Московский Физико-Технический Институт (ГУ)

Факультет Радиотехники и Кибернетики

Эссе по курсу "Защита информации", кафедра радиотехники:

SSL: безопасная передача данных по сети Интернет.

Желнин Сергей, 315 гр.

http://www.re.mipt.ru/infsec

г. Долгопрудный, 2007 год

1.Содержание.

2.Основные сведения и общее описание.

Протокол SSL (Secure Socket Layer) криптографический протокол, обеспечивающий безопасную передачу данных по сети интернет. SSL был разработан американской компанией Netscape Communications как протокол, обеспечивающий защиту данных между сервисными протоколами (такими как HTTP, FTP и т.д.) и транспортными протоколами (TCP/IP) в соединениях "точка-точка". Ранее можно было без особых технических сложностей просматривать данные, которыми обмениваются между собой клиенты и серверы. Существует даже специальный термин для этого – "sniffer". После введения в 1994 году протокола SSL, он стал де-факто стандартом для осуществления торговли через интернет и скорее всего сохранит свое положение в будущем. Сейчас SSL поддерживается всеми популярными браузерами.

Протокол SSL предназначен для решения традиционных задач обеспечения защиты информационного взаимодействия:

- пользователь и сервер должны быть взаимно уверены, что они обмениваются информацией не с подставными абонентами, а именно с теми, которые нужны, не ограничиваясь паролевой защитой;
- после установления соединения между сервером и клиентом весь информационный поток между ними должен быть защищен от несанкционированного доступа;
- при обмене информацией стороны должны быть уверены в отсутствии случайных или умышленных искажений при ее передаче.

Протокол SSL позволяет серверу и клиенту перед началом информационного взаимодействия аутентифицировать друг друга (client-server authentification), согласовать алгоритм шифрования и сформировать общие криптографические ключи. С этой целью в протоколе используются двухключевые (ассиметричные) криптосистемы, в частности, RSA.

Т.к. ассиметричное шифрование довольно медленное, то конфиденциальность информации, передаваемой по установленному защищенному соединению, обеспечивается путем шифрования потока данных на сформированном общем секретном ключе с использованием

симметричных криптографических алгоритмов (например, RC4, RC2, DES и др.). Контроль целостности передаваемых блоков данных производится за счет использования кодов аутентификации сообщений (Message Autentification Code, MAC), вычисляемых с помощью хэш-функций (например MD5).

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. 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.

Протокол SSL включает три этапа взаимодействия сторон защищаемого соединения:

- переговоры о типе SSL-сессии;
- обмен ключами и аутентификация посредством сертификатов
- защита потока данных.

На первом этапе стороны договариваются об используемых криптографических алгоритмах.

На втором этапе осуществляется аутентификация сервера и (опционально) клиента, стороны формируют общий "секрет", на основе которого создаются общие сеансовые ключи для последующей защиты соединения. Первые два этапа называют также процедурой «рукопожатия» (handshake).

На третьем этапе (защита потока данных) информационные сообщения прикладного уровня нарезаются на блоки, для каждого блока вычисляется код аутентификации сообщений, затем данные шифруются с использованием симметричных криптографических алгоритмов и отправляются приемной стороне. Приемная сторона производит обратные действия: расшифрование, проверку кода аутентификации сообщения, сборку сообщений, передачу на прикладной уровень.

SSL используется криптография с открытым (публичным) ключом, также известная как асимметричная криптография. Она использует два ключа: один -- для шифрования, другой -- для расшифровывания сообщения. Два ключа математически связаны таким образом, что данные, зашифрованные с использованием одного ключа, могут быть расшифрованы только с использованием другого, парного первому. Каждый пользователь имеет два ключа -- открытый и секретный (приватный). Пользователь делает доступным открытый ключ любому корреспонденту сети. Пользователь и любой корреспондент, имеющий открытый ключ, могут быть уверены, что данные, зашифрованные с помощью открытого ключа, могут быть расшифрованы только с использованием секретного ключа.

Целостность и аутентификация сообщения обеспечиваются использованием электронной цифровой подписи.

Теперь встает вопрос о том, каким образом распространять свои публичные ключи. Для этого используется специальная форма - сертификат (certificate). Сертификат состоит из нескольких частей:

- имя человека/организации, выпускающей сертификат;
- субъект сертификата (для кого был выпущен данный сертификат);
- публичный ключ субъекта;
- некоторые временные параметры (срок действия сертификата и т.п.).

Сертификат "подписывается" приватным ключом человека (или организации), который выпускает сертификаты. Организации, которые производят подобные операции называются Certificate authority (CA). Если в стандартном Веб-клиенте (веб-браузере), который поддерживает SSL, зайти в раздел security, то там можно увидеть список известных

организаций, которые "подписывают" сертификаты.

3.Алгоритмы SSL «рукопожатия» и аутентификации сервера.

3.1. SSL «рукопожатие» (SSL handshake).

В процесс SSL рукопожатия входят следующие этапы:

1. Клиент посылает серверу версию своего SSL протокола, криптографические настройки и другие данные, которые необходимы серверу для взаимодействия с клиентом с использованием SSL протокола.
2. Сервер посылает клиенту версию своего SSL протокола, криптографические настройки и другие данные, которые необходимы клиенту для взаимодействия с сервером посредством SSL протокола. Сервер также посылает свой сертификат и, если клиент запросил область ресурсов сервера, для обращения к которой необходим сертификат со стороны клиенты, сервер посылает запрос на сертификат клиента.
3. Клиент использует информацию, принятую от сервера, чтобы аутентифицировать сервер(см. п. 3.2. Аутентификация сервера). Если сервер не может быть аутентифицирован, пользователь информируется о том, что защищенное соединение не может быть установлено. Если сервер успешно аутентифицирован, клиент переходит к этапу 4.
4. Используя все сгенерированные до этого этапа данные, клиент(в зависимости от выбраного метода шифрования) создает предварительный секретный ключ сеанса(pre-master secret), шифрует его открытым ключом сервера(полученным из сертификата сервера, который был послан на этапе 2) и посылает его серверу.
5. Клиент подписывает следующую порцию данных, которая уникальная для текущего «рукопожатия» и известна и клиенту и серверу. Если сервер запросил аутентификацию клиента, то клиент посылает серверу кроме подписанных данных и зашифрованного предварительного секретного ключа(pre-master secret) еще и свой сертификат.
6. Если сервер запросил аутентификацию клиента, то сервер пытается аутенфицировать клиента. Если клиент не может быть аутентифицирован, сеанс заканчивается. Если клиент может быть аутентифицирован, сервер использует свой секретный ключ(private key) для расшифрования предварительного секретного ключа(pre-master secret), производит некоторые действия(которые клиент также производил) для того, чтобы сгенерировать секретный ключ(master secret).
7. И клиент и сервер используют секретный ключ(master secret), чтобы создать сеансовые ключи, которые являются симметричными ключами, необходимые для шифрования и расшифрования информации, которой обмениваются во время SSL сеанса и для проверки целостности данных(то есть определения любых изменений данных в моменты времени, когда они посылаются и когда они принимаются).
8. Клиент посылает сообщение серверу, информирующее о том, что будующие сообщения от клиента будут шифроваться сеансовыми ключами. Затем клиент посылает отдельное зашифрованное сообщение, обозначающее, что клиент закончил свою часть «рукопожатия».
9. Сервер посылает сообщение клиенту, информирующее о том, что будующие сообщения от сервера будут шифроваться сеансовыми ключами. Затем сервер посылает отдельное зашифрованное сообщение, обозначающее, что сервер закончил свою часть «рукопожатия».

10. SSL «рукопожатие» завершено и сеанс начинается. Клиент и сервер используют сеансовые ключи для шифрования и расшифрования данных, которых они посылают друг другу, и для проверки их целостности.
11. Нормальное условие защищенного канала. В любое время, вследствие внутренней или внешней причины, любая сторона провести повторное «рукопожатие», в следствие чего, процесс «рукопожатия» повторится.

3.2. Аутентификация сервера.

Во время «рукопожатия» сервер посылают клиенту сертификат, чтобы стать аутентифицированным. В процесс аутентификации сервера входят следующие этапы.

1. Текущая дата лежит в пределах периода действия сертификата? Клиент проверяет период действия сертификата сервера. Если текущяя дата лежит за пределами периода действия, то процесс аутентификации заканчивается неудачей. Если текущая дата и время лежит в этом пределе, клиент преходит к этапу 2.
2. Проверяемый Certificate Authority (CA) является доверяемым CA? Каждый клиент имеет список доверенных CA сертификатов. Если проверяемый сертификат найден в этом списке, то клиент переходит к этапу 3. Если не найден, то процесс аутентификации заканчивается неудачей.
3. Проверяемый CA открытый ключ удовлетворяет цифровой подписи? Клиент использует открытый ключ CA сертификата чтобы установить достоверность достоверность цифровой подписи сертификата. Если информация в сертификате сервера изменилась с момента, когда он был подписан CA, или открытый ключ CA сертификата не соответствует секретному ключу, который был использован CA чтобы подписать CA сертификат, то клиент не аутентифицирует сервер. Если цифровая подпись достоверна, клиент рассматривает сертификат сервера как валидный.
4. Доменное имя в сертификате сервера совпадает с доменным именем сервера? Этот этап подтверждает, что сервер действительно находится по тому сетевому адресу, заявленному в сертификате. Хотя этап 4 не является технической частью протокола SSL, он предоставляет защиту от формы атаки, называемой «человек посередине». Клиент должен выполнять этот этап и должен завершать аутентификацию сервера или установку соединения, если доменные имена не совпадают. Если доменные имена совпадают, то переход на этап 5.
5. Сервер аутентифицирован. Клиент продолжает SSL «рукопожатие».

4.Применение и реализация.

SSL работает на прикладном уровне внизу таких протоколов как HTTP, FTP, SMTP, NNTP, XMPP и на транспортном уровне поверх протоколов надежной передачи данных, например, TCP. Обычно SSL используется вместе с HTTP протоколом. Для доступа к страницам, защищённым протоколом SSL, в URL вместо обычного префикса «http», как правило, применяется префикс «https» (порт 443), указывающий на то, что будет использоваться SSL-соединение. HTTPS используется для безопасной передачи данных в сети WWW в сферах электронной коммерции, управления активами и других сферах, требующих обмена конфиденциальной информацией.

SSL так же может быть использован для работы с VPN. Это, например, реализовано в пакете OpenVPN. Были проведены значительные разработки для создания технологии, с помощью которой клиент может работать с клиент-серверными приложениями без использования

браузера. По сравнению с традиционным IPSec, SSL имеет некоторые преимущества при прохождении пакетов через файерволы и NAT.

Существует множество программных реализаций. Можно перечислить самые популярные пакеты: OpenSSL, SSLeay, GnuTSL и др.

Самой дорогой вычислительной частью SSL шифрования является стадия, когда SSL сервер должен расшифровать сеансовый ключ (симметричный ключ), который был прислан SSL клиентом во время SSL «рукопожатия». Чтобы значительно снизить нагрузку на сервер используют аппаратные SSL ускорители - со-процессоры, обычно подключаемый через шину PCI.

5.Безопасность.

В заключении можно объединить все методы и способы обеспечения безопасности в протоколе SSL и перечислить их еще раз:

- Клиент использует CA открытый ключ, чтобы подтвердить цифровую подпись CA. Если цифровая подпись может быть подтверждена, клиент принимает сертификат сервера как валидный сертификат.
- Клиент проверяет полученый Certificate Authority на предмет того, находится ли данное CA в списке доверяемый CA.
- Клиент проверяет период действия сертификата. Аутентификация останавливается, если текущее время и дата находятся за пределами периода действия.
- Для того,чтобы защититься от атак «человека посередине», клиент сравнивает DNS сервера с DNS сертификата.
- Нумерация всех записей и использование порядкового номера в MAC'e.
- Сообщение, которым завершается «рукопожатие», содержит хэш всех данных, которыми обменялись клиент и сервер.
- Псевдослучайная функция разделяет входящие данные на две части и обрабатывает каждую из частей разными хэш-алгоритмамы(MD5 и SHA), затем делает xor с ними. Это дает защиту на случай, если один из алгоритмов оказывается уязвимый.

Некоторые сайты были подвергнуты критике за то, что они неправильно использовали SSL, что сводило на нет все преимущества SSL.

1. Страницы, где находятся логин-формы, не являются HTTPS страницами, даже если данные посылаются по SSL.

Для того чтобы увеличить скорость, некоторые разработчики сайтов стали помещать формы для логина на HTTP-страницы, в то время как информация посылается через как HTTPS в безопасном режиме. На первый взгляд, это вполне достаточно, чтобы защититься от злоумышленников, ведь данные после отправления передаются через HTTPS. Но такой подход имеет 2 уязвимости: после подтверждения пользователем информации нет гарантии, что «человек посередине» не перенаправит POST на HTTPS сайт, который он контролирует; «человек посередине» может использовать так называемые keystroke-sniffers, которые сканируют нажатия клавиш при вводе текста в формы(это может быть скрипты на Jscript).

2. Добавление HTTP содержимого на HTTPS страницу.

Некоторые странички могут иметь смешанный контент(HTTP и HTTPS). Это также может

привести к взлому. Причина в том, что «человек посередине» может перехватить и переписать HTTP трафик и изменить HTTPS страницу, используя например стандартный DHTML. Он может скинировать любую информацию, которая его интересует, и перенаправлять POST на сервер, который он контролирует.

6.Литература.

1.http://en.wikipedia.org/wiki/Ssl

2.http://support.microsoft.com/

3.http://blogs.msdn.com/ie/archive/2005/04/20/410240.aspx

4.http://www.ibdarb.ru/