Auditing a MySQL Database Server
An Independent Auditors Perspective

Jeff Hoover

February 2004
SANS GSNA V. 2.1 (Option 1)

Table of Contents

Assignment 1: Research in Audit, Measurement Practice, and Control

1.1 Company Overview

Compuglobalhypermeganet Corporation provides IT services and support, emphasizing customer satisfaction and service reliability. The major gateway for client support requests is through a web interface. This interface stores support request information in a backend database server and alerts Compuglobalhypermeganet employees via email when a new request comes in. The database holds thousands of requests and is considered a critical service, as it supports the entire base of customers. Sensitive client information is contained in the database and access is restricted to support and accounting personnel. This support request system is one of the services Compuglobalhypermeganet offers to implement for clients, so showing its reliability and uptime is very important to the company.

1.2 Identify the system to be audited

The subject of this audit is the MySQL database containing the support requests and client information. Version 4.17 of MySQL is being run on a Red Hat 7.3 server. The Red Hat server has two Ethernet cards installed in it. One network card is connected to a linux web server running Apache that is accessable anywhere on the internet. The other network card is connected to a linux web server running Apache that is only available internally. The internal connection is used by the support department and the accounting department only. The internet interface is accessable by any client on the world wide web. The SQL traffic flows only between the webservers and the database server on dedicated lines. No other machines can connect to the MySQL server. The scope of this audit focuses on the MySQL database itself. It will not cover the entire OS setup/security or the apache web servers connected to it. It also does not cover the front end applications that interact with the database.

1.3 Risks to the System

Risk	Probability	Possible Impact
Published vulnerabilities	High Determined by patch management and security procedures	Data theft, system downtime, loss of credibility. Attacker could take control of the system.

Risk	Probability	Possible Impact
Unauthorized access to the OS	High/Medium Depends on company practices. There is a higher number of local root exploits available. Machines are often set up on NIS/LDAP/etc.	Data confidentiality and integrity, system downtime Attacker could take control of the machine. In this case it would most likely be internal.
Invalid or Missing Backup and Restore Procedures	Medium Backup and restore procedures for critical data are common.	Data loss, system downtime Good backups and procedures will result in less downtime
Physical Access to Machine	Medium Reduced risk if company keeps the server in a secure room.	Data confidentiality and integrity, system downtime Attacker could take control of the system.
Change/Patch Management Procedures	Medium Depends on company policies.	System downtime if untested patches applied that break something Difficult to maintain baseline comparison.
Unauthorized access to the database	High Depends on the user permissions.	Data confidentiality and integrity if attackers can see/edit things they are not supposed to. Possible system downtime.
No system audit logs	Medium	May not be able to trace or detect attacks.

1.4 Current State of Practice

The Internet was the primary tool used while conducting research for securing a MySQL database server. Search engines such as Google (http://www.google.com) and Yahoo (http://www.yahoo.com) were used to search for terms like "mysql security checklist", "mysql security", "mysql audit", "securing mysql" and many similar phrases. MySQL's website (http://www.mysql.com) and the SANS website (http://www.sans.org) was also searched for additional information pertaining to MySQL security.

The following is a list of resources referenced in creating an audit checklist:

- MySQL Manual - Section 5 - Database Administration: http://www.mysql.com/doc/en/MySQL_Database_Administration.html

- Secure MySQL Database Design: http://www.securityfocus.com/infocus/1667, Kristy Westphal, February 18, 2003

- Securing MySQL: step-by-step: http://www.securityfocus.com/infocus/1726, Artur Maj, August 28, 2003

- Securing Your MySQL Installation, Document revision: 1.01: http://www.kitebird.com/articles/ins-sec.html, Paul DuBois, January 25, 2003

- RFC 2196 - Site Security Handbook: http://www.faqs.org/rfcs/rfc2196.html, B. Fraser, September 1997

- SANS/GIAC Student Practicals: http://www.giac.org/GSNA.php

- SANS Course Materials

The MySQL Manual contains an extensive amount of information about MySQL security, but is somewhat cumbersome to read. It covers in detail the privilege system and how to grant and remove secure privileges. It also has a subsection devoted to general MySQL security issues and what to look out for. The online version contains user comments that are sometimes useful.

Westphal's article is a good general summary of things to look out for when securing MySQL, but doesn't provide very many specific details. She focuses on the entire picture, from network architecture to database backups and refers to the MySQL manual throughout the article.

Artur Maj's article focuses on securing MySQL along with Apache and PHP. The article begins with security requirements and continues with detailed instructions on installing and configuring MySQL. Although Maj assumes the reader will install MySQL and Apache on the same machine, most of the article is still relevant and useful. Communication between PHP and MySQL is also mentioned.

DuBois wrote several books on MySQL and is widely known in the MySQL community. He does a very good job of explaining how to secure an existing MySQL installation and what some of the common security risks are. DuBois does the best job of explaining how and why certain steps are taken.

The Site Security Handbook was used as a reference for general security practices. The handbook is a guide to developing computer security policies and procedures.

Assignment 2: Create an Audit Checklist

Check 1 – Unprivileged account running MySQL

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_3 MySQL Manual: http://www.mysql.com/doc/en/Security_against_attack.html
Control Objective	Run the server as mysql (or another unprivileged user) to protect system files.
Risk	MySQL can be used to read or create system files as the user running the MySQL server leading to a system compromise or data leak.
Probability	Low. Does not run as root by default.
Compliance	The 'user' directive is defined in the /etc/my.cnf file in the 'mysqld' section.
Testing	Check the /etc/my.cnf file to make sure that 'user' is defined: [mysqld] user=mysql View the current process list to ensure the server is running as the specified user: ps aux | grep -i mysql
Objective/Subjective	Objective

Check 2 – Database directory permissions

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_3 MySQL Manual: http://www.mysql.com/doc/en/Security_against_attack.html

Control Objective	Database directory only accessible to the user running the MySQL server.
Risk	Users' with local accounts may bypass the MySQL server completely to gain direct access to the data files resulting in unauthorized access to view the data and possibly modify or delete the data.
Probability	Low. Default installation permissions are secure.
Compliance	The mysql data directory is owned by mysql user and nobody else has any permissions (700).
Testing	Locate the mysql data directory. The default is /usr/local/mysql/data when installed from source. View file ownership and permissions by typing: ls –al /usr/local/mysql/data The directory should be owned by mysql.mysql with permissions of rwx------ (700).
Objective/Subjective	Objective

Check 3 – Global option file

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_6 Securing MySQL: step-by-step (Section 2.2): http://www.securityfocus.com/infocus/1726
Control Objective	The global option file should only be modifiable by the root user.
Risk	An unauthorized user could modify the option file telling the server to run as root. Many other options could also be changed. Potential risks include: system compromise, data confidentiality and system downtime.
Probability	Low. The global option file must be created in a directory owned by root (/etc).
Compliance	The my.cnf file located in /etc should be owned by root and only modifiable by root (644).

Testing	View the file ownership and permissions: ls –l /etc/my.cnf The owner should be root.root with permissions of rw-r--r-- (644).
Objective/Subjective	Objective

Check 4 – User option files

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_6
Control Objective	Access to user-specific option files should be restricted to the owner.
Risk	Local users can create a my.cnf file in their home directory and define their mysql password in it. Since the password is stored in plain text, the file needs to be secured to prevent unauthorized users from viewing the password and accessing MySQL as that user. Possible risks are determined by the permissions the MySQL user has.
Probability	Medium. An unprivileged user generally isn't concerned with file permissions and is less likely to check them.
Compliance	User-specific my.cnf files should be accessible only to its owner (600).
Testing	Look in each users' home directory and check the file permissions for my.cnf. Home directories are defined in /etc/passwd. ls –l <each home directory>/my.cnf The file owner should be the user with permissions of rw------- (600).
Objective/Subjective	Objective

Check 5 – All accounts have passwords

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_9 MySQL Manual: http://www.mysql.com/doc/en/Security_guidelines.html Securing MySQL: step-by-step (Section 4.3): http://www.securityfocus.com/infocus/1726
Control Objective	Require a password to connect to MySQL.
Risk	Unauthorized access to a MySQL account. Potential risks include: system compromise, data confidentiality and system downtime.
Probability	Very High. The default root password is not set, so this is extremely critical.
Compliance	The mysql.user table should have a value in the Password field for every row.
Testing	Run a query to see if any accounts have empty passwords: SELECT User, Host FROM user WHERE Password = ''; Zero rows should be returned.
Objective/Subjective	Objective

Check 6 – Rename root account

Reference	Securing MySQL: step-by-step (Section 4.5): http://www.securityfocus.com/infocus/1726
Control Objective	Rename the root account to harden the security.
Risk	Unauthorized access to the MySQL root account by using a brute-force or dictionary attack. Potential risks include: system compromise, data confidentiality and system downtime.
Probability	Medium, depending on password strength.

Compliance	The mysql.user table should not have an entry for a user named root.
Testing	Run a query to see if a root account exists: SELECT User, Host FROM user WHERE User = 'root'; Zero rows should be returned.
Objective/Subjective	Objective

Check 7 – No anonymous accounts

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_9
Control Objective	Require a username to connect to MySQL.
Risk	Unauthorized access to a MySQL account. Potential risks include: system compromise, data confidentiality and system downtime.
Probability	Low.
Compliance	The mysql.user table should have a value in the User field for every row.
Testing	Run a query to see if any accounts have empty user fields: SELECT User, Host FROM user WHERE User = ''; Zero rows should be returned.
Objective/Subjective	Objective

Check 8 – Limiting hosts

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_12
Control Objective	State only the hosts that need to connect to the database without wildcards.

Risk	Using a wildcard in the Host field or stating machines other than the webservers increases the number of machines an unauthorized user can attempt to connect from. Possible risks are determined by the permissions the MySQL user has that the attacker connects as.
Probability	Low. Default account can only connect from the localhost.
Compliance	The mysql.user table should not contain any '%' wildcard characters in the Host field and should specifically state only the two webservers as IP addresses and the localhost. The admin account should only be able to connect from the localhost.
Testing	To complete this test, the auditor must know the IP addresses of the two webservers that are connecting to the database. The first step of this test would be to check for wildcards in the Host field: SELECT User, Host FROM user WHERE Host LIKE '%\%%'; Zero results should be returned. Next, check the hosts that are allowed to connect to the database: SELECT User, Host FROM user; The admin account should only be allowed to connect from the localhost. The only other Hosts that should be specified are the webservers, as IP addresses.
Objective/Subjective	Objective

Check 9 – Unnecessary global privileges

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_13
Control Objective	Restrict user permissions to only the databases they need to connect to.
Risk	Granting global privileges will allow the user with the privileges access to every table and database. This could lead to unauthorized access to data as well as data integrity issues if the user was granted insert,

	update or delete privileges.
Probability	High. Novice administrators often grant global permissions.
Compliance	Check the mysql.user and mysql.db tables to make sure permissions are not global.
Testing	Permissions should be as specific as possible when granted, but most of the time the auditor won't know exactly what permissions are needed. Note that the admin user should always have global permissions. Global permissions can be tested for by checking the mysql.user table. If any of the privileges are marked 'Y' in this table, they are global unless specified as 'N' in the mysql.db table. SELECT User, Host FROM user WHERE Select_priv = 'Y' OR Insert_priv = ‘Y’ OR Update_priv = ‘Y’ OR Delete_priv = ‘Y’ OR Create_priv = ‘Y’ OR Drop_priv = ‘Y’ OR Grant_priv = ‘Y’ OR Index_priv = ‘Y’ OR Alter_priv = ‘Y’;
Objective/Subjective	Subjective. It is possible the user is supposed to have global privileges. This is something the auditor will have to ask the DBA about.

Check 10 – Process, super, shutdown and file privileges

Reference	MySQL Manual: http://www.mysql.com/doc/en/Security_against_attack.html Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_13
Control Objective	Restrict administrative privileges to only the admin user.
Risk	The PROCESS privilege gives a MySQL user the ability to see the currently executing queries. This could expose a password if a password is being set. The FILE privilege will allow a user to read files from and write files to the filesystem. The SHUTDOWN privilege will allow a user to shut down the database. The SUPER privilege allows a user to terminate client connections and change system variables.

	Possible consequences of these actions include: Data confidentiality and integrity, system downtime and system compromise.
Probability	High. Novice administrators often grant too many privileges or ALL privileges.
Compliance	Check the mysql.user tables to make sure the PROCESS, FILE, SUPER and SHUTDOWN permissions are only granted to the admin account.
Testing	Run a query on the mysql.user table to test for excessive permissions: SELECT User, Process_priv, File_priv, Super_priv, Shutdown_priv FROM user WHERE Process_priv = 'Y' OR File_priv = 'Y' OR Super_priv = 'Y' OR Shutdown_priv = 'Y'; Only the admin row should be returned.
Objective/Subjective	Objective

Check 11 – Disable LOAD DATA LOCAL INFILE

Reference	MySQL Manual: http://www.mysql.com/doc/en/LOAD_DATA_LOCAL.html Securing MySQL: step-by-step (Section 4.2): http://www.securityfocus.com/infocus/1726
Control Objective	Disable LOAD DATA LOCAL INFILE to prevent reading of local files.
Risk	There are two main risks to consider. One, the user could read any file that the webserver has access to read. Two, the server could access any file on the client host (the webserver) to which the client user has read access. This could lead to a system compromise.
Probability	High. Binary distributions are compiled with this option turned on.

Compliance	Look in the option file to see if local-infile is set. Also, try to run the command.

Testing	First, look in /etc/my.cnf to see if set-variable=local-infile=0. If it is set, the database should produce an error when the LOAD DATA LOCAL INFILE command is run.
	
	To test the command create a file in the /tmp directory that the mysql server can read:
	
	echo 1 > /tmp/audit_test.txt
	
	Next, create a database and table to import the file into:
	
	CREATE database Audit_Test;
	USE Audit_Test;
	CREATE TABLE Test (
	  ID char(1) NOT NULL default '',
	  PRIMARY KEY (ID)
	);
	
	Finally, run the command to see if an error is produced or if the import works:
	
	LOAD DATA LOCAL INFILE '/tmp/audit_test.txt' INTO TABLE Test;
	
	Verify the results with:
	
	SELECT * FROM Test;
	
	The result set should be empty.

Objective/Subjective	Objective

Check 12 – Clean .mysql_history

Reference	Securing MySQL: step-by-step (Section 4.6): http://www.securityfocus.com/infocus/1726
Control Objective	Plaintext passwords must be removed from the .mysql_history file.
Risk	Commands run from the MySQL client are stored in the .mysql_history file, including passwords. If an

	unauthorized user was able to view the .mysql_history file containing passwords, they would be able to authorize the database. Possible risks are determined by the permissions the MySQL user has including: Possible consequences of these actions include: Data confidentiality and integrity, system downtime and system compromise.
Probability	Low. The history file is only readable by the owner by default. However, it will most likely contain the root password.
Compliance	Search the .mysql_history files for passwords. None should be found.
Testing	Locate all the .mysql_history files by searching through home directories or by running the following system command: locate .mysql_history Search the files for 'Password': grep -n Password .mysql_history A match doesn't guarantee a password will be exposed. The contents of the file will have to be viewed to see if the password is stored.
Objective/Subjective	Objective

Check 13 – No passwords on command line or environment

Reference	Securing Your MySQL Installation: http://www.kitebird.com/articles/ins-sec.html#TOC_14
Control Objective	Prevent unnecessary exposure of passwords when connecting to the server.
Risk	Typing passwords on the command line or setting it in an environment variable creates a higher risk of password exposure. First, the password is shown on the screen while typed, so someone passing by could see it. Second, a user

	with the ability to view process information could see the password when the command is executed. Finally, the password may be stored in plain text in the user's .bash_history file. Again, possible risks are determined by the permissions the MySQL user has including: Possible consequences of these actions include: Data confidentiality and integrity, system downtime and system compromise.
Probability	Low.
Compliance	Check the users' .bash_history files for evidence of passwords. Ask the users how they login.
Testing	Locate all the .bash_history files on the MySQL server by searching through home directories or by running the following system command: locate .bash_history Search the files for 'mysql': grep -n mysql.*-p .bash_history This provides a place to search for the passwords, but doesn't guarantee the users are not exposing the passwords. The .bash_history file could have been edited. The auditor will have to ask the users that can login, or a sample of them, to log into the machine while watching them. The auditor should ask the users if that is how they normally log in.
Objective/Subjective	Subjective. The auditor must rely on user feedback.

Check 14 – Secure physical location

Reference	Site Security Handbook (Section 4.5.1): http://www.faqs.org/rfcs/rfc2196.html General security practices.
Control Objective	The server must be located in a physically secure

	environment to prevent unauthorized access.
Risk	Physical access to the server would allow an attacker to bring the machine down causing an interruption in service. The attacker could also guess login and passwords or attempt to boot from a floppy. Risks include: Data confidentiality and integrity, system downtime
Probability	High. Studies show that high numbers of security breaches are by internal employees.
Compliance	The server should be in a controlled environment. Levels of control may vary. A minimum level of protection would be to locate the server behind a locked door. Other precautions might include requiring a user to sign an entry log and placing the server in a locked rack.
Testing	The auditor should request to visit the server and record the process used to access it. If a key is required, determine who has access to the room.
Objective/Subjective	Objective

Check 15 – Latest OS security patches installed

Reference	Secure MySQL Database Design: http://www.securityfocus.com/infocus/1667 General security practices.
Control Objective	Reduce exposure by patching all known vulnerabilities on the server.
Risk	An attacker may take control of the server by exploiting a published vulnerability. Risks include system downtime, data integrity, data confidentiality and company reputation.
Probability	High.
Compliance	All packages are patched with the latest security updates.

Testing	If the machine is registered with the Red Hat Network, the up2date utility can be used to obtain a list of packages needing to be installed. up2date –l However, some programs may have been installed from source, not RPMs, so they won't show up in the report generated by up2date. A security scanner such as Nessus (http://www.nessus.org/) can be used to test the entire system. Using Nessus is outside the scope of this paper. Analyze and test the results.
Objective/Subjective	Objective

Check 16 – Backup procedures

Reference	Site Security Handbook (Section 4.7): http://www.faqs.org/rfcs/rfc2196.html Secure MySQL Database Design: http://www.securityfocus.com/infocus/1667 General security practices.
Control Objective	Data is being backed up and securely stored.
Risk	If the system became unavailable and/or unusable it would be necessary to restore the system from a backup. If a backup was not available, data would be lost and system downtime would increase.
Probability	Medium. Depends on backup policy.
Compliance	Ensure that full weekly backups and nightly database backups are performed successfully.
Testing	Review the backup logs to verify that the backups are completed. Also look to make sure no errors are reported. Verify the backups exist and are stored in a secure location.
Objective/Subjective	Objective

Check 17 – Restore procedures

Reference	Secure MySQL Database Design: http://www.securityfocus.com/infocus/1667 General security practices.
Control Objective	Restoration procedures should be known and tested.
Risk	In the event a backup is needed, the staff must be able to restore the files. If the backup does not work or the staff cannot restore the machine, the data may be lost and system downtime is increased.
Probability	Medium. Depends on restore policy.
Compliance	The server should be successfully restored from the backup.
Testing	A test restoration should be performed (on a test server) with the auditor choosing the backup. Verify the server is successfully restored.
Objective/Subjective	Objective.

Check 18 – Patch policies and procedures

Reference	SANS New England Conference, D Hoelzer, September 2003. General security practices.
Control Objective	Minimize exposure to published vulnerabilities and untested patches.
Risk	A fully patched server is only secure until a new vulnerability is published. The server administrator(s) must have a way to be alerted of new vulnerabilities and a process for testing and patching the server. Without proper policies and procedures the system may be compromised and/or brought down from an untested patch.

Probability	High. New exploits are constantly found by security researchers.
Compliance	Verify that the administrator(s) are monitoring published vulnerabilities and that the policies and procedures exist and are being followed.
Testing	Ask to see the policies and procedures regarding security patches. If they don't exist, ask what steps are taken to patch a security vulnerability. Ask what means of monitoring vulnerabilities the administrators are using.
Objective/Subjective	Subjective

Check 19 – Port filtering

Reference	MySQL Manual: http://www.mysql.com/doc/en/Security_guidelines.html SANS New England Conference, D Hoelzer, September 2003. General security practices.
Control Objective	Restrict the ports that can be accessed to the bare minimum, SSH and MySQL.
Risk	Unnecessary ports that are open increase the possibility of compromising the server by using published vulnerabilities. System downtime, data confidentiality and data integrity are at risk.
Probability	High. Many ports are open by default.
Compliance	Test the system with a port scanner to verify that only SSH (port 22) and MySQL (port 3306) are listening.
Testing	Use a port scanner, such as nmap, to test the database server from a remote machine: nmap –sS –p 1- <database IP address> Verify the test results to make sure only ports 22 and 3306 are open.

Objective/Subjective	Objective

Check 20 – Log Files

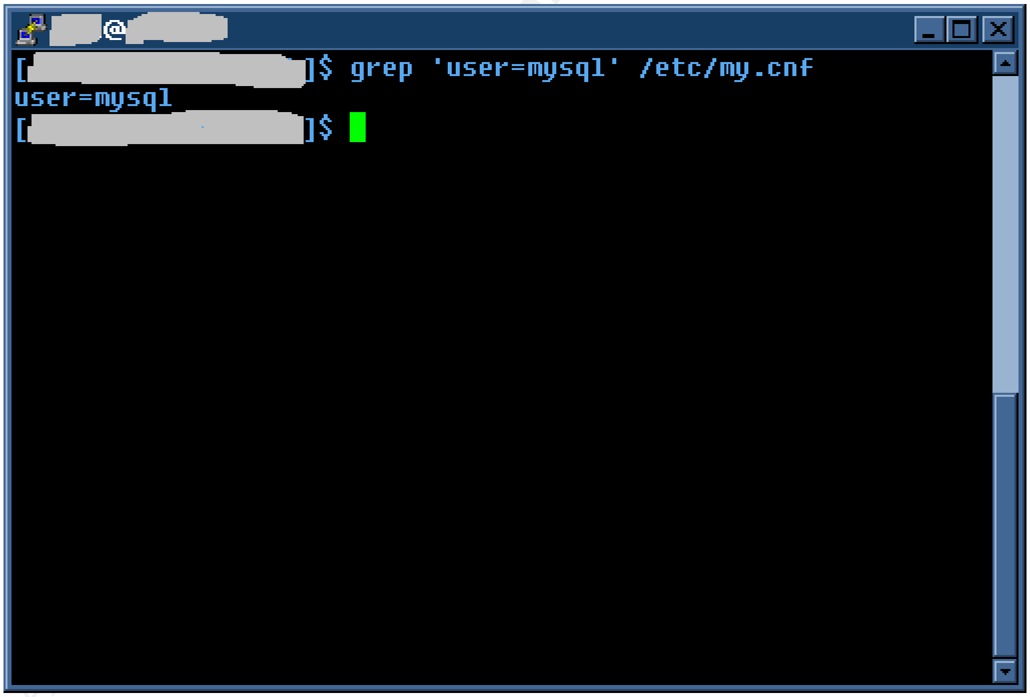
Reference	MySQL Manual: http://www.mysql.com/doc/en/Query_log.html
Control Objective	Record all connections and queries on the MySQL server to create an audit trail.
Risk	If no logs are available, there is no way to trace who is connecting to the server and what they are doing. An attacker can break in and go unnoticed.
Probability	High. Logging is not enabled by default.
Compliance	The server must be logging connections and queries in a general query log.
Testing	Check /etc/my.cnf to look for the following: log=/path/to/mysqld.log Also check the startup options to see if --log was enabled: ps auxwww | grep –i mysql If the log is located, verify that entries are being written in the log.
Objective/Subjective	Objective

Assignment 3: Audit Evidence

3.1 Conduct the Audit

Test 1

<table>
<tr><td>Audit Step</td><td>2 - Database directory permissions</td></tr>
<tr><td>Control Objective</td><td>Database directory only accessible to the user running the MySQL server.</td></tr>
<tr><td>Testing</td><td>Check the /etc/my.cnf file to make sure that ‘user=mysql’ is defined. The file was opened with pico to verify. For quick verification, here is the output from running “grep ‘user=mysql’ /etc/my.cnf”:

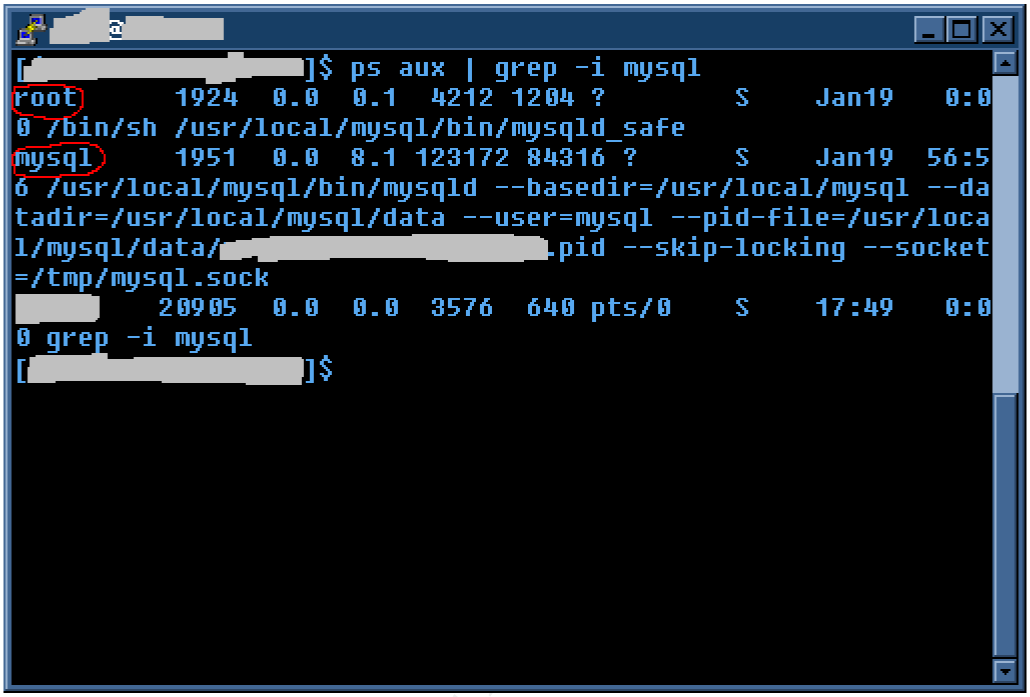


<pre>
[[redacted]]$ grep 'user=mysql' /etc/my.cnf
user=mysql
[[redacted]]$
</pre>

It is defined in the [mysql.server] section. The MySQL manual notes it is best defined in the [mysqld] section to designate the user no matter how the server is started.

Verify the server is actually running as the mysql user by viewing the current process list:

ps aux | grep -i mysql</td></tr>
</table>

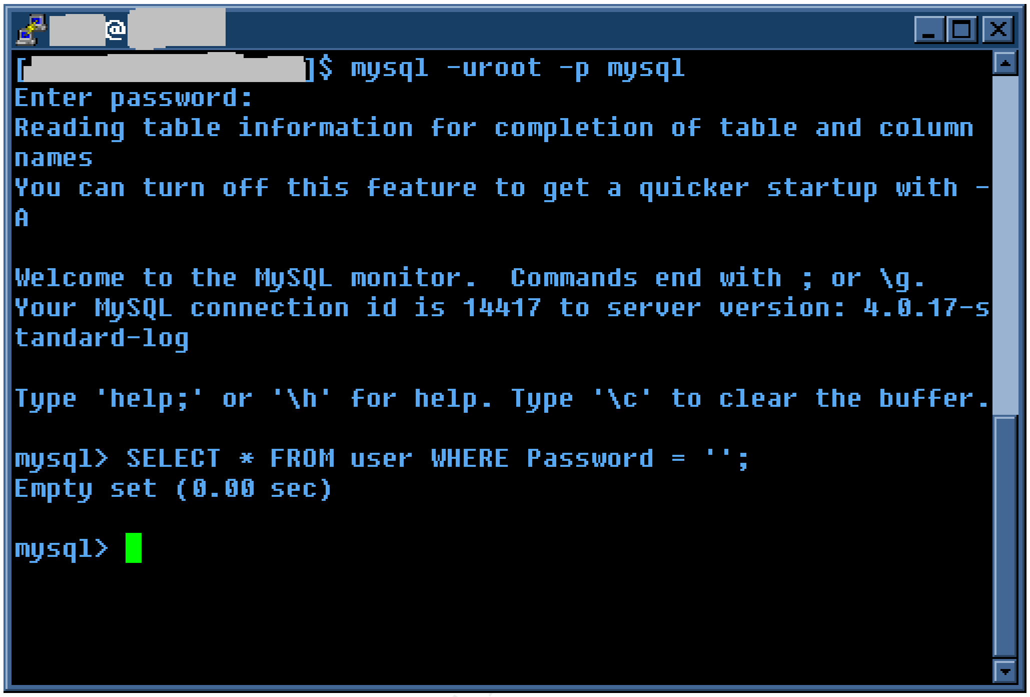


```
[[redacted]]$ ps aux | grep -i mysql
root      1924  0.0  0.1  4212 1204 ?        S    Jan19   0:0
0 /bin/sh /usr/local/mysql/bin/mysqld_safe
mysql     1951  0.0  8.1 123172 84316 ?      S    Jan19  56:5
6 /usr/local/mysql/bin/mysqld --basedir=/usr/local/mysql --da
tadir=/usr/local/mysql/data --user=mysql --pid-file=/usr/loca
l/mysql/data/[redacted].pid --skip-locking --socket
=/tmp/mysql.sock
[redacted]    20905  0.0  0.0  3576  640 pts/0    S    17:49   0:0
0 grep -i mysql
[[redacted]]$
```

Conclusion	Fail. Although mysqld is running as user ‘mysql’, the test showed that mysqld_safe is also running as user ‘root’. Also, the user=mysql should be moved from the [mysql.server] section to the [mysqld] section.

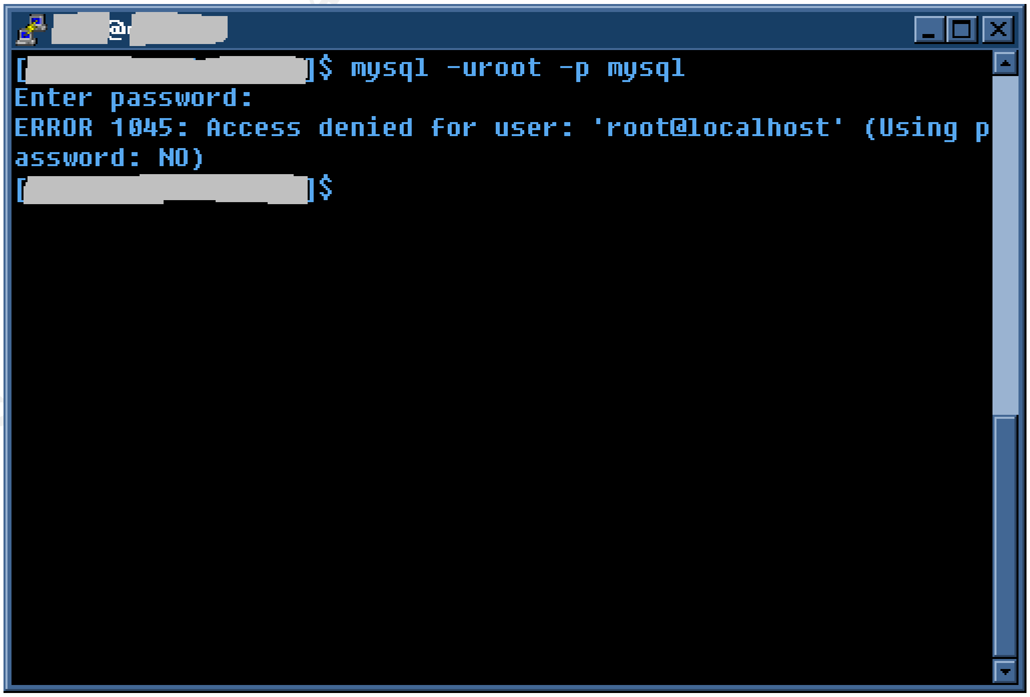
Test 2

Audit Step	5 - All accounts have passwords
Control Objective	Require a password to connect to MySQL.
Testing	Log into the MySQL client and run a query to see if any accounts have empty passwords: SELECT * FROM user WHERE Password = '';



```
[[redacted]]$ mysql -uroot -p mysql
Enter password:
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -
A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14417 to server version: 4.0.17-s
tandard-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT * FROM user WHERE Password = '';
Empty set (0.00 sec)

mysql> 
```

To verify that the current grant table is loaded and permissions are correct, try to log into MySQL with each username and no password. All login attempts were rejected. Here is an example screenshot of trying to login with the root account with no password:



```
[[redacted]]$ mysql -uroot -p mysql
Enter password:
ERROR 1045: Access denied for user: 'root@localhost' (Using p
assword: NO)
[[redacted]]$
```

Conclusion	Pass. All accounts required a password.

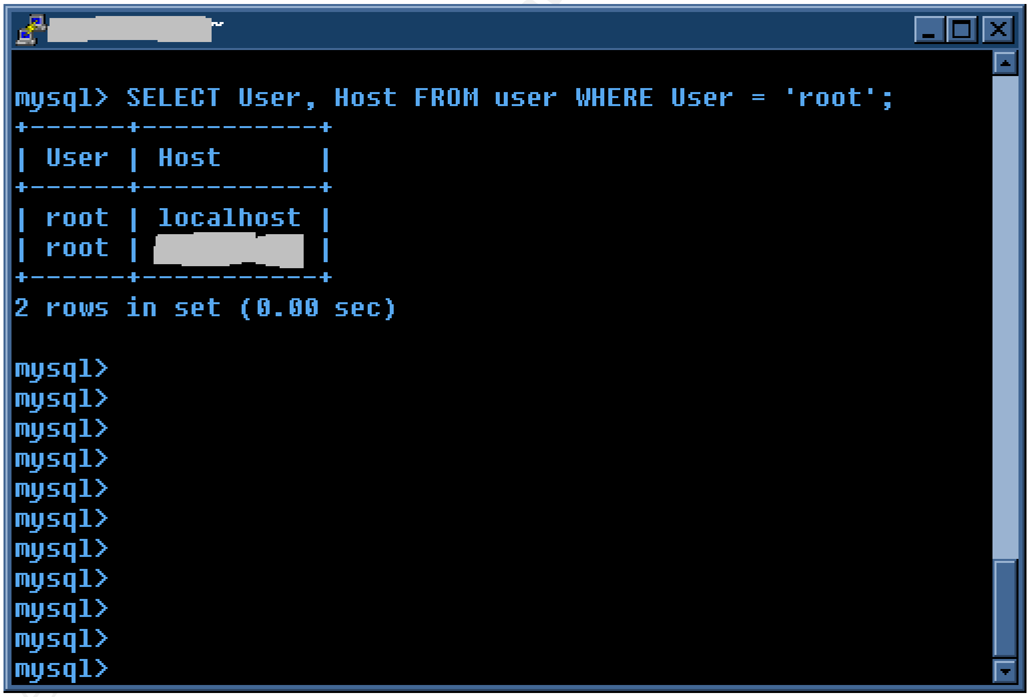
Test 3

<table>
<tr><td>Audit Step</td><td>6 – Rename root account</td></tr>
<tr><td>Control Objective</td><td>Rename the root account to harden the security.</td></tr>
<tr><td>Testing</td><td>

Run a query from the MySQL client to see if a root account exists.

SELECT User, Host FROM user WHERE User = 'root';

Zero rows should be returned. Here was the result:



```
mysql> SELECT User, Host FROM user WHERE User = 'root';
+------+-----------+
| User | Host      |
+------+-----------+
| root | localhost |
| root | [redacted] |
+------+-----------+
2 rows in set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

</td></tr>
<tr><td>Conclusion</td><td>Fail. The root account has not been renamed.</td></tr>
</table>

Test 4

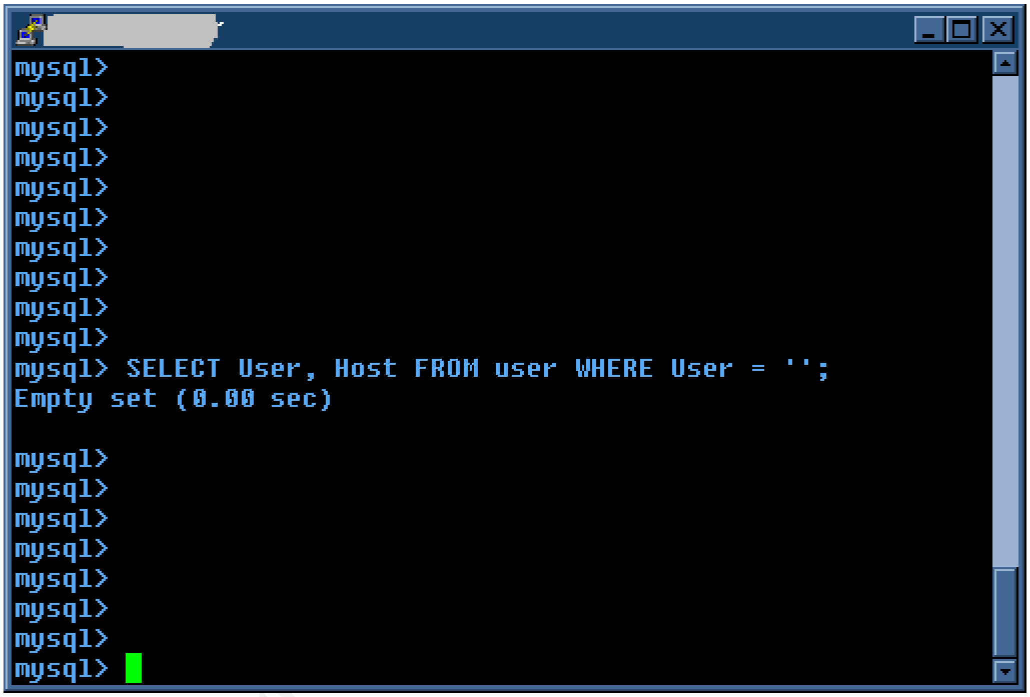
Audit Step	7 – No anonymous accounts
Control Objective	Require a username to connect to MySQL.

<table>
<tr><td>Testing</td><td>

Run a query from the MySQL client to see if any accounts have empty user fields:

SELECT User, Host FROM user WHERE User = '';

Zero rows should be returned. Here were the results:



[redacted]

```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SELECT User, Host FROM user WHERE User = '';
Empty set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

To make sure the grant tables were properly loaded, a test was administered to log in anonymously that failed.

</td></tr>
<tr><td>Conclusion</td><td>Pass. No anonymous users existed.</td></tr>
</table>

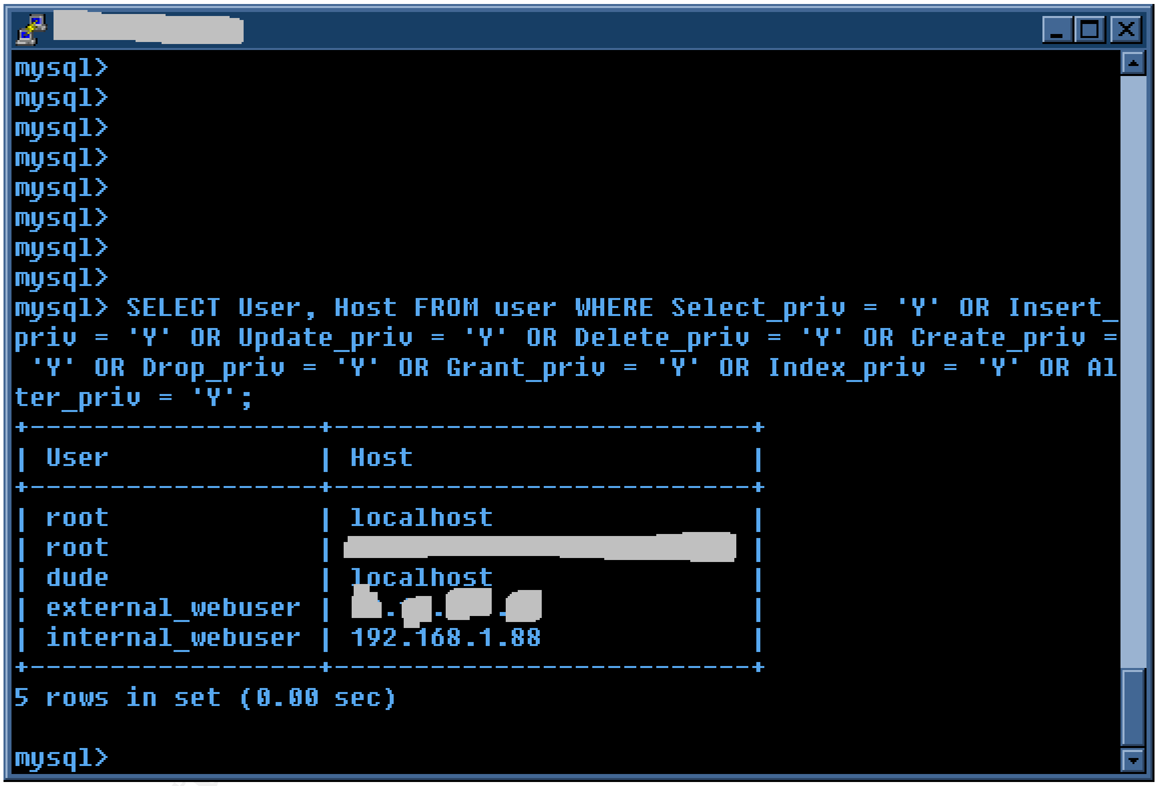
Test 5

Audit Step	9 – Unnecessary global privileges
Control Objective	Restrict user permissions to only the databases they need to connect to.
Testing	To determine what permissions the different users needed, ask the DBA. It was established that the root user needed all privileges, which is normal. It was also established that the external_webuser account only needs INSERT, UPDATE, SELECT and DELETE privileges on the Support database, as well as the internal_webuser account needing the same.

Now, run the following query against the database:

SELECT User, Host FROM user WHERE Select_priv = 'Y' OR Insert_priv = 'Y' OR Update_priv = 'Y' OR Delete_priv = 'Y' OR Create_priv = 'Y' OR Drop_priv = 'Y' OR Grant_priv = 'Y' OR Index_priv = 'Y' OR Alter_priv = 'Y';

The root user should be the only row returned. Here are the results:



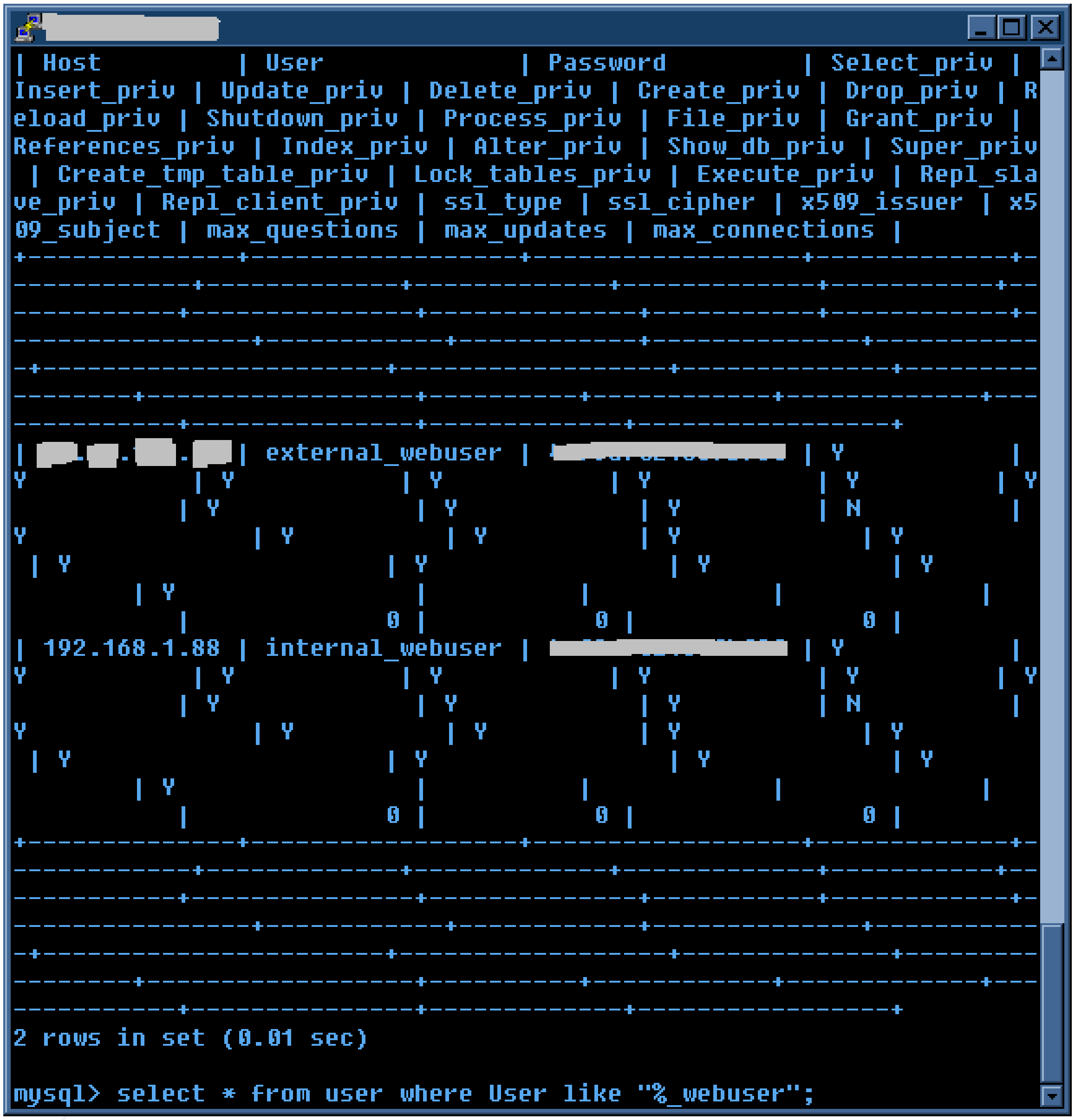
[redacted]

```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SELECT User, Host FROM user WHERE Select_priv = 'Y' OR Insert_
priv = 'Y' OR Update_priv = 'Y' OR Delete_priv = 'Y' OR Create_priv =
 'Y' OR Drop_priv = 'Y' OR Grant_priv = 'Y' OR Index_priv = 'Y' OR Al
ter_priv = 'Y';
+-------------------+---------------------------+
| User              | Host                      |
+-------------------+---------------------------+
| root              | localhost                 |
| root              | [redacted]                |
| dude              | localhost                 |
| external_webuser  | [redacted].[redacted].[redacted].[redacted] |
| internal_webuser  | 192.168.1.88              |
+-------------------+---------------------------+
5 rows in set (0.00 sec)

mysql>
```

The test showed that root, dude, external_webuser and internal_webuser have global privileges. After asking about the user ‘dude’, it was established that ‘dude’ was a valid account and is supposed to have global privileges. The two exceptions left are the webserver users. To see what privileges they have, the following query was run:

SELECT * FROM user WHERE User like "%_webuser";

The results showed:



[redacted]

```
| Host         | User             | Password         | Select_priv |
Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | R
eload_priv | Shutdown_priv | Process_priv | File_priv | Grant_priv |
References_priv | Index_priv | Alter_priv | Show_db_priv | Super_priv
 | Create_tmp_table_priv | Lock_tables_priv | Execute_priv | Repl_sla
ve_priv | Repl_client_priv | ssl_type | ssl_cipher | x509_issuer | x5
09_subject | max_questions | max_updates | max_connections |
+--------------+------------------+------------------+-------------+-
------------+-------------+-------------+-------------+-----------+--
-----------+---------------+--------------+-----------+------------+-
----------------+------------+------------+--------------+-----------
-+-----------------------+------------------+--------------+---------
--------+------------------+----------+------------+-------------+---
-----------+---------------+-------------+-----------------+
| [redacted] | external_webuser | [redacted] | Y           |
Y           | Y           | Y           | Y           | Y         | Y
           | Y             | Y            | Y         | N          |
Y               | Y          | Y          | Y            | Y
 | Y                     | Y                | Y            | Y
        | Y                |          |            |             |
           |             0 |           0 |               0 |
| 192.168.1.88 | internal_webuser | [redacted] | Y           |
Y           | Y           | Y           | Y           | Y         | Y
           | Y             | Y            | Y         | N          |
Y               | Y          | Y          | Y            | Y
 | Y                     | Y                | Y            | Y
        | Y                |          |            |             |
           |             0 |           0 |               0 |
+--------------+------------------+------------------+-------------+-
------------+-------------+-------------+-------------+-----------+--
-----------+---------------+--------------+-----------+------------+-
----------------+------------+------------+--------------+-----------
-+-----------------------+------------------+--------------+---------
--------+------------------+----------+------------+-------------+---
-----------+---------------+-------------+-----------------+
2 rows in set (0.01 sec)

mysql> select * from user where User like "%_webuser";
```

Conclusion	Fail. Two users, the webserver users, have been granted unnecessary global privileges. The test shows they were granted ALL privileges on ALL databases.

Test 6

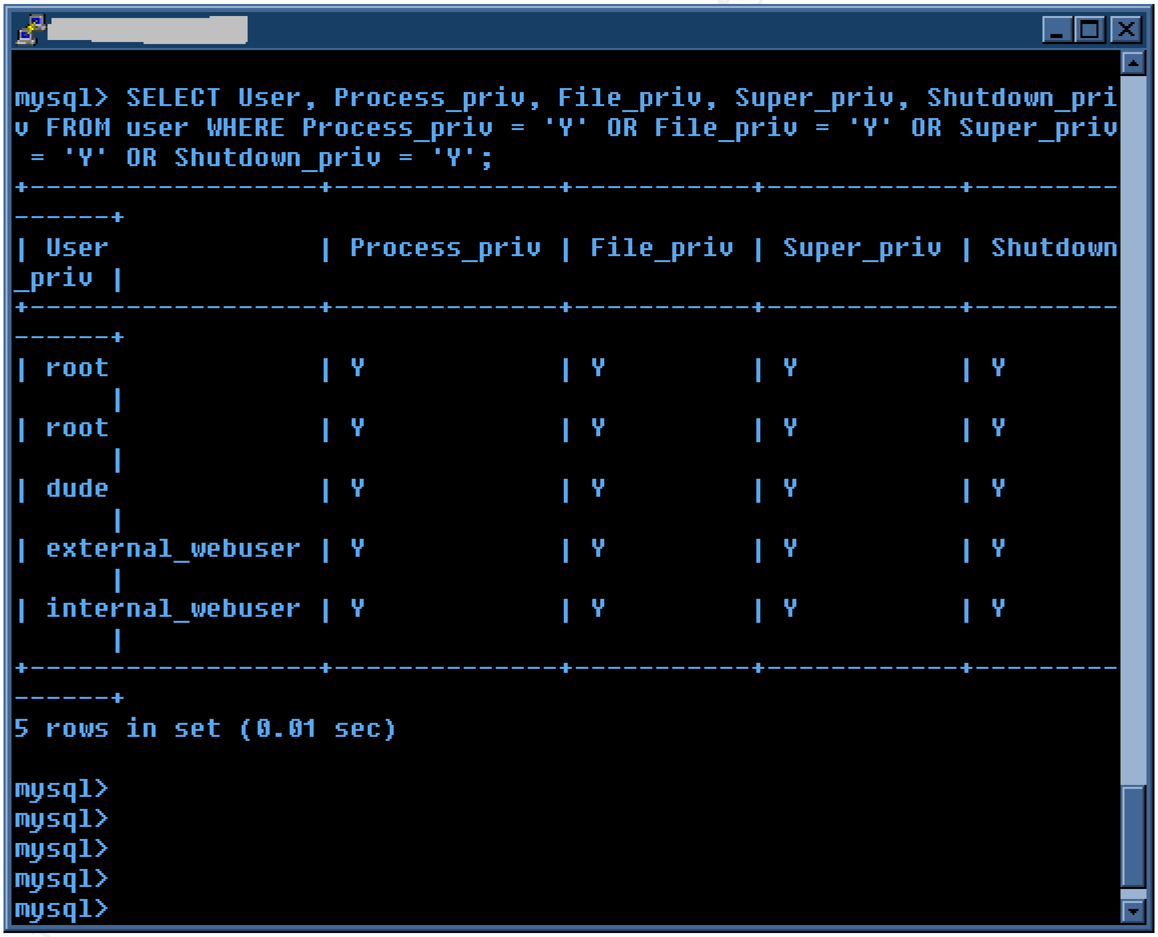
Audit Step	10 - Process, super, shutdown and file privileges
Control Objective	Restrict administrative privileges to only the admin user.

<table>
<tr><td>Testing</td><td>

Run the following query to determine which users have the administrative privileges PROCESS, SUPER, SHUTDOWN and FILE:

SELECT User, Process_priv, File_priv, Super_priv, Shutdown_priv FROM user WHERE Process_priv = 'Y' OR File_priv = 'Y' OR Super_priv = 'Y' OR Shutdown_priv = 'Y';

The test showed the following:



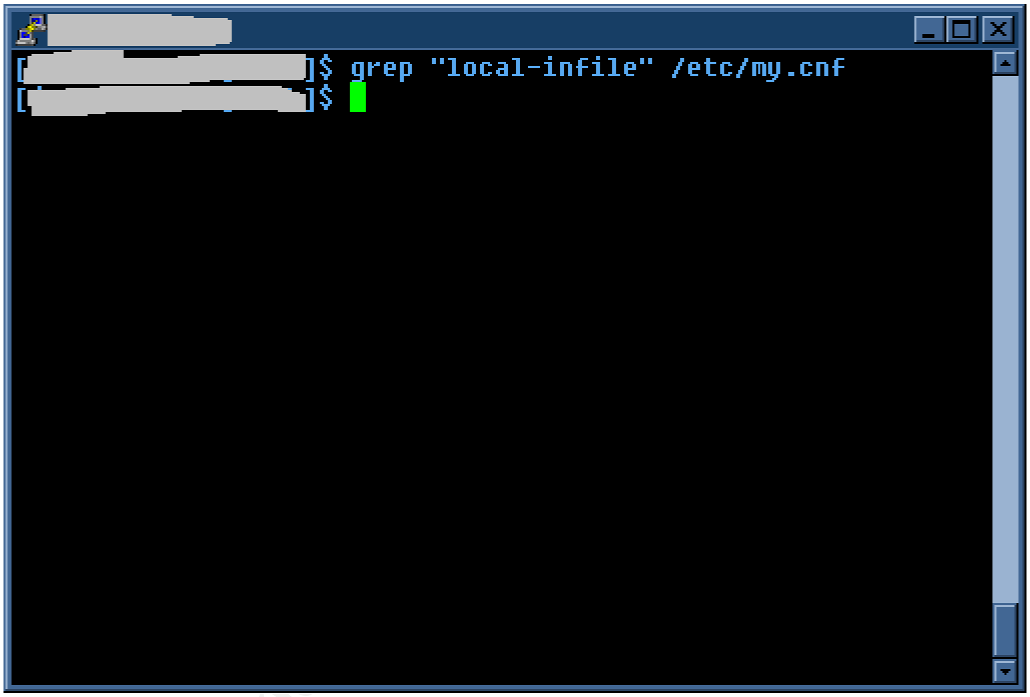
```
mysql> SELECT User, Process_priv, File_priv, Super_priv, Shutdown_pri
v FROM user WHERE Process_priv = 'Y' OR File_priv = 'Y' OR Super_priv
 = 'Y' OR Shutdown_priv = 'Y';
+-------------------+--------------+-----------+------------+---------
------+
| User              | Process_priv | File_priv | Super_priv | Shutdown
_priv |
+-------------------+--------------+-----------+------------+---------
------+
| root              | Y            | Y         | Y          | Y
      |
| root              | Y            | Y         | Y          | Y
      |
| dude              | Y            | Y         | Y          | Y
      |
| external_webuser  | Y            | Y         | Y          | Y
      |
| internal_webuser  | Y            | Y         | Y          | Y
      |
+-------------------+--------------+-----------+------------+---------
------+
5 rows in set (0.01 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
```

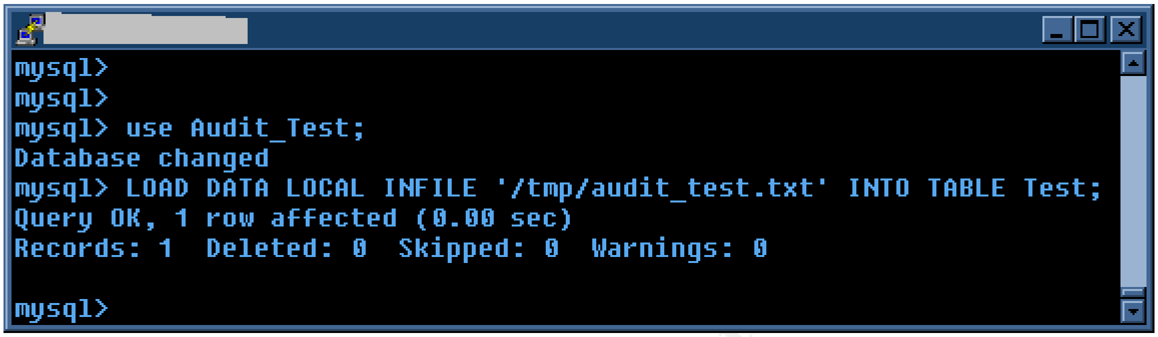
</td></tr>
<tr><td>Conclusion</td><td>Fail. It was determined in test 5 that root and dude are supposed to have global admin privileges, so ignore those. The two exceptions are the webserver users again.</td></tr>
</table>

Test 7

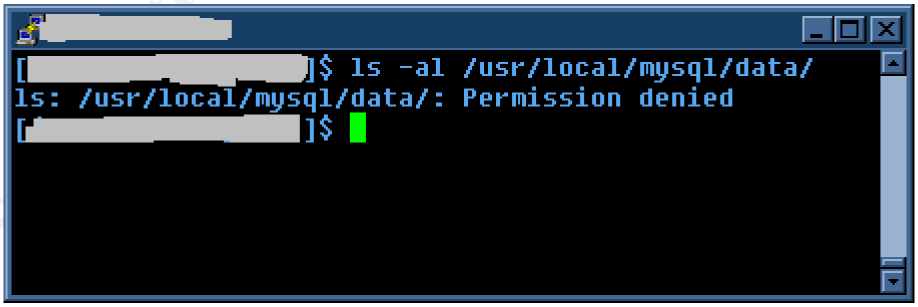
Audit Step	11 - Disable LOAD DATA LOCAL INFILE

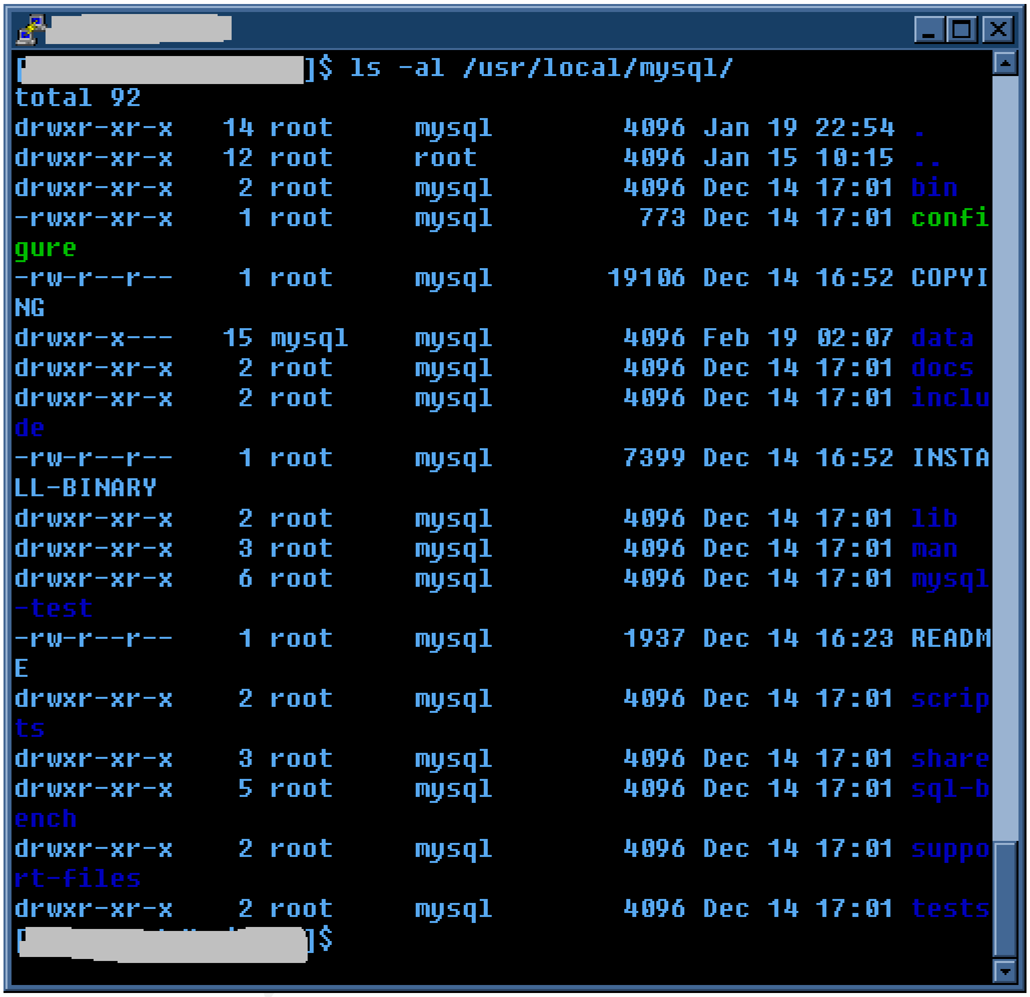
Control Objective	Disable LOAD DATA LOCAL INFILE to prevent reading of local files.

Testing	First, look in /etc/my.cnf to see if set-variable=local-infile=0. The file was checked with a text editor and the local-infile line was not found. A quick ‘grep “local-infile” /etc/my.cnf’ shows:
	
	[redacted]
	`[[redacted]]$ grep "local-infile" /etc/my.cnf`
	`[[redacted]]$`
	To test the command create a file in the /tmp directory that the mysql server can read:
	echo 1 > /tmp/audit_test.txt
	Next, create a database and table to import the file into:
	CREATE database Audit_Test;
	USE Audit_Test;
	CREATE TABLE Test (
	  ID char(1) NOT NULL default '',
	  PRIMARY KEY (ID)
	);
	Finally, run the command to see if an error is produced or if the import works:
	LOAD DATA LOCAL INFILE '/tmp/audit_test.txt' INTO TABLE Test;
	This test produced the following:

	 [redacted] mysql> mysql> mysql> use Audit_Test; Database changed mysql> LOAD DATA LOCAL INFILE '/tmp/audit_test.txt' INTO TABLE Test; Query OK, 1 row affected (0.00 sec) Records: 1 Deleted: 0 Skipped: 0 Warnings: 0 mysql>
Conclusion	Fail. Data was successfully loaded into the server, meaning LOAD DATA LOCAL INFILE is not disabled.

Test 8

Audit Step	2 – Database directory permissions
Control Objective	Database directory only accessible to the user running the MySQL server.
Testing	Locate the mysql data directory. The default is /usr/local/mysql/data when installed from source. View file ownership and permissions by typing: ls –al /usr/local/mysql/data The test produced the following:  [redacted] [[redacted]]$ ls -al /usr/local/mysql/data/ ls: /usr/local/mysql/data/: Permission denied [[redacted]]$ The results were good, however the ownership of the directory is still unknown. Try viewing up one directory level: ls –al /usr/local/mysql/data

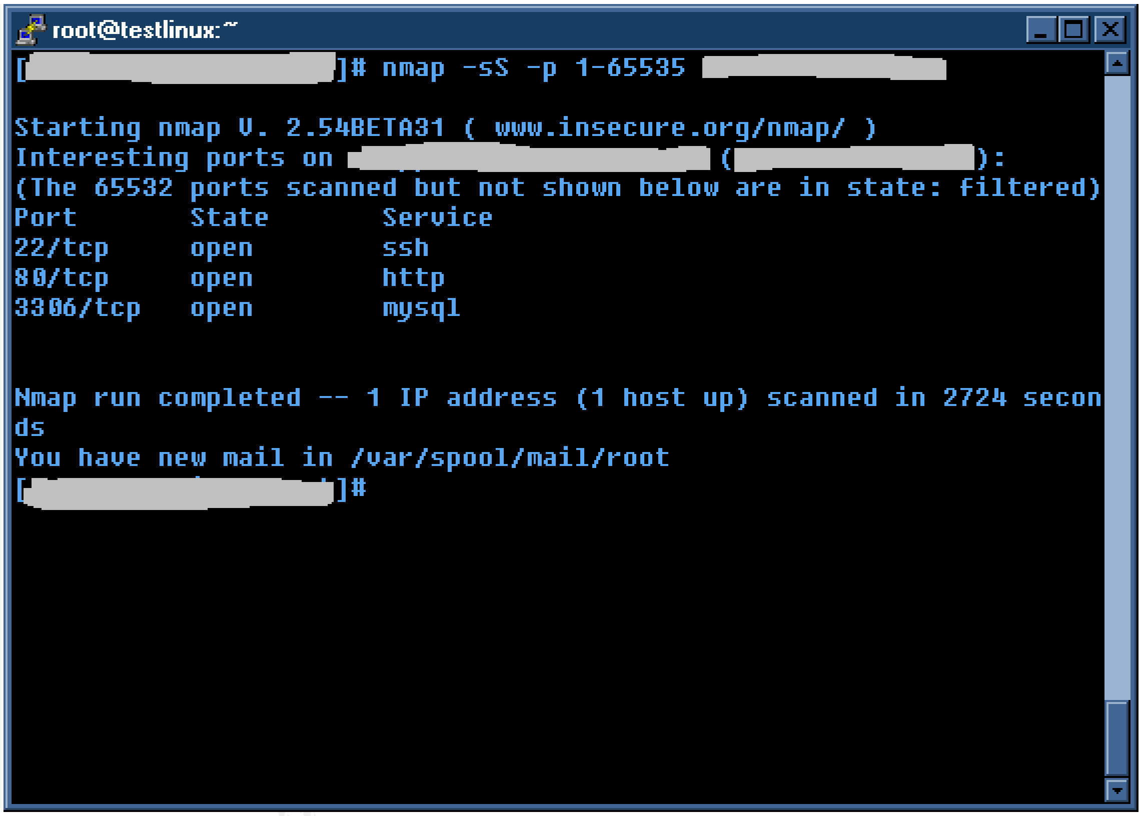
	
Conclusion	Pass. The results show the directory is owned by mysql.mysql and that nobody else has permissions. Trying to read or write a file in the directory failed.

```
[redacted]]$ ls -al /usr/local/mysql/
total 92
drwxr-xr-x   14 root     mysql        4096 Jan 19 22:54 .
drwxr-xr-x   12 root     root         4096 Jan 15 10:15 ..
drwxr-xr-x    2 root     mysql        4096 Dec 14 17:01 bin
-rwxr-xr-x    1 root     mysql         773 Dec 14 17:01 confi
gure
-rw-r--r--    1 root     mysql       19106 Dec 14 16:52 COPYI
NG
drwxr-x---   15 mysql    mysql        4096 Feb 19 02:07 data
drwxr-xr-x    2 root     mysql        4096 Dec 14 17:01 docs
drwxr-xr-x    2 root     mysql        4096 Dec 14 17:01 inclu
de
-rw-r--r--    1 root     mysql        7399 Dec 14 16:52 INSTA
LL-BINARY
drwxr-xr-x    2 root     mysql        4096 Dec 14 17:01 lib
drwxr-xr-x    3 root     mysql        4096 Dec 14 17:01 man
drwxr-xr-x    6 root     mysql        4096 Dec 14 17:01 mysql
-test
-rw-r--r--    1 root     mysql        1937 Dec 14 16:23 READM
E
drwxr-xr-x    2 root     mysql        4096 Dec 14 17:01 scrip
ts
drwxr-xr-x    3 root     mysql        4096 Dec 14 17:01 share
drwxr-xr-x    5 root     mysql        4096 Dec 14 17:01 sql-b
ench
drwxr-xr-x    2 root     mysql        4096 Dec 14 17:01 suppo
rt-files
drwxr-xr-x    2 root     mysql        4096 Dec 14 17:01 tests
[redacted]]$
```

Test 9

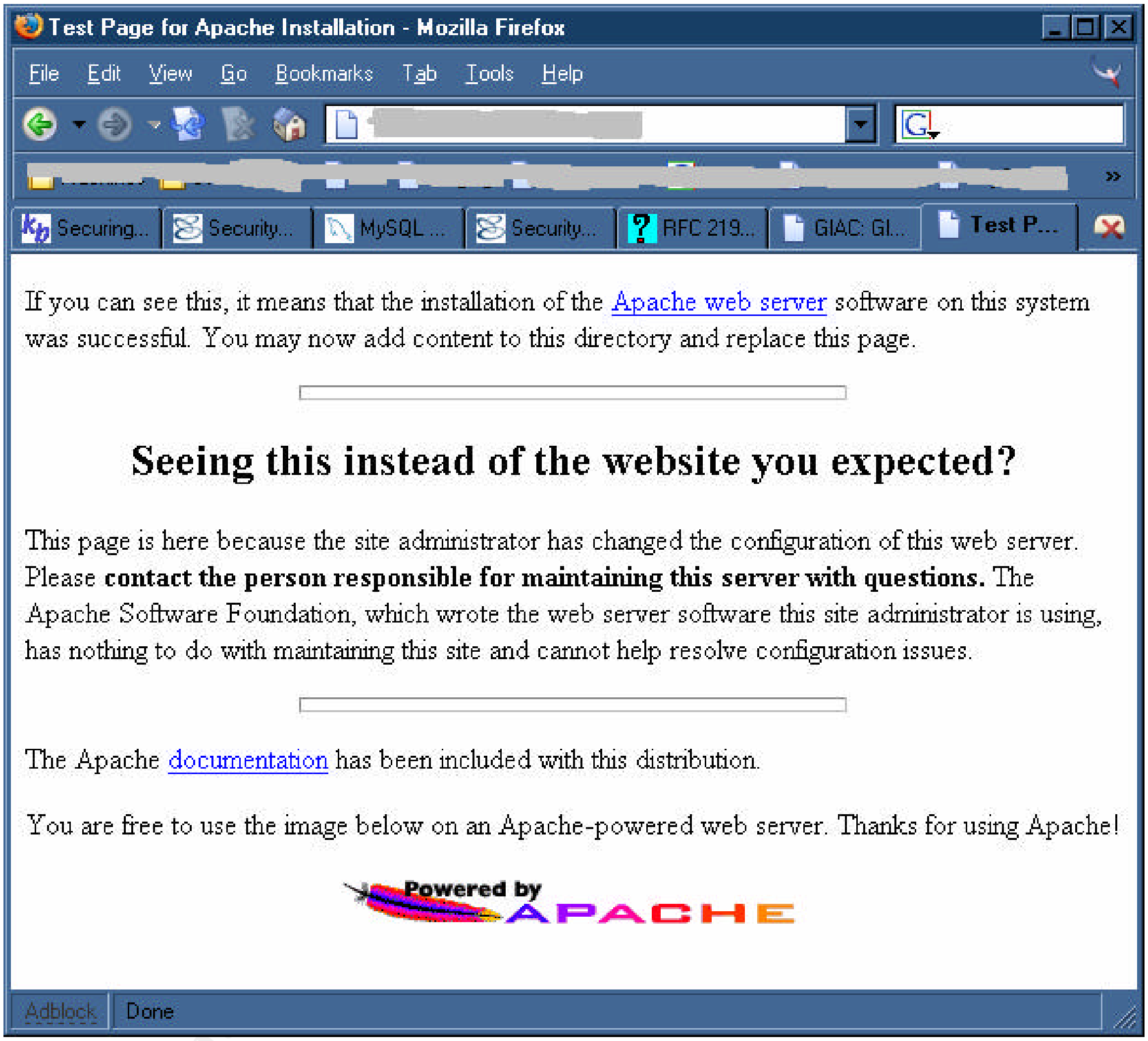
Audit Step	19 – Port filtering
Control Objective	Restrict the ports that can be accessed to the bare minimum, SSH and MySQL.
Testing	Using nmap, test the database server from a remote machine with a full port scan. The following scan was run from both webservers: nmap –sS –p 1- <database ip>

Here are the results from the scan:



```
[[redacted]]# nmap -sS -p 1-65535 [redacted]

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on [redacted] ([redacted]):
(The 65532 ports scanned but not shown below are in state: filtered)
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
3306/tcp   open        mysql


Nmap run completed -- 1 IP address (1 host up) scanned in 2724 secon
ds
You have new mail in /var/spool/mail/root
[[redacted]]#
```

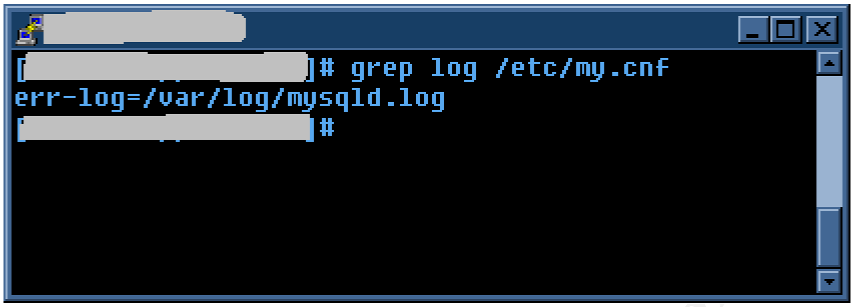
Bringing up the server in a webserver shows that Apache is running on database server.

	 Test Page for Apache Installation - Mozilla Firefox File Edit View Go Bookmarks Tab Tools Help [redacted] [redacted] Securing... Security... MySQL ... Security... RFC 219... GIAC: GI... Test P... Powered by APACHE Adblock Done If you can see this, it means that the installation of the Apache web server software on this system was successful. You may now add content to this directory and replace this page. **Seeing this instead of the website you expected?** This page is here because the site administrator has changed the configuration of this web server. Please **contact the person responsible for maintaining this server with questions.** The Apache Software Foundation, which wrote the web server software this site administrator is using, has nothing to do with maintaining this site and cannot help resolve configuration issues. The Apache documentation has been included with this distribution. You are free to use the image below on an Apache-powered web server. Thanks for using Apache!
Conclusion	Fail. In addition to the allowed ports, a webserver is running on the database server.

Test 10

Audit Step	20 – Log files
Control Objective	Record all connections and queries on the MySQL server to create an audit trail.
Testing	Check /etc/my.cnf to look for the following: log=/path/to/mysqld.log Viewing the file with pico shows that log isn't defined. A quick grep

shows:

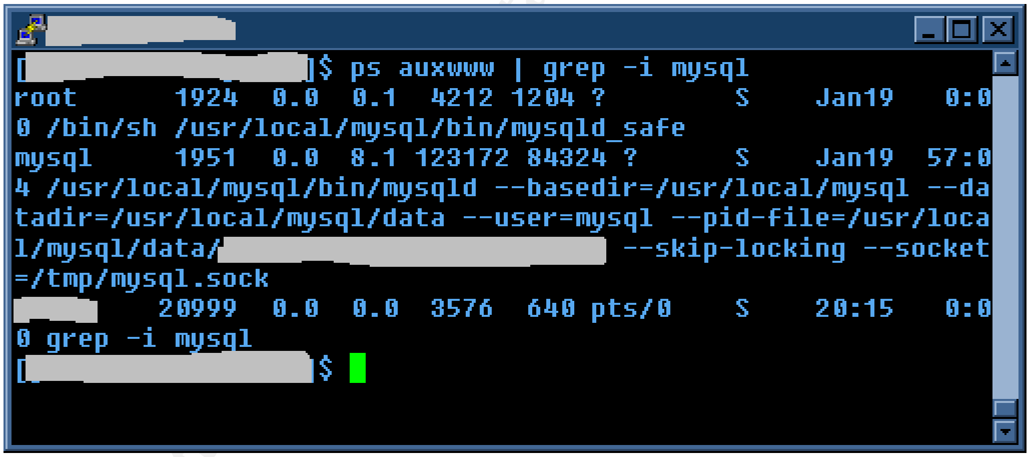


```
[[redacted]]# grep log /etc/my.cnf
err-log=/var/log/mysqld.log
[[redacted]]#
```

Check the startup options to see if --log was enabled:

ps auxwww | grep –i mysql

The results showed that –log wasn't defined:



```
[[redacted]]$ ps auxwww | grep -i mysql
root       1924  0.0  0.1  4212 1204 ?       S    Jan19   0:0
0 /bin/sh /usr/local/mysql/bin/mysqld_safe
mysql      1951  0.0  8.1 123172 84324 ?     S    Jan19  57:0
4 /usr/local/mysql/bin/mysqld --basedir=/usr/local/mysql --da
tadir=/usr/local/mysql/data --user=mysql --pid-file=/usr/loca
l/mysql/data/[redacted] --skip-locking --socket
=/tmp/mysql.sock
[redacted]     20999  0.0  0.0  3576  640 pts/0    S    20:15   0:0
0 grep -i mysql
[[redacted]]$
```

Conclusion	Fail. The general query log is not enabled.

3.2 Measure Residual Risk

A full audit of Compuglobalhypermeganet's MySQL database server established that a high level of residual risk still exists. The server failed to meet the control objectives stated in the checklist, as seven exceptions were discovered. Most of the exceptions found dealt with the configuration of MySQL and excessive privileges in the database. In general, the operating system itself, with one exception, and the policies and procedures regarding the server were top notch. The following exceptions were found:

Test 1 showed that there is one instance of the mysql server running as root. A clever attacker can use this exception to read or create system files as root leading to a system compromise or data leak. The global option file /etc/my.cnf should be edited to move the user=mysql from the [mysql.server] section to the [mysqld] sectionso the user is declared no matter how the server is started. The mysqld_safe process should be killed. The cost of making these changes to the server are minimal.

Test 3 revealed that the administrative account had not been renamed. This makes the root account more susceptible to a dictionary or brute-force attack since the username is known. This can be easily changed by updating the privilege tables in MySQL and flushing the grant tables. The cost of making this change to the server is minimal.

Test 5 shows that several users have been given unnecessary global privileges. A global privilege can be applied across all databases, not just the one(s) intended. This creates a higher risk that someone will view or edit something they are not supposed to. This exception can be mitigated by correcting the privileges in the MySQL and flushing the grant tables. The cost of making these changes to the server are minimal.

Test 6 showed that several unauthorized users have the administrative privileges of FILE, SUPER, SHUTDOWN and PROCESS. Again, the users are the webserver users. This means that the webserver users can execute these administrative commands from the webserver machines. This increases the risk of a remote attack on the database server. This is another easy fix that simply consists of altering the privilege tables and flushing the grant tables. The cost of making these changes to the server are minimal.

Test 7 revealed that the LOAD DATA LOCAL INFILE command could be executed on the server. There poses two main risks to consider. One, the user could read any file that the webserver has access to read. Two, the server could access any file on the client host (the webserver) to which the client user has read access. This problem can be fixed by making an addition to the global options file, /etc/my.cnf and restarting the database service. The cost of making these changes to the server are minimal.

Test 9 shows that in addition to SSH and MySQL, an Apache webserver is listening on port 80. This increase the possibility of compromising the server by using published vulnerabilities by adding Apache to the list. At a minimum, the apache service should be disabled and the port in the firewall closed. The cost for fixing this exception is minimal.

Finally, test 10 showed that MySQL is not logging connections and queries to a general query log. Without a general query log, there is no way to trace who is connecting to the server and what they are doing. An attacker can break in and go unnoticed. This exception can be mitigated by making an addition to the global options file, /etc/my.cnf and restarting the database service.

3.2 Is the System Auditable?

The MySQL database server is an auditable system. However, test 10 showed that a general query log is not enabled in the configuration on the server. Unfortunately, this is the default configuration for MySQL. Since a log doesn't exist, it is impossible to determine if users were inappropriately accessing data they should not have been or performing any harmful activities to the server. Other than this exception, all items can be audited against successfully in order to check compliance.

Assignment 4: Audit Report

4.1 Executive Summary

This audit focused on the security of the MySQL version 4.17 database server located on a Red Hat 7.3 linux server. The database stores sensitive client information obtained through client support requests. The purpose of the audit was to measure the current level of security on the server as compared to industry best practices by using an audit checklist. The overall goal was to improve the security of the system as a whole. The focus of the audit was the MySQL database server itself, not the webservers attached to it.

Careful analysis of the server showed that in general, the operating system itself, with one exception, and the policies and procedures regarding the server were found to be top notch. The server was fully patched with proper procedures in place to keep it this way. Backup and restore policies were also found to be flawless minimizing the possible downtime and/or data loss that could be caused if the server crashed or was compromised.

The analysis also revealed that several vulnerablities existed with the MySQL server's configuration and current privilege system that could lead to the system being compromised or the data being viewed or altered by an unauthorized person. Due to a lack of system logging, there was no way to determine if a security breach has already occurred. All of the exceptions found require minimal effort and cost to mitigate, but show there is a lack of understanding of some basic MySQL principals and security. Detection and response mechanisms should be created and adhered to once logging is enabled.

4.2 Audit Findings

The following items were discovered during the audit that do not comply with the control objectives:

Finding 1 - Unprivileged account running MySQL
Risk: High
Reference: Checklist Item #1 - page 07, Test #1 – page 23

One instance of the mysql server was found to be running as the root user of the system. The root user of the system has privileges to do anything on the machine. This process was probably started accidently, but is easily preventable.

Background/Risks

A clever attacker can use this exception to read or create system files as root, leading to several possible problems. One is the ability to view, modify or delete the data stored in the database, making it completely useless. Another would be a complete system compromise in which the attacker could use the database machine to stage additional attacks.

Recommendations

The global option file for the mysql server should be edited to prevent this from happening again. The unsafe process that was running was killed by the system administrator upon discovery.

Finding 2 – Root account not renamed

Risk: High
Reference: Checklist Item #6 - page 10, Test #3 – page 26

The default administrative account for the database has a username of 'root'. This is a commonly known fact so the administrative account should be renamed so outsiders will not know the name of the admin account.

Background/Risks

Knowing the name of the admin account solves half of the login puzzle. The other half of the puzzle is guessing the password. This is much harder to do if you don't know the name of the account. Password guessing is done by automated methods such as dictionary or brute-force attacks. If the attacker is able to gain access to the database as the root user, data confidentiality and integrity is lost. The data may become useless or deleted. The MySQL server could be shutdown and the attacker could possibly use this in combination with other exceptions to take control of the machine.

Recommendations

This can be easily changed by renaming the account in the MySQL privilege system.

Finding 3 – Unnecessary global privileges

Risk: High
Reference: Checklist Item #9 - page 12, Test #5 – page 27

MySQL allows you to grant privileges to specific databases, tables, fields or globally across the database. Users should be given the least amount of permissions required to the specific database to do the job.

Background/Risks

Several users have been given unnecessary global privileges, allowing them to edit, delete, insert and update data in any database. This creates a higher risk that someone will view or edit something they are not supposed to or even lock out other users by editing the privilege tables in MySQL. This is especially important because the users are allowed to do this from the webserver machines. Again, data confidentiality and integrity are put on the line.

Recommendations

This exception can be mitigated by correcting the privileges in the MySQL and flushing the grant tables. The users should be granted privileges to the specific database that they need only.

Finding 4 – Unnecessary admin privileges
Risk: High
Reference: Checklist Item #10 - page 13, Test #6 – page 29

Several privileges should be reserved for administrators only. The PROCESS privilege gives a MySQL user the ability to see the currently executing queries. The FILE privilege will allow a user to read files from and write files to the filesystem. The SHUTDOWN privilege will allow a user to shut down the database. The SUPER privilege allows a user to terminate client connections and change system variables. Several unauthorized users have the administrative privileges of FILE, SUPER, SHUTDOWN and PROCESS.

Background/Risks

Again, the users are the webserver users. This means that the webserver users can execute these administrative commands from the webserver machines. This increases the risk of a remote attack on the database server. Risks include: Exposing passwords if a password is being set, enabling the attacker to gain a new password, shutting down the database service, terminating client connections and gaining access to the filesystem of the server, possibly as root. Potential outcomes include the system being compromised, data integrity and data confidentiality.

Recommendations

This is another easy fix that simply consists of altering the privilege tables and flushing the grant tables. Only grant the privileges necessary to do the job.

Finding 5 – LOAD DATA LOCAL INFILE option enabled
Risk: High
Reference: Checklist Item #11 - page 14, Test #7 – page 30

LOAD DATA LOCAL INFILE command can load a file that is located on the server host, or it can load a file that is located on the client host.

Background/Risks

There poses two main risks to consider. One, the user could read any file that the webserver has access to read. Two, the server could access any file on the client host (the webserver) to which the client user has read access. This could expose information on the webserver or the database server again leading to a system compromise.

Recommendations

This problem can be fixed by making an addition to the global options file, /etc/my.cnf and restarting the database service.

Finding 6 – Port filtering
Risk: High
Reference: Checklist Item #19 - page 21, Test #9 – page 33

A server is always more secure if it is only running the absolute necessary services it needs to do its job. This provides a decreased number of attack vectors based on published vulnerabilities. Test 9 shows that in addition to SSH and MySQL, an Apache webserver is listening on port 80.

Background/Risks

This increases the possibility of compromising the server by using published vulnerabilities by adding Apache to the list. If an exploit is found for apache the server could be attacked and possibly compromised depending on the exploit. Once compromised the attacker could delete the data, change the data, view the data, stop the server or attack other machines

Recommendations

The apache service should be disabled and the port in the firewall closed. Regular scans of the machine should take place to prevent unnecessary services from being turned on in the future.

Finding 7 – No log file
Risk: High
Reference: Checklist Item #20 - page 22, Test #10 – page 35

A general query log stores connection information as well as queries. This allows the server to be audited to see if users are connecting that are not supposed to or if unauthorized changes or data views have occurred.

Background/Risks

Without a general query log, there is no way to trace who is connecting to the server and what they are doing. An attacker can break in and go unnoticed.

Recommendations

Turn on logging by making an addition to the global options file and restarting the database service. Detection and response mechanisms should be created and adhered to once logging is enabled.

4.3 Costs

The costs to fix the exceptions are mainly just minimal manpower and time to implement and test the changes. Server or hardware costs are not necessary to make the changes.

- Time to fix current exception list: approximately 8 hours

MySQL database training is recommended to bring the administrator up to par on general security guidelines and the MySQL privilege system.

- Five day 'Using and Managing MySQL' training class: $2450 plus travel expenses

Ongoing weekly port scans and daily logfile monitoring should continue to minimize risks to the server. Also continue to manage server patches.

- Logfile monitoring: 30 minutes/day
- Port scanning: 30 minutes/week
- Patch management: 5 hours/month

4.4 Compensating Controls

With the exception of the MySQL training, all of the recommendations made within this report can be made by the system administrator/DBA at a minimal costs that should be considered normal operational costs.

References

- “MySQL Manual”. MySQL. 2004. URL: http://www.mysql.com/doc/en/MySQL_Database_Administration.html

- “Secure MySQL Database Design”. Kristy Westphal. 18 February 2003. URL: http://www.securityfocus.com/infocus/1667

- “Securing MySQL: step-by-step”. Artur Maj. 28 August 2003. URL: http://www.securityfocus.com/infocus/1726

- “Securing Your MySQL Installation”, Document revision: 1.01. Paul DuBois. 25 January 2003 http://www.kitebird.com/articles/ins-sec.html

- “RFC 2196 - Site Security Handbook”. B. Fraser. September 1997. URL: http://www.faqs.org/rfcs/rfc2196.html

- SANS/GIAC Student Practicals. http://www.giac.org/GSNA.php

- SANS Course Materials