01

Interceptando as requisições

Transcrição

Tentamos fazer a validação do clique do botão *reCAPTCHA*, mas acabamos recebendo uma *Exception*. Isso aconteceu porque na assinatura do método `enviaToken()`, que está na classe `GoogleService` usa como retorno `Call<String>`, mas o Google está retornando um **JSON** em vez de **String**. O problema é que não sabemos como o JSON está formatado.

Para identificar a formatação do JSON, precisamos encontrar uma forma de interceptar as requisições que são feitas entre a aplicação e o servidor do Google. Pararemos o Tomcat e na classe `RetrofitInicializador`, passaremos a interceptação no construtor, antes de criar o objeto do **Retrofit**. Dentro do construtor, criaremos uma instância `new HttpLogginInterceptor()`.

```
public class RetrofitInicializador {

    private static final String BASE_URL = "https://www.google.com/recaptcha/api/";
    private Retrofit retrofit;

    public RetrofitInicializador(){

        HttpLogginInterceptor interceptor = new HttpLoggingInterceptor();

        retrofit = new Retrofit.Builder().baseUrl(BASE_URL).addConverterFactory(GsonConverterFa
    }

    public GoogleService getGoogleService() {
        return retrofit.create(GoogleService.class);
    }
}
```

Com o objeto criado, é necessário especificar qual é o nível de interceptação que desejamos, como cabeçalhos, corpo, nenhuma interceptação e assim por diante. Queremos todos os detalhes, por isso pediremos o corpo da requisição.

```
public class RetrofitInicializador {

    private static final String BASE_URL = "https://www.google.com/recaptcha/api/";
    private Retrofit retrofit;

    public RetrofitInicializador(){

        HttpLogginInterceptor interceptor = new HttpLoggingInterceptor();
        interceptor.setLevel(Level.BODY);

        retrofit = new Retrofit.Builder().baseUrl(BASE_URL).addConverterFactory(GsonConverterFa
    }

    public GoogleService getGoogleService() {
        return retrofit.create(GoogleService.class);
    }
}
```

Teríamos que encontrar uma forma de colocar o `interceptor` na criação do objeto `retrofit`, porém o `retrofit` não consegue trabalhar com o interceptador diretamente. Os dois objetos trabalham com uma biblioteca chamada `OkHttpClient`, por isso vamos configurá-lo para montar o objeto do `retrofit` trabalhando com o `interceptor`.

Instanciaremos `new OkHttpClient.Builder()`, em seguida com o atalho "Ctrl + 1" associaremos o objeto a uma variável `Builder client`. Com o objeto `client`, chamaremos o método `client.addInterceptor(interceptor)`.

```
public class RetrofitInicializador {

    private static final String BASE_URL = "https://www.google.com/recaptcha/api/";
    private Retrofit retrofit;

    public RetrofitInicializador(){

        HttpLogginInterceptor interceptor = new HttpLoggingInterceptor();
        interceptor.setLevel(Level.BODY);

        Builder client = new OkHttpClient.Builder();
        client.addInterceptor(interceptor);

        retrofit = new Retrofit.Builder().baseUrl(BASE_URL).addConverterFactory(GsonConverterFa
    }

    public GoogleService getGoogleService() {
        return retrofit.create(GoogleService.class);
    }
}
```

Agora basta associarmos o objeto `client` na criação do `retrofit`. Antes da chamada ao método `build()`, chamaremos o método `client()` passando o objeto construído com `client.build()`.

```
public class RetrofitInicializador {

    private static final String BASE_URL = "https://www.google.com/recaptcha/api/";
    private Retrofit retrofit;

    public RetrofitInicializador(){

        HttpLogginInterceptor interceptor = new HttpLoggingInterceptor();
        interceptor.setLevel(Level.BODY);

        Builder client = new OkHttpClient.Builder();
        client.addInterceptor(interceptor);

        retrofit = new Retrofit.Builder().baseUrl(BASE_URL).addConverterFactory(GsonConverterFa
        .client(client.build())
        .build();
    }

    public GoogleService getGoogleService() {
        return retrofit.create(GoogleService.class);
```

```
    }
}
```

Dessa forma estamos colocando o `interceptor` na em um objeto da class `OkHttpClient` e esse objeto estamos passando para a criação do `retrofit`. Se tudo deu certo veremos como o JSON está sendo retornado. Reiniciaremos o Tomcat e tentaremos efetuar o *Login* com a conta da Ana.

Olhando no Console do Eclipse, veremos que foi retornado um JSON com as informações:

```
INFORMAÇÕES: {
    "success" : true,
    "challange_ts" : "2017-10-23T13:39:44Z",
    "hostname" : "127.0.0.1"
}
```

O valor `"success"` foi `true`, isso significa que o clique foi validado. Para garantir o correto funcionamento, tentaremos fazer o *Login*, mas **sem clicar no reCAPTCHA**, dessa forma esperamos que o retorno seja `false`.

```
INFORMAÇÕES: {
    "success" : false,
    "error-codes" : [
        "missing-input-response"
    ]
}
```

Funcionou, o Google não validou o clique. Já sabemos como o Google está enviando o JSON, por isso criaremos uma classe para receber a chave `"success"`. Com o Tomcat parado, dentro do pacote `br.com.alura.owasp.retrofit` criaremos a classe `Resposta` com o atributo de exatamente igual ao parâmetro recebido no JSON:

```
package br.com.alura.owasp.retrofit;

public class Resposta {

    private boolean success;
}
```

Com a classe criada, acessaremos a interface `GoogleService` e trocaremos o tipo de retorno no método.

```
public interface GoogleService {

    @POST("siteverify")
    Call<Resposta> enviaToken(@Query("secret") String secret, @Query("response") String respons

}
```

A classe `GoogleWebClient` quebrou, também é necessário trocarmos o tipo de retorno na classe. Dentro o método `verifica()`, temos a variável `Call<String> token`, nó mudaremos para `Call<Resposta> token`. Precisaremos verificar se o valor é `true` ou `false`, então chamaremos o método `token.execute().body().isSuccess()`.

```
package br.com.alura.owasp.retrofit;

public class GoogleWebClient {

    private static final String SECRET = "6LddPTUUAAAAAOkAx05jP2N9rIPlhf3WQHMuHMjZ";

    public void verifica(String recaptcha) {

        Call<Resposta> token = new RetrofitInicializador().getGoogleService().enviaToken(SECRET,
        token.execute().body().isSuccess();

    }
}
```

O Eclipse está reclamando que o método não existe, usaremos o atalho o método "Ctrl + 1" e selecionaremos a opção *Create method*. No método nós apenas retornaremos o valor do atributo `success`.

```
package br.com.alura.owasp.retrofit;

public class Resposta {

    private boolean success;

    public boolean isSuccess() {
        return success;
    }
}
```

Com isso já temos a informação do clique validado pelo Google. Veremos se conseguimos proteger a aplicação contra o ataque de força bruta.