УДК 004.056

АЛГОРИТМ КОНТРОЛЯ ЦЕЛОСТНОСТИ ДЕПЕРСОНАЛИЗИРОВАННЫХ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ ПЕРСОНАЛЬНЫХ ДАННЫХ

Ю.А. Гатчин, О.А. Теплоухова, А.С. Куракин

Рассматривается проблема обеспечения целостности информации, обрабатываемой в информационных системах персональных данных, при использовании алгоритма деперсонализации данных. Предлагается использовать алгоритм контроля целостности, основанный на использовании хэш-функции, для подтверждения неизменности деперсонализированных данных, хранящихся в информационных системах персональных данных.

Ключевые слова: контроль целостности, хэш-функция, алгоритм деперсонализации, персональные данные, информационные системы персональных данных.

Введение

На сегодняшний день обеспечение безопасности персональных данных (ПДн) является одной из острейших проблем в информационной сфере и взаимоотношениях государства, юридических и физических лиц. Выполнение требований к защищенности ПДн при их обработке в информационных системах, устанавливаемых федеральным законом «О персональных данных» [1], является очень дорогостоящей задачей. Для снижения данных требований [2] к уровню защищенности информационных систем персональных данных (ИСПДн), как правило, применяют способы обезличивания, позволяющие существенно сократить расходы на обеспечение их информационной безопасности и снизить вероятность угроз несанкционированного доступа соответственно. Перспективным способом обезличивания персональных данных является алгоритм деперсонализации [3], применение которого делает невозможным определить принадлежность ПДн их владельцу.

Необходимо отметить, что даже после применения алгоритма деперсонализации ПДн остается актуальной угроза нарушения целостности данных, хранящихся в ИСПДн, направленная на изменение или искажение информации, приводящее к нарушению ее качества. Таким образом, для того чтобы оператор ИСПДн был уверен в том, что загружаемые им данные не были искажены, необходима реализация механизма контроля целостности (КЦ) деперсонализированных данных.

Существующие аппаратно-программные комплексы, включающие в свой функционал КЦ данных, являются достаточно дорогостоящим решением, которое не могут позволить бюджеты небольших организаций. Эффективным способом нейтрализации угрозы нарушения целостности ПДн является предлагаемый в настоящей работе механизм, основанный на использовании хэш-функций в процессе деперсонализации данных, позволяющий оператору отслеживать нарушение целостности деперсонализированных данных. Этот не применявшийся ранее подход не требует больших финансовых затрат и при совместном использовании с алгоритмом деперсонализации позволяет эффективно повысить уровень защищенности ИСПДн от НСД.

Целью данной работы является разработка алгоритма КЦ данных, который при совместном использовании с алгоритмом деперсонализации данных в ИСПДн обеспечивал бы обнаружение нарушения целостности хранящейся в постоянном запоминающем устройстве (ПЗУ) информации при ее загрузке в оперативном запоминающем устройстве (ОЗУ).

Описание алгоритма

Перспективным способом решения данной проблемы является использование алгоритма контроля целостности деперсонализированных данных. Предлагаемый алгоритм основан на вычислении хэш-значения от открытых данных при сохранении информации в ПЗУ до применения алгоритма деперсонализации и после применения обратного алгоритма при загрузке данных в ОЗУ. На основании сравнения полученных хэш-значений делается вывод о целостности загружаемой информации.

Рассмотрим таблицу D, содержащую персональные данные субъектов, обрабатываемых в ИСПДн. Во время работы оператора таблица D хранится в ОЗУ в персонализированном виде, т.е. представляет собой открытую информацию (ОИ). До того, как будет применен алгоритм деперсонализации при сохранении данных в ПЗУ, вычислим хэш-значение от ОИ: $h(D)$. После этого применим алгоритм деперсонализации данных, в результате чего будет получена таблица D', представляющая собой файл с закрытой информацией (ЗИ). Добавим вычисленное ранее хэш-значение $h(D)$ к полученному файлу, применив операцию конкатенации, и получим значение $F(D', h(D))$, которое сохраняется в ПЗУ.

Для того чтобы загрузить персональные данные в ОЗУ необходимо извлечь из хранящегося в ПЗУ файла $F(D', h(D))$ хэш-значение $h(D)$, после чего применяем к таблице D' обратный алгоритм деперсонализации, в результате чего получаем таблицу D''. Вычислим новое хэш-значение от полученной таблицы: $h(D'')$ и сравним со значением, которое было извлечено из хранящегося в ПЗУ файла. В случае если $h(D)$ совпадает с $h(D'')$, принимаем что таблица D'' совпадает с исходной таблицей ПДн D, и загружаем ее в ОЗУ для дальнейшей работы. В противном случае, если полученное хэш-значение $h(D'')$ не совпало

с $h(D)$, алгоритм должен выдать сообщение оператору о том, что была нарушена целостность файла с деперсонализированными данными.

Хэш-значение от таблицы ПДн D должно вычисляться после каждого применения к ней обратного алгоритма деперсонализации. При очередном сохранении данных в ПЗУ должен формироваться файл с ЗИ в результате конкатенации таблицы D и нового хэш-значения $h'(D)$.

При получении сообщения о нарушении КЦ оператор может принять решение о повторной проверке целостности деперсонализированных данных. Если вновь вычисленное хэш-значение от таблицы D'' не будет совпадать с извлеченным значением $h(D)$, оператор должен сообщить об этом администратору информационной безопасности. После чего в ОЗУ загружается резервная копия деперсонализированной таблицы D'.

Практическая реализация алгоритма

Использование в предлагаемом алгоритме КЦ хэш-функции обусловлено следующими ее преимуществами:

1. размер входного файла при вычислении хэш-функции может быть произвольным в отличие от шифрования данных;
2. при вычислении хэш-функции не требуется ключ, который необходимо хранить и вводить, что делает реализацию алгоритма очень простой.

Используемая функция

$h(M)\colon \{0,1\}^* \to \{0,1\}^n, n \in N,$

где $M \in \{0,1\}^*$ – произвольное цифровое сообщение, должна быть односторонней хэш-функцией, что означает выполнение следующих условий [4]:

1. значение функции h определено для любого цифрового сообщения $M \in \{0,1\}^*$;
2. для любого цифрового сообщения $M \in \{0,1\}^*$ функция h имеет фиксированный порядок $n \in$ для любого N;
3. для любого $M \in \{0,1\}^*$ значение $h(M)$ вычисляется за полиномиальное время;
4. для любого $M_1 \in \{0,1\}^*$ вычислительно сложно найти сообщение $M_2 \in \{0,1\}^*$, такое, что $M_1 \neq M_2$, $h(M_1) = h(M_2)$;
5. вычислительно невозможно (за разумное время) найти пару (M_1, M_2), $M_1 \neq M_2$, $M_i \in \{0,1\}^*$, $i = 1,2$, такую, что $h(M_1) = h(M_2)$.

Свойства 4, 5 являются важнейшими криптографическими свойствами, обеспечивающими стойкость однонаправленных хэш-функций. В случае если злоумышленнику известен алгоритм построения хэш-функции, первоначальное сообщение $M \in \{0,1\}^*$ и хэш-значение $h(M)$, то вероятность $P(k,n)$ того, что среди сообщений $N_1, \dots, N_k \in \{0,1\}^*$ существует номер $i = 1, \dots, k \in N$, такой что $h(M) = h(N_i)$ равна:

$P(k,n) = 1 - (1 - 2^{-n})^k \approx k2^{-n}$.

Таким образом, для осуществления взлома фиксированного значения для 128-битных хэш-значений (например, MD4 (Message Digest 4), MD5(Message Digest 5)) потребуется перебрать около $\mathbf{2^{120}}$ текстов, что за обозримое время не представляется возможным.

Необходимо отметить, что кроме метода прямого перебора существуют более эффективные атаки на хэш-функции. Так, например, взлом за линейное время (туннельный эффект) [5] основывается на том, что большинство хэш-функций используют сдвиговую функцию: текст M разбивается на блоки M_i, затем происходит итерационный процесс подсчета хэш-значения $h_i = f(h_{i-1}, M_i)$, $i \in N$. В силу небольшой длины раунда можно применить дифференциальный анализ и найти такой текст ΔD, что

$h(M_i + \Delta D) = h(M_i)$.

Ввиду вышесказанного, в предлагаемом алгоритме КЦ рекомендуется использовать одностороннюю хэш-функцию в совокупности с методом повышения ее криптостойкости, например, метод перестановок. В этом случае хэш-функция будет строиться по правилу:

$h(M) = h(M)||h(\pi_1(M)) \dots ||h(\pi_k(M))$.

В данном случае $\pi_i, i = 1, \dots, k,$ – произвольные перестановки текста M. Полученное хэш-значение будет хэш-функцией как конкатенация хэш-значений однонаправленной хэш-функции. Данный метод эффективен против туннельного эффекта, поскольку после перестановки полученный текст отличается от исходного не на туннельный эффект ΔD, что приводит к отличию хэш-значений данных текстов. Кроме того, данный метод может быть эффективно распараллелен, что является важным преимуществом для использования его на маломощных компьютерах.

Для получения хэш-значения таблицы D рекомендуется использовать MD5 – 128-битный алгоритм хеширования, предназначенный для вычисления хэш-функций от сообщений произвольной длины. Данный алгоритм является улучшенной в плане безопасности версией MD4.

На вход алгоритма MD5 поступает входной поток данных, хэш которого необходимо найти. Длина сообщения может быть любой (в том числе нулевой). Запишем длину сообщения в L. Это число целое

и неотрицательное. Кратность каким-либо числам необязательна. После поступления данных идет процесс подготовки потока к вычислениям. Сначала дописывают единичный бит в конец потока (байт 0×80), затем необходимое число нулевых бит. Входные данные выравниваются так, чтобы их новый размер L' удовлетворял условию:

$$L' = 512 \times N + 448.$$

Далее выравненные данные разбиваются на блоки (слова) по 32 бита, и каждый блок проходит 4 раунда из 16 операторов. Для 4 раундов используются следующие функции от трех аргументов:

1 раунд: $F(X,Y,Z) = (X \wedge Y) \vee (\overline{X} \wedge Z)$;

2 раунд: $G(X,Y,Z) = (X \wedge Z) \vee (\overline{Z} \wedge Y)$;

3 раунд: $H(X,Y,Z) = X \oplus Y \oplus Z$;

4 раунд: $I(X,Y,Z) = Y \oplus (\overline{Z} \vee X)$.

Так как множество аргументов хэш-функции счетно, а значения имеют определенный порядок, неизбежно возникновение коллизии – получение одного и того же хэша для двух разных входных значений. При использовании в данном случае одного из самых универсальных методов поиска коллизий – атаки «дней рождений», отыскание коллизии для хэш-функции разрядности n потребует в среднем около $2^{\frac{n}{2}}$ операций, что для функции MD5 составляет 2^{64}. Такая вычислительная сложность нахождения коллизий позволяет гарантировать очень низкую вероятность подмены таблиц с ПДн злоумышленником, которая не была бы обнаружена алгоритмом контроля целостности.

При практической реализации алгоритма также целесообразно хранить отдельно резервные копии деперсонализированных данных. Применение таких мер позволит избежать потери ПДн, обрабатываемых в ИСПДн, в случае, если злоумышленник получит доступ к ЗИ, хранящейся в ПЗУ, и целостность данных будет нарушена.

Заключение

Предложенный алгоритм является наиболее перспективным и оптимальным решением задач по контролю целостности персональных данных, хранящихся и обрабатываемых в информационных системах персональных данных.

Данный алгоритм обладает следующими преимуществами:

1. размер файла с персональными данными, поступающего на вход алгоритма, может быть произвольным;
2. для вычисления используемой в алгоритме хэш-функции не требуется ключ, который необходимо хранить и вводить, что значительно упрощает программную реализацию;
3. для применения данного решения не требуется дополнительных аппаратных средств, используемый алгоритм вычисления хэш-функции MD5 входит в состав открытых библиотек криптопримитивов, таких как OpenSSl, Crypt.dll;
4. алгоритм может быть реализован в качестве программной надстройки над алгоритмом деперсонализации персональных данных.

Литература

1. Федеральный закон Российской Федерации от 27 июля 2006 г. № 152-ФЗ. О персональных данных. Принят Государственной Думой Федерального Собрания Российской Федерации 8 июля 2006 г.: одобрен Советом Федерации Федерального Собрания Российской Федерации 14 июля 2006 г. // Российская газета. – 2006. – 29 июля.
2. Об утверждении Положения об обеспечении безопасности персональных данных при их обработке в информационных системах персональных данных: Постановление Правительства Российской Федерации от 17 ноября 2007 г. № 781 г. Москва // Российская газета. – 2007. – 21 ноября.
3. Куракин А.С. Алгоритм деперсонализации персональных данных // Научно-технический вестник информационных технологий, механики и оптики. – 2012. – № 6 (82). – С. 130–135.
4. Девянин П.Н., Михальский О.О., Правиков Д.И. Теоретические основы компьютерной безопасности: Учебное пособие для вузов. – М.: Радио и связь, 2000. – 192 с.
5. Лёвин В.Ю. О повышении криптостойкости однонаправленных хеш-функций // Фундаментальная и прикладная математика. – 2009. – Т. 15. – № 5. – С. 171–179.

Гатчин Юрий Арменакович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, доктор технических наук, профессор, зав. кафедрой, gatchin@mail.ifmo.ru

Теплоухова Ольга Александровна – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, teplouhovaoa@gmail.com

Куракин Александр Сергеевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, nirt@mail.ru