[Ru] Капча

Капча для авторизации

Минимальные требования к версиям:
`wlc_core`: **`1.6.0`** (#294263)
`wlc-engine`: **`1.3.76`** (#324670)
`wlc-theme`: **`scr324671`** (#324671)

Принцип работы на back-end.

За работу капчи на стороне **`back-end`** отвечает **`wlc_core`**.

Капча работает вместе с **ограничителем запросов** (**Rate Limiter**).
При этом, для удобства тестирования, сам **Rate Limiter** можно отключить через параметр **`$cfg['disableRateLimiter'] = true;`** или белый список.
Подробнее про **Rate Limiter**.

В отличие от автоматической капчи, этот вариант для конкретного IP адреса. Т. е. запрос капчи будет не у всех, а у конкретного пользователя.

> **Примечание: это утверждение не было проверено.**
>
> думаю нужно оставить одновременно работающими и рекапчу по старой схеме и свою капчу по этой
> но проверку по нашей капче сделать до рекапчи
>
> в этом случае будет:
> - если по нашей табличке с этого IP пытаются перебирать учётки, то показываем ему нашу капчу либо вообще отдаём 429
> - если с этим IP по нашей табличке всё ок, но эндпоинты долбят, то требуем гуглкапчу
>
> при такой схеме нельзя будет много перебрать учёток с одного IP вне зависимости от того как этот IP проходит капчи
> и даже имея большой пул IP нельзя поперебирать не обходя как-то гуглкапчу

С одного IP после превышения установленного количества попыток авторизации (**N**) нужно будет пройти капчу, которая возвращается в объекте по ключу `captcha` в виде строки, формата `data:image/jpeg;base64`, которую на **`ftont-end`** будут показывать юзеру.
Отправка ответа на капчу должна происходить в **заголовке** `X-Captcha`.

Если ответ неверный, будет прислана новая капча и новое поле `error` с ошибкой о недействительном коде.
Если ответ верный, счётчик "количество в час" будет сброшен до нового набора **N** неправильных попыток.

Если за сутки наберётся определенное количество неправильных попыток (**X**), то IP блокируется на сутки.
Это выражается статусом **429** и заголовком `Retry-After` с RFC7231, где указано когда IP будет разблокирован.
*Важно: успешная авторизация **НЕ** сбрасывает счётчик **X**.*

Счётчики сбрасываются, когда истекает время жизни соответствующего счётчика (**N** - 1 час, **X** - 1 день). Произойдет сброс попыток на 0.

Запросы, которые попадают под капчу
`PUT  api/v1/auth`

Принцип работы на front-end.

`wlc-theme` и **`wlc-engine`** капча работает через компонент `<wlc-captcha></wlc-captcha>` и сервис `CaptchaService.`

Включение на проекте.

Для включения нужно в конфиге **`siteconfig.php`** прописать настройки.

Файл находится в проекте в виде одного или нескольких файлов.
Ниже указаны стандартные пути, но в вашем проекте они могут быть другие.

Стандартные пути файла

```
roots/siteconfig.php
config/backend/0.site.config.php
```

Минимальные настройки

```
$cfg['rateLimitProfiles'] = true;
$cfg['enableCaptcha'] = true;

$cfg['captchaConfig'] = [
  'hour' => 2,
  'day' => 10
];
```

Полный перечень доступных настроек со значениями по умолчанию...

```
$cfg['rateLimitProfiles'] = false;

$cfg['enableCaptcha'] = false
$cfg['captchaConfig'] = [
  'hour' => 2,
  'day' => 10
];
```

Как тестировать капчу

Если Rate Limiter мешает проверке капчи

Например, **Rate Limiter на авторизацию** блокирует IP после 3 неверных запросов.

Правильным решением будет указать белый список адресов, которые не попадают под действие ограничителя.
Принимает массив строк, например ['0.0.0.0', '188.130.240.79'].
Указать надо внешний IP адрес, который смотрит в Интернет.

```
$cfg['rateLimiterIPsWhiteList'] = ['188.130.240.79'];
```

Другим решением будет отключать ограничитель запросов для всех.
Не рекомендуется к появлению на проде!

```
$cfg['disableRateLimiter'] = true;
```

Подробнее про RateLimiter.