Железов Роман, 911 группа.

Настройка протокола SSL на локальном Web сервере (под Windows)

Моя работа заключалась в том чтобы изучить протокол SSL и испытать его работу на практике. Для этого на моем рабочем компьютере (OS Windows 2000 Server) я решил установить Web server с протоколом SSL. Дать запрос на простейшую html страницу. И перехватить пакет с помощью программы Network Monitor. Сравнить перехватываемые данные при включенном SLL и при выключенном Испытывался стандартный виндовый сервер Internet Information Server. Его установка в системе не вызывает никаких трудностей. Для этого необходимо зайти в Add/Remove Windows Components и проставить галочку в Internet Information Services. Что и было сделано.
На этом локальном Web Servere была создана виртуальная директория и в нее положен тествый html файл с текстом.
После этого была изучена работа стандартного виндового Network Monitora. Программа перехватывает все фреймы на Сетевом уровне и позволяет просмотреть содержимое любого из них. После настройки фильтра и отсечения broadcastov действительно удалось увидет содержимое http запроса и соответственно его ответа. В режиме без SSL весь текст содержащийса во фреймах легко находится и читается.
Затем я приступил к основной части процесса проверки SSL. Но это оказалось не так просто. А именно.
Чтобы включить режим SSL на вебсервере достаточно поставить соответствующую галочку. Но дело в том что пока Web серверу не приписан сертификат этот пункт меню задисейблен. В меню Server Certificates я пытался получить сертификат online из интернета. Там есть такая опция, но у меня ничего не получилось. Видимо Microsoft сервера не очень мне доверяли и не стали его выдавать. А может это потому что у меня виртуальный IP.
Так как получить сертификат online не удалось, я начал искать другой способ это сделать. Во первых с помощью специальной опции в WebServere я сфрмировал текстовый файл для запроса сертификата. Первой строкой этого файла было
-----BEGIN NEW CERTIFICATE REQUEST-----
В нем содержалось что-то закодированное.
-----END NEW CERTIFICATE REQUEST-----

Далее этот файл можно было предоставить сертификационному центру и получить сертификат.
Начитавшись виндовых helpов я обнаружил, что можно поставить локальный MS Certificate Server. На него в Windows components Update была обнаружена своя галочка. И вскоре у меня уже стоял этот сервер. Я приступил к его изучению.
Этот сервер позволял отслеживать все выданные, просроченные и др. сертификаты. И эти сертификаты можно было получить предоставив файл с запросом сертификата. Причем делать это можно через http://*servername*/certsrv где servername= localhost. После предоставления необходимого файла и выбора на сервере (Get Webserver Certificate) я получил сертификат.
Этот сертификат я предоставил моему Web серверу. После его принятия я смог включить SSL протокол. Стали доступны следущие интересные опции:

Require 128 bit encryption
Этот пункт позволял включить режим максимальной безопасности.

Client Certificates(ignore client certificates; accept client certificates; require client certificates)
Этот интересный пункт позволял выставить режим как Web server сервер реагирует на клиентские сертификаты. Либо он их игнорирует, либо просто принимает, либо даже требует их предоставления.

Enable client certificate mapping
В этом пункте меню можно было сопоставить конкретным клиентским сертификатам необходимый User account. Т.е. очень надежная аутентификация.

Enable client certificate trust list
Здесь можно было выставить с какими сертификатами разрешен доступ на сервер.

Работа по настройка была закончена. Затем я подал прежний html запрос на свой Web сервер. Но на этот раз с помощью Network Monitora никакого текста обнаружить не удалось.

Вот такая практическая работа была проведена. Хотя она и не претендует на простоту и правильность реализации, но было в учебных целях была проведена большая работа.

Далее идет статья из интернета, которую я прочитал и изучил.

Протокол SSL. Безопасный уровень соединителей

Для обеспечения безопасного доступа к WEB-серверам разработан протокол SSL. Этот протокол является доминирующим для шифрования обмена между клиентом и сервером (http:// home.netscape.com/eng/ssl3/ssl-toc.html). Протокол SSL имеет два уровня: протокол записей (SSL Record Protocol) и протокол диалога (SSL Handshake Protocol). Последний позволяет клиенту и серверу идентифицировать друг друга и согласовать использование определенного алгоритма шифрования и обменяться ключами.

Потокол SSL реализует следующие функции:

- Конфиденциальность соединения. После предварительного диалога определяется секретный ключ, который используется для симметричной криптографии (например, DES или rc4)
- Партнеры идентифицируют друг друга с помощью асимметричных криптографических методов (например, RSA или DSS)
- Обеспечение надежности соединения. Пересылка включает в себя контроль целостности сообщений с применением кода аутентификации MAC (Message Autentification Code) и хэш функций (SHA или MD5).

Рассмотрим базовые принципы, на которых зиждится протокол ssl. Первая проблема, которую решает данный протокол, - идентификация клиента. Это делается с использованием алгоритма RSA, использующего схему двух ключей. Если необходимо идентифицировать клиента A, он посылает серверу свой открытый ключ. Сервер

пересылает клиенту некоторое сообщение, которое клиент шифрует с помощью своего секретного ключа и посылает серверу. Сервер дешифрует это сообщение с помощью открытого ключа клиента и сравнивает с исходным текстом. При совпадении клиент однозначно идентифицирован. Действительно ведь только он владеет секретным ключем. Перехват и любая попытка фальсификации сообщения здесь не пройдут. Но еще большую надежность достигается, когда клиент шифрует не само сообщение, а дайджест этого сообщения. Так как по дайджесту практически невозможно восстановить исходное сообщение. Такая система называется электронной подписью. Но для решения проблемы идентификации серверу необязательно присылать сообщение, текст сообщения может послать вместе с зашифрованной копией и сам клиент. Но в этом случае злоумышленнику нетрудно имитировать обмен и выдать себя за соответствующего клиента. Чтобы исключить такую возможность, разработана система сертификации. Сертификат включает в себя идентификатор эмитента сертификата, объект, для которого сформирован сертификат, общедоступный ключ объекта и временные метки. Сертификат представляет собой стандартное средство установления связи между открытым ключем и именем его владельца. Сертификат подписывается секретным ключем субъекта, сформировавшего сертификат (эмитентом). Открытый ключ эмитента сертификата общеизвестен. Злоумышленник может узнать сертификат клиента, но не его секретный ключ. В рамках данного протокола можно пересылать и специальный секретный ключ, который может использоваться в симметричных алгоритмах DES, RC4 или IDEA.

Злоумышленник, расположившийся между отправителем и получателем не может прочесть зашифрованные сообщения, но может их исказить. Чтобы противодействовать этому вводится код аутентификации сообщения (MAC). 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 $\sim 1/[20*10^{18}]$.

В рамках протокола SSL 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 предолагает 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.

Шифрование с использованием открытого ключа предполагает дешифровку с применением секретного ключа или наоборот (ключи, образующие пару, симметричны с точки зрения своего использования).

Протокол SSL предполагает последовательный переход клиента и сервера из одного состояние в другое. Каждая процедура реализуема в строго определенном состоянии объекта. 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. Разрешена также реализация произвольного числа сессий параллельно.

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 дешифруемых клиентом
client write key	Ключ шифрования данных шифруемых клиентом и дешифруемых 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, характеризующее номер обнуляется. Значение номера не может превышать 2^{64}-1.

Уровень записей протокола ssl осуществляет фрагментацию сообщений, разбивая их на блоки 2^{12} байт или короче. Все записи сжимаются с использованием согласованного для данной сессии алгоритма архивации. Сжатие всегда должно производиться без потерь и не может увеличивать длину содержимого более чем на 1024 байта. Все записи защищаются с помощью шифрования и алгоритма MAC, заданного в текущей спецификации cipherspec.

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, 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 блочных шифров (RC2 или DES) шифрование и MAC-функции преобразуют структуры sslcompressed.fragment в структуры sslciphertext.fragment.

Возможно, вы не раз пользовались протоколом ssl, даже не подозревая об этом, так как большинство броузеров Netscape и Microsoft используют именно его. Впервые SSL был введен в 1994 году в первую версию netscape navigator. Подготавливаемый IETF безопасный протокол транспортного уровня Интернет (TLS; http://www.consensus.com/ietf-tls) базируется на SSL v3.0. Любая прикладная программа, базирующаяся на протоколе TCP, может быть легко переделана для работы с SSL. Таким образом можно обезопасить, например, telnet сессии, чтение новостей и программу транспортировки электронной почты. SSL-связь предполагает использование традиционного TCP/IP-сокета (обычно это порт 443). SSL использует симметричную модель для шифрования (DESв режиме CBC, тройной DES, RC2 или RC4). Для формирования дайджеста сообщения применяется

стандарт MD5 или алгоритм хэширования SHA. Для аутентификации здесь используется система с общедоступным ключом RSA или алгоритм транспортировки закрытых ключей Диффи-Хелмана.

Конфигурации этих средств безопасности стандартизованы (см. табл. 6.5.1).

Таблица 6.5.1

Набор	**Уровень безопасности**	**Описание**
DES-CBC3-MD5	Очень высокий	Тройной DES в режиме CBC, хэш MD5, 168-битный ключ сессии
DES-CBC3-SHA	Очень высокий	Тройной DES в режиме CBC, хэш SHA, 168-битный ключ сессии
RC4-MD5	Высокий	RC4, хэш MD5, 128-битный ключ
rc4-sha	Высокий	RC4, хэш SHA, 128-битный ключ
RC2-CBC-MD5	Высокий	RC2 в режиме CBC, хэш MD5, 128-битный ключ
DES-CBC-MD5	Средний	DES в режиме CBC, хэш MD5, 56-битный ключ
DES-CBC-SHA	Средний	DES в режиме CBC, хэш SHA, 56-битный ключ
EXp-DES-CBC-SHA	Низкий	DES в режиме CBC, хэш SHA, 40-битный ключ
EXP-RC4-MD5	Низкий	Экспортное качество RC4, хэш MD5, 40-битный ключ
EXP-RC2-CBC-MD5	Низкий	Экспортное качество RC2, хэш MD5, 40-битный ключ
null-MD5	-	Без шифрования, хэш MD5, только аутентификация
null-SHA	-	Без шифрования, хэш SHA, только аутентификация

Когда клиент пытается установить связь с сервером, возникает диалог, во время которого определяется конфигурация набора средств безопасности. Обычно это наиболее эффективный набор, поддерживаемый каждым из партнеров. Некоторые WEB-сервера позволяют администратору произвести дополнительную настройку этого переговорного процесса. Например, разрешить доступ к некоторым секциям каталога только клиентам, поддерживающим высокий уровень безопасности. SSL имеет встроенный модуль сжатия исходной информации. При обмене шифруются URL запрошенного документа и сам документ, заполняемые формы и бланки, HTTP-заголовок и некоторая другая информация. Схема работы ssl пояснена на рис. 6.5.1.

Рис. 6.5.1. Алгоритм работы SSL

Сообщение *ClientHello* содержит в себе информацию о клиенте, включая номер поддерживаемой версии SSL, перечень наборов средств безопасности (см. табл. 6.5.1) и номенклатуру используемых методов сжатия информации. Сообщение *ServerHello* несет в себе идентификатор сессии и информацию о том, какой набор средств безопасности и метод сжатия данных выбрал сервер. Сертификат сервера решает проблему аутентификации. Запрос сертификата клиента является опционным (и пока используется достаточно редко). В ответ на такой запрос клиент должен прислать сертификат или уведомление о его отсутствии. Сообщение *ClientKeyExchange* служит для выбора симметричного ключа. Для разных наборов средств безопасности процедура, разумеется, варьируется. Ключ шифруется перед отправкой с помощью общедоступного RSA-ключа, извлеченного из сертификата сервера. Сообщение *CertificateVerify* является опционным и посылается лишь в случае аутентификации клиента (если клиент посылал свой сертификат). Обмен сообщениями *ChangeCipherSpec* между клиентом и сервером свидетельствует о том, что они оба готовы к обмену информацией с использованием оговоренных шифров и ключа сессии. Обмен сообщениями *Finished,* содержащими хэш-функции MD5 и SHA, позволяет убедиться партнерам, что вся информация дошла до них неповрежденной. Сообщение *Finished* посылается сразу после отправки сообщения change cipher specs, подтверждая, что обмен ключами и аутентификация завершились успешно. Подтверждение получения самого сообщения *Finished* не нужно. После этого обмена система клиент-сервер готова к шифрованному обмену и может его начать немедленно. Рассмотренный сценарий может несколько варьироваться. Так сервер вместо посылки своего сертификата (X.509v3) может послать сообщение *ServerKeyExchange*. Такая модификация может произойти, если, например, планируется применение обмена ключами по методу Диффи-Хелмана. Заметим, что в этом случае взаимная идентификация не производится и, вообще говоря, сессия может подвергнуться атаке со стороны посредника, находящегося между клиентом и сервером.

Для определения конфигурации сервера, поддерживающего протокол SSL, применяются определенные системные переменные. Эти переменные могут также использоваться и CGI-скриптами. Ниже перечислены наиболее часто используемые конфигурационные переменные SSL. Имена переменных, начинающихся с HTTPS, стандартизированы (де-факто).

HTTPS	= ON, если SSL работает, в противном случае OFF
HTTPS_SECRETKEYSIZE	Размер ключа сессии в байтах
HTTPS_KEYSIZE	Исходный размер ключа. Он равен **HTTPS_SECRETKEYSIZE,** если используется не экспортная версия алгоритма шифрования.
HTTPS_CIPHER	Точное имя шифрового набора (табл. 6.5.1)

HTTPS_EXPORT	= TRUE, если броузер и/или сервер является экспортным, в противном случае = FALSE
SSL_PROTOCOL_VERSION	Используемая версия SSL (2 или 3)
SSLEAY_VERSION	Версия используемой библиотеки **SSLEAY**
SSL_SERVER_KEY_SIZE	Используемый размер в байтах общедоступного (секретного) ключа сервера, используемых для аутентификации и выбора ключа сессии
SSL_SERVER_KEY_ALGORITHM	Используемый сервером алгоритм аутентификации с общедоступным ключом (обычно RSA)
SSL_SERVER_SIGNATURE	Алгоритм подписи сервера, например, RSA-MD5
SSL_CLIENT_KEY_SIZE	Используемый размер в байтах общедоступного (секретного) ключа клиента, используемых для аутентификации
SSL_CLIENT_KEY_ALGORITHM	Используемый клиентом алгоритм аутентификации с общедоступным ключом (обычно RSA)
SSL_CLIENT_SIGNATURE	Алгоритм подписи клиента, например, RSA-MD5
SSL_SERVER_CN, SSL_SERVER_EMAIL, SSL_SERVER_OU и т.д.	Эти конфигурационные переменные соответствуют аналогично названным атрибутам в сертификате сервера. Например, **SSL_SERVER_CN** соответствует атрибуту “common name”
SSL_CLIENT_CN, SSL_CLIENT_EMAIL, SSL_CLIENT_OU и т.д.	Эти конфигурационные переменные соответствуют аналогично названным атрибутам в сертификате клиента
SSL_CLIENT_ICN,	Эти конфигурационные

SSL_CLIENT_IEMAIL, SSL_CLIENT_IOU и т.д.	переменные соответствуют аналогично названным атрибутам в подписи, идентифицирующей полномочия сертификата клиента