一、功能说明

越权工具包是一个围绕越权问题展开的功能实现。

它有前后端无感知、高性能、轻量级的后端通用解决方案的特点, 可以全面杜绝 `越权` 问题。

背景

- 近期部门内的多个项目被安全组指出, 存在越权漏洞。
- 该漏洞的特点是: 常规编码方案工作量大, 没有现成的解决方案、涉及前后端交互、危害用户隐私
- 为了提升部门内的数据安全保护措施并永久性解决越权漏洞, 决定研发该通用工具包。

二、工具使用

- 添加依赖: pom.xml

```
<dependency>
    <groupId>cn.gmlee</groupId>
    <artifactId>tools-overstep</artifactId>
    <version>3.1.1</version>
</dependency>
```

- 启用工具

> 在启动类上或其他配置类上扫描包: cn.gmlee.tools

```
@SpringBootApplication
@ComponentScan("cn.gmlee.tools")
public class XxxApp {
    public static void main(String[] args) {
        SpringApplication.run(XxxApp.class, args);
    }
}
```

- 高阶用法

> 请联系@Author

三、工具原理

逻辑

- 参数加密
  - 发生越权的特点是: 获取数据的核心参数是有序且可预测的
  - 所以只需要对核心参数进行 **加密**、混淆, 使其需要花费数年才可能破解, 如下:

```
// 加密内容: TOOLS007888545175972659479727067
// 真实数据: 8848779
public static void main(String[] args) {
    String sn = SnUtil.getSn(8848779L);
    System.out.println(sn);
    Long id = SnUtil.getId(sn);
    System.out.println(id);
}
```

- 控制台输出如下

```
Connected to the target VM, address: '127.0.0.1:59828', transport: 'socket'

TOOLS007888545175972659479727067
8848779

Disconnected from the target VM, address: '127.0.0.1:59828', transport:
'socket'

Process finished with exit code 0
```

- 有趣的是
  - **每次加密的结果都是不一样的**;
  - **改动任何一位数字都将无法还原真实数据**;
  - 哪怕后端人员也不能重复使用它 (默认不启用)。

- 响应数据
  - 采用转换器方案将可能用于越权的参数(默认id)进行**加密**、混淆
  - 前端无感知使用id, 包括请求后端数据

- 请求处理
  - 采用转换器方案处理请求数据
  - 将指定参数(默认id)进行越权解析
  - 如果无法解析表明此次请求是越权行为

代码

- 真实数据加密: XorUtil工具对真实数据使用随机数(默认4位)进行**加密**
- 参数加密采用: SnUtil工具默认32位固定长度混淆方式
- 接口从数据库获取到的核心数据将进行混淆返回
  - SnMappingJackson2HttpMessageConverter.java
- 前端无感知使用核心数据并发送给后端接口
- 接口从请求中将核心参数进行解析得到真实数据
  - SnConverter.java
  - SnMappingJackson2HttpMessageConverter.java
- 后端无感知使用核心数据

数据

- 工具没有数据产生

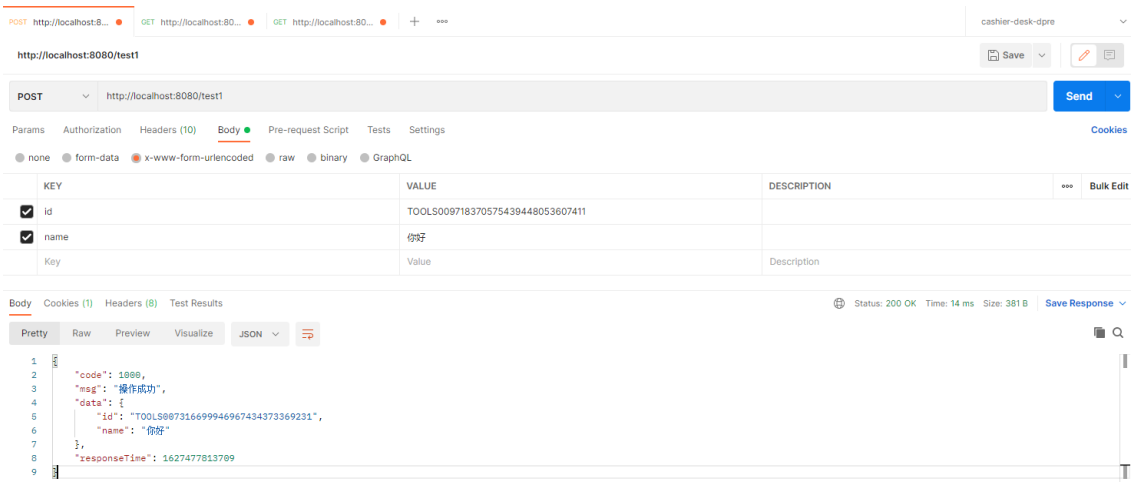
四、使用示例

- 代码示例

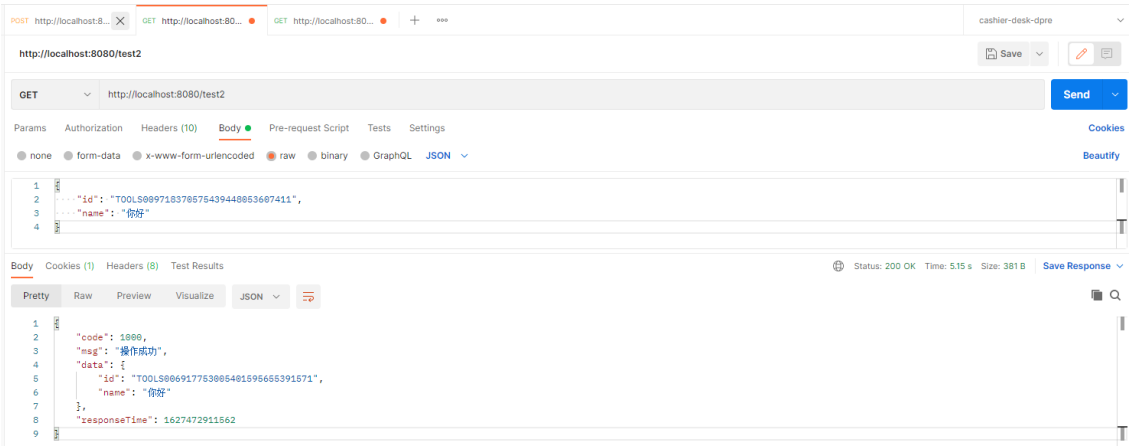
```
@Data
public class Demo {
    private Long id;
    private String name;
}
```

```
/**
 * 越权测试.
 *
 * @author Jas°
 * @date 2021/7/28 (周三)
 */
@RestController
public class DemoController {

    @RequestMapping("test1")
    public JsonResult<Demo> test1(Demo demo) throws Exception {
        return JsonResult.OK.newly(demo);
    }

    @RequestMapping("test2")
    public JsonResult<Demo> test2(@RequestBody Demo demo) throws Exception {
        return JsonResult.OK.newly(demo);
    }
}
```

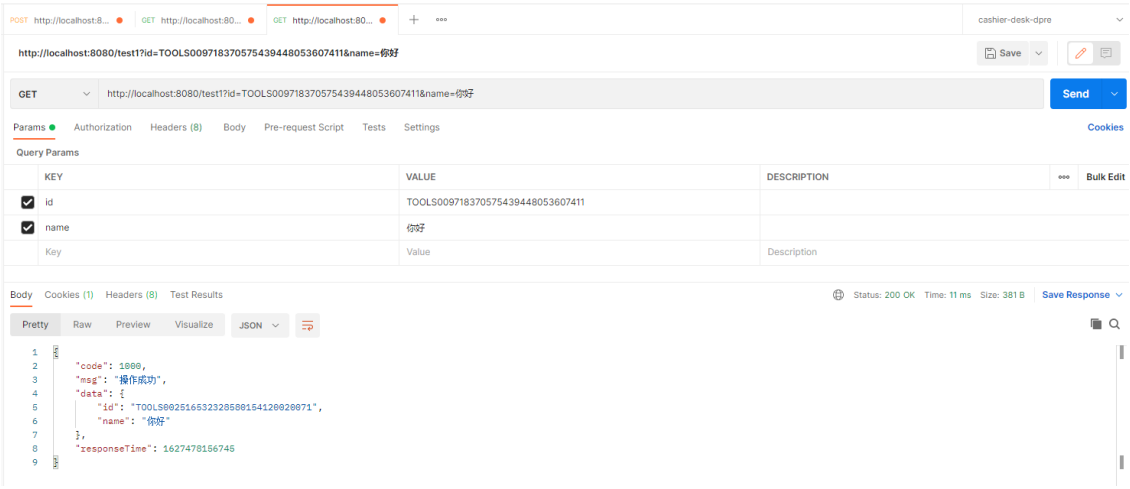
- 普通表单请求/响应



- APPLICATION 请求/响应


http://localhost:8080/test2
GET
http://localhost:8080/test2
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"id": "TOOLS00971837057543944805360741l",
"name": "你好"
}
Body Cookies (1) Headers (8) Test Results
Status: 200 OK Time: 5.15 s Size: 381 B
Save Response
Pretty Raw Preview Visualize JSON
{
"code": 1000,
"msg": "操作成功",
"data": {
"id": "TOOLS00691775300540159565539157l",
"name": "你好"
},
"responseTime": 1627472911562
}

- PATH 请求/响应


http://localhost:8080/test1?id=TOOLS00971837057543944805360741l&name=你好
GET
http://localhost:8080/test1?id=TOOLS00971837057543944805360741l&name=你好
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
id TOOLS00971837057543944805360741l
name 你好
Key Value Description
Body Cookies (1) Headers (8) Test Results
Status: 200 OK Time: 11 ms Size: 381 B
Save Response
Pretty Raw Preview Visualize JSON
{
"code": 1000,
"msg": "操作成功",
"data": {
"id": "TOOLS00251653232858015412002007l",
"name": "你好"
},
"responseTime": 1627478156745
}