

Sponsored by

 **INTIGRITI**

The fastest-growing bug bounty platform

[Click here to join the Intigriti community](#)

Source code disclosure via exposed .git folder

Posted in [Articles](#) on August 25, 2018



On this page 

Hi, I recently found a .git folder exposed on a public bug bounty program and used it to reconstruct the Web app’s source code. I can’t disclose specific details yet, but wanted to share with you this tutorial on how to find and exploit this kind of bugs.

.git exposure can pay well or not, depending on the assets found. But it is interesting anyway because:

1. It is very **easy to detect**
2. Analyzing the source code can **reveal other vulnerabilities** that are even more critical and interesting

The process is the same whether you have a list of targets during a penetration test, or a broad scope (like *.example.com) when bug hunting.

If the scope is limited to a set of Web applications, skip to step2. Otherwise, use recon to make a list of Web apps:

Make a list of Web apps

Start with **enumerating domains**. Jason Haddix’s [bug hunters methodology](#) is a very good start.

Check acquisitions in particular. Let’s say the program’s acquisition rules say that acquisitions are in scope only after 6 months. Then if you test a new acquisition at month 7, you may have more chances to find bugs than on a one or two-year old acquisition. The new one is probably less tested than the main domain too.

Then **enumerate subdomains** using tools like Amass, Sublist3r or Massdns.

After that, identify web applications, by port scanning all ips found and manually browsing all HTTP/HTTPS services. Note that the **same subdomain can host multiple Web applications**, for example on different ports or URL paths.

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:



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>



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](#).

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

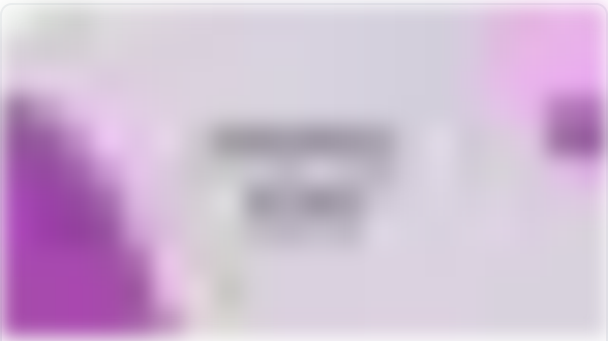
- [Git repository found](#) on Grabtaxi Holdings Pte Ltd (\$1,000)
- [Git available containing passwords.](#) on Boozt Fashion AB (\$400)
- [\[staging-engineering.gnip.com\] Publicly accessible GIT directory](#) on Twitter (\$280)
- [GIT Detected](#) on Nextcloud (\$0)

I hope this helps. If you have any questions, suggestions or requests, you know what to do!

See you next time!

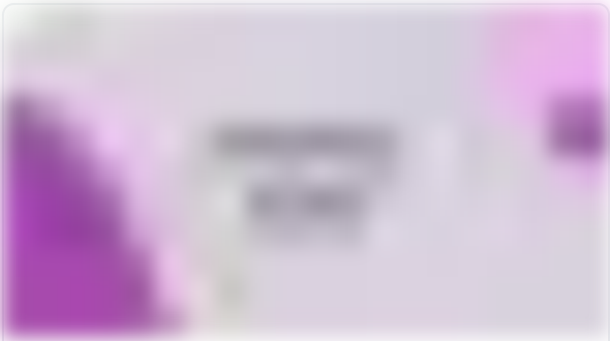
- Tutorials
- Web hacking

Related posts



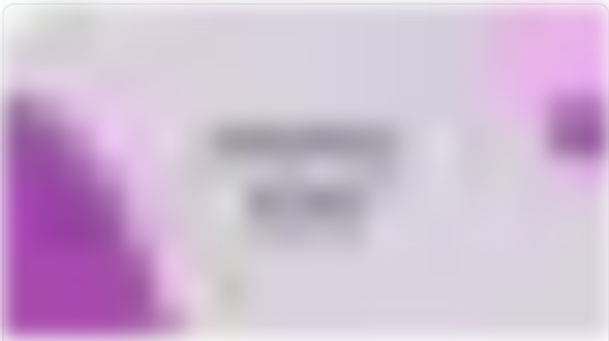
5 tips to make the most of Twitter as a pentester or bug bounty hunter

Posted in [Articles](#) on November 9, 2018



5 Kali Linux tricks that you may not know

Posted in [Articles](#) on November 9, 2018



5 things I wish I knew when I started as a junior penetration tester

Posted in [Articles](#) on October 31, 2018

