Содержание

Введение

Сетевая безопасность влечет за собой защиту данных от атак во время их передачи по сети. Для достижения этой цели было разработано много протоколов безопасности в реальном времени. Существуют популярные стандарты для сетевых протоколов безопасности в реальном времени, такие как S / MIME, SSL / TLS, SSH и IPsec. Как упоминалось ранее, эти протоколы работают на разных уровнях сетевой модели.

Причиной популярности использования безопасности на транспортном уровне является простота. Проектирование и развертывание защиты на этом уровне не требует каких-либо изменений в протоколах TCP / IP, которые реализованы в операционной системе. Только пользовательские процессы и приложения должны быть разработаны / изменены, что является менее сложным.

1. Анализ предметной области

1.1 История протокола

В середине 90х компания Netscape разрабатывает протокол, повышающий безопасность электронных платежей. Данный протокол получает имя SSL версия 1.0, но не публикуется из-за проблем безопасности в ее реализации. В феврале 1995 года выпускается SSL 2.0, но версия тоже имеет много уязвимостей. В итоге в 1996 появляется версия SSL 3.0. Из-за проблем с безопасностью, а так же проблем с Netscape. юридических вопросах, на смену приходит TLS основанный на SSL v3.0.

TLS являлся окрытым стандартном (в отличии от SSL, поддерживаемого компанией Netscape) и в результате полностью заменил собой SSL. В 1999 выходит последующая версия, которая стандартизируется инженерным советом сети Интернет (IETF). Протокол получает новое название — TLS 1.0.

Спустя 7 лет, весной 2006 года выходит следующая версия протокола — TLS 1.1. В ней значительно расширены функции и устранены актуальные уязвимости.

В 2008 году выходит TLS 1.2, в которой качественно изменились методы шифрования. Введены новые режимы блочного шифрования, а устаревшие методы криптографического хэширования запрещены.

Самая свежая версия протокола на сегодняшний день — TLS 1.3, выпущенная в 2018 году. Из нее убраны устаревшие хеши, шифры без аутентификации и открытые методы получения ключей к сессиям. Неактуальные опции, вроде вспомогательных сообщений и сжатия данных, также убраны. Введен режим обязательной цифровой подписи, разделены процессы согласования и аутентификации. Чтобы повысить параметры безопасности протокола TLS, версия 1.3 не имеет обратной совместимости с RC4 или SSL.

1.2 Принцип работы TLS

Процесс работы TLS можно разбить на три части:

1. TLS Handshake
2. TLS False Start
3. TLS Chain of trust

TLS Handshake — согласует параметры соединения между клиентом и сервером (способ шифрования, версию протокола), а также проверяет сертификаты. Данная процедура использует большое количество вычислительных ресурсов, поэтому, чтобы каждый раз не устанавливать новое соединение и не проверять сертификаты повторно, была разработана процедура TLS False Start.

TLS False Start — процедура возобновления сессии. Если ранее открывалась сессия между клиентом и сервером, данный этап позволяет пропустить процедуру Handshake, используя данные, которые были сконфигурированы ранее. Однако в целях безопасности каждая сессия имеет свой срок жизни и, если он истек, она будет повторно открыта с помощью процедуры TLS Handshake.

TLS Chain of trust — обязательная процедура TLS-соединения. Она обеспечивает аутентификацию между клиентом и сервером. Она строится на «цепочке доверия», которая основана на сертификатах подлинности, выдаваемых Сертификационными центрами. Центр сертификации проверяет подлинность сертификата и, если он скомпрометирован, данные отзываются. Благодаря данной процедуре и происходит проверка подлинности передаваемых данных.

1.3 TLS Handshake

Версии Handshake TLS 1.2 и Handshake TLS 1.3 имеют множество отличей и прежде чем перейти к каждой, рассмотрим общий принцип:

- Клиент связывается с сервером и запрашивает безопасное соединение. Сервер отвечает списком шифров - алгоритмическим набором для создания зашифрованных соединений - которым он знает, как пользоваться. Клиент сравнивает список со своим списком поддерживаемых шифров, выбирает подходящий и дает серверу знать, какой они будут использовать вдвоем.

- Сервер предоставляет свой цифровой сертификат - электронный документ, подписанный третьей стороной, который подтверждает подлинность сервера. Самая важная информация в сертификате - это публичный ключ к шифру. Клиент подтверждает подлинность сертификата.

- Используя публичный ключ сервера, клиент и сервер устанавливают ключ сессии, который они оба будут использовать на протяжении всей сессии, чтобы шифровать общение. Для этого есть несколько методов. Клиент может использовать публичный ключ, чтобы шифровать произвольное число, которое потом отправляется на сервер для расшифровки, и обе стороны потом используют это число, чтобы установить ключ сессии.

Ключ сессии действителен только в течение одной непрерывной сессии. Если по какой-то причине общение между клиентом и сервером прервется, нужно будет новое рукопожатие, чтобы установить новый ключ сессии.

Вывод

2. Классификация существующих решений

2.1 Пошаговый процесс рукопожатия в TLS 1.2

1. Первое сообщение называется «Client Hello». В этом сообщении перечислены возможности клиента, чтобы сервер мог выбрать шифронабор, который будет использовать для связи. Также сообщение включает в себя большое случайно выбранное простое число, называемое «случайным числом клиента».

2. Сервер вежливо отвечает сообщением «Server Hello». Там он сообщает клиенту, какие параметры соединения были выбраны, и возвращает своё случайно выбранное простое число, называемое «случайным числом сервера». Если клиент и сервер не имеют общих шифронаборов, то соединение завершается неудачно.

3. В сообщении «Certificate» сервер отправляет клиенту свою цепочку SSL-сертификатов, включающую в себя листовой и промежуточные сертификаты. Получив их, клиент выполняет несколько проверок для верификации сертификата. Клиент также должен убедиться, что сервер обладает закрытым ключом сертификата, что происходит в процессе обмена/генерации ключей.

4. Это необязательное сообщение, необходимое только для определённых методов обмена ключами (например для алгоритма Диффи-Хеллмана), которые требуют от сервера дополнительные данные.

5. Сообщение «Server Hello Done» уведомляет клиента, что сервер закончил передачу данных.

6. Затем клиент участвует в создании сеансового ключа. Особенности этого шага зависят от метода обмена ключами, который был выбран в исходных сообщениях «Hello». Так как мы рассматриваем RSA, клиент сгенерирует случайную строку байтов, называемую секретом (pre-master secret), зашифрует её с помощью открытого ключа сервера и передаст обратно.

7. Сообщение «Change Cipher Spec» позволяет другой стороне узнать, что сеансовый ключ сгенерирован и можно переключиться на зашифрованное соединение.

8. Затем отправляется сообщение «Finished», означающее, что на стороне клиента рукопожатие завершено. С этого момента соединение защищено сессионным ключом. Сообщение содержит данные (MAC), с помощью которых можно убедиться, что рукопожатие не было подделано.

9. Теперь сервер расшифровывает pre-master secret и вычисляет сеансовый ключ. Затем отправляет сообщение «Change Cipher Spec», чтобы уведомить, что он переключается на зашифрованное соединение.

10. Сервер также отправляет сообщение «Finished», используя только что сгенерированный симметричный сеансовый ключ, и проверяет контрольную сумму для проверки целостности всего рукопожатия.

После этих шагов SSL-рукопожатие завершено. У обеих сторон теперь есть сеансовый ключ, и они могут взаимодействовать через зашифрованное и аутентифицированное соединение.

2.2 Пошаговый процесс рукопожатия в TLS 1.3

Рукопожатие TLS 1.3 значительно короче, чем его предшественник.

1. Как и в случае TLS 1.2, сообщение «Client Hello» запускает рукопожатие, но на этот раз оно содержит гораздо больше информации. TLS 1.3 сократил число поддерживаемых шифров с 37 до 5. Это значит, что клиент может угадать, какое соглашение о ключах или протокол обмена бу-

дет использоваться, поэтому в дополнение к сообщению отправляет свою часть общего ключа из предполагаемого протокола.

2. Сервер ответит сообщением «Server Hello». Как и в рукопожатии 1.2, на этом этапе отправляется сертификат. Если клиент правильно угадал протокол шифрования с присоединёнными данными и сервер на него согласился, последний отправляет свою часть общего ключа, вычисляет сеансовый ключ и завершает передачу сообщением «Server Finished».

3. Теперь, когда у клиента есть вся необходимая информация, он верифицирует SSL-сертификат и использует два общих ключа для вычисления своей копии сеансового ключа. Когда это сделано, он отправляет сообщение «Client Finished».

Вывод

В приведённом выше объяснении рукопожатие разделено на десять отдельных этапов. В действительности же многие из этих вещей происходят одновременно, поэтому их часто объединяют в группы и называют фазами.

У рукопожатия TLS 1.2 можно выделить две фазы. Иногда могут потребоваться дополнительные, но когда речь идёт о количестве, по умолчанию подразумевается оптимальный сценарий.

В отличие от 1.2, рукопожатие TLS 1.3 укладывается в одну фазу, хотя вернее будет сказать в полторы, но это всё равно значительно быстрее, чем TLS 1.2.

Заключение