Package 'gitcreds'

September 8, 2022

Title Query 'git' Credentials from 'R'

Version 0.1.2

Description Query, set, delete credentials from the 'git' credential store. Manage 'GitHub' tokens and other 'git' credentials. This package is to be used by other packages that need to authenticate to 'GitHub' and/or other 'git' repositories.

License MIT + file LICENSE

URL https://gitcreds.r-lib.org/, https://github.com/r-lib/gitcreds

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

Depends R (>= 3.4)

Suggests codetools, covr, knitr, mockery, oskeyring, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate

Encoding UTF-8

RoxygenNote 7.2.1.9000

SystemRequirements git

Config/testthat/edition 3

NeedsCompilation no

Author Gábor Csárdi [aut, cre], RStudio [cph, fnd]

Maintainer Gábor Csárdi <csardi.gabor@gmail.com>

Repository CRAN

Date/Publication 2022-09-08 10:42:55 UTC

R topics documented:

gitcreds_cache_envvar 2
gitcreds_fill 3

gitcreds_get 4
gitcreds_list 8
gitcreds_parse_output 10

Index **12**

`gitcreds_cache_envvar` *Environment variable to cache the password for a URL*

Description

`gitcreds_get()` caches credentials in environment variables. `gitcreds_cache_envvar()` calculates the environment variaable name that is used as the cache, for a URL.

Usage

```
gitcreds_cache_envvar(url)
```

Arguments

`url`	Character vector of URLs, they may contain user names and paths as well. See details below.

Value

Character vector of environment variables.

See Also

`gitcreds_get()`.

Examples

```
gitcreds_cache_envvar("https://github.com")
gitcreds_cache_envvar("https://api.github.com/path/to/endpoint")
gitcreds_cache_envvar("https://jane@github.com")
gitcreds_cache_envvar("https://another.site.github.com")
```

`gitcreds_fill` *Access the low level credential API*

Description

These function are primarily for package authors, who want more control over the user interface, so they want to avoid calling `gitcreds_get()` and `gitcreds_set()` directly.

Usage

```
gitcreds_fill(input, args = character(), dummy = TRUE)

gitcreds_approve(creds, args = character())

gitcreds_reject(creds, args = character())
```

Arguments

`input`	Named list to pass to `git credential fill`.
`args`	Extra args, used *before* `fill`, to allow `git -c ... fill`.
`dummy`	Whether to append a dummy credential helper to the list of credential helpers.
`creds`	`gitcreds` object (named list) to add or remove.

Details

`gitcreds_fill()` calls `git credential fill` to query git credentials.

`gitcreds_approve()` calls `git credential approve` to add new credentials.

Value

The standard output of the `git` command, line by line.

See Also

`gitcreds_parse_output()` to parse the output of `gitcreds_fill()`.

`gitcreds_get` *Query and set git credentials*

Description

This manual page is for *users* of packages that depend on gitcreds for managing tokens or passwords to GitHub or other git repositories. If you are a package author and want to import gitcreds for this functionality, see `vignette("package", package = "gitcreds")`. Otherwise please start at 'Basics' below.

Usage

```
gitcreds_get(url = "https://github.com", use_cache = TRUE, set_cache = TRUE)

gitcreds_set(url = "https://github.com")

gitcreds_delete(url = "https://github.com")

gitcreds_list_helpers()
```

Arguments

`url`	URL to get, set or delete credentials for. It may contain a user name, which is typically (but not always) used by the credential helpers. It may also contain a path, which is typically (but not always) ignored by the credential helpers.
`use_cache`	Whether to try to use the environment variable cache before turning to git to look up the credentials for `url`. See `gitcreds_cache_envvar()`.
`set_cache`	Whether to set the environment variable cache after receiving the credentials from git. See `gitcreds_cache_envvar()`.

Value

`gitcreds_get()` returns a `gitcreds` object, a named list of strings, the fields returned by the git credential handler. Typically the fields are `protocol`, `host`, `username`, `password`. Some credential helpers support path-dependent credentials and also return a `path` field.

`gitcreds_set()` returns nothing.

`gitcreds_delete()` returns `FALSE` if it did not find find any credentials to delete, and thus it did not call `git credential reject`. Otherwise it returns `TRUE`.

`gitcreds_get()` errors if git is not installed, no credential helpers are configured or no credentials are found. `gitcreds_set()` errors if git is not installed, or setting the new credentials fails. `gitcreds_delete()` errors if git is not installed or the git calls fail. See `vignette("package", package = "gitcreds")` if you want to handle these errors.

`gitcreds_list_helpers()` returns a character vector, corresponding to the `credential.helper` git configuration key. Usually it contains a single credential helper, but it is possible to configure multiple helpers.

Basics

`gitcreds_get()` queries git credentials. It is typically used by package code that needs to authenticate to GitHub or another git repository. The end user might call `gitcreds_get()` directly to check that the credentials are properly set up.

`gitcreds_set()` adds or updates git credentials in the credential store. It is typically called by the user, and it only works in interactive sessions. It always asks for acknowledgement before it overwrites existing credentials.

`gitcreds_delete()` deletes git credentials from the credential store. It is typically called by the user, and it only works in interactive sessions. It always asks for acknowledgement.

`gitcreds_list_helpers()` lists the active credential helpers.

git versions:

These functions use the `git credential` system command to query and set git credentials. They need an external git installation. You can download git from https://git-scm.com/downloads. A recent version is best, but at least git 2.9 is suggested.

gitcreds should work out of the box on macOS with git versions 2.9.2 or later, and on Windows with git versions 2.12.1 or later, using the default git settings. On Windows, for git versions from 2.9.2 up until 2.12.1 you probably need to set the default credential helper to `wincred`. It is usually simpler to update git to a recent version.

To see your current git version run `git --version` from your shell. Or from R:

```
system("git --version")
```

If you need to avoid installing git, see 'Environment variables' below.

GitHub:

New setup:
To set up password-less authentication to GitHub:

1. Create a personal access token (PAT). See https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token.
2. Call `gitcreds_set()` and give this token as the password.
3. Run `gitcreds_get(use_cache = FALSE)` to check that the new PAT is set up. To see the token, you can run `gitcreds_get(use_cache = FALSE)$password`.

Migrating from the `GITHUB_PAT` *environment variable:*
If you already have a GitHub token, and use the `GITHUB_PAT` or `GITHUB_TOKEN` environment variable in your `.Renviron` file or elsewhere, no changes are neccessary. gitcreds will automatically use this variable.
However, we still suggest that you add your token to the git credential store with `gitcreds_set()` and remove `GITHUB_PAT` from your `.Renviron` file. The credential store is more secure than storing tokens in files, and command line git also uses the credential store for password-less authentication.

Advanced topics

Cached credentials:

Because querying the git credential store might not be very fast, `gitcreds_get()` caches credentials in environment variables by default. Credentials for different URLs are stored in different environment variables. The name of the environment variable is calculated with `gitcreds_cache_envvar()`. To remove the cache, remove this environment variable with `Sys.unsetenv()`.

Environment variables:

If you want to avoid installing git, or using the credential store for some reason, you can supply credentials in environment variables, e.g. via the `.Renviron` file. Use `gitcreds_cache_envvar()` to query the environment variable you need to set for a URL:

1. You can set this environment variable to the token or password itself.
2. If you also need a user name, then use the `user:password` form, i.e. separate them with a colon. (If your user name or passwrd has `:` characters, then you need to escape them with a preceding backslash.)

Proxies:

git should pick up the proxy configuration from the `http_proxy`, `https_proxy`, and `all_proxy` environment variables. To override these, you can set the `http.proxy` git configuration key. More info here: https://git-scm.com/docs/git-config#Documentation/git-config.txt-httpproxy and here: https://github.com/microsoft/Git-Credential-Manager-Core/blob/master/docs/netconfig.md

Credential helpers:

git credential helpers are an extensible, configurable mechanism to store credentials. Different git installations have different credentials helpers. On Windows the default helper stores credentials in the system credential store. On macOS, it stores them in the macOS Keychain. Other helpers cache credentials in a server process or in a file on the file system.

gitcreds only works if a credential helper is configured. For the current git version (2.29.0), this is the case by default on Windows and macOS (for git from HomeBrew), but most Linux distributions do not set up a default credential helper.

You can use `gitcreds_list_helpers()` to see the *active* credential helper(s) for a repository. Make sure you set the working directory to the git tree before calling `gitcreds_list_helpers()`.

The current working directory:

git allows repository specific configuration, via the `.git/config` file. The `config` file might specify a different credential helper, a different user name, etc. This means that `gitcreds_get()` etc. will potentially work differently depending on the current working directory. This is especially relevant for package code that changes the working directory temporarily.

Non-GitHub accounts:

Non-GitHub URLs work mostly the same way as GitHub URLs. `gitcreds_get()` and `gitcreds_set()` default to GitHub, so you'll need to explicitly set their `url` argument.

Some credential helpers, e.g. Git Credential Manager for Windows (`manager`) and Git Credential Manager Core (`manager-core`) work slightly differently for GitHub and non-GitHub URLs, see their documentation for details.

Multiple accounts:

The various credential helpers support having multiple accounts on the same server in different ways. Here are our recommendations.

macOS:

1. Use the (currently default) `osxkeychain` credential helper.
2. In Keychain Access, remove all your current credentials for the host(s) you are targeting. E.g. for GitHub, search for github.com Internet Passwords.
3. Then add the credential that you want to use for "generic access". This is the credential that will be used for URLs without user names. The user name for this credential does not matter, but you can choose something descriptive, e.g. "token", or "generic".
4. Configure git to use this username by default. E.g. if you chose "generic", then run

   ```
   git config --global credential.username generic
   ```
5. Add all the other credentials, with appropriate user names. These are the user names that you need to put in the URLs for the repositories or operations you want to use them for. (GitHub does not actually use the user names if the password is a PAT, but they are used to look up the correct token in the credential store.)

Windows with git 2.29.0 or later:

1. We suggest that you update to the latest git version, but at least 2.29.0, and use the `manager-core` helper which is now default. If you installed `manager-core` separately from git, we suggest that you remove it, because it might cause confusion as to which helper is actually used.
2. Remove all current credentials first, for the host you are targeting. You can do this in 'Credential Manager' or `gitcreds::gitcreds_list()` to find them and 'Credential Manager' or the oskeyring package to remove them. You can also use the oskeyring package to back up the tokens and passwords.
3. Then add the credential that you want to use for "generic access". This is the credential that will be used for URLs without user names. The user name for this credential does not matter, but you can choose something descriptive, e.g. "PersonalAccessToken", "token", or "generic".
4. Configure git to use this username by default. E.g. if you chose "generic", then run

   ```
   git config --global credential.username generic
   ```
5. Add all the other credentials, with appropriate user names. These are the user names that you need to put in the URLs for the repositories or operations you want to use them for. (GitHub does not actually use the user names if the password is a PAT, but they are used to look up the correct token from the credential store.)

Windows with older git versions, 2.28.0 and before:

A single GitHub account:

If you only need to manage a single github.com credential, together with possibly multiple credentials to other hosts (including GitHub Enterprise hosts), then you can use the default `manager` helper, and get away with the default auto-detected GCM authority setting.

In this case, you can add your github.com credential with an arbitrary user name, and for each other host you can configure a default user name, and/or include user names in the URLs to these hosts. This is how to set a default user name for a host called `https://example.com`:

```
git config --global credential.https://example.com.username myusername
```

Multiple GitHub credentials:

If you need to manage multiple github.com credentials, then you can still use the `manager` helper, but you need to change the GCM authority by setting an option or an environment variable, see https://github.com/microsoft/Git-Credential-Manager-for-Windows/blob/master/Docs/Configuration.md#authority.

This is how to change GCM authority in the config:

`git config --global credential.authority Basic`

You can also change it only for github.com:

`git config --global credential.github.com.authority Basic`

Then you can configure a default user name, this will be used for URLs without a user name:

`git config --global credential.username generic`

Now you can add you credentials, the default one with the "generic" user name, and all the others with their specific user and host names.
Alternatively, you can install a newer version of Git Credential Manager Core (GCM Core), at least version 2.0.252-beta, and use the `manager-core` helper. You'll potentially need to delete the older `manager-core` helper that came with git itself. With the newer version of GCM Core, you can use the same method as for newer git versions, see above.

Multiple credential helpers:

It is possible to configure multiple credential helpers. If multiple helpers are configured for a repository, then `gitcreds_get()` will go over them until a credential is found. `gitcreds_set()` will try to set the new credentials in *every* configured credential helper.

You can use `gitcreds_list_helpers()` to list all configured helpers.

Examples

```
## Not run:
gitcreds_get()
gitcreds_get("https://github.com")
gitcreds_get("https://myuser@github.com/myorg/myrepo")

## End(Not run)
```

---

gitcreds_list *List all credentials stored by a git credential helper*

Description

This function is meant to be used interactively, to help you when configuring credential helpers. It is especially useful if you have multiple accounts on a host.

Usage

```
gitcreds_list(
  url = "https://github.com",
  credential_helper = NULL,
  protocol = NULL
)
```

Arguments

`url`	URL to list credentials for. If `NULL` then the credentials are listed for all URLs. Note that for a host the results might be different if you specify or omit this argument. `gitcreds_list()` uses heuristics when the `url` is not specified. If is always best to specify the URL.
`credential_helper`	Credential helper to use. If this is `NULL`, then the configured credential helper is used. If multiple credential helpers are configured, then the first one is used, with a warning.
`protocol`	Protocol to list credentials for. If `NULL` and `url` includes a protocol then that is used. Otherwise `"https"` is used.

Details

Note that this function does not use the credential helper itself, so it does not have to be installed. But it may also give false results, so interpret the results with caution, and also use the tool provided by your OS, to look at the credentials: 'Keychain Access' on macOS and 'Credential Manager' on Windows.

Only a small number of credential helpers are supported currently. Here is a brief description of each.

`osxkeychain` **on macOS:**

This is the default credential helper on macOS.

It has some peculiarities:

- If you don't specify a username in the URL, then it will return the *oldest* credentials that match the specified host name, with an arbitrary user name.
- If the user name is specified in the URL, then it is used to look up the credentials.

To change or delete the listed credentials, see the oskeyring package or the 'Keychain Access' macOS app.

`manager`**, on Windows:**

This is Git Credential Manager for Windows, see https://github.com/microsoft/Git-Credential-Manager-for-Windows

It is currently the default helper on Windows, included in the git installer.

It has some oddities, especially with multiple GitHub users:

- The `github` authority (which is used by default for `github.com` URLs) cannot handle multiple users. It always sets the `target_name` of the Windows credential to `git:<URL>` where `<URL>` does not contain the user name. Since `target_name` is a primary key, it is not possible to add multiple GitHub users with the default configuration.
- To support multiple users, switch to the `Basic` authority, e.g. by setting the `GCM_AUTHORITY` env var to `Basic`. Then the user name will be included in `target_name`, and everything works fine.
- For this helper `gitcreds_list()` lists all records with a matching host name.

`manager-core` **on Windows:**

This is Git Credential Manager Core, see https://github.com/microsoft/Git-Credential-Manager-Core

On Windows it behaves almost the same way as `manager`, with some differences:

- Instead of *authorities*, it has providers. `github.com` URLs use the `github` provider by default. For better support for multiple GitHub accounts, switch to the `generic` provider by setting the `GCM_PROVIDER` env var to `generic`.
- `gitcreds_list()` will list all credentials with a matching host, irrespectively of the user name in the input URL.

`manager-core`**, *before* version 2.0.246-beta, on macOS:**

This is Git Credential Manager Core, see https://github.com/microsoft/Git-Credential-Manager-Core

This helper has some peculiarities w.r.t. user names:

- If the "github" provider is used (which is the default for `github.com` URLs), then it completely ignores user names, even if they are explicitly specified in the query.
- For other providers, the user name (if specified) is saved in the Keychain item.
- For this helper, `gitcreds_list()` always lists all records that match the *host*, even if the user name does not match, because it is impossible to tell if the user name would be used in a proper git credential lookup.

To change or delete the listed credentials, see the oskeyring package or the 'Keychain Access' macOS app.

`manager-core`**, version 2.0.246-beta or newer, on macOS:**

This is a newer version of Git Credential Manager Core, that supports multiple users better:

- if a user name is provided, then it saves it in the credential store, and it uses this user name for looking up credentials, even for the `github` provider.
- `gitcreds_list()` always lists all records that match the host, even if the user name does not match.
- Credentials that were created by an older version of `manager-core`, with the `generic` provider, do not work with the newer version of `manager-core`, because the format of the Keychain item is different.

Value

A list of `oskeyring_macos_item` objects. See `oskeyring::macos_item()`.

`gitcreds_parse_output` *Parse standard output from* `git credential fill`

Description

Parse standard output from `git credential fill`

Usage

```
gitcreds_parse_output(txt, url)
```

Arguments

txt	Character vector, standard output lines from `git credential fill`.
url	URL we queried, to be able to create a better error message.

Details

For dummy credentials (i.e. the lack of credentials), it throws an error of class `gitcreds_no_credentials`.

Value

`gitcreds` object.

Index

gitcreds *(*gitcreds_get*)*, 4
gitcreds_approve *(*gitcreds_fill*)*, 3
gitcreds_cache_envvar, 2
gitcreds_cache_envvar(), *4*, *6*
gitcreds_delete *(*gitcreds_get*)*, 4
gitcreds_fill, 3
gitcreds_get, 4
gitcreds_get(), *2*, *3*
gitcreds_list, 8
gitcreds_list_helpers *(*gitcreds_get*)*, 4
gitcreds_list_helpers(), *8*
gitcreds_parse_output, 10
gitcreds_parse_output(), *3*
gitcreds_reject *(*gitcreds_fill*)*, 3
gitcreds_set *(*gitcreds_get*)*, 4
gitcreds_set(), *3*

oskeyring::macos_item(), *10*

Sys.unsetenv(), *6*