Name

passwdmanapi — library used by passwdmancli and passwdmangui

Synopsis

```
#!/usr/bin/python

import passwdmanapi

open_rng()

get64(length)

get10(length)

getint(a, b)

unquote(x)

undo(passwdobj=None, honeypotobj=None)

redo(passwdobj=None, honeypotobj=None)

class common_data()

class passwd(common_data)

class honeypot(common_data)
```

class common_data():

__init__(self, xmlfile)
__iter__(self)
__next__(self)
next(self)
__getitem__(self, i)
__len__(self)
remove(self, x, xmlfile, element_name, attrib_name, is_numstring=False)
writexml(self, xmlfile)
__del__(self)

passwd(common_data):

__init__(self)
add(self, name, value, m_type, m_minlength, m_maxlength)
add_nometa(self, name, value)
remove(self, x, is_numstring=False)
__repr__(self)
mkindex(self, x, is_numstring=False)
update(self, index)
update_meta(self, index, m_type, m_minlength, m_maxlength)

class honeypot(common_data):

__init__(self)

add(self, value)
remove(self, x, is_numstring=False)
pick(self, n=1, sep=",", log_vs_raise=True)
__repr__(self)

Exceptions

class err_norandom(Exception)
class err_nolength(Exception)
class err_loaderr(Exception)
class err_notfound(Exception)
class err_duplicate(Exception)
class err_idiot(Exception)
class err_nometa(Exception)

DESCRIPTION

Unless otherwise noted, `xmlfile` is a path.

open_rng() opens random(4) (or urandom(4), if random could not be opened). Returns a file open for reading binary. Raises err_norandom.

get10() and get64() returns a random string of `length` letters. get10() returns digits. get64() returns digits, big letters, small letters, underscores and exclamation marks. Raises err_norandom and err_nolength.

getint() returns a random integer >= `a`, <= `b`. Raises err_norandom and err_nolength.

unquote() returns the string `x` without its surrounding quotes. If the string is not surrounded be quotes, the string will be returned unchanged.

undo() undoes the latest change to the password list or honey pot list, by restoring from the newest auto-generated backup. It requires `passwdobj` which is the passwd() object and `honeypotobj` which is the honeypot() object. Raises err_idiot.

redo() redoes the latest undone change to the password list or honey pot list, by restoring from the newest auto-generated backup from undo(). Raises err_idiot.

common_data() is a class defining methods used by both passwd() and honeypot().

passwd() is a class for the password list. honeypot() is a class for the honey-pot list. See FILES.

class common_data():

__init__() will load the data from `xmlfile`. Raises err_loaderr.

__iter__() resets the index and returns `self`. __getitem__() returns the password/honeypot at `i`. __len__() returns the number of passwords/honeypots.

remove() removes the password/honeypot at `x`, which can be an integer or a stringed integer or the value of the password/honeypot, from the datastructure `self` and the file `xmlfile`. `element_name` and `attrib_name` tells it what elements in the XML file and attributes it should loop through, remove and find a match for `x` in. Set `is_numstring` to True if `x` is a string containing digits. If you don't set it, then `x` will be treated as an index. Raises err_notfound.

writexml() writes the datastructure `self` to the file `xmlfile`. It creates a backup of `xmlfile` to `~/.passwdman/undoable`.

class passwd(common_data)

passwd() loads its data from the XML `~/.passwdman/passwords`.

`self[index]["name"]` is the name/purpose of the password. `self[index]["value"]` is the value of the password. `self[index]["meta"]["minlength"]` is the minimal length required for the password. `self[index]["meta"]["maxlength"]` is the maximal length allowed for the password. `self[index]["meta"]["type"]` is the type of the password, which is one of:

10 The password may only use digits.

64 The password can use big letters, small letters, digits, underscores and exclamation marks.

human The password is human generated.

If a password has no meta-data in `~/.passwdman/passwords`, its `minlength` and `maxlength` will be zero, and its `type` will be "human".

passwd.add() and passwd.add_nometa() adds a password for `name` with the value `value`. add_nometa() adds a password without real meta-data while add() requires meta-data (the `m_type` must be a string and `m_minlength` and `m_maxlength` can be either an integer or a stringed integer). Raises err_duplicate.

passwd.remove() removes the password `x`. `x` can be either a string matching a password's name or an integer (index) or a stringed integer. Set `is_numstring` to True if `x` is a string containing digits. If you don't set it, then `x` will be treated as an index. Raises err_notfound.

passwd.mkindex() find `x` and return an index. `x` can be either a string matching a password's name or a stringed integer (index). Set `is_numstring` to True if `x` is a string containing digits. If you don't set it, then `x` will be treated as an index. Raises err_notfound.

passwd.update() and passwd.update_meta() updates the password at `index` automatically by generating a password of the right type and an acceptable length. update() uses the password's own meta-data while update_meta() gives the password new meta-data from `m_type`, `m_minlength` and `m_maxlength`. `m_type` must be a string, `m_minlength` and `m_maxlength` can be either an integer or a stringed integer. Raises err_notfound, err_idiot and err_nometa.

class honeypot(common_data)

The honey pots are weak passwords supposed to only be used as traps. honeypot() loads its data from the XML `~/.passwdman/honeypots`. `self[index]` is the value of the honeypot.

honeypot.add() adds a new honeypot with the value `value`. Raises err_duplicate.

honeypot.remove() removes the honeypot `x`. `x` is either an index (integer) or a stringed integer or the value of the honeypot. Set `is_numstring` to True if `x` is a string containing digits. If you don't set it, then `x` will be treated as an index. Raises err_notfound.

honeypot.pick() picks `n` random honeypots and returns a string of honeypots separated with `sep`. If `log_vs_raise` is true, it will log an error if `n` is too big. If `log_vs_raise` is false, it will raise err_idiot.

Exceptions

err_norandom is raised when neither random(4) or urandom(4) can be opened.

- open_rng()
- get10()
- get64()
- getint()

- passwd.update()
- passwd.update_meta()
- honeypot.pick()

err_nolength is raised when a function is called with an invalid length.

- get64()
- get10()
- getint()

err_loaderr is raised if data cannot be loaded from file.

- common_data()
- passwd()
- honeypot()

err_notfound is raised if index is out of range or if it cannot find a match.

- common_data.remove()
- passwd.remove()
- passwd.mkindex()
- passwd.update()
- passwd.update_meta()
- honeypot.remove()

err_duplicate is raised if it is attempted to add a password with the same name as another or if its is attempted to add a honeypot with the same value as another.

- passwd.add()
- passwd.add_nometa()
- honeypot.add()

err_idiot is raised if the function was not used correctly.

- passwd.update_meta()
- honeypot.pick()
- undo()
- redo()

err_nometa is raised when meta-data is required, but the meta-data was nonexistent, corrupt or no good.

- passwd.update()

FILES

`~/.passwdman/passwords` is the XML file containing the passwords and their meta-data.

`~/.passwdman/honeypots` is the XML file containing the honeypots.

`~/.passwdman/undoable/` is where the auto-generated backups live.

`~/.passwdman/redoable/` is where the backups generated by undo() live.

AUTHOR

Written by Oskar Skog (oskar.skog.finland@gmail.com).

Please send patches, questions, bug reports and wish-lists.