什么是横向越权？如何防止？

典型回答

横向越权：是指攻击者尝试访问与当前用户具有相同权限的其他用户资源。
如：用户管理系统中，可以通过uid查询用户信息（包括用户手机号、家庭住址等敏感信息），且uid为数据库自增主键。此时攻击者可用自己的uid向下递减或向上递增试探性查询其他用户信息，导致用户信息泄露。

防范方法：无限增大攻击者攻击成本，包括不限于：

1. 参数以post方式提交，攻击者需要抓包才能看到参数
2. 关键参数进行加密混淆（主要手段）
3. 限制ip、当前uid在单位时间内的请求频次

扩展知识

以用户信息管理uid横向越权为例，如何对uid进行加密混淆?

基本思路：由于是查询接口，可以对uid直接进行对称加密，客户端或浏览器只要拿到加密后的密文提交查询请求即可，不需要知道uid明文是什么。服务端可以通过存储在服务器上的私钥进行解密拿到uid明文去查询数据库。

- 方案1（不可行）：使用散列算法，如md5，会导致uid不可逆，无法直接拿到数据库中查询。由于md5是32位或16位，攻击者也很容易观察出密文采用何种算法。
  `注意：md5用于生成摘要信息，并不是用来加密的。`
- 方案2（可行）：使用对称加密，如AES，服务端存储私钥，对uid加密后下发。查询接口携带密文，服务端通过私钥解密出明文uid，查询数据库即可。
- 方案3（推荐）：使用对称加密后，密文固定且通常会很长，如：

```
明文uid为1，私钥为123456，AES加密后密文为：
U2FsdGVkX1/fFmiULJXOvhkdWdBZLnf2Nw3UyFdF60g=
```

对于请求参数，要保证简单原则。同时如果能让每次请求uid密文变化，显然会大幅度增加攻击者的识别成本。
这里推荐开源的Sqids算法，官网链接。该算法本质是将一系列整数散列成一个字符串，并通过指定字符表的方式对散列进行多次打乱，确保不会被碰撞，且保证加密出的密文非常简短（同样是hash算法，md5可能出现碰撞且密文较长）。
我们可以把uid+下发uid密文时的请求时间戳组合起来进行散列，这样确保了每次同一个uid其密

文都不一样。客户端提交上来的密文，可以解出uid明文和时间戳。
使用方法：

```
// maven引入坐标
<dependency>
    <groupId>org.sqids</groupId>
    <artifactId>sqids_3</artifactId>
    <version>0.5.0</version>
</dependency>
```

```
// 创建sqids对象
SqidsOptions options=new SqidsOptions();
options.Alphabet="TGEpuRNDVtYvISsh34jz5c1db8eoPin6CJUgQwMAmLK9Farl2fW0OyHxqXkBZ7"; //指定字符表，确保密文不会被碰撞
Sqids sqids=new Sqids(options);
// 加密
String id=sqids.encode(Arrays.asList(1L,1696867066L)); //假设uid为1，时间戳为1696867066，加密后的密文为 W8mmdRiM6
// 解密
List<Long> numbers=sqids.decode(id); // 解密结果为[1, 1696867066]
```