安全

1 XSS

> 跨网站指令码（英语：`Cross-site scripting`，通常简称为：`XSS`）是一种网站应用程式的安全漏洞攻击，是代码注入的一种。它允许恶意使用者将程式码注入到网页上，其他使用者在观看网页时就会受到影响。这类攻击通常包含了 `HTML` 以及使用者端脚本语言

```
XSS` 分为三种：反射型，存储型和 `DOM-based
```

如何攻击

- `XSS` 通过修改 `HTML` 节点或者执行 `JS` 代码来攻击网站。
- 例如通过 `URL` 获取某些参数

```
<!-- http://www.domain.com?name=<script>alert(1)</script> -->
<div>{{name}}</div>
```

> 上述 `URL` 输入可能会将 `HTML` 改为 `<div><script>alert(1)</script></div>`，这样页面中就凭空多了一段可执行脚本。这种攻击类型是反射型攻击，也可以说是 `DOM-based` 攻击

如何防御

> 最普遍的做法是转义输入输出的内容，对于引号，尖括号，斜杠进行转义

```
function escape(str) {
    str = str.replace(/&/g, "&amp;");
    str = str.replace(/</g, "&lt;");
    str = str.replace(/>/g, "&gt;");
    str = str.replace(/"/g, "&quto;");
    str = str.replace(/'/g, "&##39;");
    str = str.replace(/`/g, "&##96;");
    str = str.replace(/\//g, "&##x2F;");
    return str
}
```

> 通过转义可以将攻击代码 `<script>alert(1)</script>` 变成

```
// -> &lt;script&gt;alert(1)&lt;&##x2F;script&gt;
escape('<script>alert(1)</script>')
```

> 对于显示富文本来说，不能通过上面的办法来转义所有字符，因为这样会把需要的格式也过滤掉。这种情况通常采用白名单过滤的办法，当然也可以通过黑名单过滤，但是考虑到需要过滤的标签和标签属性实在太多，更加推荐使用白名单的方式

```
var xss = require("xss");
var html = xss('<h1 id="title">XSS Demo</h1>
<script>alert("xss");</script>');
// -> <h1>XSS
Demo</h1>&lt;script&gt;alert("xss");&lt;/script&gt;
console.log(html);
```

> 以上示例使用了 `js-xss` 来实现。可以看到在输出中保留了 `h1` 标签且过滤了 `script` 标签

2 CSRF

> 跨站请求伪造（英语：`Cross-site request forgery`），也被称为 `one-click attack` 或者 `session riding`，通常缩写为 `CSRF` 或者 `XSRF`， 是一种挟制用户在当前已登录的 `Web` 应用程序上执行非本意的操作的攻击方法

> `CSRF` 就是利用用户的登录态发起恶意请求

如何攻击

> 假设网站中有一个通过 Get 请求提交用户评论的接口，那么攻击者就可以在钓鱼网站中加入一个图片，图片的地址就是评论接口

```
<img src="http://www.domain.com/xxx?comment='attack'"/>
```

如何防御

- `Get` 请求不对数据进行修改
- 不让第三方网站访问到用户 `Cookie`
- 阻止第三方网站请求接口
- 请求时附带验证信息，比如验证码或者 `token`

3 密码安全

加盐

> 对于密码存储来说，必然是不能明文存储在数据库中的，否则一旦数据库泄露，会对用户造成很大的损失。并且不建议只对密码单纯通过加密算法加密，因为存在彩虹表的关系

- 通常需要对密码加盐，然后进行几次不同加密算法的加密

```
// 加盐也就是给原密码添加字符串，增加原密码长度
sha256(sha1(md5(salt + password + salt)))
```

> 但是加盐并不能阻止别人盗取账号，只能确保即使数据库泄露，也不会暴露用户的真实密码。一旦攻击者得到了用户的账号，可以通过暴力破解的方式破解密码。对于这种情况，通常使用验证码增加延时或者限制尝试次数的方式。并且一旦用户输入了错误的密码，也不能直接提示用户输错密码，而应该提示账号或密码错误

前端加密

> 虽然前端加密对于安全防护来说意义不大，但是在遇到中间人攻击的情况下，可以避免明文密码被第三方获取