Безопасное сетевое взаимодействие

Протокол TLS/SSL

- Основная функция протокола *TLS* состоит в обеспечении защиты и целостности данных между двумя взаимодействующими приложениями, одно из которых является клиентом, а другое – сервером.
- Протокол *TLS* (Transport Layer Security) разрабатывался на основе спецификации протокола *SSL* 3.0 (Secure Socket Layer), опубликованного корпорацией Netscape.
- Различия между данным протоколом и *SSL* 3.0 несущественны, но важно заметить, что *TLS* 1.0 и *SSL* 3.0 несовместимы, хотя в *TLS* 1.0 предусмотрен механизм, который позволяет реализациям *TLS* иметь обратную совместимость с *SSL* 3.0.

задачи протокола *TLS*

- Криптографическая безопасность: *TLS* должен использоваться для установления безопасного соединения между двумя участниками.
- Интероперабельность: независимые разработчики могут создавать приложения, которые будут взаимодействовать по протоколу *TLS*, что позволит устанавливать безопасные соединения.

задачи протокола *TLS*

- Расширяемость: *TLS* формирует общий каркас, в который могут быть встроены новые алгоритмы открытого ключа и симметричного шифрования.
- Это также избавляет от необходимости создавать новый протокол, что сопряжено с опасностью появления новых слабых мест, и предотвращает необходимость полностью реализовывать новую библиотеку безопасности.

задачи протокола *TLS*

- Относительная эффективность: криптографические операции интенсивно используют ЦП, особенно операции с открытым ключом.
- Для этого вводится понятие сессии, для которой определяются алгоритмы и их параметры.
- В рамках одной сессии может быть создано несколько соединений (например, TCP). *TLS* позволяет кэшировать сессии для уменьшения количества выполняемых действий при установлении соединения.
- Это снижает нагрузку как на ЦП, так и на трафик.

протокола *TLS*

- Протокол состоит из двух уровней.
- Нижним уровнем, расположенным выше некоторого надежного протокола (а именно, протокола TCP) является *протокол Записи*.
- *Протокол Записи* обеспечивает безопасность соединения

Свойства *Протокола Записи*

- Конфиденциальность соединения.
- Для защиты данных используется один из алгоритмов симметричного шифрования.
- Ключ для этого алгоритма создается для каждой сессии и основан на секрете, о котором договариваются в *протоколе Рукопожатия*.
- *Протокол Записи* также может использоваться без шифрования.

Свойства *Протокола Записи*

- Целостность соединения.
- Обеспечивается проверка целостности сообщения с помощью MAC с ключом.
- Для вычисления MAC используются безопасные *хэш-функции SHA*-1 и MD5.
- *Протокол Записи* может выполняться без вычисления MAC, но обычно функционирует в этом режиме.

Протокол TLS/SSL

- *Протокол Записи* используется для инкапсуляции различных протоколов более высокого уровня.
- Одним из протоколов более высокого уровня является *протокол Рукопожатия*, который использует *протокол Записи* в качестве транспорта для ведения переговоров о параметрах безопасности.
- *Протокол Рукопожатия* позволяет серверу и клиенту аутентифицировать друг друга и договориться об алгоритмах шифрования и криптографических ключах до того, как прикладной протокол, выполняющийся на том же уровне, начнет передавать или принимать первые байты данных.

Протокол Рукопожатия

- *Протокол Рукопожатия* обеспечивает безопасность соединения

Свойства *Протокола Рукопожатия*

- Участники аутентифицированы с использованием криптографии с открытым ключом (т.е. с использованием алгоритмов RSA, DSS и т.д.). Эта аутентификация может быть необязательной, но обычно требуется по крайней мере для сервера.
- Переговоры о разделяемом секрете безопасны, т.е. этот общий секрет невозможно подсмотреть.
- Переговоры о разделяемом секрете надежны, если выполнена аутентификация хотя бы одной из сторон. В таком случае атакующий, расположенный в середине соединения, не может модифицировать передаваемый секрет

Протокол TLS/SSL

- Одно из преимуществ *TLS* состоит в том, что он независим от прикладного протокола.
- Протоколы более высокого уровня могут прозрачно располагаться выше протокола *TLS*.

Элементы протокола

- Криптографические операции
- HMAC и псевдослучайная функция

Криптографические операции

- Определены четыре криптографические операции: цифровая подпись, поточное шифрование, *блочное шифрование* и шифрование с открытым ключом.
- В операции цифровой подписи входом в алгоритм подписи является результат применения односторонней хэш-функции к подписываемым данным. Длина входа определяется алгоритмом подписи.
- При использовании алгоритма RSA подписывается 36-байтная структура, состоящая из конкатенации 20 байтов хэш-кода SHA-1 и 16 байтов хэш-кода MD5.
- При использовании DSS 20 байтов хэш-кода SHA-1 подаются на вход алгоритму DSA без дополнительного хэширования. При этом создается два значения: r и s.

Криптографические операции

- При поточном шифровании для незашифрованного текста выполняется операция XOR с тем же количеством значений, созданных криптографически безопасным (с ключом) генератором псевдослучайных чисел.
- При блочном шифровании каждый блок незашифрованного текста шифруется, в результате чего создается блок зашифрованного текста. Все алгоритмы блочного шифрования выполняются в режиме CBC, и длина всех шифруемых элементов должна быть кратна длине блока алгоритма шифрования.
- При шифровании с открытым ключом используется алгоритм открытого ключа, при этом данные могут быть дешифрованы только с помощью соответствующего закрытого ключа.

HMAC и псевдослучайная функция

- Для получения MAC используется HMAC, поэтому если не знать секрета MAC, подделать MAC невозможно.
- HMAC может использоваться с различными хэш-алгоритмами. *TLS* задействует при Рукопожатии два алгоритма, MD5 и SHA-1, обозначаемых какHMAC_MD5 (secret, data) и HMAC_SHA (secret, data). Могут быть определены дополнительные хэш-алгоритмы, но в настоящей версии используются только MD5 и SHA-1.
- В алгоритме определена функция, которая расширяет секрет до нужной длины для создания всех необходимых ключей. Такая псевдослучайная функция, *PRF* , получает в качестве входа секрет, "зерно" (seed – значение, которое с одной стороны является случайным, а с другой стороны не является секретным, т.е. может стать известно оппоненту) и идентификационную метку, и создает выход требуемой длины.
- Для того чтобы сделать *PRF* как можно более безопасной, используются два безопасных хэш-алгоритма.

Протокол Записи

- *Протокол Записи* состоит из нескольких уровней. На каждом уровне сообщения могут включать поля длины, описания и содержимого.
- *Протокол Записи* фрагментирует сообщение на блоки нужной длины, осуществляет сжатие данных, применяет MAC и зашифровывает их, после чего результат передается по сети.
- На другом конце соединения полученные данные дешифруются, проверяется их целостность, далее они декомпрессируются, дефрагментируются и передаются протоколам более высокого уровня.

Протокол Записи

- Выше *протокола Записи* могут располагаться следующие протоколы: *протокол Рукопожатия,* Alert-протокол, протокол изменения шифрования и протокол прикладных данных.
- Для того чтобы иметь возможность расширения протокола *TLS*, *протокол Записи* допускает создание новых типов записей.
- Если реализация *TLS* получает тип записи, который она не понимает, она просто игнорирует его.

Состояния соединения

- Существует четыре *состояния соединения*: текущие состояния чтения и записи и ожидаемые состояния чтения и записи.
- Параметры безопасности для ожидаемых состояний устанавливаются *протоколом Рукопожатия,* а протокол Изменения шифрования может делать ожидаемое состояние текущим, при этом соответствующее текущее состояние сбрасывается и заменяется ожидаемым.
- Ожидаемое состояние в этом случае инициализируется в пустое состояние.
- Разработчики не должны допускать возможности сделать текущим состояние, которое не было инициализировано параметрами безопасности.
- Начальное текущее состояние всегда определяется без использования шифрования, сжатия и MAC.

Протокол Рукопожатия TLS

- *Протокол Рукопожатия* состоит из трех протоколов, использование которых позволяет участникам согласовать параметры безопасности для *протокола Записи*, аутентифицировать друг друга, договориться о параметрах безопасности и сообщить друг другу о возникновении тех или иных ошибок.

Протокол изменения шифрования

- Протокол состоит из единственного сообщения, которое зашифровано и сжато, как определено в текущем (не ожидаемом) *состоянии соединения*

Протокол изменения шифрования

- Сообщение об изменении шифрования может посылаться как клиентом, так и сервером для уведомления получающей стороны о том, что следующие записи будут защищены алгоритмами и ключами, о которых стороны только что договорились.
- При поступлении данного сообщения получатель должен информировать *протокол Записи* о немедленном копировании ожидаемого состояния чтения в текущее состояние чтения.

Протокол изменения шифрования

- Сразу после посылки данного сообщения отправитель должен информировать *протокол Записи* на своем конце соединения о немедленном копировании ожидаемого состояния записи в текущее состояние записи.
- Сообщение об изменении шифрования посылается при *Рукопожатии*после того, как параметры безопасности согласованы, но перед тем как посылается заключительное верифицирующее сообщение.

Alert протокол

- Содержимым протокола является либо фатальное, либо предупреждающее сообщение. Фатальное сообщение должно приводить к немедленному разрыву данного соединения.
- В этом случае другие соединения, соответствующие данной сессии, могут быть продолжены, но *идентификатор сессии* должен быть сделан недействительным для предотвращения использования данной сессии для установления новых соединений.
- Подобно другим сообщениям, сообщения Alert зашифрованы и сжаты, как определено в текущем *состоянии соединения*

Alert протокол

- Клиент и сервер должны оба узнать о том, что соединение завершается. Каждый участник может инициировать обмен сообщениями закрытия.
- Сообщение close_notify уведомляет получателя о том, что отправитель не будет больше посылать никаких сообщений по данному соединению.
- Сессия становится невозобновляемой, если хотя бы одно соединение завершено без соответствующего предупреждающего сообщения close_notify.

Alert протокол

- Каждый участник может инициировать закрытие посылкой сообщения Alert типа close_notify. Любые данные, отправленные после Alert-закрытия, игнорируются.
- Требуется, чтобы каждый участник посылал close_notify Alert перед закрытием стороны записи соединения.
- Это означает, что при получении ответа другого участника с Alert типа close_notify соединение немедленно закрывается и все ожидаемые состояния сбрасываются.
- Инициатору закрытия не обязательно ждать ответного close_notify Alert перед закрытием стороны чтения соединения.

Архитектура безопасности для IP

IPsec

- ***IPsec*** предназначен для безопасного взаимодействия на основе криптографии для IPv4 и IPv6.
- Набор сервисов безопасности включает управление доступом, целостность соединения, аутентификацию исходных данных, защиту от *replay-атак* (целостность последовательности), конфиденциальность (шифрование) и конфиденциальный поток трафика.
- Эти сервисы предоставляются на уровне IP, обеспечивая защиту для IP и/или протоколов более высокого уровня.

IPsec

- *IPsec* поддерживает две формы целостности: *целостность* соединения и частичную *целостность* последовательности.
- *Целостность* соединения является сервисом безопасности, который определяет модификацию конкретной *IP датаграммы*, безотносительно последовательности *датаграмм* в потоке трафика.
- Частичная *целостность* последовательности является anti-reply сервисом, с помощью которого определяется получение дубликатов *IP датаграмм*.
- Эти сервисы реализуются с использованием двух протоколов обеспечения безопасного трафика, *Authentication Header* (*AH*) и *Encapsulating Security Payload* (*ESP*), и с помощью процедур и протоколов *управления криптографическим ключом*. Множество применяемых *IPsec*протоколов и метод их использования определяются требованиями безопасности.

IPsec

- *IPsec* выполняется на хосте или шлюзе безопасности, обеспечивая защиту *IP*- трафика.
- Термин "*шлюз* безопасности" используется для обозначения промежуточной системы, которая реализует *IPsec* -протоколы.
- Защита основана на требованиях, определенных в *Базе Данных Политики Безопасности* (*Security Policy Database- SPD*), определяемой и поддерживаемой системным администратором.
- Пакеты обрабатываются одним из трех способов на основании соответствия информации заголовка *IP* или транспортного уровня записям в *SPD*.
- Каждый пакет либо отбрасывается сервисом безопасности *IPsec*, либо пропускается без изменения, либо обрабатывается сервисом *IPsec* на основе применения определенной политики.
- *IPsec* обеспечивает сервисы безопасности на *IP*-уровне, выбирая нужные протоколы безопасности, определяя алгоритмы, используемые сервисами, и предоставляя все криптографические ключи требуемым сервисам. *IPsec* может использоваться для защиты одного или нескольких "путей" между парой хостов, между парой шлюзов безопасности или между шлюзом безопасности и хостом.

IPsec

- *IPsec* использует два протокола для обеспечения безопасности трафика – Authentication Header (*AH*) и *Encapsulating Security Payload* (*ESP*).

IPsec

- Authentication Header (*AH*) обеспечивает целостность соединения, аутентификацию исходных данных и дополнительно может обеспечивать anti-replay сервис.
- *Encapsulating Security Payload* (*ESP*) протокол может обеспечивать конфиденциальность (шифрование) трафика. *ESP* также может обеспечивать целостность соединения, аутентификацию исходных данных и дополнительно anti-replay сервис.
- Целостность обеспечивается только для протоколов более высокого уровня. Хотя бы один из этих сервисов должен быть задействован при использовании *ESP*.

Authentication Header
Encapsulating Security Payload

- Эти протоколы могут применяться как по отдельности так и в комбинации с друг другом для обеспечения необходимого набора сервисов безопасности в IPv4 и IPv6.
- Каждый протокол поддерживает два режима использования: режим транспорта и режим туннелирования.
- В транспортном режиме протоколы обеспечивают защиту главным образом для протоколов более высокого уровня; в режиме туннелирования протоколы применяются для скрытия IP-заголовков *исходных пакетов*.

IPsec

- *IPsec* позволяет системному администратору управлять детализацией, с которой предоставляется сервис безопасности.
- Например, можно создать единственный зашифрованный туннель между двумя безопасными шлюзами, или для каждого TCP соединения может быть создан зашифрованный туннель между парой хостов.

IPsec позволяет указывать следующие параметры

- Какие сервисы используются и в какой комбинации.
- Необходимый уровень детализации применяемой защиты.
- Алгоритмы, используемые для обеспечения безопасности на основе криптографии

IPsec

- Интеграция *IPsec* в конкретную реализацию IP. Это требует доступа к исходному коду IP и применимо как к хостам, так и к шлюзам безопасности.
- Bump-in-the-stack (BITS) реализации, где *IPsec* реализован "внизу" существующей реализации стека протоколов IP, между обычным IP и локальными сетевыми драйверами. Доступа к исходному коду стека IP в данном контексте не требуется, что делает такой подход пригодным для встраивания в существующие системы. Данный подход обычно реализуется на хостах.
- Использование внешнего криптопроцессора (обычно в военных и в некоторых коммерческих системах). Как правило, это является Bump-in-the-stack (BITS) реализацией. Такие реализации могут использоваться как на хостах, так и на шлюзах. Обычно BITS-устройства являются IP-адресуемыми.

Безопасные Ассоциации

Security Association – SA

- ***SA*** есть симплексное (однонаправленное) логическое соединение, создаваемое для обеспечения безопасности.
- Весь трафик, передаваемый по*SA*, некоторым образом обрабатывается в целях обеспечения безопасности. И *AH*, и *ESP* используют в своей работе *SAs*. Одной из основных функций *IKE* является установление *SA*.

Security Association – SA

- ***SA*** есть совокупность параметров соединения, которые дают возможность сервисам обеспечивать безопасный трафик. *SA* определяет использование *AH* или *ESP*. Если к потоку трафика применяются оба протокола, *AH* и *ESP*, то создаются две *SA* s. При двунаправленном соединении между двумя хостами или между двумя шлюзами безопасности требуется два *SA* (по одному на каждое направление).
- *SA* однозначно определяется тройкой, состоящей из Security Parameter Index (SPI), IP *Destination Address* (адресом назначения) и идентификатора протокола безопасности (*AH* или *ESP*). В принципе адрес назначения может быть единственным адресом, широковещательным (broadcast) адресом или групповым (multicast) адресом. Однако механизм управления *SA* в настоящее время определяется только для единственной *SA*. Следовательно, *SAs* будут описаны в контексте point-to-point соединения, даже если концепция также применяется в случае point-to-multipoint.

Security Association – SA

- Определены два режима *SA*: режим транспорта и режим туннелирования.
- *Транспортный режим SA* обеспечивает безопасную связь между двумя хостами.
- В IPv4 заголовок протокола безопасности транспортного режима появляется сразу после IP заголовка и всех опций и перед любыми протоколами более высокого уровня (TCP или UDP).
- В случае *ESP транспортный режим SA* обеспечивает сервисы безопасности только для протоколов более высокого уровня, но не для IP-заголовка. В случае *AH* защита также распространяется на отдельные части IP-заголовка.

Security Association – SA

- Другим режимом *SA* является режим туннелирования. Если хотя бы одним из концов соединения является шлюз безопасности, то *SA* обязательно должна выполняться в туннелирующем режиме.
- *SA* между двумя шлюзами безопасности всегда находится в туннелирующем режиме, так же, как и *SA* между хостом и шлюзом безопасности. Заметим, что когда трафик предназначен для шлюза безопасности, например, в случае SNMP-команд, шлюз безопасности рассматривается как хост, и допустим *транспортный режим*.
- Два хоста могут при желании так же устанавливать туннелирующий режим.

Security Association – SA

- В туннелирующем режиме *SA* существует "внешний" IP заголовок, который определяет пункт назначения *IPsec*, и "внутренний" IP заголовок, который определяет конечный пункт назначения для пакета.
- Заголовок протокола безопасности расположен после внешнего IP заголовка и перед внутренним IP заголовком.
- Если *AH* используется в туннелирующем режиме, части внешнего IP заголовка являются защищенными, как и весь туннелируемый IP пакет, т. е. все внутренние заголовки защищены, как и все протоколы более высокого уровня.
- Если применяется *ESP*, защита обеспечивается только для туннелируемого пакета, а не для внешнего IP-заголовка.

Security Association – SA

- Набор реализуемых *SA* сервисов безопасности зависит от выбранного протокола безопасности, режима *SA*, конечных точек *SA* и выбора дополнительных сервисов в протоколе.
- Например, *AH* обеспечивает аутентификацию исходных данных и целостность соединения для Ip*датаграмм*.
- "Точность" *сервиса аутентификации* является функцией от степени детализованности *SA*, для которой используется *AH*.

Authentication Header (*AH*)

- *AH* также предоставляет анти-replay сервис (целостность отдельной последовательности) для получателя, помогая предотвратить атаки отказа в сервисе.
- *AH* применяется, когда не требуется конфиденциальность. *AH* также обеспечивает аутентификацию отдельных частей IP заголовка, за исключением изменяющихся частей IP заголовка.

Encapsulating Security Payload (*ESP*).

- *SP* обеспечивает *конфиденциальность трафика*.
- Сила сервиса конфиденциальности зависит от используемого алгоритма шифрования.
- *ESP* также может дополнительно обеспечивать аутентификацию. Область аутентификации, обеспечиваемая *ESP*, является более узкой по сравнению с *AH*, т.е. IP-заголовок (заголовки), "внешние" по отношению к *ESP* заголовку, не защищены.
- Если аутентификация нужна только протоколам более высокого уровня, то аутентификация *ESP* является подходящей альтернативой, причем более эффективной, чем использование *AH*, инкапсулирующего *ESP*.
- Если для *ESP SA* используется аутентификация, получатель также может выбрать усиление использованием анти-replay сервиса с теми же самыми возможностями, что и *AH* анти-replay сервис.
- Хотя и конфиденциальность, и аутентификация являются необязательными, оба сервиса не могут быть опущены. По крайней мере, один из них должен присутствовать.

Базы данных безопасной ассоциации

- Многие детали, связанные с обработкой IP-трафика в реализации *IPsec* не являются предметом стандартизации.
- Тем не менее, некоторые внешние аспекты обработки должны быть стандартизованы для обеспечения интероперабельности *IPsec*.
- Внешнее поведение каждой реализации должно соответствовать характеристикам данной модели.

Базы данных безопасной ассоциации

- Существуют две БД: БД Политики Безопасности (*SPD*) и БД *Безопасной Ассоциации* (*SAD*).
- Первая описывает политики, которые определяют характер обработки всего IP трафика.
- Вторая БД содержит параметры, которые связаны с каждой активной *безопасной ассоциацией*.
- Селектор как множество значений полей IP протокола и протокола более высокого уровня, которые используются БД Политики Безопасности для отображения трафика на *SA*.
- Каждый сетевой интерфейс, для которого необходима обработка *IPsec*, требует определения баз данных для входящего и

База данных Безопасной Ассоциации

- Для входящей обработки следующие *поля пакета* используются для поиска *SA* в *SAD*:
- IP адрес назначения внешнего заголовка: IPv4 или IPv6 адрес назначения.
- Протокол *IPsec*: *AH* или *ESP*, используемый в качестве индекса *SA* в данной БД. Определяет протокол *IPsec*, применяемый к трафику для данной *SA*.
- SPI: 32-битное значение, применяемое для идентификации различных *SA*, заканчивающихся одним и тем же адресом назначения и использующих один и тот же *IPsec* протокол.

поля *SAD* для *IPsec* -обработки

- Sequence Number Counter: 32-битное значение, используемое для создания поля Sequence Number в *AH* или *ESP* заголовках (используется только для исходящего трафика).
- Sequence Number Overflow: флаг, указывающий, было ли переполнение Sequence Number Counter, должен вызывать событие аудита и предотвращать передачу дополнительных пакетов по данной *SA* (используется только для исходящего трафика).
- Anti-Replay Window: 32-битный счетчик или битовая карта (или некий эквивалент), используемые для проверки, является ли входящий *AH*или *ESP* пакет повтором. (Используется только для входящего трафика. Замечание: если anti-reply сервис не используется получателем, например, в случае ручных ключей *SA*, когда anti-reply window не используется.)
- Алгоритм аутентификации для *AH*, ключи и т.д.
- Алгоритм шифрования для *ESP*, ключи, режим, IV и т.д.
- Алгоритм аутентификации для *ESP*, ключи и т.д. Если

поля *SAD* для *IPsec* -обработки

- Время жизни данной *SA*: интервал времени, после которого *SA* должна быть заменена новой *SA* (и новым SPI) или завершение *SA*, а также определения того, какое из этих действий должно выполняться.
- Это может быть выражено в виде времени или количества байтов, или и того, и другого одновременно.
- Реализации должны поддерживать оба типа времени жизни и одновременное применение обоих типов. Если используется время и если *IKE* задействует сертификаты X.509 для установления *SA*, то время жизни *SA* должно входить в допустимый интервал для сертификатов. В этом смысле как инициатор, так и получатель ответственны за установление корректного времени жизни *SA*.
-

SA и Управление Ключом

- *IPsec* поддерживает как ручные, так и автоматически созданные *SA* и соответствующее *управление криптографическими ключами*.
- Протоколы *AH* и *ESP* практически не зависят от используемых технологий управления ключом, хотя эти технологии могут некоторым образом влиять на сервисы безопасности, предоставляемые протоколами.
- Например, дополнительные anti-replay сервисы требуют автоматического управления *SA*. Более того, детализированность используемого распределения ключа определяет детализированность предоставляемой аутентификации.

Ручные технологии

- Простейшей формой управления является ручное управление, при котором администратор вручную конфигурирует каждую систему материалом ключа и данными управления *безопасной ассоциацией*.
- Ручные технологии применяются в маленьких, статичных окружениях, и они не масштабируются.
- Например, компания может создать VPN, используя *IPsec* на хостах. Если количество хостов мало, и если все хосты расположены в пределах одного административного домена, то возможно применение ручных технологий управления. В данном случае хост должен выборочно защищать трафик и от других хостов в организации, используя вручную сконфигурированные ключи, допуская незащищенный трафик для других получателей.
- Данные технологии можно задействовать и в том случае, когда только выборочные коммуникации должны быть безопасны.
- Аналогичный аргумент может быть применен для использования *IPsec* в организации с небольшим числом хостов и/или шлюзов.

Автоматические SA и Управление Ключом

- Широкое использование *IPsec* требует стандартного для Internet, масштабируемого, автоматического протокола управления *SA*. Такая поддержка необходима для использования anti-replay возможностей *AH* и *ESP* и для возможности создания *SAs*.
- Протоколом автоматического управления ключом по умолчанию является *IKE*, но могут быть реализованы и другие протоколы автоматического управления ключом.

- Использование *IPsec* навязывает высокую вычислительную *стоимость* на хостах и шлюзах безопасности, которые реализуют эти протоколы.
- Эта цена связана с памятью, необходимой для структур данных *IPsec*, *вычисление* значений проверки целостности, *шифрование* и *дешифрование*, а также дополнительное управление пакетом.
- Использование протоколов управления *SA* / ключом, особенно тех, которые реализуют криптографию с открытым ключом, также добавляет соответствующую вычислительную *стоимость* в использование *IPsec*.
-

- Использование *IPsec* также увеличивает *стоимость* компонентов, осуществляющих пересылку и роутинг в инфраструктуре *Internet*, но не реализующих *IPsec*.
- Это происходит из-за возрастания размера пакета в результате добавления заголовков *AH* и/или *ESP*, *AH* и *ESP*туннелиро вания (который добавляет второй *IP*-заголовок) и возрастании трафика, связанного с протоколами управления ключом.

Безопасную Ассоциацию Internet и Протокол Управления Ключом (***ISAKMP***)

- *ISAKMP* обеспечивает полную безопасность последующих обменов (Perfect Forward *Secrecy* – *PFS*) – это означает, что при компрометации одного ключа возможен только доступ к данным, защищенным одним этим ключом.
- При *PFS* ключ, используемый для защиты передаваемых данных, не должен использоваться для получения любых дополнительных ключей, и если ключ, используемый для защиты передаваемых данных, был получен из некоторого другого ключевого материала, то этот ключевой материал не должен больше использоваться для получения других ключей.

- *SAKMP* обеспечивает аутентифицированный обмен ключа. *ISAKMP* не определяет конкретный алгоритм обмена ключа. Тем не менее, как правило, вместе с *ISAKMP* используется протокол *IKE*.
- Защита от DoS-атак является одной из наиболее трудных задач. Для этой цели в *ISAKMP* используются "Cookie" или знак анти-препятствия (anti-clogging token – ACT), которые предназначены для защиты вычислительных ресурсов от подобной атаки без расходования собственных ресурсов на ее обнаружение.
- Абсолютная защита от отказа в сервисе невозможна, но такой знак анти-препятствия предоставляет технологию для того, чтобы сделать защиту более надежной.

ISAKMP

- *ISAKMP* предотвращает создание соединения с атакующим, объединяя аутентификацию, обмен ключа и создание SA. Это объединение не позволяет злоумышленнику дождаться завершения аутентификации и затем осуществить имперсонизацию в одну из аутентифицированных сущностей.

ISAKMP

- Атаки man-in-the-middle включают перехват, вставку, уничтожение и модификацию сообщений, отправку сообщений назад отправителю, повтор старых сообщений и перенаправление сообщений. *ISAKMP* предупреждает все эти типы атак.
- Объединение сообщений *ISAKMP*защищает от возможности встроить сообщения в обмены протокола.
- Протокол *ISAKMP* позволяет хосту обнаружить уничтоженные сообщения. Требование наличия нового cookie с новой отметкой времени для каждого нового установления SA предотвращает атаки, которые включают повтор старых сообщений.
- Требование *ISAKMP сильной аутентификации* предотвращает установление SA с кем-то, кроме заданного участника. Сообщения можно перенаправить к другому получателю или модифицировать, но это будет обнаружено, и SA установлена не будет.

Фазы переговоров

- *ISAKMP* предполагает две *фазы переговоров*. Во время первой фазы две сущности (*ISAKMP* -серверы) договариваются о том, как защищать дальнейший трафик переговоров, устанавливая *ISAKMP* SA. Эта *ISAKMP* SA затем используется для защиты переговоров о требуемой SA.
- Вторая *фаза переговоров* используется для установления SA для других протоколов безопасности. Эта вторая фаза может применяться для установления нескольких безопасных ассоциаций.
-

ISAKMP

- Хотя подход, основанный на двух фазах, является достаточно дорогостоящим для большинства простых сценариев, существует несколько причин, чтобы он оказывался в большинстве случаев предпочтительным.

ISAKMP

- Во-первых, *ISAKMP* серверы могут уменьшить время установления первой фазы до нескольких секунд. Это позволяет устанавливать несколько SAs между двумя участниками за одно и то же время с начала соединения.
- Во-вторых, сервисы безопасности, которые ведут переговоры во время первой фазы, предоставляют свойства безопасности для второй фазы. Например, после первой *фазы переговоров* шифрование, предоставляемое *ISAKMP* SA, может обеспечивать защиту идентификации, потенциально допуская возможность применения более простых обменов во второй фазе. С другой стороны, если канал, устанавливаемый в течение первой фазы, адекватно не защищает идентификации, вторая фаза должна вести переговоры, учитывая это.

ISAKMP

- Заметим, что для каждой *фазы переговоров* могут применяться различные сервисы безопасности.
- Например, разные участники осуществляют аутентификацию в течение каждой *фазы переговоров*.
- На первой фазе участниками, осуществляющими аутентификацию, могут быть *ISAKMP*серверы или хосты, в то время как на второй фазе аутентификация осуществляется на уровне пользователей или прикладных программ.

Протокол безопасности:

- протокол безопасности состоит из записи в конкретной точке стека сетевых протоколов, выполняющей сервис безопасности для сетевого соединения.
- Например, IPsec ESP и IPsec AH являются двумя различными протоколами безопасности.
- Протокол безопасности может выполнять более одного сервиса, например, обеспечивая целостность и конфиденциальность в одном модуле.

Набор защиты:

- набор защиты является списком сервисов безопасности, которые могут быть применены к различным протоколам безопасности.
- Например, набор защиты может состоять из DES шифрования для ESP и MD5 с ключом для AH.

Безопасная ассоциация (SA):

- *безопасная ассоциация* определяет протокол безопасности и конкретный набор параметров сервисов и механизмов, необходимых для защиты трафика.
- Эти параметры могут включать идентификаторы алгоритмов, режимы, криптографические ключи и т.д.
- SA ссылается на связанный с ней протокол безопасности (например, "*ISAKMP* SA", "ESP SA").