**《数据库实训》 课程报告（11）**

**(2018-2019 学年第2 学期)**

**CryptDB攻击方案**

**提交日期： 2019 年 8 月 24 日**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学 院** | **软件学院** | | **专业班级** | **软件工程** | |
| **课程名称** | **数据库实训** | | **任课教师** | **曾兵** | |
| **人数** | **3** | |  |  | |
| **学 号** | | **学生姓名** | | | **分 数** |
| **201730681303** | | **成子谦** | | |  |
| **201730681112** | | **陈潮宇** | | |  |
| **201730682454** | | **林嘉轩** | | |  |
|  | |  | | |  |
|  | |  | | |  |
|  | |  | | |  |
|  | |  | | |  |
| **教师评语：**  **教师签名：** | | | | | |

CryptDB漏洞攻击方案

作者：成子谦、陈潮宇、林嘉轩

目录

[[摘要] 2](#_Toc17512681)

[第一部分 原理 3](#_Toc17512682)

[1.1加密解释 3](#_Toc17512683)

[1.2 CryptDB实现原理 4](#_Toc17512684)

[1.2.2 CryptDB的解决方案 4](#_Toc17512685)

[1.3漏洞1 5](#_Toc17512686)

[1.3.1.漏洞原理 5](#_Toc17512687)

[1.3.2 辅助数据 5](#_Toc17512688)

[1.3.3 攻击方案 5](#_Toc17512689)

[1.4漏洞2 8](#_Toc17512690)

[1.4.1漏洞原理 8](#_Toc17512691)

[1.4.2 攻击方案 8](#_Toc17512692)

[第二部分 漏洞利用演示系统 9](#_Toc17512693)

[需求分析 9](#_Toc17512694)

[分析问题 9](#_Toc17512695)

[详细设计 9](#_Toc17512696)

[小组讨论纪要 17](#_Toc17512697)

[个人总结 19](#_Toc17512698)

[项目进度安排等 20](#_Toc17512699)

[摘要]

本文旨在阐述CrypyDB的原理，针对CryptDB的两种漏洞，给出可行的攻击方案。CryptDB是操作加密数据的一个典范，CryptDB旨在解决两种加密数据库的漏洞。本文会详细阐述其漏洞原理，并且给出攻击方案。

第一部分 原理

1.1加密解释

加密。 对称加密方案SKE =（Gen，Enc，Dec）是三种算法的元组，其工作方式如下。 Gen将安全参数作为输入并返回密钥K; Enc将密钥K和消息m作为输入并返回密文c; 和Dec将密钥K和密文c作为输入并返回消息m。加密安全性的标准概念是针对选择明文攻击（CPA）的安全性。 我们将读者引用[29]来详细描述这一概念。 在这里，我们只提到对于对称密钥加密众所周知的东西，只有当Enc有状态或随机化时才能实现CPA安全性。

确定性加密。 对称DTE方案DTE =（Gen，Enc，Dec）是对称加密方案，Enc不是随机化的; 也就是说，每个消息m由Enc映射到密钥K下的单个密文。

顺序保持加密。 对称OPE方案OPE =（Gen，Enc，Dec）是具有以下属性的对称加密方案：如果m1> m2，则EncK（m1）> EncK（m2）; 如果m1 = m2则EncK（m1）= EncK（m2）; 如果m1 <m2则EncK（m1）<EncK（m2）。

加性同态加密。一种对称加性同态加密（AHE）方案。AHE =（Gen，Enc，Dec）是一种对称加密方案，具有以下附加属性：DecK EncK（m1）⊗EncK（m2）= m1 + m2，其中⊗是 在AHE的密文空间上操作而不一定是加法。

连接加密。 CryptDB系统支持两种连接操作：等值连接和范围连接。 使用EJOIN =（Gen，Enc，Dec）方案支持等值连接，它是DTE和散列的组合。 使用基于OPE的加密方案RJOIN =（Gen，Enc，Dec）支持范围连接。 我们注意到，在连接查询（任何一种）之后，两个连接的列在相同的秘钥下保持加密状态。

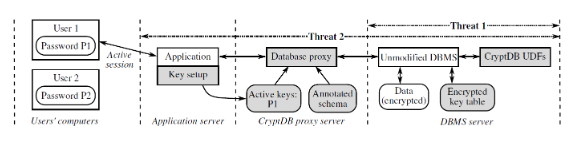
可搜索的加密。 该系统也可以使用用于关键字搜索操作的可搜索加密方案SRCH =（Gen，Enc，Token，Dec）。

洋葱加密。 Popa等人使用术语洋葱来指代加密方案的组成。 例如，给定两个加密方案SKE1 =（Gen1，Enc1，Dec1）和SKE2 =（Gen2，Enc2，Dec2）SKE1◦SKE2

消息m的加密定义为

1.2 CryptDB实现原理

我们首先了解一下CryptDB的结构。



CryptDB包含两部分：一个数据库代理和一个未经修改的DBMS。CryptDB使用用户定义的函数(UDF)在DBMS中进行加密操作。矩形和圆角矩形代表过程和数据，阴影代表CryptDB添加的组件。虚线隔离了用户计算机、应用程序服务器以及运行CryptDB数据库代理和DBMS的服务器。

CryptDB被设计以应对两种威胁(见图)：

* Thread 1：具有高权限的管理员在数据库数据未经加密时，可能试图获得或泄露数据库数据
* Thread 2：入侵者利用软件漏洞攻击软件及访问数据库数据

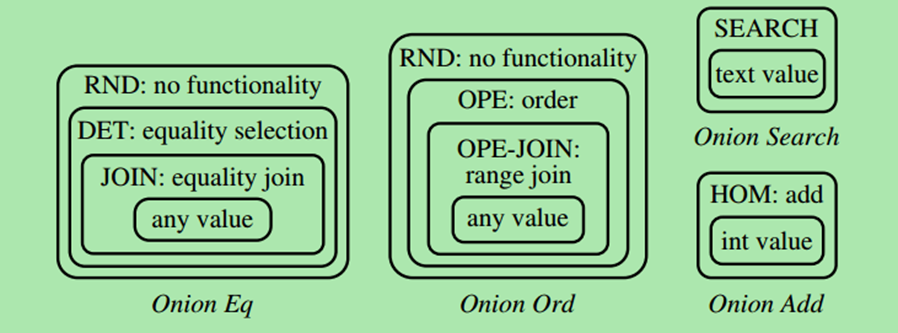
1.2.2 CryptDB的解决方案

第一类威胁危险性没有第二类威胁大。攻击者一般只获取数据信息，但没有能力去修改数据。

对于第一类威胁，CryptDB的做法是：利用数据库代理(在CryptDB中，称为mysql-proxy)截取所有传入的SQL语句并对语句中的关键字段进行加密，同时确保符合SQL语句的语法要求，然后再将加密后的SQL请求发送给mysql-server；mysql-server负责处理SQL语句，并返回加密的处理结果给mysql-proxy；最后返回的处理结果在mysql-proxy处解密，返回给客户端。CryptDB的存在，使得数据库管理员在没有解密密钥的情况下，无法获知加密之后的数据。CryptDB的效率很高，因为它主要使用的是对称密钥加密，避免完全全同态加密。

对于第二类威胁，CryptDB的保密制度指定了链式加密的规则：将加密密钥和用户密码捆绑。而且CryptDB使用不同的密钥加密不同的数据项。当解密数据库中特定的某个数据项时，需要相关用户密码中所构造的一连串密钥。这使得数据项只有使用相应的用户的密码登录才可以进行解密。这种做法使得即使服务器被攻破，整个系统被入侵者控制的情况下，只要用户没有登录，攻击者也无法解密用户的数据。

CryptDB还采用了一种基于语句的加密方式：洋葱模型。这种模型使得每个洋葱之间存储着多种加密方式加密后的数据，避免开销大的重加密操作。



1.3漏洞1

1.3.1.漏洞原理

虽然CryptDB有多层加密，使得攻击者很难轻而易举地窃取数据。但正是由于其洋葱模型，使得其安全性会随着查询语句而动态变化。由于CryptDB的DTE层使用的是确定性加密，即相同明文会映射成相同的密文，对于这种加密，目前的攻击中，最常见和有效的方案之一便是推理攻击。而推理攻击又有多种方案，包括频率分析和优化等。

1.3.2 辅助数据

我们考虑以下辅助信息来源：

•应用程序详细信息：运行在加密数据库之上的应用程序，可能通过访问应用程序（例如，如果它是Web服务）或从文档获得;

•公共统计：公开可用的统计数据，例如人口普查数据或医院统计数据;

•先前版本：数据库的先前版本，可能通过先前的数据泄露获得。

我们强调，我们的实验将使用不同的辅助源子集，并且没有任何攻击需要访问到所有的源。

1.3.3 攻击方案

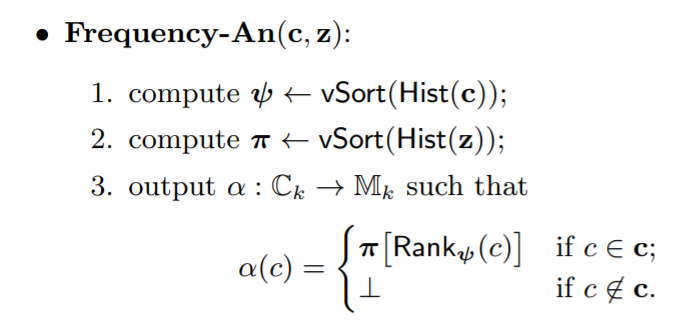
Naveed[2]和微软在2015年提出了CryptDB的一种攻击方案。在这种攻击方案中，我们假设在一个电子医疗记录的场景中，使用来自200多家美国医院的病人的真实数据经验性地分析了这些攻击。当加密数据库在稳态下操作，在这种状态下足够多的加密层已经被剥开来允许应用执行查询，实验的结果表明数量惊人的敏感信息可以被恢复。

我们研究了具体的对抗基于EDBs的CryptDB设计的推理攻击。这些系统用不同加密方案的层来加密每个DB列。当收到查询时，这些系统逐层解密，直到到达支持必要操作的层。具体说，这意味着支持范围或者相等查询的列分别地被解密到OPE和DTE层，而这就是我们攻击的切入点。

我们考虑推理攻击，它将OPE或DTE层加密的列和一个辅助的公共数据集作为输入，然后返回一个从密文到明文的映射。推理攻击有多种方案，现介绍以下4种：

1. 频率分析

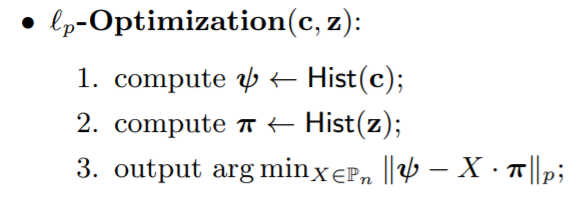
频率分析是最基本和众所周知的推理攻击。 它发展于9世纪，用于打破经典密码。 众所周知，频率分析可以破坏确定性加密，特别是确定性加密的列。 给定上的DTE加密列和上的辅助数据集，攻击通过将第i个出现频率的c元素分配给z的第i个出现频率元素来工作。 为了便于说明，我们假设和具有可以严格排序的直方图; 也就是说，对于所有且，其中且。 更准确地说，攻击定义为：



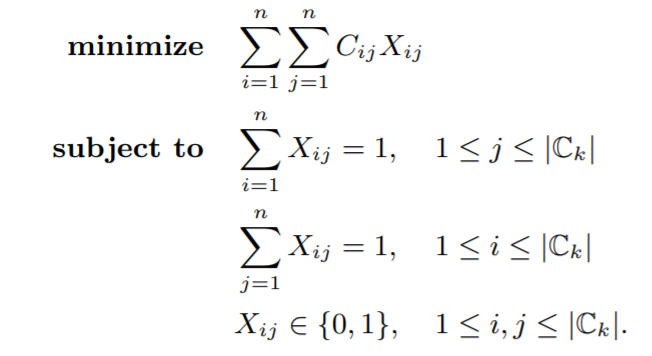
2）优化

我们现在描述针对DTE加密列的一系列攻击，我们将其称为“优化”。该家族由规范参数化。基本思想是找到从密文到明文的赋值，它最小化给定的成本函数，这里选择的成本函数是数据集直方图之间的距离。这具有使所有明文/密文对中的频率的总不匹配最小化的效果。攻击的工作原理如下。

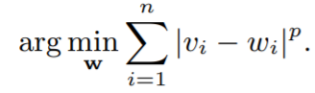
给定Ck上的DTE加密列c和Mk上的辅助信息z，攻击者首先分别计算和的直方图ψ和π。然后，它找到最小化密文直方图ψ和置换辅助直方图X·π之间的距离的置换矩阵X.直观地，攻击发现明文到密文的映射实现了他们的采样频率的最接近的整体匹配。请注意，这与频率分析非常不同，频率分析忽略了频率的幅度，只考虑了它们的等级。更确切地说，攻击定义如下：



其中是置换矩阵的集合。注意，在“优化攻击”中，步骤3可以表示为线性和分配问题(LSAP)。LSAP可以使用著名的匈牙利算法或任何线性规划(LP)求解器有效地求解。在我们的实验中，我们使用了时间为的前一种方法。精确的LSAP公式是



在这里成本矩阵给出了从明文j到密文i的匹配代价。对于p = 1,成本只是绝对频率的差异,所以我们设置。然而，对于，优化攻击的步骤3不能直接表示为LSAP，因为范数不是一个简单的线性和。尽管如此，我们证明使用快速LSAP求解器仍然可以有效地解决这个问题。为了知道原理，设是函数，设是函数。然后我们注意到向量的范数可以写成

由于是单调递增的，使最小的向量就是使最小的向量。因此，对于任意向量v，距离v 最小距离的向量w是的解。

只要，这个优化问题可以用代价矩阵C表示为LSAP，使得Cij = 。攻击取时间。

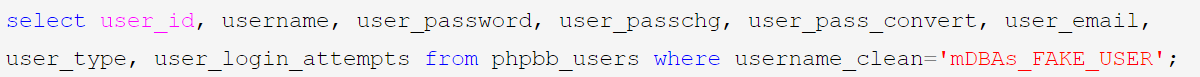
1.4漏洞2

1.4.1漏洞原理

由于CryptDB只是对于数据库的安全性进行保护，即对于数据进行不同层次的加密。但没有考虑数据完整性和真实性的约束。因此在特定情况下，即数据库代理和服务器都落入敌手的时，虽然CryptDB通过密钥链保证了未登陆用户数据的安全性，但还是有漏洞可寻。

1.4.2 攻击方案

考虑一个在线论坛。站点管理员通过将启用CryptDB的数据库移动到廉价的云服务器来降低运营成本。CryptDB的代理服务器和应用程序服务器都运行在论坛管理员严格保护的物理服务器上。要获得对应用程序服务器的访问权限，mDBA首先以普通用户身份注册到网站。mDBA的第一个目标是识别phpbb\_users表中的username\_clean字段，但此时表名被加密。为了获得正确的表，mDBA启用MySQL的查询日志来验证表名。 完成这些步骤后，mDBA将登录到网站。 为了验证用户登录尝试，phpBB执行SQL查询，例如：



显然，代理服务器必须事先显示username\_clean的洋葱的RND层才能执行查询操作。mDBA会事先在事件日志中检查读取查询，通过检查数据库日志的账户登录时间，mDBA可以识别出查询语句并得到其创建的账户用户名的加密版本，从而跟踪其创建的账户在数据库中的所有操作。而且，在当前版本的CryptDB中，创建表的时候，列名称顺序被保留了下来，由于phpBB和CryptDB的代码是开源的，这使得被加密的列名称和原来的列名称可以被一一匹配。此时，mDBA可以篡改存储在服务器上的CryptDB数据库里的表，并在mDBA创建的用户的帮助下，通过不同途经来危及CryptDB的安全：

找到特定目标。在phpBB中，用户可以在论坛上看到其他用户的用户名。对于特定目标，攻击者尝试使用目标名称登录几次，由于没有密码所以会失败。但在数据库中，攻击者会留意到对应的登录查询。通过这个查询，攻击者可以定位特定用户在phpbb\_users表里的位置。

收集用户信息。mDBA把其他用户的加密字段复制到虚假账户的相应记录字段中，然后通过phpBB网站的账户信息页面来显示该虚假账户所包含的其他用户的信息。

账户劫持。mDBA可以复制其虚假账户的信息以覆盖其他用户的信息，比如账户密码等，然后登录其他用户的账号。这使得mDBA可以把虚假账户中的信息覆盖所有用户，然后登录特定用户并恢复其他用户的信息，以实现劫持特定用户的账号。

权限升级。典型的phpBB中，user表的第二行属于administrator管理。mDBA可以使用此信息将其虚假账户的权限级别提升为管理员权限级别。攻击者一旦成为管理员，就可以通过调整previalges来访问受限区域。

第二部分 漏洞利用演示系统

需求分析

针对上述漏洞，我们设计了一个漏洞攻击程序，以mDBA的角色，旨在实现针对第一种漏洞的频率分析攻击方案。

分析问题

为了实现频率分析，我们首先需要扮演mDBA的角色，我们在一台装有ubuntu12.04的笔记本上，安装了mysql sever和cryptDB。使用两个端口模拟用户插入数据和mDBA，其中，3307端口用于模拟普通用户插入数据，3306端口用于模拟mDBA获取密文。

首先，我们模拟了数据集，假设一个登陆场景，在数据库中有datatbase health，并且有表user\_record，regist\_record等。在user\_record表中有user\_id, user\_name, password, info, type1, type2字段，在regist\_record 中有类似字段。

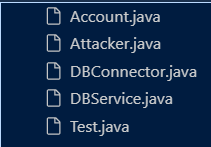
然后，用jdbc连接mysql数据库，并登陆3306端口进行查询操作，用java程序对返回的密文结果进行频率分析。

频率分析基于这样一个事实：CryptDB在执行select操作时，会将最外层的RND层剥去，即露出其内的DET层，而DET层使用确定性加密模式。在这种模式下，相同明文会被映射成相同的密文。

我们对所有表中的所有列都执行频率分析，在这里，我们首先需要一点辅助数据，这里假设我们要攻击的信息为疾病种类，这种信息在各大医院的网站上一般是公开的，但其患者人数或其他信息往往是隐私。利用辅助信息，我们对于要分析的列有了一个期望值。接着，分析并找出各列中符合此期望值的，将它们加入到候选列中。之后，若有辅助信息可以指明数据的分布，即预期此列数据会呈何种分布，则可以通过比较预期分布与候选列的分布直接的误差，选出最可能的一列。通过此方法，可以逐步还原表中的很大一部分列。

详细设计

1. 系统组成



1. 各部分功能实现

创建Account数据

import java.util.Random;

public class Account {

private static int numOfAccounts = 0;

protected String username, password, info;

protected int id, type;

public Account(int \_id, String \_username, String \_password, String \_info, int \_type) {

id = \_id;

username = \_username;

password = \_password;

info = \_info;

type = \_type;

}

public static int get\_account\_id() {

return ++numOfAccounts;

}

// define 4 types of account

public static int get\_type() {

Random random = new Random();

int a=random.nextInt(20);

if(a<3)return 0;

else if(a<8)return 1;

else if(a<13)return 2;

else return 3;

}

// define 4 types of account

public static int get\_type2() {

Random random = new Random();

int a=random.nextInt(20);

if(a<5)return 0;

else if(a<10)return 1;

else if(a<15)return 2;

else return 3;

}

}

攻击代码

public static void launch() throws SQLException {

init();

//expected distribution from public data

double[] distributionWeight={0.15,0.25,0.25,0.35};

// find specific column

String findAll = "select \* from user\_data";

final ResultSet resultSet = dbConnector.execute\_query(findAll);

ResultSetMetaData rsmd = resultSet.getMetaData();

final int rsmdColumnCount = rsmd.getColumnCount();

//found column

ArrayList<Integer> candidateColumn=new ArrayList<Integer>();

for (int i = 1; i <= rsmdColumnCount; i++) {

// get the name of each column

String name = rsmd.getColumnName(i);

columnName.addElement(name);

}

for (int i = 0; i < columnName.size(); i++) {

Boolean findColumn = true;

set.clear();

ResultSet tmp = resultSet;

while (tmp.next()) {

String currElement = tmp.getString(columnName.elementAt(i));

set.add(currElement);

if (set.size() > 4) {

findColumn = false;

break;

}

}

if (findColumn == true) {

//System.out.println("The specific column is: " + columnName.elementAt(i));

candidateColumn.add(i);

}

}

if(candidateColumn.size()==1){

System.out.println("The specific column is: " + columnName.elementAt(candidateColumn.get(0)));

}

else{

int totalRow=resultSet.getRow();

ResultSet tmp = resultSet;

ArrayList<Double> deviationList= new ArrayList<Double>();

for(int i=0;i<candidateColumn.size();i++){

int columnIndex=candidateColumn.get(i);

ArrayList<Integer> count=new ArrayList<Integer>();

ArrayList<String> columnAttributeSet=new ArrayList<String>();

while(tmp.next()){

String currElement = tmp.getString(columnName.elementAt(columnIndex));

Boolean isNew=true;

for(int j=0;j<columnAttributeSet.size();j++){

if(columnAttributeSet.get(j)==currElement){

count.set(j,count.get(j)+1);

isNew=false;

break;

}

}

if(isNew){

columnAttributeSet.add(currElement);

count.add(0);

isNew=false;

}

}

double deviation=0;

for(int j=0;j<count.size();j++){

double a=count.get(j)/totalRow;

double b=(a-distributionWeight[j])\*(a-distributionWeight[j]);

deviation+=b;

}

deviationList.add(deviation);

}

int minIndex=findMinIndex(deviationList);

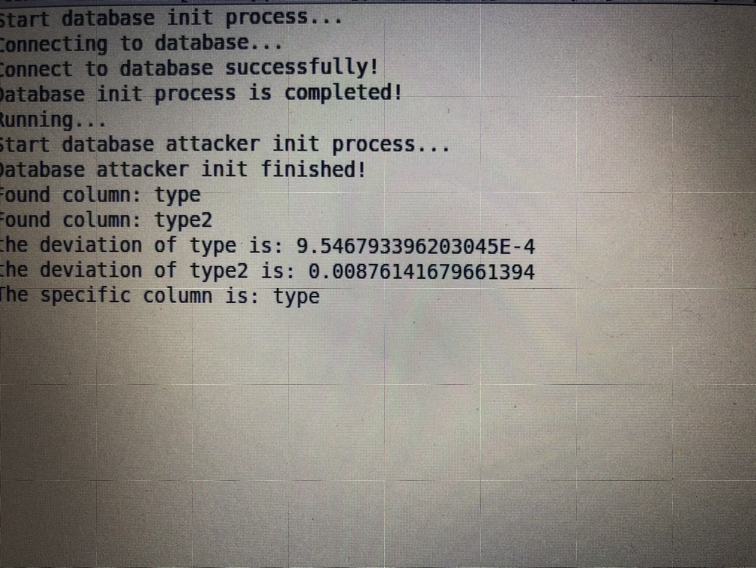
System.out.println("The specific column is: " + columnName.elementAt(candidateColumn.get(minIndex)));

}

}

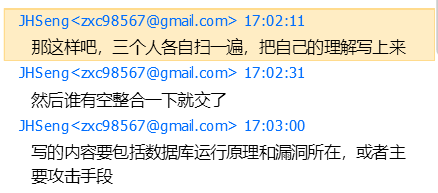
其他部分详见<https://github.com/JHSeng/CryptDB_Attack>

3）软件测试

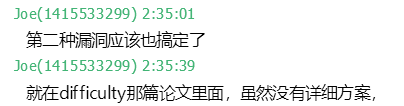


**小组讨论纪要**











**个人总结**

成子谦：这次实训让我对密码学及加密数据库有了一个初步的认识，明白了很多保全手段基于一定的数学原理，但所谓的“安全”并非绝对安全，往往有很多不为人知的漏洞。我们从理论上对数据库进行攻击，也为提高数据库安全性作出贡献。

陈潮宇：本次实训让我对密码学有了一些简单的认识和理解，对于加密数据库的功能和原理有了一定的了解。在本次实训中，我主要负责论文研读和翻译，以及部分代码的编写。在读论文过程中，对破解加密数据库用到的数学知识感到震撼，一方面，从宏观实现上看，不会想到会用到如此晦涩的数学知识，但另一方面，正是一条条公式让整个文章更加严谨，对那些为数据库安全做出贡献的学者表示敬佩。

林嘉轩：在这次实训中，我了解到了cryptdb的原理，增加了对数据库管理系统的认识，从对英语论文得阅读中了解到了很多课外知识，也知道对于数据库管理系统来说，危险存在于什么方面。如果将来需要编写数据库管理系统，就能从各个方面考虑问题。而且，阅读英语论文更能够巩固和提升我的英语水平，一方面增长我的数据库知识，另一方面提升英语阅读水平，实为一举两得

**项目进度安排**

7月15日-7月25日 安装CryptDB，通过阅读论文理解CryptDB原理及其存在的漏洞

7月16日-8月10日 基本实现攻击第一种漏洞

8月11日-8月20日 基本实现攻击第二种漏洞

8月21日-8月23日 完成结题报告，测试完善攻击方案