Методики защиты баз данных от внутренних злоумышленников

И. Б. Алябушев,
В. В. Бабушкин
Группа компаний «Астра СТ»

Вероятно, излишне еще раз говорить об актуальности вынесенной в заголовок проблемы. Стоит лишь конкретнее обозначить ее, чтобы представлять, с каким врагом мы имеем дело.

Речь пойдет о теоретических методах защиты данных от пользователей, уполномоченных с этими данными работать по долгу службы. Почему это представляется интересным? Защита информационной системы от внешнего злоумышленника на сегодня достаточно хорошо формализована, разработаны методы, стандарты и средства, на рынок выведены всевозможные продукты, подготовлены специалисты, и хотя эту проблему нельзя назвать полностью разрешенной, ее решение сегодня требует скорее количественных усилий, нежели качественных.

Иначе дело обстоит с так называемыми «атаками инсайдеров», внутренними атаками. На первый взгляд кажется, что и здесь предложения компаний пестрят всевозможными «средствами защиты от НСД» (от несанкционированного доступа), но на поверку оказывается, что все они сводятся к еще одному оригинальному способу введения пароля (что, безусловно, также важно). Попробуем разобраться в вопросе.

Предположим, мы разработчики базы данных. Пусть это будет база бюро кредитных историй[1]. После того как мы защитили все сервисы доступа к БД от проникновения извне и определились с постулатом: *пользователь должен проходить процедуру идентификации, аутентификации и авторизации при доступе к данным*, мы решили одну проблему и создали новую. Мы создали сущности данных, определили на них права пользователей, учли все нюансы, обучили администратора – защитили систему.

Теперь посмотрим на эту ситуацию со стороны владельцев базы. Мы (владельцы) заказали разработку базы данных, потребовали защитить все подступы к ней и попросили ограничить права пользователей. При этом уполномоченные пользователи системы (а в нашем примере это сотрудники головного офиса бюро кредитных историй) должны работать с данными. Например, они должны отвечать на письменные запросы в бюро, то есть могут получить любую кредитную историю из базы данных. Или, может быть, сразу все?

Итак, мы создали базу данных, защищенную от внешних атак. Отныне она требует наполнения и обслуживания. Мы предоставили к ней частичный доступ сотрудникам нашей организации и абсолютный доступ администратору системы –

[1] *Рассматриваемый пример с бюро кредитных историй – не вымышленный, с применением описанных концепций действительно была разработана информационная система, получившая название «Плеяды». На базе этой системы функционирует Межрегиональное бюро кредитных историй (http://www.mbki.ru). Эта организация под номером 2 была внесена в государственный реестр бюро кредитных историй (http://www.fcsm.ru/catalog.asp?ob_no=24284).*
Разработкой занималась компания ИВЦ-1 (http://www.ivc-1.ru), которая входит в группу компаний «Астра СТ» (http://www.astra-st.ru).

с этого момента перед нами предстала проблема суперпользователя.

Здесь проявляется важная логическая особенность проблемы внутренних атак: защититься от инсайдера невозможно принципиально. Это очень похоже на известный парадокс Рассела или парадокс брадобрея, бреющего несамостоятельных жителей деревни. Попробуем перефразировать его так, чтобы он звучал в наших терминах.

Администратор-брадобрей блокирует доступ к системе каждому, кто не должен его иметь, то есть каждому, кто не блокирует свой доступ сам. А дальше следует стандартное рассуждение: если администратор не блокирует себе доступ сам, значит, он должен его себе блокировать. Если же администратор блокирует доступ сам себе, то это уже не администратор, так как он может блокировать доступ только тем, кто сам не может его себе заблокировать. Тогда кто блокирует доступ администратору?

Это забавное рассуждение ставит владельцев базы данных перед печальными выводами. Впрочем, ситуация не так безнадежна. Нельзя забывать, что этой проблеме не одна сотня (а, может быть, тысяча) лет. И за прошедший век тоталитарные государственные машины добились в этом направлении потрясающих результатов. Ключевое слово в решении, которое они применяли, – «регламент». Конечно, эти методы по-прежнему эффективны.

Вернемся к нашему примеру – бюро кредитных историй. Очевидно, что на этапе его создания мы должны принять в штат «специалиста по безопасности». Этот сотрудник должен быть даже не IT-специалистом, а скорее специалистом по безопасности информации вообще (хорошо подходят отставные военные). В его обязанности входит составление регламента доступа к защищаемой информации. В документах, которые он разрабатывает, есть строчки: «*…отсутствовать сменные носители*», «*запретить доступ в нерабочее время*», «*…процедуру смены пароля не реже…*», «*ограничить функционал рабочего места…*», «*изолировать вычислительную сеть от…*», «*…несет личную ответственность*», «*…решается в судебном порядке*» и т. п.

Слабое место этого подхода очевидно – человеческий фактор.

Мы не будем сейчас останавливаться на регламенте безопасности информационной системы, поскольку этому посвящено множество трудов – мы коснемся его позже, но уже в другом аспекте. Также мы не будем останавливаться на психологических приемах решения этой проблемы (мотивирование сотрудников, имеющих доступ к БД, а может, запугивание?) – эта тема требует отдельного рассмотрения. Мы попробуем выработать такие **концепции проектирования** информационной системы, которые помогут **автоматизировать защиту** от внутренних атак.

Итак, если проблема не имеет решения, следует сокращать ее негативное воздействие. Если защитить данные от внутренних атак принципиально невозможно, можно попытаться сузить границу атаки, фронт нападения внутренних злоумышленников. Идеальный вариант – если база будет умещаться в бумажный блокнот, ее можно положить в карман и не выпускать из рук даже тогда, когда кто-то читает из нее данные. В этом случае единственным мгновением времени, когда она будет уязвима, останется момент, в который эту базу достают из кармана (от воровства из кармана, то есть внешних угроз, мы защитились в самом начале). От действительности эту ситуацию отделяет даже не то, что реальная база кредитных историй в блокноте не поместится, а то, что ее обслуживание требует специальных навыков, которыми владелец базы данных в общем случае не владеет.

Удобно было бы доверить управление базой некоторой интеллектуальной сущности, назовем ее «гномом». Эта сущность должна уметь передавать записи, управлять структурой базы, проводить обслуживание, но при этом «гном» должен быть закрыт от внешнего мира в «черном ящике», который даже для назначенного администратора будет служить промежуточным слоем абстракции при доступе к структуре и записям базы.

Гном в ящике

Это базовая концепция создания защищенной от внутренних атак информационной системы. Ее суть в сосредоточении всех полномочий доступа к базе в рамках одного программного модуля – «гнома». А ящиком для него будет являться комплекс мер.

«Гном» общается с внешним миром на уровне информационных сообщений (посредством web-сервисов, например), этот канал доступа должен быть универсальным и единственным. Другими словами, в «ящик» может пролезать только соответствующий формату «конверт». Каждый «конверт», то есть каждый запрос на выполнение операции, сопровождается электронной подписью отправителя.

Таким образом, данный программный модуль становится единственным способом доступа в систему (или доступа к ее закрытым данным).

Как же описанную концепцию воплотить в жизнь? В обобщенных терминах это может выглядеть так…

Для начала нужно сказать несколько слов об инфраструктуре открытых ключей. Это набор технологий, алгоритмов и стандартов, на которых, в частности, реализован сертификат электронно-цифровой подписи. Каждый сертификат содержит пару ключей для асимметричного шифрования (закрытый ключ хранится отдельно в защищенном контейнере) и описание сертификата, на которое наложена электронная подпись выдавшей организации (удостоверяющего центра). С точки

зрения пользователя, сертификат цифровой подписи позволяет подписывать блоки данных, проверять подпись и шифровать. За счет описания сертификат также удобно использовать в качестве средства идентификации. В нашем примере надежными сертификатами цифровой подписи должны быть снабжены все пользователи базы данных и, конечно, ее владелец.

Вернемся к реализации.

Так как «гном» – единственная сущность, которая имеет доступ к закрытым таблицам, она должна быть единственной сущностью, владеющей паролем суперпользователя. Возникает вопрос: где же его хранить? Очевидно, что в хранилище он может находиться только в зашифрованном виде. Расшифровываться он должен закрытым ключом сертификата владельца. Постоянно держать сертификат цифровой подписи владельца на сервере нелепо. Вывод: закрытый ключ сертификата владельца должен быть кэширован и загружен в оперативную память на стадии загрузки системы.

Этот путь приводит нас к одному ограничению и одной проблеме. Первое: при каждой загрузке системы ей потребуется инициализация сертификатом владельца. Второе: из оперативной памяти кэшированный закрытый ключ можно украсть. Однако к этой проблеме мы вернемся в следующих концепциях.

Подведем промежуточный итог: при загрузке системы запускается программный модуль «гном в ящике», который инициализируется сертификатом владельца и становится единственным окном в базу данных.

Теперь двинемся далее. Для работы пользователей «гном» должен уметь их идентифицировать. С этой целью программный модуль ведет реестр сертификатов электронных подписей субъектов, осуществляющих доступ. Сертификат владельца БД предустановлен в этот реестр. Пользователи отправляют «гному» свои сообщения, инструкции (например, упакованные в XML), «гном» идентифицирует отправителей по их сертификатам в своем реестре, проверяет подпись и принимает решение о выполнении запроса.

Первые инструкции, которые увидит «гном» в своей жизни, – это инструкции от владельца БД. В них будут указаны сертификаты, владельцы которых должны получить доступ к БД. Эти управляющие запросы будут подписаны электронной подписью владельца БД (сертификат которого предустановлен), после чего «гном» начнет обрабатывать запросы уже от вновь добавленных пользователей.

Такая системная роль, как администратор, в данной концепции претерпевает изменения. Как уже было сказано, пароль суперпользователя системы не знает ни один пользователь. Этот пароль генерируется случайным образом на стадии первичной настройки и сохраняется в независимом от базы данных контейнере (например, в запечатанном конверте в банковской ячейке или сейфе владельца БД), плюс его зашифрованная копия остается у «гнома». Тот пользователь, который все же занимается администрированием, общается с БД через «гнома» так же, как и все остальные.

Инструкции администратора, которые должен выполнять «гном», можно разбить на две категории, условно назовем их «проверенные» и «свободные».

Первые представляют собой подпрограммы на языке управления базами данных, проверенные внешним аудитором, и считаются безопасными для БД (например, создание индекса). Главная особенность таких инструкций состоит в том, что для их вызова администратор запрашивает наименование инструкции и передает только параметры вызова, но не сам код. Конечно, при аудите этих подпрограмм необходимо убедиться в невозможности превысить полномочия, используя параметры, заданные особенным образом.

Вторые – это код на языке управления БД в чистом виде. Однако этот код «гном» также пропускает через себя. Что может защитить от утечки в данном случае? Уведомление владельца о необходимости выполнить свободную инструкцию (или требование его разрешения, то есть подписи), электронная подпись администратора под каждой такой инструкцией и ведение журнала выполненных инструкций на отдельном сервере. Тогда в случае утечки можно будет найти виновника. Конечно, важно, чтобы страх злоумышленника от осознания этой неизбежности превалировал над утешением для владельца от его поимки после раскрытия информации.

Очевидно, что защищенность базы в данном случае зависит от соотношения проверенных и свободных инструкций. Другими словами, нужно наращивать интеллект «гнома» (или эрудированность) – набор тех функций, которые может выполнять этот сервис по авторизованному запросу администратора. В идеале в набор «умений» должны входить операции управления индексами, табличным пространством, буферизацией, а также работа с некоторыми базовыми сущностями предметной области: созданием/удалением пользователей, определением их полномочий и в нашем примере – добавлением сертификатов электронной подписи пользователей.

Эта концепция не лишена недостатков: ее, например, очень сложно применять в базах данных, которые находятся на начальном уровне развития своей структуры и требуют постоянного вмешательства со стороны архитекторов. «Гном в ящике» в чистом виде предполагает использование «зрелых» баз данных. Кроме того, возникает проблема доверия «гному». Она, в свою очередь делится на проблему доверия разработчику этого программного модуля (нет ли там «закладок» или просто уязвимостей) и на проблему доверия текущему экземпляру «гнома» (не был ли он подменен).

Первая проблема отчасти решается привлечением стороннего аудитора в области информационных систем, а вторая – сопровождением исполняемого кода «гнома» электронной подписью. «Отчасти» же мы говорим потому, что каждое новое «решение» поднимает новый вопрос доверия (аудитору или тому коду, который будет проверять электронную подпись «гнома»).

После рассмотрения достоинств и недостатков этой концепции стоит вернуться к уже упомянутому регла-

менту безопасности. «Тонкое место» в данной концепции – регламентное предоставление полного доступа администратору базы данных для изменений структуры (разрешение выполнения свободных инструкций). Ведь если задача, которую должен выполнить администратор, не входит в список умений «гнома» (не автоматизирована), администратору нужно предоставлять полный доступ к системе.

В этом случае весьма ответственная роль возлагается на регламент. Можно придумать свод правил, которые максимально усложнят раскрытие данных: выдавать разрешение на выполнение свободных инструкций только после прохождения администратором проверки на детекторе лжи, при работе администратора с базой в таком режиме закрывать доступ в сеть, ограничивать трафик обмена с базой данных, требовать наличия двух свидетелей, проводить предварительный аудит кода и т. д.

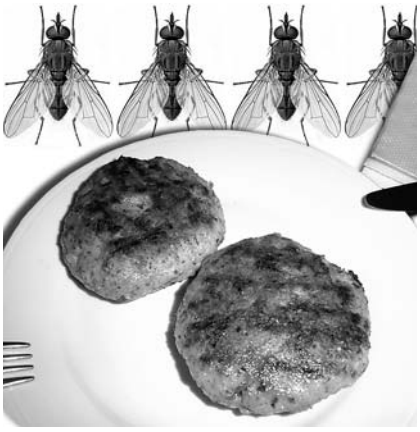
Нельзя забывать и о том, что с увеличением усилий, требуемых для воровства информации, растут и усилия, необходимые для поддержки и развития базы данных.

Если «гном в ящике» сковывает структурное развитие базы и к тому же ограничивает возможности работы со статистикой и тестовыми данными, мы можем попытаться расширить границы, сделав ящик не таким черным. Но как при этом обезопасить данные? Может ли существовать база данных, информация в которой абсолютно надежно закрыта? Да, если эта информация абсолютно ничего не стоит…

Котлеты отдельно от мух

Эта концепция ориентирована на содержание баз данных. Она работает для записей одного рода, описывающих сущность одного типа. Ее суть заключается в следующем.

На первом этапе необходимо выделить из защищаемых данных единичные записи. В нашем примере это записи кредитных историй из базы данных бюро или сообщения об изменении кредитной истории. На втором – эти единичные записи логически делятся на две части, при этом каждая часть содержит только ту информацию, на основании которой невозможно провести соответствие между частями записи. Таким образом, эти части должны быть независимы с точки зрения идентификации, например, титульная и содержательная части кредитной истории, где в первой части описывается субъект, а во второй – полученные им кредиты. Разделенные части записей хранятся в разных таблицах или даже в разных базах данных, разнесенных не только логически, но и географически.



Секретной информацией остается лишь реестр сопоставления, то есть таблица, хранящая идентификаторы для связывания частей записей, и ни что более. Нам удается «сузить покрывало секретности» над данными в базе, не раскрывая при этом информацию (здесь вопрос терминологии: данные раскрываются, но информация – нет). Это дает серьезные преимущества при работе с данными в защищенном режиме. В такой базе данных можно выполнять регламентные работы, структура базы открыта для модификаций. Таблицы, хранящие части записей, могут расширяться горизонтально, так как данное действие не повлияет на механизм объединения записей (при этом нельзя забывать: при расширении таблиц не должна добавляться идентифицирующая информация). Также появляется возможность работать с тестовыми данными, проводить обезличенный анализ, производить статистические расчеты и пр.

Возникает вопрос: как защитить секретный реестр сопоставления от суперпользователя? Обратимся к предыдущей концепции. Именно эту задачу решает «гном в ящике». Но в такой реализации «гном» будет иметь монопольный доступ только к реестру сопоставления частей записей. Пользователь-администратор должен обладать правами, близкими к суперпользователю за исключением, конечно, доступа к секретной таблице.

Приведенная концепция имеет свои недостатки. Границу, по которой можно разбить данные, провести достаточно сложно (а порой вообще невозможно). В некоторых случаях сопоставление частей можно произвести по косвенным признакам (например, если в содержательную часть необходимо включить возраст и адрес субъекта). При большом перечне хранимых полей записи каждая отдельная часть может содержать достаточное количество информации, в том числе конфиденциальной, для привлечения к себе интереса злоумышленника (например, титульные части кредитной истории содержат паспортные данные субъекта).

Для преодоления этих сложностей можно использовать экстенсивный метод: разбивать данные не на две, а на большее количество записей. В случае с бюро кредитных историй можно разбить титульную часть на ФИО, паспортные данные, дату рождения и прочие сведения, а содержательную – делить по договорам займа (не забывая о компромиссе с потерей эффективности базы данных).

Разделение данных на части – не единственный прием для понижения ценности информации. Что, если применить эту концепцию не дважды и не трижды, а бесконечное множество раз, превратив данные в бесполезный мусор?

Ржавый сундук

Основанная идея этой концепции: содержимое БД может представлять интерес только в момент авторизованного запроса, в остальное время данные на физическом

носителе должны представлять собой лишенный содержания набор символов, «ржавый хлам» в «сундуке». Другими словами, все данные должны быть зашифрованы, прежде чем будут записаны, и расшифрованы при санкционированном прочтении. Речь идет о популярной идее *«прозрачного шифрования»* или *«шифрования на лету»*. В таком случае украденное хранилище или сервер не будут представлять интереса для злоумышленника.

Как это можно реализовать? Собственно, реализация технологии «шифрования на лету» хорошо известна. В базе данных создается таблица, в которой хранятся ключи симметричного шифрования, используемые ядром базы данных для шифрования остальных таблиц. Сама же таблица ключей также зашифрована, и для инициализации ядра ее нужно расшифровать. Вся сложность заключается в том, где хранить ключ для расшифровки таблицы ключей.

С подобной проблемой мы уже встречались, обсуждая концепцию «гном в ящике» (там вопрос состоял в месте хранения пароля суперпользователя). И в первом и во втором случае можно предложить аппаратно-программное решение, которое если не разрешит проблему целиком, то значительно удорожит атаку на систему.

Для того чтобы исключить возможность считывания секретного ключа из оперативной памяти, нужно организовать процесс расшифровки таким образом, чтобы секретный ключ в память никогда не попадал. Это можно осуществить, если сама расшифровка будет выполняться не средствами центрального процессора, а дополнительным устройством, которому на вход передают зашифрованные данные, а на выходе получают расшифрованные, и наоборот.

К этому устройству мы предъявим ряд требований. Шифрование необходимо производить достаточно быстро (ведь нужно обработать весь трафик базы данных). Секретный ключ должен находиться не в ПЗУ устройства, а в оперативной памяти и при выключении питания – безвозвратно исчезать (иначе он по-прежнему останется легкой добычей администратора). Считывание секретного ключа из оперативной памяти устройства должно быть настолько же трудоемкой задачей, насколько ценна информация в базе данных.

Такие решения уже можно найти на рынке, хотя нельзя сказать, что последний ими изобилует (речь идет об устройствах, поддерживающих сертифицированные в РФ алгоритмы шифрования).

Что можно получить в результате? При загрузке сервера владелец базы данных инициализирует его (а точнее устройство шифрования) своим сертификатом электронной подписи. Закрытый ключ попадает в защищенное оперативное хранилище. Затем ключ изымается и запускается база данных. Пользователи запрашивают данные, после чего зашифрованные записи передаются в устройство шифрования, откуда возвращаются пользователям расшифрованными. При потере сервером питания закрытый ключ также будет потерян, и база данных будет оставаться «бесполезной» до инициализации ключом владельца.

Такая методика делает бессмысленной кражу образа базы данных из физического хранилища и резервных копий на сменных носителях (ведь они также зашифрованы). Однако она не защищает от администратора базы! Ведь он по-прежнему может сделать запрос в базу данных, и ядро «на лету» расшифрует все записи. Проблему можно решить, поручив функции «прозрачного» шифрования все тому же «гному», открыв доступ к устройству шифрования только для этого сервиса.

Сразу же возникает другая проблема: найти способы ограничения доступа к устройству шифрования. Этого можно добиться средствами операционной системы сервера, что порождает появление следующего препятствия – доверия администратору операционной системы сервера… Замкнутый круг.

Ответ один – регламент.

Узкий лаз

Даже когда записи в базе *поделены на части, зашифрованы* и *закрыты от администратора в «ящик»*, по-прежнему остается проблема внутреннего злоумышленника – проблема рядового пользователя. Ее суть состоит в следующем: БД создается для работы рядовых пользователей, которые должны получать доступ к хранимым в ней данным. При этом они же представляют собой опасный канал утечки информации. Проблема заключается не только в доверии сотрудникам, нельзя забывать, что секретный ключ пользователя может быть скомпрометирован (украден).

На примере бюро кредитных историй: по закону бюро должно выдать кредитный отчет субъекту кредитной истории по его личному запросу, удостоверив личность субъекта по соответствующим документам. Очевидно, эту операцию не может выполнять владелец базы данных, этим занимаются сотрудники бюро, что открывает перед ними возможность выдать «себе» нужные кредитные отчеты. Разумеется, такая вольность должна быть ограничена регламентом, который требует наличия подписанных заявлений субъектов на получение кредитных отче-

тов. Мы же попробуем автоматизировать решение.

Корень проблемы лежит в том, что правила доступа пользователей к данным в системе обычно формализованы только качественно (чтение, изменение и т. п.). Отсюда и вытекает концепция «узкого лаза». Вот ее суть: поток данных, передаваемый пользователю, ограничивается по количественным или качественно-количественным критериям. Таким образом, кража «всего и сразу» становится невозможной. Например, нельзя получить более десяти кредитных отчетов в час с одного рабочего места.

Можно выделить следующие критерии ограничения: количество информации в единицу времени, ограничение по времени (нельзя делать запросы после окончания рабочего дня), ограничение по группам пользователей (например, не более 200 запросов от группы адресов, от одного отдела, района, города) и т. п.

Для гибкости алгоритма можно ввести так называемую матрицу коэффициентов доверия и значимости. Коэффициент доверия – число в заданном диапазоне, которое характеризует уровень доверия рабочему месту пользователя базы данных. Он может зависеть от времени работы пользователя с системой, уровня защищенности рабочего места, репутации пользователя и т. п. Коэффициент значимости – другое число в том же диапазоне, характеризующее «значимость» записи в БД (и субъекта, который описан в записи). Значимость – практически субъективная величина, показывающая интерес к данной записи со стороны абстрактного злоумышленника. В качестве этой величины можно, например, использовать частоту доступа к записи, ее актуальность или годовой оборот организации (в случае с бюро кредитных историй).

Если пользователь с низким коэффициентом доверия начинает активно запрашивать записи с высоким коэффициентом значимости, он получит временный отказ. При этом владельцу БД будет отправлено уведомление о том, что система ограничила доступ этому клиенту к определенным записям. Владелец может принять решение о возобновлении доступа.

Подведем итоги наших «исследований». Из приведенных рассуждений сделаем следующий вывод: проблема еще не может быть закрыта, но поднять уровень защиты от внутренних злоумышленников на новую высоту вполне возможно, причем это потребует скорее качественных, чем количественных усилий. Можно допустить, что, как и в парадоксе Рассела о брадобрее, решение данной проблемы кроется в ином формулировании условий. Быть может, будущие разработки в области IT и законотворчества (или даже социологии и психологии) позволят смотреть на эту проблему совершенно под другим углом. ■

НОВОСТИ

Личным компьютером пользоваться запрещено!

Японское военное ведомство категорически запретило своему персоналу даже приносить с собой на службу личные компьютеры. Об этом говорится в приказе, который приняло 4 декабря Управление обороны страны. Военный и гражданский персонал, говорится в нем, не имеет права записывать любую служебную информацию на свои личные компьютеры или на переносные носители памяти. Такие устройства запрещается даже брать с собой на службу.

Приказ был принят после того, как несколькими днями ранее стало известно о том, что с личного компьютера одного из офицеров ВВС Японии в открытый доступ в Интернете попали секретные сведения вооруженных сил США. Они касались характера транспортных перевозок в Ирак, внутреннего строения расположенных там американских баз и размещения военного персонала, включая его численность и обязанности. Утечка произошла из-за того, что компьютер японского офицера был заражен вирусом и самостоятельно выдал закрытые сведения в Интернет.

Такие инциденты в японских вооруженных силах происходили уже неоднократно, сообщает ИТАР-ТАСС.

Студент из Ростова осужден за хищение денег

В Ростове-на-Дону осужден участник международной преступной группы, занимавшейся хищением денежных средств с использованием компьютерных технологий.

Студент одного из ростовских вузов Юрий Сергостьянц признан виновным по статьям «Неправомерный доступ к охраняемой законом компьютерной информации, совершенной группой лиц» и «Мошенничество, совершенное группой и в особо крупном размере» УК РФ. Он приговорен к 6 годам лишения свободы условно с испытательным сроком 4 года.

Следствием установлено, что в декабре 2003 г. группой было похищено со счетов клиентов американских компаний денежных средств на сумму эквивалентную 3 млн 120 тыс. руб., из них лично Сергостьянц получил 900 тыс. руб.

Ростовский студент через Интернет получал от своего подельника за рубежом данные счетов клиентов американских компаний. Затем он взламывал счета и переводил деньги соучастнику, находящемуся за границей. Владелец этого счета с помощью банковских операций перемещал украденные средства в страны Европы, Прибалтики и Россию.

В раскрытии данного преступления российские контрразведчики действовали в тесном взаимодействии со спецслужбами США. По словам представителя отечественных правоохранительных органов, ему неизвестна судьба находящихся за границей других участников группировки, так как это дело юрисдикции этих стран.

При вынесении приговора учтено полное признание подсудимым вины, помощь, оказанная следствию, и то, что он частично, в размере 1 млн руб. возместил нанесенный им ущерб.

www.vz.ru