OZON tech

Защита API от IDOR

Как найти проблемы в коде с semgrep правилами в join mode,
в динамике с nuclei шаблонами из QA трафика,
и перестать уже беспокоиться с auth-директивами в graphql

Дмитрий Марюшкин

Руководитель продуктовой безопасности
Ozon Fintech

Whoami

Дмитрий Марюшкин

Руководитель продуктовой безопасности в Ozon Fintech

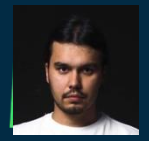
- Делаю платежные сервисы безопасней
- Развиваю направление data-аналитики в security
- Пишу внутренние appsec инструменты,
  немного open source (OWASP), статьи в][

До этого все тоже самое в Yandex, Avito, PT

dmarushkin

Так же в докладе незримо присутствуют:

Григорий Прохоров
semgrep ниндзя
grishxnder



Андрей Кан
nuclei джедай
kandrewps

Про что сегодня?

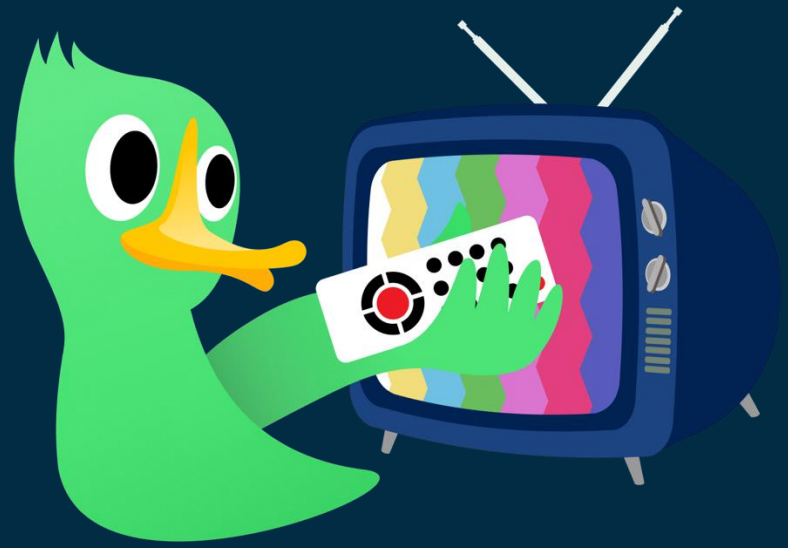
00. Что за IDOR-ы?

01. Ищем в коде с semgrep

02. Ищем в динамике с nuclei и QA

03. Пишем безопасно по default-у в graphql

* 04. Мониторим эксплуатацию IDOR



Про что сегодня?

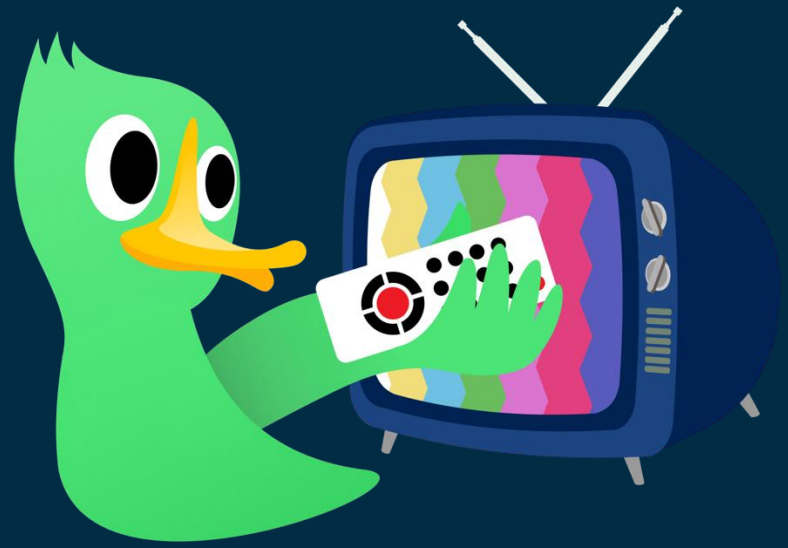
00. Что за IDOR-ы?

01. Ищем в коде с semgrep

02. Ищем в динамике с nuclei и QA

03. Пишем безопасно по default-у в graphql

* 04. Мониторим эксплуатацию IDOR



Про что сегодня?

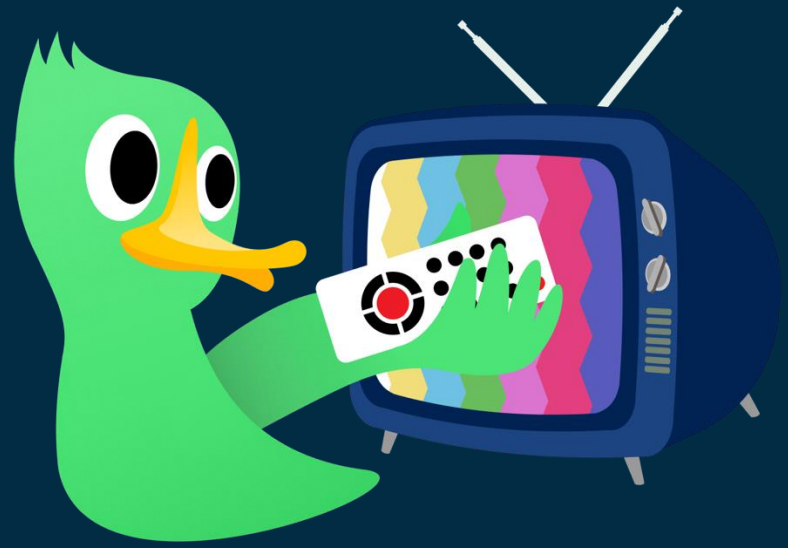
00. Что за IDOR-ы?

01. Ищем в коде с semgrep

02. Ищем в динамике с nuclei и QA

03. Пишем безопасно по default-у в graphql

* 04. Мониторим эксплуатацию IDOR



Про что сегодня?

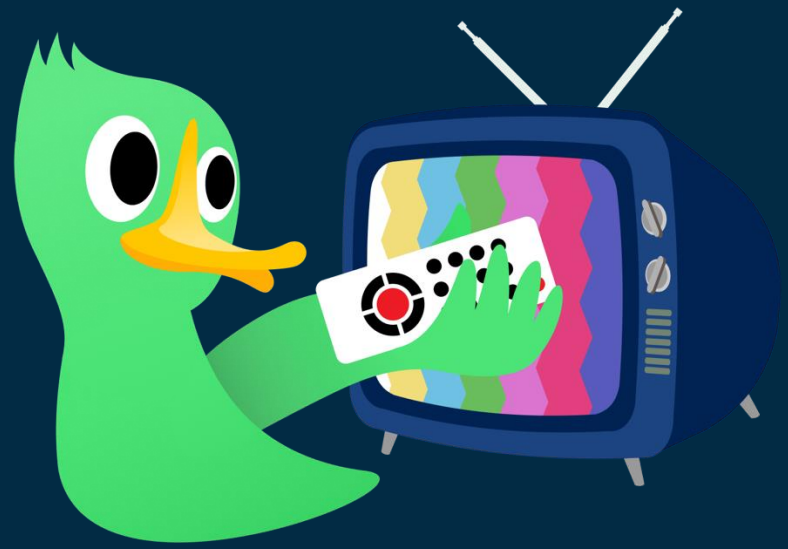
00. Что за IDOR-ы?

01. Ищем в коде с semgrep

02. Ищем в динамике с nuclei и QA

03. Пишем безопасно по default-у с graphql

* 04. Мониторим эксплуатацию IDOR



Про что сегодня?

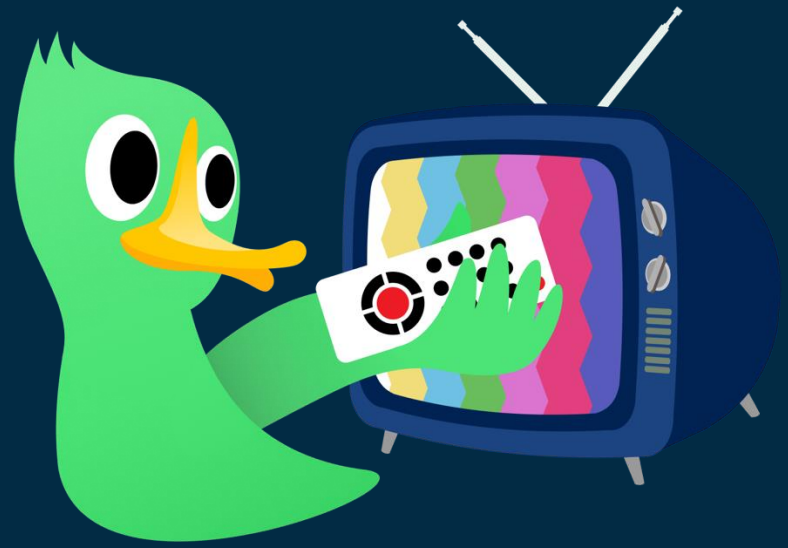
00. Что за IDOR-ы?

01. Ищем в коде с semgrep

02. Ищем в динамике с nuclei и QA

03. Пишем безопасно по default-у в graphql

*** 04. Мониторим эксплуатацию IDOR**



00

Что за IDOR-ы и где они обитают?

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Client 1

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Client 1

PUT /createOrder -d { orderData: private} => HTTP/2 200 OK { orderId: **1337**, orderData: private}

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Client 1

PUT /createOrder -d { orderData: private} => HTTP/2 200 OK { orderId: **1337**, orderData: private}

GET /getOrder/**1337** => HTTP/2 200 OK { orderId: **1337**, orderData: private}

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Client 1

PUT /createOrder -d { orderData: private}	=>	HTTP/2 200 OK { orderId: **1337**, orderData: private}
GET /getOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, orderData: private}
DELETE /delOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, result: deleted}

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Client 1

PUT /createOrder -d { orderData: private}	=>	HTTP/2 200 OK { orderId: **1337**, orderData: private}
GET /getOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, orderData: private}
DELETE /delOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, result: deleted}

Client 2

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Client 1

PUT /createOrder -d { orderData: private} => HTTP/2 200 OK { orderId: **1337**, orderData: private}

GET /getOrder/**1337** => HTTP/2 200 OK { orderId: **1337**, orderData: private}

DELETE /delOrder/**1337** => HTTP/2 200 OK { orderId: **1337**, result: deleted}

Client 2

GET /getOrder/**1337** => HTTP/2 200 OK { orderId: **1337**, orderData: private}

Что за IDOR-ы?

TLDR: Кто-то забыл проверить принадлежность объекта текущему авторизованному пользователю при обращении к этому объекту по ID.

Client 1

PUT /createOrder -d { orderData: private}	=>	HTTP/2 200 OK { orderId: **1337**, orderData: private}
GET /getOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, orderData: private}
DELETE /delOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, result: deleted}

Client 2

GET /getOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, orderData: private}
DELETE /delOrder/**1337**	=>	HTTP/2 200 OK { orderId: **1337**, result: deleted}

Но у меня же везде UUID!

Но у меня же везде UUID!

Действительно подобрать ec4c2806-64c6-**4**38b-**9**53b-d0b63dd26225 выглядит сложно.

Но у меня же везде UUID!

Действительно подобрать ec4c2806-64c6-**4**38b-**9**53b-d0b63dd26225 выглядит сложно.

Но у меня же везде UUID!

Действительно подобрать ec4c2806-64c6-**4**38b-**9**53b-d0b63dd26225 выглядит сложно.

Но, можно эти id случайно не уберечь:

Но у меня же везде UUID!

Действительно подобрать ec4c2806-64c6-**4**38b-**9**53b-d0b63dd26225 выглядит сложно.

Но, можно эти id случайно не уберечь:

1. Бизнес сценарии (сложный id в обмен на ИНН или телефон)

Но у меня же везде UUID!

Действительно подобрать ec4c2806-64c6-**4**38b-**9**53b-d0b63dd26225 выглядит сложно.

Но, можно эти id случайно не уберечь:

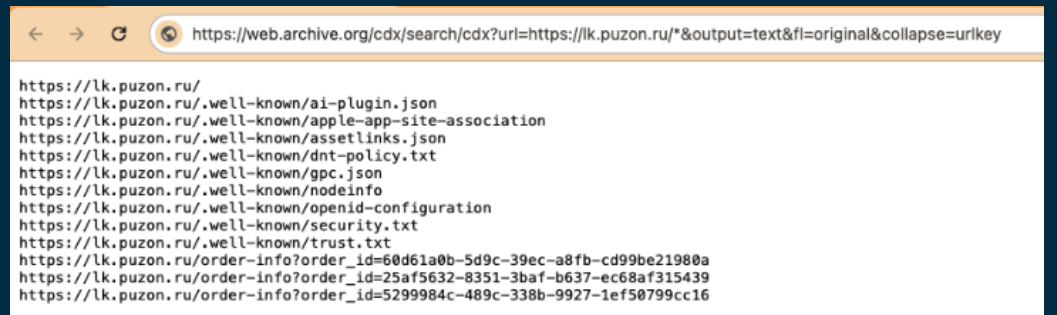
1. Бизнес сценарии (сложный id в обмен на ИНН или телефон)
2. **Внутренние логи сервисов и админки отдающие id объектов**

Но у меня же везде UUID!

Действительно подобрать ec4c2806-64c6-438b-953b-d0b63dd26225 выглядит сложно.

Но, можно эти id случайно не уберечь:

1. Бизнес сценарии (сложный id в обмен на ИНН или телефон)
2. Внутренние логи сервисов и админки отдающие id объектов
3. **Веб архив**



Но у меня же везде UUID!

Только проверка авторизации пользователя на доступ к объекту – единственная надежная защита от IDOR!

Ок, как найти?

Ок, как найти?

В коде:

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте
- Найти все способы в коде вынуть ID юзера, компании из авторизации

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте
- Найти все способы в коде вынуть ID юзера, компании из авторизации
- Выбрать ручки с ID на входе, где не вспоминают про User и Company ID

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте
- Найти все способы в коде вынуть ID юзера, компании из авторизации
- Выбрать ручки с ID на входе, где не вспоминают про User и Company ID

В динамике еще проще:

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте
- Найти все способы в коде вынуть ID юзера, компании из авторизации
- Выбрать ручки с ID на входе, где не вспоминают про User и Company ID

В динамике еще проще:

- Создать двух пользователей

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте
- Найти все способы в коде вынуть ID юзера, компании из авторизации
- Выбрать ручки с ID на входе, где не вспоминают про User и Company ID

В динамике еще проще:

- Создать двух пользователей
- Потрогать через burp все ручки под одним юзером

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте
- Найти все способы в коде вынуть ID юзера, компании из авторизации
- Выбрать ручки с ID на входе, где не вспоминают про User и Company ID

В динамике еще проще:

- Создать двух пользователей
- Потрогать через burp все ручки под одним юзером
- Пойти в repeater и потрогать интересные ручки под вторым юзером

Ок, как найти?

В коде:

- Найти все ручки с идентификаторами объектов в инпуте
- Найти все способы в коде вынуть ID юзера, компании из авторизации
- Выбрать ручки с ID на входе, где не вспоминают про User и Company ID

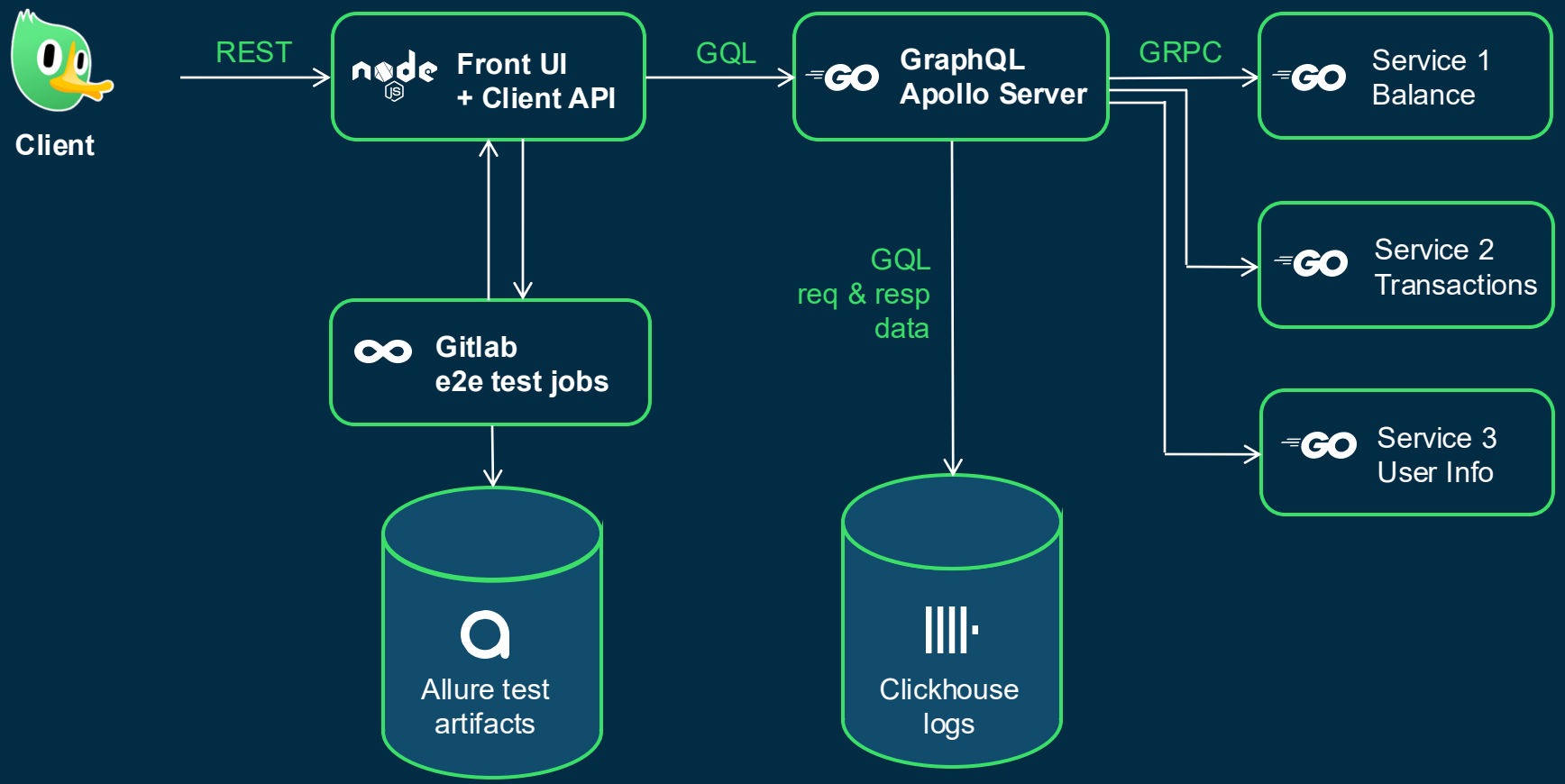
В динамике еще проще:

- Создать двух пользователей
- Потрогать через burp все ручки под одним юзером
- Пойти в repeater и потрогать интересные ручки под вторым юзером

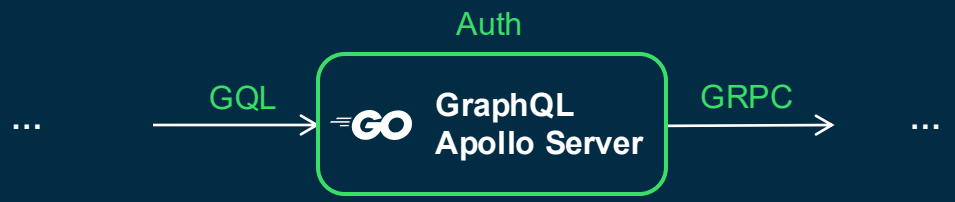
01

Ищем в коде с semgrep правилами в join mode

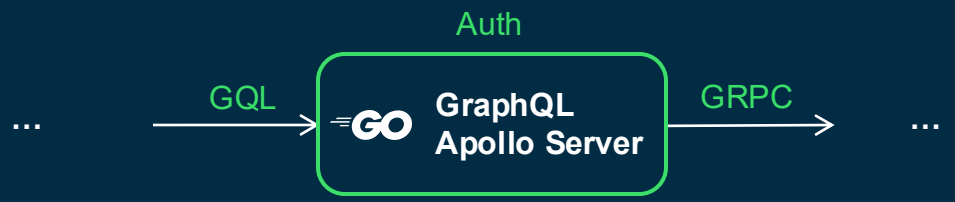
Ландшафт


Client
REST
Front UI
+ Client API
GQL
GraphQL
Apollo Server
GRPC
Service 1
Balance
Service 2
Transactions
Service 3
User Info
Gitlab
e2e test jobs
GQL
req & resp
data
Allure test
artifacts
Clickhouse
logs

Graphql server

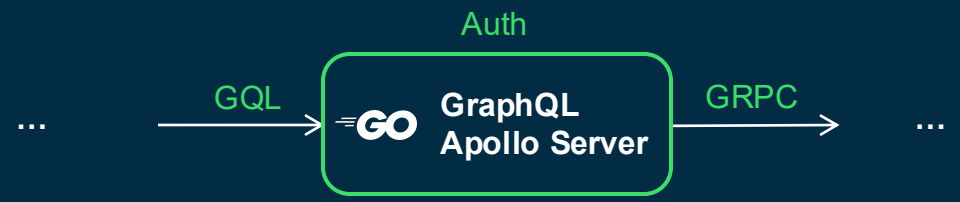


Graphql server




```
type CertificatesMutations {
    """ Запрос справки о доступном остатке """
    generateAvailableBalance(input: CertificateGenerateAvailableBalanceInput!): CertificateGenerateAvailableB
    """ Запрос справки о наличии счёта """
    generateBankAccount(input: CertificateGenerateBankAccountInput!): CertificateGenerateBankAccountPayload!
    """ Запрос справки о реквизитах счёта """
    generateAccountDetails(input: CertificateGenerateAccountDetailsInput!): CertificateGenerateAccountDetails
    """ Запрос справки о наличии карты """
```

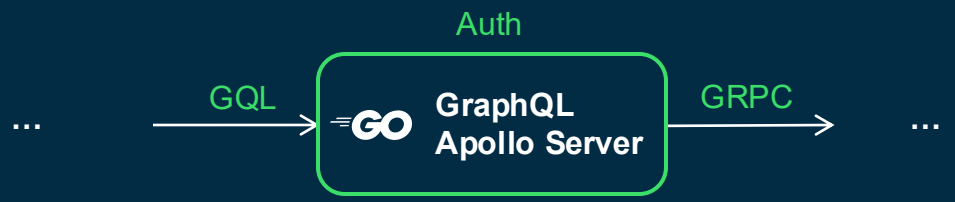
Graphql server




```
type CertificatesMutations {
    """ Запрос справки о доступном остатке """
    generateAvailableBalance(input: CertificateGenerateAvailableBalanceInput!): CertificateGenerateAvailableB
    """ Запрос справки о наличии счёта """
    generateBankAccount(input: CertificateGenerateBankAccountInput!): CertificateGenerateBankAccountPayload!
    """ Запрос справки о реквизитах счёта """
    generateAccountDetails(input: CertificateGener
    """ Запрос справки о наличии карты """
```

```
""" Входные данные для запроса генерации справки о наличии счёта """
input CertificateGenerateBankAccountInput {
    """ Ключ идемпотентности """
    idempotencyKey: String!
    """ Номер счёта """
    accountNumber: String
    """ Токен номера счёта """
    accountNumberToken: String
    """ Язык справки """
    language: CertificateLanguage!
    """ Email, на который отправить справку """
    email: String
}
```

Graphql server




```
type CertificatesMutations {
    """ Запрос справки о доступном остатке """
    generateAvailableBalance(input: CertificateGenerateAvailableBalanceInput!): CertificateGenerateAvailableB
    """ Запрос справки о наличии счёта """
    generateBankAccount(input: CertificateGenerateBankAccountInput!): CertificateGenerateBankAccountPayload!
    """ Запрос справки о реквизитах счёта """
    generateAccountDetails(input: CertificateGener
    """ Запрос справки о наличии карты """
```

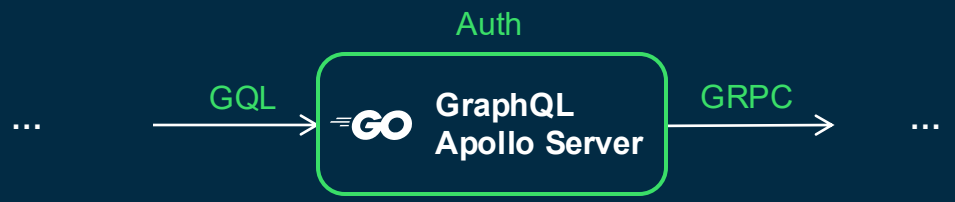
```
""" Входные данные для запроса генерации справки о наличии счёта """
input CertificateGenerateBankAccountInput {
    """ Ключ идемпотентности """
    idempotencyKey: String!
    """ Номер счёта """
    accountNumber: String
    """ Токен номера счёта """
    accountNumberToken: String
    """ Язык справки """
    language: CertificateLanguage!
    """ Email, на который отправить
    email: String
}
```

```
// GenerateAccountDetails is the resolver for the generateAccountDetails field.
func (r *certificatesMutationsResolver) GenerateAccountDetails(ctx context.Context, obj *model.CertificatesMutations, input model.CertificateGenerateAccountDetailsInput)
    session := auth.GetSession(ctx)
    if session.ClientId == `` {
        return nil, model.ErrorNoBankAuth
    }

    certificate, err := r.quillHelper.GenerateAccountDetailsCertificate(ctx, session.ClientId, &input)
    if err != nil {
        return nil, err
    }

    return &model.CertificateGenerateAccountDetailsPayload{
        Certificate: certificate,
    }, nil
}
```

Graphql server




```
type CertificatesMutations {
    """ Запрос справки о доступном остатке """
    generateAvailableBalance(input: CertificateGenerateAvailableBalanceInput!): CertificateGenerateAvailableB
    """ Запрос справки о наличии счёта """
    generateBankAccount(input: CertificateGenerateBankAccountInput!): CertificateGenerateBankAccountPayload!
    """ Запрос справки о реквизитах счёта """
    generateAccountDetails(input: CertificateGener
    """ Запрос справки о наличии карты """
```

```
""" Входные данные для запроса генерации справки о наличии счёта """
input CertificateGenerateBankAccountInput {
    """ Ключ идемпотентности """
    idempotencyKey: String!
    """ Номер счёта """
    accountNumber: String
    """ Токен номера счёта """
    accountNumberToken: String
    """ Язык справки """
    language: CertificateLanguage!
    """ Email, на который отправить
    email: String
}
```

```
// GenerateAccountDetails is the resolver for the generateAccountDetails field.
func (r *certificatesMutationsResolver) GenerateAccountDetails(ctx context.Context, obj *model.CertificatesMutations, input model.CertificateGenerateAccountDetailsInput)
    session := auth.GetSession(ctx)
    if session.ClientId == `` {
        return nil, model.ErrorNoBankAuth
    }

    certificate, err := r.quillHelper.GenerateAccountDetailsCertificate(ctx, session.ClientId, &input)
    if err != nil {
        return nil, err
    }

    return &model.CertificateGenerateAccountDetailsPayload{
        Certificate: certificate,
    }, nil
}
```

Auth!!!

Заряжаем semgrep

Начнем с резолверов без авторизации:

structure NEW | advanced

```yaml
rules:
  - id: molly-resolvers-without-auth
    languages:
      - go
    severity: INFO
    message: $FUNCNAME
    patterns:
      - pattern: >
          func ($SOME0 *$SOME1) $FUNCNAME(ctx context.Context, ..., $INPUTNAME
          model.$INPUT, ...) (...) {
            ...
          }
      - pattern-not: >
          func ($SOME0 *$SOME1) $FUNCNAME(ctx context.Context, ..., $INPUTNAME
          model.$INPUT, ...) (...) {
            ...
            $SOME2 := auth.GetSession(ctx, ...)
            ...
          }
      - metavariable-regex:
          metavariable: $INPUTNAME
          regex: (filter|input)
      - metavariable-regex:
          metavariable: $SOME1
          regex: (?i:.*resolver.*)
    paths:
      exclude:
        - "*_test.go"
      include:
        - "*.go"

```

test code | live code NEW | metadata | docs | Pro Turbo

```go
// GenerateBankAccount is the resolver for the generateBankAccount field.
func (r *certificatesMutationsResolver) GenerateBankAccount(ctx context.Context,
obj *model.CertificatesMutations, input model.CertificateGenerateBankAccountInput)
(*model.CertificateGenerateBankAccountPayload, error) {
    session := auth.GetSession(ctx)
    if session.ClientId == `` {
        return nil, model.ErrorNoBankAuth
    }

    certificate, err := r.quillHelper.GenerateBankAccountCertificate(ctx, session.
    ClientId, &input)
    if err != nil {
        return nil, err
    }

    return &model.CertificateGenerateBankAccountPayload{
        Certificate: certificate,
    }, nil
}

```

Заряжаем semgrep

Начнем с резолверов без авторизации:

structure NEW | advanced

```yaml
rules:
  - id: molly-resolvers-without-auth
    languages:
      - go
    severity: INFO
    message: $FUNCNAME
    patterns:
      - pattern: >
          func ($SOME0 *$SOME1) $FUNCNAME(ctx context.Context, ..., $INPUTNAME
          model.$INPUT, ...) (...) {
            ...
          }
      - pattern-not: >
          func ($SOME0 *$SOME1) $FUNCNAME(ctx context.Context, ..., $INPUTNAME
          model.$INPUT, ...) (...) {
            ...
            $SOME2 := auth.GetSession(ctx, ...)
            ...
          }
      - metavariable-regex:
          metavariable: $INPUTNAME
          regex: (filter|input)
      - metavariable-regex:
          metavariable: $SOME1
          regex: (?i:.*resolver.*)
    paths:
      exclude:
        - "*_test.go"
      include:
        - "*.go"

```

test code | live code NEW | metadata | docs | Pro Turbo

```go
// GenerateBankAccount is the resolver for the generateBankAccount field.
func (r *certificatesMutationsResolver) GenerateBankAccount(ctx context.Context, obj *model.CertificatesMutations, input model.CertificateGenerateBankAccountInput) (*model.CertificateGenerateBankAccountPayload, error) {

    certificate, err := r.quillHelper.GenerateBankAccountCertificate(ctx, session.ClientId, &input)
    if err != nil {
        return nil, err
    }

    return &model.CertificateGenerateBankAccountPayload{
        Certificate: certificate,
    }, nil
}

```

Заряжаем semgrep

Дальше найдем gql инпуты и фильтры с ID объектов на входе:

structure NEW | advanced

```yaml
rules:
  - id: graphql-filter
    languages:
      - generic
    severity: ERROR
    message: >
      $TYPE: $GQLTYPENAME -> $GQLFIELDNAME -> $GQLFIELDVALUE -> $IDORDIRECTIVE
    patterns:
      - pattern-either:
          - pattern-inside: |-
              $TYPE $GQLTYPENAME {
                ...
              }

      - patterns:
          - pattern-either:
              - patterns:
                  - pattern-either:
                      - pattern-regex: '(?P<GQLFIELDNAME>([a-zA-Z0-9!\[\]]+))\:\s*(?P<GQLFIELDVALUE>([a-zA-Z0-9!\[\]]+))\s+(?P<IDORDIRECTIVE>(\@\w+\s*)+)$'
                      - pattern-regex: '(?P<GQLFIELDNAME>([a-zA-Z0-9!\[\]]+))\s*\((?P<FIELDINPUT2>.*)\)\:\s*(?P<GQLFIELDVALUE>([a-zA-Z0-9!\[\]]+))\s+(?P<IDORDIRECTIVE>(\@\w+\s*)+)$'

                      - pattern-regex: '(?P<GQLFIELDNAME>([a-zA-Z0-9!\[\]]+))\:\s*(?P<GQLFIELDVALUE>([a-zA-Z0-9!\[\]]+))$'
                      - pattern-regex: '(?P<GQLFIELDNAME>([a-zA-Z0-9!\[\]]+))\s*\((?P<FIELDINPUT2>.*)\)\:\s*(?P<GQLFIELDVALUE>([a-zA-Z0-9!\[\]]+))$'
          - pattern-not-inside: (...)
          - metavariable-regex:
              metavariable: $GQLFIELDNAME
              regex: (?i:^.*(id|account).*$)
    paths:
      include:
        - "*.graphql"
```

test code | live code NEW | metadata | docs | Pro Turbo

```
"""Входные данные для запроса генерации справки о наличии карты """
input CertificateGenerateCardAvailabilityInput {
    """ Ключ идемпотентности """
    idempotencyKey: String!
    """ Идентификатор справки """
    cardID: String!
    """ Язык справки """
    language: CertificateLanguage!
    """ Email, на который отправить справку """
    email: String
}


"""Входные данные для запроса генерации справки о проведении карточной операции """
input CertificateGenerateOperationConductingInput {
    """ Ключ идемпотентности """
    idempotencyKey: String!
    """ Идентификатор операции """
    operationID: String!
    """ Язык справки """
    language: CertificateLanguage!
    """ Email, на который отправить справку """
    email: String
}
```

Run ⌘↵

Matches

Line 5
input: CertificateGenerateCardAvailabilityInput → idempotencyKey → String! → $IDORDIRECTIVE

Line 7
input: CertificateGenerateCardAvailabilityInput → cardID → String! → $IDORDIRECTIVE

Line 18
input: CertificateGenerateOperationConductingInput → idempotencyKey → String! → $IDORDIRECTIVE

Line 20
input: CertificateGenerateOperationConductingInput → operationID → String! → $IDORDIRECTIVE

Line 32
input: CertificateGenerateDepositTariffInput → idempotencyKey → String! → $IDORDIRECTIVE

Заряжаем semgrep

И наконец пересекаем два отдельных правила в новом:

gql_id_input_resolver_noauth.yaml 614 B

Blame | Edit | Lock | Replace | Delete

```yaml
rules:
- id: harvest-not-recursive-search
  mode: join
  join:
    refs:
        - rule: /rules/rules_cache/bozon-additional/gql_id_resolver_noauth/gql_filter_schema_input.yaml
          as: get-fields-with-id-input
        - rule: /rules/rules_cache/bozon-additional/gql_id_resolver_noauth/resolvers_without_auth.yaml
          as: resolver-without-auth
    on:
        - 'get-fields-with-id-input.$GQLTYPENAME idor resolver-without-auth.$FUNCNAME'
  message: |
    IDOR in 'resolver-without-auth.$FUNCNAME'. Please check that this function does not have an authorization code and make the RISK.
  severity: ERROR
```

Заряжаем semgrep

И наконец пересекаем два отдельных правила в новом:

gql_id_input_resolver_noauth.yaml 614 B

Blame | Edit | Lock | Replace | Delete

```yaml
rules:
- id: harvest-not-recursive-search
  mode: join
  join:
    refs:
        - rule: /rul
          as: get-fi
        - rule: /rul
          as: resolv
    on:
        - 'get-field
  message: |
    IDOR in 'resolve
  severity: ERROR
```

IDOR in 'Resolver-Without-Auth.$FUNCNAME'. Please Check That This Function Does Not Have an Authorization Code and Make the RISK. audit

ID	Severity	Status	Original	Duplicate Cluster	Type	Date discovered	Age	Reporter
400914	High	Inactive, Duplicate			Static	March 11, 2025	96 days	(SAST_scheduler_service)

Location
internal/app/graph/pinCode.mutation.resolvers.go

Duplicate Cluster (107)

Import History (1)

Description

Result message: IDOR in 'resolver-without-auth.$FUNCNAME'. Please check that this function does not have an authorization code and make the RISK.

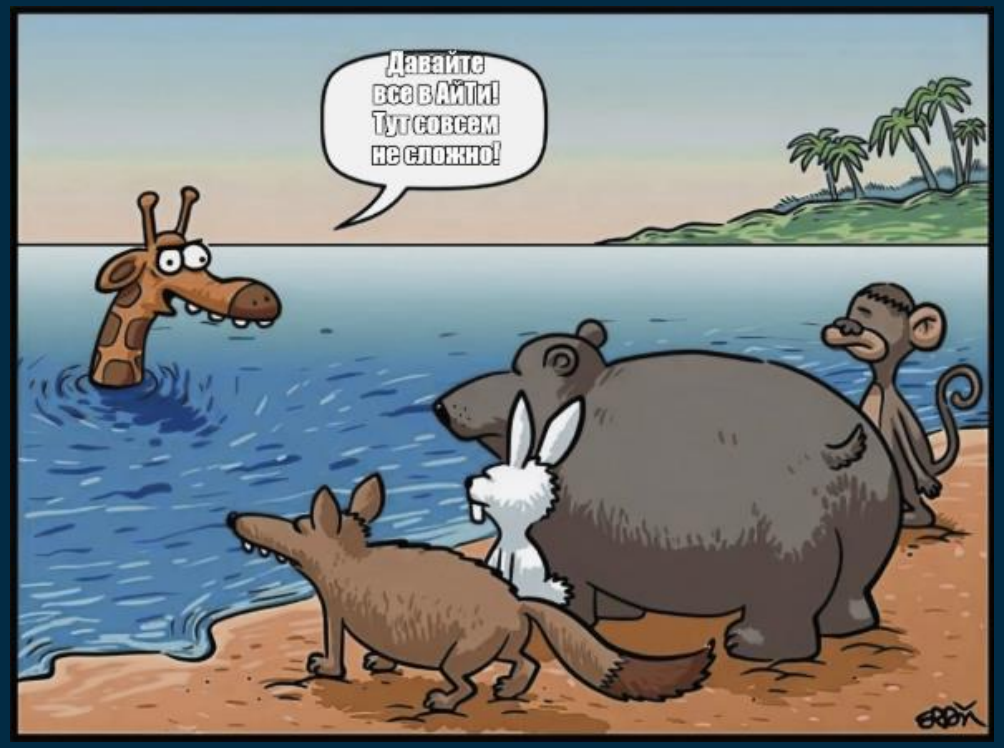
Snippet:

```
func (r *pinCodeMutationsResolver) IncomeAndCodeWordStepRequest(ctx context.Context, obj *model.PinCodeMutations, input model.IncomeAndCodeWordStepInput)
    step, err := r.recoveryService.IncomeAndCodeWordStep(ctx, input)
    if err != nil {
        return nil, err
    }
```

ozon tech

А если у нас не graphql?

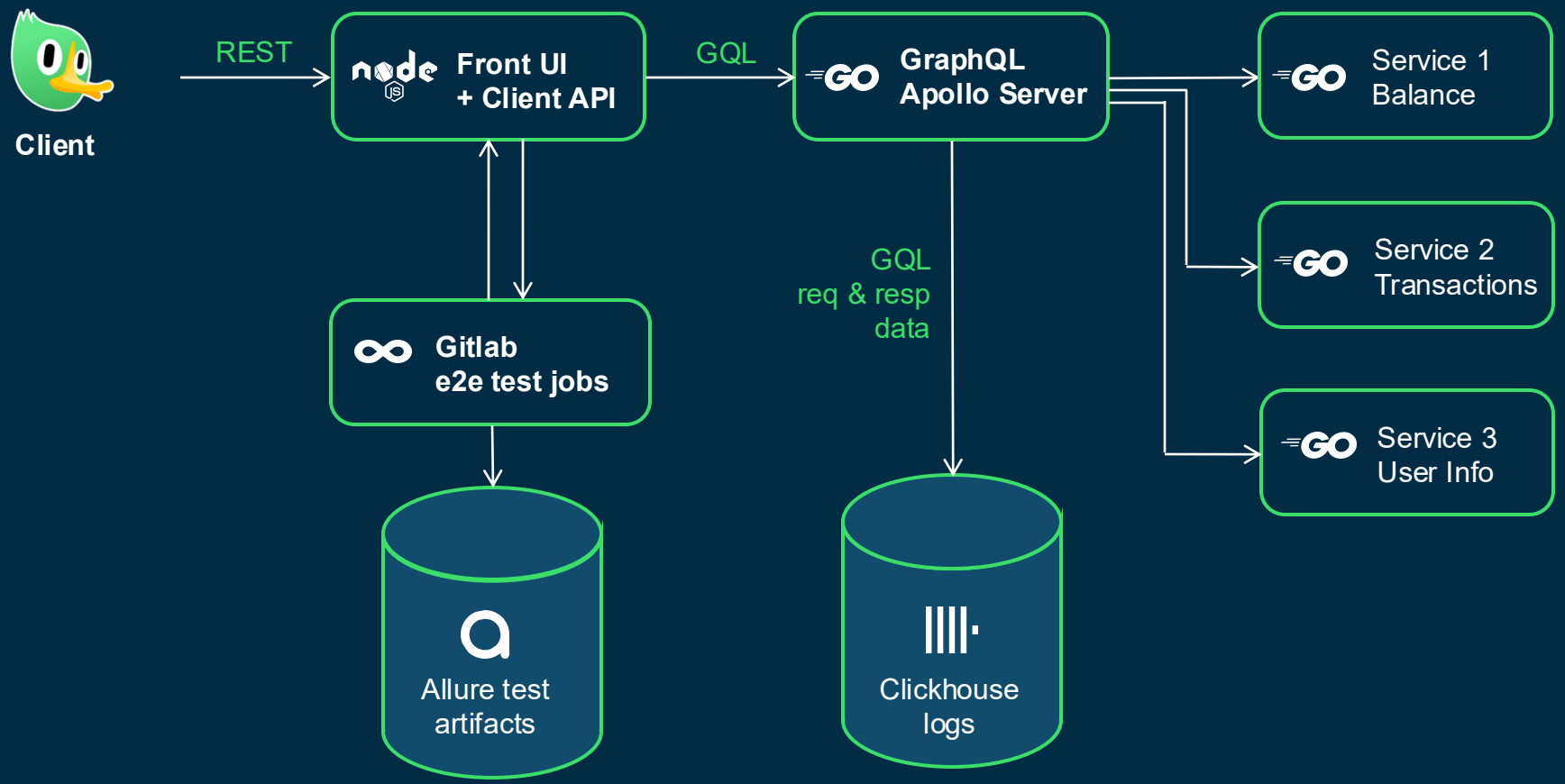
Та же история с тем, чтоб найти по имени датаклассы в python и java, структуры в go и пересечь их с инпутами в ручках



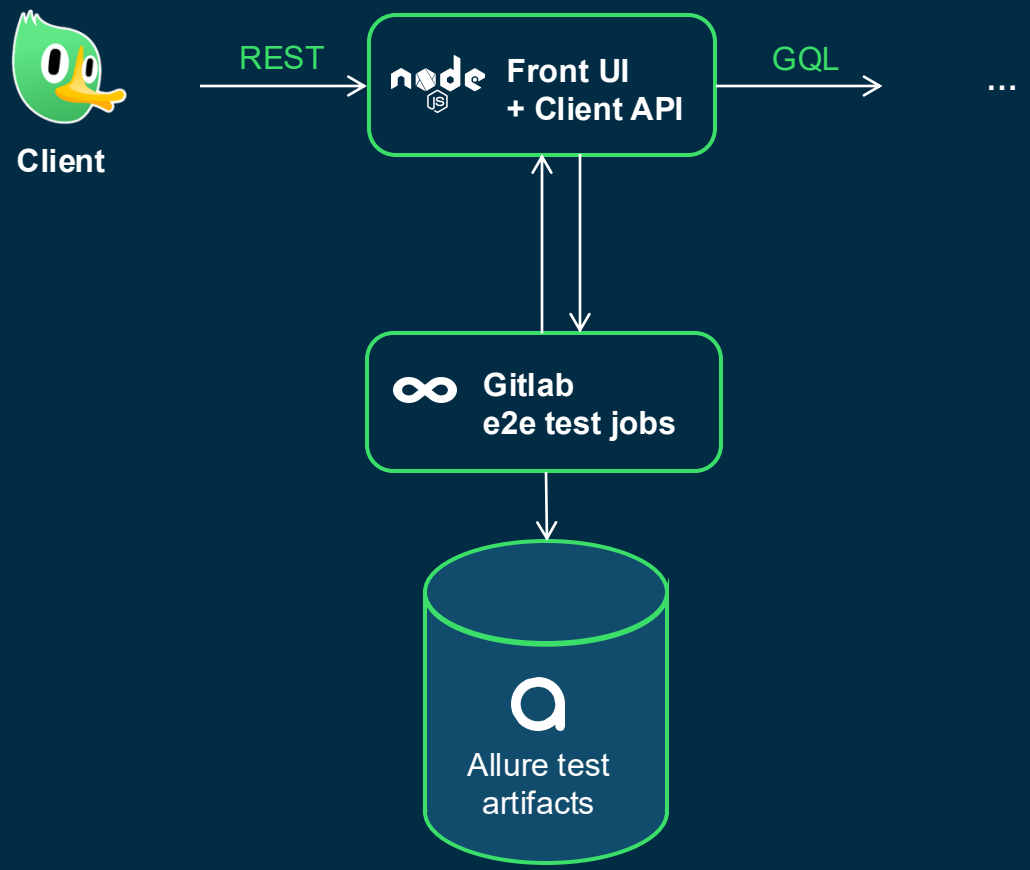
02

Ищем в динамике с nuclei шаблонами из QA трафика

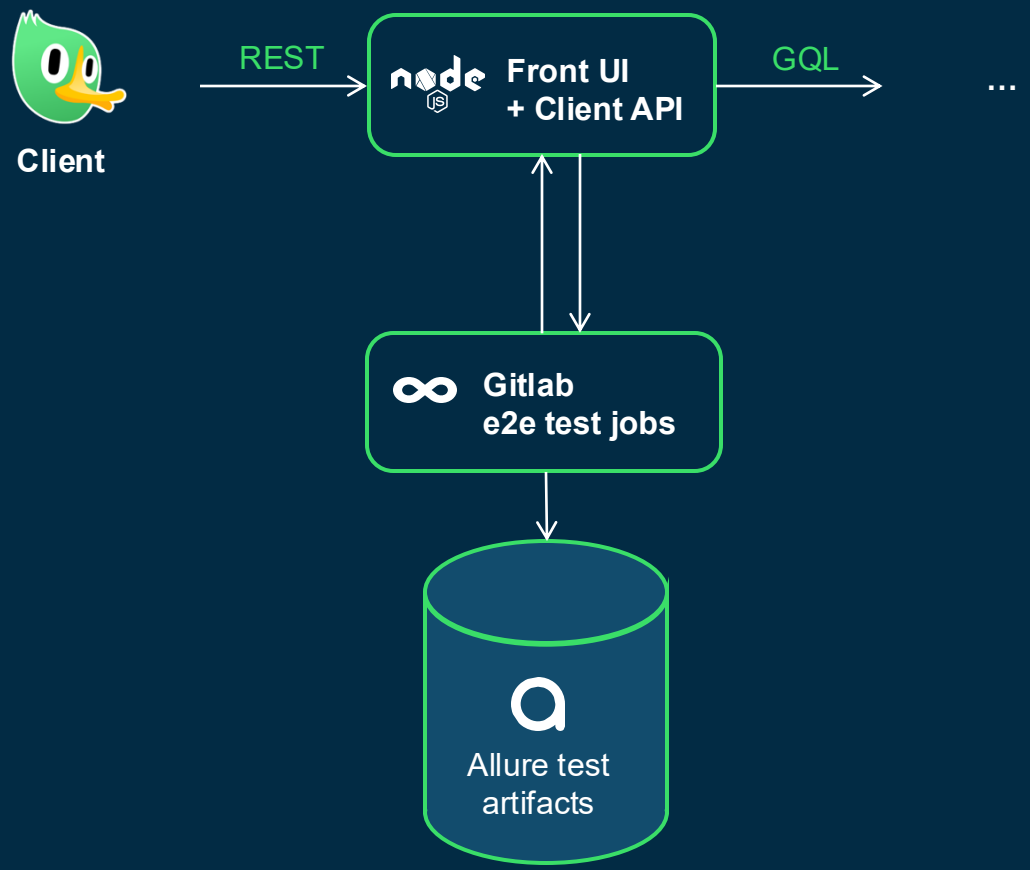
Ландшафт


Client
REST
Front UI
+ Client API
GQL
GraphQL
Apollo Server
Service 1
Balance
Service 2
Transactions
Service 3
User Info
Gitlab
e2e test jobs
GQL
req & resp
data
Allure test
artifacts
Clickhouse
logs

QA e2e тесты

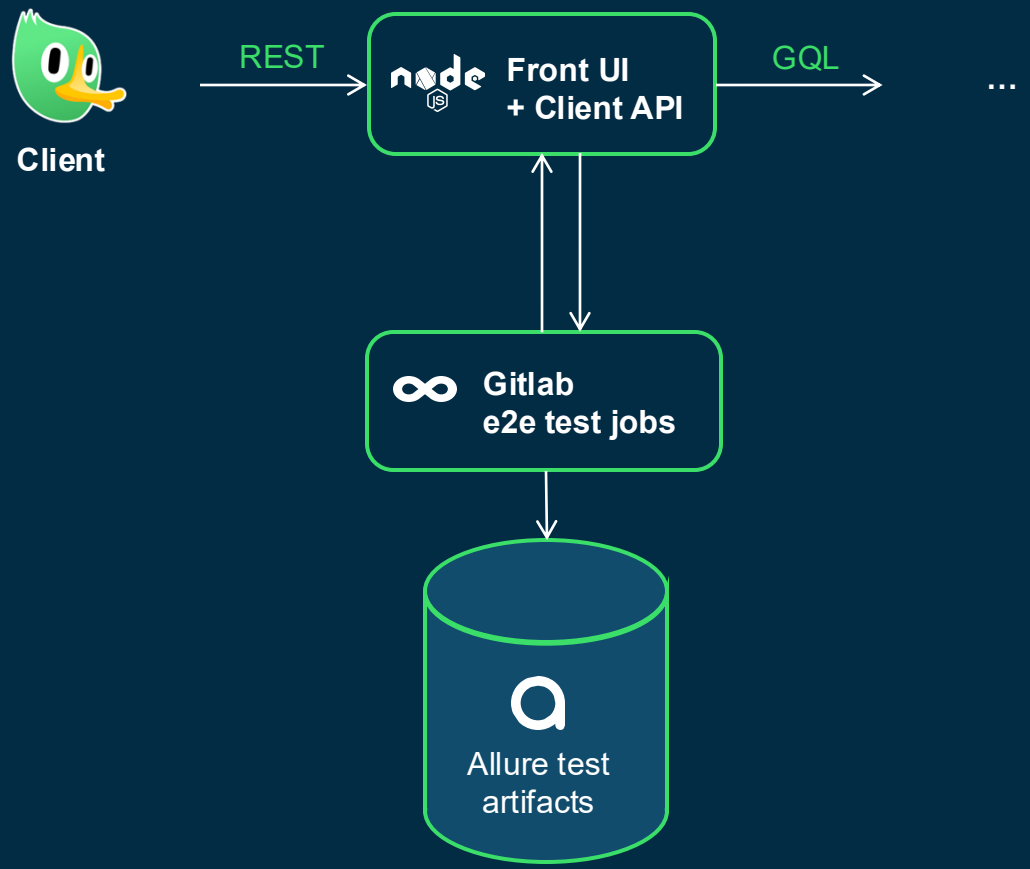

Client
REST
Front UI
+ Client API
GQL
...
Gitlab
e2e test jobs
Allure test
artifacts

QA e2e тесты



Среднестатистические e2e тесты:

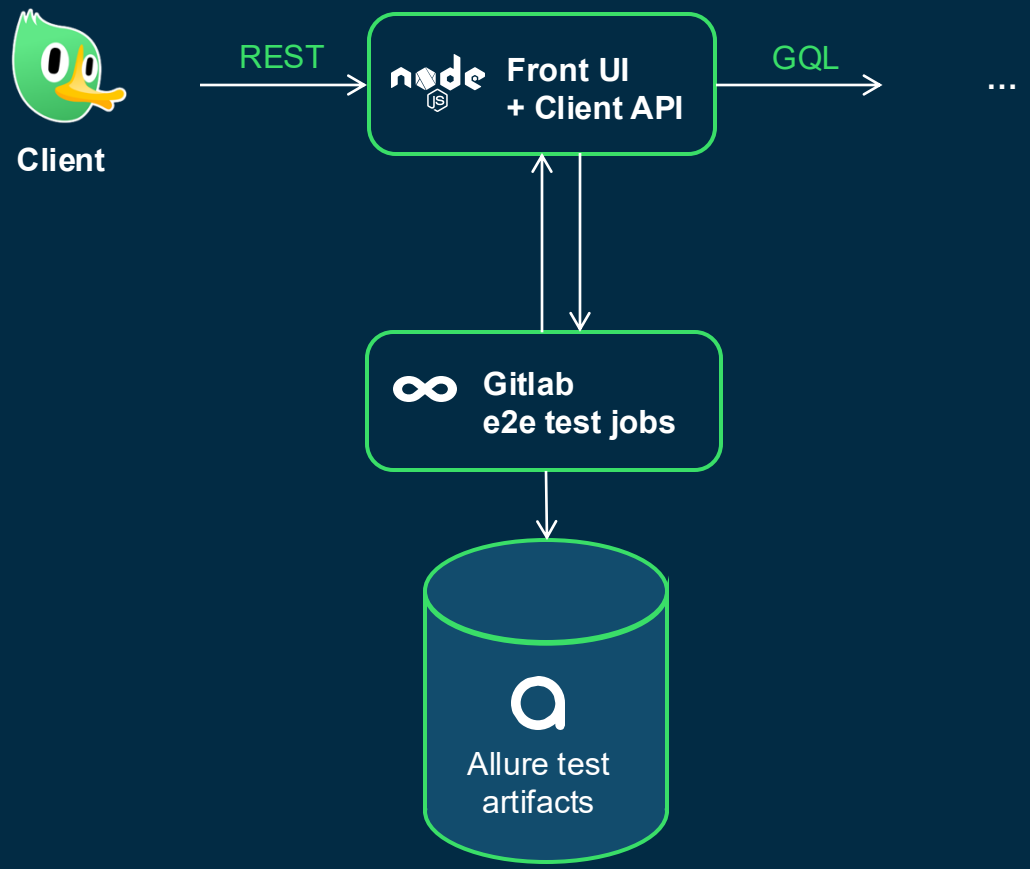
QA e2e тесты



Среднестатистические e2e тесты:

1. Создать пользователя с нужными свойствами

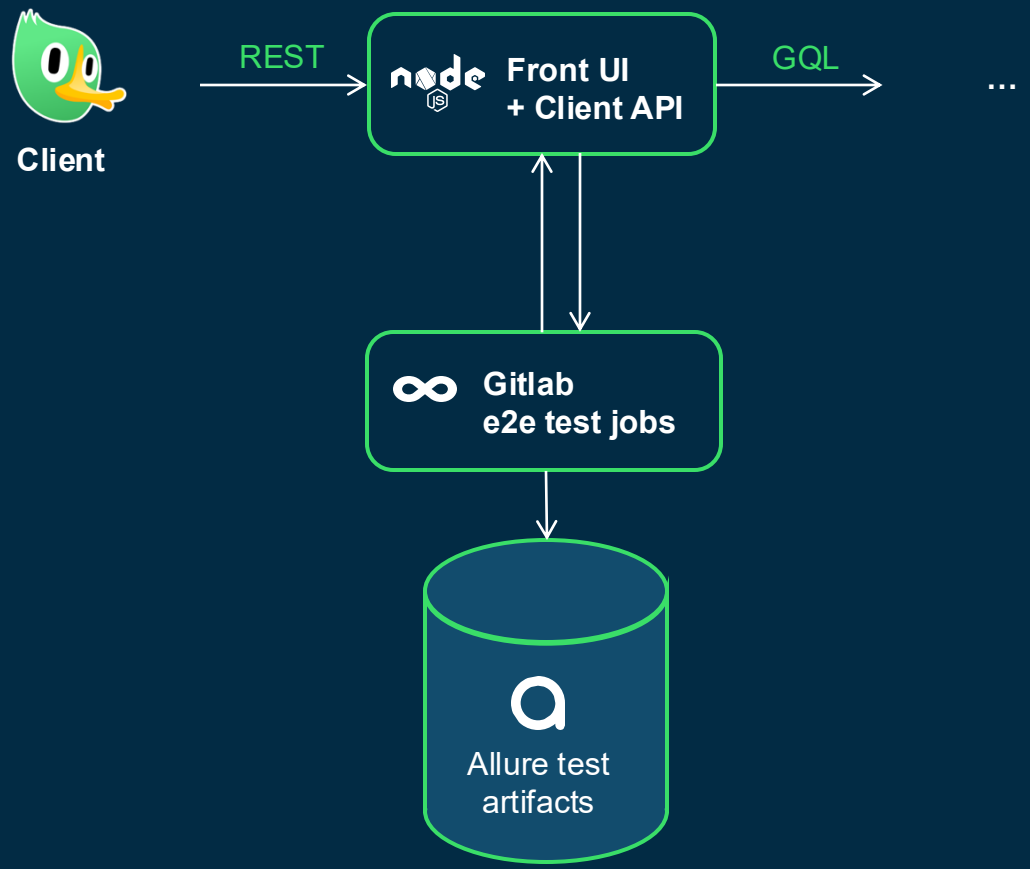
QA e2e тесты



Среднестатистические e2e тесты:

1. Создать пользователя с нужными свойствами
2. Выполнить бизнес сценарий

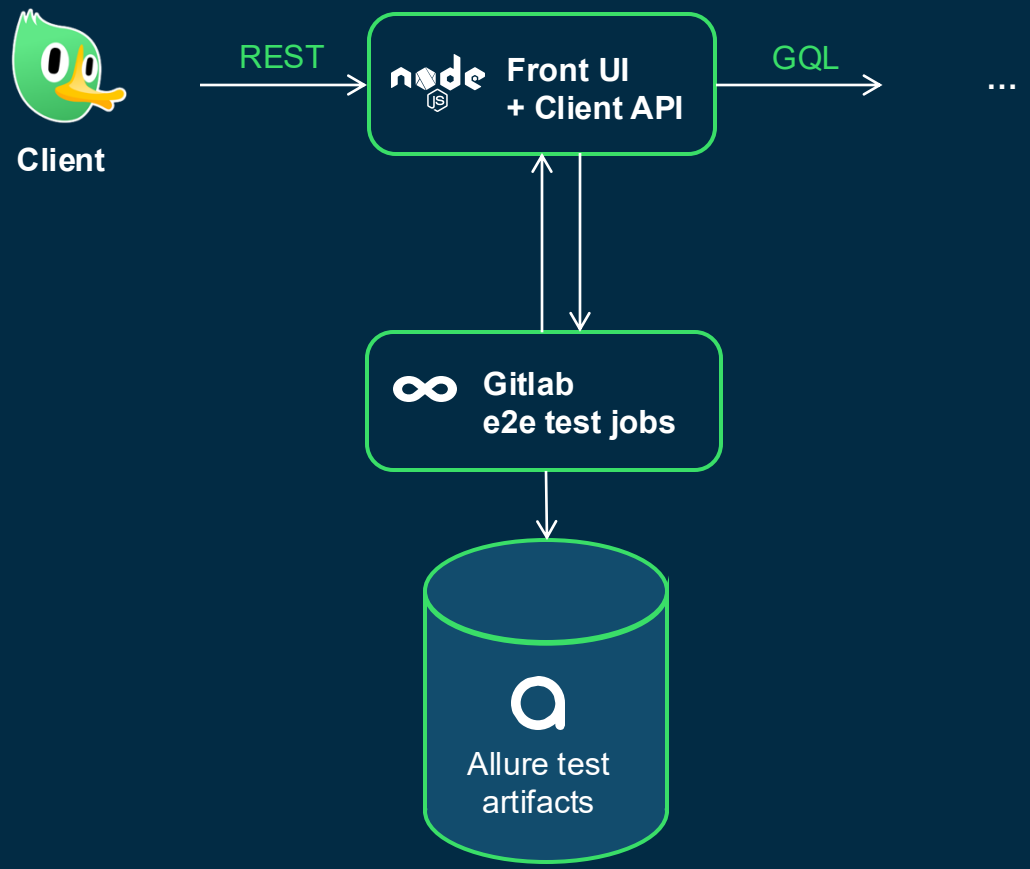
QA e2e тесты



Среднестатистические e2e тесты:

1. Создать пользователя с нужными свойствами
2. Выполнить бизнес сценарий
   - положить товар в корзину

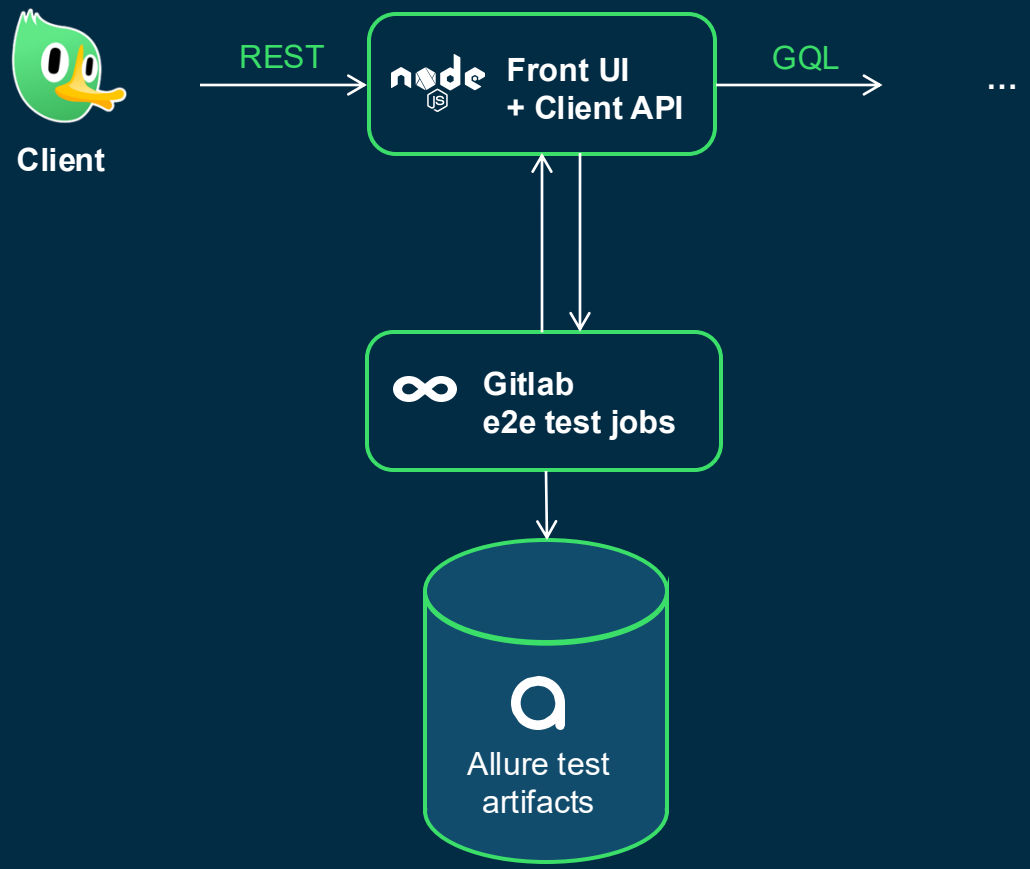
QA e2e тесты



Среднестатистические e2e тесты:

1. Создать пользователя с нужными свойствами
2. Выполнить бизнес сценарий
   - положить товар в корзину
   - оплатить заказ

QA e2e тесты

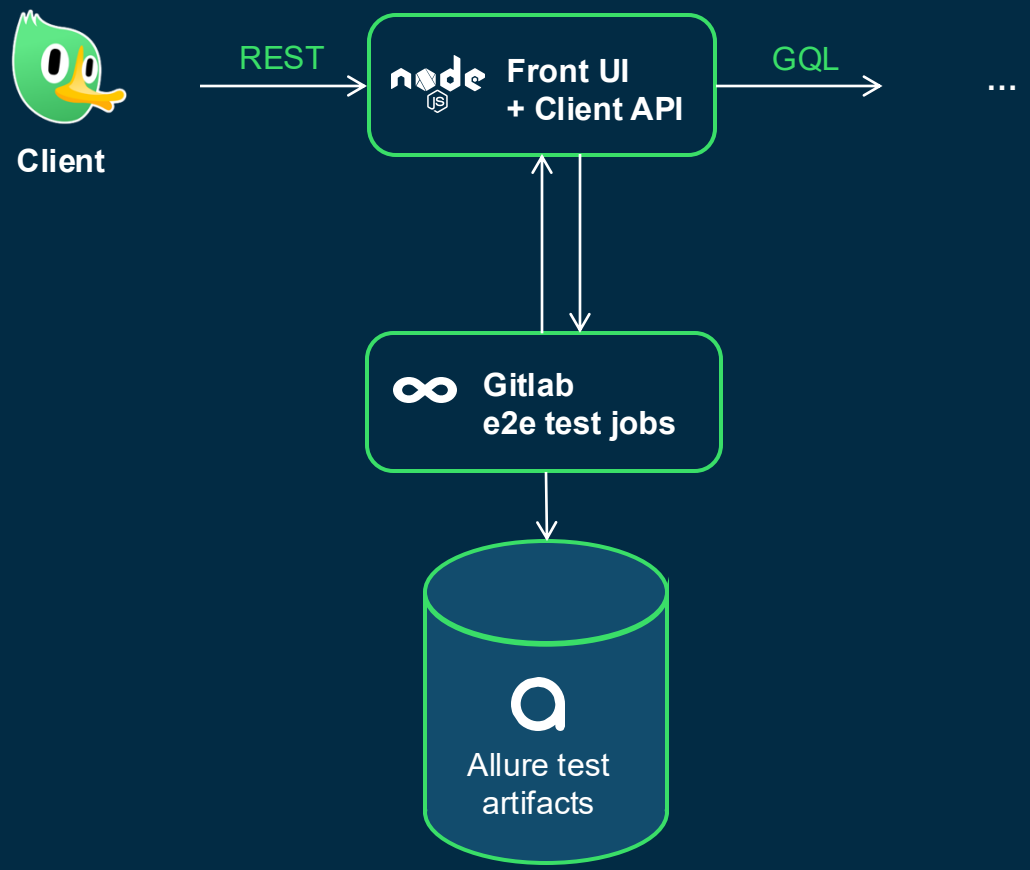


Среднестатистические e2e тесты:

1. Создать пользователя с нужными свойствами
2. Выполнить бизнес сценарий
   - положить товар в корзину
   - оплатить заказ
   - проверить уменьшение баланса
   - проверить заказ в нужном статусе

   ....

QA e2e тесты

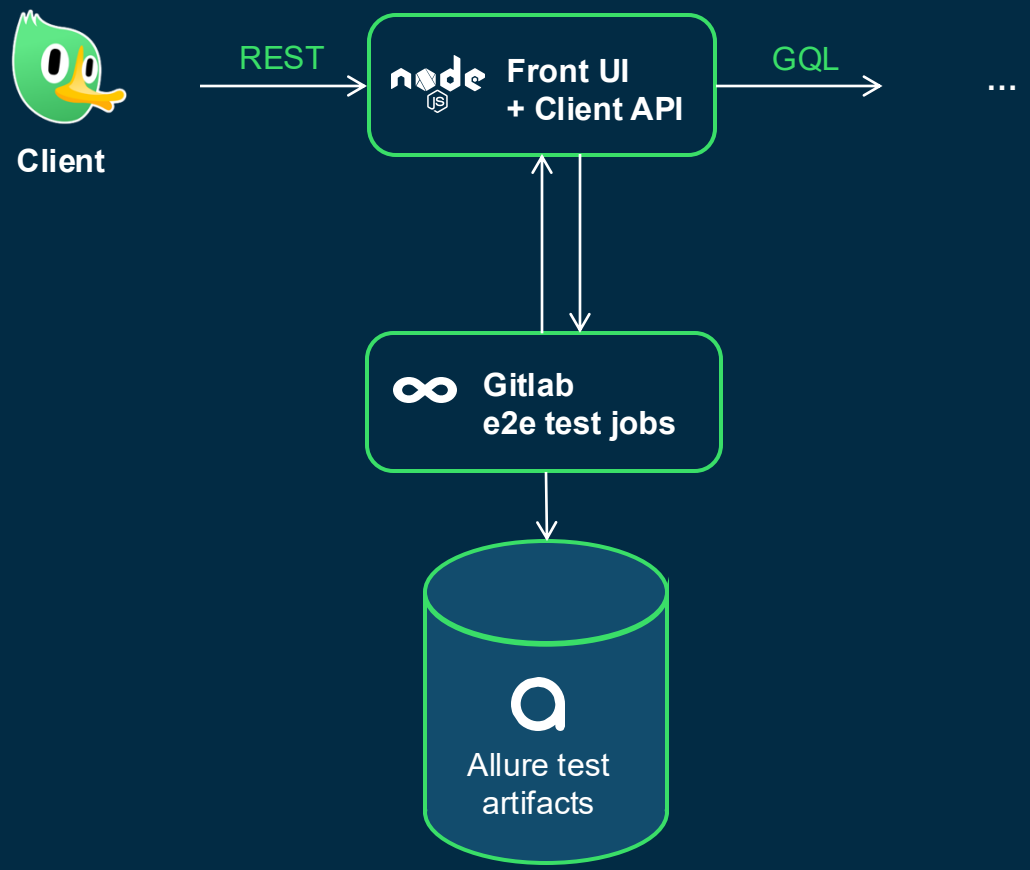


Среднестатистические e2e тесты:

1. Создать пользователя с нужными свойствами
2. Выполнить бизнес сценарий
   - положить товар в корзину
   - оплатить заказ
   - проверить уменьшение баланса
   - проверить заказ в нужном статусе

   ....
3. Сохранить артефакты (json запроса и ответа)

QA e2e тесты



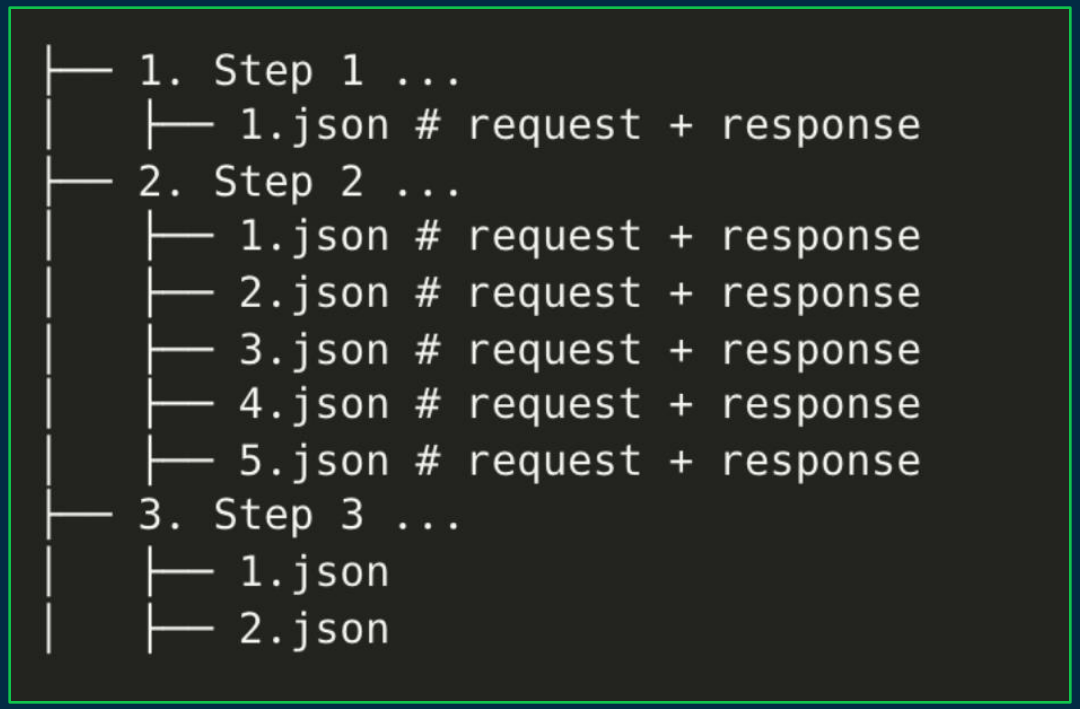
Среднестатистические e2e тесты:

1. Создать пользователя с нужными свойствами
2. Выполнить бизнес сценарий
   - положить товар в корзину
   - оплатить заказ
   - проверить уменьшение баланса
   - проверить заказ в нужном статусе

   ....
3. Сохранить артефакты (json запроса и ответа)
4. Вычистить пользователей и из объекты

QA e2e тесты

Можем ли мы на базе артефактов собрать сценарий с проверкой IDOR?



1. Создать таких же пользователей x2

2. Собрать позитивный nuclei шаблон
   - поймать все значимые параметры и пронести их из ответов первых запросов в инпуты последующих

3. Собрать шаблон с IDOR чеками

QA e2e тесты + nuclei

Пробежавшись по json-ам мы можем испечь сценарий, в котором меняющийся из прогона в прогон параметр, но повторяющийся в последующих запросах того же прогона будет извлечен nuclei экскрактором и подставлен в следующие запросы:

```
{
    "order":[
        "id":"882771",
        "price": "1200",
        "status": "true",
        "mark": 1,
        "throttled": "false",
        "items": 12,
        "owner": "Andrei"
        ]
}
```

```
{
    "order":[
        "id":"322184",
        "price": "1200",
        "status": "true",
        "mark": 1,
        "throttled": "false",
        "items": 12,
        "owner": "Andrei"
        ]
}
```

QA e2e тесты + nuclei

Пробежавшись по json-ам мы можем испечь сценарий, в котором меняющийся из прогона в прогон параметр, но повторяющийся в последующих запросах того же прогона будет извлечен nuclei экскрактором и подставлен в следующие запросы:

QA e2e тесты + nuclei

Так же можно прокинуть во все запросы авторизацию:

```
http:
  - method: POST
    path:
      - "http://craftuser.com/craftUser"
    extractors:
      - type: json
        internal: true
        part: body
        name: token
        json:
          - ".user.token"
    body:
      '{"name":"Pod Pishis", "email":"@
```

```
- method: GET
  path:
    - "http://primer.com/createOrder"
  headers:
    Content-Type: "application/json"
    Cookie: token={{token}}
  extractors:
    - type: json
      internal: true
      part: body
      name: orderID
      json:
        - ".order[0]"

- method: GET
  headers:
    Content-Type: "application/json"
    Cookie: token={{token}}
  path:
    - "http://primer.com/getOrder?orderID={{orderID}}"
```

QA e2e тесты + nuclei

Так же можно прокинуть во все запросы авторизацию:

```
http:
  - method: POST
    path:
      - "http://craftuser.com/craftUser"
    extractors:
      - type: json
        internal: true
        part: body
        name: token
        json:
          - ".user.token"
    body:
      '{"name":"Pod Pishis", "email":"@
```

```
- method: GET
  path:
    - "http://primer.com/createOrder"
  headers:
    Content-Type: "application/json"
    Cookie: token={{token}}
  extractors:
    - type: json
      internal: true
      part: body
      name: orderID
      json:
        - ".order[0]"

- method: GET
  headers:
    Content-Type: "application/json"
    Cookie: token={{token}}
  path:
    - "http://primer.com/getOrder?orderID={{orderID}}"
```

QA e2e тесты + nuclei

И в итоге получится что-то такое:

```
http:
  - method: POST
    path:
      - "http://craftuser.com/craftUser"
    extractors:
      - type: json
        internal: true
        part: body
        name: token
        json:
          - ".token"
    body:
      '{"name":"Pod Pishis", "email":"@hoockerman"}'

  - method: GET
    path:
      - "http://primer.com/createOrder"
    headers:
      Content-Type: "application/json"
      Cookie: token={{token}}
    extractors:
      - type: json
        internal: true
        part: body
        name: orderID
        json:
          - ".order.orderID"
```

```
  - method: GET
    headers:
      Content-Type: "application/json"
      Cookie: token={{token}}
    path:
      - "http://primer.com/getOrder?orderID={{orderID}}"
    extractors:
      - type: json
        internal: true
        part: body
        name: amount
        json:
          - ".order.amount"

  - method: POST
    headers:
      Content-Type: "application/json"
      Cookie: token={{token}}
    path:
      - "http://primer.com/cancelOrder"
    body:
      '{
        "orderID":"{{orderID}}",
        "codeWord": "subscribeTwitch",
        "amountToRefund": "{{refund}}",
        "comment": "please subscribe tg"
      }'
```

QA e2e тесты + nuclei

Остается добавить в нужных местах шаблона второго пользователя перед первым и фэйлить тест, если для него прошел валидный ответ в соответствующем шаге.

03

Безопасно по default-у с auth-директивами в graphql

Root cause анализ

Почему разработчик забывает делать авторизацию в резолверах?

```go
// GenerateAccountDetails is the resolver for the generateAccountDetails field.
func (r *certificatesMutationsResolver) GenerateAccountDetails(ctx context.Context, obj *model.CertificatesMutations, input model.CertificateGenerateAccountDetailsInput)
    session := auth.GetSession(ctx)
    if session.ClientId == `` {
        return nil, model.ErrorNoBankAuth
    }

    certificate, err := r.quillHelper.GenerateAccountDetailsCertificate(ctx, session.ClientId, &input)
    if err != nil {
        return nil, err
    }

    return &model.CertificateGenerateAccountDetailsPayload{
        Certificate: certificate,
    }, nil
}
```

Root cause анализ

Почему разработчик забывает делать авторизацию в резолверах? Потому что может!

```go
// GenerateAccountDetails is the resolver for the generateAccountDetails field.
func (r *certificatesMutationsResolver) GenerateAccountDetails(ctx context.Context, obj *model.CertificatesMutations, input model.CertificateGenerateAccountDetailsInput)
    session := auth.GetSession(ctx)
    if session.ClientId == `` {
        return nil, model.ErrorNoBankAuth
    }

    certificate, err := r.quillHelper.GenerateAccountDetailsCertificate(ctx, session.ClientId, &input)
    if err != nil {
        return nil, err
    }

    return &model.CertificateGenerateAccountDetailsPayload{
        Certificate: certificate,
    }, nil
}
```

Что делать?

Нужно срочно запретить ошибаться! Но как?

1. ~~Проверять все аппсеками!~~
2. Сделать с правильной авторизацией - дешево
3. И сделать без авторизации - максимально дорого

Директивы в graphql

В gql можно дополнительно вешать различные директивы на квери и мутации.

RateLimit на сессию пользователя:

```
type CertificatesMutations {
    """ Запрос справки о доступном остатке """
    generateAvailableBalance(input: CertificateGenerateAvailableBalanceInput!): CertificateGenerateAvailableBalancePayload!
    @sessionRateLimit(sessionFrom: SESSION, limit: 10, duration: 60)
```

Проверка что пользователь авторизован и тд:

```
6  type RkoAccountMutations{
7      """ Переотправить otp """
8      resendOtp(input: ResendOtpRkoAccount!): ShortIssue! @IsAuthenticatedInOzon
9      """ Подтвердить ОТП """
10     confirmOpt(input: ConfirmOtpRkoAccount!): ShortIssue! @IsAuthenticatedInOzon
```

```
schema := generated.NewExecutableSchema(generated.Config{
        Resolvers: resolver,
        Directives: generated.DirectiveRoot{
                SkipRkoCheck: func(ctx context.Context, obj interface{}, next graphql.Resolver) (res interface{}, err erro
                        return next(ctx)
                },
                SkipAuthenticate: func(ctx context.Context, obj interface{}, next graphql.Resolver) (res interface{}, err
                        return next(ctx)
                },
                IsAuthenticated: auth.Auth,
                IsAuthenticatedInOzon: func(ctx context.Context, obj interface{}, next graphql.Resolver) (res interface{},
                        ozonID, err := resolver.AuthService().ParseOzonCookie(ctx)
                        if err != nil || ozonID == 0 {
                                return nil, model.ErrorNoOzonAuth
                        }
                        return next(ctx)
                },
                BankOrOzonAuthenticated: resolver.AuthService().BankOrOzonAuthenticated,
                BankOrOzonTrusted:       resolver.AuthService().BankOrOzonTrusted,
                SessionRateLimit: func(ctx context.Context, obj interface{}, next graphql.Resolver, sessionType model.Sess
                        if !gqlThrottler.Enabled() {
                                return next(ctx)
                        }
```

Директивы в graphql

В том числе можно вешать директивы на отдельные поля, чем мы и воспользовались, решив сделать под каждый тип ID объекта со специфической проверкой свою auth-директиву:

```
"""
IDOR: check rko issueId
"""
directive @checkWalletIssueIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check issue id
"""
directive @checkIssueIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check phone belongs to client
"""
directive @checkPhoneBelongsToClientAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check daily invoice id
"""
directive @checkDailyInvoiceIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check daily invoice ids
"""
directive @checkDailyInvoiceIdsAccess on INPUT_FIELD_DEFINITION
"""
IDOR: check b2bInstallment companyOzonId
"""
directive @checkB2BInstallmentUpdateScoringCompanyOzonIDAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check challenge id
"""
directive @checkChallengeIdAccess on INPUT_FIELD_DEFINITION
```

Директивы в graphql

В том числе можно вешать директивы на отдельные поля, чем мы и воспользовались, решив сделать под каждый тип ID объекта со специфической проверкой свою auth-директиву:

```
"""
IDOR: check rko issueId
"""
directive @checkWalletIssueIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check issue id
"""
directive @checkIssueIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check phone belongs to client
"""
directive @checkPhoneBelongsToClientAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check daily invoice id
"""
directive @checkDailyInvoiceIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check daily invoice ids
"""
directive @checkDailyInvoiceIdsAccess on INPUT_FIELD_DEFINITION
"""
IDOR: check b2bInstallment companyOzonId
"""
directive @checkB2BInstallmentUpdateScoringCompanyOzonIDAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check challenge id
"""
directive @checkChallengeIdAccess on INPUT_FIELD_DEFINITION
```

```
type CancelDfaRegistrationByClientPayload {
    """ Выпуск """
    issue: SchemaIssue!
}

input CancelDfaRegistrationByClientInput {
    """ Идентификатор выпуска"""
    issueId: UUID! @checkIssueIdAccess
}

type ContinueDfaRegistrationPayload {
    """ Выпуск """
    issue: SchemaIssue!
    """ Время до следующей переотправки пин-кода """
    resendAfter: DateTime
    """ Длина OTP кода """
    otpLength: Int
}

input ContinueDfaRegistrationInput {
    """ Идентификатор выпуска"""
    issueId: UUID! @checkIssueIdAccess
}
```

Директивы в graphql

В том числе можно вешать директивы на отдельные поля, чем мы и воспользовались, решив сделать под каждый тип ID объекта со специфической проверкой свою auth-директиву:

```
"""
IDOR: check rko issueId
"""
directive @checkWalletIssueIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check issue id
"""
directive @checkIssueIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check phone belongs to client
"""
directive @checkPhoneBelongsToClientAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check daily invoice id
"""
directive @checkDailyInvoiceIdAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check daily invoice ids
"""
directive @checkDailyInvoiceIdsAccess on INPUT_FIELD_DEFINITION
"""
IDOR: check b2bInstallment companyOzonId
"""
directive @checkB2BInstallmentUpdateScoringCompanyOzonIDAccess on INPUT_FIELD_DEFINITION

"""
IDOR: check challenge id
"""
directive @checkChallengeIdAccess on INPUT_FIELD_DEFINITION
```

```
type CancelDfaRegistrationByClientPayload {
    """ Выпуск """
    issue: SchemaIssue!
}

input CancelDfaRegistrationByClientInput {
    """ Идентификатор выпуска"""
    issueId: UUID! @checkIssueIdAccess
}

type ContinueDfaRegistrationPayload {
    """ Выпуск """
    issue: SchemaIssue!
    """ Время до следующей переотправки пин-кода """
    resendAfter: DateTime
    """ Длина OTP кода """
    otpLength: Int
}

input ContinueDfaRegistrationInput {
    """ Идентификатор выпуска"""
    issueId: UUID! @checkIssueIdAccess
}
```

```
// CheckAccess checks that issueID belongs to the client
func (r *issueIDCheckerResolver) CheckAccess(ctx context.Context, value any) error {
        session := auth.GetSession(ctx)
        issueID, ok := value.(string)

        if !ok {
                return fmt.Errorf("issueIDCheckerResolver.CheckAccess: unexpected field type %T", value)
        }

        isBelong, err := r.harryHelper.IsIssueBelongToClient(ctx, issueID, session.ClientId)
        if err != nil {
                return fmt.Errorf("issueIDCheckerResolver.CheckAccess: harryHelper.IsIssueBelongToClient: %w", err)
        }
        if !isBelong {
                return model.ErrorRequestWrongIssue
        }

        return nil
}
```

ozon tech

Директивы в graphql

Но как не забыть повешать @checkAccess на поле с ID в новой ручке? Вроде уже дешево, но все еще не спасает…

Директивы в graphql

Но как не забыть повешать @checkAccess на поле с ID в новой ручке? Вроде уже дешево, но все еще не спасает…

И мы добавили линтер в CI:

```
$ $CI_PROJECT_DIR/gql_linter --blocking-mode --schemas-dir=$CI_PROJECT_DIR/api/graphql --validate-idor -commit origin/$DIFF_BRANCH -commit-schemas-dir ./api/graphql | tee gql_linter_log
Старт прасинга GraphQL
INFO:  find 25 files in ./api/graphql directory
Confluence: Проверка на IDOR через директивы gql и пакет idor:
https://confluence.o3.ru/pages/viewpage.action?pageId=603755900
Валидация GraphQL схемы завершена
exit code 0
```

Директивы в graphql

Но как не забыть повешать @checkAccess на поле с ID в новой ручке? Вроде уже дешево, но все еще не спасает…

И мы добавили линтер в CI:

```
$ $CI_PROJECT_DIR/gql_linter --blocking-mode --schemas-dir=$CI_PROJECT_DIR/api/graphql --validate-idor -commit origin/$DIFF_BRANCH -commit-schemas-dir ./api/graphql | tee gql_linter_log
Старт прасинга GraphQL
INFO:  find 25 files in ./api/graphql directory
Confluence: Проверка на IDOR через директивы gql и пакет idor:
https://confluence.o3.ru/pages/viewpage.action?pageId=603755900
Валидация GraphQL схемы завершена
exit code 0
```

В нем парсим текущую gql-схему и gql-схему в мастер ветке, по новым полям похожим на id (и другие релевантные поля типа phone email inn и тд) проверяем наличие директивы или исключения согласованного с ИБ.

Директивы в graphql

Но как не забыть повешать @checkAccess на поле с ID в новой ручке? Вроде уже дешево, но все еще не спасает…

И мы добавили линтер в CI:

```
$ $CI_PROJECT_DIR/gql_linter --blocking-mode --schemas-dir=$CI_PROJECT_DIR/api/graphql --validate-idor -commit origin/$DIFF_BRANCH -commit-schemas-dir ./api/graphql | tee gql_linter_log
Старт прасинга GraphQL
INFO:  find 25 files in ./api/graphql directory
Confluence: Проверка на IDOR через директивы gql и пакет idor:
https://confluence.o3.ru/pages/viewpage.action?pageId=603755900
Валидация GraphQL схемы завершена
exit code 0
```

В нем парсим текущую gql-схему и gql-схему в мастер ветке, по новым полям похожим на id (и другие релевантные поля типа phone email inn и тд) проверяем наличие директивы или исключения согласованного с ИБ.

Если ничего такого не находится – джоба падает и разработчик не может катнуть новый код.

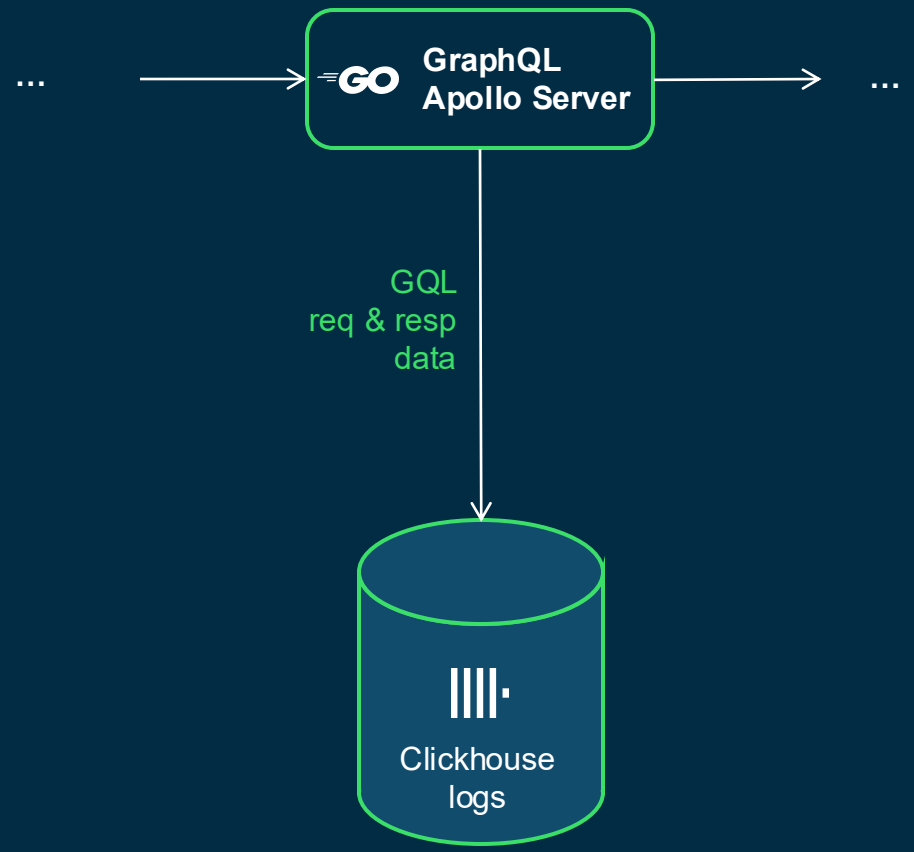
Директивы в graphql

Мы же смотрим пристально исключения там где похожее на ID тем не менее проверять не нужно:

```
""" Запрос создание подписки """
input CreateDailySubscriptionInput {
    """ Наименование подписки """
    title: String!
    """ ID провайдера """
    providerId: String @skipCheckAccess #IDOR OB-134753
    """ Тип документа """
    documentType: SubscriptionDocumentTypeInput!
    """ Номер документа """
    documentNumber: String!
    """ ID категории """
    categoryId: String! @skipCheckAccess #IDOR OB-134753
}
```

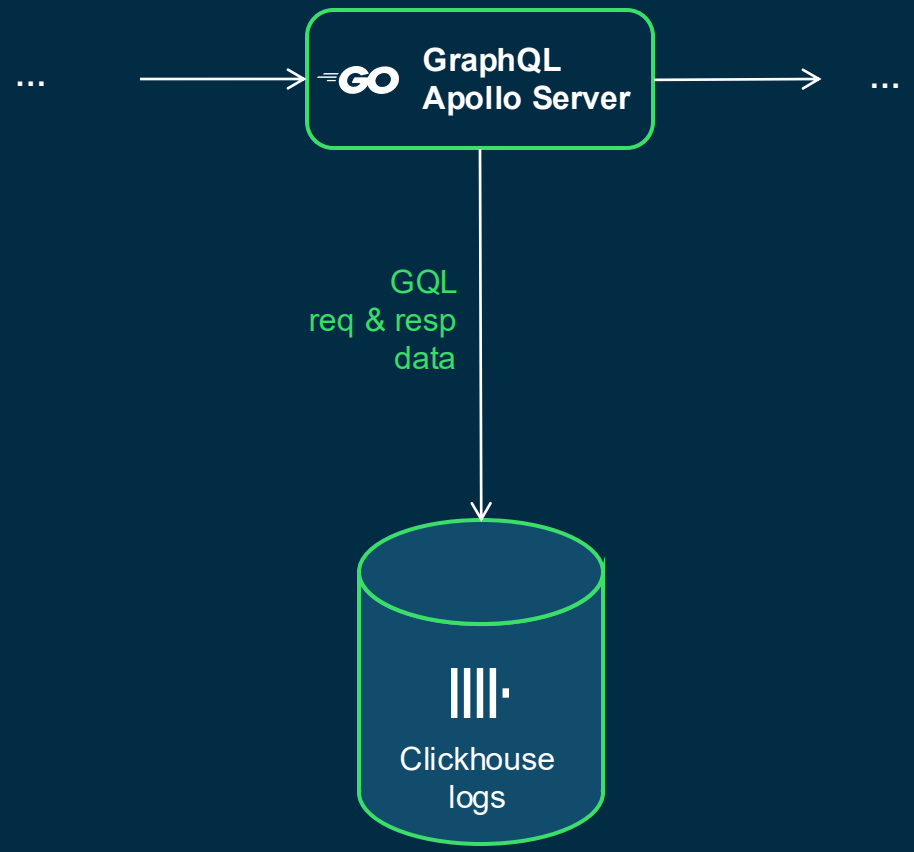
04 { Мониторинг эксплуатации IDOR

Ландшафт



Пишем в логи все что летит к нам от клиентов:

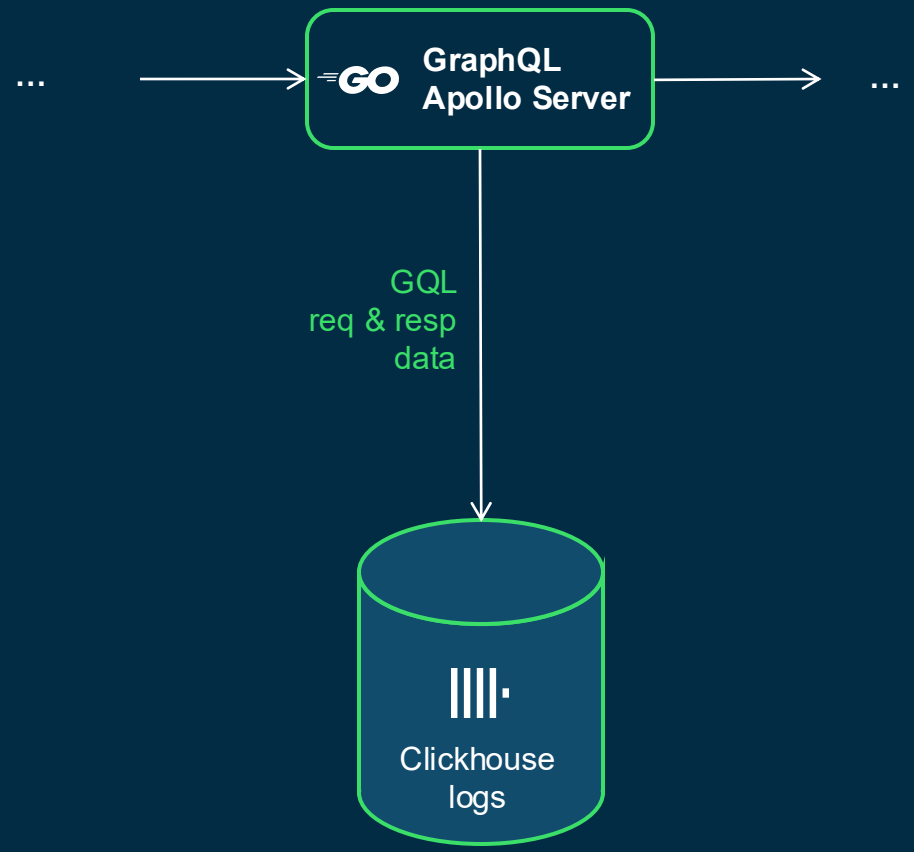
Ландшафт



Пишем в логи все что летит к нам от клиентов:

- Graphql квери и мутации

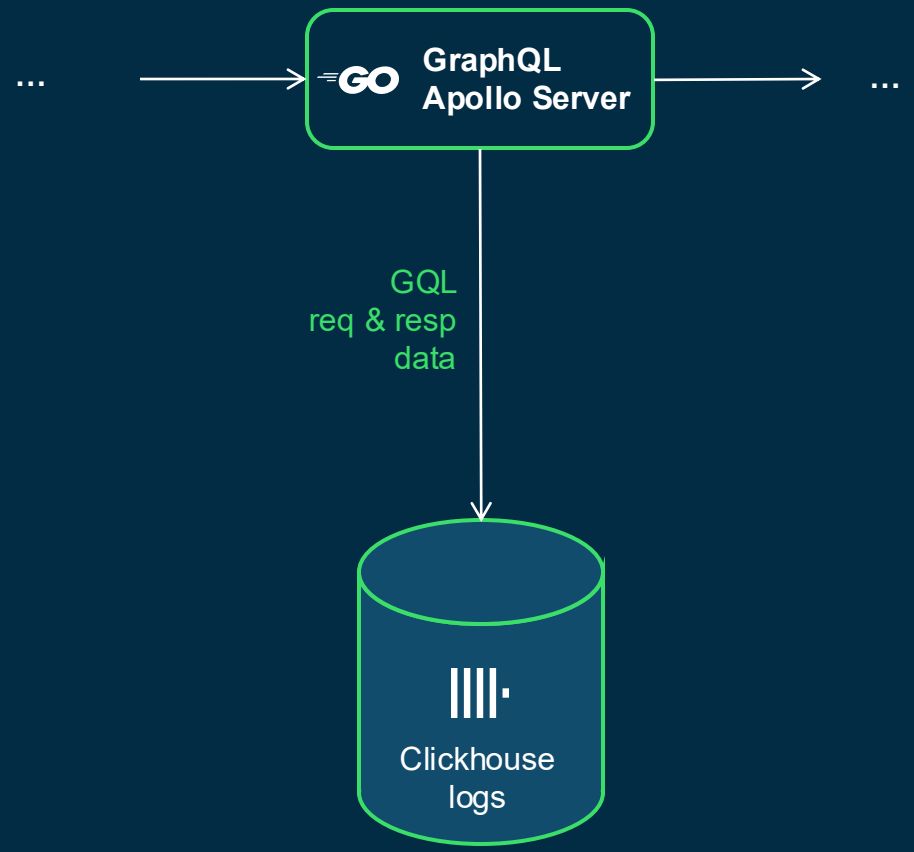
Ландшафт



Пишем в логи все что летит к нам от клиентов:

- Graphql квери и мутации
- Передаваемые на вход параметры

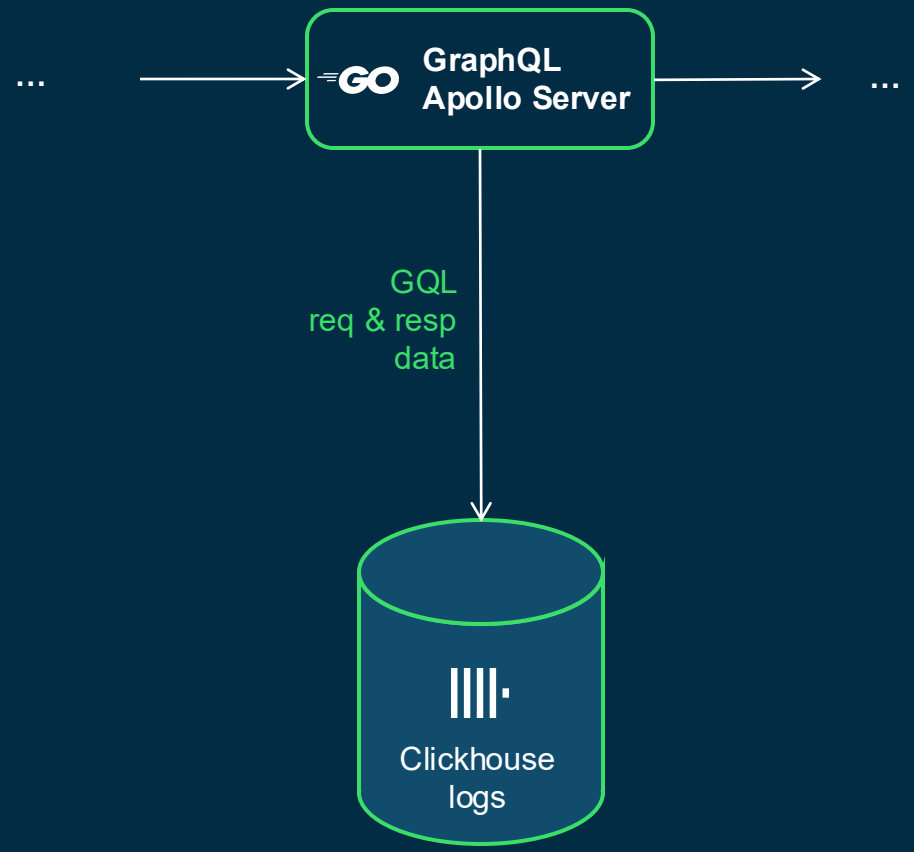
Ландшафт



Пишем в логи все что летит к нам от клиентов:

- Graphql квери и мутации
- Передаваемые на вход параметры
- Ответ сервера (не всегда)

Ландшафт



Пишем в логи все что летит к нам от клиентов:

- Graphql квери и мутации
- Передаваемые на вход параметры
- Ответ сервера (не всегда)

Хотим найти в логах перебор по ID.

Clickhouse

Достаем из входящих параметров все похожее на ID и пишем агрегат на количество уникальных значений за окно времени (сколько человек с улицы потрогал данных):

```
SELECT
    user_ozon_id,
    action,
    count(
        distinct arrayJoin(
            extractAll(
                request_gql_variables,
                '"([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}|[0-9]{20})"'
            )
        )
    ) as hits
FROM ob.logs_ng
WHERE
    request_time >= now() - INTERVAL 4 HOUR
    and action_type = 'gql'
    and user_ozon_id > 0
    and request_gql_variables REGEXP '"([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}|[0-9]{20})"'

GROUP BY user_ozon_id, action
HAVING hits > 150
ORDER BY hits DESC
```

И вешаем на него алерты

That’s all Falks!

@dmarushkin

In/dmarushkin