**Question:**

Many websites expose their “.git” files, please show how it could be dangerous.

Answer:

**Step1- Detect .git exposure using forced browsing**

Once you have a solid list of Web applications, use forced browsing to see if a .git folder is accessible on them.

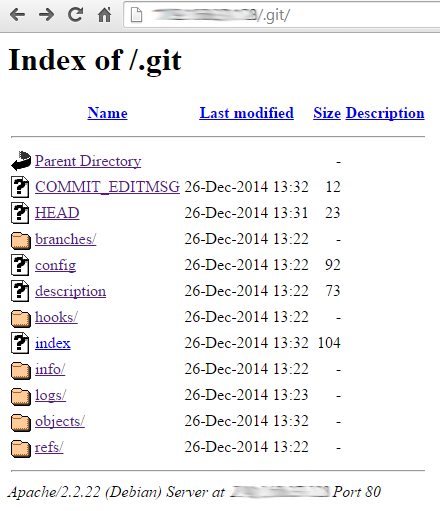
If file & directory bruteforce tools are allowed, you can use **dirsearch or dirb (with common.txt dictionary)**. They both check for .git/.

But if automated tools are not allowed (happens even on pentests!), simply **go to <web-app>/.git** (e.g. <https://example.com/.git> or <https://example.com/git/>) on a browser.

If you get a 404 error, then .git/ doesn’t exist on the server. But if you get a **403 forbidden error**, it does! The folder’s root just won’t be directly accessible if directory listing is disabled on the server:



If you’re lucky and directory listing is enabled, then you could directly browse the .git folder’s contents:

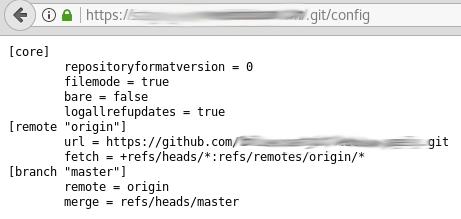


**Step2- Confirm the bug by manually browsing the .git folder**

If you “git clone” any Git project from Github and look at .git/ in its root you’ll notice that some file are always present: .git/config, .git/HEAD, .git/logs/HEAD, .git/index…

You can confirm that the .git folder’s contents are accessible (even if .git/ itself isn’t) by trying to open these different common file names, for example:

* https://example.com/.git/config
* https://example.com/.git/HEAD
* https://example.com/.git/logs/HEAD
* https://example.com/.git/index



**Step3-Automatically extract contents of .git**

This is the fun part! Browsing .git/ manually is good for proof of concept, but tedious. If you want to retrieve as many files as possible, even with directory listing disabled, the tool to use is [GitTools](https://github.com/internetwache/GitTools).

It’s really good! Just 4 lines and you’ll have all or parts of the remote Git repository on your computer:

./gitdumper.sh https://example.com/.git/ /output-directory/

git status # Returns that the files were deleted because folders are empty

git checkout -- . # To restore the files & download the directory

git log # See what other commits are there

Finally, you have to **analyze the local repository** manually. Try to detect other vulnerabilities using static code analysis, or credentials, authentication tokens, new endpoints, etc.

And don’t forget, if you find a vulnerable domain, to **check** its **development and staging subdomains** too. They would probably be vulnerable, even if the bug was fixed on the main domain/subdomain.

**Potential impact**

* Finding new vulnerabilities by analyzing the source code
* Finding files containing sensitive information like credentials, tokens, new endpoints, etc

**Examples of bug bounty reports**

* [Git repository found](https://hackerone.com/reports/248693) on Grabtaxi Holdings Pte Ltd ($1,000)
* [Git available containing passwords.](https://hackerone.com/reports/173811)on Boozt Fashion AB ($400)
* [[staging-engineering.gnip.com] Publicly accessible GIT directory](https://hackerone.com/reports/218465) on Twitter ($280)
* [GIT Detected](https://hackerone.com/reports/221298) on Nextcloud ($0)