Package 'rcreds'

October 27, 2017

Title Securely Use Credentials Within R Scripts

Version 0.6.6

Description Tools to write a list of credentials to an encrypted file and later read from that file into R. The goal is to have a useful alternative to including username/passwords as part of a script or even stored in the clear in a separate text file. Additional tools provided which are specific for connecting to a database.

Depends R (>= 3.3.1)

License MIT + file LICENSE

Suggests testthat, knitr, rmarkdown

Encoding UTF-8

LazyData TRUE

RoxygenNote 6.0.1.9000

Imports jsonlite, digest, magrittr, stats, utils, collectArgs

VignetteBuilder knitr

NeedsCompilation no

Author Rick Saporta [aut, cre],
Mike Reca [ctb]

Maintainer Rick Saporta <RickSaporta@gmail.com>

Repository CRAN

Date/Publication 2017-10-27 16:35:21 UTC

R topics documented:

. *R CMD Check compalin blocker*

Description

R CMD Check compalin blocker

Usage

```
.
```

Format

An object of class `character` of length 1.

`.onLoad` *Standard onLoad function for rcreds*

Description

Sets several options, if they are not already set when this function funs.

Usage

```
.onLoad(libname, pkgname)
```

Arguments

`libname`	libname and pkgname are default `R .onLoad()` parameters
`pkgname`	libname and pkgname are default `R .onLoad()` parameters

Value

invisible(TRUE)

(not exported)

construct_rcreds_file_full_path *Construct Rcreds File Full Path*

Description

Takes parts of a file name, info, and folder and creates a full path

Usage

```
construct_rcreds_file_full_path(file_name = getOption("rcreds.file_name",
  default = ".credentials.creds"), folder = get_default_rcreds_folder(DB =
  DB), info.file_name = "", DB = FALSE)
```

Arguments

`file_name`	string, cannot be empty Filename with extension. Default is '.credentials.creds'
`folder`	string, cannot be empty Where file will be stored
`info.file_name`	string, can be empty. Prepended to file_name.
`DB`	TRUE/FALSE flag. Is the folder for the db credentials functions? Defaults to: FALSE

Value

The full file path

credentials_functions *Credentials Functions*

Description

Securely Write/Read Sensitive Parameters to/from Disk

Usage

```
write_credentials_to_file(..., file_full_path = "..auto..",
  info.file_name = "", file_name = getOption("rcreds.file_name", default =
  ".credentials.creds"), folder = get_default_rcreds_folder(DB = FALSE),
  allow_root_user = FALSE, zArchive_existing = TRUE,
  overwrite_existing = FALSE, key = read_key_from_file(),
  showWarnings = TRUE, verbose = getOption("verbose.rcreds", default =
  TRUE))

read_credentials_from_file(file_full_path = "..auto..", info.file_name = "",
  file_name = getOption("rcreds.file_name", default = ".credentials.creds"),
  folder = get_default_rcreds_folder(DB = FALSE),
  key = read_key_from_file(), fail_if_cant_decrypt = TRUE,
  showWarnings = TRUE, verbose = getOption("verbose.rcreds", default =
  TRUE))

read_db_credentials_from_file(file_full_path = "..auto..",
  info.file_name = "", file_name = getOption("rcreds.db.file_name", default
  = ".db_credentials.creds"), folder = get_default_rcreds_folder(DB = TRUE),
  key = read_key_from_file(), fail_if_cant_decrypt = TRUE,
  showWarnings = TRUE, verbose = getOption("verbose.rcreds", default =
  TRUE))

write_db_credentials_to_file(dbname = "dev", host = "localhost",
  port = 5432, username = "you_forgot_to_specify_username",
  password = "too_many_secrets", file_full_path = "..auto..",
  info.file_name = "", file_name = getOption("rcreds.db.file_name", default
  = ".db_credentials.creds"), folder = get_default_rcreds_folder(DB = TRUE),
  allow_root_user = FALSE, zArchive_existing = TRUE,
  overwrite_existing = FALSE, key = read_key_from_file(), ...,
  verbose = getOption("verbose.rcreds", default = TRUE))
```

Arguments

`...`	values to be encrypted and written to the credentials file. if named parameters, the list which is outputed by `read_credentials_from_file` will use those same names. if empty in `write_credentials_to_file` then nothing will be written to disk in the credentials file
`file_full_path`	The full path to the creds (or key) file, where it should be read from or written to. if `"..auto.."` then will be constructed from `folder`, `file_name`, and `info.file_name` NOTE: When `file_full_path` is set explicitly, then `folder`, `file_name`, and `info.file_name` are ignored. Defaults to: "..auto.."
`info.file_name`	Will be added as a prefix to the filename. Useful when using multiple files in a given folder. Defaults to: "\"\""
`file_name`	name of the file where the credentials will be written to or read from. Should be a string of length 1 Defaults to: getOption(\"rcreds.file_name\", default = \".credentials.creds\")
`folder`	folder where the credentials will be written to or read from. Defaults to: get_default_rcreds_folder(DB=FALSE)
`allow_root_user`	A TRUE/FALSE flag. If FALSE and user is root, then writing and saving functions will fail This is a safety to make sure the user understands they are operating under root. Defaults to: FALSE

`zArchive_existing`	A TRUE/FALSE flag. If `file_full_path` already exist, should it be moved to a zArchive folder? Defaults to: TRUE
`overwrite_existing`	A TRUE/FALSE flag. If `file_full_path` already exist, should it be overwritten? This is only considered when `zArchive_existing` is FALSE Defaults to: FALSE
`key`	A key object of class `"key_rcreds"` to be used for encrypting / decrypting. Passed to `digest::AES`. Alternatively, a full file path to a key stored on disk can be given which will be read to disk. Defaults to: read_key_from_file()
`showWarnings`	A TRUE/FALSE flag. If FALSE, warnings will be silenced Defaults to: TRUE
`verbose`	A TRUE/FALSE flag. Defaults to: getOption(\"verbose.rcreds\", default = TRUE) Details The `write_..` functions take a list of parameters along with a key object, encrypt the parameters and write them to a file on disk. The `read_..` functions read said file, and given (the same) key object, decrypt the parameters and return a named list. The corresponding `.._db_..` files are wrappers that explicitly list the main five parameters used for database connections, with comonly used defaults. Namely, `host`, `username`, `password`, `port`, and `database`.
`fail_if_cant_decrypt`	A TRUE/FALSE flag. If set to TRUE, the reading functions will fail on error. If set to FALSE, NULL will be returned and a graceful exit will happen (with a possible warning if `showWarnings` is TRUE. Defaults to: TRUE
`dbname`	parameter for database connections. Will be encrypted and written to database Defaults to: "dev"
`host`	parameter for database connections. Will be encrypted and written to database Defaults to: "localhost"
`port`	parameter for database connections. Will be encrypted and written to database Defaults to: 5432
`username`	parameter for database connections. Will be encrypted and written to database Defaults to: "you_forgot_to_specify_username"
`password`	parameter for database connections. Will be encrypted and written to database Defaults to: "too_many_secrets"

Details

#' There are two sets of pairs of functions use `write_credentials_to_file()` to output to disk use `read_credentials_from_file()` to read in the credentials back to R

Similarly, there are a pair of functions with the 5 comonly-used parameters for database connections use `write_db_credentials_to_file()` and `read_db_credentials_from_file()`

Value

for `write_credentials_to_file` and `write_db_credentials_to_file` The file path where the encrypted values have been stored, reutrned invisibly. ie the value of `file_full_path`

for `read_credentials_from_file` and `read_db_credentials_from_file` a named list of the values stored in the credentials file. The names of the list correspond to the names of the argument passed to the corresponding write functions

Examples

```
 ## Not run:
   library(rcreds)

   some_login_function <- function(username, password) {
     ## does something with username/password
     ## ...
   }

   ### ---------------------------------------------- ###
   ## Default Folders need to be set. This shold be in an .Rprofile file
   ### ---------------------------------------------- ###
   ## generally use:  set_default_rcreds_ALL(parent_folder = "~/.rcreds/")
   set_default_rcreds_ALL(parent_folder = file.path(tempdir(), ".rcreds/"),
                          create_if_not_exist = TRUE)
   ### ---------------------------------------------- ###

   ## ONE TIME, DO NOT SAVE THIS
   write_db_credentials_to_file(username="cosmo", password="still too many secrets"
                      , port=1234, host="ec2-1234-567-89.us-west.compute.amazonaws.com")


   ## SEPARATELY, in a new file:
  credentials_list <- read_db_credentials_from_file(fail_if_cant_decrypt=FALSE, showWarnings=FALSE)
  ## normally, leave the above flags as their default TRUE. Using FALSE for this example only.

   some_login_function(username = credentials_list$user_name
                     , password = credentials_list$password
                      )

## End(Not run)
```

`key_functions` *Key Functions*

Description

ONE-LINER WHAT DO THESE GROUP OF FUNCS DO? (or the name of the main function)

Usage

```
show_default_rcreds_key_file()

create_key(bytes = 32, depth = 8, seed = NULL, showWarnings = TRUE,
  verbose = getOption("verbose.rcreds", default = TRUE))

is.key_rcreds(key)

save_key(file_full_path = file.path(folder, file_name),
  file_name = getOption("rcreds.key.file_name", default = ".crypt_key.rds"),
  folder = get_default_rcreds_key_folder(), key, bytes = 32, depth = 8,
  seed = NULL, zArchive_existing = TRUE, overwrite_existing = FALSE,
  showWarnings = TRUE, allow_root_user = FALSE,
  verbose = getOption("verbose.rcreds", default = TRUE))

read_key_from_file(file_full_path = file.path(folder, file_name),
  file_name = getOption("rcreds.key.file_name", default = ".crypt_key.rds"),
  folder = get_default_rcreds_key_folder(), create_if_not_exist = TRUE,
  showWarnings = FALSE, verbose = getOption("verbose.rcreds", default =
  TRUE))
```

Arguments

`bytes`	Number of bytes used for the key. Values should normally be one of `c(16, 24, 32)` Defaults to: 32
`depth`	Bit depth for key. Defaults to: 8
`seed`	An integer passed to `set.seed()` Generating the key involves random number generation. Setting the seed will make the key determenistic. Defaults to: NULL
`showWarnings`	A TRUE/FALSE flag. If FALSE, warnings will be silenced Defaults to: TRUE
`verbose`	A TRUE/FALSE flag. Defaults to: getOption(\"verbose.rcreds\", default = TRUE)
`key`	A key object of class `"key_rcreds"` to be used for encrypting / decrypting. Passed to `digest::AES`. Alternatively, a full file path to a key stored on disk can be given which will be read to disk.

`file_full_path`	The full path to the creds (or key) file, where it should be read from or written to. if `"..auto.."` then will be constructed from `folder`, `file_name`, and `info.file_name`
	NOTE: When `file_full_path` is set explicitly, then `folder`, `file_name`, and `info.file_name` are ignored.
	Defaults to: file.path(folder, file_name)
`file_name`	name of the file where the key will be written to or read from. Should be a string of length 1
	Defaults to: getOption(\"rcreds.key.file_name\", default = \".crypt_key.rds\")
`folder`	folder where the credentials will be written to or read from.
	Defaults to: get_default_rcreds_key_folder()
`zArchive_existing`	A TRUE/FALSE flag. If `file_full_path` already exist, should it be moved to a zArchive folder?
	Defaults to: TRUE
`overwrite_existing`	A TRUE/FALSE flag. If `file_full_path` already exist, should it be overwritten? This is only considered when `zArchive_existing` is FALSE
	Defaults to: FALSE
`allow_root_user`	A TRUE/FALSE flag. If FALSE and user is root, then writing and saving functions will fail This is a safety to make sure the user understands they are operating under root.
	Defaults to: FALSE
`create_if_not_exist`	A TRUE/FALSE flag. for `read_key_from_file`: If the given file does not exist, should a key be created and stored at that location?
	Defaults to: TRUE

Details

DETAILED DESCRIPTION of what these functions do

Value

for `create_key` and `read_key_from_file`: An object of class key_rcreds

for `save_key` and `show_default_rcreds_key_file`: A full file path. In the case of `save_key`, this is where the key has been written to.

for `is.key_rcreds`: A TRUE/FALSE value indidcating if the input is of class `"key_rcreds"`

Examples

```
 ## Not run:
   ### ---------------------------------------------- ###
   ## Default Folders need to be set. This shold be in an .Rprofile file
   ### ---------------------------------------------- ###
```

```
    ## generally use:  set_default_rcreds_ALL(parent_folder = "~/.rcreds/")
    set_default_rcreds_ALL(parent_folder = file.path(tempdir(), ".rcreds/"),
                           create_if_not_exist = TRUE)
    ### ---------------------------------------------- ###

    library(rcreds)

    key <- create_key()

   file_creds <- write_credentials_to_file(username="cosmo", password="too many secrets", key=key)
    file_key   <- save_key(folder="different/key/location")

    ### IN ANOTHER FILE
    key <- read_key_from_file(folder="different/key/location")
   creds <- read_credentials_from_file(key=key, fail_if_cant_decrypt=FALSE, showWarnings=FALSE)
   ## normally, leave the above flags as their default TRUE. Using FALSE for this example only.

## End(Not run)
```

`setters_getters` *Setter Getter for Default rcreds Folders*

Description

Where will your keys and credentials be saved to and read from

Usage

```
get_default_rcreds_folder(DB = FALSE, check_if_exists = TRUE,
  fail_if_not_set = TRUE, showWarnings = TRUE)

set_default_rcreds_folder(folder, DB = FALSE, create_if_not_exist = FALSE,
  showWarnings = TRUE, verbose = TRUE)

clear_default_rcreds_folder(DB = FALSE, verbose = TRUE)

get_default_rcreds_key_folder(check_if_exists = TRUE,
  fail_if_not_set = TRUE, showWarnings = TRUE)

set_default_rcreds_key_folder(folder, create_if_not_exist = FALSE,
  showWarnings = TRUE, verbose = TRUE)

clear_default_rcreds_key_folder(verbose = TRUE)

set_default_rcreds_ALL(parent_folder, create_if_not_exist = FALSE,
  showWarnings = TRUE, verbose = TRUE)
```

Arguments

`DB`	TRUE/FALSE flag. Is the folder for the db credentials functions? Defaults to: FALSE
`check_if_exists`	TRUE/FALSE flag. Should we check if the folder exists Defaults to: TRUE
`fail_if_not_set`	TRUE/FALSE flag. Should an error be thrown if the option is not set? Defaults to: TRUE
`showWarnings`	When FALSE, warnings will be supressed. Defaults to: TRUE
`folder`	full path/to/folder where to store creds or key. A quoted string
`create_if_not_exist`	If folder does not exist, should it be created. Defaults to: FALSE
`verbose`	When FALSE, output will be supressed. Defaults to: TRUE
`parent_folder`	full path/to/parent_folder in which three subfolders will be (optionally) created to store credentials and keys

Details

Ideally, rcreds will be written to `~/.rcreds/`, but the package cannot set that as a default. The user must do so.

Value

The folder set in the options.

for the `set_default_..` functions, the folder is returned invisibly.

for the `clear_default_..` functions, the previously set value, invisibly.

Examples

```
## Not run:
  library(rcreds)

  set_default_rcreds_folder("~/.rcreds/credential_files")
  creds_folder <- get_default_rcreds_folder()
  creds_folder

  set_default_rcreds_folder("~/.rcreds/db_credential_files", DB=TRUE)
  db_creds_folder <- get_default_rcreds_folder(DB=TRUE)
  db_creds_folder

  set_default_rcreds_key_folder("~/.rcreds/key_files")
  rcreds_key_folder <- get_default_rcreds_key_folder()
```

```
    rcreds_key_folder

    ## ---------------------------------------------- ##

    ## Alternatively, set them all in one shot
    set_default_rcreds_ALL(parent_folder = "~/.rcreds")

    ## All three values will be set
    get_default_rcreds_folder()
    get_default_rcreds_folder(DB=TRUE)
    get_default_rcreds_key_folder()

## End(Not run)
```

Index