MongoDB如何保护你的数据不受攻击

最近大约有26,000台MongoDB服务遭受到了黑客攻击，这些服务器大都是没有经过安全设计，服务器上的数据被清空而攻击者留言说：“我们已经备份了你的所有数据，如果你想恢复你的数据，请发送0.15个比特币，并通过邮件告诉我们你的服务器IP地址和付款信息。没有付款信息的邮件将被忽略“。截止到今天，0.15比特币价值877美元。而这些攻击者只是扫描MongoDB的默认端口27017，当发现了端口被使用时，只是简单的进行匿名访问，然后将脚本发送到服务器上进行数据库备份和删除。如此简单的攻击方法不能不说我们对数据安全的认知实在太低，我相信凡是有过数据被偷窃或者被攻击的企业都会面临严峻的问题，有些设计到客户隐私、公司机密文件、财务状况等核心数据。本文就以这个事件为引子为大家介绍一下MongoDB数据安全需要注意的问题和业内最佳实践经验。

# 你能区分 Authentication(认证) 和 Authorisation(授权）吗？

在我们讨论安全之前先来看一下经常混淆的两个概念，Authentication和Authorisation，这两个概念并不是MongoDB所独有的，在很多场合都可以使用，但是在介绍安全之间弄清楚他们的区别还是很有必要的。

Authentication：指的是认证一个用户是否合法，他回答的是“你是不是合法用户”的问题。例如，在登陆一个系统时，不论是输入用户名和密码的方式，还是通过第三方认证登录，Authentication的过程是认证改用户是否可以登录本系统。

Authorisation：指的是用户权限，与Authentication不同，他回答的是“你可以做什么“的问题。例如，登录系统以后，一般用户和管理员肯定有一些权限上的区别。此时，Authorisation用来检查用户可以访问的内容。

据一个常见的例子，如果你可以登录一台系统产看订单，但是不能删除数据，那么可以说你通过系统的Authentication过程，但是并不具备删除记录的Authorisation。

# Authentication（认证）

根据MongoDB版本不同支持的认证种类也有区别，下面列出了Community和Enterprise两种版本支持的不同认证类型。其中，SCRAM-SHA-1，MONGODB-CR和x.509是MongoDB内部支持的认证方式，而另外两个LDAP和Kerberos是借用第三方认证服务器进行Authentication。

|  | Community | Enterprise |
| --- | --- | --- |
| SCRAM-SHA-1 | ✔ | ✔ |
| MONGODB-CR | ✔ | ✔ |
| x.509 | ✔ | ✔ |
| LDAP |  | ✔ |
| Kerberos |  | ✔ |

表1: Authentication种类

由于LDAP和Kerberos只支持企业版并且他们不是MongoDB内部支持的认证类型，本文并不对这对这两种方式进行说明。

### SCRAM-SHA-1

这种是比较常见的Authentication方式，当你使用用户名密码登录MongoDB时，其实使用的就是SCRAM-SHA-1认证。创建一台需要SCRAM-SHA-1认证的MongoDB服务器非常简单，只需要在启动服务器的脚本中加上 —auth 参数，然后在第一次进行客户端连接时需要通过localhost网络地址登录，并创建一个合法账户。这个过程有一个需要注意的地方就是localhost异常，关于这方面内容请参考下面localhost异常一节。

基于这样的认证可以预防一下几种攻击：Eavesdropping，Replay，Database Compromise，Malicious Server。

### MONGODB-CR

这种认证方法在MongoDB3.0以后就被放弃了，所以本文就不在对他进行说明。仍然使用MongoDB2.X的用户可以查看MongoDB官方文档关于这种认证的解释。不过我还是建议大家升级MongoDB到3.X。

### x.509

MongoDB支持x.509认证证书允许用户通过TLS／SSL连接到服务器。关于TLS／SSL，PKI公钥原理和机制并不在本篇文章的范围内，有兴趣的读者可以查询一些相关的文章。需要指出的是，MongoDB的安装包包括支持SSL和不只是SSL两种，你要选择支持SSL的版本才能使用x.509协议。运行下面的命令可以查看你的mongo是否支持SSL，

**$ mongod --version**

**db version v3.4.0**

**git version: f4240c60f005be757399042dc12f6addbc3170c1**

**OpenSSL version: OpenSSL 0.9.8zh 14 Jan 2016**

**allocator: system**

**modules: none**

**build environment:**

**distarch: x86\_64**

**target\_arch: x86\_64**

在产品环境上，MongoDB应当使用授权机构颁发的证书来进行安全认证，在开发环境和测试环境里我们可以自行生成证书签名，需要注意的是，这种情况下连接MongoDB需要使用 —sslAllowInvalidCertificates 参数来允许非正式办法证书的连接。关于x.509的详细使用方法可以参考：https://docs.mongodb.com/manual/tutorial/configure-x509-client-authentication/

## Localhost 异常

当创建一台需要认证的MongoDB服务器后，第一次登录系统时需要创建一个用户。这是很容易出现Localhost异常错误。那么什么是Localhost异常呢。我们看一下下面的操作。

* 为了创建一台需要Authentication的MongoDB服务，我们在mongod后面加上—auth参数，如下：  
   # mongod —auth
* 服务器创建好以后，我们通过MongoDB Shell客户端连接到该服务器上  
   > mongo
* 连上以后我们并没有得到任何错误，因为当一台MongoDB第一次接收客户端连接时，需要允许客户端创建第一个用户，而这个第一次链接的客户端必须是经过Localhost网络接口连接。这样做的目的是保证没有系统以外的连接进来创建非法用户。一般情况只有系统管理员才能使用localhost网络接口。然后，如果你想查看一下当前系统的所有目录，  
   > show dbs  
  **2017-10-22T21:04:10.278+1100 E QUERY [main] Error: listDatabases failed:{**

**"ok" : 0,**

**"errmsg" : "not authorized on admin to execute command { listDatabases: 1.0 }",**

**"code" : 13,**

**"codeName" : "Unauthorized"**

**}**

这是你会得到上面的错误，这个错误的原因是服务器是需要Authentication后才能使用，任何没有认证的连接都不能执行相关操作。这种错误也被称之为Localhost异常。此时你能做的就是创建第一个用户。创建好以后，可以使用 db.auth() 命令对用户进行验证，验证通过后才能执行数据库的常规命令。

## Shard Cluster Authentication

当在Shard集群中进行验证时，情况有些不同。我们知道连接Shard集群需要经过mongos进程，而mongos并没有—auth参数可以为我们使用，这并不能说mongos没有认证机制。正是由于mongos需要连接多个Shard Cluster，所以他不能简单的通过—auth来设置验证模式，而是通过key文件在Shard Cluster内部进行认证通讯。

# Authorisation （授权）

前面介绍了MongoDB的认证机制，那么他的授权是如何工作的呢？MongoDB中，授权的方式是通过用户角色（role）来定义的，角色是对在一定数据库命名空间下，访问数据的权限（privilege）、操作（actions）的组合。

* actions: 是指用户可以对数据资源(resources)进行的操作，例如：添加、删除、修改、查询等。
* privilege: 是否允许用户对数据资源进行各种actions操作。
* resources: 这里数据资源指的是Database，Collection，Cluster等。

上面这种role，actions，privilege的授权方式被称为“基于角色的访问控制”（Role Based Access Control）。他带来的好处是MongoDB可以灵活的控制不同用户可以访问的数据资源，并根据不同操作赋予用户不同权限。听上去比较绕，我们看一下下面这幅图。

Create User

DBA

Insert Data

Developer

Read Data

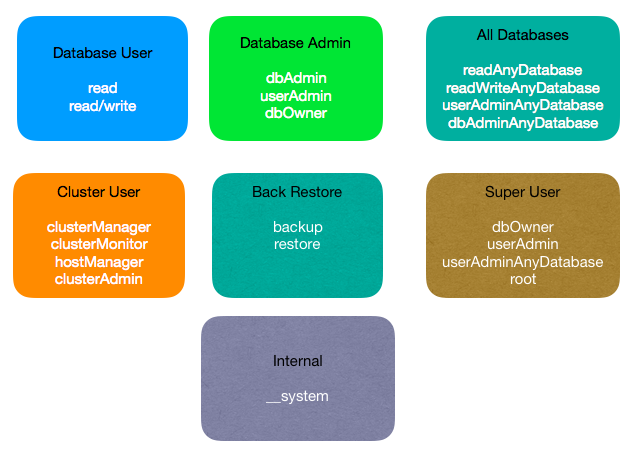
Reporter

图1: 角色种类

DBA可以向数据库添加、删除用户，但是他不需要对数据进行添加操作，这样的操作应该由开发人员进行；而负责统计数据库报表的人员只需要读取数据就可以。由于MongoDB设计了基于角色的访问控制，上面这样的描述场景实现起来非常容易，而且也有利于包括数据库中数据不会被其他人无意识的操作。

## MongoDB内置的角色

MongoDB定义了下面几种用户角色，每一种角色对应了不同的actions权限。基本上对于一般的应用程序，这些内置的角色可以满足大家的需求。



# 图2: 内置角色

# 关于内置角色的详细信息可以参考：https://docs.mongodb.com/manual/reference/built-in-roles/。

# 如果上面列出的这些角色都不满足项目需求，我们也可以自己定义适合我们的角色。下图中列出了在创建一个自定义角色时应当考虑的内容。

自定义角色：角色名称 + 数据库名称

继承自哪个角色

权限

action

action

资源

资源

图3: 自定义角色

首先需要确定自定义的角色应用在哪个数据库上，每个数据库都一个定义不同的用户角色。之后该角色继承自哪个内置角色，我们可以在内置角色基础上增加新的权限，就如同面向对象中子类继承父类那样。对后就是定义该角色的权限，每个权限对应着一对儿action和resource。关于如何创建一个自定义角色的命令我在这里就不做过多介绍了，MongoDB官方文档有非常详细的定义。这里我给大家介绍一款图形化操作权限的工具：dbKoda。下面是创建一个自定义角色的截图，有了图形化的工具，我们不需要记住各种命令的用法，在dbKoda中通过图形操作，我们可以看到对应的shell代码，非常快捷便利的帮助我们操作MongoDB。不得不说，dbKoda是我目前用过的最方便功能最强大的MongoDB IDE。

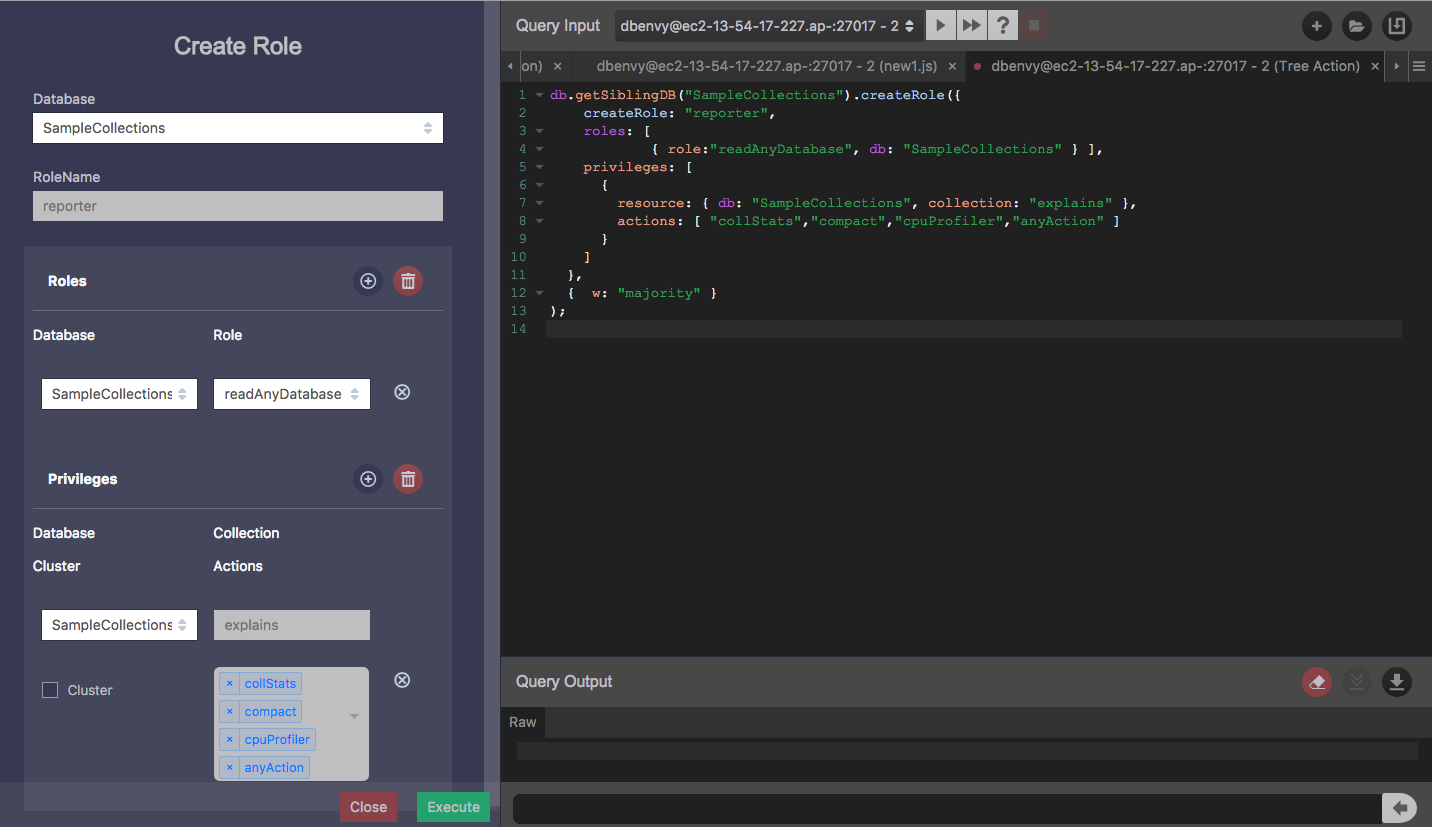


图4：dbKoda角色管理

# 加密

加密是MongoDB数据安全中重要的环节，MongoDB中的加密包括两种类型，Transport Encryption和Encryption at Rest。

## Transport Encryption

这种加密是指客户端通过MongoDB Driver链接到MongoDB服务器时，对网络传输过程中的数据进行加密操作。如下图：

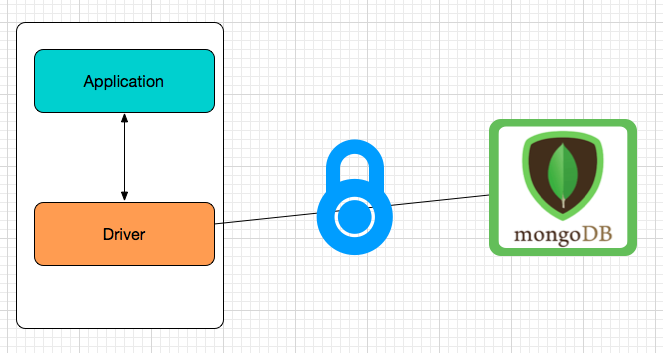


图5: Transport Encryption

在这种传输方式下，数据经过TLS／SSL加密， 机制和https类似。MongoDB采用openssl来处理SSL传输，如下图所示：

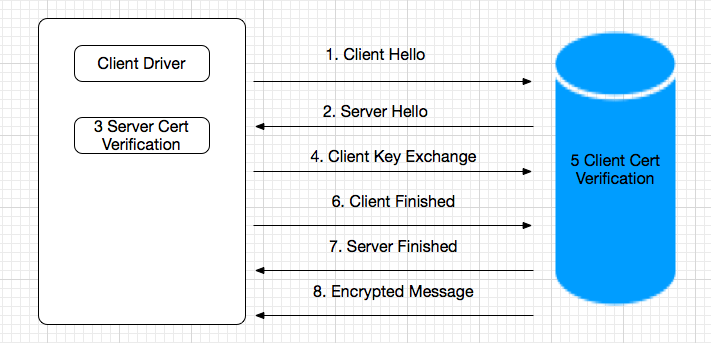


图6: TSL / SSL

MongoDB支持下面四种TSL传输类型：

* disabled  
  MongoDB服务器不需要加密操作。
* allowSSL  
  Replica Set之间的通讯不需要SSL，来自客户端的链接操作同时支持SSL和非SSL。
* preferSSL  
  与allowSSL类似，只是要求服务器之间的通讯使用SSL。
* requireSSL  
  服务器只允许SSL链接。

## Encryption at Rest

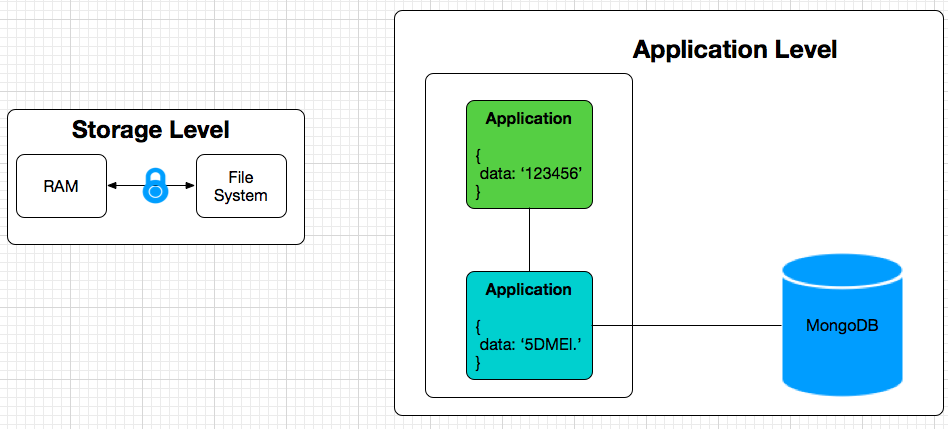
这种加密指MongoDB在磁盘上存储的数据进行加密。他包括两种级别的加密方法，一种是在存储磁盘上加密，另一种是应用程序级别上的加密。如下图所示。

图7: Encryption at Rest

### Storage Level

这种加密方式只在MongoDB企业版3.2以上才支持，并且需要使用WireTiger存储引擎。他的原理是数据在内存中是非加密格式，当MongoDB决定将数据写进磁盘时进行加密并将加密后的结构存磁盘。加密的方法可以采用外部文件方式，也可以与KMIP（Key Management Interoperability Protocol）集成。

### Application Level

这种方式是在应用程序层对数据进行加密，应用层使用数据时采用明文方式，在存进MongoDB前按照一定算法加密数据，并将加密后的数据存进数据库。这种方式比较简单不需要太多数据库层面上的支持。

# 如何保护你的数据

安全是每一个应用系统在开始设计的时候就需要考虑的因素，根据上面列出的安全措施，下面是我们要对一台MongoDB服务器进行安全设置的基本步骤，只有经过了这样的设置才能保证我们的数据不会受到非法攻击，从而减少企业损失。

* 根据上面提到的Authentication种类，选择合适的机制并强加到你的系统上。在启动MongoDB服务器时一定要—auth参数。
* 为每个用户设置角色（Role），不同的角色对应不一样的权限，不要轻易放开数据库写权限。例如，对于为报表浏览的用户而言，只读权限应该是足够的了。
* 服务器的网络连接尽量不要允许所有网络端口，只要客户端或者需要访问数据库的应用程序能够访问就可以。
* 监控数据库的运行状态和日常行为，MongoDB企业版自身包括系统检测功能，使用社区版的可以通过自行分析日志来做到监控。
* 尽量使用SSL连接，在SSL传输过程中，所有数据都会经过加密。
* 在MongoDB3.2以上版本，数据存储引擎采用WireTiger，他支持内置的加密支持，具体可以参考https://docs.mongodb.com/manual/core/security-encryption-at-rest/。
* 在启动MongoDB服务的时候不要使用系统管理员用户，尽量为MongoDB创建单独的用户，这样可以避免MongoDB对操作系统有过大的权限从而导致系统性的破坏。

# 如何察觉我的数据是否被攻击

这种情况要根据数据具体受到什么样的攻击来判断，这里我们根据目前MongoDB收到的事故中分析了一些常见的症状：

* 如果你执行了文章前面提到的方法，那么攻击者是没有办法进入到你的系统，只要你的证书或者用户名密码没有流失，基本上可以高枕无忧。
* 定期查看你的Database和Collection，在最近的攻击事故中，攻击者大多是运行删除服务器上的数据库或者Collection，然后将删除的数据库替换成随机数据。
* 如果数据库有访问控制，可以通过系统日志查看可疑的操作和活动。

# 数据被攻击以后应当怎么办

如果你的数据库没有进行安全设置，在非安全的环境下运行，不幸的是你的数据已经被攻击，那么下面的步骤可以减少你的损失：

* 首先要做的是在数据库服务器上加上安全设置，这样可以避免更多的攻击。
* 对于系统中存在的用户，确保他们的权限没有被更改，用户没有被删除，也没有新增加的用户。
* 检查系统日志，查看有没有类似drop操作，如果有记录下时间点。
* 如果你有定期对数据进行备份，可以根据被攻击的时间点恢复数据，但是需要确定在上一次有效备份的时间和被攻击的时间之间有哪些需要手动恢复的操作。
* 如果你没有备份数据，那么很不幸，你的数据很可能无法再恢复了。增加备份策略并实施安全设置是接下来要做的必要步骤。

# 小结

上面提到了很多关于如何加强MongoDB安全的措施和方法，在我经历过的各种企业产品设计过程中，很多安全问题都是由于开发设计人员对安全不够重视，为了开发方便忽视了安全上的考虑，采用明文传输数据并且匿名访问数据库。很多重量级的企业应用居然没有考虑权限设计。在这样的情况下为攻击者 提供了便利的攻击手段。希望此文可以给从事MongoDB设计开发的团队一些指导，在具体使用方面可以参考下面列出的文献，里面详细介绍了各种方法的操作过程和命令语法。

# 参考文献

<https://docs.mongodb.com/manual/tutorial/configure-x509-client-authentication/>

<https://docs.mongodb.com/manual/core/security-encryption-at-rest/>

# <https://docs.mongodb.com/manual/reference/built-in-roles/>

# https://docs.mongodb.com/manual/reference/built-in-roles/