


yang1982_0907的专栏

目录视图

摘要视图

RSS 订阅

个人资料



Bean_Young

访问：184996次

积分：2040

等级：BLOG<5

排名：第16454名

原创：33篇

转载：8篇

译文：2篇

评论：35条

文章搜索

文章分类

C/C++ (7)

Java (9)

Linux/Unix (28)

Windows系统 (0)

日语学习 (0)

数据库 (9)

网络技术 (8)

Zimbra (1)

Zabbix (1)

Android (0)

文章存档

2017年02月 (2)

2015年05月 (2)

2015年04月 (4)

2015年03月 (4)

2015年02月 (1)

展开

阅读排行

密码学中的“盐值 Salt”

2015-02-04 10:476044人阅读评论^v^分享

分类：Linux/Unix (27)

目录(?) [+]

原文链接：http://www.libuchao.com/2013/07/05/password-salt

为什么要在密码里加点“盐”

盐 (Salt)

在密码学中，是指通过在密码任意固定位置插入特定的字符串，让散列后的结果和使用原始密码的散列结果不相符，这种过程称之为“加盐”。

以上这句话是维基百科上对于 Salt 的定义，但是仅凭这句话还是很难理解什么叫 Salt，以及它究竟起到什么作用。

第一代密码

早期的软件系统或者互联网应用，数据库中设计用户表的时候，大致是这样的结构：

mysql> desc User;

+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| UserName | varchar(50) | NO | | | |
| Password | varchar(150) | NO | | | |
+-----+-----+-----+-----+-----+-----+

数据存储形式如下：

mysql> select * from User;

+-----+-----+
| UserName | Password |
+-----+-----+
| lichao | 123 |
| akasuna | 456 |
+-----+-----+

主要的关键字段就是这么两个，一个是登陆时的用户名，对应的一个密码，而且那个时候的用户名是明文存储的，如果你登陆时用户名是 123，那么数据库里存的就是 123。这种设计思路非常简单，但是缺陷也非常明显，数据库一旦泄露，那么所有用户名和密码都会泄露，后果非常严重。参见《CSDN 详解 600 万用户密码泄露始末》。

第二代密码

为了规避第一代密码设计的缺陷，聪明的人在数据库中不在存储明文密码，转而存储加密后的密码，典型的加密算法是 MD5 和 SHA1，其数据表大致是这样设计的：

mysql> desc User;

+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| UserName | varchar(50) | NO | | | |
| PwdHash | char(32) | NO | | | |
+-----+-----+-----+-----+-----+-----+

- MySQL集群搭建详解 (三)
- (24947)
- 详解CheckStyle的检查规则
- (18538)
- 详解FindBugs的各项检测
- (15781)
- MySQL的InnoDB缓存相关
- (12650)
- Spring读取properties文件
- (10157)
- RHEL 6.4 (i386) 安装手册
- (9839)
- usb_modeswitch使用详解
- (6693)
- 在CentOS 6.4中安装JDK
- (6068)
- 密码学中的“盐值 Salt”
- (6042)
- 在CentOS 6.4中编译安装
- (5964)

评论排行

- MySQL集群搭建详解 (三)
- (14)
- Spring读取properties文件
- (4)
- 堆排序——ANSI C实现
- (4)
- 密码学中的“盐值 Salt”
- (4)
- 为Android-x86编译tcpdump
- (3)
- MySQL集群的flexAsyncIO
- (2)
- RHEL 6.4 (i386) 安装手册
- (2)
- 如何使用Mondo救援故障
- (1)
- 快速排序（迭代）——AI
- (1)
- CentOS 6.4安装和配置S
- (1)

推荐文章

- * 【《Real-Time Rendering 3rd》提炼总结】(一) 全书知识点总览
- * CSDN日报20170409 ——《扯蛋的密码规则》
- * Shader2D: 一些2D效果的Shader实现
- * 一个屌丝程序员猿的人生（六十一）
- * 自定义控件三部曲视图篇（三）——瀑布流容器WaterFallLayout实现
- * 面向服务的体系架构（SOA）——架构篇

最新评论

- MySQL集群搭建详解（三种结点）
殷伟涛: 学习了，谢谢分享！
- 为Android-x86编译tcpdump工具
ye137812951: 爱死你了，感谢！超级需要这个~!
- 密码学中的“盐值 Salt”
ALLURENN: 逻辑很清晰，讲的很明白，赞赞赞
- MySQL集群搭建详解（三种结点）
Bean_Young: @qq_24649979: 连接VIP就行了，VIP下面挂着几个SQL结点的真实IP
- MySQL集群搭建详解（三种结点）
qq_24649979: 如果要用PHP连接SQL 那么我要连接那个IP呢 是SQL结点的IP,但是SQL结点有2个要连接那个...
- MySQL集群搭建详解（三种结点）
qq_24649979: 如果要用PHP连接SQL 那么我要连接那个IP呢 是SQL结点的IP,但是SQL结点有2个要连接那个...
- MySQL集群搭建详解（三种结点）
qq_24649979: 如果要用PHP连接SQL 那么我要连接那个IP呢 是

数据存储形式如下：

```
mysql> select * from User;
+-----+-----+-----+
| UserName | PwdHash |
+-----+-----+-----+
| lichao   | 202cb962ac59075b964b07152d234b70 |
| akasuna  | 250cf8b51c773f3f8dc8b4be867a9a02 |
+-----+-----+-----+
```

假如你设置的密码是 123，那么数据库中存储的就是 202cb962ac59075b964b07152d234b70 或 40bd001563085fc35165329ea1ff5c5ecbdbbeef。当用户登陆的时候，会把用户输入的密码执行 MD5（或者 SHA1）后再和数据库就行对比，判断用户身份是否合法，这种加密算法称为**散列**。

严格地说，这种算法不能算是加密，因为理论上来说，它不能被解密。所以即使数据库丢失了，但是数据库中的密码都是密文，根本无法判断用户的原始密码，所以后果也不算太严重。

第三代密码

本来第二代密码设计方法已经很不错了，只要你密码设置得稍微复杂一点，就几乎没有被破解的可能。如果你的密码设置得不够复杂，被破解出来的可能性还是比较大的。

好事者收集常用的密码，然后对他们执行 MD5 或者 SHA1，然后做成一个数据量非常庞大的数据字典，然后对泄露的数据库中的密码就行对比，如果你的原始密码很不幸的被包含在这个数据字典中，那么花不了多长时间就能把你的原始密码匹配出来。这个数据字典很容易收集，CSDN 泄露的那 600w 个密码，就是很好的原始素材。

于是，第三代密码设计方法诞生，用户表中多了一个字段：

```
mysql> desc User;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| UserName   | varchar(50)   | NO   |     |          |       |
| Salt       | char(50)      | NO   |     |          |       |
| PwdHash    | char(32)      | NO   |     |          |       |
+-----+-----+-----+-----+-----+-----+
```

数据存储形式如下：

```
mysql> select * from User;
+-----+-----+-----+-----+-----+
| UserName | Salt | PwdHash |
+-----+-----+-----+-----+-----+
| lichao   | 1ck12b13k1jmjxrg1h0129h2lj | 6c22ef52be70e11b6f3bcf0f672c96ce |
| akasuna  | 1h029kh2lj11jmjxrg13k1c12b | 7128f587d88d6686974d6ef57c193628 |
+-----+-----+-----+-----+-----+
```

Salt 可以是任意字母、数字、或是字母或数字的组合，但必须是随机产生的，每个用户的 Salt 都不一样，用户注册的时候，数据库存入的不是明文密码，也不是简单的对明文密码进行散列，而是 MD5(明文密码 + Salt)，也就是说：

```
MD5('123' + '1ck12b13k1jmjxrg1h0129h2lj') = '6c22ef52be70e11b6f3bcf0f672c96ce'
MD5('456' + '1h029kh2lj11jmjxrg13k1c12b') = '7128f587d88d6686974d6ef57c193628'
```

当用户登陆的时候，同样用这种算法就行验证。

由于加了 Salt，即便数据库泄露了，但是由于密码都是加了 Salt 之后的散列，坏人们的数据字典已经无法直接匹配，明文密码被破解出来的概率也大大降低。

是不是加了 Salt 之后就绝对安全了呢？淡然没有！坏人们还是可以从他们数据字典中的密码，加上我们泄露数据库中的 Salt，然后散列，然后再匹配。但是由于我们的 Salt 是随机产生的，假如我们的用户数据表中有 30w 条数据，数据字典中有 600w 条数据，坏人们如果想要完全覆盖的坏，他们加上 Salt 后再散列的数据字典数据量就应该是 300000* 6000000 = 1800000000000，一万八千亿啊，干坏事的成本太高了吧。但是如果只是想破解某个用户的密码的话，只需为这 600w 条数据加上 Salt，然后散列匹配。可见 Salt 虽然大大提高了安全系数，但也并非绝对安全。

实际项目中，Salt 不一定要加在最前面或最后面，也可以插在中间嘛，也可以分开插入，也可以倒序，程序设计时可以使灵活调整，都可以使破解的难度指数级增长。

SQL结点的IP,但是SQL结点有2个要连接那个...

Spring读取properties文件出现乱码_tsubasa_: 原来如此! 解决了大问题...转走了, 谢谢博主

Spring读取properties文件出现乱码_飞天猫熊: 厉害

密码学中的"盐值 Salt"
OPF_stere: 第一次了解盐值, 谢谢分享

PS, 文中所谓第一、二、三代密码的称呼, 是我自己 YY 的。

顶 4 踩 0

上一篇 CentOS 6.4中的cron计划任务配置方法

下一篇 Spring的事务传播性与隔离级别

我的同类文章

Linux/Unix (27)

• 从头构建Linux系统之一 ——...

2017-02-24

阅读 55

• usb_modeswitch使用详解 (...

2015-05-25

阅读 6690

• nginx配置文件解析

2015-04-20

阅读 271

• Zimbra协作套件 (v8.5) 产品..

2015-03-27

阅读 1482

• Install Zabbix 2.2.6 From So...

2015-01-15

阅读 665

• 通过screen解决终端模拟器断..

2017-02-24

阅读 53

• Jexus-5.6.3使用详解

2015-04-20

阅读 3997

• CentOS 6.5编译安装Mono-3...

2015-04-19

阅读 1230

• CentOS 6.4中的cron计划任...

2015-01-15

阅读 377


• Install Zabbix 2.2.6 From Pa...

2015-01-15

阅读 314


更多文章

参考知识库



MySQL知识库

21688 关注 | 1448 收录



算法与数据结构知识库

15396 关注 | 2320 收录

猜你在找

《C语言/C++学习指南》数据库篇(MySQL& sqlite)

密码学SHA1加密算法详解

在VC2015里学会使用MySQL数据库

《应用密码学》 欧几里得算法-Euclid

ssh+tomcat7.0+easyui+velocity+mysql数据库快速实战开

密码学程序之Column permutation列置换加密

MyCat实战讲解 (MySQL集群)


古典密码学上机实验

Oracle数据库


UVALive 4174 Steganography 字符串处理 密码学

查看评论


4楼 ALLURENN 2017-03-22 14:13发表

 逻辑很清晰, 讲的很明白, 赞赞赞


3楼 OPF_stere 2016-09-26 14:59发表

 第一次了解盐值, 谢谢分享

2楼 honghuzhilangzixin 2015-11-12 11:02发表

 讲解清晰, 不错。

1楼 xianxiandaoren 2015-11-03 17:47发表

 感谢, 原理说得挺清楚

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
- VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
- BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
- Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC
- coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
- Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
- Angular Cloud Foundry Redis Scala Django Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved

