# 安全设计

## 总览

不同需求对安全产生不同的定义，安全对我们来说就是数据安全，即数据不被窃取，数据不被篡改，数据不被伪造。安全设计的核心是信任域的划分，而基于安全域的原子权限则是数据的坚实保护。原子权限正是这次安全设计的核心思路。

## 系统安全分层

以应用结构视角对整个系统的安全进行划分：

1. 系统（平台）层面的安全
2. 应用框架层面的安全
3. 代码实现层面的安全
4. 数据传输层面的安全
5. 客户端层面的安全

## 系统（平台）层面的安全

软件运行所需的基础环境既是此处的系统，包括以下内容：操作系统，WEB服务器中间件，软件运行所需的程序语言环境。

在这个层面主要从这几个方面进行防护：

1. 最小服务，通过减少与外界的接触面来降低入侵发生的可能性，只开放必要的服务端口
2. 安全版本，使用各种应用的最新stable版本来防止入侵
3. 合理的iptables设置，通过设置严格的出网规则来提高入侵的成本和难度
4. 合理的配置，对于各种应用的配置不应当直接复制网上的demo，而应该了解每一个配置会产生的行为后进行设计
5. Firewall的配置，禁止所有非TCP通信
6. 安装安全软件识别已有特征的恶意软件和行为
7. 根据入侵常用手段配置蜜罐，使得我们有可能在被入侵的时候得知
8. 数据与程序环境的分离，保证在得到程序环境的时候无法直接得到数据，对各种密钥进行加密
9. 合理的文件系统权限配置，保证得到网站权限无法修改网站代码

## 框架层面的安全

框架层面的安全即我们为了快速开发使用的框架

在这个层面从以下方面做好防范：

1. 不要从网上copy配置项，应当了解每一个配置会产生的行为，再进行框架的设置
2. 对于自己所使用的框架接口应当清楚会产生怎样的行为，而不是关注会产生怎样的结果
3. 在使用框架开发时应对开发环境和生产环境进行区分，在生产环境去除一切其他不必要的功能，比如报错

## 代码层面的安全

在开发时我们应当遵守以下简单的规则进行安全防护：

1. 简单了解owasptop10 后进行开发
2. 使用框架提供的或团队提供的安全操作函数
3. 在代码开发时应遵守OWASP安全编码指南
4. 对所有web接口进行
5. 在代码上线前使用白盒安全扫描工具进行测试，比如codeql
6. 一切用户资源文件都上传至OSS中

## 传输层面的安全

在数据传输我们使用以下方法进行防御：

1. 部署CDN防止发现真实IP
2. 负载均衡增加入侵难度
3. 部署云WAF防御恶意攻击
4. 配置访问频率防止DDOS和拆解攻击
5. 使用HTTPS加密传输数据
6. 前端对敏感数据比如密码使用ECDH密钥交换+AES加密

## 客户端层面的安全

在客户端层面我们要做好以下防护：

1. Csrf token 防止跨站请求伪造
2. 验证码防止暴力拆解，资源滥用
3. CSP防止恶意js脚本
4. Httponly防止cookie窃取

## 安全设计结构图

处于成本考虑，以docker 为单元进行服务隔离。

