Hacking DBs (SQL injection)

Modul 9 (uke 7) innhold

The Injection security threat

- Injection is a HUGE security threat!

- According to OWASP, as of 2021 "Injection" is the 3rd-largest application security risk. (And it was the number-1 risk in 2017.)

- Notable weakness types included in "Injection" are:
  - SQL injection
  - Cross-site Scripting
  - External Control of File Name or Path

Get into the correct developer mindset

- The injection problem starts with software developers expecting application users to be intelligent and nice guys.
  - That is an absolutely wrong expectation!

- Your software users are, at the very best, idiots!
  - Don't except a user to input a number, just because an input box says "Phoner number".

- And idiots are not the scary ones. Some of the users are EVIL!
  - They are trying to hurt you (well, your application).

Being the GOOD guys

- So why am I teaching you how to become a malicious hacker?
  - I am not!
  - Well, I kind of am. :-\ But that's just the (unwanted) bi-product.

- I'm teaching you what the SQL injection security threat is.
  - That way you can take measures that keeps your applications and your databases safe from SQL injection threats. :-)

- We are the GOOD guys! (Right?)
  - Our intention is Ethical hacking. (Being "white hat" hackers.)

SQL injection, possibilities

- SQL injection is the process of adding ("injecting") extra SQL code into an application's DB-statements.
  - This is done by "creative" use of the input-fields.
  - Resulting in the program's behaviour altered from what the developer intended.

- Some example uses:
  - Logging in without knowing the correct password. (Also without knowing the username, if you like.)
  - Seeing (stealing) hidden content. (Like personal info for other users, etc…)
  - Changing existing content. (For example, giving yourself better grades, extra shopping credits, …)
  - Deleting content. (All or just some of it. At any time you're asked to input a value, like name, item category or a search word, you could delete the whole DB.)

SQL injection, historical examples

- Here are some examples of real-world, malicious SQL injection. These attacks could have been prevented with better security knowledge on the software developers' end!
  - Hackers targeted 53 universities using SQL injection and stole and published 36,000 personal records belonging to students, faculty, and staff.
  - Hackers used SQL injection to breach the Turkish government website and erase debt to government agencies.
  - A team of attackers used SQL injection to penetrate corporate systems at several companies, primarily the 7-Eleven retail chain, stealing 130 million credit card numbers.

- Here are a couple of discovered SQL injection vulnerabilities:
  1. Fortnite is an online game with over 350 million users. In 2019, a SQL injection vulnerability was discovered which could let attackers access user accounts.
  2. In 2014, security researchers publicized that they were able to breach the website of Tesla using SQL injection, gain administrative privileges and steal user data.

Source: SQL Injection Attack: Real Life Attacks and Code Examples

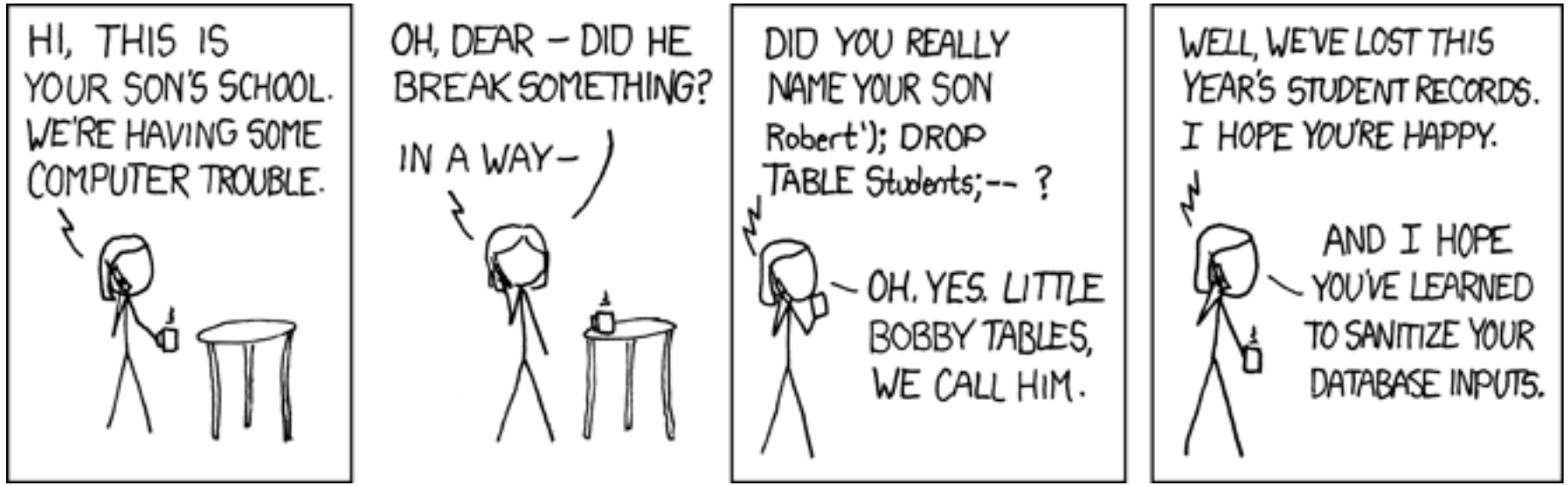
SQL injection, the technical stuff

- For those attending the lesson:
  - I'll now show a live demo of how SQL injection works.

- *Note:* I'm NOT hacking a real site!
  - I'm hacking my own "sandbox-site for showcasing SQL injections". :-)

SQL Injection som … vitsetegning?

- What's going on here? :-D



- Think: `INSERT INTO Students (Name) VALUES ('text (weird name) above');`
  - In other words, this as name: `Robert'); DROP TABLE Students;--`

SQL injection: How it happens

- Lets say we have the following SQL statement as a C# string:

```
"INSERT INTO Students (name) VALUES ('{name}');"
```

- Then, a malicious user writes the following "name" as input:

```
"Robert'); DROP TABLE Students; -- "
```

- How will this look inserted into the SQL? Will be as follows:

```
INSERT INTO Students (name) VALUES ('Robert');
DROP TABLE Students;
-- ');
```

SQL Injection: Secure your query input!

- NOTE: This is C# specific. *You* must find (google) the corresponding functionality for JavaScript.

- Do NOT write your SQL strings like this:

```
string name = "Robert'); DROP TABLE Students; -- ";
sqlCommand.CommandText = $@"INSERT INTO Students (name) VALUES ('{name}')";
```

- The correct way, using `Parameters`:
  - (Must create an SqlCommand object as well, not in this example.)

```
string name = "Robert'); DROP TABLE Students; -- ";
sqlCommand.CommandText = "INSERT INTO Students (name) VALUES (@name)";
sqlCommand.Parameters.AddWithValue("@name", name);
```


Academy