12-2013

Forensic Investigation of MySQL Database Management System

Andrew C. Lawrence

University of Arkansas, Fayetteville

Follow this and additional works at: http://scholarworks.uark.edu/csceuht

Part of the Software Engineering Commons

Recommended Citation

http://scholarworks.uark.edu/csceuht/8

This Thesis is brought to you for free and open access by the Computer Science and Computer Engineering at ScholarWorks@UARK. It has been accepted for inclusion in Computer Science and Computer Engineering Undergraduate Honors Theses by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu, ccmiddle@uark.edu.
FORENSIC INVESTIGATION OF
MYSQL DATABASE MANAGEMENT SYSTEM

An undergraduate Honors College Thesis

in the

Department of Computer Science and Computer Engineering

College of Engineering
University of Arkansas
Fayetteville, AR

By

Andrew Lawrence
ABSTRACT

For various reasons, circumstances might arise in which an investigator, enlisting the help of a system administrator, needs access to an instance of MySQL that is password protected by an individual system user. If this password is unknown and the user is uncooperative or unavailable, alternative means must be utilized to gain access to the data stored within the program. Two main approaches will be explored, each with its pros and cons. In one case, the password can be bypassed entirely, granting the investigator unfettered access to the program data. The second method allows a narrower look at only some of the data. Either method will give the investigator, at least to some extent, the opportunity to overcome the problem of a missing or uncooperative system user.
This thesis is approved.

Thesis Advisor:

Brajendra Panda

Dr. Brajendra Panda

Thesis Committee:

Dale R. Thompson

Dr. Dale R. Thompson

Craig W. Thompson

Dr. Craig W. Thompson
# TABLE OF CONTENTS

1. Introduction .................................................................................................................. 1  
   1.1 Problem .................................................................................................................. 1  
   1.2 Objective .................................................................................................................. 1  
   1.3 Approach .................................................................................................................. 1  
   1.4 Organization of this Thesis ...................................................................................... 2  

2. Background .................................................................................................................. 3  
   2.1 Permission Levels ................................................................................................... 3  
   2.2 MySQL Security ..................................................................................................... 3  

3. Approach ..................................................................................................................... 5  
   3.1 Two Approaches ..................................................................................................... 5  
   3.2 The Copy Method .................................................................................................... 7  
      3.2.1 Procedure of the Copy Method .......................................................................... 7  
      3.2.2 Results of the Copy Method ............................................................................ 16  
   3.3 The Plainview Method .............................................................................................. 19  
      3.3.1 Procedure of the Plainview Method .................................................................. 19  
      3.3.2 Results of the Plainview Method ..................................................................... 20  

4. Conclusion .................................................................................................................... 23  
   4.1 Summary .................................................................................................................. 23  
   4.2 Contributions .......................................................................................................... 23  
   4.2 Future Work ............................................................................................................ 24  

5. References .................................................................................................................... 25
LIST OF FIGURES

Figure 1: Partial display of my.cnf file showing that the data directory (datadir) is /var/lib/mysql ................................................................. 6

Figure 2: Description of the Boarders table in database Dogs ........................................... 8

Figure 3: Data entries for the Boarders table in database Dogs .......................................... 9

Figure 4: Data entries for the Strays table in database Dogs prior to the UPDATE command ........................................................................................................... 9

Figure 5: Data entries for the Strays table in database Dogs after the UPDATE command .............................................................................................................................. 10

Figure 6: Data entries for the Boarders table in database Cats ......................................... 10

Figure 7: Data entry for the Strays table in database Cats .................................................. 10

Figure 8: File structure before data creation ........................................................................... 11

Figure 9: File structure after data creation ............................................................................. 11

Figure 10: Files created within the Dogs database ................................................................ 12

Figure 11: Files created within the Cats database ................................................................ 12

Figure 12: MySQL output showing the data stored on the source machine ......................... 17

Figure 13: MySQL output showing the data that has been transferred to the clean machine ......................................................................................................................... 18

Figure 14: The output of Strays.frm from the Dogs database ............................................... 19

Figure 15: Part of the output from the ibdata1 file that pertains to the Strays table in database Dogs .................................................................................................................. 20

Figure 16: The data entries for the Strays table of the Dogs database after the UPDATE command was executed .................................................................................................... 22
LIST OF TABLES

Table 1: The encodings for various known Date entries ........................................21
1. INTRODUCTION

1.1 Problem

Since MySQL employs a password to protect the information stored by the user, problems can arise if said user forgets the password, becomes unavailable, or uses the program for nefarious purposes and thus becomes uncooperative in providing the password to his or her superiors for investigation. As such, it is prudent to search for ways in which a user with higher system permissions can bypass the password. This would allow for an investigator to engage in his or her investigation or data recovery effort without the need to elicit the cooperation of the user.

1.2 Objective

In response to the aforementioned problem, two methods will be explored that allow an investigator to bypass the password requirement imposed by lower level system users running the database management system MySQL.

1.3 Approach

Two main approaches will be explored in this thesis. Either way will result in access to at least some of the stored data as well as a history file that shows the command history of MySQL. The first method requires that the program files be copied into a new instance of MySQL on another machine (or virtual machine) where the password is known. This will reveal the databases and their entries exactly as they would appear should the password be entered on the original instance of MySQL.
The second technique involves looking at the raw program data files and sifting through the encoding to obtain information of interest. This method reveals table structure and some of the entry data. However, it does not provide a complete view of the database as not all data types are encoded in the program data files in a manner easily readable by unaided human eyes.

1.4 Organization of this Thesis

Chapter 2 covers background information concerning different user roles in a computer system as well as the security applied by the MySQL program itself. Chapter 3 describes in detail the two methods by which an investigator can bypass the password prompt in MySQL and demonstrates that these methods are effective. Chapter 4 provides a concluding overview of the thesis.
2. BACKGROUND

2.1 Permission Levels

When dealing with computer security, one should realize that there are different levels of access. An individual user typically has access only to his or her files on the computer and often is not allowed to access system files or make substantial changes to the system. This responsibility and privilege falls to the system administrator who has a “master key” of sorts in the form of an administrative password. This password grants the administrator access to nearly every aspect of the system, including both user and system files. In some ways, this makes sense. The system administrator is typically employed by the entity that owns the system. As such, the administrator should have full access to all files on the system to perform duties ranging from simple maintenance to investigations to intrusion detection and everything in between. On the other hand, this lends itself to a clear lack of privacy on the part of the individual user. Though the user has a password for his or her account and may even have a password to access certain programs or files within that account, the system administrator can often bypass such safeguards with the aforementioned “master key.” Such a situation arises with one of the most widely used open-source relational database management systems – MySQL.\[1\]

2.2 MySQL Security

Upon initial setup, MySQL prompts the user to set a password that is intended to prevent unauthorized access to the program or, more accurately, to the information stored by the program. Failure to enter the correct password results in “access denied” errors that prevent the user from accessing the databases or the information stored within them.
This is good in order to promote security and confidentiality, but problems arise should the password be forgotten or the password holder hide unscrupulous behavior within the MySQL program and refuse to divulge the password upon questioning. If there is a way to bypass the password requirement, then the data can be recovered from either scenario, but it could also be recovered, potentially, by unauthorized persons.

Some systems, such as those used in this thesis, require system administrator privileges in order to install MySQL. As such, the system administrator could set the password for MySQL. However, this thesis rests on the idea that the system user is allowed to set the password.
3. APPROACH

3.1 Two Approaches

Investigation into the MySQL database reveals that there are at least two ways to recover the password-protected information, and the two known ways will be discussed in the following pages of this thesis. The first way, henceforth referred to as the “Copy Method,” involves invoking administrator access in order to manually copy program data files, while the second requires sifting through these files in a command line environment. The Copy Method is preferred as it produces the information in an easily readable format and appears just as it would if the password to the program were known. However, the second way, henceforth termed the “Plainview Method,” has the potential to show entries that have been deleted from the table, allowing for historical look at the data. On the other hand, certain data types are not readable, or at the very least not easily, when looking directly at the program files, which appear “garbled” without the MySQL program to decode and display the information they contain.

Regardless of which method is employed, the relevant system files are the same. While MySQL creates a number of files and folders, the only ones of interest are those associated with the stored data and data structures. Which files these are and where they are located depend on a number of factors concerning the version of MySQL being run and some of its configuration settings. The version of MySQL being used for this thesis was 5.5.32, meaning that by default the storage engine is InnoDB and the innodb_file_per_table option is turned off. Given these facts, the files of interest include the following: ibdata1, ib_logfile0, ib_logfile1, the db.opt file associated with each individual database, and all .frm files associated with each individual table within
the databases.\textsuperscript{2} \textsuperscript{3} Additionally, the file .mysql\_history can be recovered to show a
history of the command line inputs within the MySQL program.\textsuperscript{5} The .mysql\_history file
is found in the home directory. The location for the other files can be found in the my.cnf
file located in the /etc/mysql/ folder. By default on Linux, MySQL’s data and folder
structure are found at /var/lib/mysql as indicated by the value of the “datadir” variable.\textsuperscript{6}
This is shown in Figure 1.

```
acl@zero:/etc/mysql$ cat my.cnf
#
# The MySQL database server configuration file.
#
# You can copy this to one of:
# - "/etc/mysql/my.cnf" to set global options,
# - "~/.my.cnf" to set user-specific options.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see

# This will be passed to all mysql clients
# It has been reported that passwords should be enclosed with ticks/quotes
# especially if they contain "#" chars...
# Remember to edit /etc/mysql/debian.cnf when changing the socket location.
[client]
port = 3306
socket = /var/run/mysqld/mysqld.sock

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

# This was formally known as [safe_mysqld]. Both versions are currently parsed.
[mysqld_safe]
socket = /var/run/mysqld/mysqld.sock
nice = 0

[mysqld]
#
# * Basic Settings
#
user = mysqld
pid-file = /var/run/mysqld/mysqld.pid
socket = /var/run/mysqld/mysqld.sock
port = 3306
basedir = /usr
datadir = /var/lib/mysql
tmpdir = /tmp
lc-messages-dir = /usr/share/mysql
```

Figure 1: Partial display of my.cnf file showing that the data directory (datadir) is
/\var\lib/mysql
The files listed contain the necessary data to reconstruct the databases and their associated entries. The metadata and the table data are stored in the ibdata1 file. It also contains the indexes for the tables that keep the data organized and available for retrieval. As their name suggests, ib_logfile0 and ib_logfile1 are log files that help MySQL recovery from crashes by keeping record of commands that made or attempted to make changes to the tables.[3] Each database has one db.opt file and, assuming it has any tables, at least one .frm file. The db.opt file contains configuration information about the database.[3][7] There will exist one .frm file for every table created within a database. This .frm file stores metadata about its table. Because the InnoDB engine also stores this metadata in the aforementioned ibdata1 file, it is not always necessary to include the .frm files just to view the database as it was. It is recommended, however, as having these files may be required in order to do a backup at a later date, should that prove to be necessary. Including them in the move with the other files is the safest option since it keeps the file structure as similar to the original instance of MySQL as possible.[3]

3.2 The Copy Method

3.2.1 Procedure of the Copy Method

The essence of the Copy Method is to use system administrator privileges to access the relevant program files where MySQL stores the data and the table structures and references. These files are then copied into another instance of MySQL the password of which is known. The source files (those being copied) will replace any identically named existing files in the new instance of MySQL. It is important to note, then, that this new instance of MySQL must be a clean instance and not have any databases, tables, or
entries associated with it. It is vital that the folder structure and the permissions for each file and folder be observed as well. This technique is dependent on putting the files where MySQL will expect them to be when it is run and commanded to use a certain database and table. In a sense, the second instance of MySQL is being tricked into reading files that it never created but that somehow just showed up in its file structure. The following example should be followed in order to illuminate the details of the Copy Method along with its associated commands.

The first virtual machine is the source; that is, it has the files the investigator is interested and is the instance of MySQL protected by the password unknown to the investigator. It was set up with the password “password00” and a few databases, tables, and entries, as demonstrated in part below.

```
mysql> create database Dogs;
Query OK, 1 row affected (0.00 sec)

mysql> use Dogs
Database changed

mysql> create table Boarders (Name varchar(20), Breed varchar(20), Sex char(1), Age int, Birthday DATE, Owner varchar(20));
Query OK, 0 rows affected (0.01 sec)

mysql> describe Boarders;
```

Figure 2: Description of the Boarders table in database Dogs
mysql> select * from Boarders;
Empty set (0.00 sec)

mysql> insert into Boarders values ('Apollo', 'Golden Retriever Mix', 'M', '12', '2001-02-12', 'Sally');
Query OK, 1 row affected (0.00 sec)

Similar statements were used to insert two other entries into the table. The result is shown in Figure 3.

mysql> select * from Boarders;
+----------+----------------+----+----+----------------+-----------+----------+
| Name     | Breed           | Sex | Age | Birthday       | Owner     |
+----------+----------------+----+----+----------------+-----------+----------+
| Apollo   | Golden Retriever Mix | M  | 12 | 2001-02-12     | Sally     |
| Samson   | German Shepherd Dog | M  | 13 | 0000-00-00     | Sally     |
| Chloe    | Pomeranian      | F  | 3  | 2010-05-14     | Jack      |
+----------+----------------+----+----+----------------+-----------+----------+
3 rows in set (0.00 sec)

**Figure 3: Data entries for the Boarders table in database Dogs**

The above process was repeated to create and populate a table named Strays, also in the database Dogs, resulting in Figure 4. An UPDATE command was given to demonstrate how it appears in each method. The effects of the UPDATE command are given in Figure 5.

mysql> select * from Strays;
+----------+----------+----+------+----------------+----------------+----------------+-----------------+----------+
| Name     | Breed    | Sex | Age  | WeightInLbs | DateCheckedIn  | DateCheckedOut  |
+----------+----------+----+------+----------------+----------------+----------------+-----------------+----------+
| JaneDoe  | Pitbull  | F   | Adult| 42           | 2013-10-10      | 2013-10-10      | 6600-00-00      | 2013-10-09 |
| Lightning| Greyhound| M   | Adolescent| 05     | 2013-10-00     | 2013-10-00      | 2013-10-00      | 2013-10-09 |
+----------+----------+----+------+----------------+----------------+----------------+-----------------+----------+
2 rows in set (0.00 sec)

**Figure 4: Data entries for the Strays table in database Dogs prior to the UPDATE command**

mysql> update Strays set Name='Sweetie' where Name='JaneDoe' and Breed='Pitbull' and DateCheckedIn='2013-10-10';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
A second database, Cats, was created. Similar to database Dogs, it also has two tables named Boarders and Strays. These were populated using INSERT statements. A DELETE command was also input to show how it appears in each method. The final forms of these two tables are shown in Figure 6 and Figure 7.

The data is now shown to be stored and password protected by the source machine’s instance of MySQL. Using the command “sudo su” and inputting the
administrative password allows a look at the files MySQL created in the above process. These files are located at the file path /var/lib/mysql. Prior to the data creation, it appeared as shown in Figure 8.

```
total 28696
  drwxr-xr-x 35 root root    4096 Oct 24 22:41 ..
  -rw-r--r--  1 root root     0 Oct 24 22:41 debian-5.5.flag
  -rw-rw-r--  1 mysql mysql   18874368 Oct 27 16:29 ibdata1
  -rw-rw-r--  1 mysql mysql   5242880 Oct 27 16:29 ib_logfile0
  -rw-rw-r--  1 mysql mysql   5242880 Oct 24 22:41 ib_logfile1
  drwx------  2 mysql root    4096 Oct 24 22:41 mysql
  -rw-rw-r--  1 root root     6 Oct 24 22:41 mysql_upgrade_info
  drwx------  2 mysql mysql    4096 Oct 24 22:41 performance_schema
  drwx------  2 mysql root    4096 Oct 24 22:41 test
```

**Figure 8: File structure before data creation**

But it now looks like Figure 9.

```
total 28704
  drwx------  7 mysql mysql   4096 Oct 27 16:27 .
  drwxr-xr-x 35 root root    4096 Oct 24 22:41 ..
  drwx------  2 mysql mysql    4096 Oct 27 16:28 Cats
  -rw-r--r--  1 root root     0 Oct 24 22:41 debian-5.5.flag
  drwx------  2 mysql mysql    4096 Oct 27 16:26 Dogs
  -rw-rw-r--  1 mysql mysql   18874368 Oct 27 16:29 ibdata1
  -rw-rw-r--  1 mysql mysql   5242880 Oct 27 16:29 ib_logfile0
  -rw-rw-r--  1 mysql mysql   5242880 Oct 24 22:41 ib_logfile1
  drwx------  2 mysql root    4096 Oct 24 22:41 mysql
  -rw-rw-r--  1 root root     6 Oct 24 22:41 mysql_upgrade_info
  drwx------  2 mysql mysql    4096 Oct 24 22:41 performance_schema
  drwx------  2 mysql root    4096 Oct 24 22:41 test
```

**Figure 9: File structure after data creation**

With subfolders Dogs and Cats appearing in Figure 10 and Figure 11.
Now because sudo is required to access the MySQL program files, they must be copied to a less protected folder (one not requiring administrative privileges) that is accessible by the sftp process and moved to the "clean" virtual machine that the investigator has total control over. There are several ways to go about this, but one method is demonstrated, in part, as follows.

```
cat@zero:~$ sudo su
[sudo] password for acl:
root@zero:/home/acl# cd ../var/lib/mysql
root@zero:/var/lib/mysql# ls -al
```

```
root@zero:/var/lib/mysql/Dogs# ls -al
total 36
  drwx------ 2 mysql mysql 4096 Oct 28 17:32 .
  drwx------ 7 mysql mysql 4096 Oct 28 17:33 ..
-rw-rw---- 1 mysql mysql 8718 Oct 28 17:32 Boarders.frm
-rw-rw---- 1 mysql mysql 65 Oct 28 17:32 db.opt
-rw-rw---- 1 mysql mysql 8790 Oct 28 17:32 Strays.frm
```

**Figure 10: Files created within the Dogs database**

```
root@zero:/var/lib/mysql/Cats# ls -al
total 36
  drwx------ 2 mysql mysql 4096 Oct 28 17:33 .
  drwx------ 7 mysql mysql 4096 Oct 28 17:33 ..
-rw-rw---- 1 mysql mysql 8686 Oct 28 17:33 Boarders.frm
-rw-rw---- 1 mysql mysql 65 Oct 28 17:33 db.opt
-rw-rw---- 1 mysql mysql 8758 Oct 28 17:33 Strays.frm
```

**Figure 11: Files created within the Cats database**

```
```
The folders and files in need of copying have been bolded. The directory DataCopy was created along with subfolders Dogs and Cats to mimic the structure of MySQL’s data files and keep the identically named files from the Dogs and Cats folders separate. While the above only shows copying one file from the Cats folder, the same should be done for the other two files in Cats, all three files in the Dogs folder, and the three files indicated in the mysql folder.

```
root@zero:/var/lib/mysql# cd ..../..../home/acl
root@zero:/home/acl# ls -al
```
total 32
```
drwxr-xr-x 3 acl acl  4096 Oct  3 10:54 .
drwxr-xr-x 3 root root  4096 Oct  3 10:57 ..
-rw------- 1 acl acl   99 Oct  3 11:08 .bash_history
-rw-r--r-- 1 acl acl  220 Oct  3 10:57 .bash_logout
-rw-r--r-- 1 acl acl  3486 Oct  3 10:57 .bashrc
drw------- 2 acl acl  4096 Oct  3 10:58 .cache
-rw------- 1 acl acl  2637 Oct 10 20:54 .mysql_history
-rw-r--r-- 1 acl acl   675 Oct  3 10:57 .profile
root@zero:/home/acl# cp .mysql_history ../../../var/lib/DataCopy
root@zero:/home/acl# cd ../../../var/lib/DataCopy
root@zero:/var/lib/DataCopy# ls -al
```
total 28692
drwxr-xr-x 4 root root  4096 Oct 10 22:03 .
```
The above also shows where to find the .mysql_history file, which is discussed later in this section. It should be noted that the chmod command (chmod 644) must be given on all ten files that sftp might be allowed access to the files. The administrator can then put the files onto a third machine and then onto the clean machine (useful if the source machine becomes the clean machine after a wipe) or directly onto the clean machine from the source.

```
root@one:/var/lib/DataCopy# chmod 644 ibdata1
```

This is setting up the folder structure to receive the source files. It should be noted that chmod 777 is executed on each directory to ensure sftp has access to them for the copy. Again, because administrator access is required to access the mysql folder, the files cannot be put directly into place but instead must be put onto the clean machine and then moved into place using the sudo command.

```
root@one:/var/lib/DataCopy# chmod 777 Cats
root@one:/var/lib/DataCopy# chmod 777 Dogs
```

```
root@one:/var/lib/mysql# ls -al
total 28696
```
The files being removed were created when MySQL was installed. However, they need to be replaced by the source files since the source files contained the data from the source machine. Now, the folders Cats and Dogs must be created in the mysql folder, and all files should be copied over into their appropriate places.

```
root@one:/var/lib/mysql# mkdir Cats
root@one:/var/lib/mysql# mkdir Dogs
root@one:/var/lib/mysql# cd ../DataCopy
root@one:/var/lib/DataCopy# cp ibdata1 ../mysql
root@one:/var/lib/DataCopy/Dogs# cd ../mysql
root@one:/var/lib/mysql# chmod 644 ibdata1
root@one:/var/lib/mysql# chmod 744 Cats
root@one:/var/lib/mysql# cd Cats
root@one:/var/lib/mysql/Cats# chmod 644 db.opt
```

The above gives a brief example of creating the folders, copying one of the files, and then changing the permissions on the files and folders such that MySQL will be allowed access to them when it is run. All of the files should be copied and their permissions changed as demonstrated.

In order for MySQL to recognize these changes and access the files appropriately, the following commands must be given. These will effectively restart the MySQL service.
root@one:/var/lib/mysql/Dogs# stop mysql
mysql stop/waiting
root@one:/var/lib/mysql/Dogs# start mysql
mysql start/running, process 20202

The .mysql_history file was copied from the source machine and can be placed on the clean machine. It can either replace the clean machine’s existing .mysql_history file or be renamed and stored in addition to it. Looking at this file using the cat command in the command line allows the investigator to look back through the commands entered into MySQL to search for deletes, updates, and other commands that may not be easily determined from looking only at the final resulting data. The feature of recording commands to this history file can be disabled, and the file can be deleted.[8] Thus, access to this file may still not paint a complete picture of the target’s activities.

3.2.2 Results of the Copy Method

The files are now in place on the clean machine, and MySQL has been restarted – a necessary step in order to sync everything together. The password-protected data can now be viewed from the clean machine. Two screenshots appear in Figures 12 and 13. Figure 12 is from the source machine, while Figure 13 is from the clean machine. It is clear that they now contain the same information, indicating that the transfer of files was a success. While a somewhat tedious process, the password protecting the data was bypassed using administrative access and a fresh instance of MySQL with a known password.
Figure 12: MySQL output showing the data stored on the source machine
root@ones:/var/lib/mysql# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \\g.
Your MySQL connection id is 117
Server version: 5.5.32-0ubuntu10.12.04.1 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use Dogs
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from Strays;

<table>
<thead>
<tr>
<th>Name</th>
<th>Breed</th>
<th>Sex</th>
<th>Age</th>
<th>WeightInLbs</th>
<th>DateCheckedIn</th>
<th>DateCheckedOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sweetie</td>
<td>Pitbull</td>
<td>F</td>
<td>Adult</td>
<td>42</td>
<td>2013-10-10</td>
<td>0000-00-00</td>
</tr>
<tr>
<td>Lightning</td>
<td>Greyhound</td>
<td>M</td>
<td>Adolescent</td>
<td>85</td>
<td>2013-10-09</td>
<td>2013-10-09</td>
</tr>
</tbody>
</table>

2 rows in set (0.00 sec)

mysql> select * from Boarders;

<table>
<thead>
<tr>
<th>Name</th>
<th>Breed</th>
<th>Sex</th>
<th>Age</th>
<th>Birthday</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apollo</td>
<td>Golden Retriever Mix</td>
<td>M</td>
<td>12</td>
<td>2001-02-12</td>
<td>Sally</td>
</tr>
<tr>
<td>Samson</td>
<td>German Shepherd Dog</td>
<td>M</td>
<td>13</td>
<td>0000-00-00</td>
<td>Sally</td>
</tr>
<tr>
<td>Chloe</td>
<td>Pomeranian</td>
<td>M</td>
<td>3</td>
<td>2010-05-14</td>
<td>Jack</td>
</tr>
</tbody>
</table>

3 rows in set (0.00 sec)

mysql> use Cats
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from Strays;

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>WeightInLbs</th>
<th>DateCheckedIn</th>
<th>DateCheckedOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>JaneDoc</td>
<td>F</td>
<td>Kitten</td>
<td>12</td>
<td>2013-10-05</td>
<td>0000-00-00</td>
</tr>
</tbody>
</table>

1 row in set (0.00 sec)

mysql> select * from Boarders;

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Birthday</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speckles</td>
<td>F</td>
<td>6</td>
<td>2007-01-17</td>
<td>Junior</td>
</tr>
<tr>
<td>DC</td>
<td>M</td>
<td>10</td>
<td>2003-04-21</td>
<td>Missy</td>
</tr>
</tbody>
</table>

2 rows in set (0.00 sec)

Figure 13: MySQL output showing the data that has been transferred to the clean machine
3.3 The Plainview Method

3.3.1 Procedure of the Plainview Method

The Plainview Method involves looking at the aforementioned files directly (by using the cat command, for example). While most of the files are unreadable to a human without MySQL to decipher them, they can still yield useful information about the structure and data associated with the tables in the databases. The results of this method are more difficult to interpret and are likely far less complete than those produced by the Copy Method. However, if another instance of MySQL is not available due to limited resources or any other factor, this may be the only option to gather any information at all. Though it does not yield as much information as an investigator would probably desire, it is better than being left without even a glimpse into the table structure and data.

Because ibdata1 is known to contain the data entries for the tables and the .frm files contain the table structure, they can be used to gain a limited look at what was stored in the databases. If the cat command is given on a .frm file, the result will be something similar to Figure 14 below.

```
rroot@one:/var/lib/mysql/Dogs$ cat Strays.frm
OK     0     2      0       0  
Name Breed Sex Age
WeightInLbs DateCheckedIn
DateCheckedOut,?

8

Fp?

Fp?

Fp?

Fp?
```

**Figure 14: The output of Strays.frm from the Dogs database**

It is seen that the last line corresponds to the column names for the Strays table in the Dogs database with a ‘?’ serving as a delimiter. This provides an idea of what
type of information is stored and gives at least a clue to the data type. Moving to look at the ibdata1 file, the cat command yields results a bit more difficult to interpret. It may be helpful to find a way to eliminate all of the question marks. One method is to copy and paste the output into Microsoft Word and use the replace feature to search for ‘?’ and replace with nothing. The information of interest is found towards the end of the file.

3.3.2 Results of the Plainview Method

Figure 15 shows the output that relates to the table Strays in database Dogs.

Figure 15: Part of the output from the ibdata1 file that pertains to the Strays table in database Dogs

The part that is of interest starts with “Dogs/Strays” to indicate that the data following pertains to that database and table. This pattern of “Database/Table” is seen at the beginning of all other segments dealing with the different databases and tables. From here, one can look character by character until getting to the phrase “SweetiePitbullFAdult” which is very much out of place compared to the seeming random characters filling the most of the rest of the screen. Using the pattern that was found in the .frm file, it can be inferred that Sweetie is the name, Pitbull is the breed, F is the sex, and Adult is the age. The above phrase is followed by “??”.
is of note that the character ‘*’ has an ASCII value of 42, which corresponds to the weight entered for Sweetie.\[9\] The preceding question mark has an unknown meaning. Further, Dates do not seem to appear in any easily decodable manner. However, the next six characters can be separated into groups of three. It does appear through testing and observation that each group of three characters could represent one of the Date entries. Table 1 contains results from an experiment not explicitly described here that determined how some known Date entries were recorded in the ibdata1 file.

Table 1: The encodings for various known Date entries

<table>
<thead>
<tr>
<th>Known Date</th>
<th>Encoding in ibdata1 File</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-01-01</td>
<td>??!</td>
</tr>
<tr>
<td>2013-01-02</td>
<td>??”</td>
</tr>
<tr>
<td>2013-01-03</td>
<td>??#</td>
</tr>
<tr>
<td>2013-01-05</td>
<td>??%</td>
</tr>
<tr>
<td>2013-01-31</td>
<td>??</td>
</tr>
<tr>
<td>2013-02-01</td>
<td>??A</td>
</tr>
<tr>
<td>2013-02-02</td>
<td>??B</td>
</tr>
<tr>
<td>2013-02-03</td>
<td>??C</td>
</tr>
<tr>
<td>2013-03-03</td>
<td>??c</td>
</tr>
<tr>
<td>2013-03-29</td>
<td>??}</td>
</tr>
<tr>
<td>2013-04-03</td>
<td>??</td>
</tr>
<tr>
<td>2013-04-04</td>
<td>??</td>
</tr>
<tr>
<td>2013-05-05</td>
<td>??</td>
</tr>
<tr>
<td>2013-12-01</td>
<td>??</td>
</tr>
<tr>
<td>2012-01-01</td>
<td>??!</td>
</tr>
</tbody>
</table>

While there does seem to be some sort of pattern here, current research has not yielded anything definitive enough to be able to allow one to decode a Date entry directly from the ibadata1 file. Further, some encodings seem to involve characters that do not visible appear when printed to the screen. Reading on, one can see “LightningGreyhoundMAdolescent” and can again infer that this is the name, breed,
sex, and age, with similar encodings following for the weight and dates as just described. Looking farther, “JaneDoe” is seen. Even though this does not appear in the table (which is shown for comparison in Figure 16), it is still in the ibdata1 file. This is because an UPDATE command was done. When a command such as UPDATE or DELETE is given, the data remains in ibdata1 but does not display in the table when called from the MySQL program.

```
mysql> select * from Strays;
+---------------+-------+---+---+-------------+-----------------+-----------------+
<table>
<thead>
<tr>
<th>Name</th>
<th>Breed</th>
<th>Sex</th>
<th>Age</th>
<th>WeightInLbs</th>
<th>DateCheckedIn</th>
<th>DateCheckedOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sweetie</td>
<td>Pitbull</td>
<td>F</td>
<td>Adult</td>
<td>42</td>
<td>2013-10-10</td>
<td>2013-10-09</td>
</tr>
<tr>
<td>Lightning</td>
<td>Greyhound</td>
<td>M</td>
<td>Adolescent</td>
<td>05</td>
<td>2013-10-00</td>
<td>2013-10-09</td>
</tr>
</tbody>
</table>
+---------------+-------+-----+-----+-------------+-----------------+-----------------+
2 rows in set (0.00 sec)
```

**Figure 16:** The data entries for the Strays table of the Dogs database after the UPDATE command was executed
4. CONCLUSION

4.1 Summary

There are times when an investigator may need access to the data stored in MySQL but may not have the individual user's password to the program. Whether the user is simply unavailable or perhaps under investigation, the investigator may need to use the higher permissions of the system administrator to bypass the user’s password. This thesis explains two methods that help achieve this goal. The Copy Method copies the system files to a new instance of MySQL and totally defeats the password protection. The Plainview Method looks at the system files through the command line and provides a narrow view that reveals some of the data.

4.2 Contributions

The methods explored in this thesis provide powerful tools to the investigator needing to gain access to a user’s MySQL database without the need for the user’s cooperation. This is useful in cases of emergency where the user is unavailable or where the user is under investigation (and thusly unaware of or uncooperative with the investigation) and the data stored in MySQL is believed to be relevant. It should be noted, however, that this could also be considered a vulnerability in MySQL’s security as someone with administrative access could use the above methods for nefarious means without the consent of the system’s owner.
4.3 Future Work

An exploration of exactly what permission levels are needed on the folders and files to be copied and accessed in the Copy Method would enhance that approach so that higher permissions than are necessary would not be set. It is generally accepted that files, like users, should be given only the level of access that is truly needed.

While it at first appeared that copying the files higher into the folder structure (to a less protected folder) before transferring them to the clean machine was a necessary step so that sftp could be used, further investigation has suggested the possibility of alternative, less tedious methods.

Further study on the patterns formed by the Date data types in the ibdata1 file would prove useful if it revealed a reliable way to decipher the encoding methods. Along the same lines, testing with the rest of the data types supported by MySQL database to study their encodings would provide useful information that would also enhance the effectiveness of the Plainview Method.
References


