$content = "Failure";

}

return $this->renderContent($content);

}

public function actionSimple()

{

$userName = Yii::$app->request->get('username');

$password = Yii::$app->request->get('password');

$passwordHash = md5($password);

$sql = "SELECT \* FROM 'user'"

." WHERE 'username' = '".$userName."'"

." AND password = '".$passwordHash."' LIMIT |1";

$result = Yii::$app->db->createCommand($sql)->queryOne();

return $this->renderContentByResult($result);

}

}

2. Let’s try to access it using the /sql/simple?username=test&password=test URL. As we are unaware of both the username and password, it will, as expected, print Failure.

3. Now try /sql/simple?username=%27+or+%271%27%3D%271%27%3B+--&password=whatever. This time, it lets us in, even though we still don’t know anything about the actual credentials. The decoded part of usernamevalue looks like the following:

' or '1'='1'; -

4. Close the quote so that the syntax stays correct. Add or 'i'='i', which makes the condition always true. Use ; -- to end the query and comment the rest.

5. As no escaping was done, the whole query executed was:

SELECT \* FROM user WHERE username = '' or '1'='1'; --' AND password =

'008c5926ca861023c1d2a36653fd88e2' LIMIT 1;

6. The best way to fix this is to use a prepared statement, as follows:

public function actionPrepared()

{

$userName = Yii::$app->request->get('username');

$password = Yii::$app->request->get('password');

$passwordHash = md5($password);

$sql = "SELECT \* FROM 'user'"

." WHERE 'username' = :username"

." AND password = :password LIMIT 1";

$command = Yii::$app->db->createCommand($sql);

$command->bindValue(':username', $userName);

$command->bindValue(':password', $passwordHash);

$result = $command->queryOne();

return $this->renderContentByResult($result);

}

7. Now check /sql/prepared with the same malicious parameters. This time everything was fine and