиент-серверной модели контроля доступа (рисунок 4.2). В качестве сервера аутентификации используется RADIUS-сервер. При этом весь процесс аутентификации пользователя производится в проводных сетях на основе протокола EAPOL (Extensible Authentication Protocol over LAN), в беспроводных – на основе протокола EAPOW (Extensible Authentication Protocol over Wireless).

Рисунок 4.2.

До тех пор, пока клиент не будет аутентифицирован, протокол IEEE 802.1x будет пропускать через сетевой порт только трафик протокола EAPOL. После успешной аутентификации обычный трафик будет пропускаться через порт. Работа протокола IEEE 802.1x основывается на трёх компонентах (рисунок 4.2), каждая из которых подробно рассмотрена в следующем разделе.

7.1. Роли устройств

Клиент – это рабочая станция, которая запрашивает доступ к локальной сети и сервисам коммутатора и отвечает на запросы коммутатора. На рабочей станции должно быть установлено клиентское ПО, реализующее протокол 802.1x (в ОС Microsoft Windows XP данное ПО является встроенным).

Сервер аутентификации выполняет фактическую аутентификацию клиента, проверяя подлинность клиента и информируя коммутатор, предоставлять или нет клиенту доступ к локальной сети.

Коммутатор (также называется аутентификатор) управляет физическим доступом к сети, основываясь на статусе аутентификации клиента. Коммутатор работает как посредник между клиентом и сервером аутентификации, получая запрос на проверку подлинности от клиента, проверяя данную информацию при помощи сервера аутентификации, и пересылая ответ клиенту. ПО коммутатора включает клиента RADIUS, который отвечает за инкапсуляцию и деинкапсуляцию кадров EAP и взаимодействие с сервером аутентификации.

7.2. Процесс аутентификации

Инициировать процесс аутентификации может коммутатор или клиент. Клиент инициирует аутентификацию, посылая кадр EAPOL-start, который вынуждает коммутатор отправить ему запрос на идентификацию. Когда клиент отправляет EAP – ответ со своей идентификацией, коммутатор начинает играть роль посредника, передающего кадры EAP между клиентом и сервером аутентификации до успешной или неуспешной аутентификации. Если аутентификация завершилась успешно, порт коммутатора становится авторизованным.

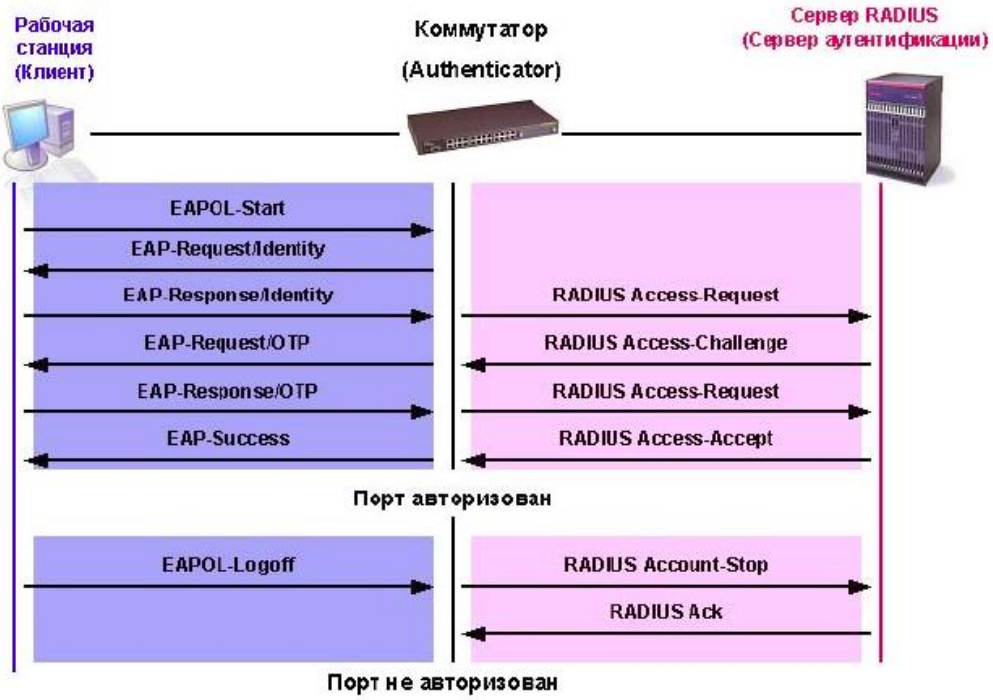


Рисунок 4.3. Временная диаграмма аутентификации клиента в сети.

Временная диаграмма обмена EAP-кадрами зависит от используемого метода аутентификации. На рисунке 4.3 показана схема обмена, инициируемая клиентом, использующая метод аутентификации с использованием одноразовых паролей (One Time Password, OTP) сервером RADIUS.

7.3. Состояние портов коммутатора

Состояние порта коммутатора определяется тем, получил или не получил клиент право доступа к сети. Первоначально порт находится в неавторизованном состоянии. В этом состоянии он запрещает прохождение всего входящего и исходящего трафика за исключением пакетов протокола IEEE 802.1x. Когда клиент аутентифицирован, порт переходит в авторизованное состояние, позволяя передачу любого трафика от него.

Возможны варианты, когда клиент или коммутатор не поддерживают протокол IEEE 802.1x. Если клиент, который не поддерживает протокол IEEE 802.1x, подключается к неавторизованному порту, коммутатор посылает клиенту запрос на аутентификацию. Поскольку в этом случае клиент не ответит на запрос, порт останется в неавторизованном состоянии и клиент не получит доступ к сети.

В другом случае, когда клиент с поддержкой протокола IEEE 802.1x подключается к порту, на котором не запущен протокол IEEE 802.1x, клиент начинает процесс аутентификации, посылая кадр EAPOL-start. Не получив ответа, клиент посылает запрос определённое количество раз. Если после этого ответ не получен, клиент, считая, что порт находится в авторизованном состоянии начинает посылать кадры.

В случае, когда и клиент и коммутатор поддерживают протокол IEEE 802.1x, при успешной аутентификации клиента, порт переходит в авторизованное состояние и начинает передавать все кадры клиента. Если в процессе аутентификации возникли ошибки, порт остаётся в неавторизованном состоянии, но аутентификация может быть восстановлена. Если сервер аутентификации не может быть достигнут, коммутатор может повторно передать запрос. Если от сервера не получен ответ после определённого количества попыток, то в доступе к сети будет отказано из-за ошибок аутентификации.

Когда клиент завершает сеанс работы, он посылает сообщение EAPOL-logoff, переводящее порт коммутатора в неавторизованное состоянии. Если состояние канала связи порта переходит из активного (up) в неактивное (down), то порт также возвращается в неавторизованное состояние.

7.4. Методы контроля доступа при использовании протокола IEEE 802.1x

Протокол IEEE 802.1x предоставляет два метода контроля доступа к сети:

1. На основе портов (Port-Based Access Control). При использовании данного метода достаточно, чтобы только один любой пользователь, подключенный к порту коммутатора, был авторизован. Тогда порт перейдёт в авторизованное состояние и доступ к сети получат любые пользователи, подключенному к данному порту.
2. На основе MAC-адресов (MAC-Based Access Control). При использовании данного метода при аутентификации также учитывается MAC-адрес клиента, подключенного к порту, и порт авторизуется только для клиента с конкретным MAC-адресом.

Контроль доступа на основе портов

Изначально протокол IEEE 802.1x разрабатывался с учётом того, что к порту коммутатора подключено не более одного устройства (рисунок 4.4). Как только устройство успешно проходило процедуру аутентификации, порт переходил в авторизованное состояние и далее пропускал весь трафик до тех пор, пока не наступало событие, которое обратно переводило его в неавторизованное состояние. Следовательно, если порт коммутатора подключен не к одному устройству, а к сегменту локальной сети, то успешная аутентификация любого устройства из этого сегмента открывает доступ в сеть всем остальным уст-

ройствам из сегмента. Естественно, это является серьёзной проблемой с точки зрения безопасности.

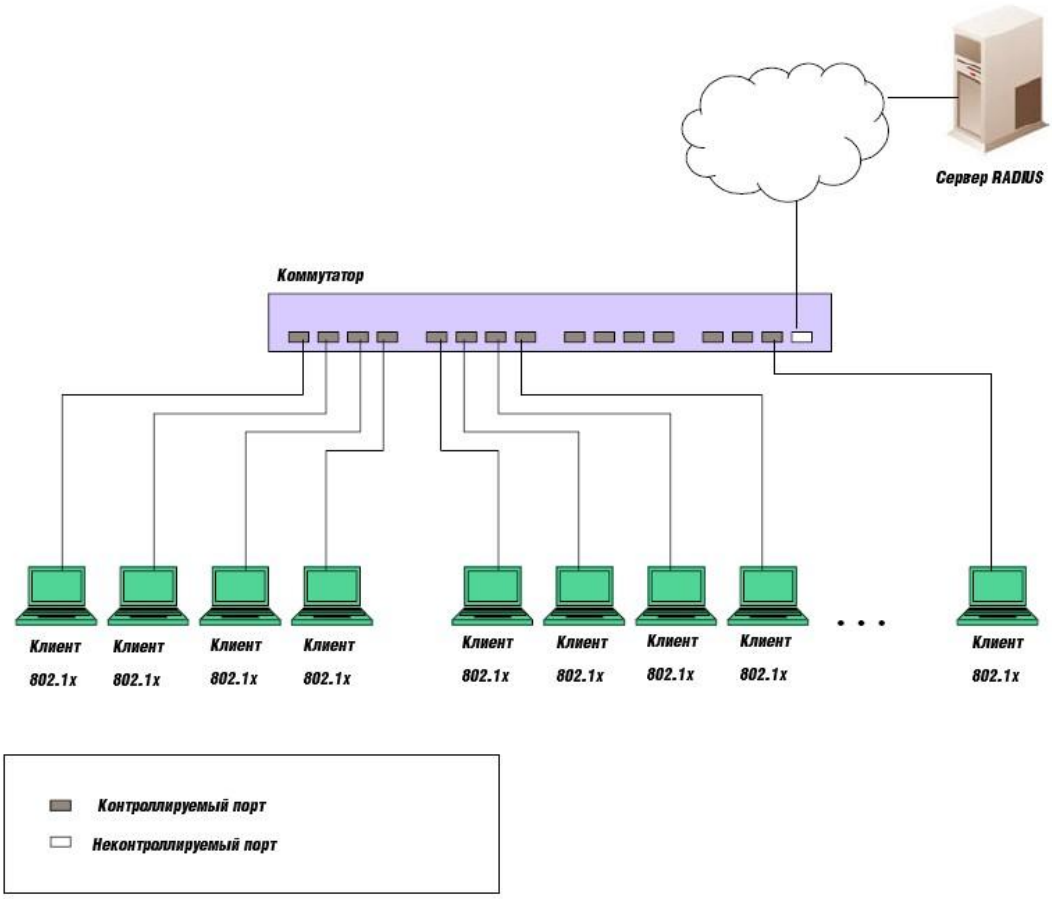


Рисунок 4.4.

Контроль доступа на основе MAC-адресов

Для того, чтобы успешно использовать протокол IEEE 802.1x в распределённых локальных сетях, необходимо создавать логические порты – по одному логическому порту на каждое устройство, подключенное к физическому порту. Таким образом, физический порт представляет собой множество логических портов, каждый из которых независимо контролирует отдельное устройство-клиента с точки зрения аутентификации и авторизации. Принадлежность устройства к определённому логическому порту осуществляется на основе MAC-адреса устройства (рисунок 4.5). Таким образом, устраняется проблема безопасности доступа множества устройств через один физический порт коммутатора.

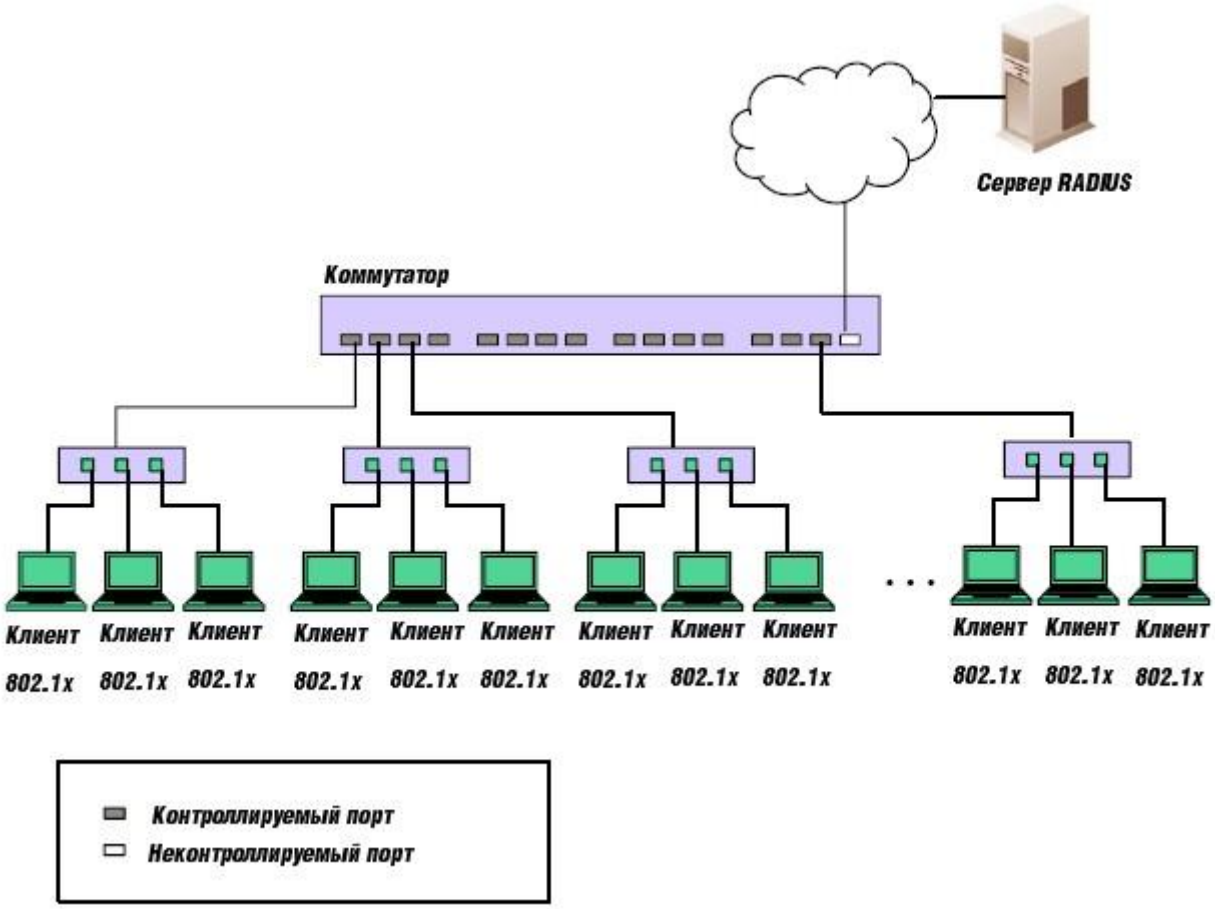

Сервер RADIUS
Коммутатор
Клиент
802.1x
Контроллируемый порт
Неконтроллируемый порт

Рисунок 4.5.

8. Виртуальные сети (Virtual LAN)

Виртуальная ЛВС (VLAN, Virtual LAN) – логическая группа компьютеров в пределах одной реальной ЛВС, за пределы которой не выходит любой тип трафика (широковещательный [broadcast], многоадресный [multicast] и одноадресный [unicast]). За счет использования VLAN администратор сети может организовать пользователей в логические группы независимо от физического расположения рабочих станций этих пользователей.

Основные цели введения виртуальных сетей в коммутируемую среду:

- повышение полезной пропускной способности сети за счет локализации широковещательного (broadcast) трафика в пределах виртуальной сети;
- повышения уровня безопасности сети за счет локализации одноадресного (unicast) трафика в пределах виртуальной сети;
- формирование виртуальных рабочих групп из некомпактно (в плане подключения) расположенных узлов;
- улучшение соотношения цены/производительности по сравнению с применением маршрутизаторов.

Классический пример, отражающий суть виртуальных сетей, приведен на рисунке 4.7. К одному коммутатору гипотетической фирмы подключены как машины бухгалтеров, так и машины бухгалтеров. При этом совершенно нет никакой необходимости во взаимодействие машин данных двух групп сотрудников. Поэтому машины бухгалтеров выделяются в одну виртуальную сеть, а машины инженеров в другую. При этом весь трафик (широковещательный, многоадресный и одноадресный) будет ограничен пределами своей виртуальной сети. Более того, повысится безопасность сети. Например, если на машине бухгалтера появится вирус «червь», то максимум он сможет заразить только машины бухгалтеров.

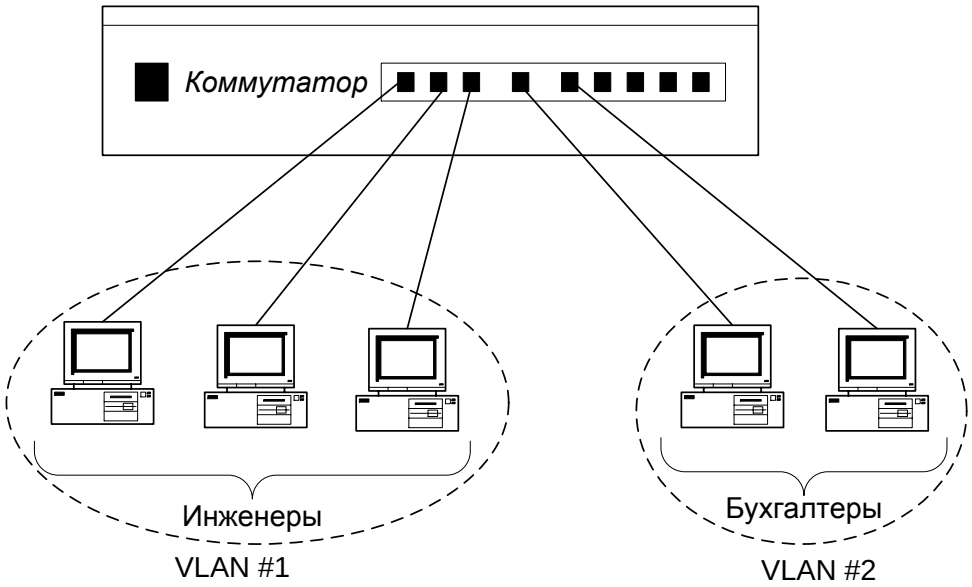


Рисунок 4.7. Пример использования технологии виртуальных сетей.

Сегодня существует достаточно много вариантов реализации VLAN. Простые варианты VLAN представляют собой набор портов коммутатора, более сложные реализации позволяют создавать группы на основе других критериев. В общем случае возможности организации VLAN тесно связаны с возможностями коммутаторов.

8.1. Сети на базе MAC-адресов (MAC-based VLANs)

Данный тип виртуальных сетей группирует устройства на основе их MAC-адресов. Для получения доступа в виртуальную сеть, устройство должно иметь MAC-адрес, который

содержится в списке адресов данной виртуальной сети. Помимо прочего, отличительной особенностью данного типа виртуальных сетей является то, что они ограничивают только широковещательный трафик. Отсюда вытекает их название – широковещательные домены на базе MAC-адресов. Теоретически один MAC-адрес может являться членом нескольких широковещательных доменов, на практике данная возможность определяется функциональностью конкретной модели коммутатора.

Настройка виртуальной сети на основе MAC-адресов может отнять много времени. Представьте себе, что вам потребуется связать с VLAN адреса 1000 устройств. Кроме того, MAC-адреса "зашиты" в оборудование и может потребоваться много времени на выяснение адресов устройств в большой, территориально распределенной сети. Более того, существуют проблемы безопасности при работе с сетью на базе MAC-адресов. Злоумышленник может узнать MAC-адрес компьютера, входящего в ту или иную VLAN, назначить его своей сетевой карте (как минимум, это можно сделать стандартными средствами MS Windows XP) и успешно подключиться к сети.

С другой стороны, широковещательные домены на базе MAC-адресов позволяют физически перемещать станцию, позволяя, тем не менее, оставаться ей в одном и том же широковещательном домене без каких-либо изменений в настройках конфигурации. Это удобно в сетях, где станции часто перемещают, например, где люди, использующие ноутбуки, постоянно подключаются в разных частях сети.

8.2. Сети на базе портов (Port-based VLANs)

Устройства связываются в виртуальные сети на основе портов коммутатора, к которым они физически подключены. То есть каждой порт коммутатора включается в одну или более виртуальных сетей. К достоинствам данного типа виртуальных сетей можно отнести высокий уровень безопасности и простоту в настройке. К недостаткам можно отнести статичность данного типа виртуальных сетей. То есть при подключении компьютера к другому порту коммутатора необходимо каждый раз изменять настройки VLAN.

8.3. Сеть на базе маркированных кадров (IEEE 802.1Q VLANs)

В отличие от двух предыдущих типов виртуальных сетей VLAN на основе маркированных кадров могут быть реализованы на двух и более коммутаторах. В заголовок каждого кадра Ethernet вставляется маркер, который идентифицирует членство компьютера в определенной VLAN. На рисунке 4.8 показан пример такой VLAN.

Механизмы добавления идентифицирующего маркера в заголовок кадра Ethernet

Маркеры с номером VLAN в виртуальных сетях 802.1Q могут быть добавлены:

- явно, если сетевые карты поддерживают стандарт IEEE 802.1Q, и на этих картах включены соответствующие опции, то исходящие кадры Ethernet от этих карт будут содержать маркеры идентификации;
- неявно, если сетевые адаптеры, подключенные к этой сети, не поддерживают стандарт IEEE 802.1Q, то добавление маркеров выполняется на коммутаторе на основе группировки по портам.

Предположим, что некоторые порты коммутатора сгруппированы в VLAN. Коммутатор с поддержкой IEEE 802.1Q будет добавлять маркер к входящим на этот порт кадрам Ethernet с соответствующим ID VLAN. Но эти маркеры будут удаленны коммутатором из исходящих с этих же портов кадров, чтобы конечные компьютеры могли разобрать заголовок кадра Ethernet.

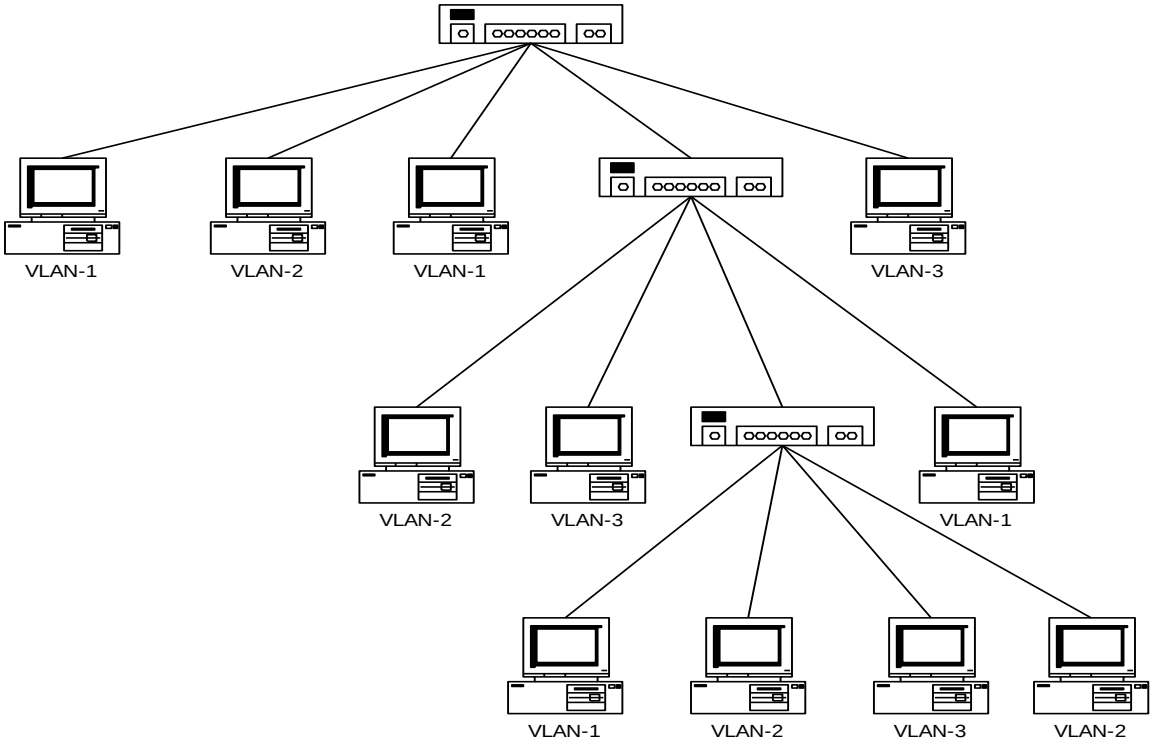


Рисунок 4.8. Виртуальная сеть на основе нескольких коммутаторов с использованием стандарта IEEE 802.1Q.

Если идентификация VLAN маркерами протокола 802.1Q была осуществлена обоими способами – явно и неявно, входящие кадры к портам коммутатора могут состоять из обоих типов кадров (с маркерами и без). В этой ситуации к неотмеченным входящим кадрам будут добавляться маркеры. В то время как маркированные кадры уже поддерживают членство в явно определенной виртуальной сети. Способность VLAN 802.1Q извлечения маркеров из заголовков кадров Ethernet позволяет VLAN работать с существующими коммутаторами и сетевыми адаптерами, которые не распознают маркеры. Способность добавления маркеров позволяет VLAN распространяться через множество 802.1Q-совместимых коммутаторов.

Компоненты конфигурации виртуальной сети на основе тэгов

Необходимо понять суть двух ключевых переменных для настройки VLAN 802.1Q:

- Port VLAN ID (PVID, идентификатор номера порта);
- VLAN ID (VID, идентификатор номера VLAN).

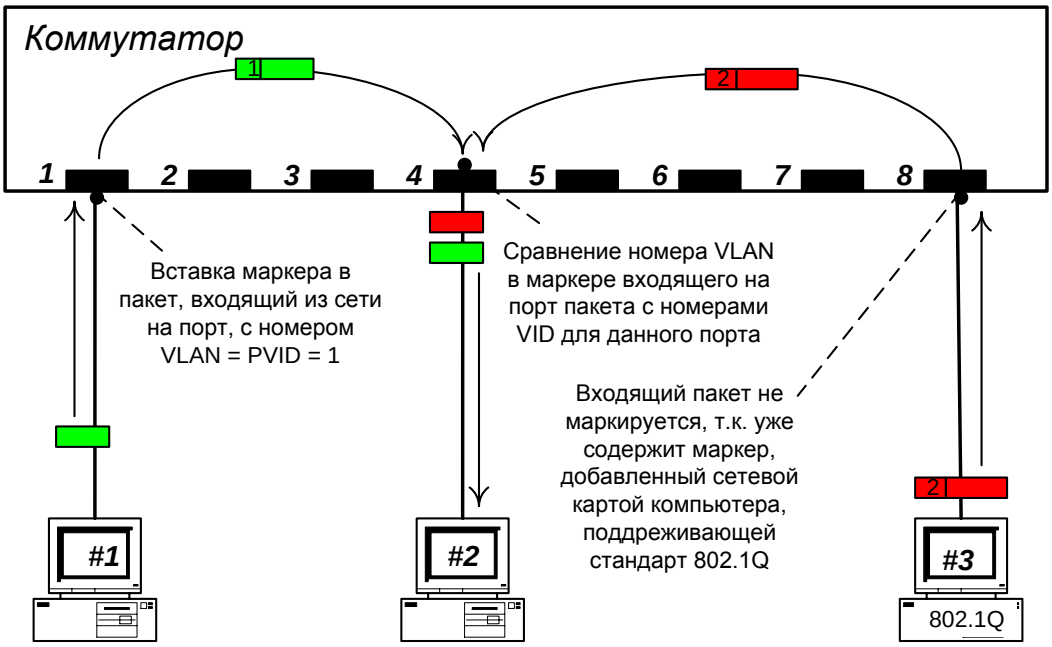
Обе переменные назначаются для одного и того же порта коммутатора, но между ними имеется существенная разница.

Пользователь может назначить ТОЛЬКО ОДИН номер PVID для каждого порта. Номер PVID определяет, к какой VLAN принадлежит данный порт. Когда ИЗ СЕТИ на порт коммутатора поступает НЕМАРКЕРОВАННЫЙ пакет, то принадлежность данного пакета к определенной VLAN осуществляется как раз на основе номера PVID порта. Номер PVID вставляется в заголовок кадра Ethernet и идентифицирует членство пакета в одной определенной VLAN. Необходимо отметить важный момент: если пакет уже содержит маркер (заданный явно сетевой картой компьютера или заданный неявно на другом коммутаторе), то повторная маркировка пакета, поступающего из вне на какой-либо порт коммутатора, НЕ производится.

С другой стороны можно задать МНОЖЕСТВО идентификаторов VID для одного и того же порта. Данные идентификаторы определяют, пакеты каких виртуальных сетей может принимать данный порт. Данная проверка осуществляется, когда пакет УЖЕ ПОЛУЧЕН ИЗ СЕТИ и МАРКИРОВАН, во время процедуры перенаправления пакета с порта на порт ВНУТРИ коммутатора.

Построение виртуальных сетей 802.1Q на одном коммутаторе

Рассмотрим следующую модель (рисунок 4.9), которая помогает понять принцип работы виртуальных сетей 802.1Q. В данном примере порт #1 является членом VLAN #1, а порты #4 и #8 являются членами VLAN #2. При этом порт #1 может принимать входящие пакеты только из своей VLAN, то есть только из первой VLAN. Порт #4 может принимать входящие пакеты из обоих VLAN (первой и второй). Порт #8, также как первый порт, может принимать входящие пакеты только из своей VLAN (второй). В результате полноценно функционировать в данной сети смогут только машины #2 и #3. Машины #1 и #2 не смогут обмениваться информацией, так как порт #4 может принимать пакеты с порта #1, но не наоборот.



PVID	Номер порта
1	1
2	4
2	8

VID	Номера портов
1	1,4
2	4,8

Рисунок 4.9. Модель функционирования виртуальной сети 802.1Q на одном коммутаторе.

Построение виртуальных сетей 802.1Q на нескольких коммутаторах

Как уже отмечалось, виртуальные сети 802.1Q могут быть построены на нескольких коммутаторах и охватывать всю локальную сеть. При этом все коммутаторы сети должны поддерживать стандарт 802.1Q. Необходимо понять следующие важные термины:

- Маркировка (Tagging) – процесс вставки информации о виртуальной сети 802.1Q в заголовок кадра Ethernet. Порт, для которого включен режим маркировки, будет оставлять без изменения заголовок ИСХОДЯЩЕГО С КОММУТАТОРА В СЕТЬ пакета Ethernet, то есть будет оставлять в нем маркер 802.1Q. Подобная возможность необходима для связи двух устройств, поддерживающих стандарт 802.1Q.
- Демаркировка (Untagging) – процесс удаления информации о виртуальной сети 802.1Q из заголовка кадра Ethernet. Порт, для которого включен режим демарки-

ровки, будет удалять маркер из заголовка пакета, ИСХОДЯЩЕГО С КОММУТАТОРА В СЕТЬ. Подобная возможность необходима для связи коммутатора и устройства, не поддерживающего стандарт 802.1Q.

- Входящий порт (Ingress Port) – порт коммутатора, на который поступают пакеты из сети. Для каждого порта можно настроить фильтр Ingress Filter. Если этот фильтр отключен (состояние Disabled) то порт функционирует в обычном режиме, то есть проверка пакета на принадлежность какой-либо VLAN (сравнение номера VLAN пакета с номерами VID порта) производится при поступлении пакета на данный порт с другого порта ВНУТРИ коммутатора (как это показано на рисунке 4.9). При включении фильтра (состояние Enabled) данная проверка ДОПОЛНИТЕЛЬНО производится для любого маркированного пакета, входящего на данный порт ИЗ СЕТИ.
- Исходящий порт (Egress Port) – порт коммутатора, с которого пакеты выходят в сеть и при этом необходимо принять решение о маркировке/демаркировке пакета.

С учетом всего ниже приведена расширенная модель функционирования виртуальной сети 802.1Q на основе нескольких коммутаторов (рисунок 4.10).

Преимущества виртуальных сетей 802.1Q

Дополнительное преимущество VLAN 802.1Q заключается в том, что можно изменять топологию сети без физического перемещения станций или изменения кабельных соединений. Станции можно назначить другую VLAN и, соответственно, общаться и совместно использовать ресурсы с членами в новой VLAN просто изменив настройки порта с одной VLAN (например, VLAN отдела продаж) на другую (VLAN отдела маркетинга). Таким образом, VLAN обеспечивают гибкость при перемещениях, изменениях и наращивании сети. Следует отметить, что в современных коммутаторах поддерживается единственный тип VLAN – тэгированные виртуальные сети.

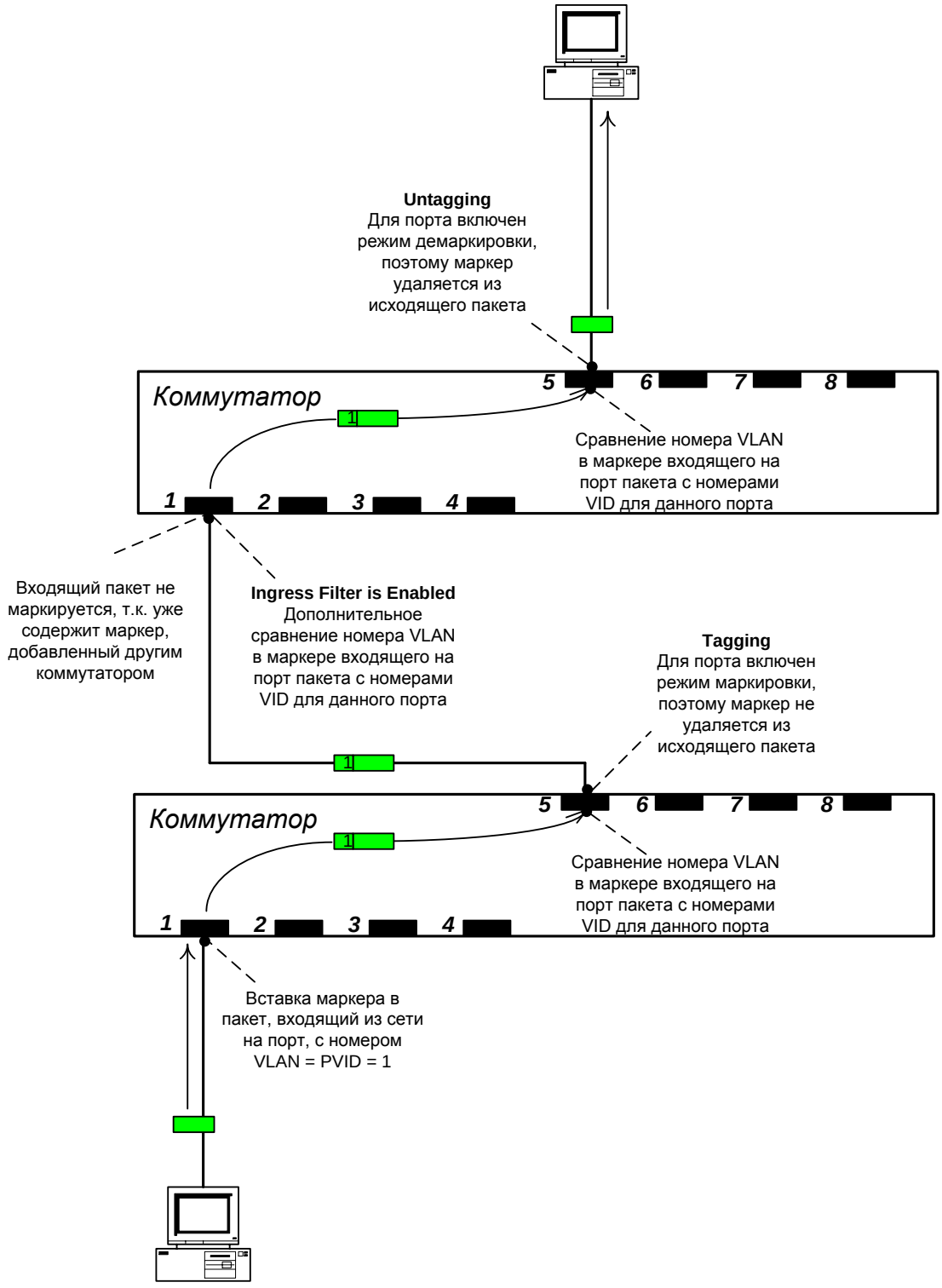


Рисунок 4.10. Модель функционирования VLAN 802.1Q на нескольких коммутаторах.

9. Аудит безопасности протокола связующего дерева STP

9.1. Обзор протокола Spanning Tree

Основное назначение протокола Spanning Tree – построение топологии ЛВС без избыточного дублирования соединений или «закольцовывания», недопустимых в силу логики работы алгоритма прозрачного моста (Transparent Bridge), являющимся основным для всех коммутаторов Ethernet. STP позволяет организовать отказоустойчивую архитектуру Ethernet-сети и определён стандартами IEEE 802.1d/s/w.

Используя STP, вы можете построить сеть, в которой существует несколько параллельных путей, и гарантировать при этом, что:

- ✓ резервные пути прохождения трафика при нормальном функционировании основного пути заблокированы;
- ✓ один из резервных путей активизируется при нарушении основного пути.

Протокол разработан в начале 90-х годов XX века (ANSI/IEEE 802.1D 1993 Edition) и дорабатывался в 1996 и 1998 годах. В качестве модели используется граф в виде дерева. Построение дерева сети начинается с корня (root) – устройства, выигравшего выборы корневого (designated root).

Важно заметить, что Spanning Tree Protocol определен для различных сред передачи данных (MAC, Media Access Control). Далее в тексте рассматриваются примеры сетей в основном на базе Ethernet, как наиболее распространенной.

BPDU – это именно пакеты, а не фреймы. BPDU пакеты инкапсулируются в используемый данной топологией тип фрейма с мультикастным MAC адресом в поле назначения, что обуславливает передачу этих пакетов через неинтеллектуальное оборудование, не знающее о существовании Spanning Tree.

Работа STP подразумевает выполнение следующих условий:

1. Возможность передачи информации между мостами, что осуществляется путем передачи каждым STP-совместимым устройством специальных блоков данных – Bridge Protocol Data Units (BPDU). Эти блоки данных передаются в пакетах с определенным групповым адресом назначения (multicast address).
2. Один из мостов функционирует как "ведущее" устройство, называемое Designated Root Bridge.

Designated root присутствует всегда, даже когда топология не содержит физических колец. Если в сети только одно STP-совместимое устройство, то до тех пор, пока STP не выключен, оно будет анонсировать себя. Каждое STP-совместимое устройство начинает работу, считая себя designated root, так что в сети с единственным STP-совместимым устройством оно и является Designated Root Bridge.

Если же в сети присутствует более одного устройства, поддерживающего STP, то Designated Root Bridge выбирается путем голосования (выборов) на основе значения параметра Bridge Identifier, обычно являющегося комбинацией уникального MAC-адреса моста и устанавливаемого для моста приоритета. На роль Designated Root Bridge назначается мост с наименьшим значением Bridge Identifier.

Для всех остальных мостов в сети определяется корневой порт Root Port, то есть порт моста, ближайший к Root Bridge. Основной величиной, влияющей на процесс выборов,

является «стоимость пути до корня» Root Path Cost. От других портов, соединенных с root bridge непосредственно или через другие мосты, корневой порт также отличается своим идентификатором, который содержит его номер и "вес", который может быть задан администратором.

Помимо Designated Root Bridge в STP вводится логическое понятие Designated Bridge. Владелец этого статуса считается главным в обслуживании данного сегмента ЛВС. Статус Designated Bridge также является выборным и может переходить от одного устройства к другому.

Аналогичным образом вводится выборное логическое понятие Designated Port – порта, который обслуживает данный сегмент сети. Для Designated Port вводится понятие Designated cost, описывающее стоимость пути.

В зависимости от содержимого получаемых от соседних устройств конфигурационных пакетов то или иное устройство перестает оспаривать статус Designated Root Bridge и начинает анонсировать устройство, выигравшее этот статус в процессе выборов. Так происходит до тех пор, пока ситуация не придет к завершающей стадии, которая характеризуется следующим образом:

1. В сети только одно устройство, считающее себя корнем, а остальные устройства периодически анонсируют его как корень, что поддерживает статус кво, обновляя таймеры на всех STP-совместимых устройствах.
2. Root Bridge периодически посылает во все свои порты пакеты с BPDU. Интервал времени, через который происходит посылка, называется Hello Time.
3. В каждом сегменте сети имеется единственный Designated Bridge Port - порт, через который проходит обмен трафиком с Root Bridge. Этот порт имеет наименьшее значение Root Path Cost по сравнению с другими портами в сегменте, либо меньший bridge ID.
4. BPDU принимаются и отправляются STP-совместимым устройством на всех его портах, даже на тех, которые были "выключены" работой STP. Однако BPDU не принимаются на портах, которые были "выключены" администратором.
5. Каждый мост осуществляет пересылку (forwarding) пакетов только между Root Port и портами, которые являются Designated Bridge Port для соответствующего сегмента. Все остальные порты находятся в состоянии "Blocking".

Как уже говорилось, корень не несет на себе иных обязанностей, кроме анонса своих параметров согласно спецификации Spanning Tree протокола, а это, в частности, значит, что в сети, не содержащей физических колец или избыточных соединений, от изменения владельца статуса Designated Root Bridge пути пакетов, пересылаемых между двумя произвольными станциями не изменится.

Спецификация протокола не накладывает ограничений на время, в которое могут начаться выборы того или иного параметра, а лишь описывает необходимые для этого условия.

К числу ситуаций, в которых возникают выборы как минимум одного из параметров, относятся:

- ✓ подключение в сеть нового STP-совместимого устройства;

- ✓ устаревание имеющихся данных (например, в результате выхода из строя или отключения одного из устройств, участвовавших в создании дерева) – определяется истечением тайм-аута;
- ✓ административная акция, изменяющая топологию сети.

В качестве параметров при выборах используются следующие величины:

1. Постоянные величины:
   - ✓ root path cost – стоимость пути к корневому устройству. Чем меньше значение этой величины, тем выше приоритет устройства;
   - ✓ bridge identifier – идентификатор устройства. Чем меньше значение, тем выше приоритет;
   - ✓ port identifier – идентификатор порта. Чем меньше значение этой величины, тем выше приоритет порта по сравнению с другими портами в этом же устройстве.
2. Выборные величины:
   - ✓ designated port – назначенный порт;
   - ✓ designated root – назначенный корень;
   - ✓ designated cost – назначенная стоимость.

Согласно спецификации выборные величины, в зависимости от контекста, могут принимать участие в различных выборах, например, designated cost может принимать участие как в выборах назначенного моста, так и в выборах корневого моста.

Во время работы устройства анонсируют себя и параметры своих портов через Configuration BPDU (далее называемые c-bpdu). Для сообщения об изменениях в топологии используются topology change notification BPDU (TCN-BPDU).

Собственно информации передается совсем немного – только статус. По факту прихода такого BPDU любой коммутатор уменьшает время жизни записей в своей таблице коммутации до минимума, определенного стандартом.

В момент получения конфигурационного bpdu STP-совместимый мост может определить, что пришедшая информация должна обновить имеющуюся у него информацию. В этом случае устройство изменяет содержимое анонсируемых им bpdu так, чтобы анонсировать выигравшее устройство, а не себя. Обратите внимание, что BPDU не проходят сквозь устройства, которые поддерживают STP - они лишь, при определенных условиях, инициируют на "промежуточном" устройстве посылку собственных BPDU и/или изменение содержимого BPDU посылаемых "промежуточным" устройством.

В момент обновления конфигурации STP-совместимое устройство изменяет состояние своих портов – они поочередно принимают одно из следующих возможных значений:

- ✓ блокирован (Blocking). Порт заблокирован, но фреймы, содержащие STP пакеты (bpdu), принимаются и обрабатываются, в отличие от пользовательских фреймов;
- ✓ слушает (Listening) - первый этап подготовки к состоянию Forwarding. Фреймы, содержащие STP пакеты (bpdu), принимаются и обрабатываются, в отличие от пользовательских фреймов;
- ✓ обучается (Learning) – второй этап подготовки к состоянию Forwarding. Фреймы, содержащие STP пакеты (bpdu), принимаются и обрабатываются, в отличие от пользовательских фреймов;
- ✓ передает (forwarding) – рабочее состояние портов устройства. Передаются как фреймы, содержащие STP пакеты, так и фреймы пользовательских протоколов.

Время, на которое порт попадает в то или иное состояние, определяется значениями пауз, которые передаются посредством C-BPDU вместе с остальными параметрами.

Согласно стандарту, во время нахождения в состояниях Blocking, Listening и Learning пересылка не-STP пакетов не разрешена. В то же время, bpdu-пакеты не обрабатываются только в случае если порт выключен администратором. Это говорит о том, что если порты находятся в режимах Blocking, Listening или Learning - сеть работает только на STP протокол, но не на пользователя. Этим можно воспользоваться для организации атаки "отказ в обслуживании" (Denial of Service, DoS).

Собственно, причина, по которой протокол работает именно таким образом, проста – необходимость избежать хаотичного дублирования и/или размножения пакетов в различные сегменты сети в момент перестройки ее топологии. Дублирование и/или размножение пакетов возможно, например, за счет возникновения временных колец, возникающих в процессе переключения состояния того или иного канала.

В силу того, что топология сети определяется при участии протокола STP возможна атака по схеме ложный объект PBC с навязыванием ложного пути.

Вообще говоря, описываемые на примере Ethernet уязвимости протокола относятся к любым топологиям, на которых может работать STP. Как следствие, протокол и рассматриваемые атаки, требующие включенной поддержки STP, могут быть применены на любых сетевых топологиях, не допускающих логических колец. В стандарте IEEE 802.1d специально обращено внимание на то, что STP может работать не только поверх Ethernet, но и поверх других сетей с другим принципом MAC.

Для большей наглядности и лучшего понимания работы STP ниже приведён пример (рисунок 4.11).

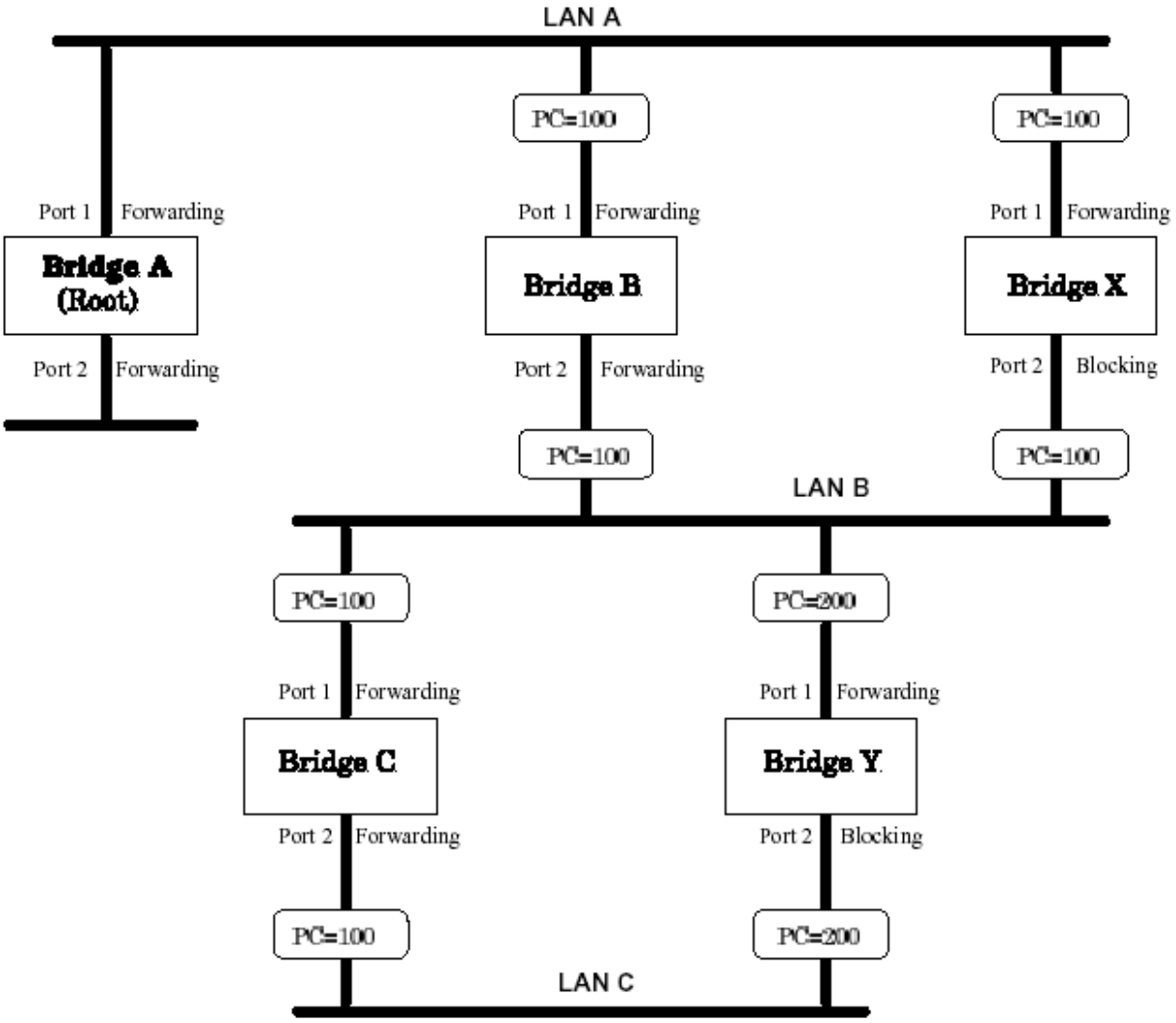


Рисунок 4.11. Пример работы протокола STP.

Каждый порт каждого моста имеет свою величину Path Cost, обозначенную как PC=XXX. Мост A является Root Bridge, так как имеет наименьшее значение Bridge Identifier. Для сети A (LAN A) Designated Bridge Port является порт 1 моста A. Один из портов каждого из четырех оставшихся мостов является Root Port (это порт, ближайший к Root Bridge).

Мосты X и B предоставляют доступ к сети B с одинаковым параметром Path Cost. Однако, в данном случае, порт моста B выбран в качестве Designated Bridge Port, так как этот мост имеет меньшее значение Bridge Identifier.

Порт моста C выбран в качестве Designated Bridge Port для сети C, так как он предоставляет более "дешевый" доступ к этой сети (стоимость пути через мосты C и B равна 200, а через мосты Y и B – 300).

В стабильном состоянии все мосты ожидают периодической посылки Root Bridge специальных пакетов – Hello BPDU. Если в течение промежутка времени, определяемого значением Max Age Time, таких пакетов от Root Bridge не поступает, мост считает, что либо между ним и Root Bridge нарушена связь, либо последний отключен. В этом случае мост инициирует реконфигурацию топологии сети. Путем установки соответствующих параметров можно регулировать, насколько быстро мосты будут обнаруживать изменения в топологии и задействовать запасные маршруты. На практике с регулировкой значений STP приходится сталкиваться в основном в двух случаях - если в силу обстоятельств STP

переводит в режим "запасного" более быстрый линк и если сеть имеет плохую сходимость на значениях по умолчанию из-за большого количества устройств.

9.2. Протокол STP и виртуальные сети VLAN

В случае, если часть портов коммутатора находится в одной VLAN, а другая часть – в другой, то тогда, с точки зрения нормального функционирования сети, можно рассматривать эту ситуацию как два независимых коммутатора. Это утверждение действительно справедливо для пользовательского трафика, однако оно не совсем верно для ситуации с STP и BPDU, обеспечивающих функциональность, требуемую алгоритмом Spanning Tree.

Миф о раздельном дереве STP в каждой VLAN

Надо отметить, что наиболее "продвинутые" устройства, поддерживающие VLAN (но не все), поддерживают отдельное STP дерево на каждую VLAN (стандарт IEEE 802.1w). Таким образом, все STP атаки на такие устройства, на первый взгляд, будут эффективны только в пределах данной VLAN. С точки зрения STP внутри VLAN имеется стандартная схема функционирования портов – каждый порт, согласно протоколу, получает и отправляет BPDU и, в зависимости от их содержимого, изменяет свое состояние. Тем не менее, часть устройств, поддерживающих VLAN, имеет единственное Spanning Tree дерево на весь коммутатор (например, модели Intel 460T), что делает всю сеть, построенную на таких устройствах, уязвимой ко всем STP атакам.

С точки зрения обсуждаемых проблем, устройства второго типа с единственным на все устройство STP-деревом по части функционирования STP атак ничем не отличаются от рассматриваемого ниже случая без поддержки VLAN, так что мы не будем рассматривать отдельно этот вариант. Со вторым типом устройств, поддерживающих собственное STP-дерево на каждый VLAN, дела обстоят немного по-другому.

Миф о физической раздельности виртуальных сетей VLAN

Port-based VLAN можно объединить, просто соединив патч-кордом соседние порты, подключенные в разные VLAN. После этого порты, объединенные в VLAN'ы, с точки зрения коммутатора по-прежнему будут состоять в разных VLAN до тех пор, пока не используется протокол GVRP. Однако, для корректного функционирования STP описанная ситуация должна отслеживаться особенным образом, то есть, с точки зрения STP, если два изначально "независимых" дерева вдруг соединились, то появился повод для STP-выборов. Действительно, аналогичная ситуация возникала на практике и алгоритм STP срабатывал. С учетом возможности конвертации VLAN ID становится понятно, что на самом деле независимость STP дерева в пределах VLAN – это миф. Возникает идея: если сгенерировать STP BPDU с параметрами, аналогичными тем BPDU, которые распространяются в соседнем VLAN, можно заставить STP устройство устроить перевыборы с участием интерфейса, находящегося в соседнем VLAN. Правда есть еще один вариант передачи BPDU, который позволил бы отслеживать ситуацию с VLAN, соединенными через нетегируемый порт, а именно – передавать BPDU всегда в тегируемом фрейме и игнорировать нетегируемые порты.

9.3. Возможные схемы атак

Идея этих атак лежит на самой поверхности. Поскольку предметом атаки могут быть любые "выборные" значения, можно выделить несколько типов атак по используемым методам их проведения:

1. Инициация перевыборов назначенного корня для всей сети.
2. Инициация перевыборов назначенного моста для сегмента сети.
3. Инициализация перевыборов назначенного порта для сегмента сети.

Случай 3 часто сопутствует случаю 2. Здесь и далее под "лучшим" понимается такое значение параметра, содержащегося в BPDU, которое позволяет добиться рассматриваемой цели. Чаще всего под "лучшим" будет пониматься такое значение параметра, которое обеспечит победу в STP-выборах. В качестве побочных эффектов STP-атак могут образовываться "пакетные штормы" внутри сети (в большинстве случаев - недолговременные).

BPDU spooffing

Тема с общим названием "spoofing" разжевана во многих источниках настолько хорошо, что, казалось, ничего нового сказать нельзя. Однако в применении к STP и с учетом уровня OSI, на котором работает Spanning Tree, у этой методики есть несколько принципиальных особенностей. Говоря кратко, spooffing – это подмена. Наиболее типична подмена адреса отправителя. Типичные способы борьбы с этой атакой в Internet заключаются во:

- ✓ введении граничных условий;
- ✓ введении дополнительных требований в протокол обмена между отправителем и получателем;
- ✓ введении требования установления соединения между участниками обмена трафиком (это частный случай 2).

Особенностями STP, важными с точки зрения детального рассмотрения BPDU spooffing, является следующее:

1. Это протокол 2-го уровня OSI, который не маршрутизируем.
2. C-BPDU пакеты этого протокола ходят исключительно между двумя STP-совместимыми устройствами. Любое STP-совместимое устройство, получив такой пакет, генерирует в качестве реакции свой (то есть другой) пакет, ограничивая таким образом зону хождения C-BPDU.
3. Этот протокол не предполагает каких-либо административных ограничений на его использование.
4. Этот протокол не предусматривает входной фильтр хождения BPDU по интерфейсу, с которого оно получено, поскольку сам факт получения BPDU на том или ином интерфейсе является событием протокола. Исключения возможны при поддержке расширений, например, Cisco BPDU Root Guard.
5. Этот протокол предусматривает динамическое конфигурирование. Причем, ограничений на факт и время появления в структуре нового устройства не предусмотрено.
6. Это протокол без установки соединения (connectionless protocol).

Из пунктов 1 и 4 следует, что граничные условия фильтрации ложных пакетов к нему не применимы. По крайней мере, до тех пор, пока нет уверенности, что на определенных интерфейсах получение BPDU принципиально невозможно (или не приемлемо по каким либо соображениям). Такую уверенность может иметь администратор, однако сам по себе стандарт в данном случае не предусматривает возможности отключения STP поинтерфейсно – в таком поведении стандарта есть смысл только до тех пор, пока безопасность имеет меньший приоритет, чем удобство.

Из пункта 2 следует, что фильтрация пакетов по пути неким STP-совместимым устройством не имеет смысла, поскольку путь STP-пакета не может содержать промежуточное STP устройство. Пункт 3 говорит сам за себя. Пункт 6 указывает на то, что подделка соединения не потребуется, а между тем она затрудняет, например, TCP/IP spooffing. Впрочем, введение требования установки соединения в данном случае поможет мало, поскольку нельзя ограничивать возможность установки такового.

В результате ситуация с BPDU spooffing такова: при правильно поставленном BPDU-spooffing принципиально невозможно отличить "легальные" пакеты от поддельных. Под правильно поставленным BPDU-spooffing понимается подделка не только параметров STP-пакета, но и адресов в MAC фрейме. При этом, благодаря пункту 5, даже если не использовать собственно spoofing, можно все равно добиться интересных результатов.

Играя с времяобразующими параметрами Max Age, Forward Delay, Hello Time в фальсифицированных пакетах C-BPDU можно управлять, по крайней мере, близлежащими STP-устройствами посредством постоянной и очень быстрой посылки таких C-BPDU. Под "фальсифицированными bpdu-пакетами" в данном случае понимаются c-bpdu с такими же параметрами, как у текущего designated bridge или designated root bridge, и с тем же, что и у него MAC-адресом отправителя, но с другими значениями параметров, определяющих различные паузы в рамках протокола, что может вызвать переконфигурацию STP-дерева с последующим частичным DoS-атаками. Для того, чтобы этого избежать, фальсифицированные пакеты не должны вызывать реконфигурацию дерева. Этого можно добиться, выставив в них port id в большее значение (согласно протоколу при этом должен выключиться этот интерфейс), или же выставить худший path-cost – в этом случае STP выключит этот порт, но пакеты BPDU, тем не менее, он будет принимать. Однако само по себе это мало что дает: все, что можно получить, меняя эти параметры, – это незначительная "дезориентация" ближайшего свитча и свитчей, который получают информацию от уже "обманутого" (те, что находятся ниже в STP-дереве). То есть максимум, что это может дать – увеличение времени жизни некорректной структуры дерева после того, как перестанут передаваться пакеты согласно этому атакующему алгоритму.

Provocation Aging

Возможна подделка вообще любых пакетов bpdu: не только configuration-bpdu (c-bpdu), но и notification bpdu (n-bpdu, tcn-bpdu). Например, используя tcn-bpdu, можно реализовать еще одну атаку – "provocation aging" – управление величиной времени хранения информации о положении в сети MAC-адресов, динамически формируемой в процессе работы устройства. Величина Aging Time может быть сброшена не ранее, чем через 10 секунд.

Сама по себе эта атака малофункциональна, но, теоретически, может быть использована для других атак, при которых одним из требований является внесение ложных записей в таблицу коммутации в качестве катализатора, обеспечивающего повышение скорости срабатывания такой атаки. Возможно, использование этой методики для помощи в provocation sniffing поможет дать боле-менее существенные результаты. Кроме того, эта атака может быть использована в случае поддержки STP-совместимым устройством расширений STP, например STP port fast (Cisco). В таких случаях aging time может быть сброшено до нулевой величины. В таком случае атака provocation aging станет синонимом provocation sniffing.

BPDU filter

Очевидно, что осуществить DoS в рамках одного коммутатора очень легко. Для этого необходимо организовать петлю, которая послужит причиной лавинного размножения и саморегенерации пакетов. Данная атака, разумеется, эффективна в пределах одной виртуальной сети vlan, однако, она затронет и остальные за счет снижения производительности всего устройства. Это происходит из-за повышения накладных расходов на обработку постоянного большого паразитного трафика, объем которого может вырасти до суммарной пропускной способности образованных между устройствами каналов связи. Идея этой атаки крайне проста: создается дополнительный канал связи между двумя STP-совместимыми устройствами, в середину которого ставится фильтр BPDU, который уда-

ляет пакеты BPDU с любыми адресами, не влияя на остальной траффик, либо, дополнительно создавая его. В силу своей неэлегантности, и в силу того, что для реализации обязательно требуется доступ к включенным портам хотя бы одного из коммутаторов этот метод является мало интересным с практической точки зрения.

Отказ в обслуживании (DoS)

Для того чтобы устроить DoS-атаку можно воспользоваться тем, что STP-совместимые устройства в момент реконфигурации работают не на пользователя, а лишь на создание Spanning Tree дерева. Поскольку реконфигурация может быть вызвана, в том числе, появлением нового STP-совместимого устройства, можно имитировать периодически появление нового устройства с параметрами, которые будут лучше установившихся, что вызовет реконфигурацию (перевыборы) одного из выборных параметров. Некоторые устройства будут пытаться пересматривать состояние всех портов при появлении любого нового устройства даже с худшими (с точки зрения выигрыша в выборах), чем у них параметрами. На более «интеллектуальное» железо эта атака может действовать исключительно в случае появления устройства с лучшими (с точки зрения STP) параметрами – в худшем случае может перейти в состояние реконфигурации только один порт. Однако даже такая ситуация будет противоречить спецификации протокола STP, в котором не предусмотрена ситуация переконфигурации устройства, к которому подключают другое с худшими (с точки зрения STP) параметрами. С использованием STP возможны следующие схемы реализации DoS.

STP DoS: "вечные выборы" или постоянный перебор

Наиболее эффективный метод: подождать появления STP пакета с текущим значением STP-root, затем по очереди перебирать значения bridge id, посылая bpdu с все меньшими значениями (id=id-1), до тех пор пока не будет достигнуто предельное значение, вызывая, таким образом, перевыборы designated root каждым посланным пакетом (для большей уверенности – посылать одинаковые c-bpdu несколько раз подряд). Когда же будет достигнуто минимально возможное значение, подождать, пока это значение не устареет из-за паузы, и начать сначала. С учетом того, что все параметры, включая время устаревания, устанавливаются в посылаемых назначенным корнем конфигурационных bpdu, можно получить ситуацию, при которой порты никогда не войдут в состояние "forwarding" пока происходит генерация фреймов, обеспечивающих отказ в обслуживании. Более того, в силу особенностей протокола состояние отказа в обслуживании будет продолжаться еще некоторое время, равное параметру Max Age, который можно выставить согласно стандарту до 40 секунд (если программисты не забыли поставить знак неравенства. А если забыли – вплоть до максимальной величины, вмещающейся в отведенную под этот параметр область памяти). Опять же, разные производители зачастую совершенно по-своему соблюдают совместимость с тем или иным IEEE стандартом

Поскольку помимо 65535 возможных приоритетов bridge id включает в себя еще и MAC адрес, то количество времени, которое потребуется чтобы перебрать все возможные значения весьма велико. На самом деле при зацикливании алгоритма состояние DoS может продолжаться сколь угодно долго, пока посылаются пакеты с фальшивыми BPDU.

STP DoS: алгоритм "исчезновения корня"

Для реализации этой атаки необходимо начать посылку BPDU с минимально возможным bridge id, то есть с максимально возможным приоритетом, периодически переставая передавать конфигурационные bpdu, чтобы это значение назначенного корня устарело и так по кругу. На первый взгляд это не настолько эффективно, поскольку может существовать небольшой промежуток времени, когда сеть работает, тем не менее, в силу того, что ог-

раничения, накладываемые спецификацией протокола, позволяют устанавливать время устаревания значений в очень широких пределах, этим методом можно достигнуть точно такой же эффективности. Более того, этот метод наиболее прост в реализации, поскольку не требует от злоумышленника ни знания текущего идентификатора назначенного корня, ни каких-либо предположений относительно его величины (в отличие от предыдущего случая).

Следует заметить, что если в некоторой ЛВС текущий идентификатор designated root не является идентификатором наивысшего возможного приоритета (имеется ввиду нулевые и Bridge Priority и MAC address), то эта ЛВС, скорее всего, подвержена обоим типам DoS в полной мере. В случае же, если на одном из STP-совместимых устройств установлен bridge id равный 0, то DoS, тем не менее, может быть реализована, но только на части устройств, а именно на тех, к которым подключен атакующий. Дело в том, что ситуация, когда мы имеем два STP-совместимых устройства с одним и тем же bridge id может быть воспринята как кольцо, и STP отключит либо порт атакующего, либо другой порт, в зависимости от соотношения номера порта, на котором производится атака, и номера порта к которому подключена атакуемая часть сети.

STP DoS: алгоритм случайного совпадения

Этот подраздел касается случаев сетей с использованием технологии Port-based VLAN. В случае такой организации сети описанные до этого момента DoS атаки могут сработать исключительно в пределах одного VLAN. Однако и в этом случае злоумышленник может наделать неприятностей всей сети – дело в том, что STP может выделяться в отдельное дерево посредством фильтрования по Bridge ID. Поэтому, если в рамках данного VLAN отправлять BPDU от имени имеющегося в соседнем VLAN designated root или просто designated bridge, коммутаторы можно убедить в том, что в некотором месте виртуальные сети стали вдруг соединены. Это вызовет перевыборы designated bridge, в том числе и в сегменте, который атакуется и принадлежит VLAN, отличной от VLAN атакующего.

Этот алгоритм назван алгоритмом случайного совпадения потому, что атакующему придется подбирать designated root bridge ID именно такой, который совпадет с имеющимся в соседней VLAN. Разумеется, наиболее точно эта атака воспроизводится в случае, если атакующий может получить «дамп» работы сети за время порядка Hello Time.

STP DoS: частичный отказ в обслуживании

В дополнение к сказанному отдельно следует рассмотреть атаки, которые могут произвести отказ обслуживания в пределах некоторого сегмента ЛВС, тем самым создав ситуацию "частичного отказа в обслуживании".

В случае, если атакующему каким-то образом, стало известно значение bridge identifier одного или более устройств в ЛВС, он может вызвать выборы designated bridge для того или иного участка ЛВС, послав соответствующее c-bpdu для оспаривания статуса назначенного моста для соответствующего сегмента ЛВС. Воздействуя исключительно на ближайшее к атакующему устройство, он может объявить себя лучшим возможным путем к соседнему с ним устройству, например, заявив, что его порт является самым дешевым по стоимости путем к нему, в результате чего "более дорогой" порт будет переведен в состояние Blocking, а поскольку реальной связи нет, все оконечные устройства, подключенные к атакованному устройству, потеряют возможность работать с отключенным сегментом ЛВС. Данная атака может оказаться невозможной без физического подключения к соответствующему сегменту.

В описанном на рисунке 4.12 случае в примере B), псевдо-бридж-система атакующего, анонсируя лучшие параметры, выиграла выборы назначенного моста (designated bridge), не являясь таковым. Одновременно порт, к которому подключен атакующий, стал назначенным портом для атакуемой части сети (статус перешел от порта перешедшего в состояние Blocking). Следует отметить, что при переходе статуса назначенного моста для сегмента в результате STP атаки, в случае, если оспаривавшие этот статус устройства были подключены к разным портам, изменяется и назначенный порт. С точки зрения пользователя сети сегмент "Victim LAN" перестает "видеть" остальную сеть (все, что подключено через Bridge 1), при этом атакующий (attacker) продолжает видеть все сегменты сети, кроме сегмента "Victim LAN". Объектом атаки является "Bridge 1", которому посылаются BPDU от имени "Bridge 2", но с "лучшими" (с точки зрения выигрыша STP-выборов) параметрами.

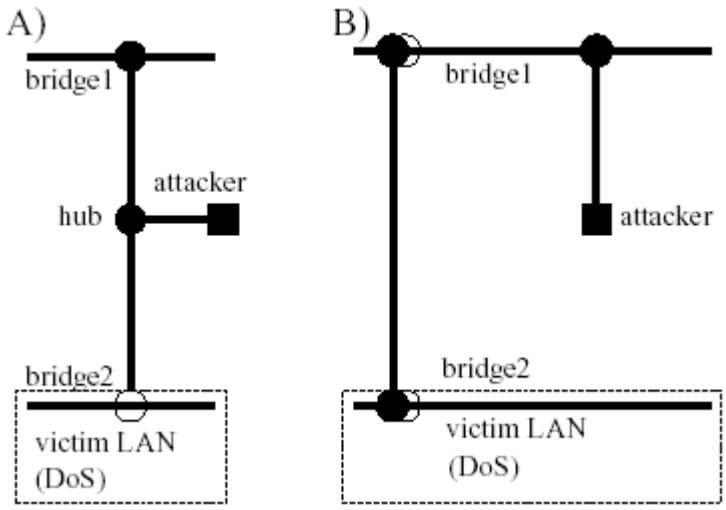


Рисунок 4.12. Частичный DoS (пример 1).

Пример A отличается от B тем, что будучи подключенным к тому же порту "Bridge 1" атакующий (attacker) также, как и жертва ("Victim LAN") перестает "видеть" остальную сеть. Объектом атаки является "Bridge 1", которому возвращаются BPDU, полученные от него же и измененные так, чтобы сэмулировать закольцовывание между этим и другим интерфейсом. При этом параметры таких BPDU должны быть подобраны именно так, чтобы выключился именно этот, а не соседний интерфейс.

Вариант A может быть также реализован в случае если атакующий (attacker) подключен не к "Bridge 1" через концентратор, а к "Bridge 2" - в этом случае объектом атаки будет "Bridge 1", которому для реализации частичной DoS по этой схеме необходимо присылать BPDU от имени "Bridge 1" с параметрами "лучшими", чем имеющееся подключение между "Bridge 1" и "Bridge 2". Собственно, по этой схеме можно добиться DoS для любого порта на том же STP-совместимом устройстве. Эти схемы проиллюстрированы рисунком 4.13. При разумном подборе параметров BPDU это становится возможным также и для портов на соседних STP-совместимых устройствах.

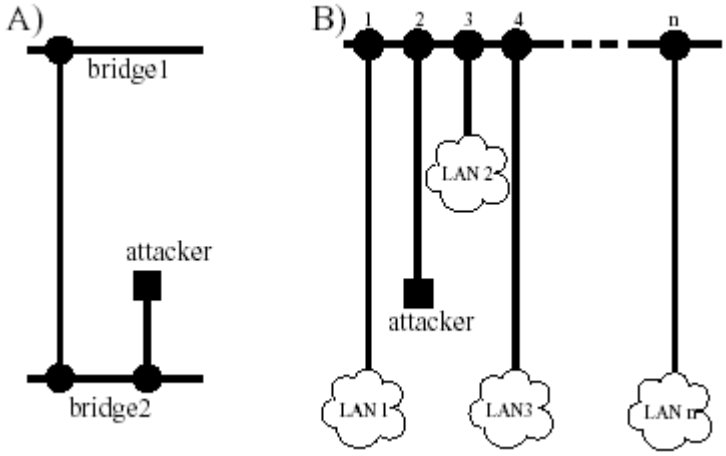


Рисунок 4.13. Частичный DoS (пример 2).

Более типичной будет схема, показанная на рисунке 4.14. В реальных условиях у злоумышленника редко есть возможность разорвать канал связи между коммутаторами, установить туда концентратор и вставлять в поток свои пакеты. Скорее возможен такой сценарий: сервера включены в один коммутатор, клиенты включены в концентраторы, подсоединенные к одному или нескольким коммутаторам. По аналогии атаки Митника на Шимомуру, атакующему надо "выключить" одного из участников соединения, а именно, сервер. Для этого он должен убедить коммутатор, к которому подключен концентратор с интересующим его клиентом, что он имеет лучший путь до второго коммутатора, к которому подключен сервер. В результате, канал связи между коммутаторами рвется, и атакующий может выдавать себя за сервер или просто злорадствовать от факта недоступности сервера клиенту. Разумеется, концентратор здесь не принципиален, хотя можно строить атаку, подобную этой, на всю сеть, не задумываясь над подсчетом величин для данного сегмента сети.

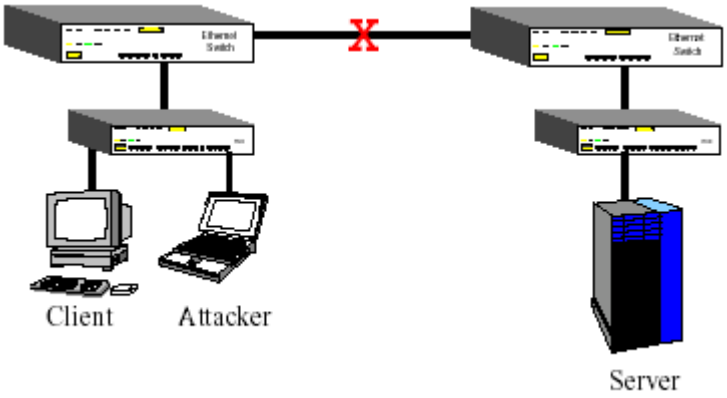


Рисунок 4.14. Частичный DoS (пример 3).

Рисунок 4.12a обращает внимание на следующую особенность: если считать, что нумерация портов идет слева направо, то атакующий, пытаясь устроить частичный DoS для канала связи между коммутаторами, может сделать его исключительно себе, если не подобрать параметры специальным образом. Дело в том, что, при прочих равных условиях, включенным останется порт с меньшим ID. Поэтому атакующий должен подобрать для фальсификации такой номер порта соседнего коммутатора, который заставит коммутатор "Bridge 2" выключить именно необходимый атакующему линк, а не тот, с которого идет атака. Для случая же с одним единственным STP-совместимым устройством (рисунок 4.12b) атака легко реализуема для всех ЛВС, подключенных к портам, большим по номеру. Однако при попытке реализовать ее по отношению к первому порту ничего не выйдет:

получив BPDU на большем по номеру порту от имени порта с меньшим номером, устройство выключит порт с большим номером.

Изменяя Max Age, forward delay, hello time в ложных c-bpdu, мы можем управлять как минимум ближайшими коммутаторами, изменяя их представление о топологии сети, а также (что принципиально при организации частичной DoS) – изменять тайм-ауты протокола, что позволяет нам выставить конфликтующие с остальной сетью значения.

Следует заметить, что в силу таймаутов реконфигурации в рамках протокола и того, что в момент реконфигурации пользовательские фреймы не передаются, применение данной методики для временного получения пакетов (например, части пакетов используемых при установлении соединения с помощью протоколов верхних уровней, например, IP) нецелесообразно, поскольку в момент изменения топологии эти пакеты блокируются.

Навязывание ложного маршрута (человек посередине), или "перетяни кольцо на себя"

Как уже упоминалось, в силу того, что топология сети определяется при участии протокола STP, возможна атака по схеме "ложный объект PBC с навязыванием ложного пути", известная по англоязычным публикациям как MitM (Man-in-the-Middle). Однако реализовать такую атаку можно только при определенных условиях конфигурации сети, а именно в тех случаях, когда жертвы, трафик между которыми надо перехватить (то есть, направить по каналу, который можно прослушивать), находятся в сети с двумя (как минимум) STP-совместимыми коммутаторами и подключены к разным коммутаторам. Причем, для реализации атаки в полной мере требуется два сетевых интерфейса, подключение которых необходимо осуществить так, чтобы образовать потенциальный дубликат имеющегося пути между источниками.

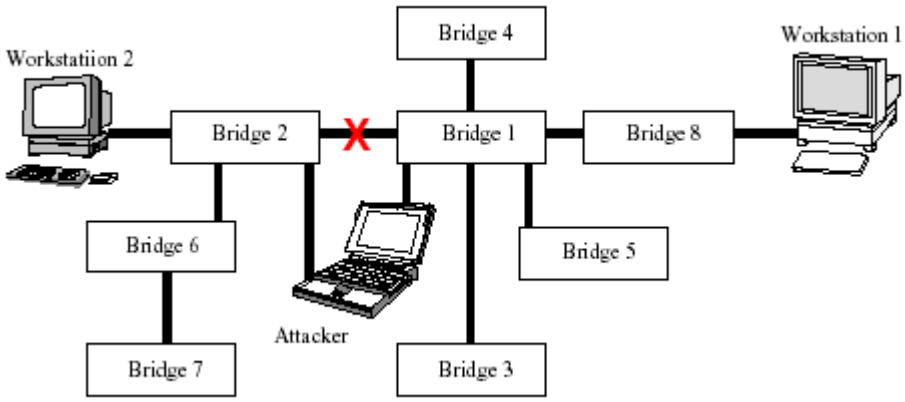


Рисунок 4.15. Пример атаки «Человек посередине».

Рассмотрим рисунок 4.15. На нем изображена сложная сеть, построенная на управляемых коммутаторах, поддерживающих протокол STP. Сеть показана в "устоявшемся" состоянии, то есть формирование топологии завершено, резервные линии переведены в состояние "blocking". В задачу злоумышленника входит осуществить перехват трафика между устройствами, подключенными к коммутаторам Bridge2 и Bridge8 (на рисунке изображены как Workstation1 и Workstation2). Результатом атаки будет перевод имеющихся соединений между Bridge1 и Bridge2 в состояние "blocking" (на рисунке отмечено крестиком) и перенаправление всего трафика между этими коммутаторами на интерфейсы атакующего. Для того, чтобы получить именно MitM, а не DoS, атакующий компьютер должен поддерживать работу обоих своих интерфейсов в режиме моста, что легко реализуемо, поскольку существует несколько свободно распространяемых реализаций bridging'a.

Заметим, что для этого даже не требуется поддержка STP ОС, работающей как мост. Более того, такая полнофункциональная поддержка только повредит, поскольку принцип MitM атаки при использовании двух сетевых интерфейсов базируется на том, что атакующий может посылать BPDU пакеты от чужого имени (например, в Bridge1 от имени Bridge2) чтобы представить себя как наилучший путь хождения пакетов. При такой логике атаки полноценная поддержка STP только помешает атакующему, поэтому для реализации MitM по схеме на рисунке 4.15 необходимо отключать поддержку Spanning Tree, либо использовать программное обеспечение, в котором нет поддержки STP. Таким образом, машина, реализующая MitM, на самом деле не нуждается в полноценной поддержке bridging'a, а может функционировать как концентратор, с единственным исключением – транзитный STP-траффик должен отбрасываться и вместо него генерироваться собственный, анонсирующий себя как наиболее выгодный путь для пакетов. Требование наличия как минимум двух коммутаторов связано с тем, что в случае подключения к одному коммутатору анонсирование себя, как лучшего пути создаст тупиковый путь. К тому же для этого случая гораздо проще воспользоваться arp-poisoning.

Важно заметить, что все так однозначно и просто только в случае, если атакующий подключен к соседним коммутаторам. В случае же, если он подключен к коммутаторам непосредственно не соединенным, придется подбирать значения root id, возможно последовательным перебором, поскольку bpdu не передаются (в неизменном виде) дальше первого встреченного STP-совместимого устройства. Поэтому такая атака кажется не слишком часто применимой. В принципе, злоумышленнику не обязательно иметь подключение к коммутаторам являющимся частью разорванного STP-кольца - он может "растянуть" кольцо на себя, если в сети есть другие дублируемые соединения. Просто в случае подключения к коммутаторам, образующим кольцо, задачу организации MitM решить легче.
Описанная атака работает не на всех конфигурациях. Однако в случае, когда злоумышленник захватил контроль над компьютером, который по какой-либо причине подключен к двум или более коммутаторам различными интерфейсами, он имеет возможность ее реализовать.

Провокационный сниффинг

Провокационный сниффинг – это алгоритм атаки, который может быть использован при определенных настройках активного сетевого оборудования с целью подслушивания трафика, проходящего через его порты. Согласно стандарту после перевыборов Designated Bridge коммутатор должен обнулить свою таблицу коммутации (кроме статически заданных администратором значений), а в случае нулевой таблицы коммутации коммутатор некоторое время работает как концентратор. Грубо говоря, посредством провокационного сниффинга коммутатор «превращается» в концентратор посредством «провокации» – фальсификации события "произошло изменение топологии сети", после которого все динамические записи в таблице коммутации очищаются. Однако использование такой провокации с толком возможно только при одном условии: коммутатор не должен успеть «обучиться». Под «обучением» в данном случае понимается формирование таблицы соответствия портов и MAC адресов подключенных к ним устройств. Поскольку обучение коммутатора происходит практически сразу же по получении им кадра от подключенного устройства, реализация полноценного сниффинга становится нетривиальной задачей, сложность которой пропорциональна скорости трафика в данной сети. Отсюда следует, что существуют условия, при которых эта атака в принципе не реализуема в полном объеме. Например, если трафик между станциями, обмен которых необходимо "подслушать", использует более 49% полосы пропускания. Для 100% эффективности данная атака предполагает «бомбардировку» коммутатора STP пакетами хотя бы вдвое чаще, чем проходят пакеты между жертвами, для того, чтобы каждый новый приня-

тый пакет приходил на коммутатор с только что очищенной таблицей коммутации. С учетом того, что от порта коммутатора начинается коллизионный домен, полноценный сниффинг кажется еще более затрудненным, что, правда, не касается full-duplex портов. Для полноценного сниффинга без потери пакетов за счет обучения коммутатора суммарный траффик, проходящий через коммутатор в единицу времени, не должен превышать 48-49% пропускной способности канала атакующего.

Такая атака в нормальных условиях, следуя букве стандарта, не реализуема, поскольку сброс таблицы коммутации не осуществляется мгновенно по приходу C-BPDU с отличной от текущей конфигурацией. Однако многие устройства поддерживают расширения стандарта, в которые заложена возможность мгновенного перехода из blocking в forwarding. Как следствие, время жизни таблицы коммутации в таких устройствах в случае возникновения изменений топологии можно сбросить в 0. В этом случае, необходимо подобрать такие условия работы устройства, чтобы оно, не прерывая работы постоянно сбрасывало свою таблицу коммутации.

Как и MitM-атака, атака provocation sniffing может быть использована, например, для получения доступа к управлению STP-совместимым устройством. Помимо этого, атаки на втором уровне OSI могут служить подготовкой к атакам на более высоких уровнях. Например, имея возможность "слушать" трафик, можно эффективно угадывать sequence numbers для "вклинивания" в tcp-соединения.

9.4. Получение дополнительной информации о сети при помощи STP

Помимо описанных выше фундаментальных атак на топологию сети существует еще одна не страшная, но тоже неприятная особенность: благодаря STP возможно получение дополнительных данных о структуре сети. При этом такое получение дополнительных знаний может быть как пассивным (прослушивание среды передачи данных), так и активным: выиграв на короткое время статус STP-корня, злоумышленник автоматически становимся целью отправки STP TCN-BPDU, которые сами по себе несут некоторый объем познавательной информации. Например, проанализировав MAC-адрес отправителя, можно понять устройство какой фирмы отправило этот пакет, а это тоже атака, хоть и пассивная.

9.5. Как администраторы сетей могут противостоять атакам

Если обеспечение безопасности сети стоит не на последнем месте, то лучше отключать поддержку Spaning Tree всегда, когда это возможно, и пытаться заменить STP другими технологиями там, где требуются решения с дублированием каналов, но все же можно обойтись без особенностей STP. Например, можно использовать технологию агрегирования каналов.

Если же выключить STP по тем или иным неутешительным причинам для всей сети неприемлемо, придется использовать расширения Spaning Tree, доступные, в частности, в решениях, предоставляемых Cisco, а также предпринять следующий шаг:

- ✓ выставить минимально возможный bridge id и прочие параметры, что сделает приоритет коммутатора максимальным, тогда он всегда будет корнем и реконфигурация может разве что затронуть часть сети, ближайшие к атакующему коммутатору, за исключением установленного раз и навсегда STP-корня. Можно это сделать, поскольку STP корень выполняет всего лишь навсего роль точки отсчета, от которой строится топология без колец. Но вот незадача - последний параметр выбора (если все одинаковое) - MAC-адрес порта, что собственно считается гарантированно разным.

9.6. Выводы

С помощью нецелевого применения STP можно:

1. Злоумышленник может осуществить MitM-атаку (man-in-the-middle - "человек посередине"), если его рабочая станция подключена двумя или более интерфейсами к различным STP-совместимым устройствам, например, коммутаторам.
2. Злоумышленник имеет возможность осуществить атаку на отказ в обслуживании (DoS) даже если его рабочая станция имеет всего один сетевой интерфейс, при условии, что функционирующие в сети STP-совместимые устройства не поддерживают BPDU-guard или STP portfast, а также в случае, если указанные возможности на этих устройствах просто не задействованы.
3. В случае, когда STP-совместимые устройства в сети поддерживают STP portfast, злоумышленник может спровоцировать ситуацию, когда STP-совместимый коммутатор переходит в режим концентратора (hub), и прослушивать при помощи сетевого анализатора весь трафик между узлами сети, подключенными к этому коммутатору. Эта атака частично реализуема также и для случая с несколькими коммутаторами.

С помощью нецелевого применения STP нельзя:

1. Невозможно организовать STP-MitM атаку, имея только один сетевой интерфейс, но это становится возможным в случае, если этот интерфейс подключить к концентратору, при этом поведение STP-совместимого устройства неочевидно.
2. Злоумышленник не может производить изменения в топологии сети, будучи подключенным только к одному порту коммутатора, не спровоцировав при этом отказ в обслуживании (DoS) для какой-либо части сети. Злоумышленник не может реализовать изменения в топологии, касающиеся некоторого сегмента сети, подключившись к коммутатору, который имеет только одно соединение с этим сегментом, не спровоцировав при этом отказ доступа к этому сегменту из той части сети, которую затрагивают внесенные изменения.
3. В Ethernet технологии невозможно организовать MitM атаку с рабочей станции злоумышленника для двух конечных узлов ЛВС до тех пор, пока нет физического кольца между этими точками. Хорошим примером будет случай двух обширных сетей, соединенных между собой по единственному не дублируемому кабелю. В таком случае организовать MitM между станциями в разных сетях невозможно без прокладки кабеля для организации кольца. Такое кольцо может быть образовано самим злоумышленником, если он имеет физический доступ к коммутаторам этой сети и порты коммутатора по умолчанию включены.

V. Механизмы обеспечения безопасности беспроводных локальных сетей

1. Классификация механизмов безопасности в сетях Wi-Fi

В современных беспроводных Wi-Fi сетях имеется следующий набор механизмов, обеспечивающих безопасность работы пользователя и конфиденциальность передаваемой информации. Данный набор представлен на рисунке 5.1.

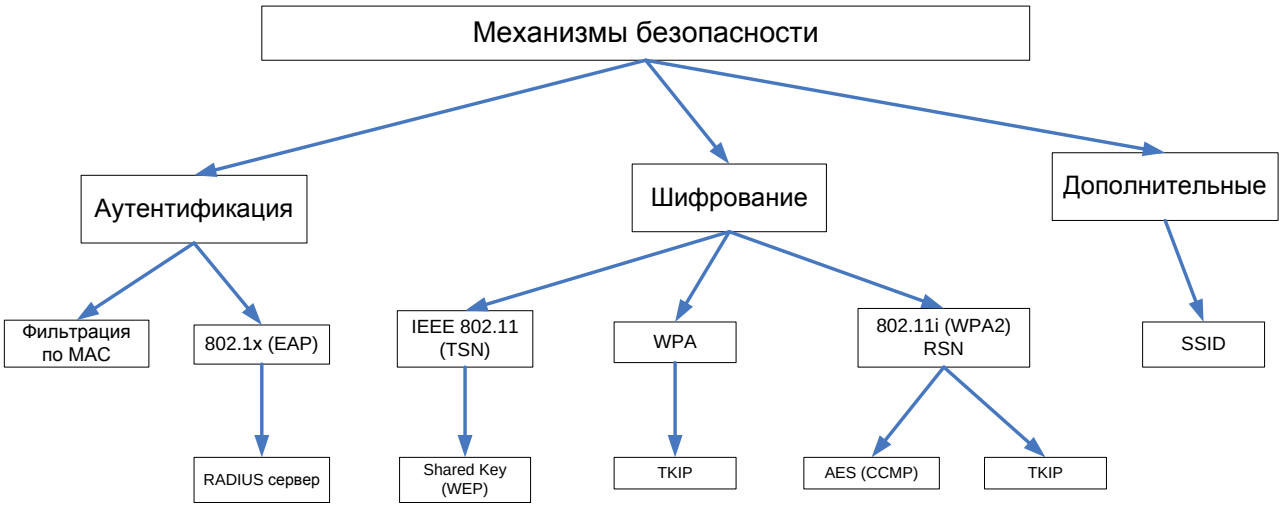


Рисунок 5.1. Классификация механизмов безопасности в Wi-Fi сетях.

2. Механизмы шифрования

2.1. Механизм шифрования WEP (IEEE 802.11)

Шифрование WEP (Wired Equivalent Privacy - секретность на уровне проводной связи) основано на алгоритме RC4 (Rivest's Cipher v.4 – код Ривеста), который представляет собой симметричное потоковое шифрование. Как было отмечено ранее, для нормального обмена пользовательскими данными ключи шифрования у абонента и точки радиодоступа должны быть идентичными.

Ядро алгоритма состоит из функции генерации ключевого потока. Эта функция генерирует последовательность битов, которая затем объединяется с открытым текстом посредством суммирования по модулю два. Дешифрация состоит из регенерации этого ключевого потока и суммирования его с шифрограммой по модулю два для восстановления исходного текста. Другая главная часть алгоритма - функция инициализации, которая использует ключ переменной длины для создания начального состояния генератора ключевого потока.

RC4 – фактически класс алгоритмов, определяемых размером его блока. Этот параметр n является размером слова для алгоритма. Обычно, n = 8, но в целях анализа можно уменьшить его. Однако для повышения уровня безопасности необходимо задать большее значение этой величины. Внутреннее состояние RC4 состоит из массива размером 2n слов и двух счетчиков, каждый размером в одно слово. Массив известен как S-бокс, и далее он будет обозначаться как S. Он всегда содержит перестановку 2n возможных значений слова. Два счетчика обозначены через i и j. Алгоритм инициализации RC4 приведен ниже. Этот алгоритм использует ключ, сохраненный в Key и имеющий длину l байт. Инициализация начинается с заполнения массива S, далее этот массив перемешивается путем перестановок, определяемых ключом. Так как над S выполняется только одно дейст-

вие, должно выполняться утверждение, что S всегда содержит все значения кодового слова.

Начальное заполнение массива:

```
for i = 0 to 2n – 1
{
      S[i] = i
      j = 0
}
```

Скрэмблирование:

```
for i = 0 to 2n – 1
{
      j = j + S[i] + Key[i mod l]
      Перестановка (S[i], S[j])
}
```

Генератор ключевого потока RC4 переставляет значения, хранящиеся в S, и каждый раз выбирает новое значение из S в качестве результата. В одном цикле RC4 определяется одно n-битное слово K из ключевого потока, которое в дальнейшем суммируется с исходным текстом для получения зашифрованного текста.

Инициализация:

```
i = 0
j = 0
```

Цикл генерации:

```
i = i + 1
j = j + S[i]
```

Перестановка ***(S[i], S[j])***

Результат: ***K = S[S[i] + S[j]]***.

Особенности WEP-протокола:

1. Достаточно устойчив к атакам, связанным с простым перебором ключей шифрования, что обеспечивается необходимой длиной ключа и частотой смены ключей и инициализирующего вектора.
2. Самосинхронизация для каждого сообщения. Это свойство является ключевым для протоколов уровня доступа к среде передачи, где велико число искаженных и потерянных пакетов.
3. Эффективность: WEP легко реализовать.
4. Открытость.
5. Использование WEP-шифрования не является обязательным в сетях стандарта IEEE 802.11.

Для непрерывного шифрования потока данных используется потоковое и блочное шифрование.

Потоковое шифрование

При потоковом шифровании выполняется побитовое сложение по модулю 2 (функция "исключающее ИЛИ", XOR) ключевой последовательности, генерируемой алгоритмом шифрования на основе заранее заданного ключа, и исходного сообщения. Ключевая по-

следовательность имеет длину, соответствующую длине исходного сообщения, подлежащего шифрованию (рисунок 5.2).

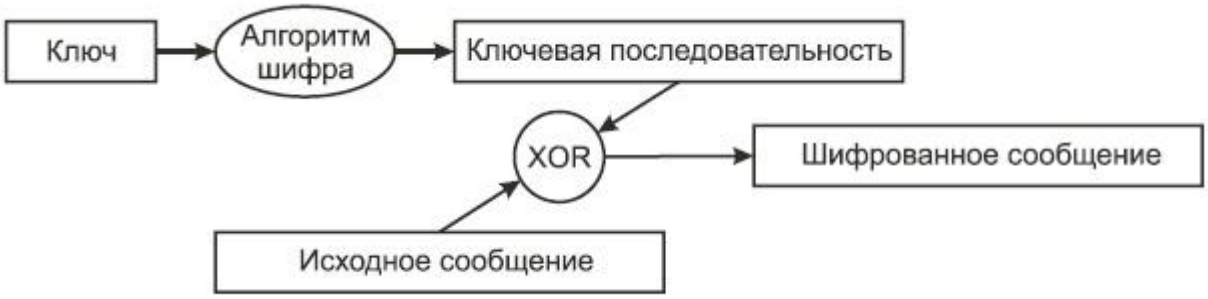


Рисунок 5.2. Потоковое шифрование.

Блочное шифрование

Блочное шифрование работает с блоками заранее определенной длины, не меняющейся в процессе шифрования. Исходное сообщение фрагментируется на блоки, и функция XOR вычисляется над ключевой последовательностью и каждым блоком. Размер блока фиксирован, а последний фрагмент исходного сообщения дополняется пустыми символами до длины нормального блока (рисунок 5.3). Например, при блочном шифровании с 16-байтовыми блоками исходное сообщение длиной в 38 байтов фрагментируется на два блока длиной по 16 байтов и 1 блок длиной 6 байтов, который затем дополняется 10 байтами пустых символов до длины нормального блока.

Потоковое шифрование и блочное шифрование используют метод электронной кодовой книги (ECB). Метод ECB характеризуется тем, что одно и то же исходное сообщение на входе всегда порождает одно и то же зашифрованное сообщение на выходе. Это потенциальная брешь в системе безопасности, ибо сторонний наблюдатель, обнаружив повторяющиеся последовательности в зашифрованном сообщении, в состоянии сделать обоснованные предположения относительно идентичности содержания исходного сообщения.

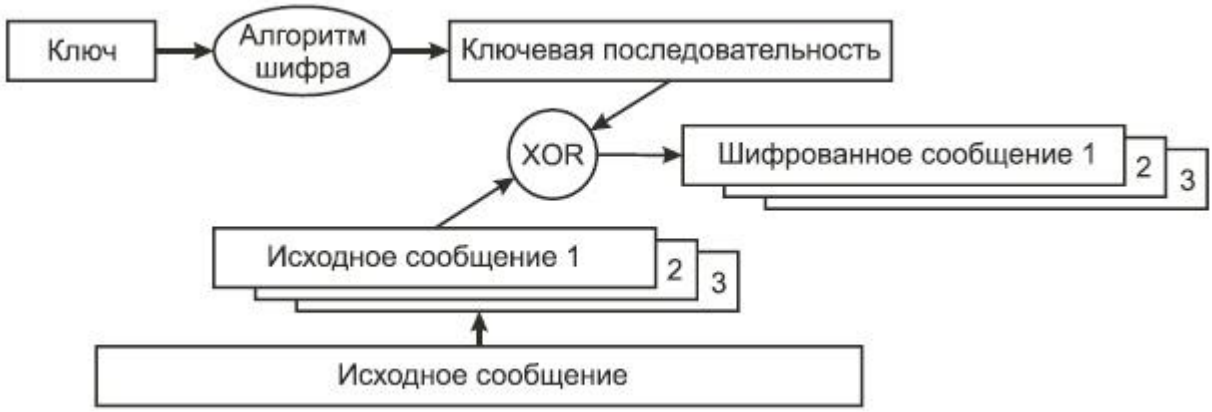


Рисунок 5.3. Блочное шифрование.

Для устранения указанной проблемы используют:

- ✓ векторы инициализации (Initialization Vectors - IVs);
- ✓ обратную связь (feedback modes).

До начала процесса шифрования 40- или 104-битный секретный ключ распределяется между всеми станциями, входящими в беспроводную сеть. К секретному ключу добавляется вектор инициализации (IV).

Вектор инициализации (Initialization Vector, IV)

Вектор инициализации используется для модификации ключевой последовательности. При использовании вектора инициализации ключевая последовательность генерируется алгоритмом шифрования, на вход которого подается секретный ключ, совмещенный с IV. При изменении вектора инициализации ключевая последовательность также меняется. На рисунке 5.4 исходное сообщение шифруется с использованием новой ключевой последовательности, сгенерированной алгоритмом шифрования после подачи на его вход комбинации из секретного ключа и вектора инициализации, что порождает на выходе шифрованное сообщение.

Стандарт IEEE 802.11 рекомендует использовать новое значение вектора инициализации для каждого нового фрейма, передаваемого в радиоканал.

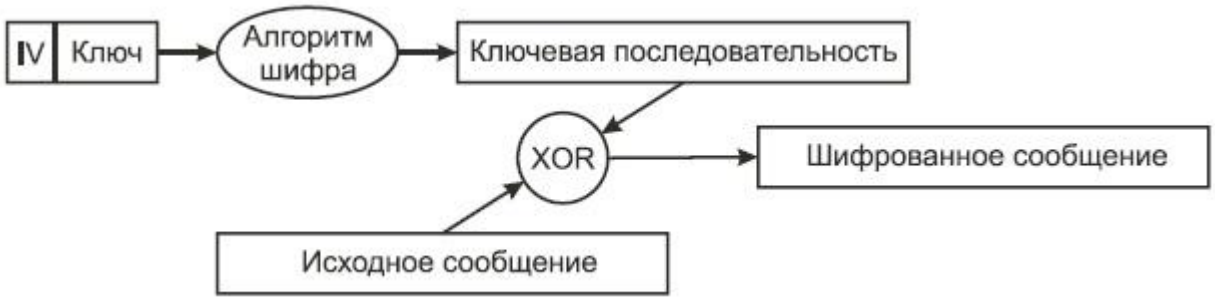


Рисунок 5.4. Алгоритм шифрования WEP.

Таким образом, один и тот же нешифрованный фрейм, передаваемый многократно, каждый раз будет порождать уникальный шифрованный фрейм. Вектор инициализации имеет длину 24 бита и совмещается с 40- или 104-битовым базовым ключом шифрования WEP таким образом, что на вход алгоритма шифрования подается 64- или 128-битовый ключ. Вектор инициализации присутствует в нешифрованном виде в заголовке фрейма в радиоканале, с тем чтобы принимающая сторона могла успешно декодировать этот фрейм. Несмотря на то, что обычно говорят об использовании шифрования WEP с ключами длиной 64 или 128 битов, эффективная длина ключа составляет лишь 40 или 104 бита по причине передачи вектора инициализации в нешифрованном виде. При настройках шифрования в оборудовании при 40-битном эффективном ключе вводятся 5 байтовых ASCII-символов (5*8 = 40) или 10 шестнадцатеричных чисел (10*4 = 40), и при 104-битном эффективном ключе вводятся 13 байтовых ASCII-символов (13*8 = 104) или 26 шестнадцатеричных чисел (26*4 = 104). Некоторое оборудование может работать со 128-битным ключом.

Обратная связь

Обратная связь модифицирует процесс шифрования и предотвращает порождение одним и тем же исходным сообщением одного и того же шифрованного сообщения. Обратная связь обычно используется при блочном шифровании. Чаще всего встречается тип обратной связи, известный как цепочка шифрованных блоков (CBC).

В основе использования цепочки шифрованных блоков лежит идея вычисления двоичной функции XOR между блоком исходного сообщения и предшествовавшим ему блоком шифрованного сообщения. Поскольку самый первый блок не имеет предшественника, для модификации ключевой последовательности используют вектор инициализации.

2.2. Спецификация WPA

До мая 2001г. стандартизация средств информационной безопасности для беспроводных сетей 802.11 относилась к ведению рабочей группы IEEE 802.11e, но затем эта проблематика была выделена в самостоятельное подразделение. Разработанный стандарт 802.11i призван расширить возможности протокола 802.11, предусмотрев средства шифрования передаваемых данных, а также централизованной аутентификации пользователей и рабочих станций.

Основные производители Wi-Fi оборудования в лице организации WECA (Wireless Ethernet Compatibility Alliance), иначе именуемой Wi-Fi Alliance, устав ждать ратификации стандарта IEEE 802.11i, совместно с IEEE в ноябре 2002 г. анонсировали спецификацию W-Fi Protected Access (WPA), соответствие которой обеспечивает совместимость оборудования различных производителей.

Новый стандарт безопасности WPA обеспечивает уровень безопасности куда больший, чем может предложить WEP Он перебрасывает мостик между стандартами WEP и 802.11i и имеет немаловажное преимущество, которое заключается в том, что микропрограммное обеспечение более старого оборудования может быть заменено без внесения аппаратных изменений.

IEEE предложила временный протокол целостности ключа (Temporal Key Integrity Protocol, TKIP).
Основные усовершенствования, внесенные протоколом TKIP:

- ✓ покадровое изменение ключей шифрования. WEP-ключ быстро изменяется, и для каждого кадра он другой;
- ✓ контроль целостности сообщения. Обеспечивается эффективный контроль целостности фреймов данных с целью предотвращения скрытых манипуляций с фреймами и воспроизведения фреймов;
- ✓ усовершенствованный механизм управления ключами.

Пофреймовое изменение ключей шифрования

Атаки, применяемые в WEP, использующие уязвимость слабых IV (Initialization Vectors), таких, которые применяются в приложении AirSnort, основаны на накоплении нескольких фреймов данных, содержащих информацию, зашифрованную с использованием слабых IV. Простейшим способом сдерживания таких атак является изменение WEP-ключа, используемого при обмене фреймами между клиентом и точкой доступа, до того как атакующий успеет накопить фреймы в количестве, достаточном для вывода битов ключа.

IEEE адаптировала схему, известную как пофреймовое изменение ключа (per-frame keying). Основной принцип, на котором основано пофреймовое изменение ключа, состоит в том, что IV, MAC-адрес передатчика и WEP-ключ обрабатываются вместе с помощью двухступенчатой функции перемешивания. Результат применения этой функции соответствует стандартному 104-разрядному WEP-ключу и 24-разрядному IV.

IEEE предложила также увеличить 24-разрядный вектор инициализации до 48-разрядного IV. На рисунке 5.5 представлен образец 48-разрядного IV и показано, как он разбивается на части для использования при пофреймовом изменении ключа.

Рисунок 5.5. Разбиение 48-разрядного IV.

Процесс пофреймового изменения ключа можно разбить на следующие этапы (рисунок 5.6):

- ✓ базовый WEP-ключ перемешивается со старшими 32 разрядами 48-разрядного IV (32-разрядные числа могут принимать значения 0-4 294 967 295) и MAC-адресом передатчика. Результат этого действия называется ключ 1-й фазы. Этот процесс позволяет занести ключ 1-й фазы в кэш и также напрямую поместить в ключ;
- ✓ ключ 1-й фазы снова перемешивается с IV и MAC-адресом передатчика для выработки значения пофреймового ключа;
- ✓ вектор инициализации (IV), используемый для передачи фрейма, имеет размер только 16 бит (16-разрядные числа могут принимать значения 0-65 535). Оставшиеся 8 бит (в стандартном 24-битовом IV) представляют собой фиксированное значение, используемое как заполнитель;
- ✓ пофреймовый ключ применяется для WEP-шифрования фрейма данных.

Когда 16-битовое пространство IV оказывается исчерпанным, ключ 1-й фазы отбрасывается и 32 старших разряда увеличиваются на 1. Значение пофреймового ключа вычисляется заново, как на этапе 2.

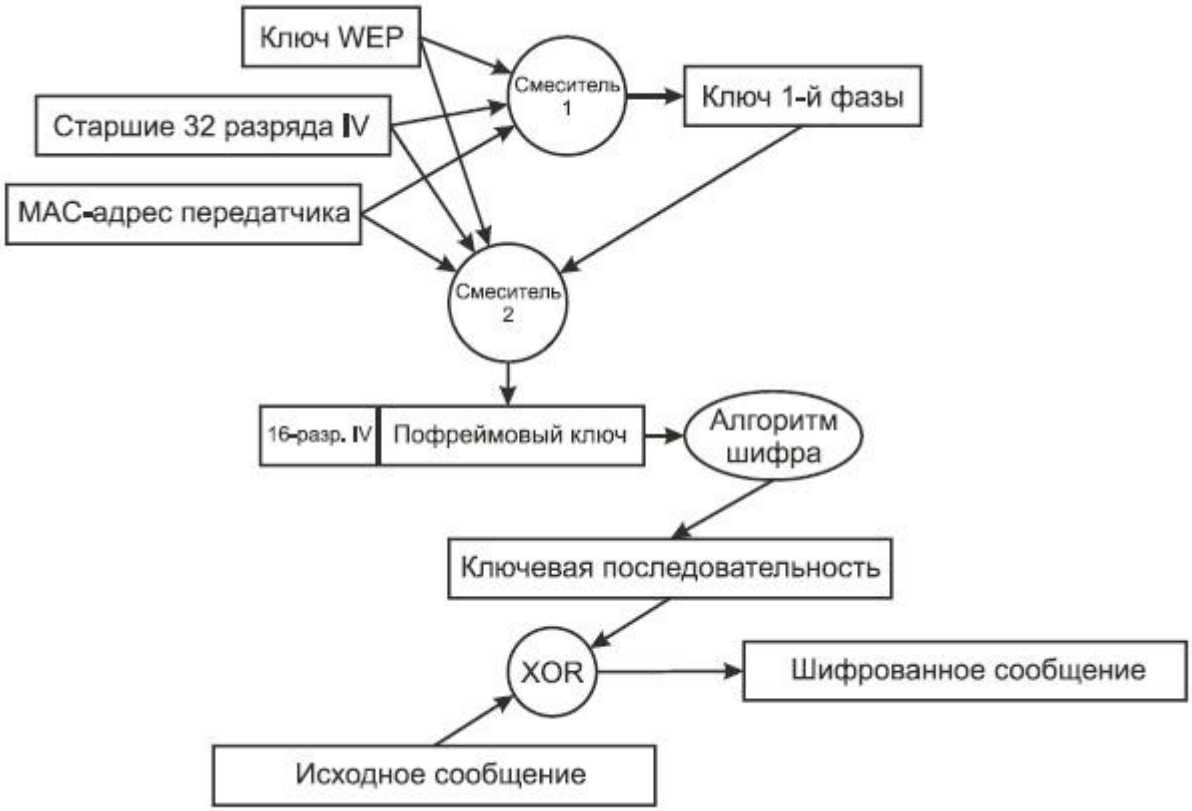


Рисунок 5.6. Процесс создания шифрованного сообщения в WPA.

Процесс пофреймового изменения ключа можно разбить на следующие этапы.

- ✓ устройство инициализирует IV, присваивая ему значение 0. В двоичном представлении это будет значение 00000000000000000000000000000000 00000000000000000;

- ✓ первые 32 разряда IV (в рассматриваемом случае - первые 32 нуля) перемешиваются с WEP-ключом (например, имеющим 128-разрядное значение) и MAC-адресом передатчика (имеющим 48-разрядное значение) для получения значения ключа 1-й фазы (80-разрядное значение);
- ✓ ключ 1-й фазы вновь перемешивается с первыми (старшими) 32 разрядами IV и MAC-адресом передатчика, чтобы получить 128-разрядный пофреймовый ключ, первые 16 разрядов которого представляют собой значение IV (16 нулей);
- ✓ вектор инициализации пофреймового ключа увеличивается на 1. После того как пофреймовые возможности IV будут исчерпаны, IV 1-й фазы (32 бита) увеличивается на 1 (он теперь будет состоять из 31 нуля и одной единицы, 00000000000000000000000000000001) и т. д.

Этот алгоритм усиливает WEP до такой степени, что почти все известные сейчас возможности атак устраняются без замены существующего оборудования. Следует отметить, что этот алгоритм (и TKIP в целом) разработан с целью устранить уязвимые места в системе аутентификации WEP и стандарта 802.11. Он жертвует слабыми алгоритмами, вместо того чтобы заменять оборудование.

Контроль целостности сообщения

Для усиления малоэффективного механизма, основанного на использовании контрольного признака целостности (ICV) стандарта 802.11, будет применяться контроль целостности сообщения (MIC). Благодаря MIC могут быть ликвидированы слабые места защиты, способствующие проведению атак с использованием поддельных фреймов и манипуляции битами. IEEE предложила специальный алгоритм, получивший название Michael (Майкл), чтобы усилить роль ICV в шифровании фреймов данных стандарта 802.11.
MIC имеет уникальный ключ, который отличается от ключа, используемого для шифрования фреймов данных. Этот уникальный ключ перемешивается с назначенным MAC-адресом и исходным MAC-адресом фрейма, а также со всей незашифрованной частью фрейма. На рисунке 5.7 показана работа алгоритма Michael MIC.

Рисунок 5.7. Работа алгоритма Michael MIC.

Механизм шифрования TKIP в целом осуществляется следующим образом:

1. С помощью алгоритма пофреймового назначения ключей генерируется пофреймовый ключ (рисунок 5.8).
2. Алгоритм MIC генерирует MIC для фрейма в целом.
3. Фрейм фрагментируется в соответствии с установками MAC относительно фрагментации.
4. Фрагменты фрейма шифруются с помощью пофреймового ключа.
5. Осуществляется передача зашифрованных фрагментов.

Рисунок 5.8. Механизм шифрования TKIP.

Аналогично процессу шифрования по алгоритму TKIP, процесс дешифрования по этому алгоритму выполняется следующим образом (рисунок 5.9):

1. Предварительно вычисляется ключ 1-й фазы.
2. На основании IV, полученного из входящего фрагмента фрейма WEP, вычисляется пофреймовый ключ 2-й фазы.
3. Если полученный IV не тот, какой нужно, фрейм отбрасывается.
4. Фрагмент фрейма расшифровывается, и осуществляется проверка признака целостности (ICV).
5. Если контроль признака целостности дает отрицательный результат, такой фрейм отбрасывается.
6. Расшифрованные фрагменты фрейма собираются, чтобы получить исходный фрейм данных.
7. Приемник вычисляет значение MIC и сравнивает его со значением, находящимся в поле MIC фрейма.
8. Если эти значения совпадают, фрейм обрабатывается приемником.
9. Если эти значения не совпадают, значит, фрейм имеет ошибку MIC, и приемник принимает меры противодействия MIC.

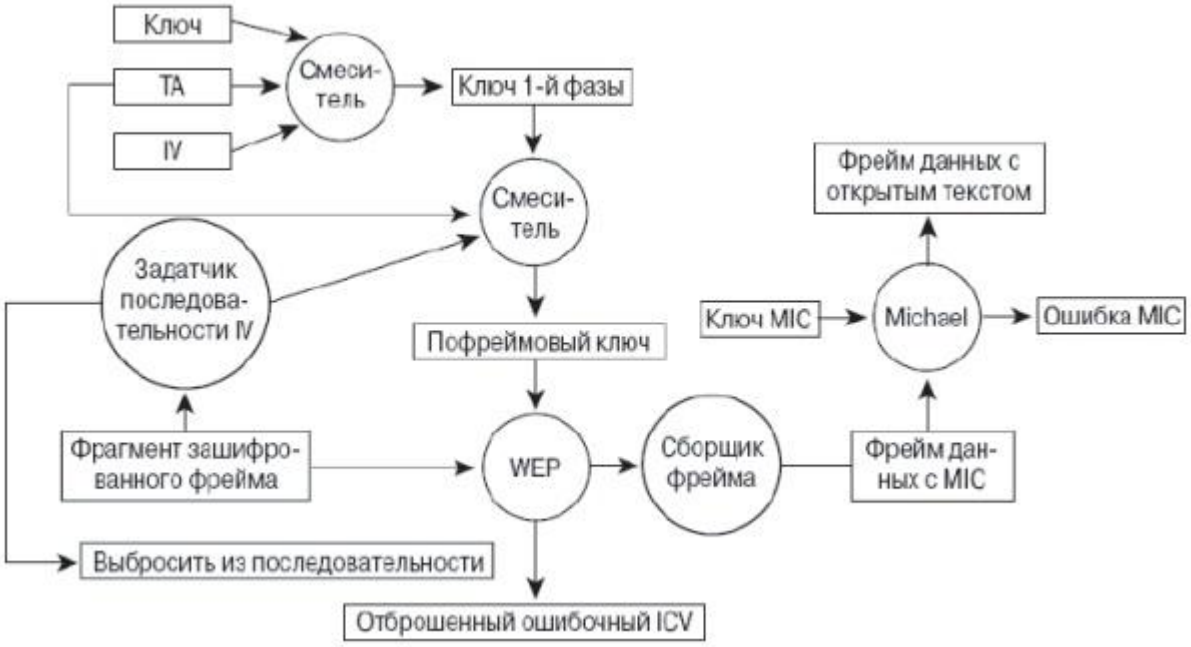


Рисунок 5.9. Механизм дешифровки TKIP.

Меры противодействия MIC состоят в выполнении приемником следующих задач:

1. Приемник удаляет существующий ключ на ассоциирование.
2. Приемник регистрирует проблему как относящуюся к безопасности сети.
3. Ассоциированный клиент, от которого был получен ложный фрейм, не может быть ассоциирован и аутентифицирован в течение 60 секунд, чтобы замедлить атаку.
4. Клиент запрашивает новый ключ.

WPA может работать в двух режимах: Enterprise (корпоративный) и Pre-Shared Key (персональный).

В первом случае хранение базы данных и проверка аутентичности по стандарту 802.1x в больших сетях обычно осуществляются специальным сервером, чаще всего RADIUS (Remote Authentication Dial-In User Service). Enterprise-режим мы рассмотрим далее.

Во втором случае подразумевается применение WPA всеми категориями пользователей беспроводных сетей, т.е. имеет место упрощенный режим, не требующий сложных механизмов. Этот режим называется WPA-PSK и предполагает введение одного пароля на каждый узел беспроводной сети (точку доступа, беспроводной маршрутизатор, клиентский адаптер, мост). До тех пор, пока пароли совпадают, клиенту будет разрешен доступ в сеть. Можно заметить, что подход с использованием пароля делает WPA-PSK уязвимым для атаки методом подбора, однако этот режим избавляет от путаницы с ключами WEP, заменяя их целостной и четкой системой на основе цифро-буквенного пароля.

Таким образом, WPA/TKIP – это решение, предоставляющее больший по сравнению с WEP уровень безопасности, направленное на устранение слабостей предшественника и обеспечивающее совместимость с более старым оборудованием сетей 802.11 без внесения аппаратных изменений в устройства.

2.3. Стандарт сети 802.11i с повышенной безопасностью (WPA2)

В июне 2004 г. IEEE ратифицировал давно ожидаемый стандарт обеспечения безопасности в беспроводных локальных сетях – 802.11i.

Действительно, WPA достоин восхищения как шедевр ретроинжиниринга. Созданный с учетом слабых мест WEP, он представляет собой очень надежную систему безопасности и, как правило, обратно совместим с существующим Wi-Fi-оборудованием. WPA - практическое решение, обеспечивающее достаточный уровень безопасности для беспроводных сетей.

Однако WPA – компромиссное решение. Оно все еще основано на алгоритме шифрования RC4 и протоколе TKIP. Вероятность выявления каких-либо слабых мест хотя и мала, но все же существует.

Абсолютно новая система безопасности, лишенная недостатков WEP, представляет собой лучшее долгосрочное и к тому же расширяемое решение для безопасности беспроводных сетей. С этой целью комитет по стандартам принял решение разработать систему безопасности с нуля. Это новый стандарт 802.11i, также известный как WPA2 и выпущенный тем же Wi-Fi Alliance.

Стандарт 802.11i использует концепцию повышенной безопасности (Robust Security Network, RSN), предусматривающую, что беспроводные устройства должны обеспечивать дополнительные возможности. Это потребует изменений в аппаратной части и программном обеспечении, т.е. сеть, полностью соответствующая RSN, станет несовместимой с существующим оборудованием WEP. В переходный период будет поддерживаться как оборудование RSN, так и WEP (на самом деле WPA/TKIP было решением, направленным на сохранение инвестиций в оборудование), но в дальнейшем устройства WEP начнут отмирать.

802.11i приложим к различным сетевым реализациям и может задействовать TKIP, но по умолчанию RSN использует AES (Advanced Encryption Standard) и CCMP (Counter Mode CBC MAC Protocol) и, таким образом, является более мощным расширяемым решением.
В концепции RSN применяется AES в качестве системы шифрования, подобно тому как алгоритм RC4 задействован в WPA. Однако механизм шифрования куда более сложен и не страдает от проблем, свойственных WEP AES - блочный шифр, оперирующий блоками данных по 128 бит. CCMP, в свою очередь, - протокол безопасности, используемый AES. Он является эквивалентом TKIP в WPA. CCMP вычисляет MIC, прибегая к хорошо известному и проверенному методу Cipher Block Chaining Message Authentication Code (CBC-MAC). Изменение даже одного бита в сообщении приводит к совершенно другому результату.

Одной из слабых сторон WEP было управление секретными ключами. Многие администраторы больших сетей находили его неудобным. Ключи WEP не менялись длительное время (или никогда), что облегчало задачу злоумышленникам.

RSN определяет иерархию ключей с ограниченным сроком действия, сходную с TKIP В AES/CCMP, чтобы вместить все ключи, требуется 512 бит - меньше, чем в TKIP В обоих случаях мастер-ключи используются не прямо, а для вывода других ключей. К счастью, администратор должен обеспечить единственный мастер-ключ. Сообщения составляются из 128-битного блока данных, зашифрованного секретным ключом такой же длины (128 бит). Хотя процесс шифрования сложен, администратор опять-таки не должен вникать в нюансы вычислений. Конечным результатом является шифр, который гораздо сложнее, чем даже WPA.

IEEE 802.11i (WPA2) – это наиболее устойчивое, расширяемое и безопасное решение, предназначенное в первую очередь для крупных предприятий, где управление ключами и администрирование доставляет множество хлопот.

Стандарт IEEE 802.11i разработан на базе проверенных технологий. Механизмы безопасности были спроектированы с нуля в тесном сотрудничестве с лучшими специалистами по криптографии и имеют все шансы стать тем решением, которое необходимо беспроводным сетям. Хотя ни одна система безопасности от взлома не застрахована, IEEE 802.11i – это решение, на которое можно полагаться, в нем нет недостатков предыдущих систем. И, конечно, WPA пригоден для адаптации уже существующего оборудования, и только когда его ресурсы будут окончательно исчерпаны, вы сможете заменить его новым, полностью соответствующим концепции RSN.

Производительность канала связи, как свидетельствуют результаты тестирования оборудования различных производителей, падает на 5-20% при включении как WEP, так и WPA. Однако испытания того оборудования, в котором включено шифрование AES вместо TKIP, не показали сколько-нибудь заметного падения скорости. Это позволяет надеяться, что WPA2-совместимое оборудование предоставит нам долгожданный надежно защищенный канал без потерь в производительности.

Шифрование по алгоритму AES

Известно, что шифрование и аутентификация, проводимые в соответствии со стандартом 802.11, имеют слабые стороны, IEEE и WPA усилили алгоритм WEP протоколом TKIP и предлагает сильный механизм шифрования по стандарту 802.11i , обеспечивающий защиту беспроводных LAN стандарта. В тоже время IEEE адаптировал механизм AES для применения его по отношению к разделу, касающемуся защищаемых данных предлагаемого стандарта 802.11i. Компоненты WPA не обеспечивают поддержку шифрования по алгоритму AES.

Алгоритм AES представляет собой следующее поколение средств шифрования. Национальным институтом стандартов и технологий (NITS) США. Названный институт предложил сообществу криптологов разработать новые алгоритмы шифрования. Эти алгоритмы должны быть полностью открыты и использоваться бесплатно. Финалистом и принятым методом стал так называемый алгоритм Рийндела. Как и многие другие шифры, AES требует режима обратной связи во избежание риска, связанного с режимом ECB(режим шифрования с помощью электронных кодов). IEEE разработал режим AES, предназначенный специально для применения в беспроводных локальных сетях. Этот режим называется режим счета сцеплений блоков шифра (Chiper Block Chaining Counter Mode, CBC-CTR) с контролем аутентичности сообщений о сцеплениях блоков шифра (Chiper Block Chaining Message Authenticity Check, CCB-MAC), все вместе это обозначается аббревиатурой AES-CCM. Режим CCM представляет собой комбинацию режима шифрования CBC-CTR и алгоритма контроля аутентичности сообщений CBK-MAC. Эти функции скомбинированы для обеспечения шифрования и проверки целостности сообщений в одном решении.

Алгоритм шифрования CBC-CTR работает с использованием счетчика для пополнения ключевого потока. Значение этого счетчика увеличивается на единицу после шифрования каждого блока. Такой процесс обеспечивает получение уникального ключевого потока для каждого блока. Фрейм с открытым текстом делится на 16-байтовые блоки. После шифрования каждого блока значение счетчика увеличивается на единицу, и так до тех пор, пока

не будут зашифрованы все блоки. Для каждого нового фрейма счетчик переустанавливается.

Алгоритм шифрования CBC-MAC выполняется с использованием результата шифрования CBC по отношению ко всему фрейму, к адресу назначения, адресу источника и данным. Результирующий 128 разрядный выход усекается до 64 бит для использования в передаваемом фрейме.

CBC-MAC работает с известными криптографическими функциями, но имеет издержки, связанные с выполнением двух операций для шифрования и целостности сообщений. Этот процесс требует серьезных вычислительных затрат и значительно увеличит «накладные расходы» шифрования.

3. Аутентификация в беспроводных Wi-Fi сетях

3.1. Принцип аутентификации абонента в IEEE 802.11

Аутентификация в стандарте IEEE 802.11 ориентирована на аутентификацию абонентского устройства радиодоступа, а не конкретного абонента как пользователя сетевых ресурсов. Процесс аутентификации абонента беспроводной локальной сети IEEE 802.11 состоит из следующих этапов (рисунок 5.10):

1. Абонент (Client) посылает фрейм Probe Request во все радиоканалы.
2. Каждая точка радиодоступа (Access Point - AP), в зоне радиовидимости которой находится абонент, посылает в ответ фрейм Probe Response.
3. Абонент выбирает предпочтительную для него точку радиодоступа и посылает в обслуживаемый ею радиоканал запрос на аутентификацию (Authentication Request).
4. Точка радиодоступа посылает подтверждение аутентификации (Authentication Reply).
5. В случае успешной аутентификации абонент посылает точке радиодоступа фрейм ассоциации (Association Request).
6. Точка радиодоступа посылает в ответ фрейм подтверждения ассоциации (Association Response).
7. Абонент может теперь осуществлять обмен пользовательским трафиком с точкой радиодоступа и проводной сетью.

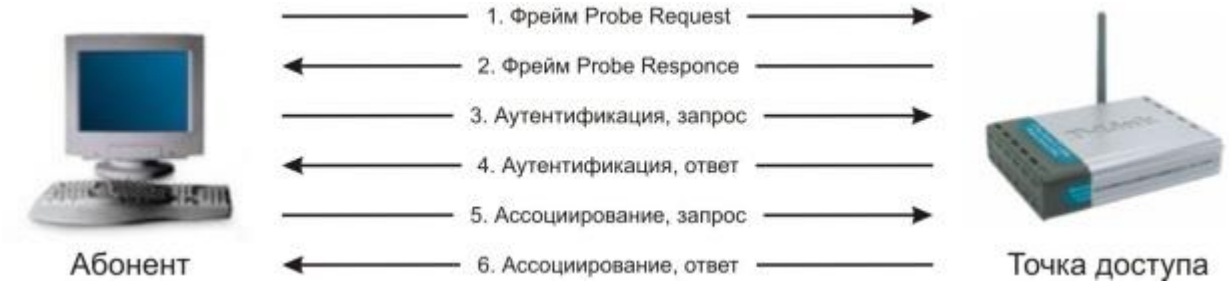


Рисунок 5.10. Аутентификация по стандарту IEEE 802.11.

При активизации беспроводный абонент начинает поиск точек радиодоступа в своей зоне радиовидимости с помощью управляющих фреймов Probe Request. Фреймы Probe Request посылаются в каждый из радиоканалов, поддерживаемых абонентским радиоинтерфейсом, чтобы найти все точки радиодоступа с необходимыми клиенту идентификатором SSID и поддерживаемыми скоростями радиообмена. Каждая точка радиодоступа из находящихся в зоне радиовидимости абонента, удовлетворяющая запрашиваемым во фрейме Probe Request параметрам, отвечает фреймом Probe Response, содержащим синхронизирующую информацию и данные о текущей загрузке точки радиодоступа. Абонент определяет, с какой точкой радиодоступа он будет работать, путем сопоставления

поддерживаемых ими скоростей радиообмена и загрузки. После того как предпочтительная точка радиодоступа определена, абонент переходит в фазу аутентификации.

1.2. Открытая аутентификация

Открытая аутентификация по сути не является алгоритмом аутентификации в привычном понимании. Точка радиодоступа удовлетворит любой запрос открытой аутентификации. На первый взгляд использование этого алгоритма может показаться бессмысленным, однако следует учитывать, что разработанные в 1997 году методы аутентификации IEEE 802.11 ориентированы на быстрое логическое подключение к беспроводной локальной сети. Вдобавок к этому многие IEEE 802.11-совместимые устройства представляют собой портативные блоки сбора информации (сканеры штрих-кодов и т. п.), не имеющие достаточной процессорной мощности, необходимой для реализации сложных алгоритмов аутентификации. В процессе открытой аутентификации происходит обмен сообщениями двух типов:

- ✓ запрос аутентификации (Authentication Request);
- ✓ подтверждение аутентификации (Authentication Response).

Таким образом, при открытой аутентификации возможен доступ любого абонента к беспроводной локальной сети. Если в беспроводной сети шифрование не используется, любой абонент, знающий идентификатор SSID точки радиодоступа, получит доступ к сети. При использовании точками радиодоступа шифрования WEP сами ключи шифрования становятся средством контроля доступа. Если абонент не располагает корректным WEP-ключом, то даже в случае успешной аутентификации он не сможет ни передавать данные через точку радиодоступа, ни расшифровывать данные, переданные точкой радиодоступа (рисунок 5.11).

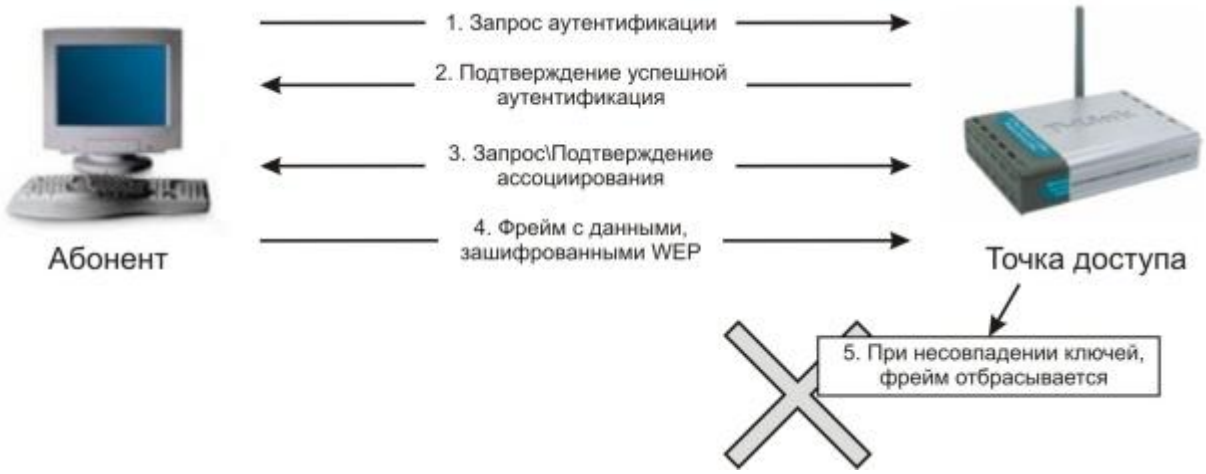


Рисунок 5.11. Открытая аутентификация.

1.3. Аутентификация с общим ключом

Аутентификация с общим ключом является вторым методом аутентификации стандарта IEEE 802.11. Аутентификация с общим ключом требует настройки у абонента статического ключа шифрования WEP. Процесс аутентификации иллюстрирует рисунок 5.12:

1. Абонент посылает точке радиодоступа запрос аутентификации, указывая при этом необходимость использования режима аутентификации с общим ключом.
2. Точка радиодоступа посылает подтверждение аутентификации, содержащее Challenge Text.
3. Абонент шифрует Challenge Text своим статическим WEP-ключом и посылает точке радиодоступа запрос аутентификации.

4. Если точка радиодоступа в состоянии успешно расшифровать запрос аутентификации и содержащийся в нем Challenge Text, она посылает абоненту подтверждение аутентификации, таким образом предоставляя доступ к сети.

Рисунок 5.12. Аутентификация с общим ключом.

1.4. Аутентификация по MAC-адресу

Аутентификация абонента по его MAC-адресу не предусмотрена стандартом IEEE 802.11, однако поддерживается многими производителями оборудования для беспроводных сетей, в том числе D-Link. При аутентификации по MAC-адресу происходит сравнение MAC-адреса абонента либо с хранящимся локально списком разрешенных адресов легитимных абонентов, либо с помощью внешнего сервера аутентификации (рисунок 5.13). Аутентификация по MAC-адресу используется в дополнение к открытой аутентификации и аутентификации с общим ключом стандарта IEEE 802.11 для уменьшения вероятности доступа посторонних абонентов.

Рисунок 5.13. Аутентификация с помощью внешнего сервера.

1.5. Стандарт IEEE 802.1x/EAP (Enterprise-режим)

Проблемы, с которыми столкнулись разработчики и пользователи сетей на основе стандарта IEEE 802.11, вынудили искать новые решения защиты беспроводных сетей. Были выявлены компоненты, влияющие на системы безопасности беспроводной локальной сети:

1. Архитектура аутентификации.
2. Механизм аутентификации.
3. Механизм обеспечения конфиденциальности и целостности данных.

Архитектура аутентификации IEEE 802.1x – стандарт IEEE 802.1x описывает единую архитектуру контроля доступа к портам с использованием разнообразных методов аутентификации абонентов.

Алгоритм аутентификации Extensible Authentication Protocol или EAP (расширяемый протокол идентификации) поддерживает централизованную аутентификацию элементов инфраструктуры беспроводной сети и ее пользователей с возможностью динамической генерации ключей шифрования.

Архитектура IEEE 802.1x

Архитектура IEEE 802.1x включает в себя следующие обязательные логические элементы (рисунок 5.14):

1. Клиент (Supplicant) находится в операционной системе абонента.
2. Аутентификатор (Authenticator) находится в программном обеспечении точки радиодоступа.
3. Сервер аутентификации (Authentication Server) находится на RADIUS-сервере.

IEEE 802.1x предоставляет абоненту беспроводной локальной сети лишь средства передачи атрибутов серверу аутентификации и допускает использование различных методов и алгоритмов аутентификации. Задачей сервера аутентификации является поддержка разрешенных политикой сетевой безопасности методов аутентификации.

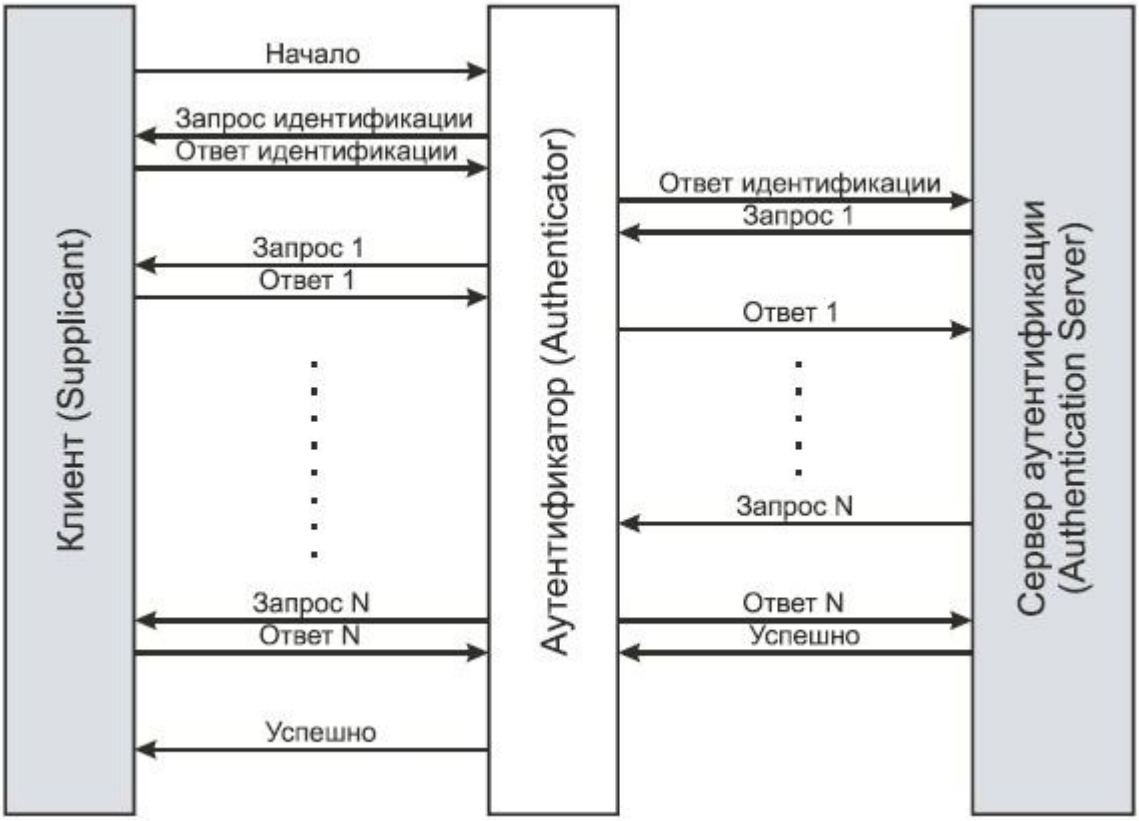


Рисунок 5.14. Архитектура IEEE 802.1x.

Аутентификатор, находясь в точке радиодоступа, создает логический порт для каждого клиента на основе его идентификатора ассоциирования. Логический порт имеет два канала для обмена данными. Неконтролируемый канал беспрепятственно пропускает трафик из беспроводного сегмента в проводной и обратно, в то время как контролируемый канал требует успешной аутентификации для прохождения фреймов.

Таким образом, в терминологии стандарта 802.1x точка доступа играет роль коммутатора в проводных сетях Ethernet. Очевидно, что проводной сегмент сети, к которому подключена точка доступа, нуждается в сервере аутентификации. Его функции обычно выполняет RADIUS-сервер, интегрированный с той или иной базой данных пользователей, в качестве которой может выступать стандартный RADIUS, LDAP, NDS или Windows Active Directory. Коммерческие беспроводные шлюзы высокого класса могут реализовывать как функции сервера аутентификации, так и аутентификатора.

Клиент активизируется и ассоциируется с точкой радиодоступа (или физически подключается к сегменту в случае проводной локальной сети). Аутентификатор распознает факт

подключения и активизирует логический порт для клиента, сразу переводя его в состояние "неавторизован". В результате через клиентский порт возможен лишь обмен трафиком протокола IEEE 802.1x, для всего остального трафика порт заблокирован. Клиент также может (но не обязан) отправить сообщение EAP Start (начало аутентификации EAP) (рисунок 5.15) для запуска процесса аутентификации.

Аутентификатор отправляет сообщение EAP Request Identity (запрос имени EAP) и ожидает от клиента его имя (Identity). Ответное сообщение клиента EAP Response (ответ EAP), содержащее атрибуты, перенаправляется серверу аутентификации.

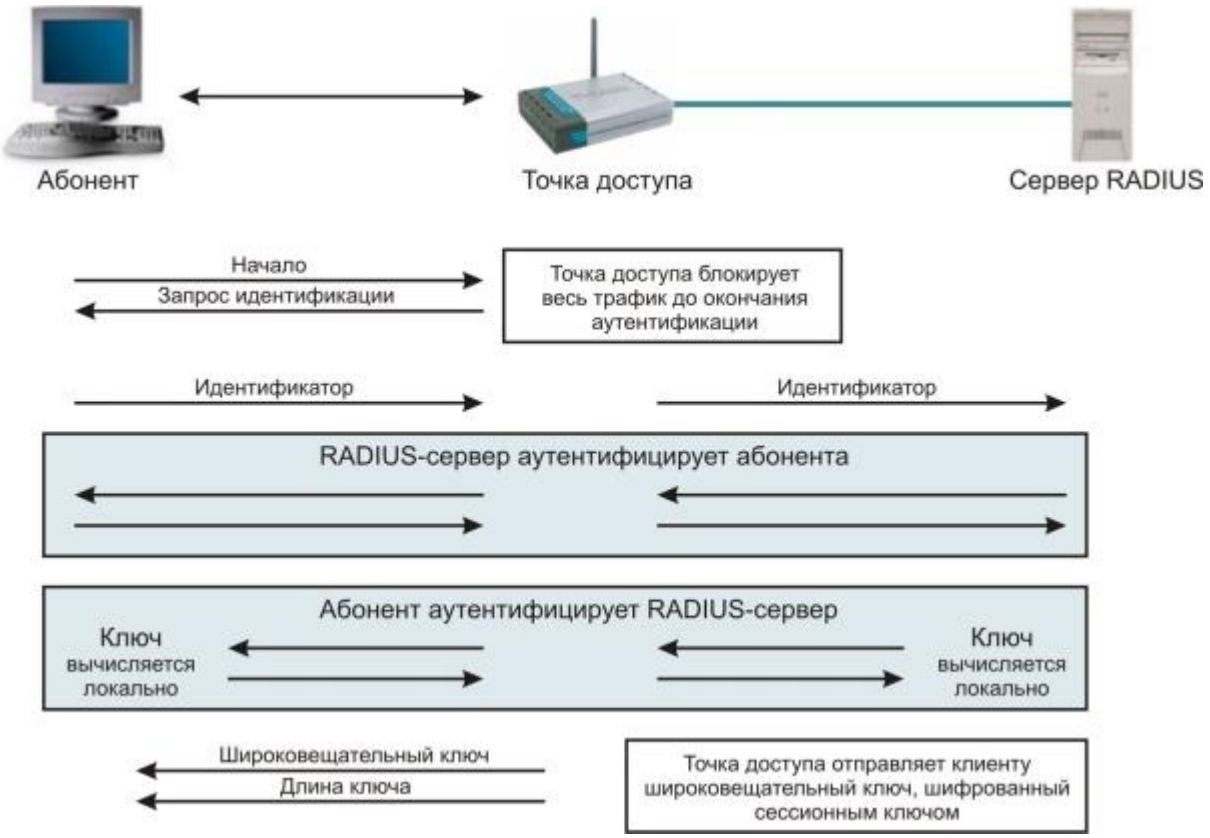


Рисунок 5.15. Обмен сообщениями в 802.1x/EAP.

После завершения аутентификации сервер отправляет сообщение RADIUS-ACCEPT (принять) или RADIUS-REJECT (отклонить) аутентификатору. При получении сообщения RADIUS-ACCEPT аутентификатор переводит порт клиента в состояние "авторизован", и начинается передача всего трафика абонента.

Механизм аутентификации

Первоначально стандарт 802.1x задумывался для того, чтобы обеспечить аутентификацию пользователей на канальном уровне в коммутируемых проводных сетях.
Алгоритмы аутентификации стандарта 802.11 могут обеспечить клиента динамическими, ориентированными на пользователя ключами. Но тот ключ, который создается в процессе аутентификации, не является ключом, используемым для шифрования фреймов или проверки целостности сообщений. В стандарте WPA для получения всех ключей используется так называемый мастер-ключ (Master Key).
Механизм генерации ключей шифрования осуществляется следующим образом:

1. Клиент и точка доступа устанавливают динамический ключ (он называется парный мастер-ключ, или PMK, Pairwise Master Key), полученный в процессе аутентификации по стандарту 802.1x.

2. Точка доступа посылает клиенту секретное случайное число, которое называется временный аутентификатор (Authenticator Nonce, ANonce), используя для этого сообщение EAPoL-Key стандарта 802.1x.
3. Этот клиент локально генерирует секретное случайное число, называемое временный проситель (Supplicant Nonce, SNonce).
4. Клиент генерирует парный переходный ключ (Pairwise Transient Key, PTK) путем комбинирования PMK, SNonce, ANonce, MAC-адреса клиента, MAC-адреса точки доступа и строки инициализации. MAC-адреса упорядочены, MAC-адреса низшего порядка предшествуют MAC-адресам высшего порядка. Благодаря этому гарантируется, что клиент и точка доступа "выстроят" MAC-адреса одинаковым образом (рисунок 5.16).
5. Это комбинированное значение пропускается через псевдослучайную функцию (Pseudo Random Function, PRF), чтобы получить 512-разрядный PTK.
6. Клиент посылает число SNonce, сгенерированное им на этапе 3, точке доступа с помощью сообщения EAPoL-Key стандарта 802.1x, защищенного ключом EAPoL-Key MIC.
7. Точка доступа использует число SNonce для вычисления PTK таким же образом, как это сделал клиент.
8. Точка доступа использует выведенный ключ EAPoL-Key MIC для проверки целостности сообщения клиента.
9. Точка доступа посылает сообщение EAPoL-Key, показывающее, что клиент может установить PTK и его ANonce, защищенные ключом EAPoL-Key MIC. Данный этап позволяет клиенту удостовериться в том, что число ANonce, полученное на этапе 2, действительно.
10. Клиент посылает сообщение EAPoL-Key, защищенное ключом EAPoL-Key MIC, указывающее, что ключи установлены.

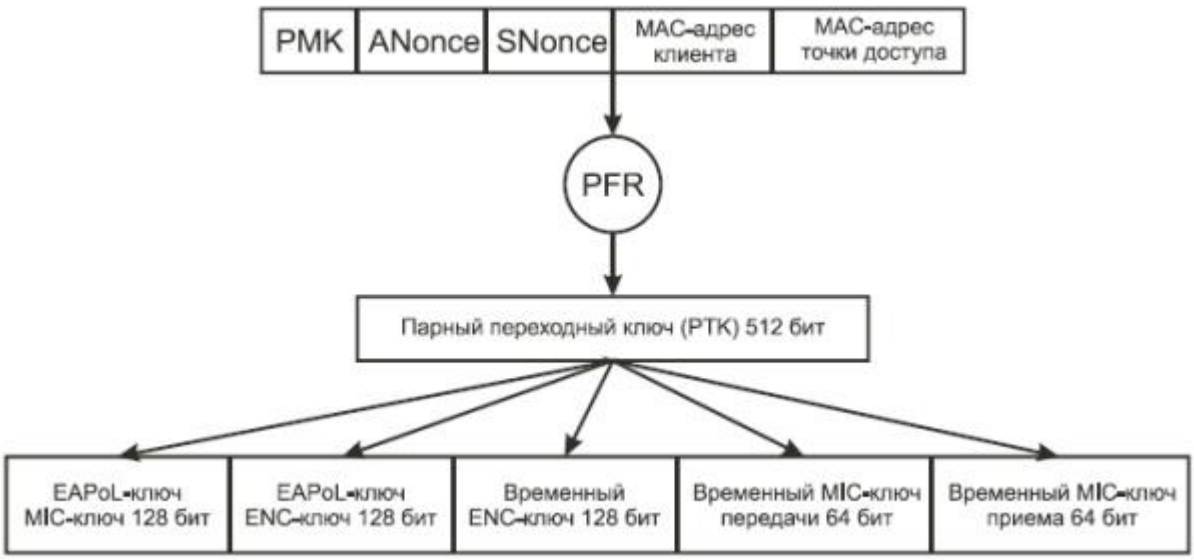


Рисунок 5.16. Генерация парного переходного ключа.

Парный мастер-ключ (PMK) и парный переходный ключ (PTK) являются одноадресными. Они только шифруют и дешифруют одноадресные фреймы, и предназначены для единственного пользователя. Широковещательные фреймы требуют отдельной иерархии ключей, потому что использование с этой целью одноадресных ключей приведет к резкому возрастанию трафика сети. Точке доступа (единственному объекту BSS, имеющему право на рассылку широковещательных или многоадресных сообщений) пришлось бы посылать один и тот же широковещательный или многоадресный фрейм, зашифрованный соответствующими пофреймовыми ключами, каждому пользователю.

Широковещательные или многоадресные фреймы используют иерархию групповых ключей. Групповой мастер-ключ (Group Master Key, GMK) находится на вершине этой иерархии и выводится в точке доступа. Вывод GMK основан на применении PRF, в результате чего получается 256-разрядный GMK. Входными данными для PRF-256 являются шифровальное секретное случайное число (или Nonce), текстовая строка, MAC-адрес точки доступа и значение времени в формате синхронизирующего сетевого протокола (NTP). На рисунке 5.17 представлена иерархия групповых ключей.

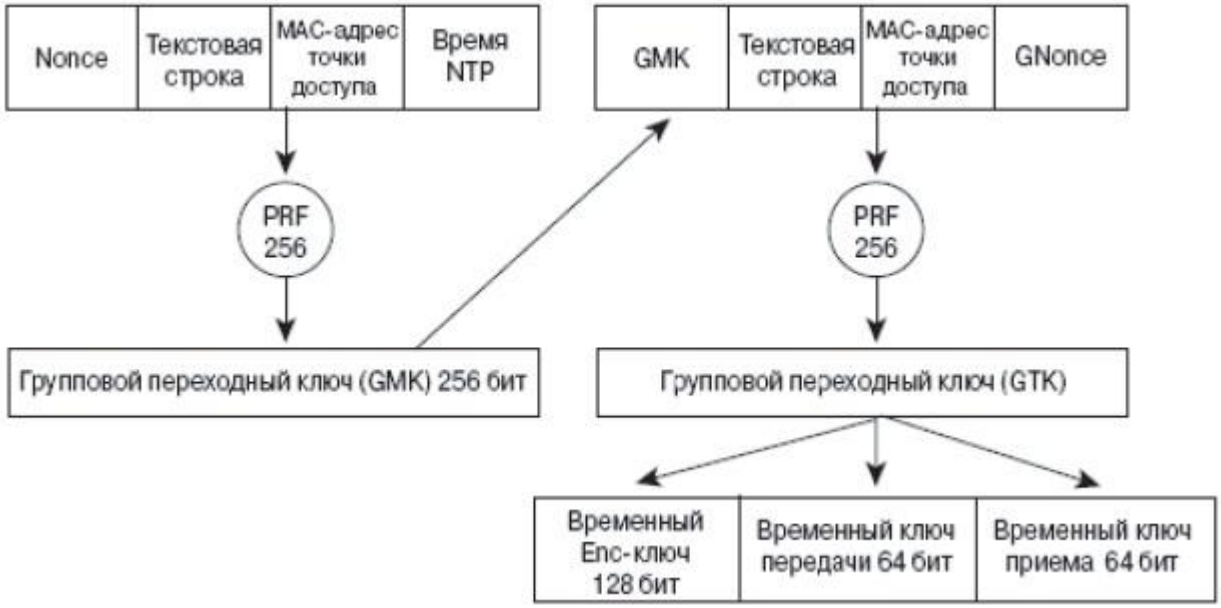


Рисунок 5.17. Иерархия групповых ключей.

Групповой мастер-ключ, текстовая строка, MAC-адрес точки доступа и GNonce (значение, которое берется из счетчика ключа точки доступа) объединяются и обрабатываются с помощью PRF, в результате чего получается 256-разрядный групповой переходный ключ (Group Transient Key, GTK). GTK делится на 128-разрядный ключ шифрования широковещательных/многоадресных фреймов, 64-разрядный ключ передачи MIC (transmit MIC key) и 64-разрядный ключ приема MIC (MIC receive key).

С помощью этих ключей широковещательные и многоадресные фреймы шифруются и дешифруются точно так же, как с помощью одноадресных ключей, полученных на основе парного мастер-ключа (PMK).

Клиент обновляется с помощью групповых ключей шифрования через сообщения EAPoL-Key Точка доступа посылает такому клиенту сообщение EAPoL, зашифрованное с помощью одноадресного ключа шифрования. Групповые ключи удаляются и регенерируются каждый раз, когда какая-нибудь станция диссоциируется или деаутентифицируется в BSS. Если происходит ошибка MIC, одной из мер противодействия также является удаление всех ключей с имеющей отношение к ошибке приемной станции, включая групповые ключи.

В домашних сетях или сетях, предназначенных для малых офисов, развертывание RADIUS-сервера с базой данных конечных пользователей маловероятно. В таком случае для генерирования сеансовых ключей используется только предварительно разделенный PMK (вводится вручную). Это аналогично тому, что делается в оригинальном протоколе WEP.

Поскольку в локальных сетях 802.11 нет физических портов, ассоциация между беспроводным клиентским устройством и точкой доступа считается сетевым портом доступа. Беспроводный клиент рассматривается как претендент, а точка доступа – как аутентификатор.

В стандарте 802.1x аутентификация пользователей на канальном уровне выполняется по протоколу EAP, который был разработан Группой по проблемам проектирования Internet (IETF). Протокол EAP - это замена протокола CHAP (Challenge Handshake Authentication Protocol - протокол взаимной аутентификации), который применяется в PPP (Point to Point Protocol - протокол соединения "точка-точка"), он предназначен для использования в локальных сетях. Спецификация EAPOL определяет, как фреймы EAP инкапсулируются во фреймы 802.3, 802.5 и 802.11. Обмен фреймами между объектами, определенными в стандарте 802.1x, схематично изображен на рисунке 5.18.

EAP является "обобщенным" протоколом в системе аутентификации, авторизации и учета (Authentication, Authorization, and Accounting - AAA), обеспечивающим работу разнообразных методов аутентификации. AAA-клиент (сервер доступа в терминологии AAA, в беспроводной сети представлен точкой радиодоступа), поддерживающий EAP, может не понимать конкретных методов, используемых абонентом и сетью в процессе аутентификации. Сервер доступа туннелирует сообщения протокола аутентификации, циркулирующие между абонентом и сервером аутентификации. Сервер доступа интересует лишь факт начала и окончания процесса аутентификации.

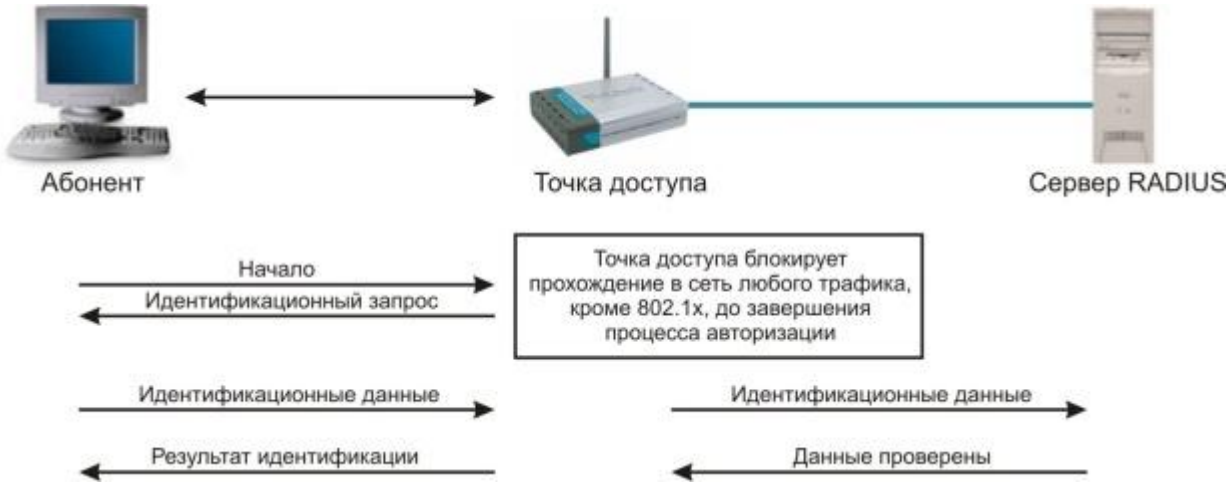


Рис. 5.18. Механизм аутентификации в 802.1x/EAP.

Есть несколько вариантов EAP, спроектированных с участием различных компаний-производителей. Такое разнообразие вносит дополнительные проблемы совместимости, так что выбор подходящего оборудования и программного обеспечения для беспроводной сети становится нетривиальной задачей. При конфигурировании способа аутентификации пользователей в беспроводной сети вам, вероятно, придется столкнуться со следующими вариантами EAP:

- ✓ EAP-MD5 - это обязательный уровень EAP, который должен присутствовать во всех реализациях стандарта 802.1x, именно он был разработан первым. С точки зрения работы он дублирует протокол CHAP. Мы не рекомендуем пользоваться протоколом EAP-MD5 по трем причинам. Во-первых, он не поддерживает динамическое распределение ключей. Во-вторых, он уязвим для атаки "человек посередине" с применением фальшивой точки доступа и для атаки на сервер аутентификации, так как аутен-

тифицируются только клиенты. И наконец, в ходе аутентификации противник может подслушать запрос и зашифрованный ответ, после чего предпринять атаку с известным открытым или шифрованным текстом;

- ✓ EAP-TLS (EAP-Transport Layer Security - протокол защиты транспортного уровня) поддерживает взаимную аутентификацию на базе сертификатов. EAP-TLS основан на протоколе SSLv3 и требует наличия удостоверяющего центра. Протоколы TLS и SSL используют ряд элементов инфраструктуры PKI (Public Key Infrastructure): Абонент должен иметь действующий сертификат для аутентификации по отношению к сети. AAA-сервер должен иметь действующий сертификат для аутентификации по отношению к абоненту. Орган сертификации с сопутствующей инфраструктурой управляет сертификатами субъектов PKI. Клиент и RADIUS-сервер должны поддерживать метод аутентификации EAP-TLS. Точка радиодоступа должна поддерживать процесс аутентификации в рамках 802.1x/EAP, хотя может и не знать деталей конкретного метода аутентификации. Общий вид EAP-TLS выглядит примерно так (рисунок X).

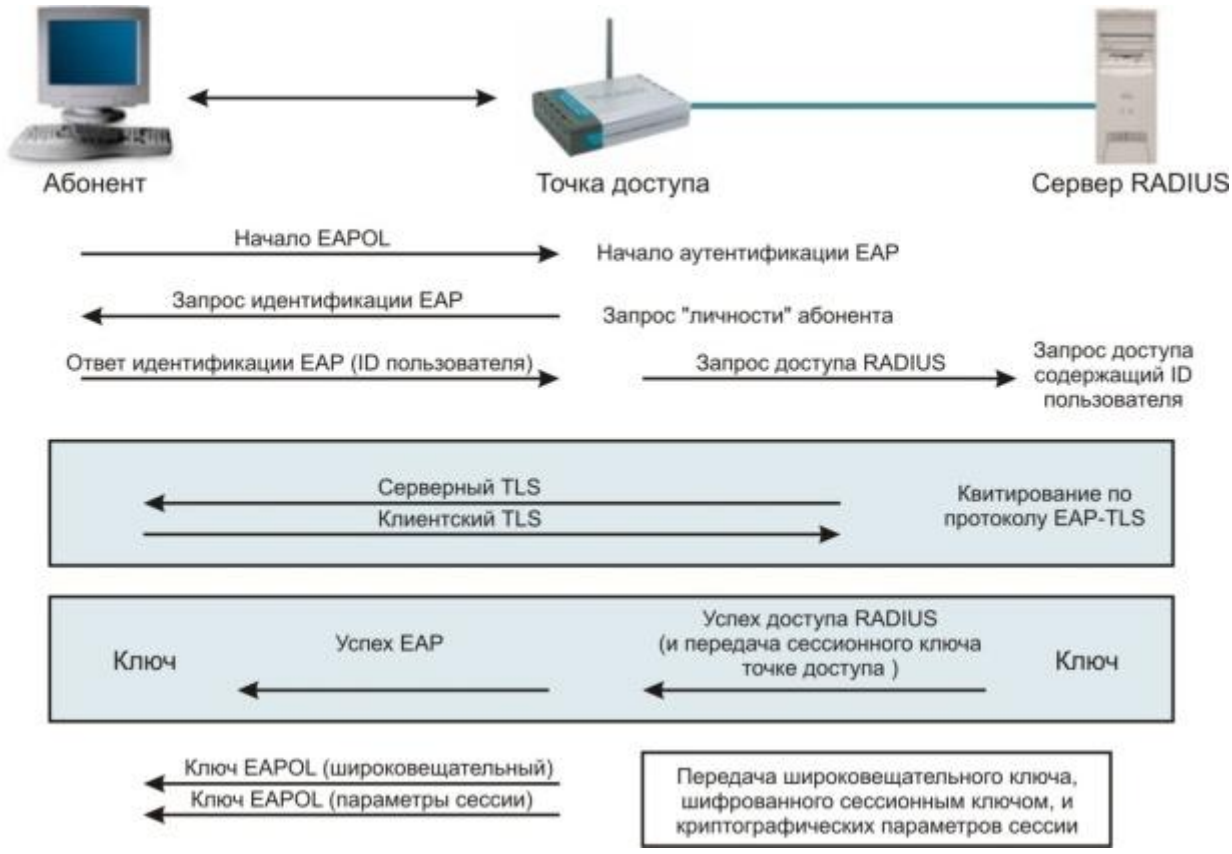


Рисунок X. Процесс аутентификации EAP-TLS.

- ✓ EAP-LEAP (Lightweight EAP, облегченный EAP) - это запатентованный компанией Cisco вариант EAP, реализованный в точках доступа и беспроводных клиентских картах Cisco. LEAP был первой (и на протяжении длительного времени единственной) схемой аутентификации в стандарте 802.1x, основанной на паролях. Поэтому LEAP приобрел огромную популярность и даже поддержан в сервере Free-RADIUS, несмотря на то, что это запатентованное решение. Сервер аутентификации посылает клиенту запрос, а тот должен вернуть пароль, предварительно выполнив его свертку со строкой запроса. Основанный на применении паролей, EAP-LEAP аутентифицирует пользователя, а не устройство. В то же время очевидна уязвимость этого варианта для атак методом полного перебора и по словарю, нехарактерная для методов аутентификации с применением сертификатов.

- ✓ PEAP (Protected EAP - защищенный EAP) и EAP-TTLS (Tunneled Transport Layer Security EAP, протокол защиты транспортного уровня EAP), разработанный компанией Certicom and Funk Software. Эти варианты также достаточно развиты, и поддерживаются производителями, в частности D-link. Для работы EAP-TTLS требуется, чтобы был сертифицирован только сервер аутентификации, а у претендента сертификата может и не быть, так что процедура развертывания упрощается. EAP-TTLS поддерживает также ряд устаревших методов аутентификации, в том числе PAP, CHAP, MS-CHAP, MS-CHAPv2 и даже EAP-MD5. Чтобы обеспечить безопасность при использовании этих методов, EAP-TTLS создает зашифрованный по протоколу TLS туннель, внутри которого эти протоколы и работают. Примером практической реализации EAP-TTLS может служить программное обеспечение для управления доступом в беспроводную сеть Odyssey от компании Funk Software. Протокол PEAP очень похож на EAP-TTLS, только он не поддерживает устаревших методов аутентификации типа PAP и CHAP. Вместо них поддерживаются протоколы PEAP-MS-CHAPv2 и PEAP-EAP-TLS, работающие внутри безопасного туннеля. Поддержка PEAP реализована в пакете программ точек доступа D-link и успешно реализована в Windows XP, начиная с Service Pack 2. В общем виде схема обмена PEAP выглядит следующим образом (рисунок X).

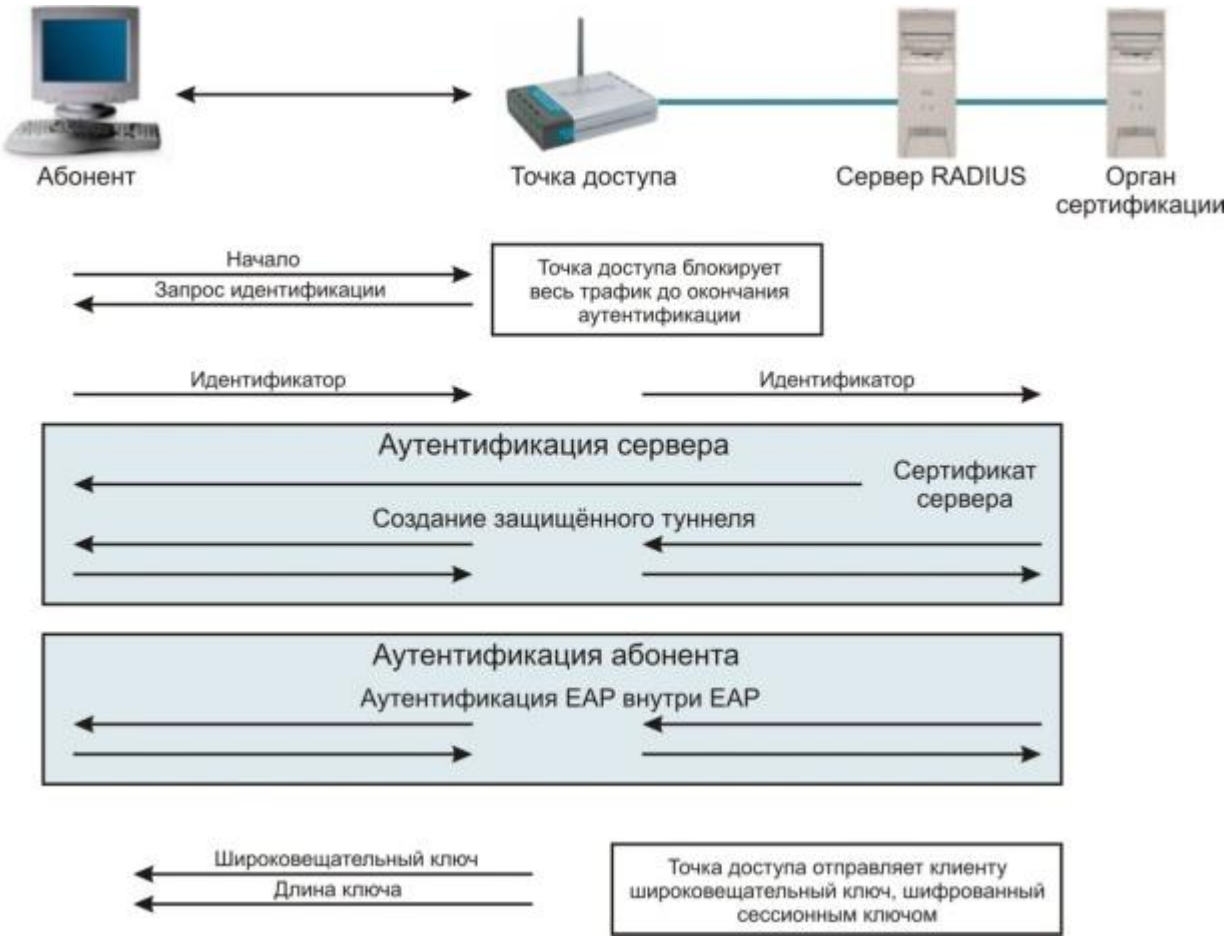


Рисунок X. Процесс аутентификации PEAP.

- ✓ Еще два варианта EAP – это EAP-SIM и EAP-AKA для аутентификации на базе SIM и USIM. В настоящий момент оба имеют статус предварительных документов IETF и в основном предназначены для аутентификации в сетях GSM, а не в беспроводных сетях 802.11. Тем не менее, протокол EAP-SIM поддержан в точках доступа и клиентских устройствах некоторых производителей.

Наглядно уровни архитектуры 802.1x показаны на рисунке 5.21. Здесь в качестве механизма обеспечения конфиденциальности и целостности данных выступают стандарты шифрования WPA и WPA2.

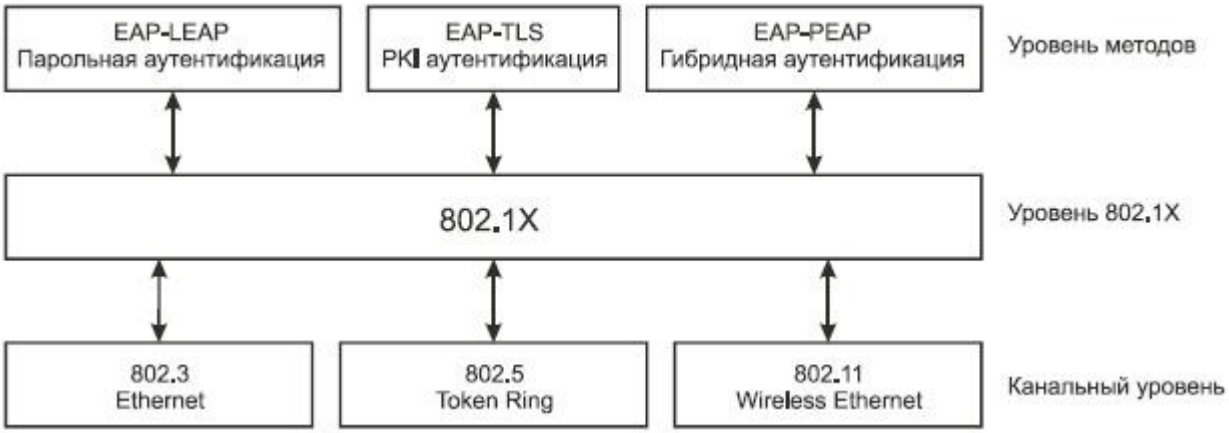


Рисунок 5.21. Уровни архитектуры 802.1x.

4. Дополнительные механизмы защиты

4.1. SSID

Идентификатор беспроводной сети (SSID) изначально не вошел в состав IEEE 802.11 как базовый механизм защиты. Для подключения к беспроводной сети, нужно знать её имя или SSID. Современные точки доступа оснащены функцией отключения широковещания SSID, т.е. подключится к сети могут только те клиенты которым известен SSID. SSID представляет собой элементарный механизм защиты, но уже сейчас многие сканеры беспроводных сетей способны извлекать имя сети даже при отключение широковещания. Поэтому применения данного механизма хорошо сочетается с другими методами защиты для уменьшения риска взлома беспроводной сети.

VI. Общие механизмы обеспечения безопасности локальных сетей

Общим механизмом обеспечения безопасности как для проводных, так и беспроводных локальных сетей является стандарт IEEE 802.1x, описанный выше для обоих случаев.

VII. Механизмы межсетевой безопасности

Службами, реализующими механизмы межсетевой безопасности, являются:

- ✓ межсетевые экраны (МЭ);
- ✓ системы трансляции адресов и портов (СТАП);
- ✓ системы обнаружения атак и вторжений (СОАВ);
- ✓ системы туннелирования.

Системы трансляции адресов (Network Address Translation, NAT) обычно являются компонентой МЭ. Службы межсетевой безопасности обычно устанавливаются и работают на маршрутизаторах. Дополнительно к этому межсетевые экраны (МЭ) и системы обнаружения атак (СОАВ) желательно устанавливать и на компьютеры, находящиеся во внешнем периметре сети.

1. Межсетевые экраны

1.1. Архитектура межсетевого экрана

Межсетевой экран (брандмауэр, файерволл) – это комплекс программных или аппаратных средств, который позволяет реализовать набор правил, определяющих условия прохождения пакетов между сетями. Межсетевой экран включает следующие средства (рисунок 7.1):

- фильтр пакетов;
- фильтр состояний;
- транслятор адресов;
- транспортный шлюз;
- шлюз приложений (прокси-сервер).

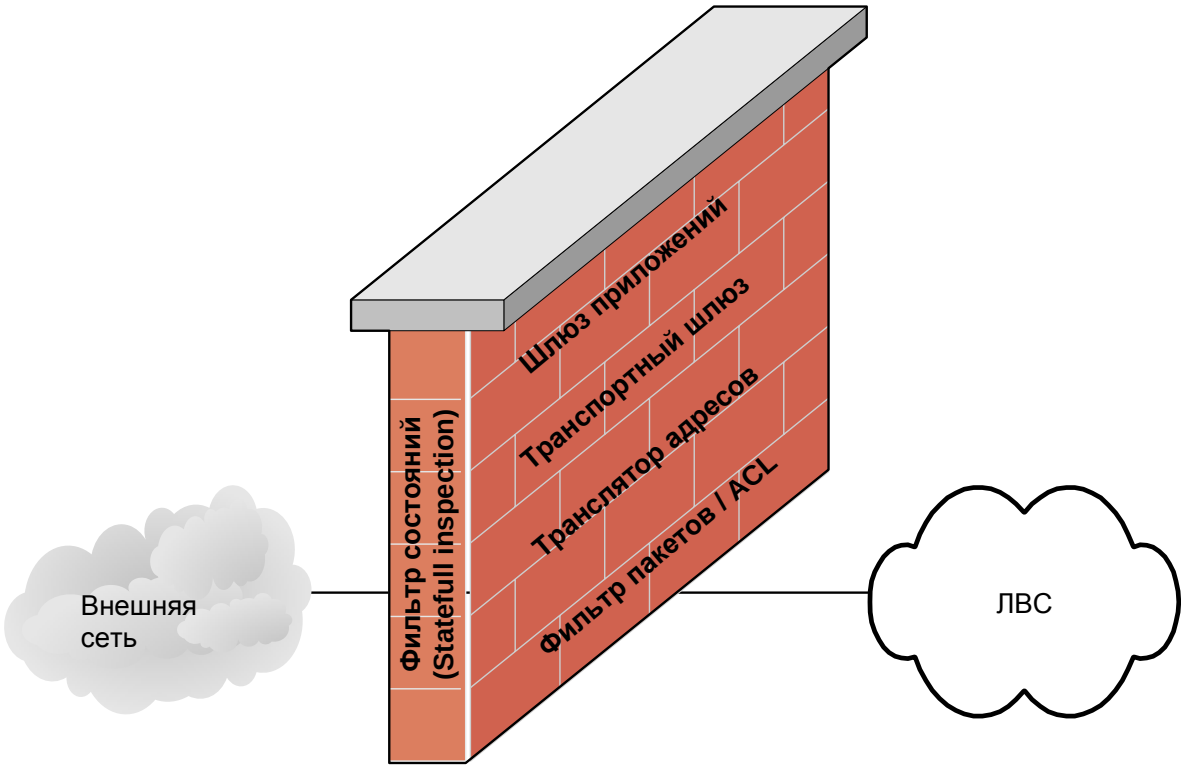


Рисунок 7.1. Упрощенная архитектура межсетевого экрана.

1.2. Фильтр пакетов

ФП разрешает или запрещает прохождение пакета в соответствии с заданными администратором правилами. Эти правила опираются на информацию, имеющуюся в заголовке каждого пакета и в сетевой системе:

- IP-адрес источника и назначения;
- протокол Транспортного уровня;
- порт источника и назначения;
- Флаги;
- Тип сообщения ICMP;
- Входящий и исходящий сетевой интерфейс.

Таким образом, для описания правил прохождения пакетов составляются таблицы типа:

Действие	Тип пакета	Адрес источника	Порт источника	Адрес назначения	Порт назначения	Флаги	Входящий интерфейс	Исходящий интерфейс
DROP	TCP	194.186.1.2	80	-	-	-	-	
ACCEPT	UDP	83.142.162.49	53	-	-	-	-	
ACCEPT	ICMP	-	-	-	-	-	-	
DROP	-	-	-	-	-	-	eth1	eth0

Фильтры работают быстро, поскольку они просто просматривают информацию о пакете при принятии решения. Но фильтры пакетов имеют один существенных недостаток – они не в состоянии отслеживать конкретный сетевой сеанс и не в состоянии анализировать содержимое сообщения конкретного приложения, которое в свою очередь может содержать зловредный код.

1.3. Фильтр инспекции состояний (statefull фильтры)

Технология инспекции состояний (statefull inspection) используется многими разработчиками, но, поскольку наименование запатентовано компанией Check Point, они вынуждены присваивать ему различные наименования (expert inspection, smart filtering, adaptive screening, multilevel inspection и др.).

Данная технология является дальнейшим развитием фильтра пакетов и включает следующую дополнительную функцию – поддержка таблицы состояний TCP-соединений и контроль сессий на её основе.

Жизненый цикл TCP-соединения имеет несколько фаз, отраженных на рисунке 7.2. Таблица состояний (netstat) содержит информацию о соединение и состояние, в котором находится данное соединение. На основе этой информации происходит фильтрация пакетов. Например, если не был произведн запрос, то пакет-ответа не будет пропушен в локальную сеть.

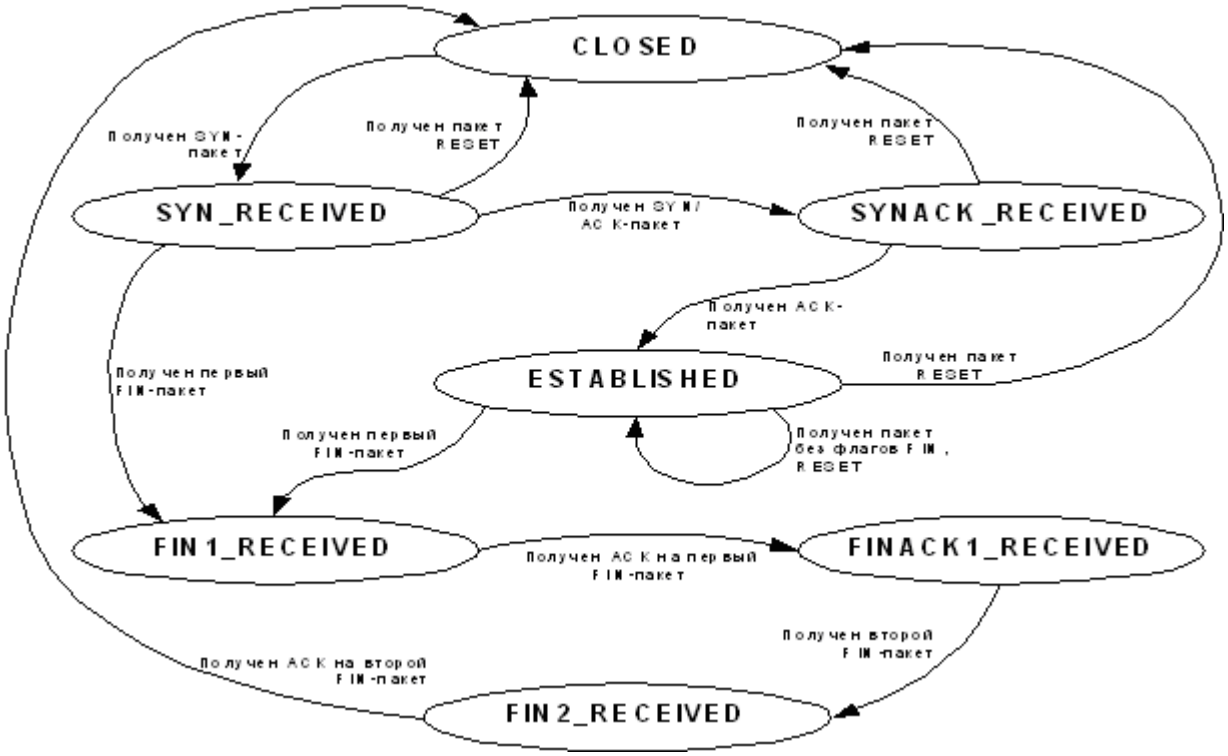


Рисунок 7.2. Диаграмма состояний протокола TCP.

1.4. Транслятор адресов

Трансляция адресов – технология, позволяющая изменять такие параметры IP-пакета, как адрес и порт источника пакета, адрес и порт назначения. При этом ведётся таблица измененных данных, и при получении обратного пакета происходит обратная трансляция адреса, если она нужна. Существуют 2 причины, по которым имеет смысл осуществлять трансляцию адресов:

- ✓ предоставление компьютерам локальной сети полноценного доступа в Интернет при условии, что имеющееся число внешних адресов или подключений к провайдеру меньше, чем число компьютеров, которым необходим доступ в Интернет. Стоит учесть, что некоторые протоколы не поддерживают трансляцию адресов (например, PPTP), либо использование трансляции накладывает ограничения на использование служб (например, пассивный режим FTP был разработан для решения проблемы NAT). Для обхода таких проблем маршрутизатор должен иметь возможность разбирать пакеты, вносить изменения и собирать их снова;
- ✓ скрытие внутренней структуры локальной сети. За счёт выполнения трансляции адресов запросы компьютеров локальной сети выглядят так, будто выполняются с одного и того же компьютера. Однако на самом деле это могут быть запросы от разных машин. Также можно с помощью трансляции адреса совместно с портом осуществлять скрытие серверной инфраструктуры.

Существуют 2 способа трансляции адресов:

- ✓ Network Address Translation (NAT) – замена адрес источника на адрес маршрутизатора. При этом порт остаётся неизменным;
- ✓ Static Address Translation (SAT) – замена адрес источника или приёмника на некоторый адрес. Возможна одновременная замена порта. SAT подразделяется на 2 типа:
  - Source SAT – трансляция адреса источника
  - Destination SAT – трансляция адреса назначения

Типичный пример применения NAT

Некоторой организации необходимо предоставить сотрудникам доступ в интернет. Приобретается выделенная линия до провайдера, настраивается трансляция адресов на маршрутизаторе из частных во внешний

Типичный пример применения DSAT

Необходимо перенаправить соединение по порту 80/tcp (HTTP-трафик) на кэширующий сервер

Типичный пример применения SSAT

Организация приобрела не один, а 2 канала, но маршрутизатор один. Тогда настраивается SSAT, преобразование производится в адрес интерфейса, подключенного к нужному каналу. Второй канал может использоваться, например, для подключения серверной фермы.

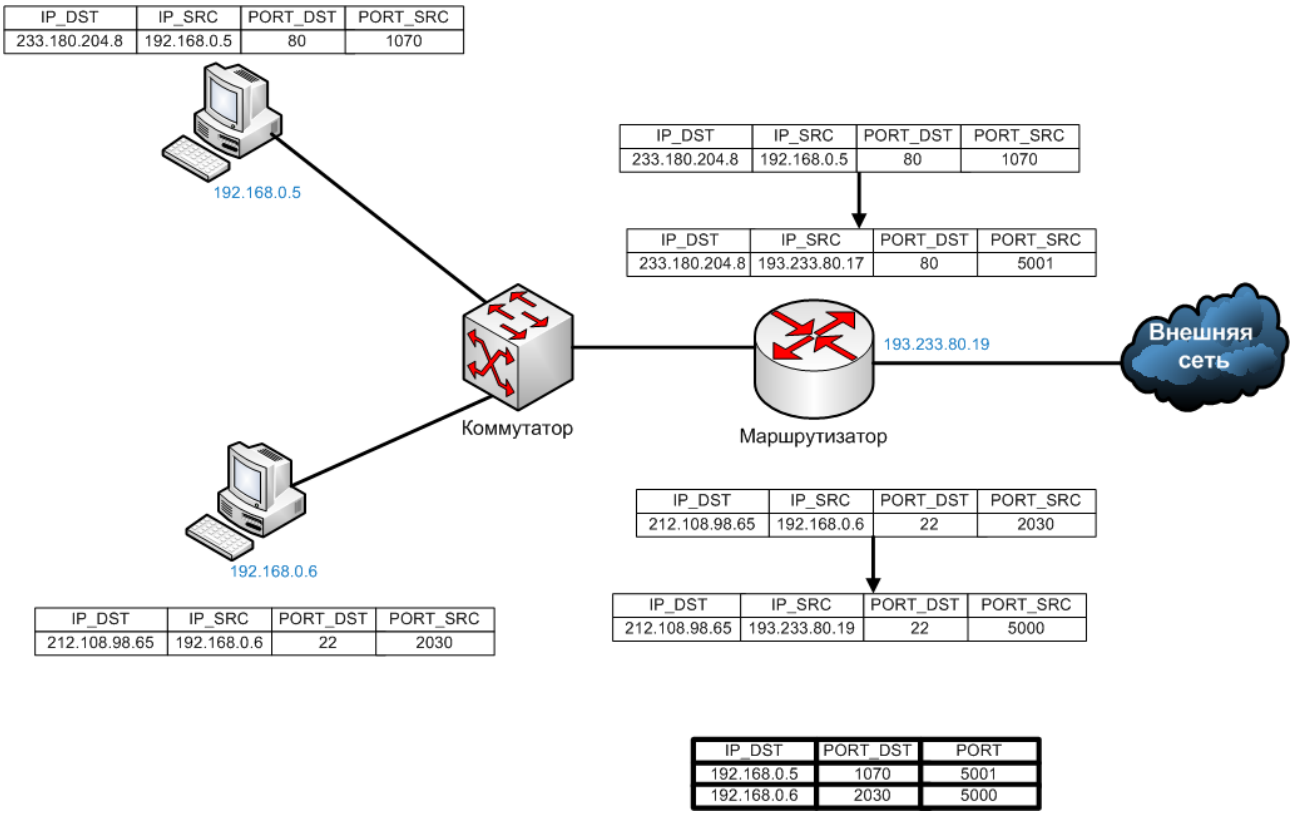


Рисунок 7.3. Пример трансляции сетевых адресов.

ОС Linux содержит системную службу netfilter, которая реализует функции фильтра пакетов (в том числе statefull), трансляции адресов, прозрачного прокси и журналирования трафика. Для управления этой службой имеется утилита iptables.

1.5. Транспортные шлюзы (виртуальные сервера)

Транспортный шлюз представляет из себя транслятор TCP соединения. Пользователь образует соединение с определенным портом на брандмауэре, после чего последний производит соединение с местом назначения по другую сторону от брандмауэра. Во время сеанса этот транслятор копирует байты в обоих направлениях, действуя как провод.

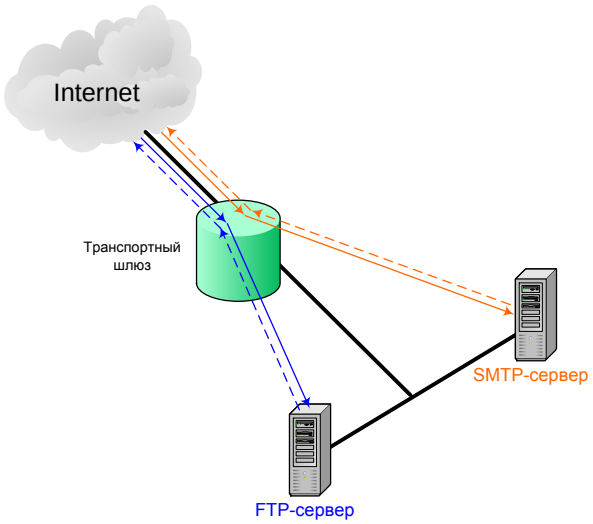


Рисунок 7.4. Схема работы транспортного шлюза.

Такой тип сервера позволяет создавать транслятор для любого определенного пользователем сервиса, базирующегося на TCP, осуществлять контроль доступа к этому сервису, сбор статистики по его использованию. Наиболее распространенным транспортным шлюзом для Linux-платформ является утилита xinetd.

1.6. Шлюзы приложений (прокси-сервера)

Технология проксирования (прокси-серверов) является частью межсетевых экранов (МЭ). Прокси-сервера относят к МЭ прикладного уровня. Из многочисленных значений английского слова «proxy» в данном контексте применимы такие: «доверенное лицо», «полномочный представитель». То есть некто, кто действует от вашего имени по вашему поручению вместо вас. В компьютерах прокси-сервер – это служба, которая передает запросы ваших программ (браузеров и других) в Интернет, получает ответы и передает их обратно.

Назначение и принцип работы прокси-серверов

Функции прокси-сервера:

1. Трансляция сетевых адресов.
2. Кэширование – за счет кэширования страниц на прокси-сервере снижается трафик работы в Интернет и происходит ускорение доступа пользователей. Если запрашиваемый файл уже скачивался, то можно выдать его с диска прокси-сервера без обращения в Интернет. Статистика показывает, что экономия составляет 10-15% от общего скачанного объёма трафика.

3. Протоколирования работы в Интернете – обычно обращения пользователей в Интернет протоколируются и на основе логов делаются отчеты, например, об объеме трафика, скачанного пользователем.
4. Управление разрешениями – можно определить, каким пользователям/в какое время/по каким протоколам/к каким ресурсам/в каком объеме можно выходить в Интернет.
5. Фильтрации трафика - например, режутся баннерные сети, спам-трафик по SMTP и т.п.
6. Анализ трафика на наличие зловредного кода.
7. Дополнительные возможности, например, фильтрация пакетов, средств обнаружения вторжений и т.д.

Исходя из функций прокси-сервера, становится ясно, что для каждого протокола прикладного уровня должен существовать свой прокси-сервер. Исходя из этого существуют следующие прокси-сервера:

1. HTTP-прокси**.** Он предназначен для организации работы браузеров и других программ, использующих протокол HTTP. Браузер передает прокси-серверу URL ресурса, прокси-сервер получает его с запрашиваемого веб-сервера и отдает браузеру. Для реализации кэширования веб-серверы обычно вставляют в HTTP-заголовки специальные указания об сроке валидности страницы, чтобы браузеры и прокси имели это в виду. Хотя многие прокси-серверы можно настроить так, чтобы эти указания частично игнорировались – например, перечитывать страницу не чаще одного раза в день.
2. FTP-прокси бывает двух основных видов в зависимости от протокола работы самого прокси. С ftp-серверами этот прокси, конечно, всегда работает по протоколу FTP. А вот с клиентскими программами – браузерами и ftp-клиентами (CuteFTP, FAR, и др.) прокси может работать как по FTP, так и по HTTP. Второй способ удобнее для браузеров, т.к. исторически является для них «родным». Браузер запрашивает ресурс у прокси, указывая протокол целевого сервера в URL – http или ftp. В зависимости от этого прокси выбирает протокол работы с целевым сервером, а протокол работы с браузером не меняется – HTTP. Поэтому, как правило, функцию работы с FTP-серверами также вставляют в HTTP-прокси, т.е. HTTP-прокси, описанный выше, обычно с одинаковым успехом работает как с HTTP, так и с FTP-серверами. Но при «конвертации» протоколов FTP<->HTTP теряется часть полезных функций протокола FTP. Поэтому специализированные ftp-клиенты предпочитают и специальный прокси, работающий с обеими сторонами по FTP.
3. HTTPS-прокси – фактически часть HTTP-прокси. «S» в названии означает “secure”. Не смотря на то, что программно это часть HTTP-прокси, обычно HTTPS выделяют в отдельную категорию (и есть отдельное поле для него в настройке браузеров). Обычно протокол HTTPS применяют, когда требуется передача конфиденциальной информации, – все передаваемые данные при этом шифруется. Прокси-серверу при этом дается только команда «соединится с таким-то сервером», и после соединения прокси передает в обе стороны шифрованный трафик, не имея возможности узнать подробности (соответственно и многие средства управления доступом – такие как фильтрация картинок – не могут быть реализованы для HTTPS, так как прокси в этом случае неизвестно, что именно передается). Собственно в процессе шифрации/дешифрации прокси тоже участия не принимает – это делают клиентская программа и целевой сервер. Наличие команды «соединиться с таким-то сервером» в HTTPS-прокси приводит к интересному и полезному побочному эффекту, которым все чаще пользуются разработчики клиентских программ. Так как после соединения с указанным сервером HTTPS-прокси лишь пассивно передает данные в обе стороны, не производя никакой обработки этого потока вплоть до отключения клиента или сервера, это позволяет использовать прокси для передачи трафика почти любого

протокола прикладного уровня, а не только HTTP. То есть HTTPS-прокси одновременно является и простым POP3-прокси, SMTP-прокси, IMAP-прокси, NNTP-прокси и т.д. – при условии, что соответствующая клиентская программа умеет так эксплуатировать HTTPS-прокси (увы, далеко не все еще это умеют, но есть вспомогательные программы, «заворачивающие» трафик обычных клиентов через HTTPS-прокси). Никаких модификаций целевого сервера не требуется.

4. Mapping-прокси – способ заставить работать через прокси те программы, которые умеют работать с Интернетом только напрямую. При настройке такого прокси администратор создает как бы «копию» целевого сервера, но доступную через один из портов прокси-сервера для всех клиентов локальной сети – устанавливает локальное «отображение» заданного сервера. Например, пользователи локальной сети хотят работать с почтовым сервером mail.ru не через браузер, а с использованием почтовой программы Outlook Express или TheBat?. Эти программы не умеют работать через прокси. Простейший способ работать с mail.ru по POP3 через прокси – установить локальное отображение сервера pop.mail.ru. И в Outlook'ах вместо pop.mail.ru написать имя прокси-сервера и порт отображения. Outlook будет соединяться с прокси-сервером ("думая", что это почтовый сервер), а прокси при этом будет соединяться с pop.mail.ru и прозрачно передавать всю информацию между Outlook и pop.mail.ru, таким образом «превращаясь» на время соединения в POP3-сервер. Неудобство mapping-прокси в том, что для каждого необходимого внешнего сервера нужно вручную устанавливать отдельный порт на прокси. Но зато не требуется модификация ни серверов, ни клиентов.
5. Socks-прокси.

Большинство современных прокси-серверов поддерживают все упомянутые протоколы.

Существующие прокси-сервера

Как правило, в настоящее время на предприятии практического любого размера сейчас используются прокси-серверы для выхода в Интернет. Несколько слов относительно самых распространенных решений:

- ✓ самое простое решение, которое встроено во все версии Windows (и серверные, и клиентские), начиная с Windows 98 SE – это применение Internet Connection Sharing. Это решение самое простое, но и наименее функциональное. Оно предназначено для совсем крохотных сетей и практически не поддерживает никаких функции, кроме прямого - преобразования сетевых адресов. Работать умеет только с внутренней сетью с IP-адресами 192.168.0.x, а также автоматически выполняет роль DHCP-сервера, раздающего адреса из этой сети. Как правило, администраторов больше волнует вопрос - как полностью запретить использование ICS на компьютерах пользователей (чтобы не мог появиться альтернативный сервер DHCP), чем вопросы его установки и настройки. (Обычно такое запрещение производится при помощи групповых политик). Многие администраторы считают, что ICS - вообще неработоспособное средство из-за проблем с DNS и соединениями по HTTPS;
- ✓ другое решение, встроенное в Windows (только серверные версии Windows 2000 и 2003) - это служба Network Address Translation (NAT), которая устанавливается и настраивается как компонент Routing and Remote Access Server. Это уже более работоспособное решение, однако функциональность его минимальна - не поддерживаются кэширование, разрешения для пользователей и групп, протоколирование и т.п. Тем не менее по причине простоты и малых требований к ресурсам этот продукт активно используется на предприятиях с 10-20 компьютерами;
- ✓ наиболее мощное решение от Microsoft - ISA Server 2004 Enterprise Edition. Именно этим средством пользуются на большинстве крупных и средних предприятий (в част-

ности, в Санкт-Петербурге - в Промстройбанке, Сбербанке, Центробанке, КНОС и т.п.). К преимуществам ISA Server можно отнести:

- масштабируемость и корпоративные возможности (работа в массивах серверов, совместное использование кэшей и т.п.)
- полная интеграция со средствами аутентификации Windows (практически только в ISA Server вы можете назначить права на выход в Интернет пользователям и группам Windows без необходимости установки дополнительных клиентов аутентификации);
- большое количество программных расширений (в основном третьих фирм), которые позволяют реализовать большое количество дополнительных функций (например, ограничить количество трафика, которое пользователь может скачать в течение дня), специализированные антивирусные пакеты и т.п.

К недостаткам можно отнести дороговизну ISA, большую ресурсоемкость, недостаточность средств защиты (обычно вместе с ISA используются дополнительные брандмауэры и системы обнаружения проникновений).

Из прокси-серверов третьих фирм под Windows к наиболее распространенным продуктам можно отнести Eserv, Eproxy, WinGate фирмы Qbik и WinRoute фирмы Kerio (сейчас сведен в один продукт с Kerio Firewall и переименован в Kerio WinRoute Firewall).

Kerio Winroute Firewall не требует обязательного наличия домена, как ISA Server, умеет (в отличие от ISA Server) выдавать квоты пользователям по трафику на день и месяц, пользователи при помощи него могут самостоятельно смотреть свою статистику и остаток трафика и т.п.

Возможно, самое распространенный прокси-сервер по числу инсталляций в мире – это SQUID, практически стандартный прокси-сервер для Unix-систем. Он менее ресурсоемок и считается более надежным и защищенным, чем ISA Server, однако защищенную аутентификацию стандартными средствами Windows не поддерживает. Во многих фирмах SQUID работает вместе с ISA: SQUID стоит на входе в локальную сеть предприятия, а ISA – за ним и обеспечивает раздачу разрешений пользователям и протоколирование по пользователям.

2. Системы обнаружения атак и вторжений

Межсетевые экраны, являясь первой линией эшелонированной обороны, не могут обеспечить полную адекватную защиту в силу следующих причин:

- ✓ ошибки или недостатки проектирования – различные технологии МЭ не охватывают всех возможностей проникновения в защищаемую сеть;
- ✓ недостатки реализации – поскольку каждый МЭ представляет собой сложный программный (программно-аппаратный) комплекс, его реализация содержит ошибки. Кроме того, не существует общей методологии тестирования, которая позволила бы определить качество программной реализации и убедиться, что в МЭ реализованы все специфицируемые свойства;
- ✓ недостатки применения (эксплуатации) – применение МЭ для реальной защиты сетей наталкивается на несколько серьезных обстоятельств, к числу которых можно отнести сложность реализации заданной сетевой политики безопасности механизмами МЭ, наличие ошибок конфигурирования МЭ и наличие ошибок администрирования. Администрирование МЭ является достаточно сложной задачей и требует от администратора МЭ определенной квалификации и опыта. Кроме того, атаки могут производиться и со стороны защищаемой сети, что осложняется тем, что они инициируются пользователями, которые, во-первых, имеют доступ к элементам сети, а во-вторых, знают или представляют организацию системы защиты.
- ✓ МЭ является «единственной точкой», через которую проходят все соединения с внешним миром. Поэтому любая программная ошибка, уязвимость ПО или ошибка конфигурирования МЭ могут привести к проникновению в защищаемую сеть.

Эти и подобные причины вызвали необходимость ведения подробных журналов аудита, в результате анализа которых специалисты (эксперты) делают заключение о совершившихся проникновениях. Результаты такого анализа используются для устранения причин нарушения или для формирования новых правил (сигнатур), позволяющих защититься от проникновений. Проведение подобного анализа осложняется множеством причин, среди которых основными являются:

- ✓ громадный объем данных журналов аудита;
- ✓ отсутствие значимых для обнаружения вторжений признаков в журналах;
- ✓ сложность анализа данных различных журналов;
- ✓ анализ по свершившимся фактам (получение данных об атаке, которая уже удачно прошла);
- ✓ необходимость высокой квалификации проводящего анализ специалиста;
- ✓ сложность правил (сигнатур) по обнаруженным признакам атаки.

Поэтому усилия исследователей направлены на разработку процессов обнаружения вторжений и автоматизацию процесса обнаружения. Подобные системы получили название систем обнаружения вторжений (Intrusion Detection System). При рассмотрении систем обнаружения вторжений (СОВ) используется предметная область, элементами которой часто являются общеупотребительные термины. Поскольку различное толкование терминов приводит к путанице, оговорим отдельные термины, которые будут использоваться в дальнейшем.

Атака – действия, предпринимаемые злоумышленником, против компьютера (или сети) потенциальной жертвы. С точки зрения нейтрального наблюдателя атака может быть безуспешной (неудачной) и успешной (удачной). Успешную атаку называют вторжением.

Вторжение – несанкционированный вход в информационную систему (в результате действий, нарушающих политику безопасности или обходящих систему защиты).

Система обнаружения вторжений – комплекс (аппаратура и программное обеспечение), который по результатам анализа контролируемых и собираемых данных принимает решение о наличии атаки или вторжения.

Ложная тревога (false positive) – генерация сигнала об обнаружении атаки (вторжения), которой не было.

Пропуск (false negative) – пропуск атаки или вторжения (отсутствие сигнала тревоги при наличии вторжения).

Общепринятый термин – система обнаружения вторжений – во многих случаях применяется и для систем обнаружения атак (СОА). В тех случаях, когда не возникает путаница, будем использовать сокращение СОВ.

2.1. Классификация СОВ

Для построения таксономии необходимо выбрать критерии, согласно которым будет проводиться классификация. Рассмотрим подход, в котором в качестве таких критериев выбраны типичные функции и особенности проектирования и реализации СОВ (классификация систем обнаружения вторжений представлена на рисунке 7.5).
К числу этих функций относятся следующие:

- ✓ подход к обнаружению;
- ✓ защищаемая система;
- ✓ структура СОВ;
- ✓ источник данных (для принятия решения);
- ✓ время анализа;
- ✓ характер реакции.

Подход к обнаружению. Выделяют два основных подхода – обнаружение сигнатур (signature detection, misuse detection), обнаружение аномалий (anomaly detection) – и гибридный подход.

Защищаемая система включает в себя хостовые, сетевые и гибридные СОВ. При контроле на уровне хоста можно рассматривать СОВ следующих подуровней: операционная система, база данных, приложение.

Сетевые СОВ осуществляют сбор и анализ сетевых пакетов, на основании которых проводится обнаружение. Сенсоры таких систем могут быть распределены по элементам сети. Сетевые системы обнаружения вторжений являются системами обнаружения атак.

Хостовые СОВ осуществляют анализ активности отдельного компьютера. В этом случае системе обнаружения вторжений предоставляется гораздо больший набор параметров для анализа. Кроме того, данные системы обнаружения вторжений могут легко реагировать на обнаруженную атаку. Хостовые СОВ подуровней осуществляют контроль событий, связанных с соответствующим приложением, поэтому хостовые СОВ являются системами обнаружения вторжений. Хостовые СОВ могут проводить и анализ пакетов, прибывающих на данный хост. Хостовые СОВ, использующие и анализ сетевых пакетов, приходящих на данный хост, являются гибридными, использующими обе технологии одновременно.

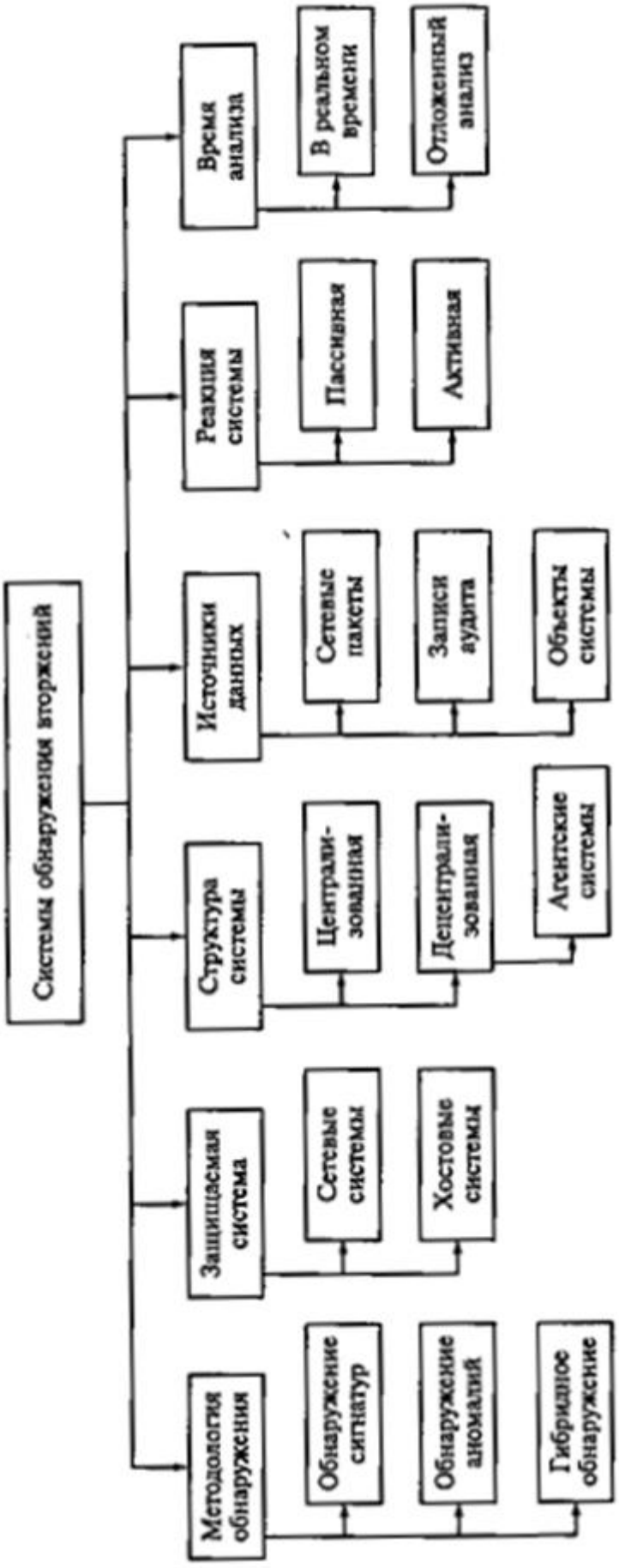


Рисунок 7.5. Классификация систем обнаружения вторжений.

Структура. По структуре СОВ подразделяются на централизованные и децентрализованные. Многие из существующих СОВ, как хостовые, так и сетевые, имеют монолитную архитектуру, в которой реализуются все операции системы: сбор данных, их анализ и принятие решения, включая генерацию сигнала тревоги. Сложность современных атак, затрагивающих различные аспекты безопасности сети организации, ее сетевых устройств и хостов, а также относительная дороговизна отдельных систем обнаружения вторжений привели к необходимости создания распределенных систем обнаружения. В последние годы появилось значительное число исследований и разработок децентрализованных систем обнаружения вторжений на основе технологии мобильных агентов.

Источник данных. Основными источниками данных для СОВ являются сетевые пакеты и данные аудита. Под записями аудита понимаются не только специально организованные записи собственно аудита, но и системные журналы, которые могут вестись, например, операционной системой. Такие системные журналы могут включать в себя конфигурационные файлы системы, данные для авторизации пользователей и т.д. Эта информация создает основу для последующего принятия решения.

Сенсор интегрируется с компонентом, ответственным за сбор данных – генератором событий (e-box). Поэтому способ сбора информации определяется политикой генерации событий, которая определяет режимы фильтрации просматриваемых событий.

Генератор событий (ОС, сеть, приложение) генерирует множество событий, зависящих от политики, которые могут записываться в журналы (системные и аудита) или определяться сетевыми пакетами. Данные в зависимости от политики могут сохраняться как внутри защищаемой системы, так и вне ее. В отдельных случаях данные могут не сохраняться, но тогда потоки данных о событиях передаются прямо в анализатор. Это касается, в частности, сетевых пакетов.

Существуют проблемы, связанные с обработкой следов аудита. Сохранение отчетов аудита в единственном файле может быть небезопасно, так как нарушитель может использовать его в своих интересах. Лучше иметь определенное число копий, распределенных по сети, но это увеличивает нагрузку на сеть и систему. С функциональной точки зрения регистрация каждого события означает значительное увеличение нагрузки на систему. Сжатие журналов регистрации также увеличивает системную нагрузку, высвобождая место на дисках. Определение того, какие события необходимо включать в аудит, достаточно затруднительно, так как определенные типы атак (вторжений) могут остаться необнаруженными. Трудно предсказать возможный объем файлов аудита и определить периоды их сохранения. Системы обработки журналов регистрации уязвимы к атакам отказа в обслуживании.

Время анализа. СОВ могут функционировать непрерывно или периодически получая информацию для обработки (real time, interval-based). Это разделение определяет применение двух различных подходов к обнаружению, которые мы будем называть режимом реального времени и отложенным режимом.

При обработке в реальном масштабе времени осуществляется текущая верификация системных событий для хостовых систем и анализ сетевых пакетов для сетевых СОВ. Из-за вычислительной сложности в сетевых СОВ используемые алгоритмы ограничиваются скоростью выполнения и своей эффективностью, поэтому часто используются наиболее простые алгоритмы.

Системы обнаружения атак должны функционировать только в реальном масштабе времени, а системы обнаружения вторжений могут функционировать как в реальном, так и в отложенном режимах.

Характер реакции. По характеру реакции различают пассивные (генерирующие только сигналы тревоги) и активные (выполняющие обнаружение и реализацию реакции на атаки) СОВ. В качестве реакции могут использоваться, например, попытки установить соответствующие «заплатки» в ПО перед взломом или блокировки атакуемых служб. В течение нескольких лет на страницах Интернета даже обсуждалась проблема активных атакующих действий против нарушителя. Пассивные СОВ могут выдавать сигнал тревоги на консоль администратора, выдавать сообщение на пейджер, посылать сообщение по электронной почте и даже выполнять звонок на заданный сотовый телефон.

VIII. Системы тунелирования

1. Протокол PPPoE

1.1. Обзор протокола PPPoE

Point-to-Point Protocol over Ethernet (PPPoE) – способ установления сеанса и инкапсуляции PPP-пакетов при передаче их по Ethernet. Применяется как протокол туннелирования, используемый для подключения нескольких пользователей в одной Ethernet-подсети к Интернету через общий последовательный интерфейс, например, DSL-линия, беспроводной канал, или кабельный модем. Все пользователи сети делят одно подключение, однако контроль доступа может быть выполнен для каждого пользователя.
PPPoE даёт провайдеру следующие преимущества:

- ✓ аутентификация по логину/паролю;
- ✓ автоматическое выделение IP-адресов пользователям (похоже на DHCP);
- ✓ автоматическое определение PPPoE-сервера.

Так как физически устанавливается PPP-сеанс, то PPPoE-соединение обладает всеми свойствами PPP-соединения.

Процесс установления PPPoE-соединения состоит из 2 этапов:

- ✓ поиск сервера (Discovery);
- ✓ установление PPP-соединения.

На этапе поиска клиент производит поиск PPPoE-сервера. Если в сети используется несколько серверов, то происходит выбор нужного. Так как PPPoE-соединение организуется на канальном уровне, то для его установления участникам достаточно знать MAC-адреса друг друга. Этап поиска происходит в 4 шага:

1. Клиент передаёт инициирующее сообщение (Initiation) по широковещательному адресу канального уровня (FF:FF:FF:FF:FF:FF).
2. Один или несколько серверов передают сообщение-предложение (Offer) на аренду MAC-адрес клиента.
3. Клиент передаёт сообщение о запросе сеанса (Session Request) по MAC-адресу выбранного сервера.
4. Выбранный сервер передаёт сообщение о подтверждении (Confirmation) установлении сеанса.

Существует 5 видов PPPoE-сообщений:

1. Active Discovery Initiation (PADI) – начало поиска клиентом сервера.
2. Active Discovery Offer (PADO) – оповещение клиента о наличии сервера.
3. Active Discovery Request (PADR) – запрос клиента на установление соединения.
4. Active Discovery Session-confirmation (PADS) – подтверждение установления сеанса связи сервером.
5. Active Discovery Terminate (PADT) – завершение соединения.

На этом шаге заканчивается первый этап. Второй этап представляет из себя PPP-сеанс. Отличие PPPoE от PPP заключается в том, что блоки данных передаются не по выделенной линии, а по сети Ethernet. Заголовок PPPoE занимает 8 байт, что необходимо учитывать при расчёте MTU (1500-8=1492).

1.2. Обзор протокола PPP

Point-to-Point Protocol (PPP) — протокол для связи между двумя компьютерами, например, для соединения абонентской станции с провайдером через коммутируемое соединение.

Функциями PPP являются:

1. Управление адресами протокола сетевого уровня (IP).
2. Асинхронное и синхронное формирование пакета данных.
3. Мультиплексирование протоколов сетевого уровня.
4. Конфигурирование канала связи.
5. Проверка качества канала связи.
6. Обнаружение ошибок при передаче данных.
7. Согласование параметров сжатия информации.

Типичный сеанс PPP-соединения состоит из нескольких процедур:

1. Установление соединения. Система, инициирующая соединение, использует Link Control Protocol (LCP) для согласования параметров связи, которые бы устроили обе стороны.
2. Аутентификация. Необязательная процедура. Система может вовлекать в процесс соединения протокол аутентификации, например, Password Authentification Protocol (PAP) или другой для проведения диалога открытия доступа к системе.
3. Установление соединения на уровне протоколов Сетевого уровня. Для каждого протокола Сетевого уровня, используемого системами во время сеанса связи, выполняется отдельная процедура установления соединения с использованием Network Control Protocol (NCP).

Рисунок 8.1. Формат PPP-кадра.

Формат PPP-кадра представлен на рисунке 8.1. Функции полей кадра перечислены ниже:

- ✓ Флаг (1 байт). Содержит шестнадцатеричное число 7e и работает в качестве разделителя пакетов, аналогично символу END протокола SLIP.
- ✓ Адрес (1 байт). Представляет собой шестнадцатеричное число ff, указывая, что пакет адресован всем станциям.
- ✓ Поле управления (1 байт). Значение в этом поле идентифицирует пакет как содержащий ненумерованное информационное сообщение HDLC (High-level Data Link Control).
- ✓ Протокол (2 байта). Предназначено для идентификации протокола, сформировавшего информацию в поле данных:
  - 0021. Не сжатые IP-дейтаграммы.
  - 002b. Дейтаграмма протокола Novell IPX.
  - 002d. IP-дейтаграммы со сжатыми TCP- и IP-заголовками.
  - c021. Link Control Protocol (LCP).
  - c023. Password Authentification Protocol (PAP).
  - c223. Challenge Handshake Authentification Protocol (CHAP).
- ✓ Данные (до 1500 байтов).
- ✓ Контрольная последовательность кадра (2 или 4 байта). Хранит контрольную сумму, вычисленную для целого кадра, за исключением полей флага и контрольной последовательности, в целях обнаружения возможных ошибок.

Протокол LCP

Данный протокол применяет в PPP-соединениях для ведения переговоров о своих возможностях во время процесса установления соединения с целью достижения наиболее эффективного соединения их всех возможных. Сообщения протокола LCP переносятся кадрами протокола PPP и содержат информацию о возможной конфигурации соединения. Как только две системы останавливаются на конфигурации, которую они обе могут поддерживать, процесс установления соединения продолжается.

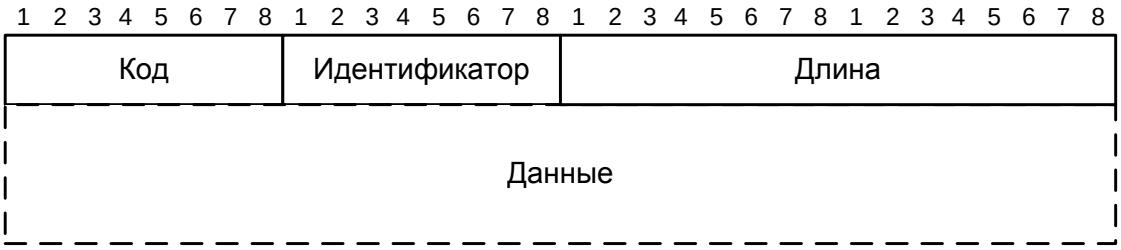


Рисунок 8.2. Формат LCP-пакета.

Функции полей LCP-пакета следующие:

- ✓ Код (1 байт). Определяет тип LCP-сообщения. Применяются следующие коды:
  - 1 – Configure-Request.
  - 2 – Configure-Ack.
  - 3 – Configure-Nak.
  - 4 – Configure-Reject.
  - 5 – Terminate-Request.
  - 6 – Terminate-Ack.
  - 7 – Code Reject.
  - 8 – Protocol Reject.
  - 9 – Echo-Request.
  - 10 – Echo-Reply.
  - 11 – Discard-Request.
- ✓ Идентификатор (1 байт). Содержит код, используемый при ассоциации запросов и ответов на них для индивидуальной LCP-транзакции.
- ✓ Длина (2 байта). Определяет длину LCP-сообщения, включая все поля.
- ✓ Данные (переменный размер). Содержит элементы, состоящие из трех полей: тип, длина и данные:
  - Тип (1 байт). Определяет конфигурируемую опцию, например:
    - 1 – максимально принимаемый блок.
    - 3 – протокол аутентификации.
    - 14 – время соединения.
  - Длина (1 байт). Определяет длину элемента, включая все поля.
  - Данные (переменный размер).

Протокол аутентификации PAP

Самый простой и слабый протокол аутентификации: использует двухстороннее квитирование установление связи и передает логины и пароли открытым текстом. Формат кадра протокола PAP аналогичен протоколу LCP.

- ✓ Код (1 байт):
  - 1 – запрос аутентификации
  - 2 – подтверждение аутентификации

- 3 – неподтверждение аутентификации

✓ Идентификатор (1 байт). Содержит код для ассоциации запросов и ответов на них.
✓ Длина (2 байта).
✓ Данные (переменный размер). Содержит несколько подполей в зависимости от значения в поле кода:
- Длина поля идентификатора узла (1 байт).
- Идентификатор узла (переменный размер).
- Длина пароля (1 байт).
- Пароль (переменный размер).
- Длина сообщения (1 байт).
- Сообщение (переменный размер).

Процесс установления соединения по протоколу PPP
Представлен на рисунке 8.3.

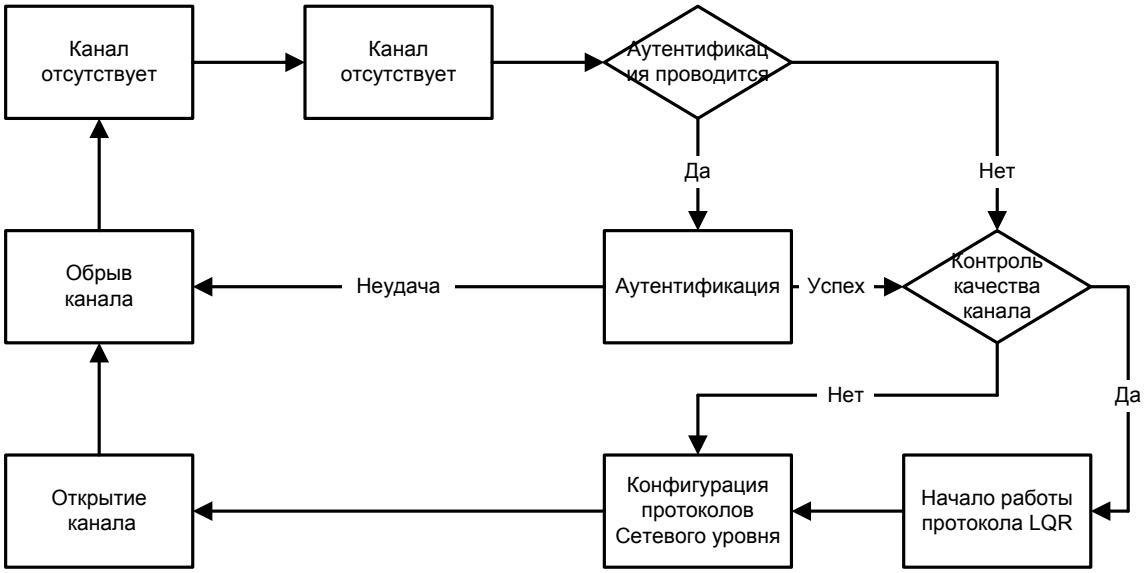


Рисунок 8.3. Процесс установления соединения по протоколу PPP.

Канал отсутствует
Обе системы начинают и заканчивают сеанс связи с фазы под названием «Канал отсутствует» (link dead), которая предполагает, что между данными системами не существует связь на Физическом уровне. Во время типичной сессии приложение на одной из сторон инициирует соединение на Физическом уровне путем набора номера по модему или иными способами. Как только установлена связь на Физическом уровне, системы переходят к стадии установления канала.

Установление канала
На данной стадии в работу вступает PPP-протокол. В фазе «установление канала» система, инициировавшая соединение, передает LCP-сообщение Configure-Request в систему назначения. Это сообщение содержит конфигурационные опции, которые эта система хотела бы использовать. Если система, получившая сообщение, может поддерживать все указанные в нем возможности, она отправляет сообщение Configure-Ack, которое содержит те же значения конфигурационных опций, и данная фаза заканчивается.
Когда система-получатель распознает запрашиваемые элементы конфигурации, но не может их обеспечить, она посылает обратно сообщение Configure-Nak, где указывает значение тех опций, которая система-получатель не поддерживает. Вместе с ними вклю-

чается значения всех опций из предложенных, которые могут поддерживаться, а также те варианты, которые она сама хотела бы использовать. С получением этого сообщения первая система генерирует сообщение Configure-Request, где отражает все полученные опции и в ответ получает сообщение Configure-Ack.
Но если системе получателю не удается распознать каких-либо опций, то она отвечает сообщением Configure-Reject.

Аутентификация
Фаза «Аутентификация» является необязательной и запускается включением опции Authentification Protocol в LCP-сообщении Configure-Request. В состоянии организации канала по протоколу LCP, пройденном ранее, две системы договариваются о протоколе аутентификации, который будет востребован в рассматриваемой фазе. Наиболее часто применяются протоколы PAP и CHAP из набора TCP/IP.
Контроль качества канала
Использование протокола контроля качества канала является также необязательным элементом в процессе соединения, который активируется включением опции Quality Protocol в LCP-сообщение Configure-Request. Хотя опция и позволяет системе-отправителю определить для этой цели любой протокол, но только один из стандартизован, а именно – Link Quality Protocol. Процесс переговоров позволяет системам прийти к соглашению об интервалах времени, через которые будут передаваться сообщения, содержащие трафик канала и статистику ошибок за время текущей сессии.

Конфигурация протоколов Сетевого уровня
Для всех протоколов Сетевого уровня, которые будут использоваться в PPP-соединении, требуются индивидуальные NCP-переговоры. Структура обмена NCP-сообщениями сходна со структурой обмена сообщениями протокола LCP, за исключением того, что опциональные элементы, включаемые в сообщение Configure-Request, отвечают требованиям конкретного Сетевого протокола. Например, в ходе конфигурации протокола IPCP, системы информируют друг друга о своих IP-адресах и договариваются о том, применять или нет сжатие заголовков пакета. NCP-процедуры инициализации или прекращения работы могут также происходить в любой момент в течение всего времени соединения.

Процесс разъединения соединения по протоколу PPP
Когда пользователь завершает сессию, или при неудачной аутентификации, или истечении времени простоя канала, системы вступают в фазу «Обрыв канала». Для этого одна из систем посылает LCP-сообщение Terminate-Request, на которое другая отвечает сообщением Terminate-Ack.

2. Виртуальные частные сети (Virtual Private Network, VPN)

2.1. Введение

Виртуальные частные сети, или защищенные виртуальные сети (Virtual Private Network, VPN), – это подключение, установленное по существующей общедоступной инфраструктуре и использующее шифрование и аутентификацию для обеспечения безопасности содержания передаваемых пакетов. Виртуальная частная сеть создает виртуальный сегмент между любыми двумя точками доступа к сети. Она может проходить через общедоступную инфраструктуру локальной вычислительной сети, подключения к глобальной сети (Wide area Network, WAN) или Интернет.

Рассмотрим VPN, организующие безопасные каналы передачи данных, использующие общедоступную инфраструктуру или Интернет. Все VPN по конфигурации можно подразделить на три основных типа:

1. узел-узел (host-to-host);
2. узел-шлюз (host-to-gateway);
3. шлюз-шлюз (gateway-to-gateway).

Для организации канала связи, проходящего через Интернет, можно использовать VPN любого типа. Основной концепцией VPN является защита шифрованием канала связи на различных уровнях модели TCP/IP, а именно:

- ✓ прикладном (5-й уровень);
- ✓ транспортном (4-й уровень);
- ✓ сетевом (3-й уровень);
- ✓ канальном (2-й уровень).

Схема расположения протоколов VPN по уровням модели приведена в таблице 7.1.
На прикладном уровне шифрование можно применять при помощи программ, подобных пакету Pretty Good Privacy (PGP), или через каналы типа Secure Shell (SSH). Такие программы работают на участке сети от узла до узла, что означает, что они предлагают защиту только душ содержимого (payload) пакета, а не всего пакета в целом. Исключение составляет протокол SSH, который может использовать режим port-forwarding для создания туннеля.

Таблица 7.1 – Схема расположения протоколов VPN по уровням модели.

Уровни TCP/IP	Основные протоколы
Прикладной	PGP, S/MIME SSH, Kerberos, RADIUS
Транспортный	SSL, TSL SOCKS v5
Уровень межсетевого взаимодействия (сетевой)	IPSec (AH, ESP)
Уровень сетевых интерфейсов (канальный)	L2TP, PPTP

На транспортном уровне для защиты содержимого пакетов конкретного сеанса между двумя сторонами можно использовать протоколы, аналогичные протоколу защищенных сокетов (Secure Sockets Layer, SSL). Обычно такой метод используется при соединениях, установленных посредством Web-браузера. При этом также защищается только содержательная часть передаваемых пакетов, а IP-датаграммы, которые несут эту информацию, доступны для просмотра.

На сетевом уровне протоколы, подобные IPSec, не только зашифровывают содержательную часть пакета (полезную нагрузку), но и зашифровывают информацию заголовков протоколов TCP/IP.

На канальном уровне протокол туннелирования (Layer 2 Tunneling Protocol, L2TP) является расширением протокола соединения типа «точка-точка» (Point-to-Point Protocol, PPP), который допускает шифрование пакетов, посланных по PPP-протоколу на канальном уровне передачи данных.

Несмотря на то, что эти технологии шифрования применяются на разных уровнях, они все могут быть частью VPN. Необходимо отметить, что некоторые из этих технологий не могут обрабатывать все режимы работы VPN без дополнительной помощи со стороны других приложений или протоколов.

2.2. Обзор протоколов PPTP и L2TP

На канальном уровне существуют два протокола для реализации VPN: протокол туннелирования типа «точка-точка» (Point-to-point Tunneling Protocol, PPTP) и протокол туннелирования второго уровня (Layer Two Tunneling Protocol, L2TP).

Протокол PPTP

Протокол PPTP является дальнейшим развитием протокола PPP, который распространился в связи с появлением модемного доступа к сети Интернет. Протокол PPTP был одним из первых протоколов для создания VPN через коммутируемое соединение и был разработан консорциумом таких производителей, как Microsoft, US Robotics, Ascend и 3Com. Для шифрования протокола PPP был использован протокол двухточечного шифрования Microsoft (Microsoft Point-to-Point Encryption, MPPE), который использует алгоритм RC4. Однако большинство проблем в области безопасности было связано с ненадежностью используемого метода аутентификации — протокола Microsoft аутентификации с предварительным согласованием вызова (Microsoft Challenge/Reply HandShake Protocol, MSCHAP). Для устранения недостатков был выпущен протокол MSCHAP версии 2. Протокол PPTP использует все связанные протоколы, которые подобны протоколу MSCHAP, протоколу аутентификации пароля (Password Authentication Protocol, PAP), протоколу аутентификации с предварительным согласованием вызова (Challenge Handshake Authentication Protocol, CHAP), расширенному протоколу аутентификации (Extensible Authentication Protocol, EAP).

Протокол PPTP использует два канала, работающих совместно. Первый — канал управления (порт 1723/tcp). Этот канал посылает в обе стороны все команды, которые управляют сеансом подключения. Второй — инкапсулированный канал передачи данных, являющийся вариантом протокола общей инкапсуляции для маршрутизации (Generic Routing Encapsulation, GRE, IP-протокол 47). Преимуществом туннеля протокола общей инкапсуляции для маршрутизации является то, что он может инкапсулировать и передавать протоколы, отличающиеся от протокола IP. Первоначально клиент устанавливает соединение с провайдером с помощью PPP, а затем устанавливает TCP/IP-соединение с PPTP-сервером через интернет.

В отличие от L2TP PPTP не требует развёртывания инфраструктуры для сертификатов, а использует парольную аутентификацию, что упрощает развёртывание VPN, но в некоторых случаях снижает безопасность. PPTP-сервер имеет ограничение по количеству одновременных подключений. PPTP не использует IPSec, поэтому проблем с трансляцией адресов не возникает (не требуется механизм NAT Traversal).

Протокол PPTP реализован во многих аппаратных устройствах. Протокол PPTP при инициализации связи использует протокол PPP, по этому может оказаться уязвимым к атакам типа spoofing и «человек посередине».

Протокол L2TP

Протокол L2TP (Layer 2 Tunneling Protocol) – открытый стандарт IETF, который решает многие проблемы PPTP, и определен в документе RFC 2661. Протокол L2TP фактически является гибридом двух предыдущих протоколов туннелирования: протокола пересылки второго уровня (Layer Two Forwarding, L2F) компании Cisco и протокола PPTP.

Протокол L2TP, подобно протоколу PPTP, использует при аутентификации пользователя возможности протокола PPP (протоколы MSCHAP, CHAP, EAP, PAP и т.д.). Аналогично протоколу PPTP протокол L2TP использует два канала связи: сообщения управления и сообщения туннеля для передачи данных. Первый бит заголовка протокола PPTP служит для опознания этих типов сообщений (1 — для сообщений управления, 0 — для сообщений данных). Сообщениям управления дается более высокий приоритет по отношению к сообщениям данных, чтобы гарантировать, что важная информация администрирования сеанса будет передана настолько быстро, насколько это возможно.

Подключение канала управления устанавливается для туннеля, который затем сопровождается инициированием сеанса протокола L2TP. После завершения инициирования обоих подключений информация в виде кадров протокола PPP начинает передаваться по туннелю.

Формирование защищенного канала происходит в три этапа:

1. Установление соединения клиента с сервером удаленного доступа.
2. Аутентификация пользователя.
3. конфигурирование защищенного туннеля.

Для установления соединения с сервером удаленного доступа (сетевой сервер L2TP) удаленный пользователь связывается по протоколу PPP с концентратором доступа L2TP (Local Access Concentrator, LAC), обычно функционирующем на сервере провайдера. Концентратор доступа может выполнить аутентификацию пользователя от имени провайдера. По заданному имени получателя концентратор доступа определяет адрес сетевого сервера L2TP (L2TP Network Server, LNS), который защищает сеть с заданным адресом. Между концентратором доступа и сервером L2TP устанавливается соединение. Далее производится аутентификация пользователя сервером L2TP. В случае успешной аутентификации устанавливается защищенный туннель между концентратором доступа и сервером L2TP. С помощью управляющих сообщений производится настройка параметров туннеля, причем в одном туннеле может быть несколько сеансов пользователя. При использовании IPSec пакеты L2TP инкапсулируются в UDP-па-кеты, которые передаются концентратором доступа и сервером L2TP через IPSec-туннель (порт 1701/tcp). В большинстве случаев клиент сам выступает в роли LAC.

L2TP использует сертификаты для аутентификации, за счёт чего упрощается администрирование сервера с большим количеством клиентов и повышается безопасность. В отличие от PPTP имеется возможность создания нескольких виртуальных сетей через один туннель. В случае, когда L2TP использует IPSec, при необходимости трансляции адресов требуется NAT Traversal на LNS.

Сравнение PPTP/L2TP

1. В PPTP шифрование данных начинается после установления PPP-соединения. В L2TP/IPSec шифрование начинается до установления такого соединения.
2. В PPTP для шифрования применяется протокол MPPE, основанный на алгоритме RSA RC4, который обеспечивает возможность применения 40-, 56- и 128-битных ключей. В L2TP/IPSec возможно применение любых алгоритмов шифрования, поддерживаемых IPSec.
3. В PPTP используется аутентификация PPP (сервер аутентифицирует клиента по паре логин-пароль). В L2TP/IPSec наряду с учётной парой применяется и аутентификация по сертификатам.

2.3. Протокол IPSEC

Несмотря на то, что протокол IP стал наиболее используемым протоколом связи во всем мире и базовой технологией Интернета, он имеет множество существенных недостатков. Среди них необходимо отметить ограниченность адресного пространства и недостатки внутренней безопасности. Главной причиной этих недостатков является то, что IP-протокол изначально не был предназначен для массового использования.

В качестве развития IP-протокола была разработана его новая версия IPv6 (IP-протокол версии 6), в которой пытались решить проблемы предшествующих версий. Поскольку принятие новой версии IP-протокола затруднительно, что вызвано постоянным ростом Интернета и громадным разнообразием установленной аппаратуры и программного обеспечения, то меры безопасности, включенные в IPv6, были перенесены в текущую версию IPv4 в виде дополнительного комплекта протоколов. Этот набор протоколов назвали комплектом протоколов IPSec (IPSec Protocol Suite).

Подключение по протоколу IPSec имеет два основных режима: транспортный (transport) и туннельный (tunnel).Транспортный режим — это форма связи типа узел-узел, где применяется шифрование только содержательной части пакета. Из-за двухточечного характера связи соответствующее программное обеспечение необходимо загрузить (установить) на все связывающиеся между собой узлы сети, что представляет собой достаточно серьезную проблему. Этот режим VPN удобно использовать для зашифрованной связи между узлами одной сети.

Режим туннелирования применяется при создании большинства VPN; потому что он шифрует весь оригинальный пакет. Режим туннелирования может применяться для организации связи типа узел-узел, узел-шлюз или шлюз-шлюз. При организации связи типа шлюз-шлюз значительно упрощается связь между сетями, не требуется установка специального ПО на узлах сети.

Первая цель семейства протоколов IPSec состоит в обеспечении конфиденциальности, целостности и аутентификации информации, передаваемой посредством IP-протокола. Это достигается с помощью протокола обмена интернет-ключами (Internet Key Exchange, IKE), протокола ESP и протокола AH. Комбинация этих трех протоколов обеспечивает безопасный обмен информацией.

Второй целью семейства протоколов IPSec является предоставление разработчикам ПО набора стандартов. Установление безопасного соединения начинается с формирования ассоциации обеспечения безопасности (Security Association, SA) между двумя сообщающимися сторонами.

Ассоциация обеспечения безопасности

Основа ассоциации обеспечения безопасности заключается в соглашении двух сторон о том, как они могут безопасно передавать свою информацию. В процессе соглашения стороны оговаривают детали защищенного обмена. Результатом такого соглашения и является ассоциация обеспечения безопасности. Каждому сеансу связи сопоставляются две ассоциации — по одной на каждого партнера связи.

Положительными чертами протокола IPSec являются открытость его стандарта для поддержки множества протоколов и режимов связи, а также поддержка различных алгоритмов шифрования и различных хэш-функций. Прежде чем договариваться об ассоциации обеспечения безопасности, необходимо локальное конфигурирование элементов протокола IPSec, которые данный партнер собирается поддерживать. Эти параметры настройки хранятся в базе данных политики безопасности (Security Policy Database, SPD).

После согласования ассоциация обеспечения безопасности содержится в базе данных ассоциации обеспечения безопасности (Security Association Database, SAD). Это необходимо, поскольку узел сети может инициализировать несколько сеансов, каждому из которых может соответствовать своя SA.

Поскольку для каждого сетевого устройства доступно множество сеансов протокола IPSec, для правильного функционирования процесса необходимо, чтобы каждый сеанс согласования SA имел свой собственный уникальный идентификатор. Этот идентификатор составляется из уникального индекса параметра обеспечения безопасности (Security Parameter Index, SPI), который определяет, какая запись БД SA соответствует рассматриваемому подключению. Кроме того, учитывается адрес назначения и идентификатор используемого протокола (ESP или AH).

Протокол обмена интернет-ключами

Протокол обмена интернет-ключами предназначен для аутентификации и согласования параметров обмена протокола IPSec. Протокол IKE представляет собой комбинацию двух протоколов: протокола управления ассоциациями и протокола управления ключами обеспечения безопасности в сети Интернет (Internet Security Association and Key Management Protocol, ISAKMP), называемых фазами установления. Управление ключами можно выполнять вручную или используя альтернативы протокола IKE, такие как безопасная служба доменных имен (Secure DNS), Photuris или простой протокол обмена интернет-ключами (Simple Key Internet Protocol, SKIP).

Первая фаза протокола IKE. На первой фазе протокола IKE удаленный пользователь начинает сеанс со шлюзовым устройством VPN. Первая фаза выполняет две функции: аутентификацию удаленного пользователя и обмен информацией об открытых ключах, которые будут использоваться во второй фазе.

Аутентификацию можно выполнить несколькими различными способами. Наиболее часто используются технология предварительно распространяемых ключей (pre-shared keys) и технология цифровых сертификатов. Термин «предварительно распространяемые ключи» означает, что значения ключей предварительно задаются на всех компьютерах, которые собираются устанавливать соединения через VPN (что является существенным недостатком). При втором способе используются цифровые удостоверения (цифровые сертификаты, digital certificates), которые могут назначаться отдельно для каждого объекта, который соединяется с VPN. Цифровыми сертификатами можно удаленно управлять и администрировать из уполномоченного центра сертификации (Certificate Authority, CA).

Сертификационная служба является центральным элементом структуры, называемой инфраструктурой с открытым ключом (Public Key Infrastructure, PKI). За инфраструктурой PKI стоит концепция публично доступной структуры, распределяющей информацию об открытых (публичных) ключах.

В первой фазе при обмене аутентификационной информацией и параметрами безопасности могут использоваться два режима: основной (main mode) и агрессивный (aggressive mode). Различия между ними заключаются в количестве сетевых пакетов, которыми обмениваются стороны, и во времени, за которое генерируется открытый ключ. Агрессивный режим использует дополнительный заголовок меньшего размера, но основной режим обладает большей безопасностью и используется чаще всего.

Вторая фаза протокола IKE. Во второй фазе протокола IKE согласовываются конкретные параметры ассоциации обеспечения безопасности IPSec. Данное согласование подобно агрессивному режиму обмена информацией первой фазы. После завершения второй фазы формируется SA и пользователь получает подключение к VPN.

Во второй фазе возможен единственный режим согласования – быстрый режим (quick mode). Быстрый режим представляет собой короткий обмен, использующий три пакета. Все обмены второй фазы зашифрованы с помощью согласованных во время первой фазы протоколов и типов кодирования. При этом используется только защита, основанная на использовании хэш-функции и нонсе (nonce), включаемых в сетевые пакеты для подтверждения их оригинальности (нонсе является подтверждением того факта, что информация о ключе исходит из ожидаемого источника. Нонсе – это случайное число, генерируемое инициатором связи, которое заверяется респондентом цифровой подписью и посылается обратно).

Данная реализация включает в себя также идентификатор поставщика (vendor ID, VID), позволяющий участникам межплатформенных взаимодействий делать предположения о возможностях и конфигурации их партнеров, которые могут иметь различных изготовителей. После создания ассоциации обеспечения безопасности, используемой протоколом IKE, можно применять протоколы обеспечения безопасности. При построении VPN, основанной на протоколе IPSec, можно выбрать использование одного из протоколов (AH или ESP) или использовать их одновременно.

Управление ключами. Применяемый по умолчанию для IPSec протокол автоматизированного управления ключами называется ISAKMP/Oakley и состоит из следующих элементов:

- ✓ протокол определения ключей Oakley – протокол на основе алгоритма Диффи-Хеллмана, но обеспечивающий дополнительную защиту. Протокол Oakley называется общим, так как он не диктует использования каких-либо конкретных форматов;
- ✓ протокол защищенных связей и управления ключами в Интернете – обеспечивает основу схемы управления ключами и поддержку специального протокола и необходимых форматов процедуры согласования атрибутов защиты.

ISAKMP не заставляет использовать какой-то конкретный алгоритм обмена ключами, а предлагает использовать любой подобный алгоритм.

Протокол Oakley разработан в целях сохранения преимуществ алгоритма Диффи-Хеллмана и устранения его недостатков. Алгоритм Oakley характеризуется следующими особенностями:

- ✓ использование механизмов рецептов (cookies) для защиты от атак засорения;
- ✓ соглашение двух сторон о группе, которая определяет параметры алгоритма обмена ключами Диффи-Хеллмана;
- ✓ использование оказий для противостояния атакам повтора сообщений;
- ✓ обмен открытыми ключами Диффи-Хеллмана;
- ✓ аутентификация обмена для противостояния атакам «человек посередине».

Протокол аутентификации заголовка

Протокол AH – это IP-протокол 51. Он поддерживает функциональные возможности аутентификации и проверки целостности, но не поддерживает конфиденциальность содержимого пакета.

Обеспечение аутентификации и защиты целостности достигается добавлением дополнительного заголовка к IP-пакету. Этот заголовок содержит цифровую подпись, называемую значением проверки целостности (Integrity Check Value, ICV), которая является в основном значением хэш-функции, подтверждающей, что пакет не был изменен во время транспорта.

Информация IP-протокола, содержащаяся в пакете, гарантирует правильность содержимого пакета, но она передается в открытом виде. Поскольку протокол AH просматривает заголовок IP-пакета при вычислении цифровой подписи, можно убедиться, что IP-адрес отправителя подлинный и что пакет исходит от того, кого требуется.

Протокол AH также поддерживает использование порядковых номеров (sequence numbers), помогающих предотвращать напад ния, основанные на повторном использовании пакета (replay attack). Эти номера используются коммуникационными устройствами для отслеживания потока сетевых пакетов сеанса связи.

Следующий заголовок	Длина содержимого пакета	Зарезервировано
Индекс параметра обеспечения безопасности (SPI)		
Порядковый номер		
Информация аутентификации (переменная длина, кратная 32 байтам)		

Рисунок 8.4. Структура заголовка AH-пакета.

Использование информации IP-заголовка протоколом AH делает его несовместимым с использованием NAT. Структура заголовка AH-пакета представлена на рисунке 8.4.

Поле следующего заголовка содержит идентификатор, определяющий тип заголовка пакета, следующего за заголовком AH-пакета. Поле длины содержимого пакета определяет длину заголовка AH-пакета. Индекс параметра обеспечения безопасности показывает, частью какого уникального потока связи ассоциации обеспечения безопасности является рассматриваемый пакет.

Рисунок 8.5. Схема преобразования исходного пакета в AH-пакет.

Порядковый номер – уникальное увеличивающееся значение, которое предназначено для противодействия повторному использованию пакета. Поле информации аутентификации содержит значение проверки целостности и цифровую подпись, подтверждающую подлинность рассматриваемого пакета. Схема преобразования исходного пакета в AH-пакет для различных режимов приведена на рисунке 8.5.

Протокол безопасной инкапсуляции содержимого пакета

Протокол ESP – это IP-протокол 50. Он обеспечивает конфиденциальность при помощи полного шифрования содержимого IP-пакетов. Протокол ESP реализован в виде модулей и может использовать любое количество доступных симметричных алгоритмов шифрования, в числе которых DES, 3DES, IDEA. Применение протокола ESP различается в зависимости от используемого режима протокола IPSec.

В транспортном режиме протокол ESP просто добавляет свой собственный заголовок после IP-заголовка и зашифровывает остальную часть сетевого пакета начиная с транспортного уровня. Если при этом определена служба аутентификации, то протокол ESP добавляет концевую метку (trailer). Концевая метка предназначена для подтверждения целостности пакета и аутентификации (в отличие от протокола AH значение проверки целостности вычисляется без использования информации из IP-заголовка).

При использовании туннельного режима протокол ESP инкапсулирует оригинальный пакет полностью, шифруя его целиком и создавая новый IP-заголовок и ESP-заголовок в устройстве туннелирования. Концевая метка также добавляется в случае выбора аутентификационного сервиса протокола ESP.

В любом режиме протокол ESP использует в каждом сетевом пакете порядковые номера. При работе протокола ESP в туннельном режиме можно использовать NAT. Структура заголовка пакета ESP приведена на рисунке 8.6.

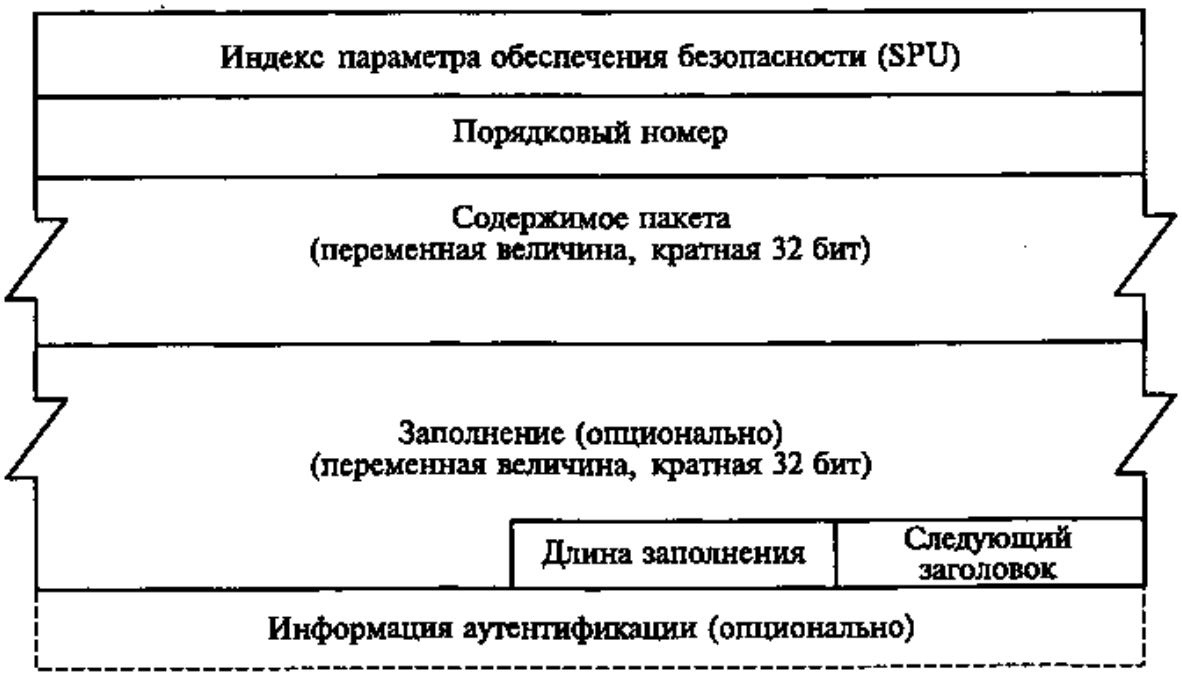


Рисунок 8.6. Структура заголовка пакета ESP.

Поле длины заполнения указывает, насколько заполнено содержимое пакета (если такое вообще имеет место), чтобы длина содержимого пакета и последующие поля заголовка соответствовали требованию выравнивания длины сетевого пакета.

Поле следующего заголовка сообщает номер протокола пакета, который инкапсулирован внутри пакета протокола ESP. Поле информации аутентификации содержит дополнительное значение проверки целостности, которое доступно для пакетов протокола ESP.

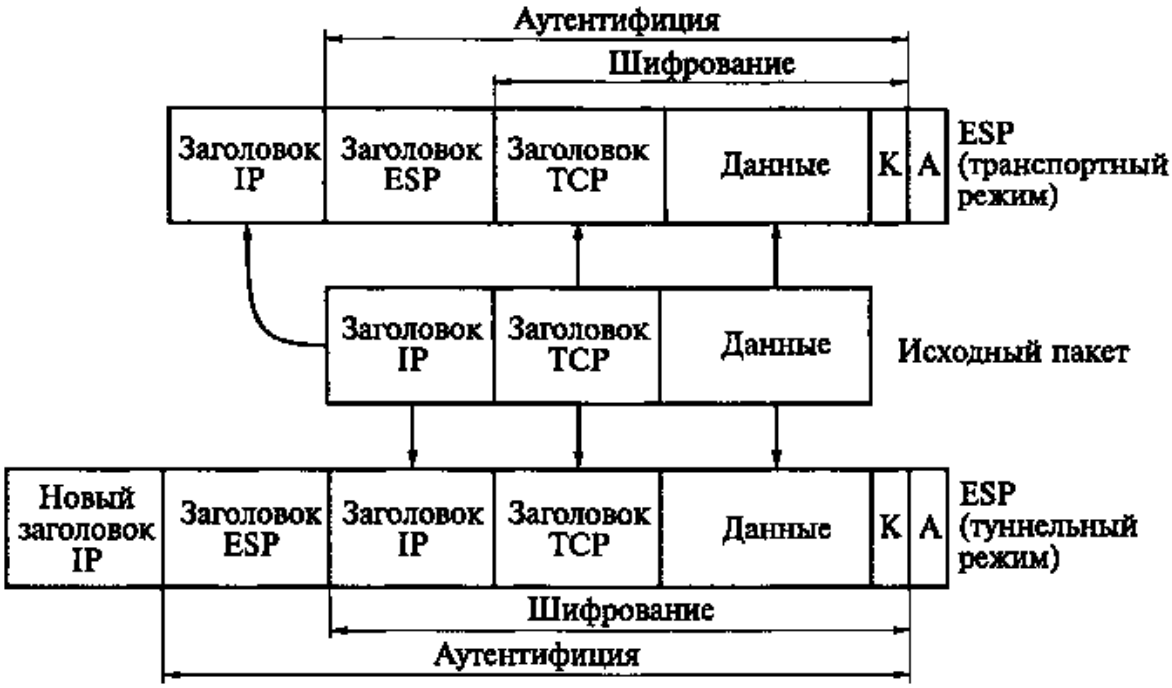


Рисунок 8.7. Схема преобразования исходного пакета в пакет ESP.

Схема преобразования исходного пакета в пакет ESP для различных режимов приведена на рисунке 8.7.

Совместное использование протоколов ESP и AH

Отдельная защищенная связь может использовать либо протокол AH, либо протокол ESP, но никак не оба эти протокола одновременно. Тем не менее, иногда конкретному потоку данных может требоваться и сервис AH, и сервис ESP. Во всех случаях одному потоку для получения комплекса услуг IPSec требуется несколько защищенных связей. Такой набор защищенных связей называется пучком защищенных связей (security association bundle), посредством которого потоку должен предоставляться необходимый набор услуг IPSec. При этом защищенные связи в пучке могут завершаться в различных конечных точках.

Защищенные связи могут быть объединены в пучки следующими двумя способами:

1. Транспортной смежности – в этом случае применяются несколько протоколов защиты к одному IP-пакету без создания туннеля. Эта комбинация AH и ESP эффективна только для одного уровня вложенности, так как обработка выполняется в одной точке — IPSec конечного получателя.
2. Повторного туннелирования – в этом случае применяются несколько уровней протоколов защиты с помощью туннелирования IP. Этот подход допускает множество уровней вложения, поскольку туннели могут начинаться и завершаться в разных использующих IPSec узлах сети вдоль маршрута передачи данных.

Кроме того, эти два подхода можно объединить. Тогда при рассмотрении пучков защищенных связей возникает вопрос – в каком порядке могут применяться аутентификация и шифрование между данной парой конечных узлов. Рассмотрим несколько подходов к такому комбинированию.

В случае применения ESP с опцией аутентификации пользователь сначала применяет ESP к требующим защиты данным, а затем добавляет поле данных аутентификации. В зависимости от используемых режимов возможны следующие варианты (в обоих вариантах аутентификации подлежит шифрованный текст):

- ✓ транспортный режим ESP – аутентификация и шифрование применяются к полезному грузу IP, доставляемому узлу адресата, но при этом заголовок IP не защищается;
- ✓ туннельный режим ESP – аутентификация применяется ко всему пакету IP, доставляемому по адресу IP внешнего получателя (например, МЭ), и выполняется этим получателем. Весь внутренний пакет IP защищается механизмом секретности, поскольку предназначен для доставки внутреннему адресату IP.

В случае транспортной смежности используется пучок из двух защищенных связей в транспортном режиме, где внутренняя связь является защищенной связью ESP, а внешняя — защищенной связью AH. В этом случае ESP используется без опции аутентификации. Поскольку внутренняя защищенная связь используется в транспортном режиме, шифрованию подлежит полезный груз IP. Получаемый в результате пакет состоит из заголовка IP, за которым следуют данные ESP. Затем применяется AH в транспортном режиме, так что аутентификация будет охватывать данные ESP и оригинальный заголовок IP, за исключением изменяемых полей. Достоинством данного подхода является то, что при комбинированном подходе аутентификация охватывает больше полей, включая поля адресов IP источника и назначения. Недостаток связан с использованием двух защищенных связей вместо одной.

Рассмотрим транспортно-туннельный режим. В некоторых случаях целесообразным является применение аутентификации до шифрования. Для этого используется пучок защищенных связей, состоящий из внутренней защищенной транспортной связи AH и

внешней защищенной туннельной связи ESP. В этом случае аутентификация охватывает полезный груз IP вместе с заголовком IP (и расширениями), за исключением изменяемых полей. Полученный таким образом пакет затем обрабатывается в туннельном режиме ESP, в результате чего внутренний пакет вместе с данными аутентификации оказывается зашифрованным и имеющим новый внешний заголовок IP (с необходимыми расширениями.

2.4. Протокол SSL/TLS

SSL (Secure Sockets Layer, уровень защищённых сокетов) – криптографический протокол, обеспечивающий безопасную передачу данных по сети Интернет. При его использовании создаётся защищённое соединение между клиентом и сервером. SSL изначально разработан компанией Netscape Communications. Впоследствии на основании протокола SSL 3.0 был разработан и принят стандарт RFC, получивший имя TLS. TLS является протоколом более низкого уровня, хотя на модели TCP/IP это не отражается (и SSL, и TLS – протоколы прикладного уровня).

Использует шифрование с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надёжность передачи данных за счёт использования корректирующих кодов и безопасных хэш-функций. Обычно применяются следующие алгоритмы:

1. Для обмена ключами: RSA, Diffie-Hellman, ECDH, SRP, PSK.
2. Для аутентификации: RSA, DSA, ECDSA.
3. Симметричные шифры: RC4, Triple DES, AES, IDEA, DES, или Camellia. В старых версиях SSL также использовался RC2.
4. Для криптографических хэшей: HMAC-MD5 или HMAC-SHA для TLS, MD5 и SHA для SSL.
5. В старых версиях SSL также использовались MD2 и MD4.

SSL состоит из двух уровней. На нижнем уровне многоуровневого транспортного протокола (например, TCP) он является протоколом записи и используется для инкапсуляции (то есть формирования пакета) различных протоколов (SSL работает совместно с таким протоколами как POP3, IMAP, XMPP, SMTP и HTTP). Для каждого инкапсулированного протокола он обеспечивает условия, при которых сервер и клиент могут подтверждать друг другу свою подлинность, выполнять алгоритмы шифрования и производить обмен криптографическими ключами, прежде чем протокол прикладной программы начнёт передавать и получать данные. Если некоторая программа не поддерживает SSL/TLS, можно воспользоваться программой для создания защищенных туннелей stunnel.

Для обеспечения безопасности применяются следующие процедуры:

1. Клиент может использовать публичный ключ центра сертификации для проверки подлинности цифровой подписи серверного сертификата.
2. Клиент проверяет, что центр сертификации, подписавший ключ, относится к списку доверенных центров сертификации.
3. Клиент проверяет срок действия серверного сертификата.
4. Имеется защита от понижения версии протокола и от слабого алгоритма шифрования
5. Сообщение, завершающее "рукопожатие" (согласование), содержит хэш всех сообщений о согласовании, полученных обоими сторонами.
6. Псевдослучайная функция делит входные данные пополам и обрабатывает каждую часть своим алгоритмом хэширования (MD5 и SHA-1), а затем складывает их по модулю 2 (XOR), чтобы создать MAC (message authentification codes, коды аутентифика-

ции сообщения). Это обеспечивает защиту даже в том случае, если в одном из алгоритмов обнаружат уязвимость (только для TLS)

7. SSLv3 отличается от SSLv2 добавлением SHA-1 и поддержкой аутентификации сертификата. Также была улучшена процедура согласования и усилена устойчивость к атаке «человек посередине».

Область применения протоколов SSL/TLS весьма широка:

- ✓ создание шифрованных туннелей;
- ✓ шифрование текстовых протоколов интернета HTTP, POP3, IMAP, SMTP, XMPP;
- ✓ аутентификация на интернет-сервисах или в локальной сети

Процедура установления шифрованной связи выглядит следующим образом:

1. Сервер и клиент, использующие TLS, устанавливают надёжное соединение с помощью процедуры согласования.
2. Согласование начинается, когда клиент подключается к использующему TLS серверу, запрашивая безопасное соединение, и представляет собой список поддерживаемых шифров и хэшей.
3. Из этого списка сервер выбирает наиболее мощные шифр и хэш и уведомляет клиента о решении.
4. Сервер отсылает свою идентификацию в виде цифрового сертификата. Сертификат обычно содержит имя сервера, доверенный центр сертификации и публичный ключ шифрования сервера.

Перед продолжением процедуры клиент может связаться с центром сертификации, выдавшим сертификат, чтобы убедиться, что сертификат действителен:

1. Чтобы сгенерировать сеансовый ключ, используемый для защиты соединения, клиент шифрует случайное число с помощью публичного ключа сервера и посылает его серверу. Только сервер может расшифровать его (с помощью своего закрытого ключа). Это позволяет сделать ключ недосягаемым для третьих лиц.
2. Из этого случайного числа обе стороны генерируют ключ для шифрования и расшифровки.
3. Этот шаг завершает согласование и начинает защищенное соединение, передающиеся данные шифруются полученным ключом до закрытия соединения.

Если один из описанных шагов не был выполнен, то согласование прекращается и соединение не устанавливается.

Структура TLS-сообщения:

- ✓ Content Type – 1 байт (тип содержимого);
- ✓ Version (MSB) – 1 байт (версия протокола);
- ✓ Version (LSB) – 1 байт (версия протокола);
- ✓ Length (MSB) – 1 байт (длина сообщения);
- ✓ Length (LSB) – 1 байт (длина сообщения);
- ✓ Protocol Message(s) (сообщения протокола);
- ✓ MAC (коды аутентификации сообщения, необязательно);
- ✓ Padding (сдвиг, для блоковых шифров).

Последние 3 поля имеют переменную длину.

IX. Безопасность удалённого управления

1. Аудит безопасности протокола SNMP

1.1. Определение и функции протокола

Протокол SNMP (Simple Network Management Protocol, простой протокол управления сетью) является протоколом Прикладного уровня, разработанный для выполнения двух задач:

- мониторинг сетевых устройств и сети в целом;
- управление сетевыми устройствами.

Протокол SNMP предоставляет возможность станциям управления считывать и изменять настройки шлюзов, маршрутизаторов, коммутаторов и прочих сетевых устройств.

1.2. Версии протокола SNMP

Опишем различия между версиями протокола SNMP и документы, определяющие эти версии. По состоянию на 2006 год единственной не устаревшей версией SNMP является SNMPv3, определённая в RFC 3411-3418.

SNMPv1

Первые RFC, описывающие стандарты SNMP, появились в 1988 году. Версия 1 подверглась критике за её посредственную модель безопасности на основе сообществ. В то время безопасность в Интернете не входила в круг первоочередных задач рабочих групп IETF.

SNMPv2

Версия 2, известная так же, как Party-based SNMPv2, или SNMPv2p, не получила широкого распространения из-за серьёзных разногласий по поводу инфраструктуры безопасности в стандарте. SNMPv2 улучшал версию 1 в области быстродействия, безопасности, конфиденциальности и взаимодействий «менеджер-менеджер». Он представил новый тип PDU Get-Bulk-Request, альтернативу Get-Next-Request для получения больших объёмов информации при помощи одного запроса. Тем не менее, новая система безопасности на основе сторон выглядела для многих как чересчур сложная и не была широко признана.

SNMPv2c

Community-based SNMPv2, или SNMPv2c, представил SNMPv2 без новой модели безопасности версии 2. Вместо неё предлагалось использовать старую модель безопасности версии 1 на основе сообществ. Соответствующее предложение RFC было принято только как черновик стандарта, однако стало де факто стандартом SNMPv2. Безопасность SNMP снова оказалась нерешённым вопросом.

SNMPv2u

User-based SNMPv2, или SNMPv2u, является компромиссом между незащищённостью SNMPv1 и чрезмерной сложностью SNMPv2p. Предложенная модель безопасности на основе пользователей была положена в основу SNMPv3.

SNMPv3

SNMPv3 наконец-то решил проблемы с безопасностью способом, который многие посчитали приемлемым. Версия 3 SNMP принята IETF как стандарт Интернета (IETF STD 62). Почти все предыдущие RFC признаны устаревшими. Документы, описывающие протокол SNMPv3, приведены ниже:

- *Общая информация*.
  RFC 3411. An Architecture for Describing SNMP Management Frameworks.
- *Обработка сообщений*.
  - Привязки к транспорту.
    RFC 3417. Transport Mappings for the SNMP.
  - Разбор и диспетчеризация сообщений.
    RFC 3412. Message Processing and Dispatching for the SNMP.
  - *Безопасность*.
    RFC 3414. User-based Security Model (USM) for SNMPv3.
- Обработка PDU.
  - Операции протокола.
    RFC 3416. Version 2 of the Protocol Operations for SNMP.
  - Приложения SNMP.
    RFC 3413. SNMP Applications.
  - Управление доступом.
    RFC 3415. View-based Access Control Model (VACM) for the SNMP.
- Модули MIB.
  RFC 3418. MIB for the SNMP.

1.3. Модель протокола SNMP

Общая модель

Модель приведена на рисунке 9.1. Основными взаимодействующими элементами протокола являются агенты (agent) и системы управления сетью (NMS, network management system). С точки зрения концепции «клиент-сервер» роль сервера выполняют агенты, то есть те самые устройства, для опроса состояния которых используется протокол SNMP. Соответственно, роль клиентов отводится системам управления – сетевым приложениям, необходимым для сбора информации о функционировании агентов. Взаимодействие агентов и систем управления осуществляется на основе сообщений протокола SNMP.

Агентами в SNMP являются программные модули, которые работают в управляемых устройствах. Агенты собирают информацию об управляемых устройствах, в которых они работают. Агент содержат всю информация об управляемом сетевом устройстве в базе управляющей информации (MIB, management information base). MIB представляет собой совокупность объектов, доступных для операций записи–чтения.

В любой управляемой сети может иметься одна или более NMS. NMS выполняют прикладные программы сетевого управления, которые представляют информацию управления конечному пользователю.

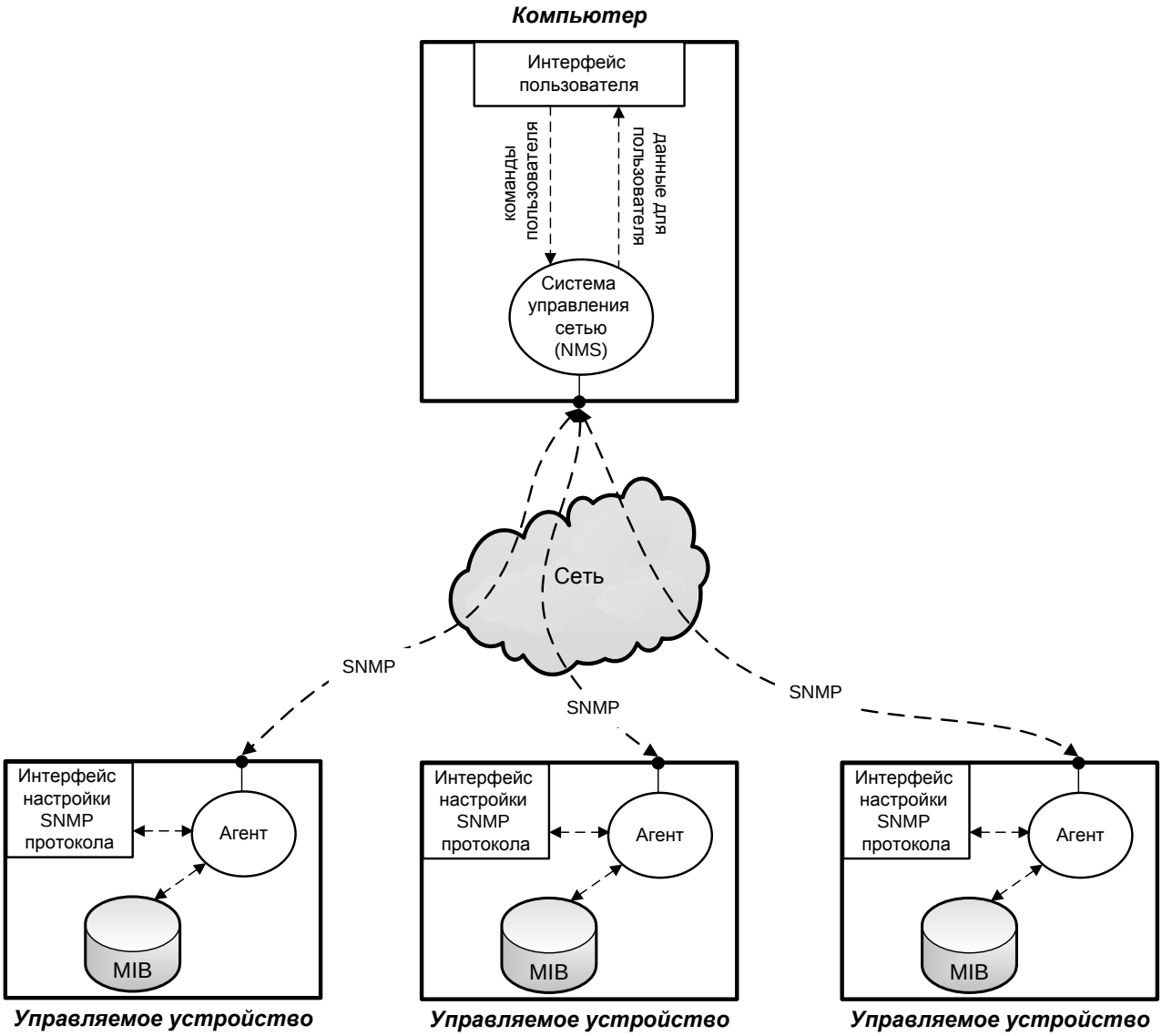


Рисунок 9.1.

Структура базы MIB

На данный момент существует четыре типа информационной базы MIB:

1. Internet MIB – информационная база объектов для обеспечения диагностики ошибок и конфигураций. Включает в себя 171 объект (в том числе и объекты MIB I).
2. LAN manager MIB – база из 90 объектов – пароли, сессии, пользователи, общие ресурсы.
3. WINS MIB – база объектов, необходимых для управления и диагностики WINS-сервера (в серверах Microsoft Windows физически находится в файле WINSMIB.DLL).
4. DHCP MIB – база объектов, необходимых для управления и диагностики DHCP-сервера (в серверах Microsoft Windows физически находится в файле DHCPMIB.DLL).

Структуру MIB определяет документ, называемый SMI (Structure of Management Information, структура управляющей информации). Все MIB имеют иерархическую древовидную структуру. Все базы содержат десять корневых алиасов (ветвей), представленных на рисунке 9.2:

1. *System* – данная группа MIB II содержит в себе семь объектов, каждый из которых служит для хранения информации о системе (версия ОС, время работы и т.д.).
2. *Interfaces* – содержит 23 объекта, необходимых для ведения агентами статистики по сетевым интерфейсам управляемого устройства (количество интерфейсов, размер MTU, скорость передачи данных, физические адреса и т.д.).
3. *AT* – содержит 3 объекта, отвечающих за трансляцию адресов. Более не используется. Была включена в MIB I. Примером использования объектов AT может послужить простая ARP таблица соответствия физических (MAC) адресов сетевых карт IP адресам машин. В SNMP v2 эта информация была перенесена в MIB для соответствующих протоколов.
4. *IP* – содержит 42 объекта, в которых хранятся данные о проходящих IP пакетах.
5. *ICMP* – содержит 26 объектов со статистикой об ICMP-сообщениях.
6. *TCP* – содержит 19 объектов, хранящих статистику по протоколу TCP (соединения, открытые порты и т.д.).
7. *UDP* – содержит 6 объектов, хранящих статистику по протоколу UDP (входящие/исходящие датаграммы, порты, ошибки).
8. *EGP* – содержит 20 объектов – данные о трафике Exterior Gateway Protocol.
9. *Transmission* – зарезервирована для специфических задач.
10. *SNMP* – содержит 29 объектов, в которых хранится статистика по SNMP-протоколу (входящие/исходящие пакеты, ограничения пакетов по размеру, ошибки, данные об обработанных запросах и многое другое).

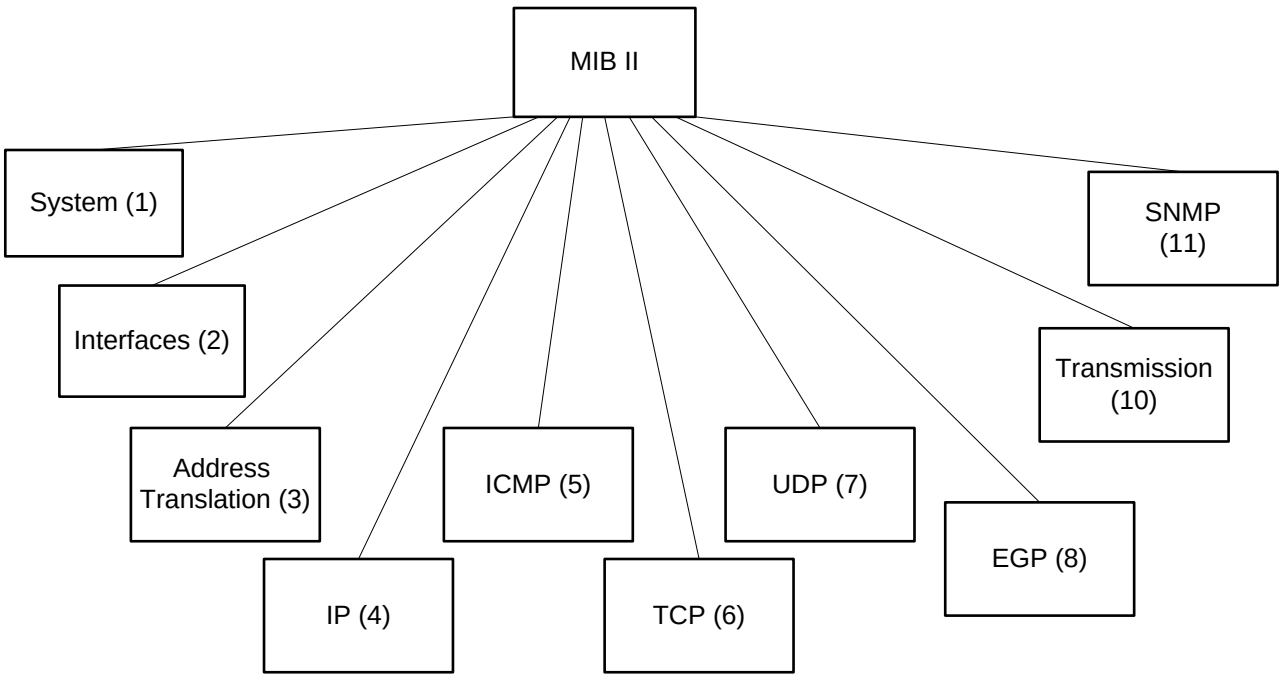


Рисунок 9.2.

Каждая из ветвей в свою очередь также представима в виде дерева. Например, к адресу администратора мы можем обратиться посредством такого пути: system.sysContact.0, ко времени работы системы system.sysUpTime.0. С другой стороны те же данные могут задаваться и в точечной нотации. Так system.sysUpTime.0 соответствует значение 1.3.0, так как system имеет индекс «1» в группах MIB II, а sysUpTime – «3» в иерархии группы system. Ноль в конце пути говорит о скалярном типе хранимых данных. В процессе работы SNMP-протокол использует точечную нотацию, то есть если менеджер запрашивает у

агента содержимое параметра system.sysDescr.0, то в строке запроса ссылка на объект будет преобразована в «1.1.0».

Дерево MIB расширяемо благодаря экспериментальным и частным ветвям. Например, поставщики могут определять свои собственные ветви для включения реализаций своих изделий. В настоящее время вся работа по стандартизации ведется на экспериментальной ветви.

SMI определяет следующие типы данных MIB:

1. Network addresses (сетевые адреса) – символьные строки, представляющие адреса из конкретного стека протоколов. В настоящее время единственным примером сетевых адресов являются 32-битовые IP-адреса.
2. Counters (счетчики) – неотрицательные целые числа, которые монотонно увеличиваются до тех пор, пока не достигнут максимального значения, после чего они сбрасываются до нуля. Примером счетчика является общее число байтов, принятых интерфейсом.
3. Gauges (измерители) – неотрицательные целые числа, которые могут увеличиваться или уменьшаться, но фиксируются при достижении максимального значения. Примером типа «gauges» является длина очереди, состоящей из выходных пакетов.
4. Ticks (тики) – сотые доли секунды, прошедшие после какого-нибудь события. Примером типа «ticks» является время, прошедшее после вхождения интерфейса в свое текущее состояние.
5. Opaque (непрозрачный) – произвольное тип данных. Используется для передачи произвольных информационных последовательностей, находящихся вне пределов точного печатания данных, которое использует SMI.

Основные команды системы NMS

Если NMS хочет проконтролировать какое-либо из управляемых устройств, она делает это путем отправки ему сообщения с указанием об изменении значения одной из его переменных. В целом управляемые устройства отвечают на четыре типа команд (или инициируют их):

1. Reads.
   Для контролирования управляемых устройств NMS считывают переменные, поддерживаемые этими устройствами.
2. Writes.
   Для контролирования управляемых устройств NMS записывают переменные, накопленные в управляемых устройствах
3. Traversal operations.
   NMS используют операции прослеживания, чтобы определить, какие переменные поддерживает управляемое устройство, а затем собрать информацию в таблицы переменных.
4. Traps.
   Управляемые устройства используют «ловушки» для асинхронных сообщений в NMS о некоторых событиях.

1.4. Протокол SNMPv3

Начиная с января 1998 года, выпущен набор документов, посвященных SNMPv3. В этой версии существенно расширена функциональность, разработана новая система безопасности.

Протокол обмена данными

Ниже на рисунке 9.3 приведена временная диаграмма, на которой в общем виде представлен протокол обмена SNMP-сообщениями. Для своей работы протокол SNMP использует транспортный протокол UDP, в основном 161 порт. Но для trap-сообщений используется 162 порт. Возможные команды протокола SNMPv3 приведены в таблице 9.1.

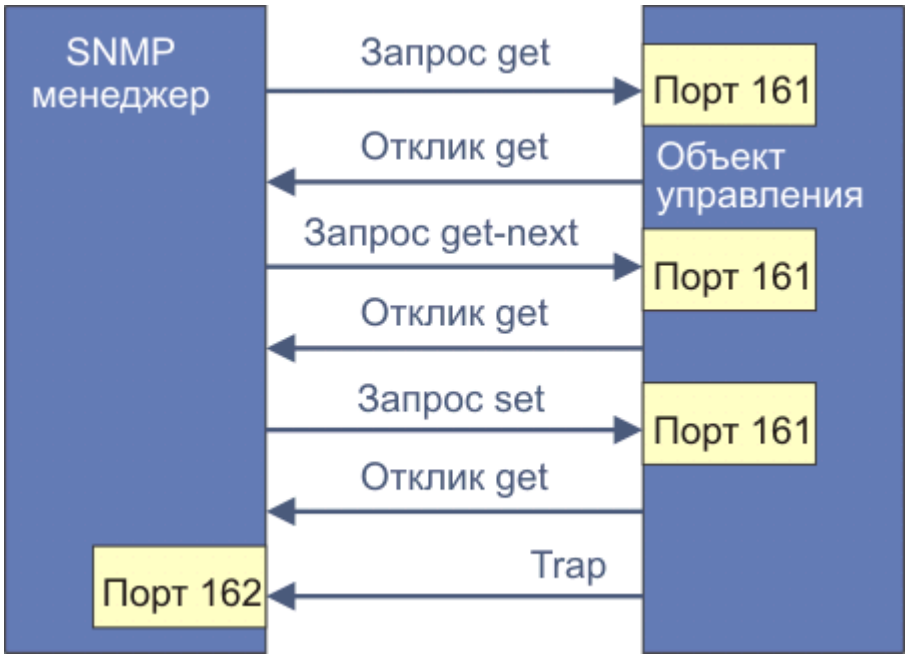


Рисунок 9.3.

Таблица 9.1 – Основные команды протокола SNMPv3

Команда SNMP	Тип PDU	Назначение
GET-request	0	Получить значение указанной переменной или информацию о состоянии сетевого элемента.
GET-next-request	1	Получить значение переменной, не зная точного её имени (следующий логический идентификатор на дереве MIB).
SET-request	2	Присвоить переменной соответствующее значение. Используя для описания действия, которое должно быть выполнено.
GET-response	3	Отклик на GET-request, GET-next-request и SET-request. Содержит также информацию о состоянии (коды ошибок и другие данные).
TRAP	4	Отклик сетевого объекта на событие или на изменение состояния.
GetBulkRequest	5	Запрос пересылки больших объемов данных, например, таблиц.
InformRequest	6	Менеджер обращает внимание партнёра на определенную информацию в MIB.
SNMPv3-Trap	7	Отклик на событие (расширение по отношению к v1 и v2).
Report	8	Отчёт (функция пока не задана).

Формат SNMP-сообщения

Полное описание формата сообщения протокола SNMPv3 дано в документе RFC-3412 в разделе 6 «The SNMPv3 Message Format». Формат сообщения представлен на рисунке 9.4.

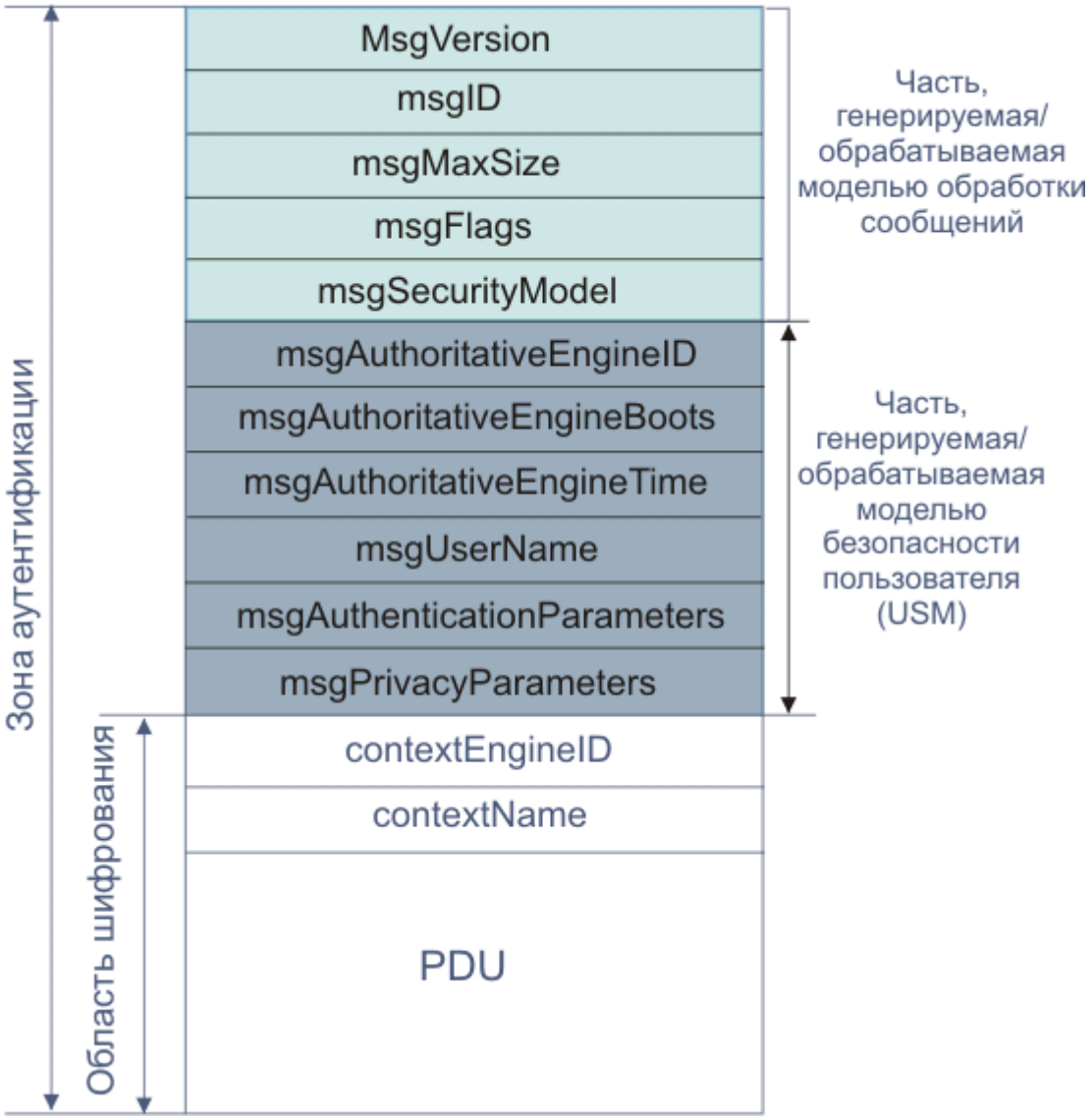


Рисунок 9.4.

SNMP-сообщение логически разделено на три части:

1. Часть, которая формируются отправителем в рамках модели обработки сообщений и обрабатываются получателем.
2. Часть, отвечающая за функции безопасности.
3. Собственно поле данных.

В данном разделе дано описание полей первой и третей частей. Описание полей безопасности дано в разделе 6.5.

Для реализации модели обработки сообщений используются следующие поля:

- **msgVersion**. Версия протокола. Для протокола SNMPv3 значение в поле равно 3.
- **msgID.** Уникальный идентификатор, используемый SNMP-сущностями для установления соответствия между запросом и откликом. Значение msgID лежит в диапазоне 0 - (2^{31}-1).

- **msgMaxSize.** Максимальный размер сообщения в октетах, поддерживаемый отправителем. Его значение лежит в диапазоне 484 - (2^{31}-1) и равно максимальному размеру сегмента, который может воспринять отправитель.
- **msgFlags.** Однобайтовая строка, содержащая три флага в младших битах:
  - *reportableFlag*. Если reportableFlag=1, то должно быть прислано сообщение с отчётом (команда Report). Флаг reportableFlag устанавливается отправителем во всех сообщениях запроса (команды Get, Set, Inform). Флаг устанавливается равным нулю в откликах и Trap-уведомлениях;
  - *privFlag*;
  - *authFlag*.

  Флаги privFlag и authFlag устанавливаются отправителем для индикации уровня безопасности для данного сообщения. Для privFlag=1 используется шифрование, а для authFlag=0 - аутентификация. Допустимы любые комбинации значений флагов кроме privFlag=1 AND authFlag=0 (шифрование без аутентификации).
- **msgSecurityModel**. Идентификатор со значением в диапазоне 0 - (2^{31}-1), который указывает на модель безопасности, используемую при формировании данного сообщения. Зарезервированы значения 1 – для SNMPv1, 2 и 3 – для SNMPv3.

1.5. Безопасность протокола SNMPv3

Модели безопасности протоколов SNMPv1-v3

Перечислим модели безопасности, применяющиеся в соответствующих версиях протокола SNMP:

1. *SNMPv1*
   SNMPv1 – Community-based Security Model
2. *SNMPv2*
   SNMPv2p – Party-based Security Model
   SNMPv2c – Community-based Security Model
   SNMPv2u – User-based Security Model
3. *SNMPv3*
   SNMPv3 – USM User-based Security Model

Модель безопасности на основе сообществ

Модель безопасности на основе сообществ (Community-based Security Model) была первой, самой простой и самой небезопасной. Она подразумевает лишь аутентификацию на основе «строки сообщества», фактически, пароля, передаваемого по сети в теле сообщения SNMP в открытом тексте. Эта модель безопасности не в состоянии бороться ни с одной из угроз информационной безопасности. Тем не менее, она часто используется до сих пор в связи со своей простотой, а также благодаря наличию внешних, не связанных с SNMP систем безопасности, например, межсетевых экранов.

Модель безопасности на основе сторон

Модель безопасности на основе сторон (Party-based Security Model) подразумевает введение понятие стороны. Сторона — это виртуальное окружение исполнения, в котором набор допустимых операций ограничен административно. Сущность SNMP при обработке сообщения действует как сторона, поэтому ограничена операциями, определёнными для этой стороны. Сторона определяется следующими параметрами:

1. Уникальный идентификатор стороны.
2. Логический сетевой адрес (адрес транспортного протокола).
3. Протокол аутентификации и параметры, требующиеся для аутентификации всех сообщений стороны.

4. Протокол шифрования и параметры, требующиеся для шифрования всех сообщений стороны.

Могут использоваться различные алгоритмы для протоколов аутентификации и шифрования. Обычно в качестве алгоритма для протокола аутентификации используют хэш-функцию Message Digest 5 (MD5), а для протокола шифрования — алгоритм Data Encryption Standard (DES) в режиме Cipher Block Chaining (CBC). При использовании соответствующих протоколов аутентификации и шифрования модель успешно справляется с большинством угроз безопасности. Данная модель безопасности не была широко принята, поскольку показалась многим слишком сложной и запутанной.

Модель безопасности на основе пользователей

Модель безопасности на основе пользователей (User-based Security Model) вводит понятие пользователя, от имени которого действует сущность SNMP. Этот пользователь характеризуется именем пользователя, используемыми протоколами аутентификации и шифрования, а также закрытым ключом аутентификации и шифрования. Аутентификация и шифрование являются необязательными. Модель безопасности во многом похожа на модель на основе сторон, но она упрощает идентификацию пользователей, распределение ключей и протокольные операции.

Модель безопасности USM

Модель безопасности USM (User-Based Security Model) использует концепцию авторизованного сервера (authoritative Engene). При любой передаче сообщения одна или две сущности, передатчик или приемник, рассматриваются в качестве авторизованного SNMP-сервера. Это делается согласно следующим правилам:

1. Когда SNMP-сообщение содержит поле данных, которое предполагает отклик (например, Get, GetNext, GetBulk, Set или Inform), получатель такого сообщения считается авторизованным.
2. Когда SNMP-сообщение содержит поле данных, которое не предполагает посылку отклика (например, SNMPv2-Trap, Response или Report), тогда отправитель такого сообщения считается авторизованным.

Таким образом, сообщения, посланные генератором команд, и сообщения Inform, посланные отправителем уведомлений, получатель является авторизованным. Для сообщений, посланных обработчиком команд или отправителем уведомлений Trap, отправитель является авторизованным. Такой подход имеет две цели:

1. Своевременность сообщения определяется с учетом показания часов авторизованного сервера. Когда авторизованный сервер посылает сообщение (Trap, Response, Report), оно содержит текущее показание часов, так что неавторизованный получатель может синхронизовать свои часы. Когда неавторизованный сервер посылает сообщение (Get, GetNext, GetBulk, Set, Inform), он помещает туда текущую оценку показания часов места назначения, позволяя получателю оценить своевременность прихода сообщения.
2. Процесс локализации ключа, описанный ниже, устанавливает единственного принципала, который может владеть ключом. Ключи могут храниться только в авторизованном сервере, исключая хранение нескольких копий ключа в разных местах.

Когда исходящее сообщение передается процессором сообщений в USM, USM заполняет поля параметров безопасности в заголовке сообщения. Когда входное сообщение передается обработчиком сообщений в USM, обрабатываются значения параметров безопас-

ности, содержащихся в заголовке сообщения. В параметрах безопасности содержатся следующие поля:

- **msgAuthoritativeEngeneID.** Идентификатор авторизованного сервера, участвующего в обмене. Это значение идентификатора отправителя для Trap, Response или Report или адресата для Get, GetNext, GetBulk, Set или Inform.
- **msgAuthoritativeEngeneBoots.** snmpEngeneBoots авторизованного сервера, участвующего в обмене. Объект snmpEngeneBoots содержит целочисленные значения в диапазоне 0 - (2^{31}-1). Это поле содержит число, показывающее сколько раз SNMP-сервер был перезагружен с момента конфигурирования.
- **msgAuthoritativeEngeneTime.** Время работы авторизованного сервера, участвующего в обмене. Значение этого поля лежит в диапазоне 0 - (2^{31}-1). Это поле характеризует число секунд, которое прошло с момента последней перезагрузки сервера. Каждый авторизованный сервер должен инкрементировать это поле один раз в секунду.
- **msgUserName.** Имя пользователя, который послал сообщение.
- **msgAuthenticationParameters.** Поле содержит ноль, если при обмене не используется аутентификация. В противном случае данное поле содержит аутентификационный параметр.
- **msgPrivacyParameters.** Поле содержит ноль, если не требуется соблюдения конфиденциальности. В противном случае данное поле содержит параметр безопасности. В действующей модели USM используется алгоритм шифрования DES.

Механизм аутентификации в SNMPv3 предполагает, что полученное сообщение действительно послано пользователем, имя которого содержится в заголовке сообщения, и это имя не было модифицировано во время доставки сообщения. Для реализации аутентификации каждый из пользователей, участвующих в обмене должен иметь секретный ключ аутентификации, общий для всех участников (определяется на фазе конфигурации системы). В посылаемое сообщение отправитель должен включить код, который является функцией содержимого сообщения и секретного ключа. Одним из принципов USM является проверка своевременности сообщения, что делает маловероятной атаку с использованием копий сообщения.

Модель управления доступом

Система конфигурирования агентов позволяет обеспечить разные уровни доступа к базе MIB для различных SNMP-менеджеров. Это делается путем ограничения доступа некоторым агентам к определенным частям MIB, а также с помощью ограничения перечня допустимых операций для заданной части MIB. Такая схема управления доступом называется VACM (View-Based Access Control Model). В процессе управления доступом анализируется контекст (vacmContextTable), а также специализированные таблицы vacmSecurityToGroupTable, vacmTreeFamilyTable и vacmAccessTable.

2. Протокол SSH

2.1. Введение

SSH (Secure Shell) – это сетевой протокол, используемый для удаленного управления компьютером и для передачи файлов. SSH похож по функциональности на протоколы telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH позволяет передавать через безопасный канал любой другой сетевой протокол. Также, SSH может использовать сжатие передаваемых данных для последующей их шифрации.

Первая версия протокола, SSH-1, была разработана в 1995 году исследователем Tatu Ylönen из Технологического университета Хельсинки, Финляндия. SSH-1 был написан для обеспечения большей конфиденциальности, чем протоколы rlogin, telnet и rsh. В 1996 году была разработана более безопасная версия протокола, SSH-2, уже несовместимая с SSH-1. Протокол приобрел еще большую популярность и к 2000 году его использовало уже порядка двух миллионов пользователей. В 2006 году протокол был утвержден рабочей группой IETF в качестве Интернет-стандарта.

Однако, до сих пор в некоторых странах (Франция, Россия, Ирак и Пакистан) требуется специальное разрешение в соответствующих структурах для использования определенных методов шифрования, включая SSH (см. закон Российской Федерации "О федеральных органах правительственной связи и информации").

Распространены две реализации SSH: коммерческая (закрытая) и свободная (открытая). Свободная реализация называется OpenSSH. К 2006 году 80% компьютеров Интернет использовало именно OpenSSH. Коммерческая реализация разрабатывается организацией SSH Inc., http://ssh.com/ - закрытая реализация, бесплатная для некоммерческого использования. Свободная и коммерческая реализации SSH содержат практически одинаковый набор команд.

В первой версии протокола есть существенные недостатки, поэтому в настоящее время SSH-1 практически нигде не применяется Наверное, вы помните знаменитый эпизод из фильма "Матрица-2: Перезагрузка", в котором Тринити использует эксплоит SSHNuke для взлома городской электростанции? Вначале она нашла уязвимый SSH-сервер, просканировав сеть программой Nmap. Затем, через "дыру" "SSH1 CRC2", она получила максимальные права доступа к этому серверу.

Многие взломщики сканируют сеть в поиске открытого порта SSH. Поэтому в целях безопасности необходимо запрещать доступ по ssh для суперпользователя. Обычно злоумышленники подбирают именно пароль суперпользователя. Протокол SSH-2 устойчив к атакам "man-in-middle", в отличие от протокола telnet. То есть, прослушивание трафика, "снифинг", ничего не дает злоумышленнику. Протокол SSH-2 также устойчив к атакам путем присоединения посредине (session hijacking) и обманом сервера имен (DNS spoffing).

Далее по тексту под SSH имеется ввиду именно вторая версия протокола, SSH-2.

Для работы по SSH нужен SSH-сервер и SSH-клиент. Сервер прослушивает соединения от клиентских машин и при установлении связи производит аутентификацию, после чего начинает обслуживание клиента. Клиент используется для входа на удаленную машину и выполнения команд.

SSH – это протокол прикладного уровня. SSH-сервер обычно слушает соединения на TCP-порту номер 22. Спецификация протокола SSH-2 содержится в RFC 4251. Для аутентификации сервера SSH использует алгоритм Diffie-Hellman'a. Для аутентификации клиента – шифрование с открытым ключом (оно сравнительно медленное). Для шифрования передаваемых данных – симметричное шифрование (оно более быстрое). Среди алгоритмов шифрования с открытым ключом чаще всего используются RSA и DSA. Из симметричных алгоритмов – AES, Blowfish и 3DES. Целостность переданных данных проверяется с помощью CRC32 в SSH1 или HMAC-SHA1/HMAC-MD5 в SSH2. Для сжатия шифруемых данных используется алгоритм LempelZiv (LZ77), обеспечивает такую же компрессию, что и архиватор zip.

Для подключения клиента требуется сгенерировать пару из открытого и закрытого ключей. Если используется PuTTY под Windows, то это делается утилитой puttygen.exe. Под Linux обычно используется команда puttygen (для PuTTY) или ssh-keygen (для OpenSSH) и далее указывается, где находится закрытый ключ, а затем производится соединение с вводом пароля. Возможно также беспарольное соединение.

2.2. Протокол SSH

Проект стандарта ssh описывает протоколы ssh и состоит из нескольких документов, которые описывают общую архитектуру протокола, а также протоколы трех уровней:

- ✓ протокол транспортного уровня;
- ✓ протокол аутентификации;
- ✓ протокол соединения.

Их задача – обеспечивать безопасную сетевую службу наподобие удаленного login поверх небезопасной сети.

Протокол транспортного уровня обеспечивает аутентификацию сервера, конфиденциальность и целостность. Протокол аутентификации обеспечивает аутентификацию клиента для сервера. Наконец, протокол соединения ssh мультиплексирует безопасный (шифруемый) канал, представляя его в виде нескольких логических каналов, которые используются для различных целей (различных видов служб).

Протокол транспортного уровня предусматривает возможность сжатия данных. Этот протокол работает поверх соединения TCP/IP. Протокол аутентификации работает поверх протокола транспортного уровня, а протокол соединения – поверх протокола аутентификации.

С целью повышения безопасности осуществляется не только аутентификация клиента сервером, к которому обращается клиент, но и аутентификация сервера клиентом – другими словами, происходит аутентификация обеих сторон.

Клиент шлет запрос на обслуживание в первый раз, когда устанавливается безопасное соединение транспортного уровня ssh. Второй запрос направляется уже после завершения аутентификации пользователя (клиента).

Прежде чем анализировать протоколы ssh подробнее, следует определить понятие ключ хоста. Каждый работающий с ssh хост, на котором может выполняться как клиент, так и сервер, может иметь не менее одного ключа. Несколько хостов могут иметь общий ключ хоста. Однако каждый хост должен иметь хотя бы один ключ, с которым работает каждый из требуемых алгоритмов работы с открытыми ключами. Ключ сервера используется при

обмене открытыми ключами с целью проверки того, что клиент действительно общается с настоящим (а не подменённым) сервером. Для этого клиент должен знать открытый ключ сервера. Это знание реализуется в рамках одной из двух моделей. В первой клиент просто имеет некий локальный файл, в котором каждому имени хоста ставится в соответствие его открытый ключ. Во второй модели вводится понятие сертификационного агента, который и отвечает за проверку соответствия имени хоста его открытому ключу. При этом клиент знает только открытый ключ самого сертификационного агента. В последнем случае упрощается поддержка клиента (ему нужно знать всего один открытый ключ), но появляются высокие требования к сертификационному агенту, который должен иметь открытые ключи всех хостов, к которым обращаются клиенты.

Протоколом предусмотрена возможность отказа от проверки ключа сервера при самом первом обращении клиента к этому серверу. При этом соединение клиент-сервер будет защищено от пассивного прослушивания сети, но возникает опасность атаки типа человек в середине (man-in-the-middle), то есть попытки временной подмены сервера. Если эта возможность используется, ключ хоста-сервера будет автоматически передан клиенту и сохранен в его локальном файле.

Разработчики проекта протокола ssh особенно заботились о его долголетии. Протокол расширяемым; планируется возможность дополнения криптографических алгоритмов, используемых при работе ssh. С этой целью проектом предусмотрено, что между клиентом и сервером происходят переговоры, в результате которых выбираются методы шифрования, форматы открытых ключей и т.п., которые будут использованы в данном сеансе. При этом с целью обеспечения интероперабельности должен поддерживаться некоторый минимальный набор.

Отдельного упоминания заслуживают вопросы увеличения трафика в связи с применением протоколов ssh. Ясно, что при передаче в сети больших пакетов дополнительная нагрузка, вызванная передачей управляющих заголовков ssh, невелика. Основное внимание следует обратить на приложения, для которых характерны короткие пакеты, например, telnet. Минимальный размер заголовка TCP/IP равен 32 байта; минимальный же размер пакета при использовании ssh увеличится с 33 до (примерно) 51 байта.

Учитывая, что в Ethernet минимальная длина поля данных пакета равна 46 байт, дополнительный нагрузкой в 5 байт можно пренебречь. Наиболее существенным влияние ssh оказывается, вероятно, при использовании протокола PPP на низкоскоростных модемных соединениях, поскольку PPP сжимает заголовки TCP/IP. Однако существенный прогресс в скоростях передачи данных позволяет рассчитывать, что дополнительные задержки будут измеряться несколько миллисекундами и останутся незаметны человеку.

Наконец, несколько слов о кодировании сетевых адресов. Поскольку DNS – «ненадежный» протокол, в ssh он не используется. При адресации применяются IP-адреса, причем в проекте заложена поддержка IPv6. Все сообщения (пакеты) ssh содержат номер сообщения число от 1 до 255. Этот диапазон разбит на подинтервалы, причем разным уровням протокола ssh отвечают разные диапазоны.

Таблица 9.2 – Номера сообщений ssh и их назначение

Интервал номеров	Тип сообщений	Протокол
1-19	Транспортный уровень (общая часть)	Транспортный
20-29	Переговоры клиента и сервера о выборе алгоритма	
30-49	Специфические для метода обмена ключами	
50-59	Протокол аутентификации (общая часть)	Аутентификации
60-79	Протокол аутентификации (часть, специфическая для метода аутентификации)	
80-89	Протокол соединения (общая часть)	Соединения
90-127	Сообщения, относящиеся к каналам	
90-128	Резерв (для протоколов клиентов)	
192-255	Локальные расширения	

2.3. Существующее программное обеспечение

Существующие распространённые SSH-сервера:

- ✓ Debian GNU/Linux: dropbear, lsh-server, openssh-server, ssh;
- ✓ MS Windows: freeSSHd, OpenSSH sshd, WinSSHD, ProSHHD, Dropbear SSH Server.

Существующие распространённые SSH-клиенты и оболочки:

- ✓ Debian GNU/Linux: kdessh, lsh-client, openssh-client, putty, ssh;
- ✓ MS Windows: PuTTY, SecureCRT, ShellGuard, Axessh, ZOC, SSHWindows, ProSSHD;
- ✓ Mac OS: NiftyTelnet SSH;
- ✓ Symbian OS: PuTTY;
- ✓ Java: MindTerm, AppGate Security Server.

2.4. Рекомендации по безопасности использования протокола SSH

Следующие действия могут повысить безопасность при использовании протокола SSH:

1. Запрещение удаленного root-доступа.
2. Запрещение подключения с пустым паролем или отключение входа по паролю.
3. Выбор нестандартного порта для SSH-сервера.
4. Использование длинных SSH2 RSA-ключей (2048 бит и более). По состоянию на 2006 год система шифрования на основе RSA считалась надёжной, если длина ключа не менее 1024 бит. Безопасность алгоритма RSA (криптосистемы Ривеста-Шамира-Адельмана) основана на трудности задачи разложения больших чисел на множители. Это означает, что с появлением мощных компьютеров, основанных на квантовых вычислениях, не составит большого труда взломать RSA. Используя законы квантовой механики, можно легко решать задачу факторизации чисел.
5. Ограничение списка IP-адресов, с которых разрешен доступ.
6. Запрещение доступа с некоторых, потенциально опасных адресов.
7. Отказ от использования распространенных или широко известных системных логинов для доступа по SSH.
8. Регулярный просмотр сообщений об ошибках аутентификации.
9. Установка систем обнаружения атак и вторжений.
10. Использование ловушек, подделывающих SSH-сервис.