Package: credentials (via r-universe)

October 29, 2026

Type Package

Title Tools for Managing SSH and Git Credentials

Version 2.0.3

Description Setup and retrieve HTTPS and SSH credentials for use with 'git' and other services. For HTTPS remotes the package interfaces the 'git-credential' utility which 'git' uses to store HTTP usernames and passwords. For SSH remotes we provide convenient functions to find or generate appropriate SSH keys. The package both helps the user to setup a local git installation, and also provides a back-end for git/ssh client libraries to authenticate with existing user credentials.

License MIT + file LICENSE

SystemRequirements git (optional)

Encoding UTF-8

Imports openssl (>= 1.3), sys (>= 2.1), curl, jsonlite, askpass

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.2.1

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Language en-US

URL https://docs.ropensci.org/credentials/
https://r-lib.r-universe.dev/credentials

BugReports https://github.com/r-lib/credentials/issues

Config/pak/sysreqs git libssl-dev

Repository https://r-lib.r-universe.dev

Date/Publication 2025-09-12 08:55:50 UTC

RemoteUrl https://github.com/r-lib/credentials

RemoteRef HEAD

RemoteSha a656a37c4ba8ff9d59679c77153ddc887a1054ac

Contents

credential_api 2
credential_helper 3
http_credentials 3
set_github_pat 4
ssh_credentials 5

Index **7**

`credential_api` *Retrieve and store git HTTPS credentials*

Description

Low-level wrappers for the git-credential command line tool. Try the user-friendly git_credential_ask and git_credential_update functions first.

Usage

```
credential_fill(cred, verbose = TRUE)

credential_approve(cred, verbose = TRUE)

credential_reject(cred, verbose = TRUE)
```

Arguments

`cred`	named list with at least fields `protocol` and `host` and optionally also `path`, `username` ,`password`.
`verbose`	emit some useful output about what is happening

Details

The credential_fill function looks up credentials for a given host, and if none exists it will attempt to prompt the user for new credentials. Upon success it returns a list with the same `protocol` and `host` fields as the `cred` input, and additional `username` and `password` fields.

After you have tried to authenticate the provided credentials, you can report back if the credentials were valid or not. Call credential_approve and credential_reject with the `cred` that was returned by credential_fill in order to validate or invalidate a credential from the store.

Because git credential interacts with the system password manager, the appearance of the prompts vary by OS and R frontend. Note that credential_fill should only be used interactively, because it may require the user to enter credentials or unlock the system keychain. On the other hand credential_approve and credential_reject are non-interactive and could be used to save or delete credentials in a scripted program. However note that some credential helpers (e.g. on Windows) have additional security restrictions that limit use of credential_approve and credential_reject to credentials that were actually entered by the user via credential_fill. Here it is not possible at all to update the credential store without user interaction.

Examples

```
# Insert example cred
example <- list(protocol = "https", host = "example.org",
  username = "test", password = "secret")
credential_approve(example)

# Retrieve it from the store
cred <- credential_fill(list(protocol = "https", host = "example.org", path = "/foo"))
print(cred)

# Delete it
credential_reject(cred)
```

`credential_helper` *Credential Helpers*

Description

Git supports several back-end stores for HTTPS credentials called helpers. Default helpers include `cache` and `store`, see the git-credentials manual page for details.

Usage

```
credential_helper_list()

credential_helper_get(global = FALSE)

credential_helper_set(helper, global = FALSE)
```

Arguments

`global`	if FALSE the setting is done per git repository, if TRUE it is in your global user git configuration.
`helper`	string with one of the supported helpers from credential_helper_list

---

`http_credentials` *Load and store git HTTPS credentials*

Description

This requires you have the `git` command line program installed.The git_credential_ask function looks up a suitable username/password from the `git-credential` store. If none are available it will prompt the user for credentials which may be saved the store. On subsequent calls for the same URL, the function will then return the stored credentials without prompting the user.

Usage

```
git_credential_ask(url = "https://github.com", save = TRUE, verbose = TRUE)

git_credential_update(url = "https://github.com", verbose = TRUE)

git_credential_forget(url = "https://github.com", verbose = TRUE)
```

Arguments

url	target url, possibly including username or path
save	in case the user is prompted for credentials, attempt to remember them.
verbose	print errors from `git credential` to stdout

Details

The appearance and security policy of the credential store depends on your version of git, your operating system, your R frontend and which credential_helper is used. On Windows and MacOS the credentials are stored in the system password manager by default.

It should be assumed that reading credentials always involves user interaction. The user may be asked to unlock the system keychain or enter new credentials. In reality, user interaction is usually only required on the first authentication attempt, but the security policy of most credential helpers prevent you from programmatically testing if the credentials are already unlocked.

See Also

Other credentials: `ssh_credentials`

---

set_github_pat *Set your Github Personal Access Token*

Description

Populates the `GITHUB_PAT` environment variable using the git_credential manager, which `git` itself uses for storing passwords. The credential manager returns stored credentials if available, and securely prompt the user for credentials when needed.

Usage

```
set_github_pat(force_new = FALSE, validate = interactive(), verbose = validate)
```

Arguments

force_new	forget existing pat, always ask for new one.
validate	checks with the github API that this token works. Defaults to `TRUE` only in an interactive R session (not when running e.g. CMD check).
verbose	prints a message showing the credential helper and PAT owner.

Details

Packages that require a GITHUB_PAT can call this function to automatically set the GITHUB_PAT when needed. Users may call this function in their .Rprofile script to automatically set GITHUB_PAT for each R session without hardcoding any tokens on disk in plain-text.

Value

Returns TRUE if a valid GITHUB_PAT was set, and FALSE if not.

---

ssh_credentials *Managing Your SSH Key*

Description

Utility functions to find or generate your SSH key for use with git remotes or other ssh servers.

Usage

```
ssh_key_info(host = NULL, auto_keygen = NA)

ssh_keygen(file = ssh_home("id_ecdsa"))

ssh_setup_github()

ssh_home(file = NULL)

ssh_agent_add(file = NULL)

ssh_update_passphrase(file = ssh_home("id_rsa"))

ssh_read_key(file = ssh_home("id_rsa"), password = askpass)
```

Arguments

host	target host (only matters if you have configured specific keys per host)
auto_keygen	if TRUE automatically generates a key if none exists yet. Default NA is to prompt the user what to.
file	destination path of the private key. For the public key, .pub is appended to the filename.
password	a passphrase or callback function

Details

Use ssh_key_info() to find the appropriate key file on your system to connect with a given target host. In most cases this will simply be ssh_home('id_rsa') unless you have configured ssh to use specific keys for specific hosts.

To use your key to authenticate with GitHub, copy the pubkey from ssh_key_info() to your profile: https://github.com/settings/ssh/new.

If this is the first time you use ssh, ssh_keygen can help generate a key and save it in the default location. This will also automatically opens the above Github page in your browser where you can add the key to your profile.

ssh_read_key reads a private key and caches the result (in memory) for the duration of the R session. This prevents having to enter the key passphrase many times. Only use this if ssh-agent is not available (i.e. Windows)

See Also

Other credentials: http_credentials

Index

∗ **credentials**
    http_credentials, 3
    ssh_credentials, 5
.Rprofile, *5*

credential_api, 2
credential_approve, *2*
credential_approve *(*credential_api*)*, 2
credential_fill, *2*
credential_fill *(*credential_api*)*, 2
credential_helper, 3, *4*
credential_helper_get
        *(*credential_helper*)*, 3
credential_helper_list, *3*
credential_helper_list
        *(*credential_helper*)*, 3
credential_helper_set
        *(*credential_helper*)*, 3
credential_reject, *2*
credential_reject *(*credential_api*)*, 2
credentials *(*http_credentials*)*, 3

git_credential, *4*
git_credential_ask, *2*, *3*
git_credential_ask *(*http_credentials*)*, 3
git_credential_forget
        *(*http_credentials*)*, 3
git_credential_update, *2*
git_credential_update
        *(*http_credentials*)*, 3

http_credentials, 3, *6*

set_github_pat, 4
ssh_agent_add *(*ssh_credentials*)*, 5
ssh_credentials, *4*, 5
ssh_home *(*ssh_credentials*)*, 5
ssh_key_info *(*ssh_credentials*)*, 5
ssh_key_info(), *6*
ssh_keygen, *6*
ssh_keygen *(*ssh_credentials*)*, 5
ssh_read_key *(*ssh_credentials*)*, 5
ssh_setup_github *(*ssh_credentials*)*, 5
ssh_update_passphrase
        *(*ssh_credentials*)*, 5