2010 Математические основы компьютерной безопасности №3(9)

DOI 10.17223/20710410/9/8

УДК 004.056

АУТЕНТИФИКАЦИЯ В МОДЕЛИ ДОВЕРЕННОЙ ПОДСИСТЕМЫ НА ОСНОВЕ КОММУТАТИВНОГО ШИФРОВАНИЯ[1]

П. А. Паутов

Томский государственный университет, г. Томск, Россия

E-mail: __Pavel__@mail.ru

В работе рассматривается подход к организации аутентификации в многоуровневой системе, известный как «модель доверенной подсистемы». Для данного подхода формируются требования безопасности и приводится протокол аутентификации, удовлетворяющий этим требованиям. Описываемый протокол построен с использованием коммутативного алгоритма шифрования. Рассматриваются несколько конкретных коммутативных алгоритмов шифрования, применимых в описываемом протоколе.

Ключевые слова: *многоуровневые системы, аутентификация в многоуровневых системах, коммутативное шифрование.*

Введение

Рассмотрим систему, состоящую из трёх взаимодействующих подсистем: клиент, внешний сервер, внутренний сервер. Клиент взаимодействует только с внешним сервером, внешний сервер взаимодействует как с клиентом, так и с внутренним сервером (внешний сервер является клиентом внутреннего сервера). Внешний сервер взаимодействует с внутренним только для обработки запросов своих клиентов.

В таких многоуровневых системах обычно используется одна из двух следующих моделей организации аутентификации [1]:

1) модель делегирования;
2) модель доверенной подсистемы.

В модели делегирования внешний сервер взаимодействует с внутренним от имени клиента, т. е. внутренний сервер содержит учётную запись для каждого клиента внешнего сервера. В модели доверенной подсистемы внешний сервер взаимодействует с внутренним от имени фиксированного набора учётных записей, т. е. одна учётная запись внутреннего сервера соответствует нескольким клиентам внешнего сервера. В данной работе рассматривается модель доверенной подсистемы.

1. Постановка задачи

В модели доверенной подсистемы для взаимодействия с внутренним сервером используется учётная запись, соответствующая привилегиям клиента внешнего сервера. Например, на внешнем сервере клиенты делятся на группы по привилегиям: «гости», «операторы», «администраторы». Тогда для взаимодействия с внутренним сервером можно использовать три учётных записи, соответствующих группам клиентов. Если внешний сервер сам выбирает учётную запись для взаимодействия с внутренним сервером, то в случае компрометации первого злоумышленник сможет использовать

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

учётную запись с максимальными привилегиями. Возникает задача разработки такой схемы аутентификации, при которой внешний сервер смог бы использовать для взаимодействия с внутренним сервером только ту учётную запись, которая соответствует клиенту, и только тогда, когда клиент взаимодействует с внешним сервером. То есть если клиент относится к группе «гости», то внешний сервер может пройти аутентификацию перед внутренним сервером только от имени учётной записи «гость». И внешний сервер не может пройти аутентификацию перед внутренним сервером от имени учётной записи «гость» без помощи клиента, относящегося к группе «гости».

Искомая схема аутентификации должна удовлетворять следующим двум условиям:

С1. При взаимодействии с клиентом внешний сервер может пройти аутентификацию перед внутренним сервером только от имени учётной записи, соответствующей данному клиенту.

С2. Внешний сервер не может пройти аутентификацию перед внутренним сервером от имени какой-либо учётной записи без помощи клиента.

В работе автора [2] предложено несколько схем для случая, когда между клиентом и внешним сервером и между внешним сервером и внутренним используется парольная аутентификация. При использовании парольной аутентификации внешнему серверу необходим пароль учётной записи внутреннего сервера для того, чтобы пройти аутентификацию от имени данной учётной записи. То есть если злоумышленник скомпрометирует внешний сервер так, что он получит доступ на чтение к памяти внешнего сервера, то когда клиент инициирует выполнение протокола аутентификации, описанного в [2], злоумышленник получит пароль учётной записи внутреннего сервера, соответствующей привилегиям клиента. После одного сеанса связи клиента со скомпрометированным внешним сервером злоумышленник получает возможность использовать внутренний сервер без помощи клиента.

2. Протокол на основе коммутативного шифрования

Предлагается схема аутентификации, при использовании которой даже после сеанса связи клиента со скомпрометированным внешним сервером злоумышленник не сможет использовать внутренний сервер без помощи клиента.

Для построения предлагаемой схемы необходимы следующие криптографические примитивы:

1) E — коммутативный алгоритм шифрования, D — соответствующий алгоритм расшифрования;
2) H — хэш-функция.

Каждой учётной записи внутреннего сервера ставится в соответствие некоторый секрет S. Каждому клиенту внешнего сервера ставится в соответствие ключ K алгоритма E. На внешнем сервере для каждого клиента хранится результат шифрования $E_K(S)$, где S соответствует учётной записи внутреннего сервера для данного клиента. Алгоритм аутентификации клиента будет выглядеть следующим образом.

1. Клиент посылает внешнему серверу своё имя.
2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.
3. Внутренний сервер генерирует случайный ключ шифрования K_r алгоритма E и передаёт его внешнему серверу.
4. Внешний сервер находит запись $E_K(S)$, соответствующую данному клиенту, вычисляет $E_{K_r}(E_K(S))$ и передаёт полученный результат клиенту.

5. Клиент вычисляет $H(D_K(E_{K_r}(E_K(S)))) \equiv H(E_{K_r}(S))$ и передаёт данное значение внешнему серверу.
6. Внешний сервер передаёт полученное значение внутреннему серверу.
7. Внутренний сервер вычисляет $H(E_{K_r}(S))$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер перед внутренним.

Как видно из описания, данный протокол обеспечивает выполнение условий C1, C2. Если внешний сервер попытается использовать запись $E_{K'}(S')$, не соответствующую данному клиенту, то проверка на шаге 7 не выполнится, так как $H(D_K(E_{K_r}(E_{K'}(S')))) \neq H(E_{K_r}(S'))$. Для того чтобы пройти аутентификацию перед внутренним сервером без помощи клиента, внешнему серверу понадобится знание S, но это значение не доступно внешнему серверу в открытом виде.

Хэш-функция используется для того, чтобы предотвратить атаку, в которой внешний сервер генерирует собственное значение K'_r и, получив от клиента $E_{K'_r}(S)$, раскрывает значение S.

Используемый коммутативный алгоритм может быть как симметричным, так и асимметричным. В асимметричном варианте каждому клиенту ставится в соответствие пара ключей: открытый K_e и закрытый K_d. На внешнем сервере для каждого клиента хранятся значение $E_{K_e}(S)$ и открытый ключ клиента K_e. Алгоритм аутентификации аналогичен симметричному варианту (но на шаге 3 внутреннему серверу достаточно сгенерировать только открытый ключ).

3. Операции по управлению пользователями

Применение описанной схемы аутентификации повлияет на операции по управлению пользователями. Рассматриваются следующие операции:

1) создание нового пользователя;
2) смена ключа пользователем;
3) изменение учётной записи внутреннего сервера;
4) смена учётной записи внутреннего сервера для данного пользователя системы.

Алгоритмы операций по управлению пользователями строятся по аналогии с [2].

3.1. Операции по управлению пользователями для симметричного E

Создание нового пользователя

1. Администратор генерирует новый ключ K и вычисляет $E_K(S)$ для S, соответствующего новому пользователю.
2. Администратор отправляет K пользователю, а $E_K(S)$ внешнему серверу.

Смена ключа пользователем

1. Пользователь проходит аутентификацию перед внешним сервером.
2. Пользователь запрашивает смену ключа.
3. Внешний сервер генерирует случайный ключ K_r, вычисляет $E_{K_r}(E_K(S))$ и передаёт полученный результат клиенту.
4. Клиент расшифровывает полученное значение на старом ключе и шифрует на новом, отправляет серверу.
5. Сервер расшифровывает полученное от клиента значение с помощью K_r и записывает результат вместо старого значения $E_K(S)$.

Смена учётной записи внутреннего сервера

На внешнем сервере для каждого пользователя хранится запись вида $E_K(S)$. При смене учётной записи внутреннего сервера будет необходимо заменить S на S'. Для этого потребуется зашифровать новое значение S' на ключе пользователя, а так как ключ пользователя известен только пользователю, то провести данную операцию проблематично.

Изменение учётной записи внутреннего сервера для данного пользователя системы

Проведение операции проблематично по тем же причинам, что и проведение операции «Смена учётной записи внутреннего сервера».

3.2. Операции по управлению пользователями для асимметричного E

Создание нового пользователя

1. Пользователь генерирует свои закрытый и открытый ключи и отсылает администратору открытый ключ.
2. Администратор шифрует открытым ключом пользователя соответствующее S и отправляет на внешний сервер.

Смена ключа пользователем

1. Пользователь запрашивает смену ключа.
2. Внешний сервер генерирует случайную пару ключей (K_{er}, K_{dr}), вычисляет $E_{K_{er}}(E_{K_e}(S))$ и передаёт полученный результат клиенту.
3. Клиент расшифровывает полученное значение на старом ключе и шифрует на новом, отправляет серверу значение и новый открытый ключ.
4. Сервер расшифровывает полученное значение с помощью K_{dr} и записывает результат вместо старого значения $E_{K_e}(S)$.

Смена учётной записи внутреннего сервера

Администратор системы обновляет записи $E_{K_e}(S)$ для каждого пользователя, используя открытые ключи пользователей K_e.

Изменение учётной записи внутреннего сервера для данного пользователя системы

Администратор системы обновляет запись $E_{K_e}(S)$ для данного пользователя, используя соответствующий открытый ключ K_e.

4. Применимые коммутативные алгоритмы шифрования

4.1. Сложение по модулю 2

Сложение по модулю 2 можно рассматривать как симметричный коммутативный алгоритм шифрования. Функции шифрования и расшифрования в данном случае совпадают и имеют вид $E_K(X) = D_K(X) = X \oplus K$, где $\oplus$ — побитовое сложение по модулю 2 (X и K рассматриваются как булевы векторы одинаковой длины n). Злоумышленник, скомпрометировавший внешний сервер, получит доступ к значениям вида $E_K(S)$ для всех клиентов и для всех учётных записей внутреннего сервера. Пусть внешний сервер имеет m клиентов и использует одну учётную запись внутреннего сервера. Тогда для получения доступа к внутреннему серверу злоумышленник должен

будет решить следующую систему уравнений:

$$\begin{cases} K_1 \oplus S = e_1, \\ K_2 \oplus S = e_2, \\ \quad ... \\ K_m \oplus S = e_m, \end{cases}$$

где $K_1, K_2, ..., K_m$ (ключи клиентов) и S (секрет внутреннего сервера) являются неизвестными, а $e_1, e_2, ..., e_m$ — известные значения, хранимые на внешнем сервере. В данной системе S является свободной переменной, и, следовательно, система будет иметь 2^n решений, так как S — булев вектор длины n. Таким образом, злоумышленнику придётся произвести полный перебор всех возможных значений S.

4.2. Возведение в степень по модулю простого числа

В качестве симметричного коммутативного алгоритма шифрования можно выбрать функцию возведения в степень по модулю большого простого числа. Пусть p — большое простое число; p является общеизвестным параметром системы. В качестве ключа выбирается K $(1 \leqslant K \leqslant p-2)$, взаимно простое с $p-1$. Тогда функции шифрования, расшифрования будут выглядеть следующим образом: $E_K(X) = X^K \bmod p$, $D_K(X) = X^{K^{-1}} \bmod p$. Данный алгоритм известен в литературе как алгоритм Полига — Хеллмана [3]. В качестве коммутативного шифра он используется в [4].

4.3. RSA

Как видно из описания предлагаемого протокола, на внешнем и внутреннем серверах необходимо лишь выполнить операцию шифрования S на некотором случайном ключе K_r. Так как операция расшифровки не требуется, то для вычислений на внутреннем и внешнем серверах можно использовать некоторую ключевую хэш-функцию H'_K, коммутативную с алгоритмом E_K (т. е. $H'_{K_1}(E_{K_2}(X)) = E_{K_2}(H'_{K_1}(X))$). С учётом сказанного предлагаемый протокол будет выглядеть следующим образом.

1. Клиент посылает внешнему серверу своё имя.
2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.
3. Внутренний сервер генерирует случайный ключ K_r хэш-функции H' и передаёт его внешнему серверу.
4. Внешний сервер находит запись $E_K(S)$, соответствующую данному клиенту, вычисляет $H'_{K_r}(E_K(S))$ и передаёт полученный результат клиенту.
5. Клиент вычисляет $H(D_K(H'_{K_r}(E_K(S)))) \equiv H(H'_{K_r}(S))$ и передаёт данное значение внешнему серверу.
6. Внешний сервер передаёт полученное значение внутреннему серверу.
7. Внутренний сервер вычисляет $H(H'_{K_r}(S))$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер перед внутренним.

В качестве асимметричного варианта алгоритма E можно использовать шифрсистему RSA с модулем n, открытой экспонентой e и закрытой экспонентой d. Каждому клиенту ставится в соответствие пара ключей $(K_e = (e, n), K_d = (d))$. На внешнем сервере для каждого клиента хранится пара $(E_{K_e}(S), K_e) = (S^e \bmod n, (e, n))$ для соответствующего S. Функцию H' можно выбрать как $H'_K(X) = X^K \bmod n$. Тогда предлагаемый протокол будет выглядеть следующим образом.

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи и модуль RSA n, соответствующие клиенту.
3. Внутренний сервер генерирует случайное число r и передаёт его внешнему серверу.
4. Внешний сервер находит запись $S^e \bmod n$, соответствующую данному клиенту, вычисляет $S^{er} \bmod n$ и передаёт полученный результат клиенту.
5. Клиент вычисляет $H(S^{erd} \bmod n) \equiv H(S^r \bmod n)$ и передаёт данное значение внешнему серверу.
6. Внешний сервер передаёт полученное значение внутреннему серверу.
7. Внутренний сервер вычисляет $H(S^r \bmod n)$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер перед внутренним.

Заключение

Рассмотренный в данной работе протокол аутентификации предоставляет более сильные гарантии по сравнению со схемой, описанной в [2]. При использовании данного протокола, даже после сеанса связи клиента со скомпрометированным внешним сервером, злоумышленник не сможет использовать внутренний сервер без помощи клиента. Однако для внедрения данного протокола потребуется добавить его поддержку на все уровни приложения, в то время как описанная в работе [2] схема опирается на широко распространённую парольную аутентификацию. В зависимости от требований к конкретной системе можно использовать тот или иной подход к организации аутентификации.

ЛИТЕРАТУРА

1. *Chong F.* Trusted Subsystem Design // MSDN. 2006. `http://msdn.microsoft.com/en-us/library/aa905320.aspx`
2. *Паутов П. А.* Проблема аутентификации в многоуровневых приложениях // Прикладная дискретная математика. 2008. № 2. С. 87–90.
3. *Schneier B.* Applied Cryptography: Protocols, Algorithms, and Source Code in C. Second Edition. Wiley, 1996. 785 p.
4. *Bao F., Deng R. H., Feng P.* An Efficient and Practical Scheme for Privacy Protection in the E-Commerce of Digital Goods // LNCS. 2001. V. 2015. P. 162–170.