GitHub Authentication

1 Accessing Git Repositories on GitHub

Git repositories on GitHub can be accessed using either the HTTPS/HTTP or SSH protocols. The HTTPS/HTTP protocol is easier to use relative to the SSH protocol, as the former requires less configuration. Therefore, the use of the HTTPS/HTTP protocol is recommended. The use of the SSH protocol is more complicated as it requires managing SSH key pairs and configuration of SSH client software.

2 Accessing Git Repositories on GitHub via the HTTPS/HTTP Protocol

To configure GitHub for access via the HTTPS/HTTP protocol, a personal-access token (PAT) is required for the GitHub account. For detailed instructions on how to create a PAT, see:

- https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token

The PAT created should have the following scopes:

- repo (i.e., all scopes in the repo category)

Note that the PAT is only viewable at the time that it is first created. After that time, the PAT can no longer be viewed. So, it must be saved somewhere (securely) for future use. Since a PAT can be used to gain access to GitHub, it is extremely important to store the PAT securely.

With GitHub, a repository URL using the HTTPS/HTTP protocol has the general form:

- `https://${USER}@github.com/${ORG}/${REPO}.git`

where `${USER}` is a username (which is ignored by GitHub), `${ORG}` is the GitHub organization or user associated with the repository, and `${REPO}` is the repository name.

The username `${USER}` is ignored by GitHub. The password must be a PAT. The username is ignored by GitHub as the PAT implicitly specifies a GitHub user.

Although GitHub ignores the username `${USER}` in the Git URL, this username may be useful to other software. For example, the username can be useful for credential-caching purposes. If multiple credentials are to be cached simultaneously (e.g., to accommodate the use of multiple GitHub accounts), the username in the Git URL can be used to distinguish between users for credential caching. For example, suppose that a user has a GitHub account called `home-account` for personal use and a work account called `work-account` for work use. To allow the passwords (i.e., PATs) for both accounts to be cached simultaneously, Git repositories that are to be accessed using the `home-account` GitHub user could be specified with `${USER}` in the Git URL set to `home-account`, while Git repositories that are to be accessed using the `work-account` GitHub user could be specified with `${USER}` in the Git URL set to `work-account`. Then, when the credential-caching software is queried for a password (i.e., PAT), it will be provided with a different username (i.e., `home-account` or `work-account`) in each case, allowing for the password (i.e., PAT) for the correct account to be provided. Again, GitHub ignores the value of `${USER}`. So, changing this value has no impact on how GitHub processes a request. The value of `${USER}`, however, is used when the credential cache is queried. In particular, the password (i.e., PAT) for the user `${USER}` is requested. So, changing the value of `${USER}` does impact which password (i.e., PAT) is returned by the credential cache.

3 Accessing Git Repositories on GitHub via the SSH Protocol

To access Git repositories on GitHub via the SSH protocol, a public/private key pair is needed and the GitHub account and SSH client software must be configured to use this key. This configuration consists of the following steps (in order):

1. Create a public/private key pair. For more details, see:
   - https://docs.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
2. Add the private key to the SSH client. For more details, again see:
   - https://docs.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
3. Add the public key to the GitHub account. For more details, see:
   - https://docs.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account

For further details on how to configure a GitHub account with a SSH key, one can also (if needed) refer to:

- https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh

With GitHub, a Git repository URL using the SSH protocol has the general form:

- `ssh://${USER}@github.com/${ORG}/${REPO}.git`

where `${USER}` is a username (which must be `git`), `${ORG}` is the GitHub organization or user associated with the repository, and `${REPO}` is the repository name.

The username `${USER}` in the URL must be `git`. The actual GitHub user associated with the repository is determined by the SSH key used for authentication.

If the `ssh` program is being used, it is important to note that this program may remap hostnames depending on the contents of the `ssh` configuration file. This can be very useful if one needs to use multiple SSH keys with GitHub (which is sometimes necessary when a user has multiple GitHub accounts). For example, entries like the following could be used in the `ssh` configuration file in order to use a different SSH key, depending on whether the hostname given to the `ssh` program is specified as `work.github.com` or `home.github.com` in the URL for a Git repository:

```
# Settings to use for work account.
Hostname work.github.com
        Username git
        Hostname github.com
        IdentityFile ~/.ssh/key_file_for_work_account

# Settings to use for personal account.
Hostname home.github.com
        Username git
        Hostname github.com
        IdentityFile ~/.ssh/key_file_for_home_account
```

In this example, the hostnames `work.github.com` and `home.github.com` are both remapped by the `ssh` program to the hostname `github.com`. The same username `git` is used in both cases. The key file used in each case, however, is different.

The public keys configured for a GitHub account can be accessed via a URL of the form

- `https://github.com/${USER}.keys`

where `${USER}` is the username of the GitHub account. For example, to show the SSH public key for the GitHub account `adamstac` (which is not an account owned by the course instructor), simply access the following URL:

- https://github.com/adamstac.keys

At the time of this writing, this returned the following:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA1cXZQat1072d/Y5AD9y8GnamOiEDiel7/0
FVCsUTqF7WAJXvWEMlLMmqspeVMaRPSJvb1xzm5z/UUGDi6QscBx4IAVJ/OimJXS7NsNUAZJPC/
efVNn6RO+NIpfHPDpES6EoSXLHxLPyLP4OeNXxMylyzF0h+iboWqKaCKsKdSglgo9t/
zgZW4kG0dtGnQzmqrbYf3KOkV0MZOEeDyacmGWm+rcZ0mRyVAmnzd0ncHRy1YRgEnTqauX6Z3Bo+
qSfkj0cC6vl2HDodM9TRkYkPGSNeUOhj/
K48uIHCLF6bKBVjJeHQ7nmSxZErXFsw52uUcjhYqdXgXeRDGOUP6X2k7w==
```

(Note that the original text is all on a single line. Line breaks were added above so that the text would not run outside the page margins.)

4 Credential Caching

During the authentication process, a user needs to provide a password/passphrase. Since having to enter a password repeatedly is inconvenient, a credential cache is often extremely useful. A credential cache remembers each password so that it only needs to be provided by the user when requested for the first time. In the case of subsequent requests (assuming that the cache entry does not expire), the password is provided automatically by the credential caching software (using the cached password).

On Linux systems, two credential caches are commonly used with Git:

- libsecret (via `git-credential-libsecret`)
- Git credential cache (i.e., `git-credential-cache`) (https://git-scm.com/docs/git-credential-cache)

Only relatively newer versions of Git (2.16 or later?) have support for credential caching via libsecret. If libsecret is supported by the version of Git installed, the program `git-credential-libsecret` should be available somewhere on the system (such as in the `/usr/libexec/git-core` directory). This program is the one that needs to be specified to Git as the credential helper program in order for the libsecret credential cache to be used by Git. The use of libsecret is recommended if available.

[Note: Last time that I checked, the version of Git installed on the UGLS lab machines did not appear to have support for libsecret. The VM disk images for the course should have support for libsecret, however.]

4.1 Credential Caching via libsecret

The libsecret library provides access to keyrings that store credentials. Provided that the version of Git being used is new enough, it should have support for libsecret.

To enable the use of the libsecret credential cache with Git, use the command:

```
git config --global credential.helper \
  /usr/libexec/git-core/git-credential-libsecret
```

The cache maintained by libsecret can also be accessed using a number of other tools. A command-line interface (CLI) front-end to libsecret is provided by the `secret-tool` program. A GUI front-end to libsecret is available via the `seahorse` program. The GUI is quite straightforward to use. So, only examples of the CLI are given herein.

Example (using `secret-tool`*).* Suppose that the GitHub user `jdoe` has a GitHub PAT stored in the file `github_token_file`. The PAT can be placed in the libsecret cache for GitHub access via the HTTPS/HTTP protocol using the command:

```
secret-tool store --label "GitHub Account jdoe" \
  xdg:schema org.gnome.keyring.NetworkPassword \
  server github.com protocol https user jdoe < github_token_file
```

After the PAT is added to the cache, the credential cache will automatically provide the PAT whenever requested subsequently.

4.2 Credential Caching via Git Credential Cache

If the version of Git being used does not support libsecret, the Git credential cache can be used for caching credentials. To enable the global caching of user credentials for 1 hour (i.e., 3600 seconds), use the command:

```
git config --global credential.helper 'cache --timeout=3600'
```

4.3 Disabling Git Credential Caching

Sometimes, it may be desirable to disable credential caching in Git (regardless of whether the libsecret library or Git credential cache is being used for caching). To disable all caching of user credentials (i.e., at the system, global, and repository levels) and purge any cached values, use the following command sequence:

```
git config --unset credential.helper
git config --global --unset credential.helper
git config --system --unset credential.helper
git credential-cache exit
```

4.4 Avoiding Pop-Up Windows for Entering Credentials

When a password must be directly entered by a user, this can potentially be accomplished by using either the terminal or a GUI. In some cases, the use of a GUI may not be desirable (e.g., when remote login is used). If a GUI is not desired, it is possible to disable the use of one. To ensure that prompting for user credentials employs the terminal (as opposed to, say, a pop-up window), use the following command sequence:

```
git config --unset core.askPass
git config --global --unset core.askPass
git config --system --unset core.askPass
unset GIT_ASKPASS
unset SSH_ASKPASS
```

4.5 Accessing the Gnome Keyring Without Graphical Login

The libsecret library stores its passwords in the Gnome keyring. Normally, when a graphical desktop is used, the login session is automatically configured (at login time) with access to the Gnome keyring. When a graphical desktop is not used (such as in the case of remote login), some set up is required to gain access to the Gnome keyring. This can be accomplished with the following command sequence:

```
dbus-run-session bash
eval $(gnome-keyring-daemon -r)
export GNOME_KEYRING_CONTROL
export SSH_AUTH_SOCK
```

The `dbus-run-session bash` command will start a new (Bash) shell that is configured with access to the Gnome keyring.