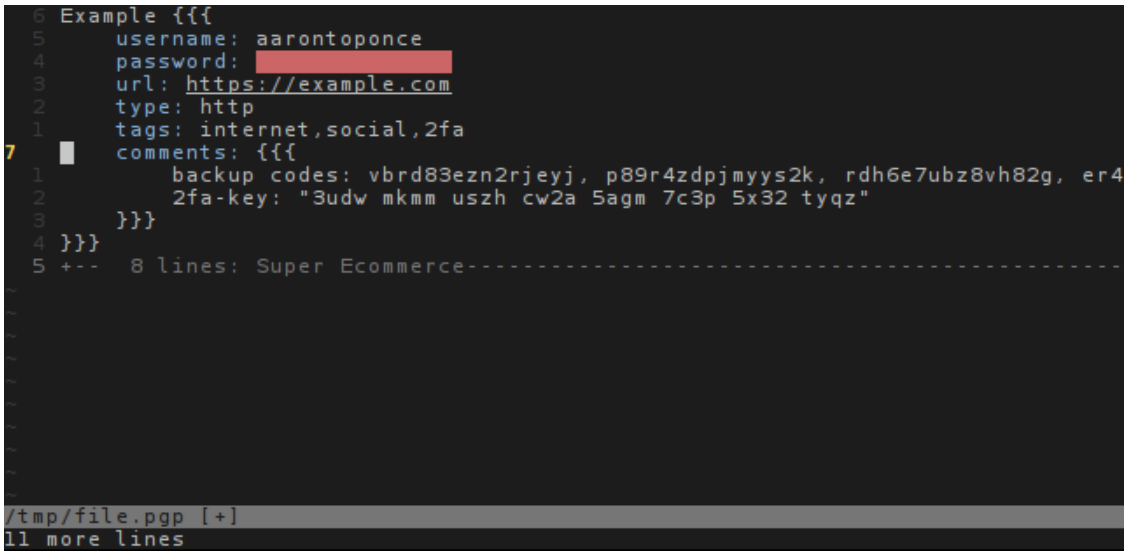
Aaron Toponce

{ 2015 12 31 }

Encrypted Account Passwords with Vim and GnuPG

Background



I've been a long-time KeepassX user, and to be honest, I don't see that changing any time soon. I currently have my password database on an SSH-accessible server, of which I use kpcli as the main client for accessing the db. I use Keepass2Android with SFTP on my phone to get read-only access to the db, and I use sshfs mounts on my workstations with KeepassX for read-only GUI access. It works great, and allows me to securely access my password databases from any client, mobile or otherwise.

However, I recently stumbled on this post on how to use Vim with GnuPG to create an encrypted file of passwords: http://pig-monkey.com/2013/04/password-management-vim-gnupg/. I've heard about a GnuPG plugin for Vim for years now, and know friends that use it. I've even recommended that others use

it as a simplistic means of keeping an encrypted password database, instead of relying on 3rd-party tools. However, I've never really used it myself. Well, after reading that post, I decided to give it a try.

Defining a specification

Ultimately, everything in that post I'm carrying over here, with only a couple modifications. First, fields should end with a colon, which include the comments. Comments could just be just a single line, or multi-line, but it's still a field just as much as "user" or "pass". Further, there should be a little flexibility in the field keywords, such as "user" or "username". Additionally, because I exported my Keepass db to an XML file, then used a Python script to convert it into this syntax, I also carried over some additional fields. So, I've defined my database with the following possible fields:

- comment|comments
- expire|expires
- pass|password
- tag|tags
- type
- url
- user|username

Notice that I did not define a "title" as would be the case in the Keepass XML. The entry itself is the title, so I find this redundant. Also, you'll noticed I defined an additional "type" field. While not explicitly defined in the Keepass XML, it is implicitly defined with icons for entries. This could be useful for defining "ssh" vs "mysql" vs "ldap" vs "http" authentications when doing searching in the file.

So, an invalid example on pig-monkey.com is:

```
Super Ecommerce{{{
    user:   foobar
    pass:   g0d
    Comments{{{
        birthday:   1/1/1911
        first car:  delorean
    }}}
}}}
```

This is invalid due to the "Comments" field. Fixed would be:

```
Super Ecommerce{{{
    user:   foobar
    pass:   g0d
    Comments:{{{
        birthday:   1/1/1911
        first car:  delorean
    }}}
}}}
```

Another valid entry could be:

```
Example {{{
    username: aarontoponce
    password: toomanysecrets
    url: https://example.com
    type: http
    tags: internet,social,2fa
    comments: {{{
        backup codes: vbrd83ezn2rjeyj, p89r4zdpjmyys2k, rdh6e7ubz8vh82g, er4ug6vp25xsgn5
        2fa-key: "3udw mkmm uszh cw2a 5agm 7c3p 5x32 tyqz"
```

```
    }}}
}}}
```

Notice that I have not defined file comments, such as those found in configuration files or source code. There is a comment section per entry, so that seems to be the fitting place for any and all comments.

I really liked the post, and how thought out the whole thing was, including automatically closing the PGP file after an inactivity timeout, automatically folding entries to prevent shoulder surfing, and clearing the clipboard when Vim closes. However, one oversight that bothered me, was not concealing the actual password when the entry is expanded. Thankfully, Vim supports syntax highlighting. So, we just need to define a filetype for GnuPG encrypted accounts, and define syntax rules.

Vim syntax highlighting

EDITED TO ADD: I tried getting the Vim syntax working in this post, but WordPress is clobbering it. So, you'll need to get it from pastebin instead. Sorry.

To get this working, we need a syntax file. I don't know if one exists already for this syntax structure, but it isn't too difficult to define one. Let's look at what I've defined in this pastebin, then I'll go over it line-by-line.

The first four lines in the syntax file define are just comments. Next is just a simple if-statement checking if syntax highlighting is enabled. If so, use it. The first interesting line is the following:

```
let b:current_syntax = "gpgpass"
```

This defines our syntax. Whenever we load a file with syntax highlighting enabled, and we set the "filetype" to "gpgpass", this syntax will be applied.

```
syntax case ignore
```

This just allows us to have "comment" or "Comment" or "COMMENT" or any variations on the letter case, while still matching and proving a highlight for the match.

After that, we get into the meat of it. This "syntax match" section allows me to conceal the passwords on the terminal to prevent shoulder surfing, even when the entry is expanded. this is done with setting the background terminal color to "red" and the foreground text color also to "red". Thus, we have red text on a red background. The text is still yankable and copyable, even with the mouse cursor, it's just not visible on screen.

The actual concealment is done with the regular expression. An atom is created to match "pass:" or "password:" surrounded by whitespace as the first word on the line. However, I don't want to conceal the actual text "pass:", just the password itself. So, the regular expression "\@<=" says to ignore our atom in the match, and only match "\S\+" for concealing. The concealment is achieved with red foreground text on a red background with:

```
highlight gpgpassPasswords ctermbg=red ctermfg=red
```

The rest of the syntax matching in that pastebin is for identifying our fields, and highlighting them as a "Keyword" using regular expressions. All field names will be highlighted the same color based on your colorscheme, as they are all defined the same. Thus, aside from the hidden password, there is uniformity and elegance in the presentation of the syntax.

Using the syntax in Vim

This syntax file won't do us much good if it isn't installed and Vim isn't configured to use it. We could save it system-wide to "/usr/share/vim/vim74/syntax/gpgpass.vim", or just keep it in our home directory

at "~/.vim/syntax/gpgpass.vim". Whatever works.

Now that the syntax file is installed, we need to call it when editing or viewing GnuPG password files. We can use the vimrc from pig-monkey.com, with one addition- we're going to add "set filetype=gpgpass" under the "SetGPGOptions()" function. Now, I understand that you may edit encrypted files that are not GnuPG password files. So, you're going to get syntax highlighting in those cases. Or, you could enable the modeline and set a modeline in the password file. The problem with the modeline, is its long history of vulnerabilities. Most distributions, including Debain, disable it, and for good reason too. So, I'd rather have it set here, and unset the "filetype" if it's bothering me.

Here's the relevant config:

```
if has("autocmd")
    """""""""""""""""""""
    " GnuPG Extensions "
    """""""""""""""""""""

    " Tell the GnuPG plugin to armor new files.
    let g:GPGPreferArmor=1

    " Tell the GnuPG plugin to sign new files.
    let g:GPGPreferSign=1

    augroup GnuPGExtra
    " Set extra file options.
        autocmd BufReadCmd,FileReadCmd *.\(gpg\|asc\|pgp\) call SetGPGOptions()
    " Automatically close unmodified files after inactivity.
        autocmd CursorHold *.\(gpg\|asc\|pgp\) quit
    augroup END
```

```
    function SetGPGOptions()
    " Set the filetype for syntax highlighting.
        set filetype=gpgpass
    " Set updatetime to 1 minute.
        set updatetime=60000
    " Fold at markers.
        set foldmethod=marker
    " Automatically close all folds.
        set foldclose=all
    " Only open folds with insert commands.
        set foldopen=insert
    endfunction
endif " has ("autocmd")
```

Conclusion

What I like about this setup is the portability and simplicity. I am in a terminal on a GNU/Linux box most of my waking hours. It makes sense to use tools that I already enjoy, without needing to rely on 3rd party tools. This also closes the gap of potential bugs with 3rd party password managers leaking my passwords. I'm not saying that Vim and GnuPG won't be vulnerable, of course, but I do place more trust in these tools than the Keepass ones, to be honest.

As of right now, however, I am still a Keepass user. But, I wanted to put this together, and try it out for size, and see how the shoe fits. As such, I've exported my KeepassX database, encrypted it with GnuPG, configured Vim, and I'm off to the races. I'll give this a go for a few months, and see how I like it. I know it's going to pose issues for mp on my phone, even with ConnectBot and SSH keys. But, maybe I don't need it on my phone anyway. Time will tell.

Oh, and I can still view the database as read-only and still enjoy the syntax highlighting benefits by using "view /path/to/passwords.gpg" instead of "vim /path/to/passwords.gpg".

Posted by Aaron Toponce on Thursday, December 31, 2015, at 5:40 am. Filed under Cryptology, Debian, Linux, Scripting, Security. Follow any responses to this post with its comments RSS feed. You can post a comment or trackback from your blog. For IM, Email or Microblogs, here is the Shortlink.

{ 3 } Comments

1. Johannes Larsen | December 31, 2015 at 9:33 am | Permalink

   pass, [1], is an open-source password managment system that uses GnuPG for encrypting the passwords. It offers more or less the same functionality, but organizes passwords as a directory structure filled with GnuPG encrypted files instead one large GnuPG encrypted file with lots of entries. Additionally it has browser plugin support to make it a bit easier to use.

   Occasionally I copy passwords to my phone via VX ConnectBot, but it is very cumbersome to mark the correct string in the SSH-session, so I often end up typing in a password shown on anther computer.

   [1]: http://www.passwordstore.org/

2. Aaron Toponce | December 31, 2015 at 1:43 pm | Permalink

Yeah. I'm familiar with it. I prefer my passwords easily searchable in a single utility in one file, rather than scattered about nested directories on the filesystem. To each their own.

However, I do know what you mean regarding copying passwords out of the phone. Marking the correct string is full of UI fail, and it really sucks. The only way I have around that is selecting more than the password, pasting in a buffer, and getting just the password string I need. Still, PITA. This is why I've stuck with Keepass for so long- Android integration Just Works (tm). But, this is an interesting test to see what I think about it.

3. Josh Kirkwood | January 20, 2016 at 4:58 pm | Permalink

   What font are you using in this screenshot Aaron?
   My eyes love it.