ПРОТОКОЛЫ СЕТЕВОЙ БЕЗОПАСНОСТИ

Кузьменко Григорий Сергеевич

студент, Северо-Кавказский федеральный университет, РФ, г. Ставрополь

Гиш Татьяна Александровна

научный руководитель, доцент кафедры информационной безопасности автоматизированных систем, Северо-Кавказский федеральный университет, РФ, г. Ставрополь

Пелешенко Виктор Сергеевич

научный руководитель, доцент кафедры информационной безопасности автоматизированных систем, Северо-Кавказский федеральный университет, РФ, г. Ставрополь

Андрусенко Юлия Алексеевна

научный руководитель, 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 разрешающий реализовывать удаленное управление ОС и туннелирование TCP-соединений. Основной недостаток 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 разрешающий реализовывать удаленное управление ОС и туннелирование TCP-соединений. Протокол похож на работу Telnet, но в отличии от них, шифрует все, даже пароли. Протокол работает с разными алгоритмами шифрования. *SSH* -соединение может создаваться разными способами:

- реализация socks-прокси для приложений, которые не умеют работать с ssh-туннелями
- VPN-туннели также могут использовать протокол ssh

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

Безопасная реализация SSH:

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

1.2. Протоколы защиты транспортного уровня 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 шали при установки соединения:

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

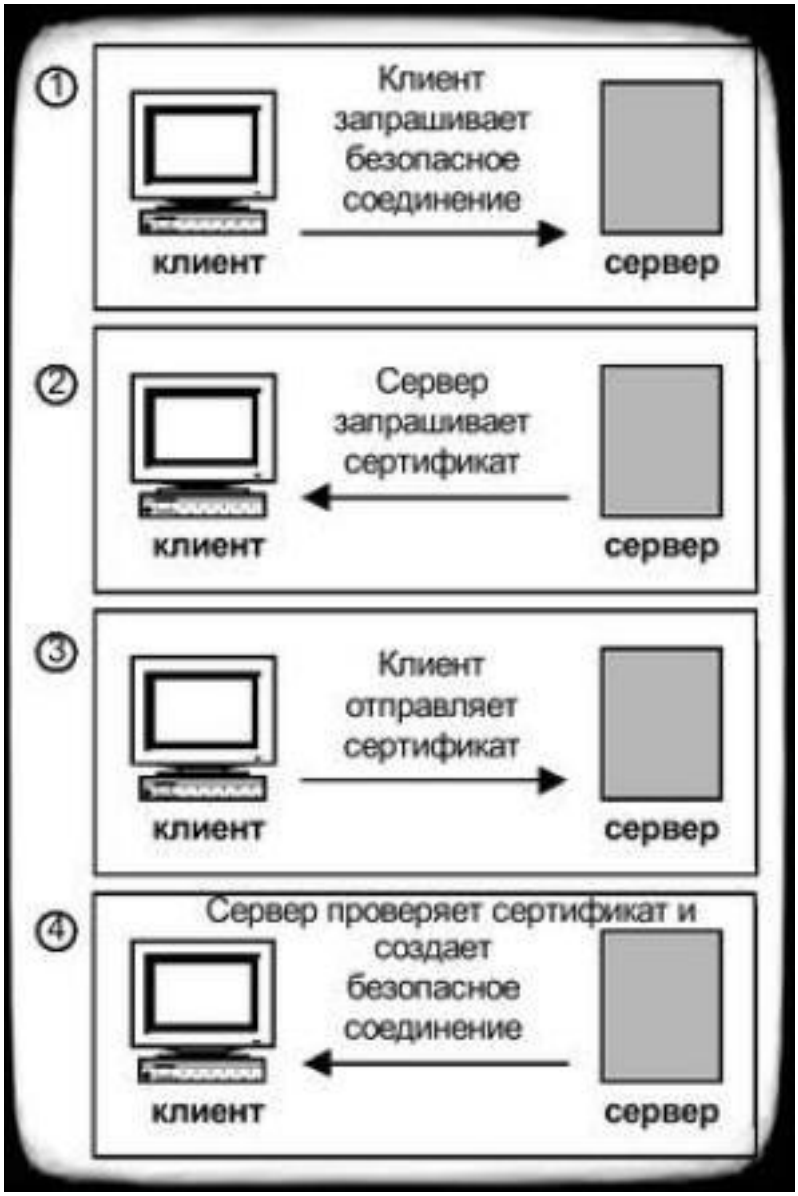

Рисунок 1. Процесс аутентификации клиента сервером с помощью протокола SSL

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 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 поддержать
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

Ограничения защиты на транспортном уровне

- Используется в ПО на основе TCP
- Заголовки TCP / IP в открытом виде.
- Подходит для прямой связи между клиентом и сервером. Не обслуживает защищенные приложения, использующие цепочку серверов
- SSL не обуславливает отказ от авторства, поскольку аутентификация клиента не является обязательной.
- При необходимости, аутентификация пользователя должна быть выполнена выше SSL.

Применение безопасности транспортного уровня имеет множество преимуществ, однако протокол безопасности, разработанный на этих уровнях, может использоваться только с протоколом TCP. Они не обеспечивают безопасность связи, реализованной с использованием UDP.

1.1. Протоколы защиты прикладного уровня

протокол **IPSec**

Главная задача протокола *IPSec* это реализация безопасности передачи информации по сетям IP.

Доступность — протокол не реализует, это входит в задачу протоколов транспортного уровня TCP. Реализуемая защиты на сетевом уровне делает такую защиту невидимой для приложений. Протокол работает на основе криптографических технологий:

- обмен ключами с помощью алгоритма Диффи-Хеллмана
- криптография открытых ключей для подлинности двух сторон, что бы избежать атак типа «человек по середине»
- блочное шифрование
- алгоритмы аутентификации на основе хеширования

IPSec позволяет защитить сеть от множества сетевых атак, откидывая чужие пакеты до того, как они дойдут к уровню IP на узле. На узел могут войти те пакеты, которые приходят от аутентифицированных пользователей.

1.2. Протоколы защиты канального уровня

Обеспечение безопасности беспроводных сетей — достаточно сложная задача. Затруднения вызваны невозможностью физически изолировать злоумышленников от сети или отследить их местоположение.

Канальный уровень в сетях Ethernet очень подвержен нескольким атакам. Наиболее распространенные атаки –

- ARP спуфинг- Подмена ARP может позволить злоумышленнику выдать себя за легитимного хоста, а затем перехватить кадры данных в сети, изменить или остановить их.
- MAC Flooding- При атаке с использованием MAC-адреса злоумышленник заполняет коммутатор MAC-адресами, используя поддельные пакеты ARP, до тех пор, пока таблица CAM не заполнится.
- Порт Кража - Атака кражи портов использует эту способность коммутаторов.

Атакующий заполняет коммутатор поддельными кадрами ARP с MAC-адресом целевого хоста в качестве адреса источника.

Безопасность так же сильна, как и самое слабое звено. Когда дело доходит до сетей, уровень 2 может быть очень слабым звеном. Общая черта этих протоколов видна в реализации организации защищенного многопротокольного удаленного доступа к ресурсам сети через открытую сеть. Для передачи конфиденциальной информации из одной точки в другую сначала используется протокол PPP, а затем уже протоколы шифрования.

Протокол **PPTP**

Протокол *PPTP* определяет реализацию крипто-защищенного туннеля на канальном уровне OSI. *PPTP* отлично работает с протоколами IP, IPX или NETBEUI.

Протокол **L2TP и L2f**

Протокол *L2TP* основан на протоколе *L2F*, который был создан компанией Cisco Systems, как альтернатива протоколу *PPTP*. Протокол *L2TP* был создан как протокол защищенного туннелирования PPP-трафика через сети с произвольной средой. Этот протокол не привязан к протоколу IP, а поэтому может работать в сетях ATM (сети с асинхронным режимом транспортировки) или же в сетях с ретрансляцией кадров. Архитектура протокола видна на рисунке

Рисунок 2. Архитектура протокола L2TP

Соединение реализуется в 3 этапа:

1. этап: производится соединение с сервером удаленного доступа локальной сети. Пользователь создает PPP-соединение с провайдером ISP. Концентратор доступа LAC

устанавливает соединение, и создает канал PPP. Также концентратор выполняет аутентификацию пользователя и конечного узла. На основе имени клиента, провайдер ISP решает, нужно ли ему туннель на основе L2TP, если нужно — создается туннель.
2. этап: сервер LSN локальной сети реализует аутентификацию пользователя. Для этого может быть использован любой протокол аутентификации клиента.
3. этап: при успешной аутентификации, создается защищенный туннель между концентратором доступа LAC и сервером LNS локальной сети.

Протокол L2TP работает поверх любого транспорта с коммуникацией пакетов. Также L2TP не определяет конкретные методы криптозащиты.

Заключение

В данной статье были рассмотрены основные протоколы защиты информации в сети. Из всего вышеописанного можно сделать вывод, что для любой цели можно подобрать протокол защиты так, чтобы информация сохраняла конфиденциальность, целостность и доступность

Список литературы:

1. Протоколы сетевой безопасности ssh, ssl, tls, smtp, l2f, ipsec, l2tp, pptp, socks. // ЗИ URL: http://infoprotect.net/protect_network/protokolyi-ssh-ssl-smtp-ipsec-l2tp-pptp-socks (дата обращения 22.05.2022)

2. Сетевая безопасность // Tutorialspoint URL: https://translated.turbopages.org/proxy_u/en-ru.ru.6344e6e8-62947df0-5d6cd101-74722d776562/https/www.tutorialspoint.com/network_security/network_security_quick_guide.htm (дата обращения 22.05.2022)

3. TCP/IP — сетевая модель передачи данных // Википедия URL: https://ru.wikipedia.org/wiki/TCP/IP (дата обращения 22.05.2022)