



ПРОТОКОЛЫ СЕТЕВОЙ БЕЗОПАСНОСТИ

Андрусенко Юлия Алексеевна

преподаватель кафедры информационной безопасности автоматизированных систем,
Северо-Кавказский федеральный университет, РФ, г. Ставрополь

1. Механизмы безопасности на сетевых уровнях

Безопасность на прикладном уровне – меры безопасности, используемые на этом уровне, зависят от конкретного приложения. Для различных типов приложений потребуются отдельные меры безопасности. Для обеспечения безопасности прикладного уровня приложения необходимо модифицировать. Примером протокола безопасности прикладного уровня является SSH — это протокол позволяющий реализовывать удаленное управление ОС и туннелирование TCP-соединений. Основным недостатком состоит в том, что на прикладном уровне в общем случае невозможно предотвратить несанкционированное событие, т.к. контролируется сам факт того, что событие произошло, поэтому на подобное событие лишь можно отреагировать (максимально оперативно), с целью минимизаций последствий.

Безопасность на транспортном уровне – меры безопасности на этом уровне могут использоваться для защиты данных в одном сеансе связи между двумя хостами. Наиболее распространенными протоколами являются TLS и SSL. Безопасность на сетевом уровне – меры безопасности на этом уровне могут использоваться во всех приложениях. В некоторых средах протокол безопасности сетевого уровня (IPSec) обеспечивает гораздо лучшее решение, чем элементы управления транспортного или прикладного уровня, из-за трудностей добавления элементов управления в отдельные приложения. Однако протоколы безопасности на этом уровне обеспечивают меньшую гибкость связи, которая может потребоваться некоторым приложениям.

1.1. Протоколы защиты прикладного уровня

SSh — это протокол позволяющий реализовывать удаленное управление ОС и туннелирование TCP-соединений. Протокол похож на работу Telnet, но в отличие от них, шифрует все, даже пароли. Протокол работает с разными алгоритмами шифрования. SSH-соединение может создаваться разными способами.

Обычно протокол работает с 22 портом. Также протокол использует алгоритмы электронной цифровой подписи для реализации аутентификации. Также протокол подразумевает сжатия данных, но используется редко и по запросу клиента.

Безопасная реализация SSH:

- запрещение подключения с пустым паролем
- выбор нестандартного порта для ssh-сервера
- использовать длинные ключи более 1024 бит

1.2. Протоколы защиты транспортного уровня

Протоколы SSL и TLS

Сразу нужно отметить, что это один и тот же протокол. Сначала был SSL, но его взломали, он был доработан и выпущен как TLS. Конфиденциальность реализуется шифрованием данных с реализацией симметричных сессионных ключей. Сессионные ключи также шифруются, только на основе открытых ключей взятых из сертификатов абонентов. Протокол SSL предполагает следующие шаги при установке соединения:

1. аутентификация сторон
2. согласование криптоалгоритмов для реализации
3. создание общего секретного мастер-ключа
4. генерация сеансовых ключей на основе мастер-ключа

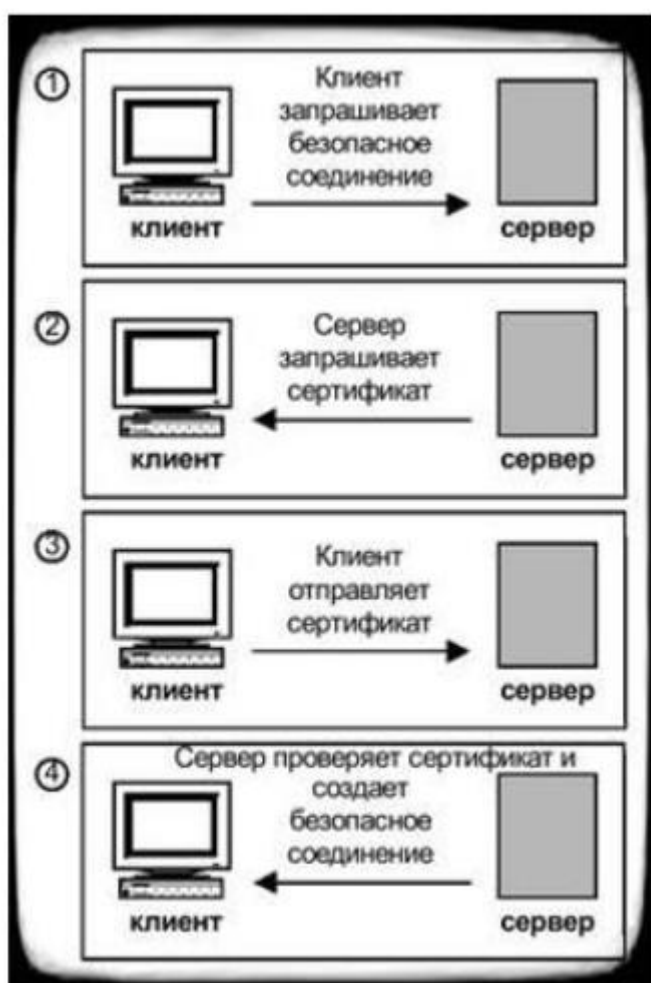


Рисунок 1. Процесс аутентификации клиента сервером с помощью протокола SSL

Следует отметить, что TLS и SSL работают только с одним протоколом сетевого уровня — IP.

Протокол SOCKS

Протокол SOCKS реализует алгоритмы работы клиент/серверных связей на транспортном уровне через сервер-посредник или прокси-сервер. Такой алгоритм уже разрешает создавать функцию трансляции сетевых IP-адресов NAT.

Замена у исходящих пакетов внутренних IP-адресов отправителей разрешает скрыть топологию сети от 3 лиц, тем самым усложняя задачу несанкционированного доступа.

С помощью этого протокола межсетевые экраны и VPN могут реализовывать безопасное соединение между разными сетями. Относительно спецификации протокола SOCKS разделяют SOCKS-сервер, который ставят на шлюзы сети, и SOCKS-клиент, который ставят на конечные узлы.

Схема создания соединения по протоколу SOCKS v5 описана следующими шагами:

1. Запрос клиента перехватывает SOCKS-клиент на компьютере
2. После соединения с SOCKS-сервером, SOCKS-клиент отправляет все идентификаторы всех методов аутентификации, которые он может поддерживать
3. SOCKS-сервер выбирает один метод. Если сервер не поддерживает ни один метод, соединение разрывается
4. Происходит процесс аутентификации
5. После успешной аутентификации SOCKS-клиент отправляет SOCKS-серверу IP или ВТЫ нужного узла в сети.
6. Далее сервер выступает в роли ретранслятора между узлом сети и клиентом