Протоколы аутентификации и обмена ключом

- Понятие мастер-ключа
- Протоколы аутентификации
  - Использование симметричного шифрования
    - Протокол Нидхэма и Шредера
    - Протокол Деннинга
    - Протокол аутентификации с использованием билета
  - Использование шифрования с открытым ключом
    - Протокол аутентификации с использованием аутентификационного сервера
    - Протокол аутентификации с использованием KDC

Понятие мастер-ключа

- При симметричном шифровании два участника, которые хотят обмениваться конфиденциальной информацией, должны иметь один и тот же ключ. Частота изменения ключа должна быть достаточно большой, чтобы у противника не хватило времени для полного перебора ключа. Следовательно, сила любой криптосистемы во многом зависит от технологии распределения ключа. Этот термин означает передачу ключа двум участникам, которые хотят обмениваться данными, таким способом, чтобы никто другой не мог ни подсмотреть, ни изменить этот ключ. Для двух участников A и B распределение ключа может быть выполнено одним из следующих способов.

1. Ключ может быть создан A и физически передан B.
2. Третья сторона может создать ключ и физически передать его A и B.
3. A и B имеют предварительно созданный и недолго используемый ключ, один участник может передать новый ключ другому, применив для шифрования старый ключ.
4. Если A и B каждый имеют безопасное соединение с третьим участником C, C может передать ключ по этому безопасному каналу A и B.

Понятие мастер-ключа

- Первый и второй способы называются ручным распределением ключа. Это самые надежные способы распределения ключа, однако во многих случаях пользоваться ими неудобно и даже невозможно. В распределенной системе любой хост или сервер должен иметь возможность обмениваться конфиденциальной информацией со многими аутентифицированными хостами и серверами. Таким образом, каждый хост должен иметь набор ключей, поддерживаемый динамически. Проблема особенно актуальна в больших распределенных системах.
- Количество требуемых ключей зависит от числа участников, которые должны взаимодействовать. Если выполняется шифрование на сетевом или IP-уровне, то ключ необходим для каждой пары хостов в сети. Таким образом, если есть N хостов, то необходимое число ключей [N (N – 1)]/2. Если шифрование выполняется на прикладном уровне, то ключ нужен для каждой пары прикладных процессов, которых гораздо больше, чем хостов.

Понятие мастер-ключа

- Третий способ распределения ключей может применяться на любом уровне стека протоколов, но если атакующий получает возможность доступа к одному ключу, то вся последовательность ключей будет раскрыта. Более того, все равно должно быть проведено первоначальное распространение большого количества ключей.
- Поэтому в больших автоматизированных системах широко применяются различные варианты четвертого способа. В этой схеме предполагается существование так называемого центра распределения ключей (Key Distribution Center - KDC), который отвечает за распределение ключей для хостов, процессов и приложений. Каждый участник должен разделять уникальный ключ с KDC.
- Использование центра распределения ключей основано на использовании иерархии ключей. Как минимум используется два типа ключей: мастер-ключи и ключи сессии.

Понятие мастер-ключа

- Для обеспечения конфиденциальной связи между конечными системами используется временный ключ, называемый ключом сессии. Обычно ключ сессии используется для шифрования транспортного соединения и затем уничтожается. Каждый ключ сессии должен быть получен по сети из центра распределения ключей. Ключи сессии передаются в зашифрованном виде, используя мастер-ключ, который разделяется между центром распределения ключей и конечной системой.
- Эти мастер-ключи также должны распределяться некоторым безопасным способом. Однако при этом существенно уменьшается количество ключей, требующих ручного распределения. Если существует N участников, которые хотят устанавливать соединения, то в каждый момент времени необходимо [N (N – 1)]/2 ключей сессии. Но требуется только N мастер-ключей, по одному для каждого участника.

Понятие мастер-ключа

- Время жизни ключа сессии как правило равно времени жизни самой сессии.
- Чем чаще меняются ключи сессии, тем более безопасными они являются, так как противник имеет меньше времени для взламывания данного ключа сессии. С другой стороны, распределение ключей сессии задерживает начало любого обмена и загружает сеть. Политика безопасности должна сбалансировать эти условия для определения оптимального времени жизни конкретного ключа сессии.
- Если соединение имеет долгое время жизни, то должна существовать возможность периодически менять ключ сессии.

Понятие мастер-ключа

- Для протоколов, не поддерживающих соединение, таких как протокол, ориентированный на транзакции, нет явной инициализации или прерывания соединения. Следовательно, неясно, как часто надо менять ключ сессии. Большинство подходов основывается на использовании нового ключа сессии для каждого нового обмена. Наиболее часто применяется стратегия использования ключа сессии только для фиксированного периода времени или только для определенного количества транзакций.

Протоколы аутентификации

- Рассмотрим основные протоколы, обеспечивающие как взаимную аутентификацию участников, так и аутентификацию только одного из участников.

Взаимная аутентификация

- Здесь протоколы применяются для взаимной аутентификации участников и для обмена ключом сессии.
- Основной задачей таких протоколов является обеспечение конфиденциального распределения ключа сессии и гарантирование его своевременности, то есть протокол не должен допускать повторного использования старого ключа сессии. Для обеспечения конфиденциальности ключи сессии должны передаваться в зашифрованном виде. Вторая задача, обеспечение своевременности, важна, потому что существует угроза перехвата передаваемого сообщения и повторной его пересылки. Такие повторения в худшем случае могут позволять взломщику использовать скомпрометированный ключ сессии, при этом успешно подделываясь под другого участника. Успешное повторение может, как минимум разорвать операцию аутентификации участников.

Протоколы аутентификации

Такие повторы называются replay-атаками. Рассмотрим возможные примеры подобных replay-атак:

1. *Простое повторение:* противник просто копирует сообщение и повторяет его позднее.
2. *Повторение, которое не может быть определено:* противник уничтожает исходное сообщение и посылает скопированное ранее сообщение.

Один из возможных подходов для предотвращения replay-атак мог бы состоять в присоединении последовательного номера (sequence number) к каждому сообщению, используемому в аутентификационном обмене. Новое сообщение принимается только тогда, когда его последовательный номер правильный. Трудность данного подхода состоит в том, что каждому участнику требуется поддерживать значения sequence number для каждого участника, с которым он взаимодействует в данный момент.

Протоколы аутентификации

Поэтому обычно sequence number не используются для аутентификации и обмена ключами. Вместо этого применяется один из следующих способов:

1. *Отметки времени:* участник А принимает сообщение как не устаревшее только в том случае, если оно содержит отметку времени, которая, по мнению А, соответствует текущему времени. Этот подход требует, чтобы часы всех участников были синхронизированы.
2. *Запрос/ответ:* участник А посылает в запросе к В случайное число (nonce - number only once) и проверяет, чтобы ответ от В содержал корректное значение этого nonce.

Протоколы аутентификации

- Считается, что подход с отметкой времени не следует использовать в приложениях, ориентированных на соединение, потому что это технически трудно, так как таким протоколам, кроме поддержки соединения, необходимо будет поддерживать синхронизацию часов различных процессоров. При этом возможный способ осуществления успешной атаки может возникнуть, если временно будет отсутствовать синхронизация часов одного из участников. В результате различной и непредсказуемой природы сетевых задержек распределенные часы не могут поддерживать точную синхронизацию. Следовательно, процедуры, основанные на любых отметках времени, должны допускать окно времени, достаточно большое для приспособления к сетевым задержкам, и достаточно маленькое для минимизации возможности атак.
- С другой стороны, подход запрос/ответ не годится для приложений, не устанавливающих соединения, так как он требует предварительного рукопожатия перед началом передач, тем самым отвергая основное свойство транзакции без установления соединения. Для таких приложений доверие к некоторому безопасному серверу часов и постоянные попытки каждой из частей синхронизировать свои часы с этим сервером может быть оптимальным подходом.

Использование симметричного шифрования

- Для обеспечения аутентификации и распределения ключа сессии часто используется двухуровневая иерархия ключей симметричного шифрования. В общих чертах эта стратегия включает использование доверенного центра распределения ключей (KDC). Каждый участник разделяет секретный ключ, называемый также мастер-ключом, с KDC. KDC отвечает за создание ключей, называемых ключами сессии, и за распределение этих ключей с использованием мастер-ключей. Ключи сессии применяются в течение короткого времени для шифрования только данной сессии между двумя участниками.
- Большинство алгоритмов распределения секретного ключа с использованием KDC, включает также возможность аутентификации участников.

Протокол Нидхэма и Шредера

- Предполагается, что секретные мастер-ключи K_A и K_B разделяют соответственно A и KDC и B и KDC. Целью протокола является безопасное распределение ключа сессии K_S между A и B. Протокол представляет собой следующую последовательность шагов:

1. **A → KDC:** $ID_A \| ID_B \| N_1$
2. **KDC → A:** $E_{Ka} [K_S \| ID_B \| N_1 \| E_{Kb} [K_S \| ID_A]]$
3. **A → B:** $E_{Kb} [K_S \| ID_A]$
4. **B → A:** $E_{Ks} [N_2]$
5. **A → B:** $E_{Ks} [f (N_2)]$

Протокол Нидхэма и Шредера

1. A запрашивает у KDC ключ сессии для установления защищенного соединения с B. Сообщение включает идентификацию A и B и уникальный идентификатор данной транзакции, который обозначен как N_1 и называется *nonce.* Nonce может быть временной меткой, счетчиком или случайным числом; минимальное требование состоит в том, чтобы он отличался для каждого запроса. Кроме того, для предотвращения подделки желательно, чтобы противнику было трудно предугадать nonce. Таким образом, случайное число является лучшим вариантом для nonce.
2. KDC отвечает сообщением, зашифрованным ключом K_A. Таким образом, только A может расшифровать сообщение, и A уверен, что оно получено от KDC, так как предполагается, что кроме A и KDC этот ключ не знает никто.

Протокол Нидхэма и Шредера

Это сообщение включает следующие элементы, предназначенные для A:

- Одноразовый ключ сессии.
- Идентификатор B.
- nonce, который идентифицирует данную сессию.

- A должен убедиться, что полученный nonce равен значению nonce из первого запроса. Это доказывает, что ответ от KDC не был модифицирован при пересылке и не является повтором некоторого предыдущего запроса. Кроме того, сообщение включает два элемента, предназначенные для B:

- Одноразовый ключ сессии K_S.
- Идентификатор A ID_A.

- Эти два последних элемента шифруются мастер-ключом, который KDC разделяет с B. Они посылаются B при установлении соединения и доказывают идентификацию A.

Протокол Нидхэма и Шредера

3. A сохраняет у себя ключ сессии и передает B информацию от KDC, предназначенную B: $E_{Kb}[K_S || ID_A]$. Так как эта информация зашифрована K_B, она защищена от просмотра. Теперь B знает ключ сессии (K_S), знает, что другим участником является A, (ID_A) и что начальная информация передана от KDC, т.к. она зашифрована с использованием K_B.

B этой точке ключ сессии безопасно передан от A к B, и они могут начать безопасный обмен.

Тем не менее, существует еще два дополнительных шага:

4. Используя созданный ключ сессии, B пересылает A nonce N_2.

5. Также используя K_S, A отвечает f (N_2), где f – функция, выполняющая некоторую модификацию N_2.

Эти шаги гарантируют B, что сообщение, которое он получил, не изменено и не является повтором предыдущего сообщения.

Протокол Нидхэма и Шредера

- Заметим, что реальное распределение ключа включает только шаги 1 – 3, а шаги 4 и 5, как и 3, выполняют функцию аутентификации.
- A безопасно получает ключ сессии на шаге 2. Сообщение на шаге 3 может быть дешифровано только B. Шаг 4 отражает знание B ключа K_S, и шаг 5 гарантирует B знание участником A ключа K_S и подтверждает, что это не устаревшее сообщение, так как используется nonce N_2. Шаги 4 и 5 призваны предотвратить общий тип replay-атак. В частности, если противник имеет возможность захватить сообщение на шаге 3 и повторить его, то это должно привести к разрыву соединения.

Протокол Нидхэма и Шредера

- Разрывая рукопожатие на шагах 4 и 5, протокол все еще уязвим для некоторых форм атак повторения. Предположим, что противник X имеет возможность скомпрометировать старый ключ сессии. Маловероятно, чтобы противник мог сделать больше, чем просто копировать сообщение шага 3. Потенциальный риск состоит в том, что X может заставить взаимодействовать A и B, используя старый ключ сессии. Для этого X просто повторяет сообщение шага 3, которое было перехвачено ранее и содержит скомпрометированный ключ сессии. Если B не запоминает идентификацию всех предыдущих ключей сессий с A, он не сможет определить, что это повтор. Далее X должен перехватить сообщение рукопожатия на шаге 4 и представиться A в ответе на шаге 5.

Протокол Деннинга

- Деннинг предложил преодолеть эту слабость модификацией протокола Нидхэма и Шредера, которая включает дополнительную отметку времени на шагах 2 и 3:

1. **A → KDC:** $ID_A \| ID_B$
2. **KDC → A:** $E_{Ka} [K_S \| ID_B \| T \| E_{Kb} [K_S \| ID_A \| T]]$
3. **A → B:** $E_{Kb} [K_S \| ID_A \| T]$
4. **B → A:** $E_{Ks} [N_1]$
5. **A → B:** $E_{Ks} [f (N_1)]$

Протокол Деннинга

- T – это отметка времени, которая гарантирует A и B, что ключ сессии является только что созданным. Таким образом, и A, и B знают, что распределенный ключ не является старым. A и B могут верифицировать временную отметку проверкой, что

$$|\text{Clock} - T| < \Delta t_1 + \Delta t_2$$

- где Δt_1 – оцениваемое нормальное расхождение между часами KDC и локальными часами (у A или B) и Δt_2 – ожидаемая сетевая задержка времени. Каждый участник может установить свои часы, ориентируясь на определенный доверенный источник. Поскольку временная отметка T шифруется с использованием секретных мастер-ключей, взломщик, даже зная старый ключ сессии, не сможет достигнуть цели повторением шага 3 так, чтобы B не заметил искажения времени.

Протокол Деннинга

- Шаги 4 и 5 не были включены в первоначальное представление, но были добавлены позднее. Эти шаги подтверждают A, что B получил ключ сессии.
- Протокол Деннинга обеспечивает большую степень безопасности по сравнению с протоколом Нидхэма и Шредера. Однако данная схема требует доверия к часам, которые должны быть синхронизированы в сети. В этом есть определенный риск, который состоит в том, что распределенные часы могут рассинхронизироваться в результате диверсии или повреждений. Проблема возникает, когда часы отправителя спешат по отношению к часам получателя. В этом случае противник может перехватить сообщение от отправителя и повторить его позднее, когда отметка времени в сообщении станет равной времени на узле получателя. Это повторение может иметь непредсказуемые последствия.
- Один способ вычисления атак повторения состоит в требовании, чтобы участники регулярно сверяли свои часы с часами KDC. Другая альтернатива, при которой нет необходимости всем синхронизировать часы, состоит в доверии протоколам рукопожатия, использующим nonce.

Протокол аутентификации с использованием билета

- Данный протокол пытается преодолеть проблемы, возникшие в предыдущих двух протоколах. Он выглядит следующим образом:

1. **A → B:** $ID_A \| N_A$
2. **B → KDC:** $ID_B \| N_B \| E_{Kb} [ID_A \| N_A \| T_B]$
3. **KDC → A:** $E_{Ka} [ID_B \| N_A \| K_S \| T_B] \| E_{Kb} [ID_A \| K_S \| T_B] \| N_B$
4. **A → B:** $E_{Kb} [ID_A \| K_S \| T_B] \| E_{Ks} [N_B]$

Протокол аутентификации с использованием билета

1. A инициализирует аутентификационный обмен созданием nonce N_a и посылкой его и своего идентификатора к B в незашифрованном виде. Этот nonce вернется к A в зашифрованном сообщении, включающем ключ сессии, гарантируя A, что ключ сессии не старый.
2. B сообщает KDC, что необходим ключ сессии. Это сообщение к KDC включает идентификатор B и nonce N_b. Данный nonce вернется к B в зашифрованном сообщении, которое включает ключ сессии, гарантируя B, что ключ сессии не устарел. Сообщение B к KDC также включает блок, зашифрованный секретным ключом, разделяемым B и KDC. Этот блок используется для указания KDC, когда заканчивается время жизни данного ключа сессии. Блок также специфицирует намеченного получателя и содержит nonce, полученный от A. Этот блок является своего рода «верительной грамотой» или «билетом» для A.

Протокол аутентификации с использованием билета

3. KDC получил nonces от A и B и блок, зашифрованный секретным ключом, который B разделяет с KDC. Блок служит билетом, который может быть использован A для последующих аутентификаций. KDC также посылает A блок, зашифрованный секретным ключом, разделяемым A и KDC. Этот блок доказывает, что B получил начальное сообщение A (ID_B), что в нем содержится допустимая отметка времени и нет повтора (N_a). Этот блок обеспечивает A ключом сессии (K_S) и устанавливает ограничение времени на его использование (T_b).
4. A посылает полученный билет B вместе с nonce B, зашифрованным ключом сессии. Этот билет обеспечивает B ключом сессии, который тот использует для дешифрования и проверки nonce. Тот факт, что nonce B расшифрован ключом сессии, доказывает, что сообщение пришло от A и не является повтором.

Протокол аутентификации с использованием билета

- Данный протокол аутентифицирует A и B и распределяет ключ сессии. Более того, протокол предоставляет в распоряжение A билет, который может использоваться для его последующей аутентификации, исключая необходимость повторных контактов с аутентификационным сервером. Предположим, что A и B установили сессию с использованием описанного выше протокола и затем завершили эту сессию. Впоследствии, но до истечения лимита времени, установленного протоколом, A может создать новую сессию с B. Используется следующий протокол:

1. **A → B:** $E_{Kb}\ [ID_A \| K_S \| T_B], N_A'$
2. **B → A:** $N_B', E_{Ks}\ [N_A']$
3. **A → B:** $E_{Ks}\ [N_B']$

Протокол аутентификации с использованием билета

- Когда B получает сообщение на шаге 1, он проверяет, что билет не просрочен. Заново созданные nonces N_a' и N_b' гарантируют каждому участнику, что не было атак повтора.
- Время T_b является временем относительно часов B. Таким образом, эта временная метка не требует синхронизации, потому что B проверяет только им самим созданные временные отметки.

Использование шифрования с открытым ключом

Протокол аутентификации с использованием аутентификационного сервера

Рассмотрим протокол, использующий отметки времени и аутентификационный сервер:

1. **A → AS:** $ID_A \| ID_B$
2. **AS → A:** $Sig_{KRas}[ID_A \| KU_A \| T] \| Sig_{KRas}[ID_B \| KU_B \| T]$
3. **A → B:** $Sig_{KRas}[ID_A \| KU_A \| T] \| Sig_{KRas}[ID_B \| KU_B \| T] \| E_{KUb}[Sig_{KRa}[K_S \| T]]$

- В данном случае третья доверенная сторона является просто аутентификационным сервером AS, потому что третья сторона не создает и не распределяет секретный ключ. AS просто обеспечивает сертификацию открытых ключей участников. Ключ сессии выбирается и шифруется A, следовательно, не существует риска, что AS взломают и заставят распределять скомпрометированные ключи сессии. Отметки времени защищают от повтора скомпрометированных ключе й сессии.
- Данный протокол компактный, но, как и прежде, требует синхронизации часов.

Протокол аутентификации с использованием KDC

- Другой подход использует nonces. Этот протокол состоит из следующих шагов:

1. **A → KDC:** $ID_A \| ID_B$
2. **KDC → A:** $Sig_{KRkdc} [ID_B \| KU_B]$
3. **A → B:** $E_{KUb} [N_A \| ID_A]$
4. **B → KDC:** $ID_B \| ID_A \| E_{KUkdc} [N_A]$
5. **KDC → B:** $Sig_{KRkdc} [ID_A \| KU_A] \| E_{KUb} [Sig_{KRkdc} [N_A \| K_S \| ID_B]]$
6. **B → A:** $E_{KUa} [Sig_{KRkdc} [N_A \| K_S \| ID_B] \| N_B]$
7. **A →B:** $E_{Ks} [N_B]$

Протокол аутентификации с использованием KDC

- На первом шаге A информирует KDC, что хочет установить безопасное соединение с B. KDC возвращает A сертификат открытого ключа B (шаг 2). Используя открытый ключ B, A информирует B о создании защищенного соединения и посылает nonce N_a (шаг 3). На 4-м шаге B спрашивает KDC о сертификате открытого ключа A и запрашивает ключ сессии. B включает nonce A, чтобы KDC мог пометить ключ сессии этим nonce. Nonce защищен использованием открытого ключа KDC. На 5-м шаге KDC возвращает B сертификат открытого ключа A плюс информацию $\{N_a, K_S, ID_B\}$. Эта информация означает, что K_S является секретным ключом, созданным KDC в интересах B и связан с N_a. Связывание K_S и N_a гарантирует A, что K_S не устарел. Эта тройка шифруется с использованием закрытого ключа KDC, это гарантирует B, что тройка действительно получена от KDC. Она также шифруется с использованием открытого ключа B, чтобы никто другой не мог подсмотреть ключ сессии и использовать эту тройку для установления соединения с A. На шаге 6 тройка $\{N_a, K_S, ID_B\}$, зашифрованная закрытым ключом KDC, передается A вместе с nonce N_b, созданным B. Все сообщение шифруется открытым ключом A. A восстанавливает ключ сессии K_S, использует его для шифрования N_b, который возвращает B. Это последнеее сообщение гарантирует B, что A знает ключ сессии.

Протокол аутентификации с использованием KDC

- Это достаточно безопасный протокол при различного рода атаках. Однако авторы предложили пересмотренную версию данного алгоритма:

1. **A → KDC:** $ID_A \| ID_B$
2. **KDC → A:** $Sig_{KRkdc} [ID_B \| KU_B]$
3. **A → B:** $E_{KUb} [N_A \| ID_A]$
4. **B → KDC:** $ID_B \| ID_A \| E_{KUkdc} [N_A]$
5. **KDC → B:** $Sig_{KRkdc} [ID_A \| KU_A] \| E_{KUb} [Sig_{KRkdc} [N_A \| K_S \| ID_A \| ID_B]]$
6. **B → A:** $E_{KUa} [Sig_{KRkdc} [N_A \| K_S \| ID_A \| ID_B] \| N_B]$
7. **A → B:** $E_{Ks} [N_B]$

Протокол аутентификации с использованием KDC

- Добавляется идентификатор A ID_A к данным, зашифрованных с использованием закрытого ключа KDC на шагах 5 и 6 для идентификации обоих участников сессии. Это включение ID_A приводит к тому, что значение nonce N_a должно быть уникальным только среди всех nonces, созданных A, но не среди nonces, созданных всеми участниками. Таким образом, пара $\{ID_A, N_a\}$ уникально идентифицирует соединение, созданное A.