信息安全第一弹

开发实践中的基本安全思维

第一弹

有弹友 不孤单

汪洋

CTO@第一弹APP

信息安全是什么?

信息安全是什么?

- 保证信息的机密性、完整性和可用性。
- 保证用户的隐私。

我们需不需要信息安全?

我们不需要安全，只需要产品体验。

– 用户

真的吗?

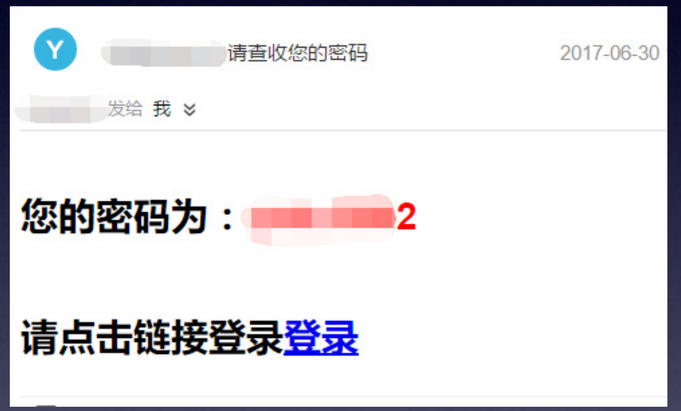
中国特色隐私泄漏

- 使用破解版（汉化版）软件
  - XcodeGhost
  - Putty
- 附近的人
- 提供密码导入信用卡账单
- 裸贷

中国特色隐私泄漏

我的密码忘了怎么办

您可以在登录页面，发送您的密码到注册邮箱。立即发送
本站不支持密码更改，请谅解！



真的吗?

索尼7700万网络用户数据遭泄漏 或含信用卡号

中国网 china.com.cn 时间： 2011-04-27 发表评论>>

·索尼第三次遭黑客攻击 损失或超10亿美元

索尼近日第三次遭遇网络黑客攻击，黑客窃取了2500名索尼用户的姓名和部分电子邮件地址，并将这些资料在一家网站上公布。有业内人士认为，在第三次黑客攻击面前，索尼似乎仍束手无策，其游戏网络服务的恢复将遥无定期，这使本来就饱受指责的索尼信誉度更加雪上加霜，经济损失更是可能超10亿美元。

- ·索尼自检黑客 大部分网络娱乐服务已恢复
- ·索尼被盗用户账号可能过亿 已暂停相关网络服务
- ·索尼承认信用卡资料或外泄 面临天价索赔

迄今互联网最大数据泄漏事件

数据外泄或涉及上亿人

索尼公司3日宣布，新发现大约2460万索尼网络服务用户的个人信息疑遭黑客窃取。这使得索尼数据遭窃案受影响的用户可能超过1亿人，成为迄今规模最大的用户数据外泄案。

索尼将损失超15亿美元

据了解，索尼公司此次的信息泄露事件将涉及用户信用卡问题，在一家名叫PSX-Scene的论坛上，黑客已经开始兜售220万个来自索尼PSN网络数据泄漏受害者的个人信息，并索价10万美元，不过其可信性有待商榷。

分析人士预计，此次事件索尼将损失超15亿美元，而对信用卡发卡机构损失的估计则是第一次。如果每位信用卡持卡人都要求更换新卡，那么信用卡发卡机构的成本将增加3亿美元。

不注重信息安全的公司的下场

> 慌了...正式环境忘改支付代码，所有的产品都以 1 分钱卖出去了。。。

财政部某站任意文件上传

历史仍在重演

国家电网某网站存在sql注入漏洞

中国联通沃云0元购买云主机

【内网案例】内网漫游某厂商

乐视网某站SQL注入3枚

信息安全，非常重要。

– 开发者

在产品执行过程中如何保证安全性

如何构建一个安全的登录、注册功能?

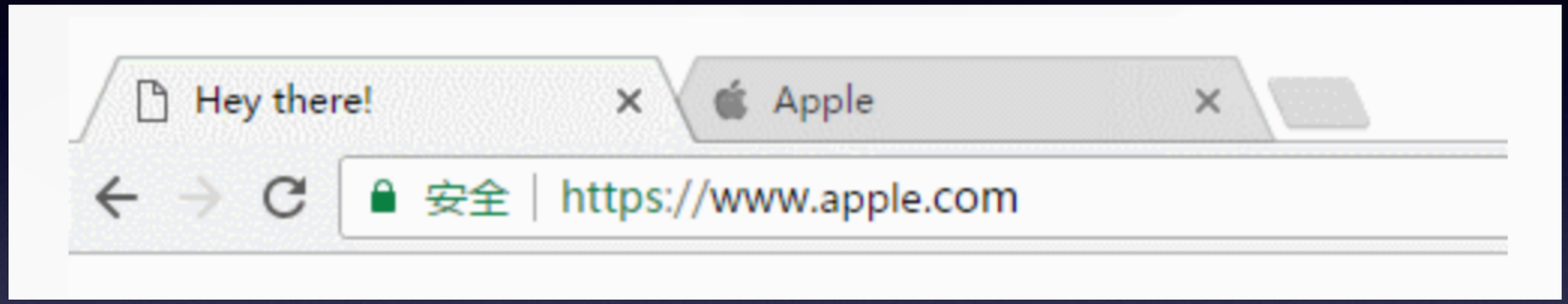
产品的登录、注册

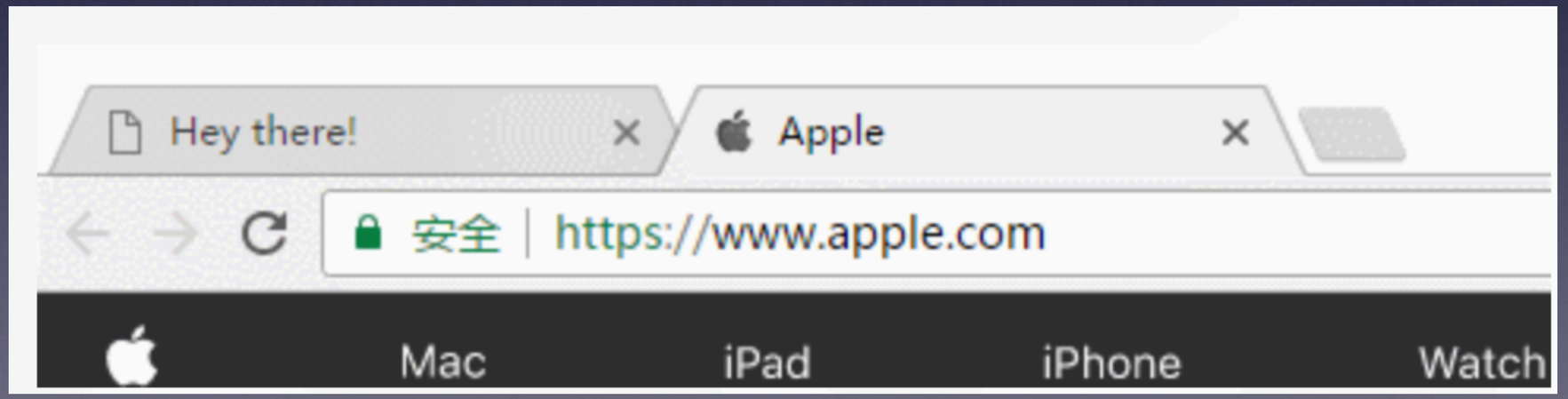
- 输入框展示
- 用户输入
- 上传数据
- 后台检验、存储
- 返回结果

输入框展示

- 让用户确信，他面前的输入框是安全的。
- 在pc端，其实就是传统的钓鱼网站攻击。
- 移动端理论上也可以通过覆盖界面进行攻击。

输入框展示


Hey there!
Apple
安全
https://www.apple.com


Hey there!
Apple
安全
https://www.apple.com
Mac
iPad
iPhone
Watch

用户输入

- 身边是否有人偷窥
- 远处是否有摄像头在拍摄
- 输入法是否恶意
- 是否有其他恶意应用存在

用户输入

- 使用内置输入法
- 打乱按键顺序
- 使用安全控件（TrustZone)

上传数据

- 保证用户密码安全地到达服务器，不被中间人窃取到。
- 传输层安全。
- 应用层安全。

上传数据

- 传输层安全：
- 是否使用HTTPS?
- 是否正确使用HTTPS?
  - 校验证书了吗?
  - 选择正确的加密方法了吗？（Android 5.0以后推荐用TLS1.2+ECDHE+RSA+AES 128 GCM+SHA256）

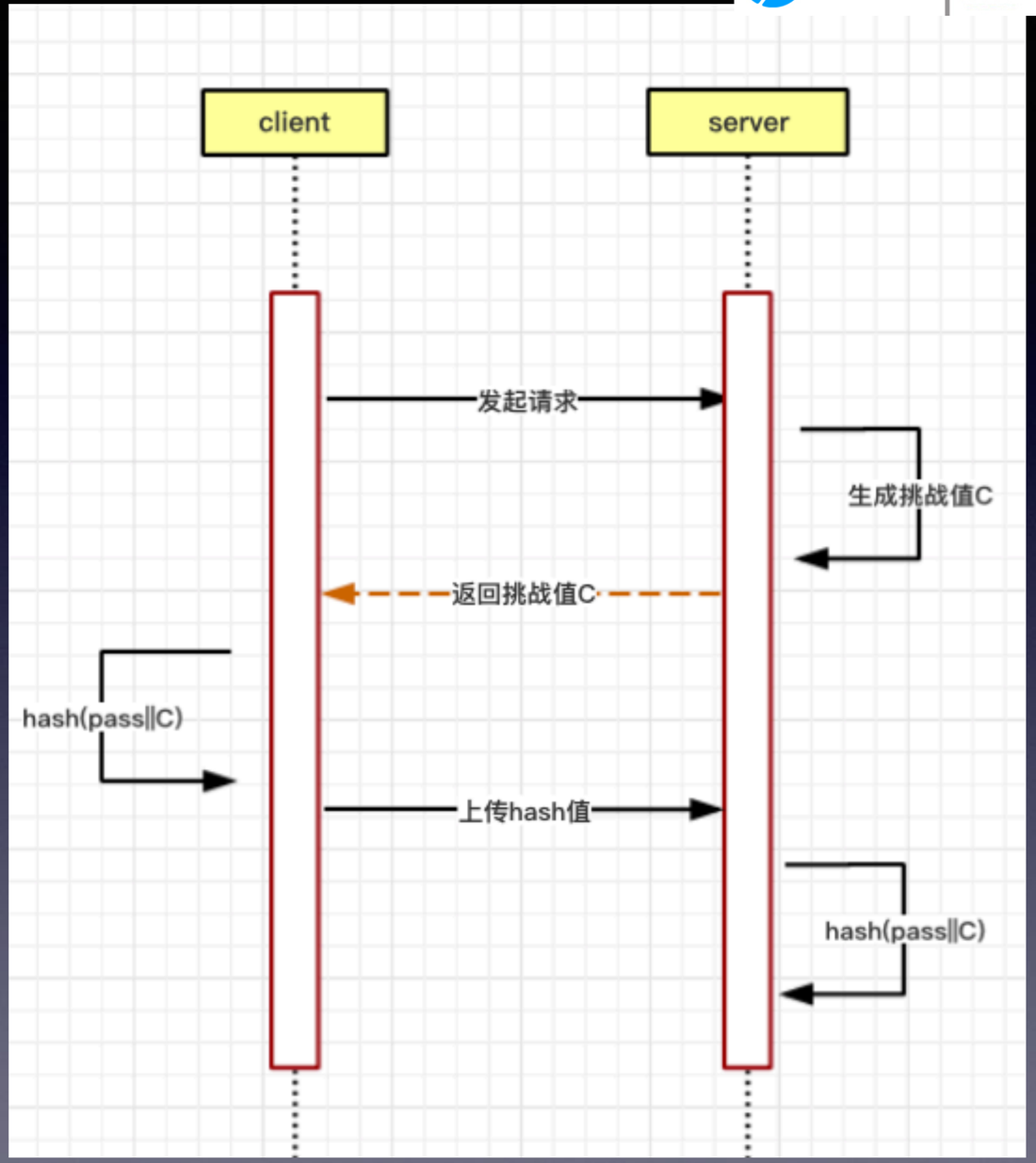
```
public class AcceptAllTrustManager implements X509TrustManager {

    @Override
    public void checkClientTrusted(X509Certificate[] chain, String authType) throws Ce
        //do nothing, 接受任意客户端证书
    }

    @Override
    public void checkServerTrusted(X509Certificate[] chain, String authType) throws Ce
        //do nothing, 接受任意服务端证书
    }

    @Override
    public X509Certificate[] getAcceptedIssuers() {
        return null;
    }
```

上传数据

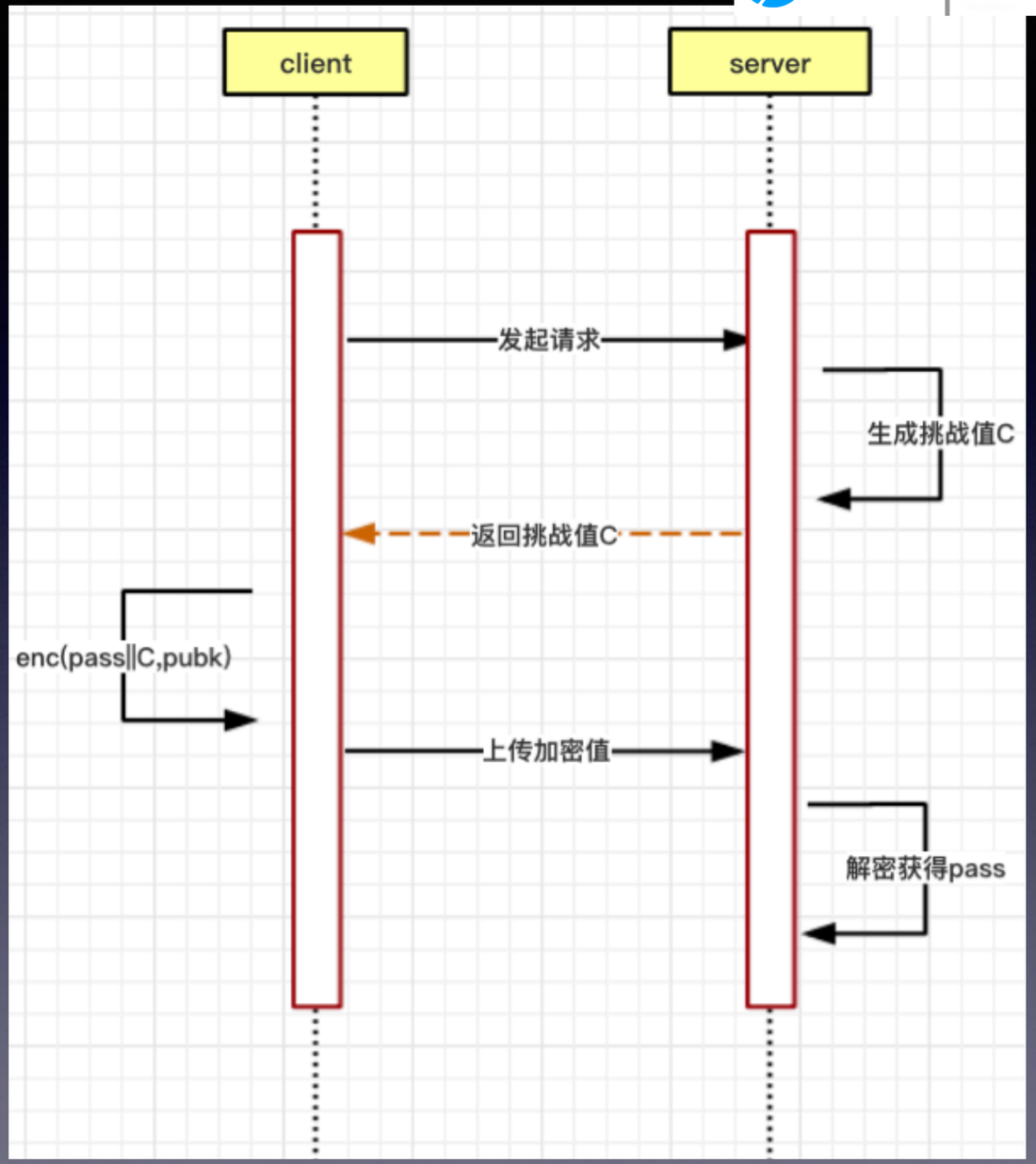
- 应用层安全：
- 数据是否进行额外加密处理?
- 计算密码的md5值进行上传。 
- 使用AES进行加密后上传。 
- 使用CHAP协议进行上传。 
- 使用公钥进行加密后上传。 

使用CHAP协议

chap协议存在一个重大缺陷。



使用公钥加密数据



后台校验、存储

- 保证用户密码安全地进行存储和读取。
- 明文存储。 
- 使用hash算法进行计算。 
- 使用非对称加密算法进行计算。 
- 使用对称加密算法进行计算。 

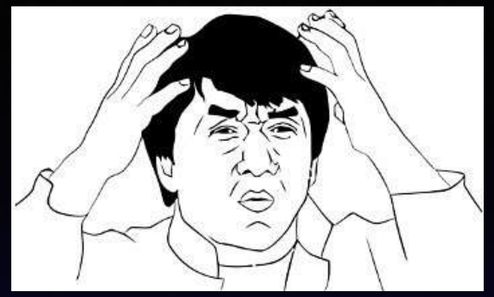
后台校验、存储

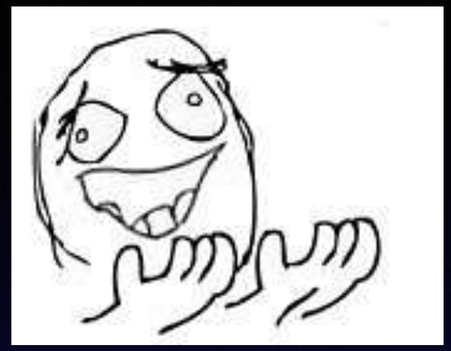
- 常见的hash算法。
- MD5
- SHA1
- SHA256
- PBKDF
- Bcrypt
- Scrypt
- Argon2

算法	计算速度	破解10位数字密码需要的时间
MD5	569.1 MH/s	1.8s
SHA1	304.0 MH/s	3.3s
SHA256	126.1 MH/s	7.9s
Bcrypt	2973 H/s	38d
Scrypt	31311 H/s	3d

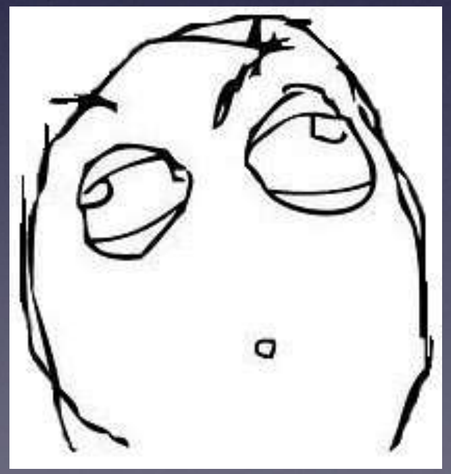
后台校验、存储

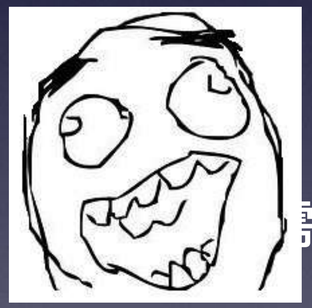
- 刚刚介绍了hash的算法，接下来看使用非对称加密的方案。
- 我们来看这么一个需求。

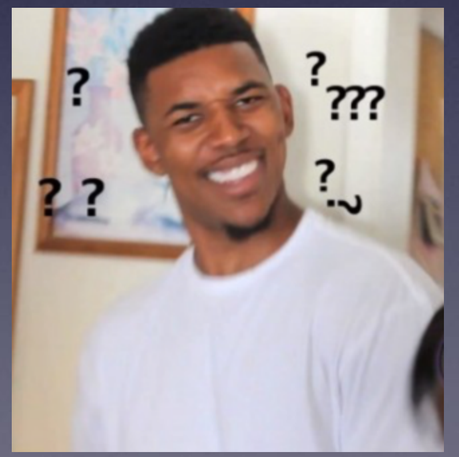




需要能随时查询用户的密码，但是不能被泄漏。





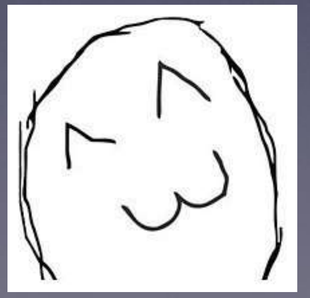


后台校验、存储

- 懵逼的同时，当然也是有解决方案的。
- 密码学中，最常见的我能知道，但是你不能知道的就是：
- 非对称加密算法。

后台校验、存储

- 我们可以这么做：
- 首先，生成一对公私钥，然后把私钥藏好，把公钥放到程序里。
- 用户进行注册时，将明文密码使用公钥进行加密后存储。
- 用户登录时，将用户上传的密码也经公钥加密后与数据库进行比对。
- 需要查询的时候，使用私钥对存储的数据进行解密即可。



后台校验、存储

- 计算和存储以后，在用户登录的时候，就需要进行比对：
- String1 == String2 
- 绝大部分语言在比较两个字符串的时候，当然遇到第一个不相等的字符时即返回结果。
- 也有一部分语言存在额外处理。<?php '0' == '0e12345'; ?>

后台校验、存储

- 所以，请使用各个语言提供的hash摘要比较函数。
- hmac.compare_digest
- hash_equals
- MessageDigest.isEqual

```
public static boolean isEqual(byte[] a, byte[] b) {
    if (a.length != b.length) {
        return false;
    }

    int result = 0;
    for (int i = 0; i < a.length; i++) {
      result |= a[i] ^ b[i];
    }
    return result == 0;
}
```

后台校验、存储

- 当然，请检查在查询用户密码的时候，是否还存在SQL注入现象。
- 除了使用普通数据库存储用户密码之外，还可以使用LDAP之类的目录数据库进行存储。

返回结果

- 安全防御方法和上传类似。
- 当前大部分登录结果都以 session 形式维持。
- 可以使用非对称加密来进行登录状态维持。

返回结果

- 安全防御方法和上传类似。
- 当前大部分登录结果都以 session 形式维持。
- 可以使用非对称加密来进行登录状态维持。

产品的登录、注册

- 输入框展示
- 用户输入
- 上传数据
- 后台检验、存储
- 返回结果

你所忽略掉的每一个细节，都可能造成重大安全事故。

– 墨菲定律

几个案例

- 曾经，我把一个RockMongo暴露在了公网，而且没加密码。
- 是的，数据最后都没了。
- 但是是以你想不到的方式被删除的。
- 黑客没来，Google来了。

- Google很尽职地请求了每一条记录的 delete 方法。

几个案例

- 在进行AES加密的时候，是否都是用的以下写法：

```
public static void main(String[] args) {
    String key = "Bar12345Bar12345"; // 128 bit key
    String initVector = "RandomInitVector"; // 16 bytes IV

    System.out.println(decrypt(key, initVector,
            encrypt(key, initVector, "Hello World")));
}
```

- CBC加密中，若iv可预测，则该加密不安全。

几个案例

- 在编写接受验证码绑定手机的时候。
- 是不是只校验了验证码，而没有校验手机号是不是发送的号码。
- 是不是。
- 不是。
- 是

防人之心不可无，偷懒之心不可无。

– 总结一下

谢谢大家