МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ

Методические рекомендации к лабораторным работам для студентов направлений подготовки 09.03.04 «Программная инженерия» и 09.03.01 «Информатика и вычислительная техника» дневной формы обучения

Могилев 2019

УДК 004.4
ББК 32.973-018.2
М 54

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «04» сентября 2018 г., протокол № 2

Составитель ст. преподаватель Е. А. Зайченко

Рецензент Ю. С. Романович

Даны методические рекомендации по выполнению лабораторных работ по дисциплине «Методы и средства защиты информации», а также приведены задания к ним и список литературы для подготовки.

Учебно-методическое издание

МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ

Ответственный за выпуск	А. И. Якимов
Технический редактор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 31 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

Введение

Целью преподавания дисциплины «Методы и средства защиты информации» является обучение студентов основным методам обеспечения информационной безопасности, средствам защиты информации, современным аппаратным и программным алгоритмам шифрования информации, построения надежных систем хранения информации, а также изучение перспективных направлений в развитии современных средств обеспечения информационной безопасности.

Методические рекомендации имеют целью помочь студентам в подготовке и выполнению лабораторных работ по дисциплине.

Даны методические рекомендации по выполнению лабораторных работ по дисциплине «Методы и средства защиты информации», а также приведены задания к ним и список литературы для подготовки.

Лабораторная работа № 1. Хеширование информации SHA

Цель работы: ознакомление с понятием хеширования, изучение криптографических хеш-функций.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.
2 Получить задание у преподавателя, выполнить типовые задания.
3 Сделать выводы по результатам исследований.
4 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования.
4 Выводы.

Основные теоретические положения

Понятие хеширования. Хеширование – преобразование входного массива данных в короткое число фиксированной длины (которое называется хешем или хеш-кодом) таким образом, чтобы, с одной стороны, это число было значительно короче исходных данных, а с другой, с большой вероятностью однозначно им соответствовало. Преобразование выполняется при помощи хеш-функции. Ясно, что в общем случае однозначного соответствия между исходными данными и хеш-кодом быть не может. Обязательно будут возможны массивы данных, дающие одинаковые хеш-коды, но вероятность таких совпадений в каждой конкретной задаче должна быть сведена к минимуму выбором хеш-функции.

Хеширование применяется для сравнения данных: если у двух массивов хеш-коды разные, массивы гарантированно различаются; если одинаковые – массивы, скорее всего, одинаковы. В общем случае однозначного соответствия между исходными данными и хеш-кодом нет в силу того, что количество значений хеш-функций меньше, чем вариантов входного массива; существует множество массивов, дающих одинаковые хеш-коды – так называемые коллизии. Вероятность возникновения коллизий играет немаловажную роль в оценке качества хеш-функций.

Существует множество алгоритмов хеширования с различными характеристиками (разрядность, вычислительная сложность, криптостойкость и т. п.). Выбор той или иной хеш-функции определяется спецификой решаемой задачи.

Простым примером хеширования может служить нахождение циклической контрольной суммы или *CRC*, когда берётся текст (или другие данные) и суммируются коды входящих в него символов, а затем отбрасываются все цифры, за исключением нескольких последних. Полученное число может являться при-

мером хеш-кода исходного текста.

Контрольные суммы – несложные, крайне быстрые и легко реализуемые аппаратные алгоритмы, используемые для защиты от непреднамеренных искажений, в том числе ошибок аппаратуры.

По скорости вычисления они в десятки и сотни раз быстрее, чем криптографические хеш-функции, и значительно проще в аппаратной реализации.

Платой за столь высокую скорость является отсутствие криптостойкости, возможность подогнать сообщение под заранее известную сумму. Также обычно разрядность контрольных сумм (типичное число 32 бита) ниже, чем криптографических хешей (типичные числа: 128, 160 и 256 бит), что означает возможность возникновения непреднамеренных коллизий.

Простейшим случаем такого алгоритма является деление сообщения на 32- или 16-битные слова и их суммирование, что применяется, например, в *TCP/IP*.

Как правило, к такому алгоритму предъявляются требования отслеживания типичных аппаратных ошибок, таких как несколько подряд идущих ошибочных бит до заданной длины. Семейство алгоритмов так называемых «циклических избыточных кодов» удовлетворяет этим требованиям. К ним относится, например, *CRC32*, применяемый в аппаратуре *Ethernet* и в формате упакованных файлов *ZIP*.

Криптографические хеш-функции. Среди множества существующих хеш-функций принято выделять криптографически стойкие, применяемые в криптографии. Для того чтобы хеш-функция H считалась криптографически стойкой, она должна удовлетворять основным требованиям, на которых основано большинство применений хеш-функций в криптографии:

– необратимости (невозможность вычислить исходные данные по результату преобразования): для заданного значения хеш-функции m должно быть вычислительно неосуществимо найти блок данных X, для которого $H(X) = m$;

– стойкости к коллизиям (два различных набора данных должны иметь различные результаты преобразования): для заданного сообщения M должно быть вычислительно неосуществимо подобрать другое сообщение N, для которого $H(N) = H(M)$.

Следует отметить, что не доказано существование необратимых хеш-функций, для которых вычисление какого-либо прообраза заданного значения хеш-функции теоретически невозможно. Обычно нахождение обратного значения является лишь вычислительно сложной задачей.

Применение хеш-функций. Хеш-функции также используются в некоторых структурах данных – хеш-таблицах, фильтрах Блума и декартовых деревьях. Требования к хеш-функции в этом случае другие: хорошая перемешиваемость данных; быстрый алгоритм вычисления.

Сверка данных. Это применение можно описать, как проверку некоторой информации на идентичность оригиналу без использования оригинала. Для сверки используется хеш-значение проверяемой информации. Различают три основных направления этого применения: проверку на наличие ошибок; проверку парольной фразы; ускорение поиска данных.

Проверка на наличие ошибок. Например, контрольная сумма может быть передана по каналу связи вместе с основным текстом. На приёмном конце контрольная сумма может быть рассчитана заново и её можно сравнить с переданным значением. Если будет обнаружено расхождение, то это значит, что при передаче возникли искажения и можно запросить повтор.

Вопросы для контроля

1 Что такое хеш-функция?
2 В чем состоит алгоритм контрольной суммы?
3 Перечислите требования к криптостойким хеш-функциям.

Лабораторная работа № 2. Использование технологий криптографии для передачи конфиденциальной информации по вычислительным сетям

Цель работы: изучение видов атак и технологий криптографии, используемых для передачи информации в сетях.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.
2 Получить задание у преподавателя, выполнить типовые задания.
3 Сделать выводы по результатам исследований.
4 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования.
4 Выводы.

Основные теоретические положения

Одной из основных причин уязвимости информационных систем в сети Интернет являются слабости сетевого протокола IP стека протоколов TCP/IP, который служит базой сетевых коммуникаций. При проектировании этого протокола на заре развития Интернета проблемы безопасности стояли не так остро, и поэтому внимания защищенности протокола практически не уделялось. В настоящее время ведутся разработки новой версии протокола – IPv6, который, как ожидается, будет содержать встроенные механизмы защиты и поможет снять многие проблемы. Рассмотрим основные сетевые угрозы.

Атаки типа «отказ в обслуживании» (denial of service, DoS). Под этим названием объединены методы, призванные расстроить работу некоторого сетевого устройства путем перегрузки какого-либо ограниченного ресурса, отказа устройства, изменения его настроек. Ограниченными ресурсами являются оперативная память, емкость жесткого диска, процессорное время. Типичная DoS-атака представляет собой генерацию большого потока сетевых пакетов, которые не успевают обрабатываться сетевыми серверами, что приводит либо к отказам в их работе, либо к невозможности обрабатывать запросы обычных пользователей. В качестве примера DoS-атаки можно привести атаку SYN flooding, когда злоумышленник отправляет на атакуемый компьютер множество TCP-пакетов с установленным флагом SYN и несуществующим адресом отправителя. Атакуемый компьютер в ответ на каждый такой пакет создает внутренние структуры данных, необходимые для поддержки соединения, и отправляет ответный пакет для подтверждения соединения. При большом количестве атакующих пакетов память атакуемого компьютера может закончиться, что приведет к его отказу. Для предотвращения подобных атак используются интеллектуальные фильтры пакетов, в момент возникновения атаки можно динамически уменьшать время ожидания ответа на пакет. Особенно разрушительными являются распределенные DoS-атаки (distributed DoS, DDoS). При реализации атаки данного типа злоумышленник сначала должен получить доступ ко множеству слабо защищенных компьютеров, подсоединенных к сети Интернет, и установить на них специальное программное обеспечение DDoS. Удаленный компьютер с установленным программным обеспечением DDoS называется агентом. По команде с компьютера злоумышленника агенты начинают передавать сетевые пакеты на атакуемый компьютер.

Средства криптографической защиты соединений в вычислительных сетях. Для обеспечения конфиденциальности информации, передаваемой по сети, необходимо обеспечить ее шифрование на стороне отправителя и дешифрацию на стороне получателя. Самым простым средством является шифрование информации на прикладном уровне. В этом случае шифрованию подвергается только непосредственно передаваемая информация, никакая служебная информация из заголовков сетевых пакетов не кодируется. Примером программы, которая осуществляет подобного рода шифрование, можно назвать PGP (Pretty Good Privacy). Одно из главных достоинств этой программы состоит в том, что версии PGP представляют собой криптосистему, которая позволяет шифровать данные (содержимое файлов, буфера обмена) по асимметричной схеме, а также формировать ЭЦП для передаваемых сообщений. В PGP используются следующие алгоритмы: RSA, SHA, DES, CAST, IDEA, DSS. Закодированная информация сохраняется в виде файла, который может быть передан по сети любым способом.

Применение программы PGP и подобных ей может оказаться неудобным вследствие того, что пользователю необходимо самому принимать меры для шифрации сообщений. Для того чтобы сделать процедуру шифрования прозрачной для пользователя, существуют различные сетевые протоколы, реализующие технологии защищенных соединений. Рассмотрим один из подобных

протоколов – протокол SSL (Secure Socket Layer). Он предоставляет также возможность аутентификации сервера и клиента. SSL работает на представительском уровне модели OSI поверх протокола TCP.

Преимуществом SSL является то, что он независим от прикладного протокола. Прикладные протоколы, такие как HTTP, FTP, TELNET и другие, могут работать поверх протокола SSL совершенно прозрачно. Протокол SSL может согласовывать алгоритм шифрования и ключ сессии, а также аутентифицировать сервер до того, как приложение примет или передаст первый байт данных. Все протокольные прикладные данные передаются зашифрованными с гарантией конфиденциальности. Протокол SSL предоставляет безопасный канал, который имеет три основных свойства.

1 Канал является частным. Шифрование используется для всех сообщений после простого диалога, который служит для определения секретного ключа.

2 Канал аутентифицирован. Серверная сторона диалога всегда аутентифицируется, в то время как клиентская – аутентифицируется опционно.

3 Канал надежен. Транспортировка сообщений включает в себя проверку целостности.

Протокол SSL использует следующие криптографические алгоритмы: DES, DSA, MD5, RC2, RC4.

Вопросы для контроля

1 Перечислите основные характеристики DoS-атаки.

2 Что называют DDoS-агентом?

3 Перечислите сетевые протоколы, используемые в DoS-атаках.

4 Охарактеризуйте основные протоколы, применяемые для криптографической защиты в сетях.

Лабораторная работа № 3. Разработка программы аутентификации пользователя на основе клавиатурного почерка

Цель работы: изучение методов аутентификации пользователя на основе клавиатурного почерка.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования.
4 Выводы.

Основные теоретические положения

Аутентификация пользователя, построенная на непрерывном динамическом анализе клавиатурного почерка и работы с мышью, имеет существенные преимущества по сравнению с другими методами биометрической идентификации в связи с простотой реализации, высокой надежностью, точностью, незаметностью для испытуемого и не требует применения дорогостоящего оборудования.

Одной из достаточно сложных задач, повседневно решаемых многими людьми, является быстрый ввод текстов с клавиатуры компьютера. Обычно быстрого клавиатурного ввода информации удается достичь за счет использования всех пальцев обеих рук, при этом у каждого человека появляется свой уникальный клавиатурный почерк. Под этим понятием понимается система индивидуальных особенностей начертаний и динамики воспроизведения букв, слов и предложений на клавиатуре. Следует заметить, что применение способа идентификации по клавиатурному почерку целесообразно только по отношению к пользователям с достаточно длительным опытом работы с компьютером и сформировавшимся почерком работы на клавиатуре, т. е. к программистам, секретарям и т. д.

При оценке клавиатурного почерка пользователей анализируются следующие характеристики: время нажатия клавиш; время между нажатиями клавиш; индивидуальные особенности работы, такие как использование служебных клавиш, работа с цифровым регистром.

Анализ эффективности данного метода аутентификации показывает, что значения ошибки второго рода (порядка 0,03…0,05) являются недостаточными для использования в системах допуска пользователей, но вполне применимы для поддержки принятия решения администратором о легальности пользователя.

Данный способ включает в себя три основные функции: сбор информации; обработка информации (механизмы сравнений с эталонными значениями); принятие решений по результатам аутентификации.

Существуют следующие алгоритмы распознавания:

– на основе геометрических методов распознавания, использующих различные меры близости предъявляемого вектора *V* к биометрическому эталону *VC* (мера Хэмминга, евклидова мера и др.);

– на основе применения искусственных нейронных сетей (ИНС).

Геометрические методы распознавания наиболее просты, системы, их реализующие, обладают высоким быстродействием, однако ошибки аутентификации для таких систем могут оказаться неприемлемо большими. Это обусловлено тем, что используемые в геометрических методах меры близости фактически

не учитывают конфигурацию областей распределения векторов биометрических параметров.

Широкое распространение получили две модели геометрических методов аутентификации пользователей: с мультипликативным способом сравнения биометрических характеристик; с аддитивным способом сравнения биометрических характеристик.

Построение первой модели биометрической аутентификации заключается в анализе отношений вновь образованных биометрических характеристик к соответствующим эталонным значениям, т. е. времени удержаний клавиш к своим эталонам и времени интервалов между нажатиями клавиш к соответствующим своим эталонным значениям.

Механизм аддитивного сравнения характеристик заключается в том, что из интервалов между нажатиями клавиш одной матрицы вычитаются соответствующие эталонные значения другой матрицы. Если значение (результат) меньше нуля, то сравниваемое время меньше условно-эталонного, а если больше – то больше. Отклонение от эталонного значения будет приниматься в процентах, причем отклонение является положительным, если отношение больше нуля, и отрицательным – в противном случае.

После получения результатов аддитивной характеристики все отклонения, которые лежат в пределах допустимых значений отклонений, обнуляются, а отклонения, которые остались за допустимыми пределами, остаются без изменений и выступают в качестве так называемых пиков аддитивной характеристики. Эта процедура называется аддитивным фильтром.

Методы, основанные на применении обучаемых нейронных сетей, потенциально обладают большей точностью, но им присущи две группы принципиальных проблем: собственные проблемы искусственных нейронных сетей, связанные с возможностью возникновения неопределенно долгого процесса обучения, тупиков, состояния «паралича», а также проблемы, определяемые биометрической природой распознаваемых образов, главная из которых – обучение нейронной сети идентификации всех возможных «чужих» пользователей (невозможность формирования представительной обучающей выборки для всех возможных «чужих»).

На практике особенности компьютерного почерка используют в качестве дополнительной степени защиты при вводе парольной фразы.

Отпечаток пальца, радужка глаза и т. п. – каждый из этих методов имеет свою степень надежности и требует покупки определенного оборудования, а следовательно, и затрат. Преимущество защиты на основе распознавания клавиатурного почерка состоит в том, что данный способ не требует дополнительного оборудования и проверка может происходить в незаметном для пользователя, а значит и злоумышленника, режиме.

Для оптимальной идентификации парольная фраза должна быть легко запоминаемой и содержать более 20 нажатий на клавиши. Однако системы, имеющиеся на рынке, работают и с более короткими парольными фразами (следует учесть, что в парольную фразу обычно входят и логин, и пароль). Биометриче-

ский эталон ввода получают на основе статистической обработки контролируемых параметров.

Использование метода в паре с парольной или ключевой аутентификацией позволяет гибко настраивать систему безопасности, обеспечивая высокий уровень надежности. Наиболее перспективно применение данного метода при использовании удаленного рабочего места, в дистанционном обучении и в системах обнаружения несанкционированного доступа.

Вопросы для контроля

1 Какие характеристики применяются для анализа клавиатурного почерка?

2 В чем отличие мультипликативного и аддитивного геометрических способов распознавания?

3 Каким должен быть оптимальный объем данных для распознавания клавиатурного почерка?

Лабораторная работа № 4. Разработка программы голосовой аутентификации пользователя

Цель работы: изучение методов голосовой аутентификации пользователя.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.

2 Постановка задачи.

3 Результаты исследования.

4 Выводы.

Среди различных биометрических систем голосовая аутентификация имеет следующие преимущества:

– привычный для человека способ аутентификации;

– голос не отчуждаем от человека, «его невозможно забыть дома»;

– личность автора звучащей речи может быть определена без непосредственного контакта с пропускной системой (как это необходимо для отпечатка пальца, ладони, подписи), возможно использование телефонного канала;

– возможность проводить скрытую аутентификацию;

– для аутентификации не требуются сложные дорогостоящие считыватели

биометрической информации.

Для аутентификации пользователя необходимо решить следующие задачи:

– анализ уникальных индивидуальных признаков, характеризующих личность говорящего;

– обоснование и выбор этих признаков;

– обоснование и выбор методов аутентификации дикторов.

Аутентификация диктора – способ проверки подлинности, позволяющий достоверно убедиться в том, что субъект действительно является тем, за кого он себя выдает, на основании сравнения голоса с хранящимся в системе эталоном.

Под голосовой аутентификацией понимается следующая ситуация. Говорящий произносит фразу, а компьютеризированная система распознавания индивидуальных характеристик голоса должна подтвердить или отвергнуть индивидуальность говорящего. В принципе, произнести фразу может как истинный пользователь, так и злоумышленник. Задаваясь стоимостью возможных потерь в случае возможного несанкционированного доступа злоумышленника, можно (для данной системы) рассчитать вероятность, с которой система не должна пропускать чужого.

Задачей начального этапа аутентификации диктора по тембру голоса является преобразование в речевой сигнал генерируемых речевой системой человека звуков. Звук, как известно, представляет собой механические колебания, распространяющиеся в окружающей среде (средой распространения служит воздух). Давление звуковой волны воспринимается микрофоном и преобразуется им в электрический аналоговый сигнал.

Для дальнейшей обработки необходимо провести преобразование информационного образа речи из аналогового сигнала в дискретный. Эту задачу решает аналого-цифровой преобразователь. Аналого-цифровой преобразователь (АЦП) осуществляет дискретизацию и квантование речевого сигнала.

Дискретизация заключается в разбиении непрерывного сигнала на ряд дискретных отсчетов, каждый из которых представляет значение аналогового сигнала в соответствующий момент времени. Дискретизация позволяет сократить количество информации, подлежащей дальнейшей обработке, до необходимого минимума. Согласно теореме Котельникова, частота дискретизации f_0 должна быть, как минимум, в 2 раза выше максимальной частоты преобразуемого сигнала f_{max}. При меньшей частоте дискретизации начинает теряться информация, которая активно используется при распознавании. Особенно это важно для распознавания в условиях шумов. Но сильно увеличивать частоту дискретизации нет смысла: при незначительном увеличении полезной информации начинает увеличиваться количество бесполезной информации (шумов). На практике частоту дискретизации следует выбирать даже несколько больше, чем рекомендует теорема Котельникова, так как в теореме рассматривается идеализированный случай. Частотный диапазон речи находится в пределах 100…4000 Гц. Так как максимальная частота речи f_{max} = 4 кГц, то f_0 должна быть несколько больше, чем $3f_{max}$ = 12 кГц. На практике чаще всего используют частоту дискретизации f_0 = 22050 Гц.

Квантование заключается в округлении замеренного аналогового сигнала с точностью до младшего разряда АЦП. Таким образом, квантованный сигнал может принимать только фиксированные значения с шагом, равным цене младшего разряда, в то время как исходный сигнал был непрерывным и мог принимать любое значение. Необходимое количество разрядов АЦП n можно определить из выражения

$$D = 6n + 1{,}8, \quad (1)$$

где D – требуемый динамический диапазон, дБ.

Интенсивность звука во время речи изменяется примерно от 20 дБ (шепот) до 70 дБ (громкий разговор), таким образом, динамический диапазон может достигать 50 дБ. Исходя из этого, количество разрядов АЦП должно быть не менее 8 (на практике используют разрядность 16 бит).

Важнейшим параметром систем аутентификации является коэффициент надежности – вероятность ошибок 1-го и 2-го рода.

Ошибки возникают в результате того, что при сравнении реального идентификатора и идентификатора в базе данных существует сложность достижения идеального соответствия (100 % совпадения). Поэтому вводится порог, который однозначно должен определить процент соответствия, при котором идентификатор будет признан соответствующим. Введение такого порога может привести к ошибкам ложного нераспознавания или ложного опознавания:

– ошибка первого рода (*FRR – False Rejection Rate*) – «не узнать своего», т. е. принимается решение «чужой», хотя на самом деле субъект присутствует в списке зарегистрированных пользователей;

– ошибка второго рода (*FAR – False Acceptance Rate*) – «пропустить чужого», т. е. принимается решение «свой», хотя на самом деле субъект отсутствует в списке зарегистрированных пользователей.

Каждая данная система может перестраиваться таким образом, что ошибки одного рода могут быть уменьшены за счет увеличения ошибок другого рода (даже при сохранении всех других факторов, влияющих на вероятность ошибки: длительность и характер речевого сообщения, помехи и т. п.). Изменение соотношения ошибок первого и второго рода достигается за счет изменения порога принятия решения и выбора набора признаков.

Важнейшим элементом успешного распознавания дикторов является выбор информативных признаков (речевых параметров), способных эффективно представлять информацию об особенностях речи конкретного диктора.

К ним предъявляются следующие требования: эффективность представления информации об особенностях речи конкретного диктора; простота измерения; стабильность во времени; частое и естественное появление в речи; невосприимчивость к имитации.

Для обработки звуковых сигналов и устранения шумов широко применяют преобразование Фурье. Преобразование Фурье позволяет осуществить восстановление, разделение информационных потоков, подавление шумов, сжатие данных, фильтрацию и усиление сигналов. Например, приём сигнала на фоне

шума описывается в виде процедуры фильтрации сигнала посредством фильтра, при этом ставится задача максимально ослабить шумы и помехи и минимально исказить принимаемый сигнал.

Преобразование Фурье вычисляется всякий раз, когда мы слышим звук. Ухо автоматически выполняет вычисление, проделать которое наш сознательный ум способен лишь после нескольких лет обучения математике. Наш орган слуха строит преобразование, представляя звук – колебательное движение частиц упругой среды, распространяющееся в виде волн в газообразной, жидкой или твёрдых средах – в виде спектра последовательных значений громкости для тонов различной высоты. Мозг превращает эту информацию в воспринимаемый звук.

Вопросы для контроля

1 Перечислите преимущества и недостатки голосовой аутентификации.
2 Какого рода ошибки возникают при выполнении распознавания речи?
3 Почему для голосовой аутентификации возможно применение преобразования Фурье?

Лабораторная работа № 5. Основы MS Crypto API

Цель работы: изучение интерфейса программирования приложений для работы с криптопровайдерами.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.
2 Получить задание у преподавателя, выполнить типовые задания.
3 Сделать выводы по результатам исследований.
4 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования.
4 Выводы.

Основные теоретические положения

Открытые сети, такие как Интернет, не предоставляют средств обеспечения защищенного взаимодействия между объектами. При удаленном взаимодействии через такие сети может происходить чтение или даже изменение передаваемой информации неправомочными третьими лицами. Использование криптографии обеспечивает защиту данных от просмотра или изменения, а также создает защищенные каналы связи на основе незащищенных каналов.

Например, данные могут быть зашифрованы с помощью некоторого криптографического алгоритма, переданы в зашифрованном виде, а затем расшифрованы лицом, которому они предназначались. Если зашифрованные данные будут перехвачены третьим лицом, расшифровать их будет трудно.

CryptoAPI – интерфейс программирования приложений, который обеспечивает разработчиков Windows-приложений стандартным набором функций для работы с криптопровайдером. CryptoAPI поддерживает работу с асимметричными и симметричными ключами, то есть позволяет шифровать и расшифровывать данные, а также работать с электронными сертификатами.

Любой криптопровайдер должен экспортировать набор обязательных функций, которые формируют системный программный интерфейс CryptoAPI, при этом каждая из этих функций соответствует некоторой функции CryptoAPI. Также криптопровайдер должен обеспечивать:

– реализацию стандартного интерфейса криптопровайдера;

– работу с ключами шифрования, предназначенными для обеспечения работы алгоритмов, специфичных для данного криптопровайдера;

– невозможность вмешательства третьих лиц в схемы работы алгоритмов.

Приложения не работают напрямую с криптопровайдером. Вместо этого они вызывают функции CryptoAPI из библиотек Advapi32.dll и Crypt32.dll. Операционная система фильтрует вызовы этих функций и вызывает соответствующие функции CryptoAPI, которые непосредственно работают с криптопровайдером. Структура взаимодействия CryptoAPI и криптопровайдеров показана на рисунке 1.

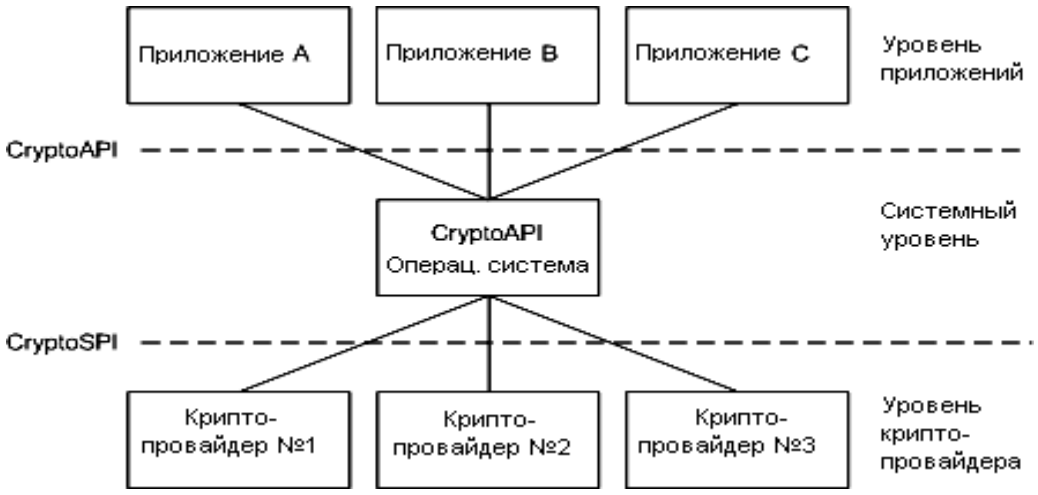

Рисунок 1 – Структура взаимодействия CryptoAPI и криптопровайдеров

Минимальный состав криптопровайдера — одна DLL. Обычно эта библиотека хранится в папке \WINDOWS\system32\. Обязательным является контроль целостности этой DLL.

Кроме стандартных функций CryptoAPI, криптопровайдер обычно поддерживает ряд собственных функций. Если собственные функции не реализованы, то DLL действует, по сути, как промежуточный слой между операционной системой и исполнителем криптографических операций.

Пространства имен *System.Security* содержат классы, представляющие разрешения и систему безопасности *.NET Framework*. Дочерние пространства имен содержат типы, управляющие доступом к защищаемым объектам и их аудитом, обеспечивающие проверку подлинности, службы шифрования, управляющие доступом к операциям и ресурсам на основе политик и поддерживающие управление правами для содержимого, создаваемого в приложениях.

Классы в пространстве имен *System.Security.Cryptography* платформы *.NET Framework* управляют многими деталями шифрования. При использовании этих классов вовсе не обязательно быть экспертом в криптографии. При создании нового экземпляра одного из классов, реализующих алгоритмы шифрования, ключи создаются автоматически с целью удобства использования, а принятые по умолчанию значения свойств призваны обеспечить максимальную защищенность.

Пространство имен *System.Security.Cryptography* содержит набор классов, которые позволяют осуществлять симметричное и асимметричное шифрование, хеширование, а также обеспечивают генерацию псевдослучайных чисел. Успешное применение криптографии основывается на совместном использовании этих операций. Основные компоненты пространства имен приведены в таблице 1.

Таблица 1 – Классы пространства имен System.Security

Пространство имен	Описание
System.Security.AccessControl	Пространство имен System.Security.AccessControl содержит программные элементы, обеспечивающие управление доступом к защищаемым объектам и аудит операций, связанных с безопасностью этих объектов
System.Security.Authentication	Пространство имен Authentication предоставляет ряд перечислений, описывающих безопасность подключения
System.Security.Cryptography	Пространство имен System.Security.Cryptography предоставляет криптографические службы, включающие безопасное кодирование и декодирование данных, а также целый ряд других функций, таких как хеширование, генерация случайных чисел и проверка подлинности сообщений
System.Security.Cryptography.Pkcs	Пространство имен System.Security.Cryptography.Pkcs содержит программные элементы, обеспечивающие поддержку стандартов шифрования с открытым ключом (Public Key Cryptography Standards, PKCS), в том числе методы подписывания данных, обмена ключами, запроса сертификатов, шифрования и расшифровки с открытым ключом и другие функции обеспечения безопасности

При создании экземпляра одного из конкретных классов исходные конструкторы всегда записывают в параметры объекта разумные и безопасные значения (если это возможно). Так, алгоритмы асимметричного шифрования, опирающиеся на криптографию с открытым ключом, генерируют случайную пару ключей, а алгоритмы симметричного шифрования – случайный ключ и вектор инициализации.

Вопросы для контроля

1 Какие средства защиты от несанкционированного доступа предоставляет CryptoAPI?

2 Перечислите основные функции криптопровайдеров.

3 Какие классы пространства имен System.Security позволяют осуществлять симметричное и асимметричное шифрование?

4 Какие классы пространства имен System.Security позволяют осуществлять хеширование?

Лабораторная работа № 6. Изучение распространенных типов уязвимостей в программном обеспечении и механизмов соответствующих атак на них

Цель работы: изучение типов уязвимостей в программном обеспечении и механизмов атак.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.

2 Постановка задачи.

3 Результаты исследования.

4 Выводы.

Основные теоретические положения

Уязвимость ассоциируется с нарушением политики безопасности, вызванным неправильно заданным набором правил или ошибкой в обеспечивающей безопасность компьютера программе. Стоит отметить, что теоретически все компьютерные системы имеют уязвимости. Но то, насколько велик потен-

циальный ущерб от вирусной атаки, использующей уязвимость, позволяет подразделять уязвимости на активно используемые и не используемые вовсе.

Уязвимость – это состояние вычислительной системы (или нескольких систем), которое позволяет исполнять команды от имени другого пользователя; получать доступ к информации, закрытой для данного пользователя; показывать себя как иного пользователя или ресурс; производить атаку типа «отказ в обслуживании».

Открытость – это состояние вычислительной системы (или нескольких систем), которое не является уязвимостью, но позволяет атакующему производить сбор защищенной информации; позволяет атакующему скрывать свою деятельность; содержит возможности, которые работают корректно, но могут быть легко использованы в неблаговидных целях; является первичной точкой входа в систему, которую атакующий может использовать для получения доступа или информации.

Когда хакер пытается получить неавторизованный доступ к системе, он производит сбор информации (расследование) о своем объекте, собирает любые доступные данные и затем использует слабость политики безопасности («открытость») или какую-либо уязвимость. Существующие уязвимости и открытости являются точками, требующими особенно внимательной проверки при настройке системы безопасности против неавторизованного вторжения.

SQL-инъекции – встраивание вредоносного кода в запросы к базе данных – наиболее опасный вид атак. С использованием *SQL*-инъекций злоумышленник может не только получить закрытую информацию из базы данных, но и при определенных условиях внести туда изменения.

Уязвимость по отношению к *SQL*-инъекциям возникает из-за того, что пользовательская информация попадает в запрос к базе данных без должной обработки: чтобы скрипт не был уязвим, требуется убедиться, что все пользовательские данные попадают во все запросы к базе данных в *экранированном* виде. Требование *всеобщности* и является краеугольным камнем: допущенное в одном скрипте нарушение делает уязвимой всю систему.

***Подготовка к написанию программного кода* (*установка необходимых приложений*).** Необходимо установить себе на компьютер локальный сервер (рисунок 2). Локальный сервер позволит работать со своим проектом на локальном компьютере, т. е., не имея возможности выйти в Интернет, сайт будет нам доступен.

Рисунок 2 – Ярлыки для управления локальным сервером

Теперь понадобится программа, в которой будем писать программный код (*php, Mysql*). Воспользуемся последним приложением – *Adobe Dreamweaver CS3*.

Реализация кода. Создадим папку на локальном сервере для будущего проекта. Для этого идем на *local_server_disk_name:\home\localhost\www* в проводнике и здесь создаем папку для проекта, например, *myproject*. Конечный путь к папке с проектом будет иметь вид:

local_server_disk_name:\home\localhost\www\myproject

Теперь запускаем установленный локальный сервер и затем *AdobeDreamweaver CS3*. В главном меню программы выбираем пункт меню *Dreamweaver Site*.

Переходим на вкладку *Advanced*. В поле *SiteName* указываем имя проекта, например, *mysite*. В поле *Localrootfolder* указываем путь к недавно созданной папке проекта. Это *local_server_disk_name:\home\localhost\www\myproject*.

Нажимаем кнопку *OK*.

Для создания первого файла можно воспользоваться главным меню программы и в колонке *Create new* выбрать тип создаваемого файла *PHP* или же создать файл через пункты меню *File ->New*, выбрать тип *PHP* и нажать кнопку *Create*.

Сразу сохраним файл (*File ->SaveAs*…) под именем *authorize* (расширение *php* к имени файла добавится автоматически, но можно указать и принудительно).

Рассмотрим несколько подробнее исходный текст созданного *php*-файла. Сейчас он имеет вид:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Untitled Document</title>
</head>
<body>
</body>
</html>
```

Весь последующий код будем писать между тэгами <*body*> и </*body*>. Для начала следует проверить работоспособность сервера. Пропишем следующий код после строки <*body*>:

```
<?
  echo “Проверка работоспособности сервера”;
?>
```

Конструкция <? ?> означает границы *php*-скрипта. Оператор *echo* является оператором вывода в *php*. Теперь получаем, что если обработчик *php*-кода на сервере работает исправно, то в браузере должно выводиться сообщение «Проверка работоспособности сервера».

Открываем браузер *Opera*, в адресной строке прописываем *http://localhost/myproject/authorize.php*, нажимаем *Enter*. Если всё сделано пра-

вильно, увидим соответствующее сообщение (рисунок 3).

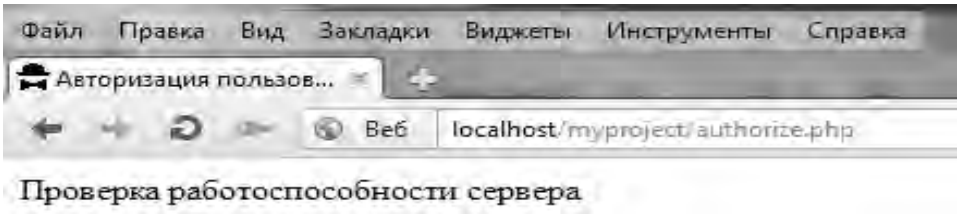

Рисунок 3 – Проверка работоспособности сервера

Создадим форму для заполнения пользователем данных для авторизации. Будем использовать для этого следующие элементы управления (рисунок 4).

1 – создать новую форму; *2* – создать текстовое поле; *3* – создать кнопку

Рисунок 4 – Создание новой формы

Сейчас необходимо создать ещё один файл-обработчик, который будет обрабатывать введенные пользователем данные в форму. Создаем новый *php*-файл и сохраняем его как *welc.php*.

Можем удалить строку вывода фразы для проверки работы сервера и конструкцию <? ?>. Нажимаем кнопку создания новой формы. В поле Action указываем путь к файлу-обработчику. Поскольку он находится в корневой папке проекта (в каталоге 1-го уровня), указываем просто имя файла *welc.php*. В выпадающем списке Method выбираем тип передачи переменных *POST*. Нажимаем кнопку *OK*. Для удобства код формы приводим к такому виду:

```
<form action="welc.php" method="post">
</form>
```

Помещаем курсор внутрь полученной структуры и нажимаем кнопку создания *текстового поля*. В поле *Name* вводим имя будущей переменной (одновременно это и имя создаваемого текстового поля). Укажем имя name. Нажимаем *OK*. Видим появившуюся строку, которая создаст *textbox*. Для наглядности и удобства пользователя выше этой строки напишем следующее:

```
<p>Введите Ваш логин:
<input name="name" type="text">
</p>
```

Тэги * * говорят о том, что содержимое между ними находится в одном абзаце.

Проделываем то же самое для пользовательского пароля, только в поля Name впишем имя переменной, например, *pass*.

Теперь необходимо создать кнопку, инициирующую вызов файла-обработчика и передающую ему введенные данные. Нажимаем кнопку под номером *3* (см. рисунок 4). Выбираем *Type* (тип) как *submit*. В поле *Name* пишем тоже *submit*.

Таким образом, конечный код формы будет иметь вид:

```
<form action="welc.php" method="post">
<p> Введите Ваш логин:
<input name="name" type="text">
</p>
<p>
   Введите Ваш пароль:
<input name="pass" type="text">
</p>
<p>
<input name="submit" type="submit" value="Отправить">
</p>
</form>
```

Сохраняем файл, переключаемся в браузер и обновляем страницу. Если всё сделано верно, будет видна созданная веб-страница для авторизации (рисунок 5).

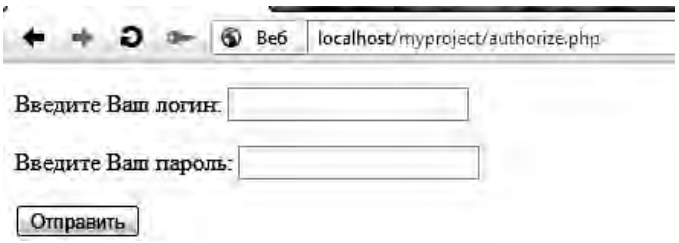

Рисунок 5 – Окно авторизации пользователя

Создание БД и таблицы данных. В адресной строке браузера прописываем *http://localhost/phpmyadmin*. В единственном текстовом поле «Создать новую БД» указываем имя базы данных и нажимаем кнопку *Создать*.

Создаем единственную таблицу, где будут храниться пользовательские данные. Имя вводим *users*, количество полей – три. Нажимаем кнопку *Пошел*.

Вводим необходимую информацию по полям (рисунок 6).

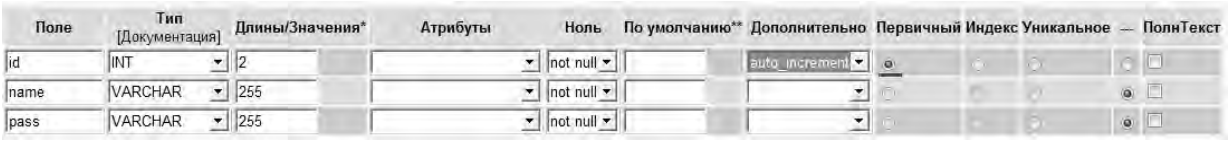

Рисунок 6 – Ввод информации по полям создаваемой таблицы

Нажимаем кнопку *Сохранить*.

Теперь необходимо создать пользователя для базы. В *phpmyadmin* нажимаем ссылку «*К началу*» (рисунок 7).

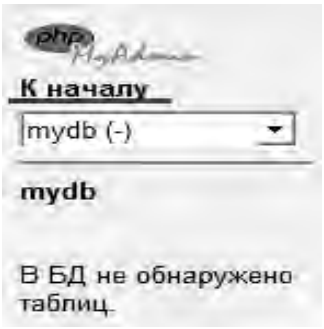

Рисунок 7 – Создание пользователя для базы данных

Далее идем по ссылкам *Привилегии ->Добавить нового пользователя*. Имя пользователя указываем *myroot*, хост выбираем *local*, пароль сделаем «111» (для простоты). Галочки привилегий ставим все, сейчас права доступа не слишком важны. Страница добавления нового пользователя в результате должна иметь вид (рисунок 8).

Рисунок 8 – Страница добавления нового пользователя

Для сохранения изменений нажимаем кнопку *Пошел*.

Теперь нужно добавить в таблицу *users* данные хотя бы одного пользователя. Для этого выбираем в списке базу (рисунок 9).

Кликаем левой кнопкой мыши по единственной таблице *users*, идем на вкладку *Вставить*. Поле *id* оставляем пустым (оно будет заполняться автоматически по мере появления новых записей в таблице, так как ему присвоено свойство автоинкримента при создании таблицы). Заполняем таблицу, как показано на рисунке 10.

Для сохранения изменений нажимаем кнопку *Пошел*.

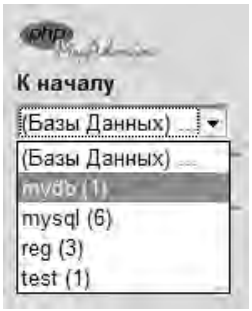

Рисунок 9 – Добавление в таблицу нового пользователя

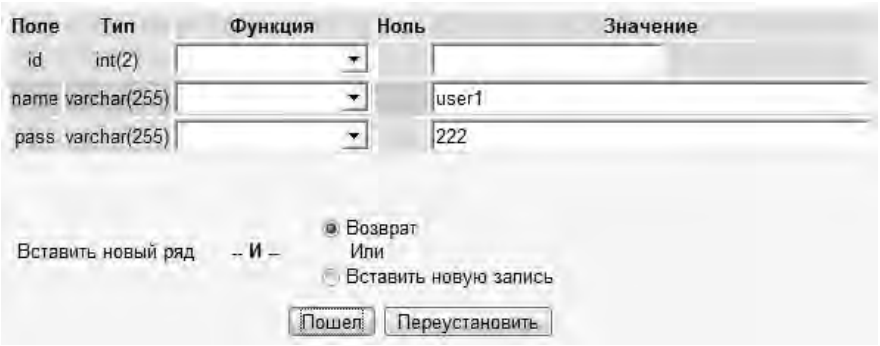

Рисунок 10 – Заполнение таблицы

Теперь, если откроем вкладку *Обзор*, то убедимся в том, что в таблице *users* действительно появилась запись о пользователе (рисунок 11).

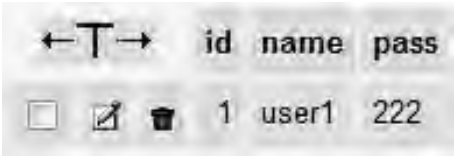

Рисунок 11 – Просмотр содержимого таблицы

Возвращаемся в *AdobeDreamweaver* и открываем файл *welc.php*. Конечный код файла с комментариями представлен ниже:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Добропожаловать</title>
</head>
<body>
<?
/*Создаем идентификатор подключения к локальному серверу пользователем myroot с паролем доступа 111*/
$db = mysql_connect ("localhost", "myroot", "111");
/*Выбираембазу mydb*/
mysql_select_db ("mydb", $db);
```

```
/*Существует ли идентификатор подключения к серверу?*/
if(isset($db))
  echo "Подключение совершилось!";
else
  echo "Не удалось подключиться к базе!";
/*Присваиваем переменной name значение текстового поля name на нашей форме*/
$name  = $_POST['name'];
/*Если переменная оказалось пустой*/
if ($name == '')
/*уничтожаем ее*/
  unset($name);
$pass  = $_POST['pass'];
if ($pass == '')
  unset($pass);
  /*Если существуют переменные name и pass*/
if((isset($name))&&(isset($pass)))
{/*Запрос на выборку записей из таблицы*/
$result = mysql_query("SELECT * FROM users WHERE name = '$name' AND pass =
'$pass'");
/*Есть ли хоть одна запись в запросе?*/
  if(mysql_num_rows($result) > 0)
        echo "<br>Добро пожаловать в систему!";
  else
        echo "<br>Пользователя с такими данными не существует!";}
else
  echo "<br>Ошибка! Заполнены не все поля формы!";
?>
</body>
</html>
```


Сохраняем изменения в файле и теперь можем проверить работу скрипта. Открываем браузер и прописываем в строке адреса *http://localhost/myproject/authorize.php*, нажимаем *Enter*. Вводим в текстовые поля данные пользователя (рисунок 12).

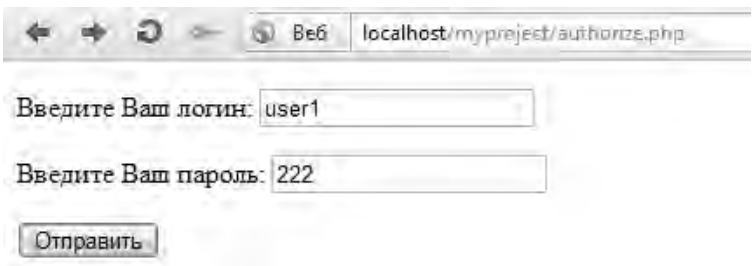

Рисунок 12 – Пример авторизации

Нажимаем кнопку *Отправить* и смотрим за результатом выполнения скрипта (рисунок 13).

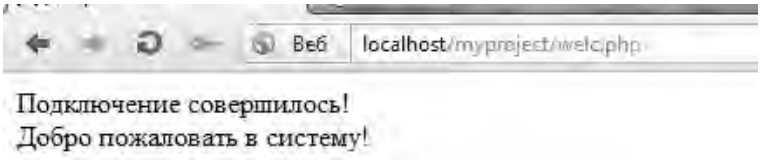

Рисунок 13 – Результат авторизации

Как видим, подключение к серверу прошло удачно и система позволила войти, так как пользовательские данные введены верно.

Способы защиты от sql-инъекции. Чтобы защитить текстовые поля от ввода в них вредоносных скриптов, можно использовать *php*-функцию *str_replace()*. Синтаксис данной функции следующий:

mixed str_replace(mixed search, mixed replace, mixed subject[, int &count])

Эта функция возвращает строку или массив *subject*, в котором все вхождения *search* заменены на *replace*. Применимо к данному случаю функции в качестве проверки на посторонний скрипт можно записать следующим образом:

```
$_POST['name']=str_replace(" ' ", "", $_POST['name);'
$_POST['pass']=str_replace(" ' ", "", $_POST['pass']);
```

Обе эти проверки следует поместить сразу перед запросом к базе, т. е. конечный вариант кода в файле *welc.php* будет выглядеть следующим образом:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Добро пожаловать</title>
</head>
<body>
<?
/*Создаем идентификатор подключения к локальному серверу пользователем myroot с паролем доступа 111*/
$db = mysql_connect ("localhost", "myroot", "111");
/*Выбираембазу mydb*/
mysql_select_db ("mydb", $db);
/*Существует ли идентификатор подключения к серверу?*/
if(isset($db))
  echo "Подключение совершилось!";
else
  echo "Не удалось подключиться к базе!";
/*Присваиваем переменной name значение текстового поля name на нашей форме*/
$name = str_replace(" ' ", "", $_POST['name']);
$pass = str_replace(" ' ", "", $_POST['pass']);
//*Если переменная оказалось пустой*/
if ($name == '')
/*уничтожаем ее*/
  unset($name);
```

```
$pass  = $_POST['pass'];
if ($pass == '')
  unset($pass);
  /*Если существуют переменные name и pass*/
if((isset($name))&&(isset($pass)))
{/*Безопасностьлогина*/
  $_POST['name']=str_replace("'", "", $_POST['name']);
  /*Безопасностьпароля*/
  $_POST['pass']=str_replace("'", "", $_POST['pass']);
  /*Запрос на выборку записей из таблицы*/
  $result = mysql_query("SELECT * FROM users WHERE name = '$name' AND pass =
'$pass'");
  /*Есть ли хоть одна запись в запросе?*/
  if(mysql_num_rows($result) > 0)
        echo "<br>Добро пожаловать в систему!";
  else
        echo "<br>Пользователя с такими данными не существует!";
}
else
  echo "<br>Ошибка! Заполнены не все поля формы!";
?>
</body>
</html>
```

Если теперь ввести данные некорректно, программа об этом сообщит.

Вопросы для контроля

1 Объясните различие понятий «уязвимость» и «открытость» применительно к вычислительной системе.

2 Что такое *SQL*-инъекции?

3 Каким образом можно построить защиту от *SQL*-инъекции?

Лабораторная работа № 7. Изучение различных сетевых атак, методов взлома паролей и механизмов защиты от атак

Цель работы: изучение видов сетевых атак и приобретение навыков использования механизмов защиты от них.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования.
4 Выводы.

Основные теоретические положения

Сетевые атаки столь же разнообразны, как и системы, против которых они направлены. Некоторые атаки отличаются большой сложностью. Другие может осуществить обычный оператор, даже не предполагающий, какие последствия может иметь его деятельность. Для оценки типов атак необходимо знать некоторые ограничения, изначально присущие протоколу *TPC/IP*. Сеть Интернет создавалась для связи между государственными учреждениями и университетами в помощь учебному процессу и научным исследованиям. Создатели этой сети не подозревали, насколько широко она распространится. В результате в спецификациях ранних версий Интернет-протокола (*IP*) отсутствовали требования безопасности. Именно поэтому многие реализации *IP* являются изначально уязвимыми. Через много лет, получив множество рекламаций (*RFC – RequestforComments*), стали внедрять средства безопасности для *IP*. Однако ввиду того, что изначально средства защиты для протокола *IP* не разрабатывались, все его реализации стали дополняться разнообразными сетевыми процедурами, услугами и продуктами, снижающими риски, присущие этому протоколу.

Рассмотрим типы атак, которые обычно применяются против сетей *IP*, и перечислим способы борьбы с ними.

Взломщик паролей – программа, которая может расшифровывать пароли или каким-либо другим способом снимать парольную защиту. Если механизмы парольной защиты используют слабое шифрование, то иногда можно восстановить изначальный пароль или подобрать другой, который считается верным. В противном случае взломщики паролей могут использовать метод подбора, который проверяет одно слово за другим, часто с достаточно большой скоростью.

Каковы основные методы взлома? Эти методы основаны на уязвимостях, которые существуют в криптоалгоритмах, и их реализации. В случае абсолютно слабого алгоритма или вопиющих ошибок в его исполнении может быть использован **«метод изменения одного байта»**. Тогда код программы модифицируется таким образом, что независимо от введенного пароля программа считает его правильным. Удивительно, но такие программы все еще существуют.

При слабых алгоритмах или неправильном использовании сильных можно применять другие простые методы восстановления паролей. Они различаются в зависимости от конкретного приложения, но суть состоит в том, чтобы **значительно уменьшить количество возможных паролей, основываясь на имеющейся дополнительной информации**.

Для стойких алгоритмов (когда атакующий может только генерировать и проверять пароли) существуют два основных метода: **атака перебором и по словарю**. Атака перебором используется тогда, когда нет никакой дополнительной

информации о пароле и атакующий просто пробует все возможные пароли – 1-символьные, 2-символьные и т. д. Чтобы противостоять этой атаке, криптосистема должна поощрять длинные смешанные пароли и иметь долгое время установки ключа (или время отклика), что значительно снижает скорость перебора.

Если взломщик знает, что пароль – это некое существующее слово, он может использовать атаку по словарю. Тогда в качестве паролей-кандидатов проверяются только слова из словаря. В словаре содержится менее 100 000 слов, так что их можно проверить очень быстро – в большинстве случаев это занимает всего несколько секунд.

Сочетание двух описанных выше атак называется **«атака по слогам»**. Она используется в том случае, если пароль искажен или представляет собой несуществующее слово. Тогда взломщик может объединять слоги, чтобы подобрать нужное слово.

Самая мощная атака – **«атака на основе правил»**. Она может быть использована в тех случаях, когда взломщик обладает какой-либо информацией о пароле, который он хочет взломать. Например, ему известно, что пароль состоит из слова и одно- или двузначного числа. Он пишет правило, и программа генерирует только подходящие пароли (user1, mind67, snapshot99 и т. д.). Или другой пример: атакующий знает, что первая буква в верхнем регистре, вторая – гласная и что пароль не длиннее шести символов. Такая информация может уменьшить количество возможных паролей в сотни раз. Этот метод включает все атаки – перебором, по словарю и по слогам.

И, наконец, некоторые слабые алгоритмы позволяют использовать атаку **«по известному открытому тексту»**. Это означает, что взломщик имеет несколько файлов или фрагментов файлов в расшифрованном виде и хочет расшифровать другие.

Чем длиннее ключ, тем сложнее его вскрыть – если 40-битный ключ взломать под силу любому из нас перебором на современном домашнем компьютере за несколько дней, то для взлома 64-битных ключей необходимо объединять мощность многих компьютеров в сети Интернет и нужны уже будут месяцы.

Способы получения паролей для почтовых ящиков. Один из способов получения пароля для почтового ящика – использование специальных программ – «брутов». Для ознакомления рассмотрим пример восстановления пароля для почтового ящика *momail@list.ru* с паролем «*murzila*»:

– должно быть установлено соединение с Интернетом;

– запускаем программу для восстановления паролей;

– в поле «Логин» вводим почтовый адрес. Например, *momail@list.ru*;

– в поле «Словарь паролей» прописываем путь к текстовому файлу с вариантами паролей для перебора;

– вводим адрес сервера (электронная почта находится на сервере pop.mail.ru);

– нажимаем «Начать», проверяется каждый из вариантов паролей;

– при нахождении верного пароля соединение разрывается и в поле «Пароль» виден результат;

– если же из файла с паролями ни один не подойдет, поле «Пароль» останется пустым.

Вопросы для контроля

1 Что такое взломщик паролей?
2 Какие методы взлома паролей существуют? Назовите их.
3 Что такое «брут»?

Лабораторная работа № 8. Исследование средств защиты информации и идентификации пользователей в ОС Windows

Цель работы: изучение средств защиты информации и идентификации пользователей в ОС Windows.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.
2 Получить задание у преподавателя, выполнить типовые задания.
3 Сделать выводы по результатам исследований.
4 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования.
4 Выводы.

Основные теоретические положения

Одна из главных задач системного администратора *Windows* состоит в защите информации, которая хранится в базе данных учетных записей пользователей (*Security Account Management Database*, сокращенно *SAM*) от несанкционированного доступа. Эта база имеется на каждом компьютере с ОС *Windows*. В ней хранится вся информация, используемая для аутентификации пользователей *Windows* при интерактивном входе в систему и при удаленном доступе к ней по компьютерной сети. Для этого необходимо:

– ограничить физический доступ к компьютерам сети и прежде всего к контроллерам доменов;

– установить пароли *BIOS* на включение компьютеров и на изменение их настроек *BIOS*;

– отключить загрузку компьютеров со сменных носителей (компакт-дисков и флэшек);

– обеспечить контроль доступа к файлам и папкам *Windows* – системный

раздел жесткого диска должен иметь формат *NTFS*;

– следить за тем, где и как хранятся диски аварийного восстановления (*Emergency Repair Disks*) и архивные копии, если на последних присутствует дубликат системного реестра *Windows*.

Кроме того, если компьютер с *Windows* входит в домен, то по умолчанию имена и хешированные пароли последних десяти пользователей, регистрировавшихся на этом компьютере, сохраняются (кэшируются) в его локальном системном реестре (в разделе *SECURITY\Policy\Secrets* раздела *HKEY_LOCAL_MACHINE*). Чтобы отменить кэширование паролей на компьютерах домена, нужно с помощью утилиты *REGEDIT32* в разделе *Microsoft\WindowsNT\CurrentVersion\Winlogon* раздела *HKEY_LOCAL_MACHINE* добавить параметр *CashedLogonsCount*, установив его значение равным нулю, а тип – *REG_SZ*.

Для повышения стойкости паролей пользователей операционной системы *Windows* к взлому рекомендуется с помощью утилиты *Диспетчер пользователей* (*User Manager*) установить длину пользовательских паролей не менее восьми символов и активизировать режим устаревания паролей, чтобы пользователи периодически их обновляли. Чем выше вероятность атак на парольную защиту *Windows*, тем короче должен быть срок такого устаревания. А чтобы пользователи не вводили свои старые пароли повторно, необходимо включить режим хранения некоторого числа ранее использовавшихся паролей.

Вопросы для контроля

1 Что рекомендуется выполнять для повышения стойкости паролей пользователей?

2 Какие разделы реестра отвечают за настройки безопасности операционной системы Windows?

3 Какие действия необходимо предпринять для обеспечения безопасности при интерактивном входе в систему?

Список литературы

1 **Васильев, В. И.** Интеллектуальные системы защиты информации : учебное пособие / В. И. Васильев. – 2-е изд., испр. – Москва : Машиностроение, 2013. – 172 с.

2 **Новиков, В. А.** Информационные системы и сети. С электронным приложением : учебное пособие / В. А. Новиков, А. В. Новиков, В. В. Матвеенко. – Минск : Изд-во Гревцова, 2014. – 448 с.