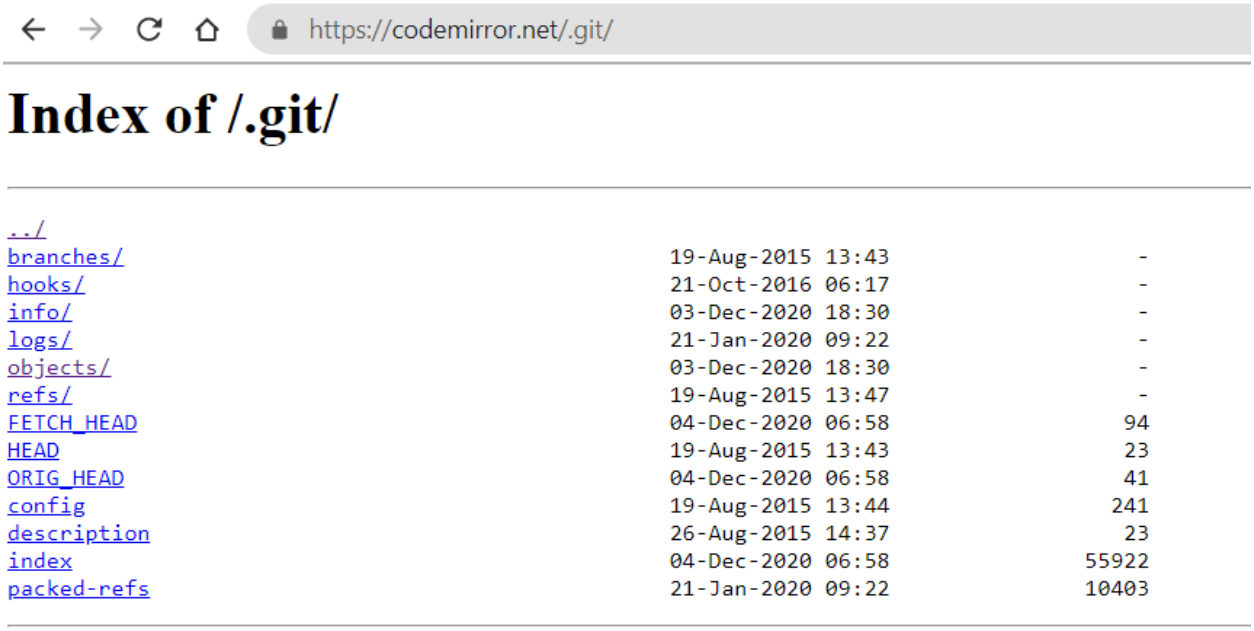
Q1: Many websites expose their ".git" files, please show how it could be dangerous.

By using Google Dork which filters results from Google, it allows other user to identify websites with publicly visible and accessible Git Repository. Like the screenshot of the website with a publicly visible Git Repository below.



Then I started downloading the files in the Git Repository using wget.

```
C:\Users\ahmo1>wget -r --no-check-certificate https://codemirror.net/.git/objects/
```

Then, I went to objects folder and randomly chose a folder and checked it.

There are many hash files and I could open them using "git cat-file -p" which provides type and size information.

Some files contain information such as login detail also SQL files that can be downloaded and so the database will be available.

It is also worth noticing that Trees, in git, are just data structures with references to other git objects. Thus, directories in git are represented as trees with references to blobs (files) and other trees.

In a real attack, this can be done via an automated script which makes it quiet faster and easier.

```
040000 tree 43dd98ca55b7ffabc65fa32def837528ac7d82a5    sparql
040000 tree 7d405231b37d06d751da0012bba5c6f5e0656868    spreadsheet
040000 tree 0f3467044521e7f1d6ba31696b8ea602b94a4eb0    sql
040000 tree 2c574d748fbb4872a4770000c3a4a9ffdc968383    stex
040000 tree f39584d7715f8224d7643787df976df8ff542529    stylus
```

```
100644 blob 05cafbe50d5f3a55a6d2bc64366cbb4d42e7c4e7    index.html
100644 blob 4127cd9a0523b1a286a1ba04896b008b3484ed0e    sql.js
```

Q2: Imagine that we have 248 text files. Explain how can we find which files are the same**

To find similar documents in very large amount of document sets, one of the best way is using hash table for instance SHA256 or using locally sensitive hashing (LSH). The idea of LSH is to hash items several times, so similar items are more likely to be hashed to the same bucket than dissimilar items are. Then we can consider any pair that hashed to the same bucket for any of the hashing to be a candidate pair. Only these candidate pairs need to be checked for similarity.

Q3: Write a hello-world C program and explain how we can dump its binary code with radare2

The Hello World C code is shown below:

```c
#include <stdio.h>
int main()
{
   printf("Hello World");
}
```

Nest step using gcc command to compile the code.

```
C:\Users\ahmo1>gcc -o c.exe hello.c
hello.c:3:1: warning: return type defaults to 'int' [-Wimplicit-int]
    3 | main()
      | ^~~~
```

Then run radare2.exe {filename.exe} in the terminal, and using V command we can see how the program works.

```
;-- _main:
0x00401410      55              push ebp
0x00401411      89e5            mov ebp, esp
0x00401413      83e4f0          and esp, 0xfffffff0
0x00401416      83ec10          sub esp, 0x10
0x00401419      e872050000      call sym.___main            ;[2]
0x0040141e      c70424445040.   mov dword [esp], str.Hello_World
0x00401425      e8ba290000      call sym._printf            ;[3]
0x0040142a      b800000000      mov eax, 0
0x0040142f      c9              leave
0x00401430      c3              ret
```