Минобрнауки России
федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Омский государственный университет им. Ф.М. Достоевского»
Кафедра кибернетики

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Гуц А.К.
«___» _____ 2015 г.

ЗАЩИТА ЛИЧНЫХ ДАННЫХ ПОЛЬЗОВАТЕЛЯ В БРАУЗЕРАХ

Курсовой проект
по направлению 10.05.01 – Компьютерная безопасность

Научный руководитель:
к.ф.-м.н., доцент
_____ Лаптев А.А
«___» _____ 2015 г.

Выполнил:
студент группы СБС-001-О
_____ Сиганов И.Д
«____» _____ 2015 г.

Омск
2015

Содержание.

Введение.

Архитектура современных браузеров подразумевает, что пользователь браузера совпадает с пользователем операционной системы. Поэтому данные, хранящиеся в браузере, не предполагается защищать. За конфиденциальность всех данных отвечает операционная система. Таким образом данным браузером может пользоваться только один пользователь. На практике оказывается, что браузером в рамках одной сессии операционной системы может пользоваться несколько человек. Типичные пример - домашний компьютер или рабочая станция на предприятии, где по политике безопасности все компьютеры имеют определенные пароли, известные многим сотрудникам. В таких случаях пользоваться браузером с обычными настройками не безопасно, а стирать все данные после завершения работы не удобно.

В данной работе было проведено исследование браузеров Firefox и Chromium на предмет защиты локальных личных даных пользователей, а так же возможность использовать один браузер в одной сессии ОС многими пользователями с сохранение конфиденциальности всех данных в браузере. После чего был сделан вывод о невозможности обеспечить требуемый функционал в этих браузерах. Были предложены альтернативные способы защиты данных, а также исследована возможность модификации исходных кодов этих браузеров для достижения требуемых возможностей по обеспечению безопасности данных пользователя браузера.

Глава 1

1.1 О браузерах

Браузеры - это ПО для просмотра веб страниц. Существует огромное множество различных браузеров, наиболее популярные из них это Firefox, Chromium, Safari, Opera, Internet Explorer. Не смотря на огромное разнообразие все они решают одну основную задачу - обработку и отображение веб страниц. Благодаря существующим веб стандартам и рекомендациям W3C - консорциума всемирной паутины, производителям удается добиваться предсказуемости в визуальном отображении информации веб страниц. За последние несколько лет функционал браузеров вышел за рамки обычного рендеринга и стал дополнятся различными возможностями, позволяя веб сайтам получать доступ к функциям ОС, таким как определение координат по GPS, использованию функций WebGL, доступ к файловой системе, камере, микрофону. Также и диапазон проблем безопасности стал расширяться. Раньше он ограничивался только обеспечением безопасного рендеринга страниц, в том числе чтобы не дать исполнить вредоносный код на компьютере пользователя и получить контроль над браузером или всей ОС. Теперь же разработчикам приходится решать проблемы безопасности доступа к различным функциям ОС, описанным выше. Кроме того уделяется много внимания на установление безопасного соединения с серверами, защиту от кражи cookies, удаленное исполнение программ и так далее, что не предполагается рассматривать в рамках данной работы. Тем не менее существует еще один аспект безопасности браузеров, который стал тем более важен сейчас, в силу увеличения возможностей браузеров - это локальная защита данных пользователя. Далее в работе будут описано как именно хранят данные пользователя браузеры, какие риски это приносит, а так же будет предоставлен набор возможных решений.

1.2 Хранение данных пользователя

Браузеры хранят локально следующую информацию - пароли, HTTP cookies, кэш данных, историю посещенных страниц, базу подстановки строк в

поля ввода, закладки. С учетом нового API из стандарта html5, браузер хранит данные localstorage, WebSQL, IndexedDB. Вся эта информация будет составлять личные данные пользователя браузера.

Далее в работе будет произведен анализ двух браузеров с открытым исходным кодом - Firefox и Chromium. На данный момент они поддерживают большинство новых стандартов и рекомендаций W3C, имеют исчерпывающую документацию.

Для сравнения приведена таблица расположения каталогов пользователя для разных операционных систем в Firefox и Chromium, где %username% - это имя пользователя системы, а $hash - некоторая случайно сгенерированная строка.

	Chromium	Firefox
Linux	/home/%username%/.config/google-chrome/Default	/home/%username%/.mozilla/firefox/$hash.default/
Windows	C:\Users\%username%\AppData\Local\Google\Chrome\UserData\Default	C:\Users\%username%\AppData\Roaming\Mozilla\Firefox\Profiles\$hash.default
OsX	/Users/%username%/Library/Application Support/Google/Chrome/Default	/Users/%username%/Library/Application Support/Firefox/Profiles/$hash.default

Исходя из места расположения профиля пользователя, можно уже сделать несколько выводов. Chromium хранит данные в строго определенном месте всегда, в отличие от Firefox. Последний генерирует $hash при создании нового пользователя браузера. Из этого уже можно сделать вывод, что Firefox предполагает существование нескольких профилей с разными идентификаторами.

Так как браузеры хранят все данные пользователя на файловой системе в определенных файлах, то пользователь подвергается следующим опасностям: кража всех данных профиля каким либо вредоносным ПО или обычным физическим доступом к машине.

1.3. Хранение паролей пользователя в Chromium

Из всех данных, которые хранит браузер в профиле пользователя наиболее важными являются пароли и cookies. Для начала рассмотрим как разные браузеры хранят пароли.

Chromium хранит все пароли в файле «%PROFILE%/Login Data», который является sqlite[9] базой данных. Sqlite - это такая компактная встраиваемая база данных, т.е. она не требует сервер. Для работы с файлами базы данных программа должна использовать API библиотеки sqlite. Кроме паролей в аналогичных sqlite файлах так же хранятся cookies, websql, indexedb и много других данных. Sqlite поддерживает очень ограниченное количество типов данных - это целые числа, числа с плавающей запятой, текст и бинарные данные, чего достаточно для хранения пользовательских данных.

В файле «Login Data» хранятся адреса сайтов, соответственно логин-пароли, дата их создания и модификации, количество использований, url аватар и другая мета-информация. Этот файл можно открыть с помощью программы sqlitebrowser и извлечь всю информацию. В зависимости от операционной системы и некоторых других факторов, о которых будет сказано ниже, с помощью шифрования защищаются только пароли, остальная информация хранится всегда в открытом виде.

В операционных системах Windows, начиная с XP, Chromium для шифрования паролей всегда использует функцию DPAPI[12] CryptProtectData[3]. Эта функция использует данные Logon для определения пользователя который зашифровывает и расшифровывает данные. В общем случае получается, что зашифрованные данные можно будет расшифровать только на локальной машине и только от этого пользователя (точнее от пользователя с такими же данными logon), если дополнительный флаг CRYPTPROTECT_LOCAL_MACHINE не был установлен. Благодаря этому API в chromium появилась функция ввода пароля пользователя ОС, перед тем как показать текст пароля в настройках chrome://settings/passwords. Таким образом

chromium защищает пароли от кражи и удаленного взлома, но не от локального злоумышленника, которым может выступать вредоносное ПО или другие пользователи той же самой сессии ОС. Процедура получения паролей не составляет труда. Исходя из того, как chromium использует[11] CryptProtectData, был написан небольшой скрипт, получающий список паролей[приложение А] из sqlite базы «Login Data» с использованием функции winapi CryptUnprotectData.

В операционных системах Linux chromium по-умолчанию не использует никакого шифрования, т.е. пароли хранятся в открытом виде. Но если запущен и используется gnome keyring[9] - менеджер паролей, или аналогичное приложение, такое как kwallet, то ответственность за хранение данных в «Login Data» делегируется ему. В свою очередь такие менеджеры гарантируют безопасность хранения паролей, используя шифрование AES-128 и хэширование мастер-ключа. В этом случае в базе паролей chromium не хранится ничего.

Аналогично Linux в OsX ответственность за хранение паролей передается менеджеру связок ключей Apple Keychain.

Интересно отметить, что IE так же как и chromium в windows использует тот же самый набор функций для шифрования из DPAPI. Но за тем исключением, что в IE пароль дополнительно подсаливают хэшем SHA1 от url адреса с которым связана пара логина и пароля. Добавление соли защищает зашифрованные пароли от полного перебора в лоб, так как для восстановления пароля требуется знать еще и url с которым связан этот аккаунт. Данные хранятся в реестре windows.

1.4 Хранение паролей пользователя в Firefox

Firefox, в отличает от chromium, для хранения и шифрования паролей использует специальный набор библиотек NSS[4] - network security service, разработанный Mozila. Эта библиотека публикуется с открытыми исходными кодами и, по рекомендации Mozila, может использоваться в любых

приложениях. В firefox для шифрования и расшифрованная паролей используется специальный API из набора NSS под названием Secret Decoder Ring(SDR).

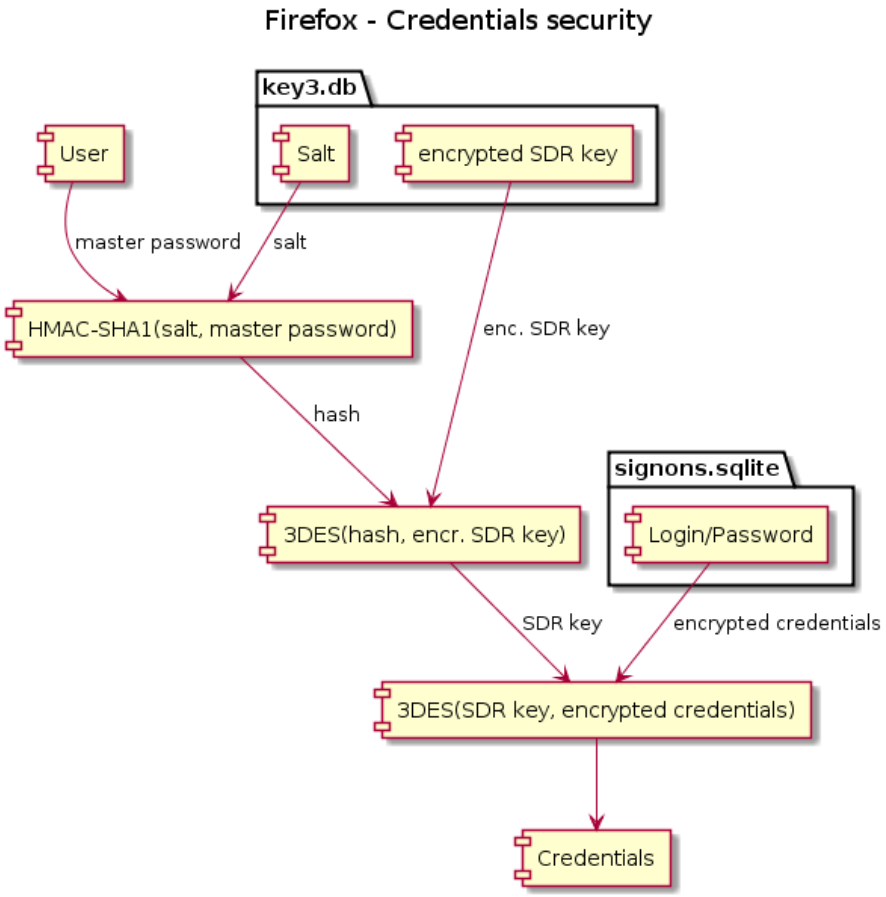

Рисунок 1: Диаграмма создания пользователя в Firefox

В отличие от chromium, при создании профиля пользователя firefox генерирует случайный хэш для идентификации пользователя. Поэтому название каталога с профилем состоит из случайных символов. Кроме того создается специальный файл key3.db, в котором хранится зашифрованный случайный SDR ключ с солью. SDR ключ будет использоваться для того, чтобы расшифровать по алгоритму 3DES зашифрованные пары логин-пароля, причем firefox шифрует и логин и пароль, в отличие от chromium, шифрующего только пароль. Сам SDR ключ тоже зашифрован с помощью 3DES, где в качестве ключа используется SHA1 хэш от мастер пароля, который задает пользователь при создании профиля, и соль из файла key3.db. На момент написании работы в firefox по-умолчанию мастер пароль не установлен, точнее мастер пароль

является пустой строкой. Но когда пользователь устанавливает новый мастер пароль, то firefox заново генерирует значения для SDR ключа и соли. Именно поэтому все ранее сохраненные пароли удаляются, их просто невозможно будет прочитать с новым SDR ключом.

Для наглядности, была составлена блок-схема, описывающая цикл расшифровки пользовательских данных, хранящихся либо в файле signons.sqlite, либо в logins.json, в зависимости от версии.

1.5 Оценка безопасности

Для того чтобы оценить, какой из двух подходов предоставляет наиболее безопасное средство хранения паролей, составим список угроз:

1. Вредоносное ПО может получить всю папку с данными пользователя браузера. Похитить пароли, cookies, файлы баз данных web-api. Проанализировать и скомпрометировать пользователя по истории браузера, кэшу и БД.
2. Любой пользователь ОС имея доступ к одной сессии может забрать весь каталог профиля и запустить у себя браузер от имени жертвы. Особенно актуально на рабочих местах, когда по политике безопасности предприятия на всех машинах строго определенные пароли и все их знают. Тогда средства шифрования, основанные на сессии строго говоря бессмысленны.

Исходя из угроз можно сделать вывод, что наиболее безопасным с точки зрения хранения паролей и логинов можно считать firefox. Так как он отвязан от функций ОС, то вредоносное ПО, запущенное от имени пользователя не сможет получить данные все равно, так как потребуется непосредственный ввод мастер пароля через форму самого браузера. Internet exlorer, с url-ом ассоциированным с паролем и логином используемым как соль, можно считать более защищенным чем chromium. В linux/osx у chromium есть возможность использовать сторонние программы для хранения конфиденциальной информации, что делает его не менее защищенным. Но несмотря на то, что и

firefox, и chromium используют шифрование паролей, существует огромное множество утилит, позволяющих восстановить пароли из профиля браузера[13, 14].

Тем не менее остается еще одна проблема - все браузеры так или иначе полагаются на то, что пользователь ОС совпадает с пользователем браузера, а это зачастую не так. В качестве примера могут выступать школьные или студенческие лаборатории, рабочие компьютеры, где не всегда возможно создавать профиль в ОС с шифрование домашнего каталога для каждого пользователя компьютера или такое противоречит политике безопасности предприятия. В таких случаях хотелось бы иметь возможность создавать профили в браузерах с шифрованием данных профиля.

Стоит заметить, что хоть браузеры и шифруют пароли, тем не менее другие данные, такие как cookies, история посещения страниц, история автозаполнения полей ввода, закладки, базы данных websql и indexedb остаются в открытом виде, хотя там тоже может содержаться конфиденциальная информация. В следующей главе предлагается рассмотреть как можно обойти эти недостатки.

Глава 2

2.1. Постановка проблемы и варианты решения

Итак, проблема — каталоги с пользовательскими данными во всех браузерах хранятся в строго определенных местах и не защищаются шифрованием полностью. Именно поэтому сами браузеры рекомендуют никогда не сохранять свои пароли и прочие данные при работе на неизвестных компьютерах. Все браузеры поддерживают режим анонимных вкладок, при закрытии которых все данные, связанные с ними, стираются. В firefox можно настроить такое поведение, чтобы он никогда не сохранял данные. То есть во время работы он хранит все в оперативной памяти, а после закрытия не выгружает их на жесткий диск в каталог профиля. Именно так и поступают все пользователи на компьютерах с неограниченным общим доступом - удаляют после себя все данные. Такой подход обеспечивает очень высокий уровень безопасности и поэтому рекомендуется, но он крайне неудобен. Для того чтобы не запоминать все пароли и не полагаться на автоподстановку браузера, пользователям рекомендуют устанавливать специальное ПО для хранения паролей, такое как keepassX.

Если же необходимо сохранять состояние профиля со всеми файлами cookies и другими, то подойдет следующее распространенное решение — это шифровать каталог профиля средствами ОС или какими-то сторонними программами. Например, cryptkeeper или EFS в windows. Далее нужно будет запускать браузер с определенным ключом, чтобы изменить расположение каталога пользователя. Например, для chromium нужен такой ключ: --user-data-dir=/path/to/foo. Такой метод защищает от второго типа угрозы, т.е. ограничивает доступ случайных пользователей имеющий доступ к сессии ОС, но к сожалению не от первого, так как расшифрованная папка существует как обычная примонтированная директория, т.е. ВПО так или иначе получит доступ к вашим файлам. Для полной защиты можно запускать браузер внутри docker контейнера или любой виртуальной машине, в котором сама файловая система будет зашифрована. Но этот метод плох тем, что он неудобный и требует много

ресурсов машины на обеспечение работы контейнера. На практике чаще всего используется подход с шифрованием каталога профиля как наиболее удобный.

Если бы браузеры сами прозрачно для себя шифровали все файлы с которыми работали, тогда на ФС хранились бы только зашифрованные данные и не было бы момента времени, когда они существуют в открытом виде. Использование браузера и дополнительно программы для шифрования не так удобно для пользователя, как единый мастер пароль в браузере. Т.е. решение проблемы — это использовать профили браузера, а не операционной системы, прозрачно шифровать данные методами самого браузера, а не ОС. Такой подход, может несколько снизить производительность, а именно скорость работы IO браузера, но зато даст требуемый уровень защищенности.

Благодаря полному шифрованию профиля пользователя браузера можно получить следующие преимущества. Во-первых, полную защищенность данных, начиная от истории браузера, заканчивая паролями и данными web-sql. Во-вторых, можно будет использовать один браузер разным пользователям из-под одной сессии ОС. В-третьих, все cookies будут привязаны к профилю браузера и не надо будет каждый раз при запуске вводить пароли на всех сайтах, т.е. будет сохранённое зашифрованное состояние браузера.

Возникает закономерный вопрос, почему же мейтнеры ни одного браузера не добавили такой функциональности? После некоторого исследования вопроса, я нашел рубрику вопрсов-ответов по поводу безопасности в chromium. В основном там описываются вопросы безопасности со стороны веба. Но есть один интересный пункт: «Why aren't physically-local attacks in Chrome's threat model?» - *почему не рассматриваются физические атаки на Chrome в его модели безопасности* [приложение Б]. Далее приводится приблизительный перевод с пояснениями.

2.2 Почему не рассматриваются физические атаки на Chrome

«Пользователи иногда жалуются, что они могут компрометировать Chromium, если установят вредоносное ПО, заменят dll или используют

уязвимости в api. Мы(мейтнеры) считаем, что все эти уязвимости вне компетенции хромиума, мы не можем защищаться от вредоносного ПО, которое уже есть на вашем компьютере. Так как если кто-то уже получил доступ к вашему компьютеру и вашей сессии, то он сможет так же подменить dll у других приложений и вообще получить всю информацию, хранящуюся у вас на компьютере. Это не проблема хрома — в общем приложения должны доверять локальному пользователю. Чтобы уменьшить риск сторонними пользователями получить физический доступ к вашему компьютеру следуйте этим рекомендациям:

- Используйте FDE – full disk encryption(полное шифрование диска). FDE - это стандартная функция в большинстве ОС, таких как Windows Mac OS X и некоторые дистрибутивы Linux. Используйте пароли и usb-ключи для расшифровывания диска. ChromeOS шифрует домашнюю директорию.
- Если вы разделяете компьютер с несколькими людьми, то пользуйтесь средствами ОС, такими как сессии и шифрование домашнего каталога. В ChromeOS так же есть гостевой режим, после выхода из которого все данные стираются.
- Используйте функцию ОС блокировки экрана.
- Вы можете ограничить количество информации(включая пароли и cookies), которое Chromium сохраняет локально в настройках (chrome:// settings/content). Отключите автоподстановку паролей.

Заметим, что невозможно избежать рисков на общедоступных компьютерах. Всё что хранится на таких компьютерах становится автоматически общедоступным. Используйте режим инкогнито в таких случаях»

Учитывая этот ответ от разработчиков chromium, можно сделать вывод, что функции шифрования профиля пользователя не появится. Тем не менее судя по исходным кодам и новым релизам совсем скоро в chromium добавят поддержку многопользовательского режима. Пока что эту функцию предлагают

использовать для разграничения рабочего и домашнего профиля, но не для использования одного браузера разными людьми. В общем, разработчики не считают открытость данных в профиле какой-то уязвимостью. В их защиту можно сказать, что для личного домашнего использования защита профиля не совсем нужна. Действительно, шифрование домашнего каталога пользователя ОС и установленный антивирус позволяют достичь приемлемого уровня защиты.

При нынешнем подходе к управлению пользователями и их данными в браузерах приходится использовать средства для обеспечения должного уровня безопасности дополнительные средства для шифрования каталога профиля или всегда удалять после работы все данные из браузера, т.е. работать в режиме инкогнито. Но хотелось бы увидеть встроенную поддержку защиты пользовательских данных с их сохранением, например, чтобы продолжать работу без дополнительного восстановления всех сессии, вводя пароли каждый раз. В виде развития этой темы было решено сделать форк одного из браузеров firefox или chrome и сделать модификацию модуля, отвечающего за профили, включив в него слой шифрования данных, перед их сохранением и, соответственно, расшифрованное при доступе к ним. Предполагается, что результат работы будет оформлен в виде динамически подключаемой библиотеки. Далее предложить изменение в официальную сборку браузера и постараться убедить менторов включить это в официальный релиз. В любом случае, необходимо будет добавить в UI браузера дополнительные настройки. При запуске программы пользователю, как и в менеджере логина ОС, предложат выбрать пользователя и ввести мастер ключ или подключить usb-token. Далее используя этот ключ браузер построит ключ для шифрования/ расшифрования всех сохраняемых данных профиля. В процессе изучения исходных кодов браузера chrome оказалось, что среди модулей программы есть несколько участков, отвечающих за обработку профильных данных. Именно в этих местах можно будет добавить прослойку шифрования.

Дальнейшая работа будет связана с браузером Chromium, так как он имеет исчерпывающую документацию почти всех важных модулей. Так же есть удобный крос-референс по проекту, что позволяет очень быстро искать нужный код, в отличие от firefox, где придется строить такие индексы самому. Это не было бы проблемой, если бы проекты не были настолько большими, что собранная версия для отладки занимает около гигабайта, а сборка проекта длится около двух часов.

Заключение

В работе было проведено исследование браузеров Firefox и Chromium на предмет защиты локальных личных даных пользователей, особое внимание было уделено хранению паролей. Каждый браузер предлагает свою систему защиты, Firefox полностью берет на себя задачу по сохранению конфиденциальности паролей, а Chromium использует методы API операционной системы, если работает в Windows, или сторонние менеджеры паролей в Linux и Mac Os X. Тем не менее ни один браузер не поддерживает шифрования всех данных пользователя, поэтому было предложено несколько способов как обеспечить должный уровень защиты данных - от предложения использовать только режим инкогнито на публичных компьютерах, до использования ПО для шифрования только папки профиля браузера. В заключении работы была предложена идея адаптировать браузеры под требуемые условия, изменив исходный код браузера Chromium.

Список литературы

1. Описание физических угроз на браузер Chromium. URL: http://dev.chromium.org/Home/chromium-security/security-faq#TOC-Why-aren-t-physically-local-attacks-in-Chrome-s-threat-model-
2. Статья о хранении паролей в браузерах http://raidersec.blogspot.in/2013/06/how-browsers-store-your-passwords-and.html
3. Документация по функции CryptProtectData. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/aa380261(v=vs.85).aspx
4. Страница проекта Network Security Services. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS
5. Описания хранения паролей профиля в Firefox: https://support.mozilla.org/en-US/kb/profiles-where-firefox-stores-user-data
6. Описание многопользовательского режима в Firefox. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Multiple_Firefox_Profiles
7. Главаня страница проекта Chromium. URL: http://dev.chromium.org/Home
8. Описание Encrypted File System. URL: https://ru.wikipedia.org/wiki/EncFS
9. Официальная документация sqlite. URL: https://www.sqlite.org/docs.html
10. Страница проекта Gnome Keyring. URL: https://wiki.gnome.org/action/show/Projects/GnomeKeyring?action=show&redirect=GnomeKeyring
11. Раздел проекта chromium отвечающий за шифрование паролей. URL: https://code.google.com/p/chromium/codesearch#chromium/src/components/os_crypt/
12. Описание работы DPAPI - windows data protection api. URL: https://msdn.microsoft.com/en-us/library/ms995355.aspx
13. Список программ для востановления паролей. URL: http://securityxploded.com/password-recovery-tools.php

Приложение А

Исходный код: https://gist.github.com/blan4/4dfe9008dddd8bb65d8f

Листинг:

```ruby
require 'sqlite3'
require 'ffi' # extension for programmatically loading dynamic libraries
# Decrypt Chrome Login Data file that contains user passwords

module DPAPI
  extend FFI::Library
  # import crypt32 to use winapi function CryptUnprotectData
  ffi_lib 'crypt32'

  class DecryptError < StandardError; end

=begin
typedef struct _CRYPTOAPI_BLOB {
  DWORD cbData;
  BYTE  *pbData;
} DATA_BLOB;
=end
  class DataBlob < FFI::Struct
    layout :cbData, :uint32,
           :pbData, :pointer
  end

=begin
BOOL WINAPI CryptUnprotectData(
  _In_         DATA_BLOB *pDataIn,
  _Out_opt_    LPWSTR *ppszDataDescr,
  _In_opt_     DATA_BLOB *pOptionalEntropy,
  _Reserved_   PVOID pvReserved,
  _In_opt_     CRYPTPROTECT_PROMPTSTRUCT *pPromptStruct,
  _In_         DWORD dwFlags,
  _Out_        DATA_BLOB *pDataOut
);
=end
  attach_function :CryptUnprotectData,
    [:pointer, :pointer, :pointer, :pointer, :pointer, :uint32, :pointer],
    :int32

  def self.decrypt(ciphertext)
    plaintext_blob  = DataBlob.new
    ciphertext_blob = DataBlob.new
    ciphertext_blob[:cbData] = ciphertext.bytesize
    ciphertext_blob[:pbData] = FFI::MemoryPointer.from_string(ciphertext)

    #Contains a password or other additional entropy
    #used when the data was encrypted. This parameter can be set to NULL
    entropy = DataBlob.new
    pwd = ''
    entropy[:cbData] = pwd.bytesize
    entropy[:pbData] = FFI::MemoryPointer.from_string(pwd)

    raise DecryptError if CryptUnprotectData(
      ciphertext_blob, nil, entropy, nil, nil, 0, plaintext_blob).zero?

    plaintext_blob[:pbData].get_bytes(0, plaintext_blob[:cbData])
  end

end

db = SQLite3::Database.new 'Login Data' # Load sqlite file
rows = db.execute('SELECT username_value, password_value FROM logins')
pwd=rows.map do |enc_pwd|
  {enc_pwd.first => DPAPI::decrypt(enc_pwd.last)}
end

puts pwd # Show all decrypted login/password pairs
```

Рисунок 2: листинг программы для получения пар логин-пароля из chromium

Приложение Б

Текст оригинала из статьи в документации chromium:

«People sometimes report that they can compromise Chrome by installing a malicious DLL on a computer in a place where Chrome will find it and load it, or by hooking APIs (see https://code.google.com/p/chromium/issues/detail?id=130284 for an example).

We consider these attacks outside Chrome's threat model, because there is no way for Chrome (or any application) to defend against a malicious user who has managed to log into your computer as you, or who can run software with the privileges of your operating system user account. Such an attacker can modify executables and DLLs, change environment variables like PATH, change configuration files, read any data your user account owns, email it to themselves, and so on. Such an attacker has total control over your computer, and nothing Chrome can do would provide a serious guarantee of defense. This problem is not special to Chrome — all applications must trust the physically-local user.

There are a few things you can do to mitigate risks from people who have physical control over your computer, in certain circumstances.

To stop people from reading your data in cases of device theft or loss, use full disk encryption (FDE). FDE is a standard feature of most operating systems, including Windows Vista and later, Mac OS X Lion and later, and some distributions of Linux. (Some older versions of Mac OS X had partial disk encryption: they could encrypt the user’s home folder, which contains the bulk of a user’s sensitive data.) Some FDE systems allow you to use multiple sources of key material, such as the combination of both a password and a key file on a USB token. When available, you should use multiple sources of key material to achieve the strongest defense. Chrome OS encrypts users’ home directories.

If you share your computer with other people, take advantage of your operating system’s ability to manage multiple login accounts, and use a distinct account for each person. For guests, Chrome OS has a built-in Guest account for this purpose.

Take advantage of your operating system’s screen lock feature.

You can reduce the amount of information (including credentials like cookies and passwords) that Chrome will store locally by using Chrome's Content Settings (chrome://settings/content) and turning off the form auto-fill and password storage features (chrome://settings/search#password).

There is almost nothing you can do to mitigate risks when using a public computer.

Assume everything you do on a public computer will become, well, public. You have no control over the operating system or other software on the machine, and there is no reason to trust the integrity of it.

If you must use such a computer, consider using an incognito mode window, to avoid persisting credentials. This, however, provides no protection when the system is already compromised as above.»