# coordinate-embargoed-releases

Content-type: text/asciidoc Abstract: When a vulnerability is reported, we follow these guidelines to assess the vulnerability, create and review a fix, and coordinate embargoed security releases.

How we coordinate embargoed releases ----------

To protect Git users from critical vulnerabilities, we do not just release fixed versions like regular maintenance releases. Instead, we coordinate releases with packagers, keeping the fixes under an embargo until the release date. That way, users will have a chance to upgrade on that date, no matter what Operating System or distribution they run.

The `git-security` mailing list ----------

Responsible disclosures of vulnerabilities, analysis, proposed fixes as well as the orchestration of coordinated embargoed releases all happen on the `git-security` mailing list at <git-security@googlegroups.com>.

In this context, the term "embargo" refers to the time period that information about a vulnerability is kept under wraps and only shared on a need-to-know basis. This is necessary to protect Git's users from bad actors who would otherwise be made aware of attack vectors that could be exploited. "Lifting the embargo" refers to publishing the version that fixes the vulnerabilities.

Audience of the `git-security` mailing list ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Anybody may contact the `git-security` mailing list by sending an email to <git-security@googlegroups.com>, though the archive is closed to the public and only accessible to subscribed members.

There are a few dozen subscribed members: core Git developers who are trusted with addressing vulnerabilities, and stakeholders (i.e. owners of products affected by security vulnerabilities in Git).

Most of the discussions revolve around assessing the severity of the reported issue (including the decision whether the report is security-relevant or can be redirected to the public mailing list), how to remediate the issue, determining the timeline of the disclosure as well as aligning priorities and requirements.

Communications ~~~~~~~~~~~~~~~

If you are a stakeholder, it is a good idea to pay close attention to the discussions, as pertinent information may be buried in the middle of a lively conversation that might not look relevant to your interests. For example, the tentative timeline might be agreed upon in the middle of discussing code comment formatting in one of the patches and whether or not to combine fixes for multiple, separate vulnerabilities into the same embargoed release. Most mail threads are not usually structured specifically to communicate agreements, assessments or timelines.

Typical timeline ----------

- A potential vulnerability is reported to the `git-security` mailing list.
- The members of the git-security list start a discussion to give an initial assessment of the severity of the reported potential vulnerability. We aspire to do so within a few days.
- After discussion, if consensus is reached that it is not critical enough to warrant any embargo, the reporter is redirected to the public Git mailing list. This ends the reporter's interaction with the `git-security` list.
- If it is deemed critical enough for an embargo, ideas are presented on how to address the vulnerability.
- Usually around that time, the Git maintainer or their delegate(s) open a draft security advisory in the `git/git` repository on GitHub (see below for more details).
- Code review can take place in a variety of different locations, depending on context. These are: patches sent inline on the git-security list, a private fork on GitHub associated with the draft security advisory, or the git/cabal repository.
- Contributors working on a fix should consider beginning by sending patches to the git-security list (inline with the original thread), since they are accessible to all subscribers, along with the original reporter.
- Once the review has settled and everyone involved in the review agrees that the patches are nearing the finish line, the Git maintainer, and others determine a release date as well as the release trains that are serviced. The decision regarding which versions need a backported fix is based on input from the reporter, the contributor who worked on the patches, and from stakeholders. Operators of hosting sites who may want to analyze whether the given issue is exploited via any of the repositories they host, and binary packagers who want to make sure their product gets patched adequately against the vulnerability, for example, may want to give their input at this stage.
- While the Git community does its best to accommodate the specific timeline requests of the various binary packagers, the nature of the issue may preclude a prolonged release schedule. For fixes deemed urgent, it may be in the best interest of the Git users community to shorten the disclosure and release timeline, and packagers may need to adapt accordingly.

- Subsequently, branches with the fixes are pushed to the git/cabal repository.
- The tags are created by the Git maintainer and pushed to the same repository.
- The Git for Windows, Git for macOS, BSD, Debian, etc. maintainers prepare the corresponding release artifacts, based on the tags created that have been prepared by the Git maintainer.
- The release artifacts prepared by various binary packagers can be made available to stakeholders under embargo via a mail to the `git-security` list.
- Less than a week before the release, a mail with the relevant information is sent to <distros@vs.openwall.org> (see below), a list used to pre-announce embargoed releases of open source projects to the stakeholders of all major distributions of Linux as well as other OSes.
- Public communication is then prepared in advance of the release date. This includes blog posts and mails to the Git and Git for Windows mailing lists.
- On the day of the release, at around 10am Pacific Time, the Git maintainer pushes the tag and the `master` branch to the public repository, then sends out an announcement mail.
- Once the tag is pushed, the Git for Windows maintainer publishes the corresponding tag and creates a GitHub Release with the associated release artifacts (Git for Windows installer, Portable Git, MinGit, etc).
- Git for Windows release is then announced via a mail to the public Git and Git for Windows mailing lists as well as via a tweet.
- Ditto for distribution packagers for Linux and other platforms: their releases are announced via their preferred channels.
- A mail to <oss-security@lists.openwall.org> (see below for details) is sent as a follow-up to the <distros@vs.openwall.org> one, describing the vulnerability in detail, often including a proof of concept of an exploit.

Note: The Git project makes no guarantees about timelines, but aims to keep embargoes reasonably short in the interest of keeping Git's users safe.

Opening a Security Advisory draft ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The first step is to open an advisory. Technically, this is not necessary. However, it is the most convenient way to obtain the CVE number and it gives us a private repository associated with it that can be used to collaborate on a fix.

Notifying the Linux distributions ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

At most two weeks before release date, we need to send a notification to <distros@vs.openwall.org>, preferably less than 7 days before the release date. This will reach most (all?) Linux distributions. See an example below, and the guidelines for this mailing list at here.

Once the version has been published, we send a note about that to oss-security. As an example, see the v2.24.1 mail; Here are their guidelines.

The mail to oss-security should also describe the exploit, and give credit to the reporter(s): security researchers still receive too little respect for the invaluable service they provide, and public credit goes a long way to keep them paid by their respective organizations.

Technically, describing any exploit can be delayed up to 7 days, but we usually refrain from doing that, including it right away.

As a courtesy we typically attach a Git bundle (as `.tar.xz` because the list will drop `.bundle` attachments) in the mail to distros@ so that the involved parties can take care of integrating/backporting them. This bundle is typically created using a command like this:

```
git bundle create cve-xxx.bundle ^origin/master vA.B.C vD.E.F
tar cJvf cve-xxx.bundle.tar.xz cve-xxx.bundle
```

Example mail to distros@vs.openwall.org
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
To: distros@vs.openwall.org
Cc: git-security@googlegroups.com, <other people involved in the report/fix>
Subject: [vs] Upcoming Git security fix release

Team,

The Git project will release new versions on <date> at 10am Pacific Time or
soon thereafter. I have attached a Git bundle (embedded in a `.tar.xz` to avoid
it being dropped) which you can fetch into a clone of
https://github.com/git/git via `git fetch --tags /path/to/cve-xxx.bundle`,
containing the tags for versions <versions>.

You can verify with `git tag -v <tag>` that the versions were signed by
the Git maintainer, using the same GPG key as e.g. v2.24.0.

Please use these tags to prepare `git` packages for your various
distributions, using the appropriate tagged versions. The added test cases
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
help verify the correctness.

The addressed issues are:

<list of CVEs with a short description, typically copy/pasted from Git's
release notes, usually demo exploit(s), too>

Credit for finding the vulnerability goes to <reporter>, credit for fixing
it goes to <developer>.

Thanks,
<name>
```

Example mail to oss-security@lists.openwall.com
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
To: oss-security@lists.openwall.com
Cc: git-security@googlegroups.com, <other people involved in the report/fix>
Subject: git: <copy from security advisory>

Team,

The Git project released new versions on <date>, addressing <CVE>.

All supported platforms are affected in one way or another, and all Git
versions all the way back to <version> are affected. The fixed versions are:
<versions>.

Link to the announcement: <link to lore.kernel.org/git>

We highly recommend to upgrade.

The addressed issues are:
* <list of CVEs and their explanations, along with demo exploits>

Credit for finding the vulnerability goes to <reporter>, credit for fixing
it goes to <developer>.

Thanks,
<name>
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

□□□ #2
□ □□□ □□□ 2026-10-31 20:14:44 UTC
□ □□□ □□□ 2026-10-31 20:14:45 UTC