QGIS Application - Bug report #13568
Master password is suspectible to brute-force attacks

2015-10-11 01:47 PM - Nyall Dawson

Status:	Closed		
Priority:	Severe/Regression		
Assignee:	Larry Shaffer		
Category:	Authentication system		
Affected QGIS version:	master	**Regression?:**	No
Operating System:		**Easy fix?:**	No
Pull Request or Patch supplied:	No	**Resolution:**	
Crashes QGIS or corrupts data:	No	**Copied to github as #:**	21610

Description

Currently there's no limit on the number of times a user can attempt to enter a master password. This could lead to brute-force attacks where thousands of generated passwords are sent to the dialog.

The dialog should have a timer - so enter 3 wrong passwords = dialog locked for 10 seconds, enter 4 wrong passwords = locked for 20 seconds, etc to prevent this type of attack.

Associated revisions

Revision dfb476a5 - 2015-10-23 09:25 AM - Larry Shaffer

[auth] Fix #13568; disable auth system after 5 wrong password attempts

History

#1 - 2015-10-13 03:11 PM - Jürgen Fischer

- Category set to Authentication system

#2 - 2015-10-15 01:45 PM - Larry Shaffer

- Target version set to Version 2.12

- Status changed from Open to Feedback

Hi Nyall,

How about something simpler: if the user types their master password incorrectly 5 times, the auth system is set to **disabled** (similar to the state it goes into when some of its components are missing, like the QCA OpenSSL plugin)? Then, a relaunch of QGIS would be necessary. I figure if you can't input your password after 5 tries, then you probably just don't know it.

Regarding brute force attack scripts, this assumes someone other than the user has access to install and run scripts in the user account, which is a pretty large failure of basic user account security and the user (whose account it is) is really quite screwed. If an attacker actually has access enough to install a user-executable script, then there is nothing stopping them from doing any of the following:

- Use a pre-compiled C++ binary linked to the qgis* libs (possibly statically) to do the attack at the API level.
- Use a pre-compiled C++ plugin to do the attack at the API level.
- Use a Python plugin to do the attack at the bindings API level (though these could be blocked).
- Use almost any type of programming (and knowledge of the auth C++ classes source) to attack the SQLite auth database directly.
- Copy the SQLite auth database from the user's computer and attack it relentlessly on a different computer.

In all API access scenarios, disabling or blocking makes sense to me, because the attacking program can always just generate a new QgsApplication and start over, regardless of any timeouts.

The takeaway here is:

- A timeout or disabled state after a number of tries really only protects against casual access to the user's QGIS GUI
- The real concern is how to block automated access, via the API, from C++ and Python plugins, and how to ensure nefarious plugins don't just grab the user's auth db and upload it elsewhere.

Any other ideas or concerns would be greatly appreciated.

#3 - 2015-10-15 02:12 PM - Nyall Dawson

So to clarify, you're proposing:

- Disabling the auth system if more 5 wrong passwords in the GUI
- Also disabling the auth system if 5 invalid attempts are made through the API?

If so, sounds reasonable to me. It's obviously not fool proof, but we should do as much as possible to lock this down...

#4 - 2015-10-15 03:36 PM - Larry Shaffer

Nyall Dawson wrote:

> *So to clarify, you're proposing:*
>
> *- Disabling the auth system if more 5 wrong passwords in the GUI*
> *- Also disabling the auth system if 5 invalid attempts are made through the API?*

Yes. The disabling will happen in the QgsAuthManager instance, upon password-setting attempts, so both the API and GUI essentially get disabled, since the isDisabled() check is done on all password functions first. Again, the attacker could always just generate a new QgsApplication, if scripting, or relaunch QGIS to try again via GUI or plugin.

If we try to make the timeout or disabling sticky across app restarts or QgsApplication instantiation, we run the risk of annoying users; and, any techniques to produce such permanence would be discoverable via the source code anyhow.

> *If so, sounds reasonable to me. It's obviously not fool proof, but we should do as much as possible to lock this down...*

Best solution: good master password. :^)

Second best: lock down EVERYTHING for the first release and label the new auth system as *experimental* until the public has had a chance to play with or break it during a release cycle. This means getting rid of almost all Python bindings, which would preclude any PyQGIS plugins from accessing credentials to connect to network resources, i.e. grab the credentials and set up a httplib2 or requests or even QgsNetworkAccessManager connection to a network resource.

Plugin authors could still use the auth config selector widget and add the selected authcfg ID to a layer being added to the app. They just shouldn't be allowed to use the auth system for their *own* connections until we figure out the best way to do that, e.g. trust PKI certificates or API access tokens issued by the project, etc.

#5 - 2015-10-23 12:25 AM - Larry Shaffer

- Status changed from Feedback to Closed

Fixed in changeset commit:"dfb476a527128b4958c9c807d8d38632cb54287d".