

golang / glog Public[Code](#) [Pull requests](#) 2 [Actions](#) [Projects](#) [Security and quality](#) [Insights](#)

Fail if log file already exists #74

[New issue](#)[Merged](#)

stapelberg merged 3 commits into `golang:master` from `chressie:fail-already-exist` on Jan 13, 2025

[Conversation](#) 11[Commits](#) 3[Checks](#) 0[Files changed](#) 2

Changes from **1 commit** [File filter](#) [Conversations](#) [Jump to](#) [Settings](#)

glog: have createInDir fail if the file already exists

[< Prev](#) [Next >](#)

This prevents an attack like the one described

[here](https://owasp.org/www-community/vulnerabilities/Insecure_Temporary_File#:~:text=0n%20Unix%20based,with%20elevated%20permissions.).

An unprivileged attacker could use symlinks to trick a privileged logging process to follow a symlink from the log dir and write logs over an arbitrary file.

The components of the log names are program, host, username, tag, date, time and PID. These are all predictable. It's not at all unusual for the logdir to be writable by unprivileged users, and one of the fallback directories (`/tmp`) traditionally has broad write privs with the sticky bit set on Unix systems.

As a concrete example, let's say I've got a glog-enabled binary running as a root cronjob. I can gauge when that cron job will run and then use a bash script to spray the log dir with glog-looking symlinks to `/etc/shadow` with predicted times and PIDs. When the cronjob runs, the `os.Create` call will follow the symlink, truncate `/etc/shadow` and then fill it with logs.





This change defeats that by setting `O_EXCL`, which will cause the open call to fail if the file already exists.

Fixes [CVE-2024-45339](#)

c1/712795111 (google-internal)

 **chressie** committed on Jan 10, 2025

commit b8741656e406e66d6992bc2c9575e460ecaa0ec2

  7  glog_file.go 

143	143	func createInDir(dir, tag string, t time.Time) (f *os.File, name string, err error) {
144	144	name, link := logName(tag, t)
145	145	fname := filepath.Join(dir, name)
146		- f, err = os.Create(fname)
	146	+ // O_EXCL is important here, as it prevents a vulnerability. The general idea is that logs often
	147	+ // live in an insecure directory (like /tmp), so an unprivileged attacker could create fname in
	148	+ // advance as a symlink to a file the logging process can access, but the attacker cannot. O_EXCL
	149	+ // fails the open if it already exists, thus prevent our this code from opening the existing file
	150	+ // the attacker points us to.
	151	+ f, err = os.OpenFile(fname, os.O_RDWR os.O_CREATE os.O_EXCL, 0666)
147	152	if err == nil {
148	153	symlink := filepath.Join(dir, link)
149	154	os.Remove(symlink) // ignore err

  11  glog_test.go 

772	772	len(c), logsink.MaxLogMessageLen, c)
773	773	}
774	774	}
	775	+
	776	+ func TestCreateFailsIfExists(t *testing.T) {
	777	+ tmp := t.TempDir()
	778	+ now := time.Now()
	779	+ if _, _, err := create("INFO", now, tmp); err != nil {
	780	+ t.Errorf("create() failed on first call: %v", err)
	781	+ }
	782	+ if _, _, err := create("INFO", now, tmp); err == nil {
	783	+ t.Errorf("create() succeeded on second call, want error")
	784	+ }
	785	+ }