Погребец А. 016

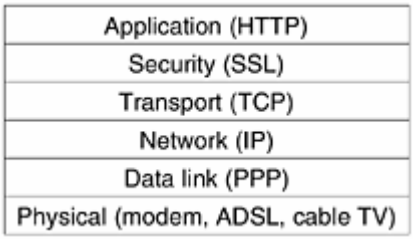
SSL - The Secure Sockets Layer

Один из этапов защиты информации – это защита канала связи. Мы будем рассматривать, какие безопасные соединения доступны. Протокол SSL (secure socket layer) был разработан в 1995 году фирмой Netscape, как протокол обеспечивающий защиту данных между сервисными протоколами (HTTP, NNTP, FTP и т.д.) и транспортными протоколами (TCP/IP). Не секрет, что можно без особых технических ухищрений просматривать данные, которыми обмениваются между собой клиенты и серверы. Был даже придуман специальный термин для этого - "sniffer". А в связи с увеличением объема использования Интернета в коммерческих целях, неизбежно вставал вопрос о защите передаваемых данных. В общем, появление такого протокола как SSL было вполне закономерным явлением. С одной стороны остаются все возможности сервисных протоколов (для программ-серверов), плюс к этому все данные передаются в зашифрованном виде. И раскодировать их довольно трудно. В данный момент протокол SSL принят W3 консорциумом (W3 Consortium) на рассмотрение, как основной защитный протокол для клиентов и серверов (WWW browsers and servers) в сети Интернет.

Работа SSL строится на принципе открытых ключей. Принцип заключается в использовании пары асимметричных ключей (открытого и личного/секретного) для кодирования/раскодирования информации. Открытый ключ раздается всем желающим. И с его помощью шифруются необходимые данные, которые можно расшифровать только с помощью личного ключа.

Расположение SSL в обычном стеке протокола иллюстрировано на **Рис.1** Фактически - это новый слой, вставленный между прикладным слоем (application layer) и транспортным слоем (transport layer), принимая запросы от браузера и посылая вниз к TCP для передачи серверу. Как только безопасная связь была установлена, главная работа SSL заключается в сжатии и шифровании. Когда HTTP-соединение производится через SSL, это называют HTTPS-соединением (Безопасный HTTP). Обычно это соединение доступно по порту (443) вместо стандартного порта (80). Как технология, SSL не ограничена требованием в использовании только Web-браузерами, но именно такой способ использования технологии самый распространенный.

Рис.1. Слои (и протоколы) для соединения с SSL.



SSL состоит из двух подпротоколов, один чтобы установить безопасную связь и один для того, чтобы использовать ее. Передача данных начинается только после того, как безопасное соединение установлено. Процесс установления безопасного соединения показан на **Рис.2**.

Аутентификация (как часть handshake protocol)

В протоколе SSL используется открытые ключи.Теперь закономерно встает вопрос о том, каким образом распространять свои открытые ключи. Для этого (и не только) была придумана специальная форма - сертификат (certificate). Сертификат состоит из следующих частей:

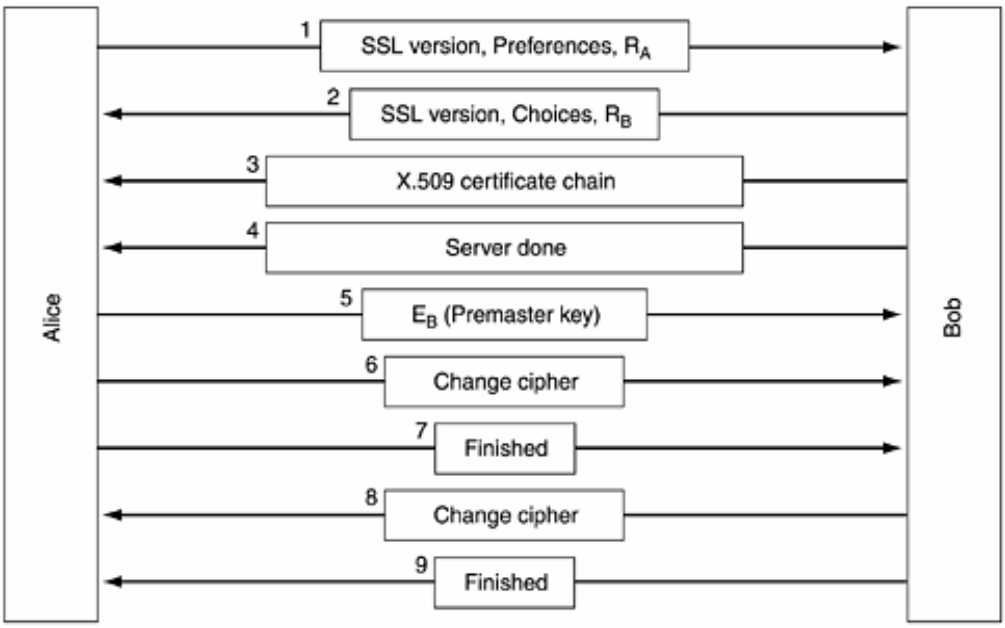
- Имя человека/организации выпускающего сертификат.
- Для кого был выпущен данный сертификат (субъект сертификата).
- Открытый ключ субъекта.
- Некоторые временные параметры (срок действия сертификата и т.п.).

Сертификат "подписывается" личным ключом человека (или огрганизации), который выпускает сертификаты. Организации, которые производят подобные операции называются - "Certificate authority (CA)". Обычно в SSL используются сертификаты стандарта X.509. Именно при помощи таких сертификатов проверяется подлинность пользователя.

Для контроля над пересылкой сообщений (от случайного/преднамеренного изменения) используется специальный алгоритм - Message Authentication Code (MAC). Довольно распространенным является алгоритм MD5. Обычно, и сам MAC-code так же шифруется. В связи с этим достоверность сообщений повышается в несколько раз. И внести изменения в процесс обмена практически невозможно.

Установка соединения (handshake protocol)

Рис.2. Подпротокол установки соединения с SSL.



Алиса (Alice) отправляет Бобу (Bob) запрос, чтобы установить связь. Запрос определяет SSL версию, которую Алиса использует или имеет возможность использовать для соединения, а так же ее предпочтение относительно сжатия и шифровальных алгоритмов. Там так же содержится текущее время, R_A. Боб делает выбор среди различных алгоритмов, с которыми может работать Алиса и посылает собственное время, R_B в зашифрованном виде (шифрует с помощью

своего личного ключа). Тогда в сообщении 3, он посылает сертификат, содержащий его открытый ключ. А так же Боб оповещает Алису что сейчас ее очередь говорить (сообщение 4). Алиса расшифрует это сообщение (с помощью открытого ключа Боба). И сравнив это сообщение с посланным, может убедиться в том, что его действительно послал Боб. Алиса отвечает, выбирая случайные 384 бита под ключ и посылает его Бобу в зашифрованном виде (сообщение 5). После того, как сообщение 5 было получено - и Алиса и Боб способны вычислить ключ сессии. По этой причине, Алиса говорит Бобу переключаться на новый шифр (сообщение 6), а также что она закончила с подпротоколом установления соединения (сообщение 7). Боб в ответ признает ее (сообщения 8 и 9). На данный момент ключ, который будет использоваться в дальнейшем – известен обеим сторонам (master secret).

Передача данных

Как упомянуто выше, SSL поддерживает многократные алгоритмы шифрования. В рамках протокола определены четыре вида шифрования:

- digitally-signed (шифрование электронной подписи)
- stream-ciphered (поточное шифрование)
- block-ciphered (блочное шифрование)
- publick-key-encrypted (шифрование с помощью открытого ключа),

Электронная подпись предполагает использование хэширования до шифрования. В RSA-подписи 36-байтовая структура двух хэш-функций SHA и MD5 шифруется с помощью закрытого ключа. В DSS 20-байтовый блок хэш-функции SHA непосредственно передается на обработку алгоритму цифровой подписи без дополнительного хэширования.

При поточном шифровании исходный текст подвергается обработе с использованием операции XOR с помощью криптографического ключа эквивалентной длины, вырабатываемого посредством генератора псевдослучайных чисел.

В блочном шифровании каждый блок исходного текста преобразуется в равный ему по размеру шифрованный текст. Обычно размер блока равен 64 байтам. Если необходимо исходный текст дополняется до требуемого размера блока нулями.

Протокол SSL предполагает последовательный переход клиента и сервера из одного состояние в другое. Диалоговая часть протокола SSL позволяет координировать работу машин состояний клиента и сервера. Логически любое состояние представляется дважды, в качестве рабочего (operating) состояния и рассматриваемого состояния (pending). Предусмотрены, кроме того, состояния чтения и записи. Когда клиент или сервер получает сообщение **change cipher spec**, он копирует рассматриваемое состояние в текущее состояние чтения. При посылке сообщения **change cipher spec** клиент или сервер копирует рассматриваемое состояние в текущее состояние записи. Когда диалог согласования завершен, клиент и сервер обмениваются сообщениями **change cipher spec**, после чего взаимодействуют друг с другом, используя согласованную спецификацию шифрования. Протокол SSL допускает любое число соединений между клиентом и сервером в рамках одной сессии.

Состояние сессии харакетризуется рядом параметров:

Параметр	Описание параметра
session identifier	Произвольная последовательность байтов, выбранная сервером чтобы идентифицировать активную сессию
peer sertificate	Сертификат партнера - X509.v3. Этот элемент может быть равен нулю
compression method	Алгоритм, используемый для сжатия информации перед шифрованием
cipher spec	Спецификация алгоритма шифрования данных (например, нуль или DES) и алгоритм MAC (например MD5 или SHA). Она определяет криптографические атрибуты, такие как hash_size.
master secret	48-байтовый секретный код, общий для клиента и сервера
is resumable	Флаг, указывающий, что сессия может использоваться для формирования новых соединений

Состояние соединения характеризуется следующими параметрами.

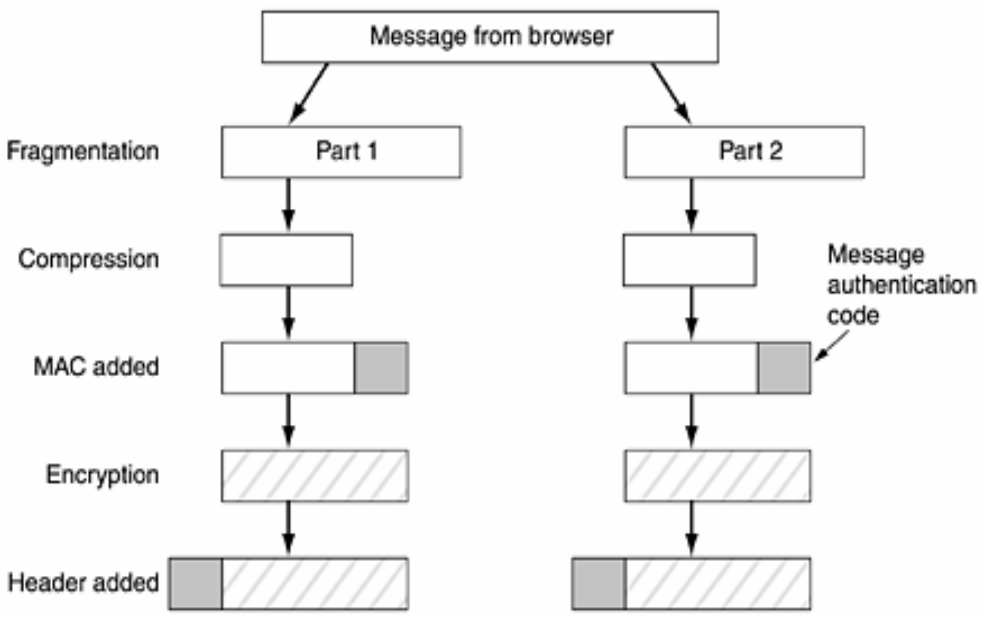
Параметр	Описание параметра
server and client random	Последовательность байтов, выбираемая сервером и клиентом для каждого соединения
server write MAC secret	Секретный код, используемый в MAC-операциях с данными, записанными сервером
client write MAC secret	Секретный код, используемый в MAC-операциях с данными, записанными клиентом
server write key	Ключ шифрования данных шифруемых сервером и расшифруемых клиентом
client write key	Ключ шифрования данных шифруемых клиентом и расшифруемых сервером
Initialization vectors	Когда используется блочный шифр в режиме CBC, для каждого ключа поддерживается инициализационный вектор (IV). Это поле устанавливается первым в процессе стартового диалога.
sequence numbers	Каждая из сторон поддерживает свои номера по порядку для переданных и полученных сообщений для каждого из соединений. Когда партнер посылает или получает сообщение **change cipher spec**, соотвествующее число, обнуляется. Значение номера не может превышать 2^{64}-1.

В процессе диалога (handshake protocol) фиксируются следующие атрибуты: версия протокола, идентификатор сессии, шифровой набор и метод сжатия информации. Когда диалог согласования завершен, партнеры имеют общий

секретный код (они считают его по формуле, описанной ниже), который используется для шифрования записей и для вычисления аутентификационных кодов MAC. Методики шифрования и вычисления MAC заданы в спецификации cipherspec. MAC вычисляется до шифрования основных данных. Протокол диалога является одним из клиентов высокого уровня протокола записей SSL

Для фактической транспортировки, второй подпротокол используется, как показано на **Рис.3**. Сообщения от браузера сначала разбиты на блоки по 16 Кб. Если допускается сжатие, каждая блоки сжимается отдельно. В качестве MAC используется фрагмент сообщения, зашифрованный с помощью специального секретного ключа. При этом может использоваться также технология MD5, что дает 128-битовый дайджест*. Применение такого дайджеста делает вероятность случайного подбора равной ~1/[20*10^{18}].. Сжатый фрагмент с MAC далее зашифровывается с согласованным симметрическим алгоритмом шифрования (обычно операция XOR с RC4 от ключевого потока). Наконец, подключается заголовок и информация передается по TCP-соединению.

Рис.3. Передача данных используя SSL.



Общеизвестно, что RC4 имеет довольно слабые ключи, которые легко поддаются криптоанализу. Безопасность SSL, использующего RC4, сомнительна. Браузеры, которые позволяют пользователю выбирать алгоритмы шифрования, должны формироваться так, чтобы можно было использовать 3DES с ключами на 168 бит и SHA-1 все время, даже при том, что эта комбинация медленнее чем RC4 и MD5.

* Иногда информацию передают в виде двух сообщений. Первое сообщение передается открытым текстом, а второе сообщение зашифровано с помощью личного ключа посылающего. Такое двойное сообщение называется **message digest.**

Дополнительная информация

Контроль аутентичности

Сервер, с которым на основании протокола SSL связывается клиент, всегда обязан для своей идентификации представлять т.н. сертификат, т.е. файл, содержащий его открытый ключ и другие параметры, характеризующие сервер. Сертификат всегда подписан электронной подписью третьего-доверенного лица. Используя эту электронную подпись клиент всегда контролирует, заранее приобретенный отрытый ключ третьего-доверенного лица. Если сертификат не подписан или подписан неизвестным лицом, то об этом сообщается клиенту и связи не происходит.

Схема, описанная выше, не предусматривает аутентификации клиента. Предполагается, что аутентификация клиента на сервере будет осуществлена после установления секретного соединения (например, путем передачи на сервер пароля). Таким образом, эта функция возлагается на того, кто реализует конкретный сервис. Спецификация SSL 3.0 предусматривает аутентификацию клиента, однако используется она достаточно редко. В зависимости от использования SSL соединения можно настраивать протокол так, чтобы он запрашивал сертификат клиента, а можно настроить чтобы он этого не делал. По RFC 2246 (спецификация протокола SSL и TSL) аутентификации клиента не обязательна, но является допустимой. В тех случаях, где она нужна, достаточно правильно задать параметры соединения. В случае аутентификации клиента – используется тот же способ, что и при аутентификации сервера.

Создание сессионного ключа

Используется способ шифрования, где для передачи зашифрованного сообщения используется ключевая пара, т.е. используют в паре два ключа отрытый и секретный. Открытый ключ пересылается своему партнеру. Другой, секретный ключ никогда и никому не оглашают, он - секретный. Ключевая пара симметрична - данные, которые шифруются открытым ключом можно расшифровать только секретным ключом. И наоборот. Такая симметрия весьма полезна в случае шифрования открытым ключем. Если сервер подлинный, то оба - клиент и сервер генерируют один временный и действующий только в течение данного сеанса связи ключ, которым они будут в дальнейшем шифровать отправляемую информацию. Пересылаемый другой стороне ключ шифруется открытым ключом получаемой стороны. Это гарантирует, что ключ, предназначенный для шифрования сеанса связи, получит только адресат, потому что только он имеет секретный ключ для расшифрования.

Если вернуться к Алисе и Бобу, то Алиса знает сеансовый ключ, так как сама его сгенерировала, перед тем как послать Бобу (это сообщение 5). Боб знает сеансовый ключ, так как расшифровал сообщение Алисы на своем секретном ключе. Поскольку оба располагают одним и тем же секретным ключом, они могут воспользоваться схемой симметричного шифрования для обмена секретными сообщениями (один и тот же ключ для шифрования и расшифрования). Очевидно, что сеансовый ключ меняют от сеанса к сеансу.

Реализованы следующие схемы симметричного шифрования:

- RC4, ключ 128 бит;
- RC4, ключ 40 бит;

- RC2, ключ 128 бит;
- RC2, ключ 40 бит;
- IDEA, ключ 128 бит;
- DES, ключ 64 бит;
- DES, режим EDE 3, ключ 192 бит.

Обозначим случайные сообщения клиента (Алиса) и сервера (Боб) как ClientHello.random и ServerHello.random. Pre_master_secret – ключ, определенный в сообщении 5. Тогда ключ (master secret) вычисляется по следующей формуле :

```
master_secret =
    MD5(pre_master_secret + SHA('A' + pre_master_secret +
        ClientHello.random + ServerHello.random)) +
    MD5(pre_master_secret + SHA('BB' + pre_master_secret +
        ClientHello.random + ServerHello.random)) +
    MD5(pre_master_secret + SHA('CCC' + pre_master_secret +
        ClientHello.random + ServerHello.random));
```

Список литературы и материалов :

1. Secure Programming for Linux and Unix HOWTO
http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html

2. Computer Networks, Fourth Edition
By Andrew S. Tanenbaum, March 17, 2003

3. Apache The Definitive Guide, 3rd Edition
By Ben Laurie, Peter Laurie, December 2002

4. Apache-SSL
http://www.apache-ssl.org/

5. SSL 3.0 Specification
http://wp.netscape.com/eng/ssl3/

6. RFC 2246 (SSL and TSL)
http://lattice.itep.ru/old/UNIX/RFC/rfc2246.html