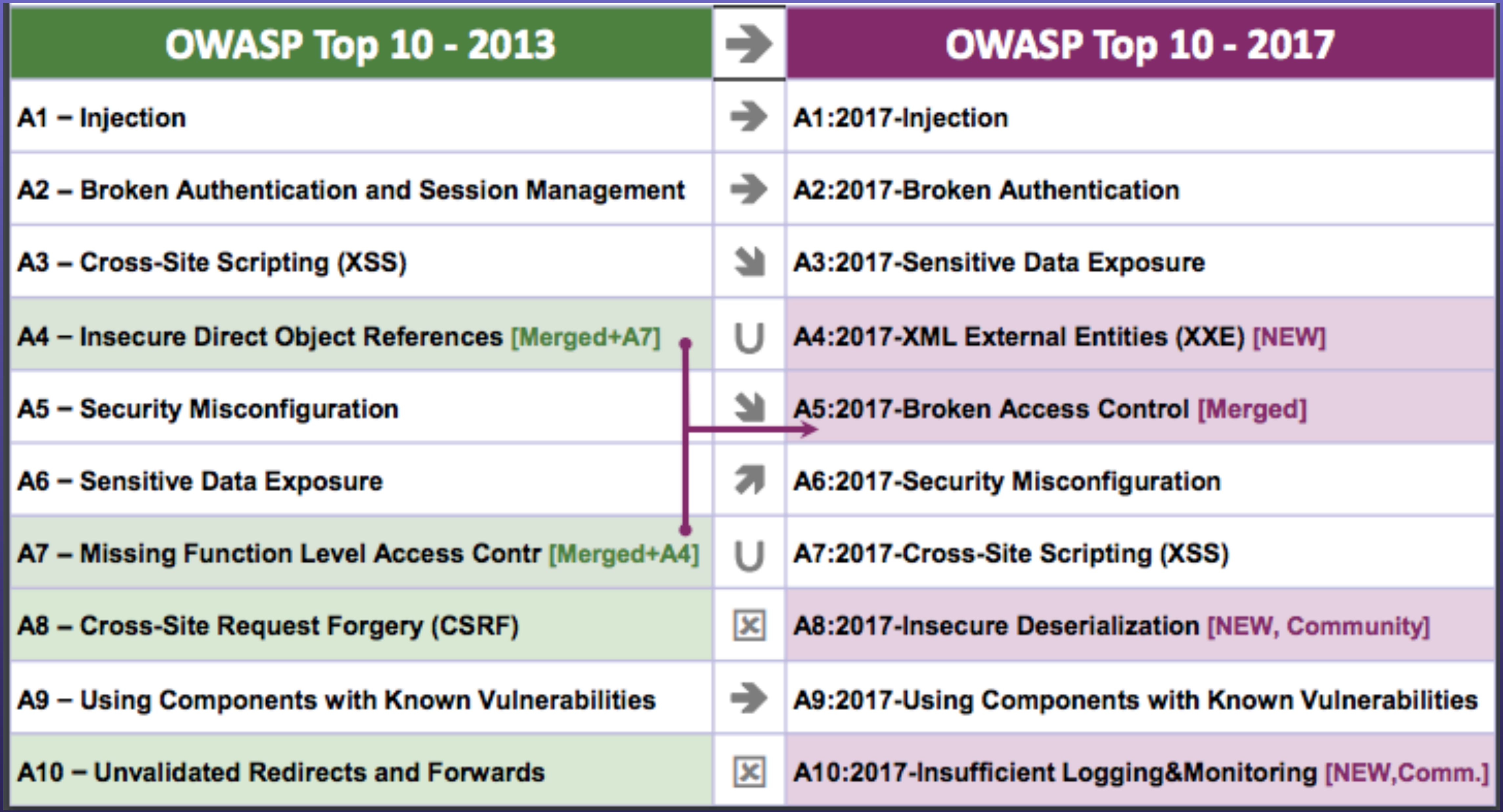
SQL injection

Håvard Raddum

Simula UiB

Last updated on 21.04.2021

Motivation



OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Outline

- Short introduction to databases and SQL
- SQL injection attacks
- Real-world cases of SQL injection
- How to protect against SQL injection

Databases

- A (relational) database contains one or more tables
- Each table contains a number of rows (records) and a number of columns
- Each column has a name indicating what information is stored in that column
- The records contain the actual data in the database

Database Example

Table with name «Profiles»

Person number	Name	Address	Card number	Account number	Email	Mobile number
01126927569	Helge Hentetakst	Karamellstien 5 8514 Narvik	4625 9051 8723 7465	0562 14 34806	hhtaxt@outlook.com	45092371
11118346601	Solveig Sundet	Bilringen 18 0304 Oslo	4569 1209 3048 5570	1384 35 90447	ssundet@gmail.com	90094456
23057339082	Marie Munte	Taugaten 3D 5005 Bergen	4569 0432 9447 2211	9713 52 21739	marie.munte@vps.no	41734890

What is SQL?

- Structured Query Language (SQL) is a standard computer language for accessing and manipulating databases
- SQL works with database programs like MySQL, MS Access, MS SQL Server, etc.
- There are several different versions of SQL, but all must support the same major keywords in a similar manner

Some major keywords

- SELECT - returns some records from a table
- UPDATE - changes existing data in a table
- INSERT INTO - adds new data to table
- DELETE - deletes some records in a table
- DROP - deletes a table, or a whole database
- WHERE - conditional statement; indicates which records to act upon

More syntax…

- AND - combines conditions, all must be satisfied
- OR - combines conditions, at least one must be satisfied
- * - wildcard, used in many queries
- LIKE - partial matching of string
- ‘ ‘ - strings must be written in single quotes

SQL queries

- Specific actions on a table are specified by building a command using the allowed key words
- May add, delete, change or read specific data by building a syntactically correct sentence using the key words
- Sentences are called SQL queries

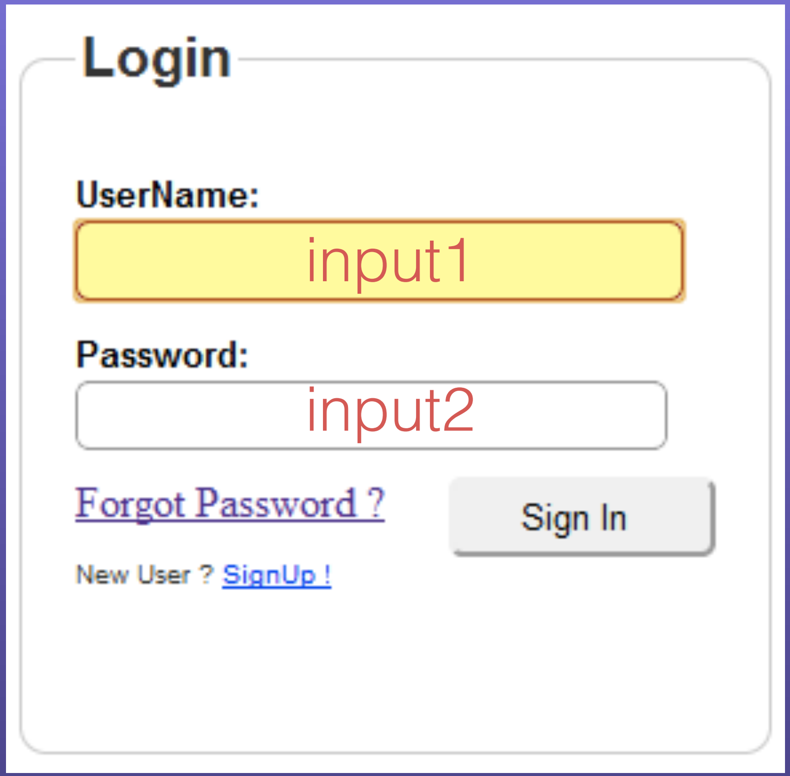
SQL query examples

- SELECT * FROM Profiles WHERE card_number=‘4625 9051 8723 7465’;
- UPDATE BilRegister SET rabatt_prosent=100 WHERE bilnummer LIKE ‘EL%’ OR bilnummer LIKE ’EK%’ OR bilnummer LIKE ‘EV%’;
- DROP TABLE Customers;

SQL on the internet

- Many web server applications use databases
- SQL queries can be specified on the web server, where inputs from users are inserted into the query before it is executed
- Users are allowed to act on a database, but should only be able to do the specific actions programmed on the server

Example: login



Login

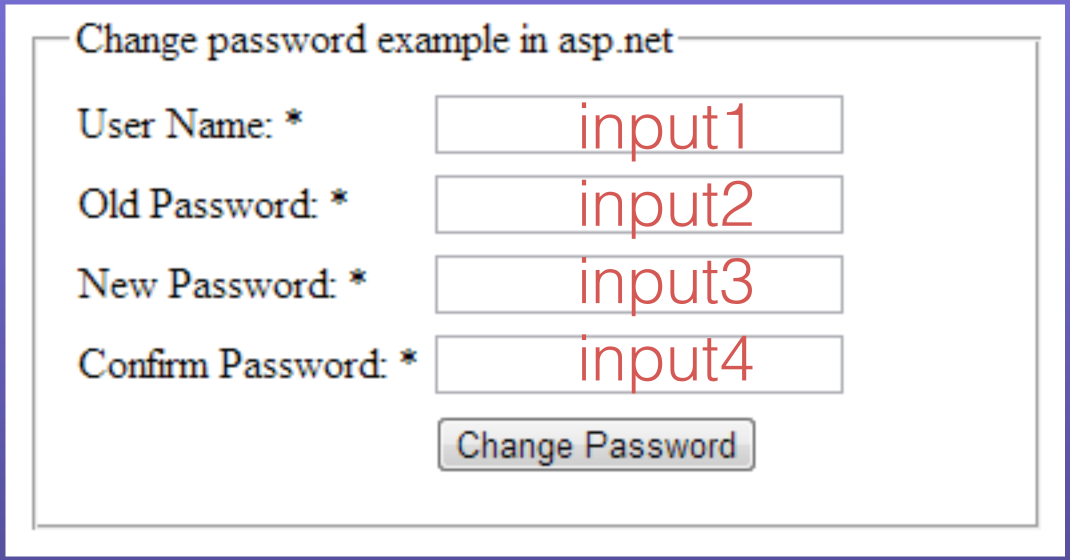
UserName:
input1

Password:
input2

- User enters user name and password
- When clicking the button, the following SQL query might be sent from web server to database

SELECT * FROM users WHERE UserName=‘input1’ AND Password=‘input2’;

Example: password change



Change password example in asp.net

User Name: *	input1
Old Password: *	input2
New Password: *	input3
Confirm Password: *	input4

- A user fills in the four fields
- When the user clicks the button, the following SQL statement might be executed:

UPDATE profiles SET password=‘input3’ WHERE user_name=‘input1’ AND password=‘input2’ AND ‘input3’=‘input4’;

Is it safe?

- Apparently, users can only specify values to be searched for or inserted in some table
- Only predefined SQL statements can be executed
- Proper checks are hard coded into the SQL queries, i.e. making sure the user has entered the correct password etc.
- Appears to be safe to let users initiate predefined actions on the database

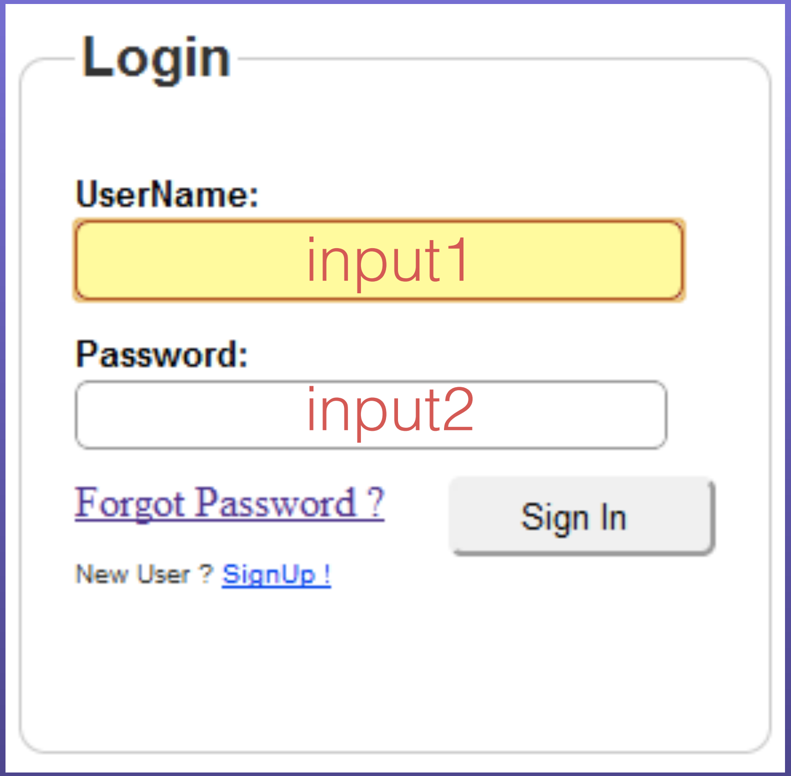
Not safe!

- Users may enter *anything* they want in the fields
- If the application does not check what some user has typed into the fields, the web server is vulnerable to SQL injection
- SQL injection attacks can do a lot of harm!

Learning the SQL query

- The attacker can enter something containing a single ‘-sign, creating an invalid SQL statement
- If the web server has not been programmed to give generic error messages for invalid SQL requests, the error message from the database will be displayed to the attacker
- The error message from the database includes the whole SQL query, with names of tables and columns

login example



If attacker enters
input1=vic'
input2=
The generated SQL query will be

SELECT * FROM users WHERE
UserName='vic' ' AND Password='';

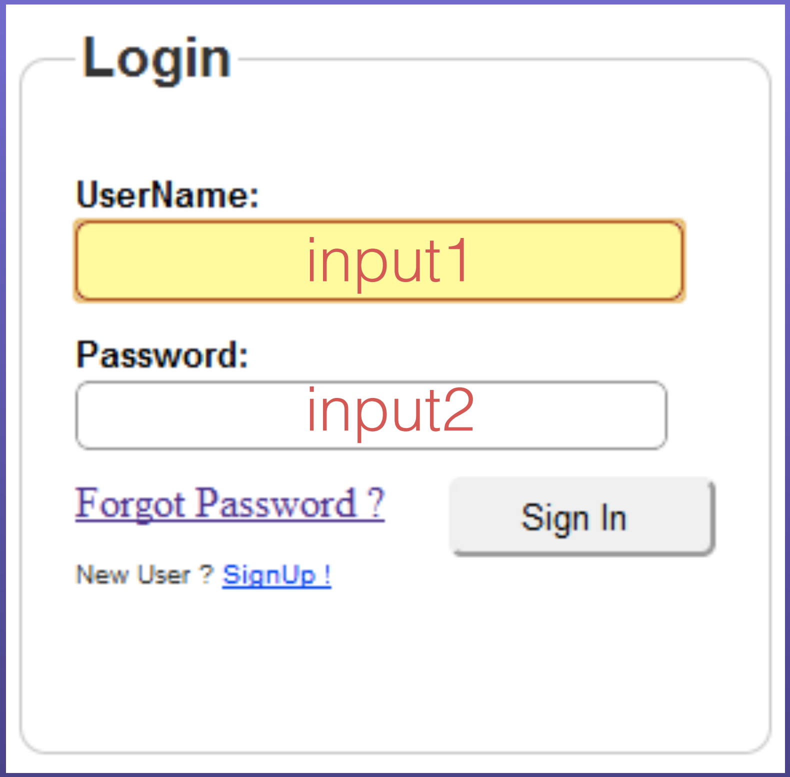
Two quotes

This may result in the following error message displayed on the attackers screen

Syntax error in SQL statement: SELECT * FROM users WHERE UserName='vic' ' AND Password='';

The attacker learns the structure of the SQL query, the name of the table and the names of the columns involved!

SQL injection 1


Login

UserName:

input1

Password:

input2

Forgot Password ?

Sign In

New User ? SignUp !

SELECT * FROM users WHERE UserName='input1' AND Password='input2';

The attacker can enter some (partial) SQL code into the fields

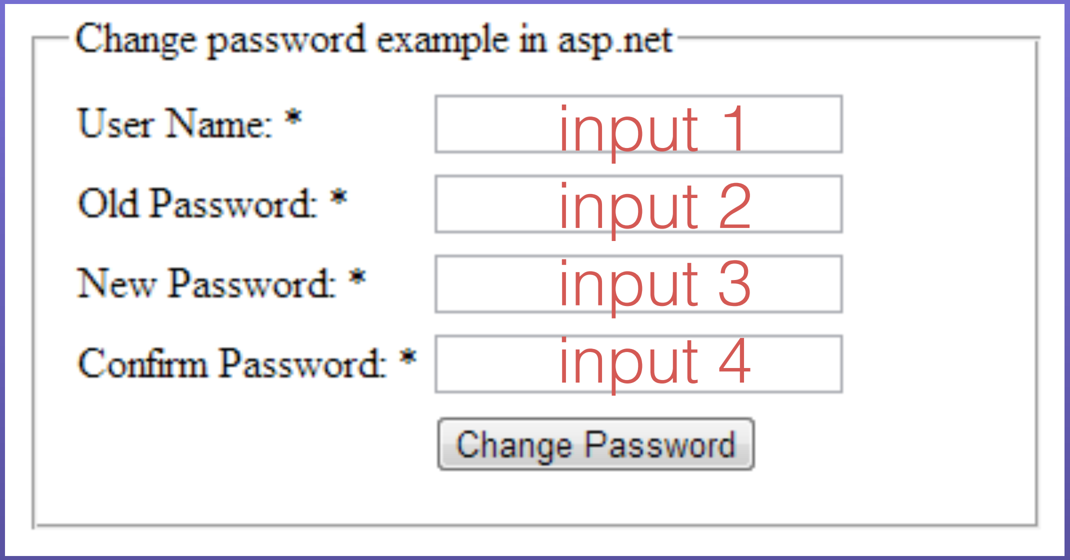
input1 = victim'; - -
input2 =

two dashes indicate comment in SQL

SELECT * FROM users WHERE UserName='victim'; - -' AND Password='';

Attacker will be logged in as the user «victim»!
(password check is commented out and not done)

SQL injection 2



Change password example in asp.net

User Name: *	input 1
Old Password: *	input 2
New Password: *	input 3
Confirm Password: *	input 4

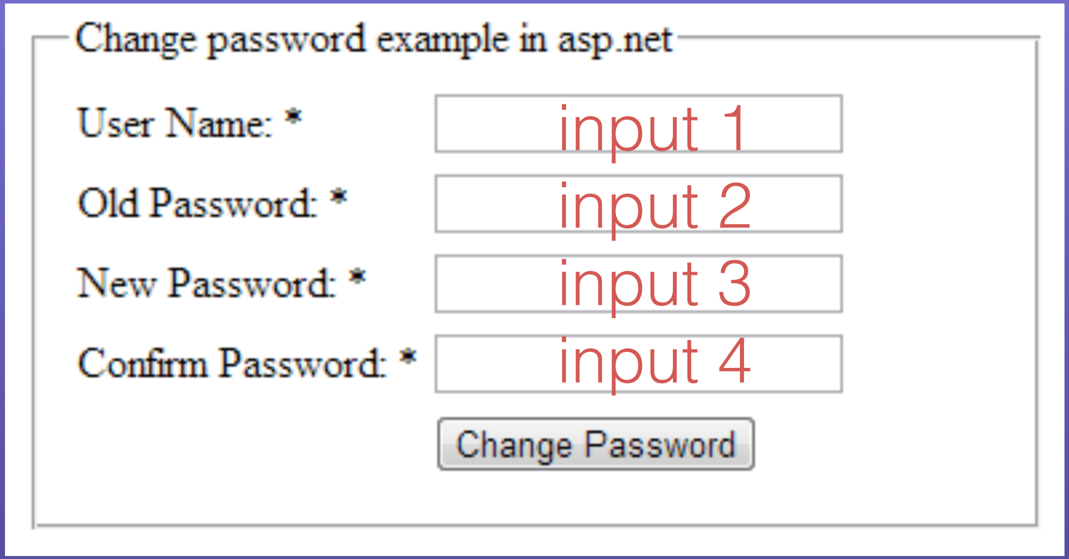
UPDATE profiles SET password='input3'
WHERE user_name='input1' AND
password='input2' AND 'input3'='input4';

Attacker may choose:
input1=victim'; - -
input2=
input3=abc123
input4=

UPDATE profiles SET password='abc123'
WHERE user_name='victim'; - -' AND
password='' AND 'abc123'='';

Attacker has successfully changed «victim»'s password!

SQL injection 3



Change password example in asp.net

User Name: *	input 1
Old Password: *	input 2
New Password: *	input 3
Confirm Password: *	input 4

UPDATE profiles SET password='input3'
WHERE user_name='input1' AND
password='input2' AND 'input3'='input4';

Attacker may choose:
input1=victim'; DROP TABLE profiles; - -
input2=
input3=abc123
input4=

UPDATE profiles SET password='abc123'
WHERE user_name='victim'; DROP TABLE profiles; - -' AND
password='' AND 'abc123'='';

The attacker has deleted the table with all user data!

SQL injection in the URL

- Some web sites pass parameters from URLs directly into SQL queries
- If GET method is used, parameters are visible and may be manipulated in URL, but not if POST method is used
- SQL code in the URL may do just as much harm as SQL code in form fields

SQL in URL

A web shop may allow the URL
http://www.myshop.com/showProduct.asp?id=123
to execute the SQL query
SELECT * FROM Products WHERE productID=123;

Entering the URL
http://www.myshop.som/showProduct.asp?id=123; DROP TABLE Products
will generate SELECT * FROM Products WHERE productID=123; DROP TABLE Products;

Valid syntax, will delete the table with all products

Blind SQL injection

- If the web server is configured to give only generic error messages on invalid SQL queries, the attacker does not learn the structure of the hard coded SQL string

- An attacker can still do SQL injection, by sending SQL code to the web server and look for differences in responses

- This is known as *blind SQL injection*

Blind SQL injection example

Online magazine http://www.mymag.com

Attacker opens an article, and notes the URL:
http://www.mymag.com/showArticle.php?id=2379

Attacker changes URL: …/showArticle?id=2379 AND 1=2

and also checks

…/showArticle?id=2379 AND 1=1

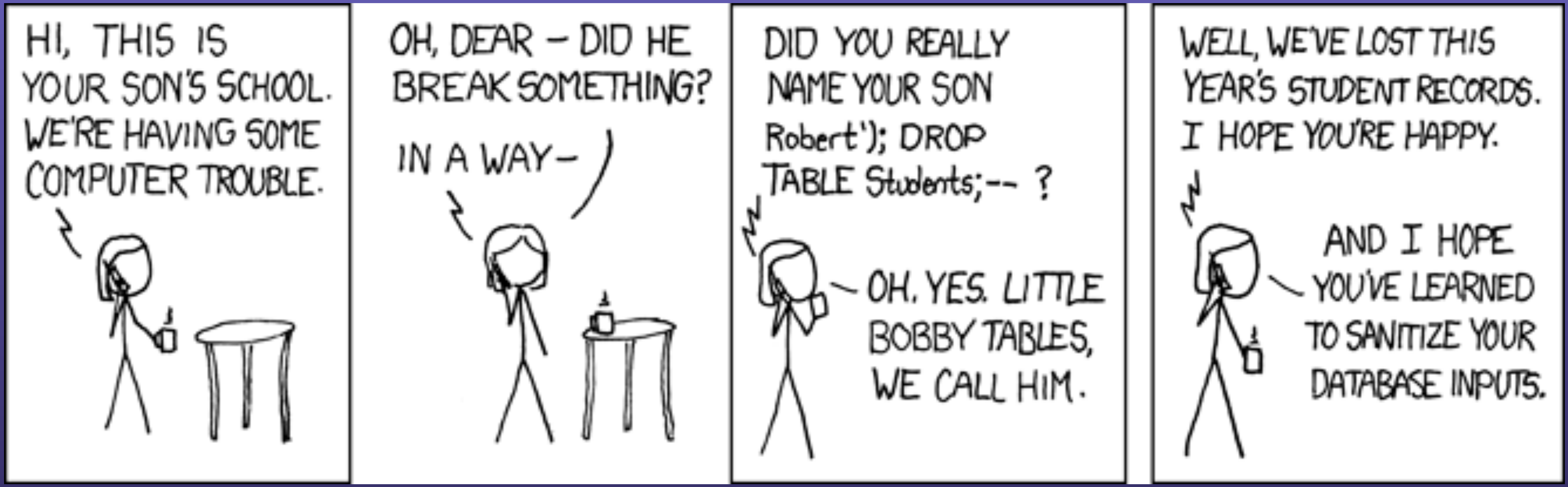
If the responses from the web server are different, the attacker knows the server reacts to the injected SQL code

Blind SQL injection

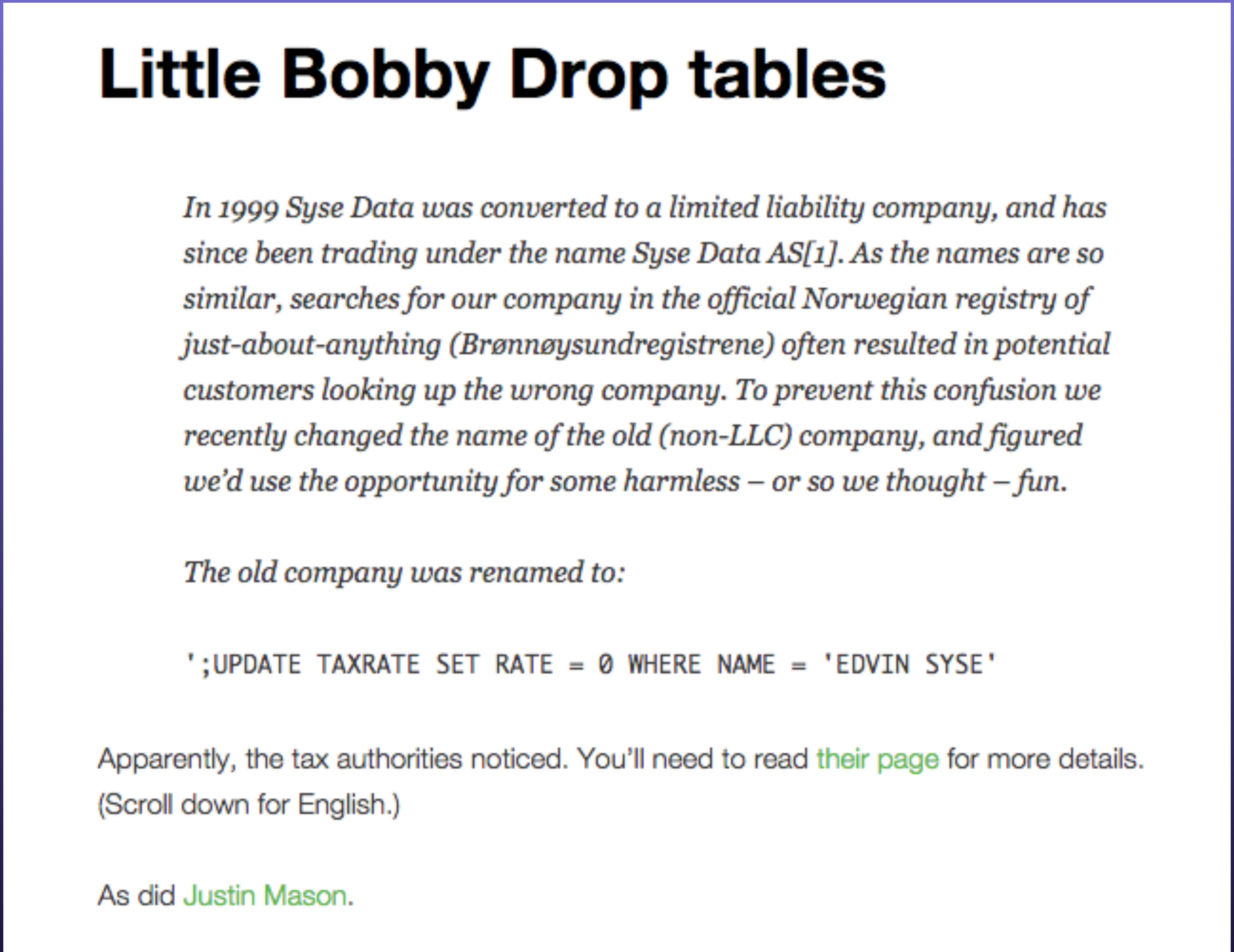
- Attacker may test true or false statements and learn about the database
- …id=2379 AND substring(@@version, 1, 1)=5 tells if the MySQL database version is 5 (or not)
- …id=2379 AND EXISTS(SELECT * FROM …) tells if the query in the brackets actually returned something

Potential harm

- An attacker can perform a large number of harmful and unauthorized actions on a web site that is vulnerable to SQL injection
- An attacker can
  - read private data
  - modify data
  - delete data
  - add data to the database


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Real story (2009)



Little Bobby Drop tables

> *In 1999 Syse Data was converted to a limited liability company, and has since been trading under the name Syse Data AS[1]. As the names are so similar, searches for our company in the official Norwegian registry of just-about-anything (Brønnøysundregistrene) often resulted in potential customers looking up the wrong company. To prevent this confusion we recently changed the name of the old (non-LLC) company, and figured we'd use the opportunity for some harmless – or so we thought – fun.*
>
> *The old company was renamed to:*

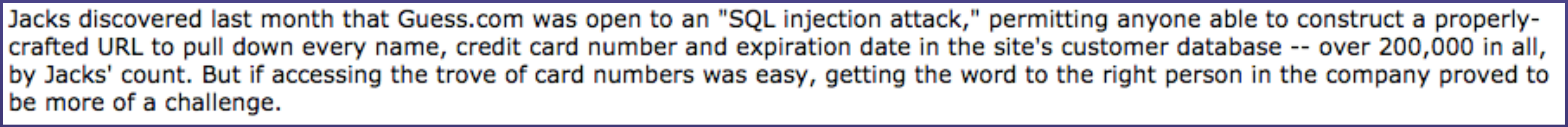
```
';UPDATE TAXRATE SET RATE = 0 WHERE NAME = 'EDVIN SYSE'
```

Apparently, the tax authorities noticed. You'll need to read their page for more details. (Scroll down for English.)

As did Justin Mason.

SQL injection in the wild (2002)

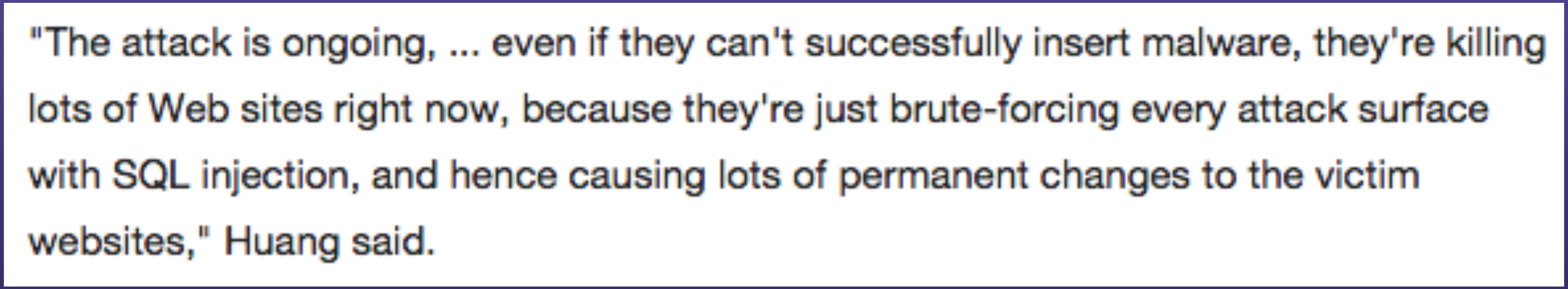
- guess.com, a fashion retailer, had a database containing 200.000 credit card numbers of their customers exposed due to SQL injection


Jacks discovered last month that Guess.com was open to an "SQL injection attack," permitting anyone able to construct a properly-crafted URL to pull down every name, credit card number and expiration date in the site's customer database -- over 200,000 in all, by Jacks' count. But if accessing the trove of card numbers was easy, getting the word to the right person in the company proved to be more of a challenge.

http://www.securityfocus.com/news/346

SQL injection in the wild (2008)

- Chinese hackers destroyed many web sites using automated scripts searching for servers vulnerable to SQL injection

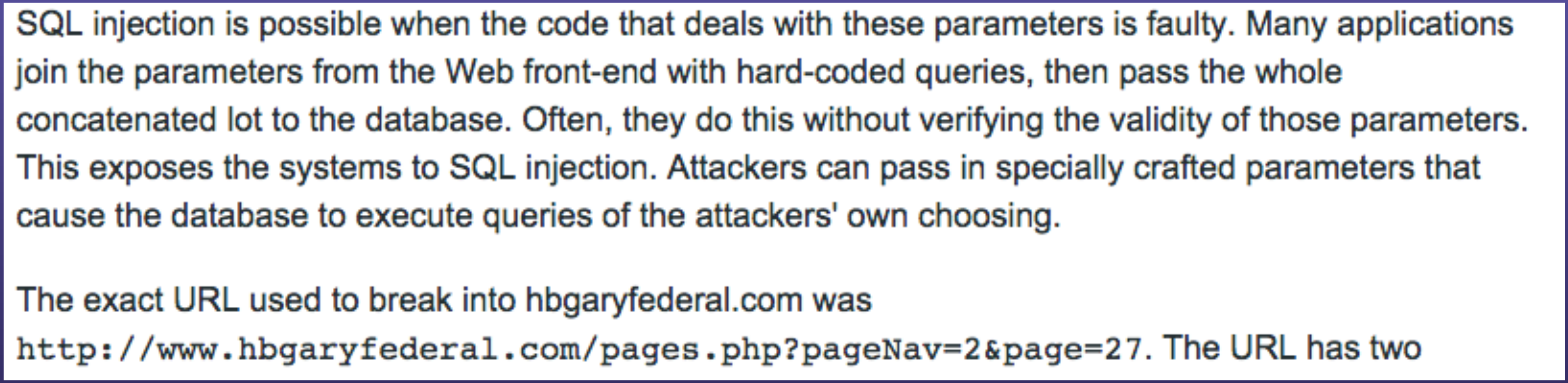


"The attack is ongoing, ... even if they can't successfully insert malware, they're killing lots of Web sites right now, because they're just brute-forcing every attack surface with SQL injection, and hence causing lots of permanent changes to the victim websites," Huang said.

http://www.pcworld.com/article/146048/article.html

SQL injection in the wild (2011)

- HBGary, a company selling security solutions to the US government, was attacked by SQL injection



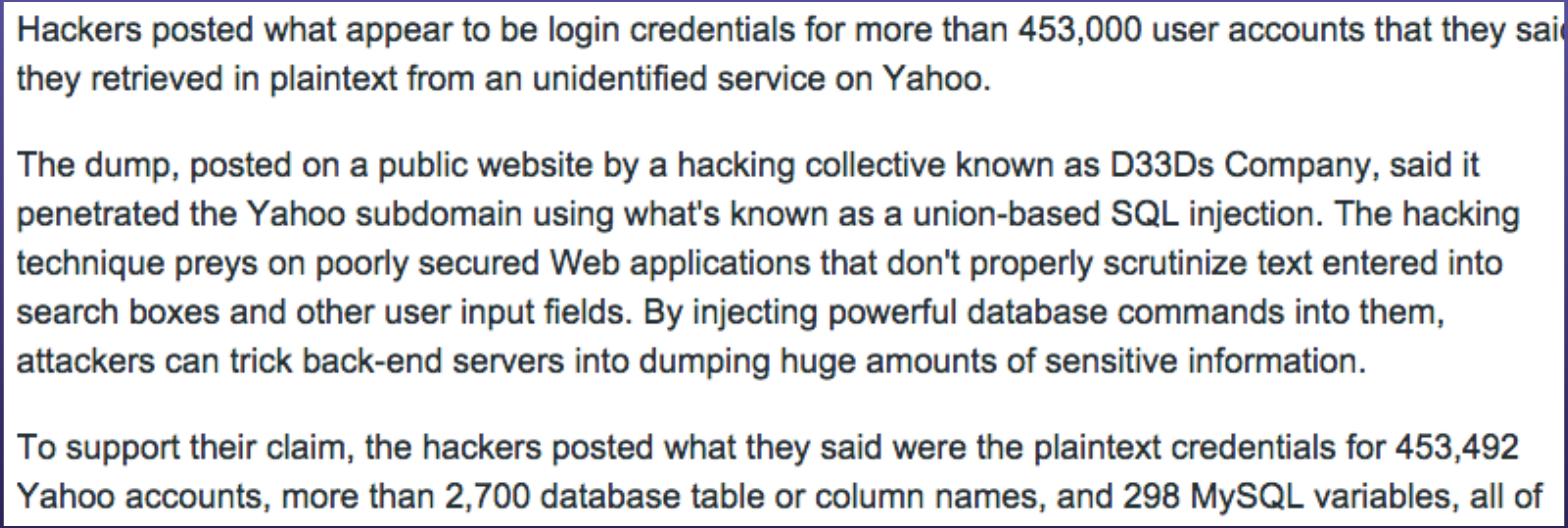
SQL injection is possible when the code that deals with these parameters is faulty. Many applications join the parameters from the Web front-end with hard-coded queries, then pass the whole concatenated lot to the database. Often, they do this without verifying the validity of those parameters. This exposes the systems to SQL injection. Attackers can pass in specially crafted parameters that cause the database to execute queries of the attackers' own choosing.

The exact URL used to break into hbgaryfederal.com was `http://www.hbgaryfederal.com/pages.php?pageNav=2&page=27`. The URL has two

http://arstechnica.com/tech-policy/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack/

SQL injection in the wild (2012)

- The database of a Yahoo! service, containing over 450.000 user accounts was leaked using SQL injection


Hackers posted what appear to be login credentials for more than 453,000 user accounts that they sai
they retrieved in plaintext from an unidentified service on Yahoo.

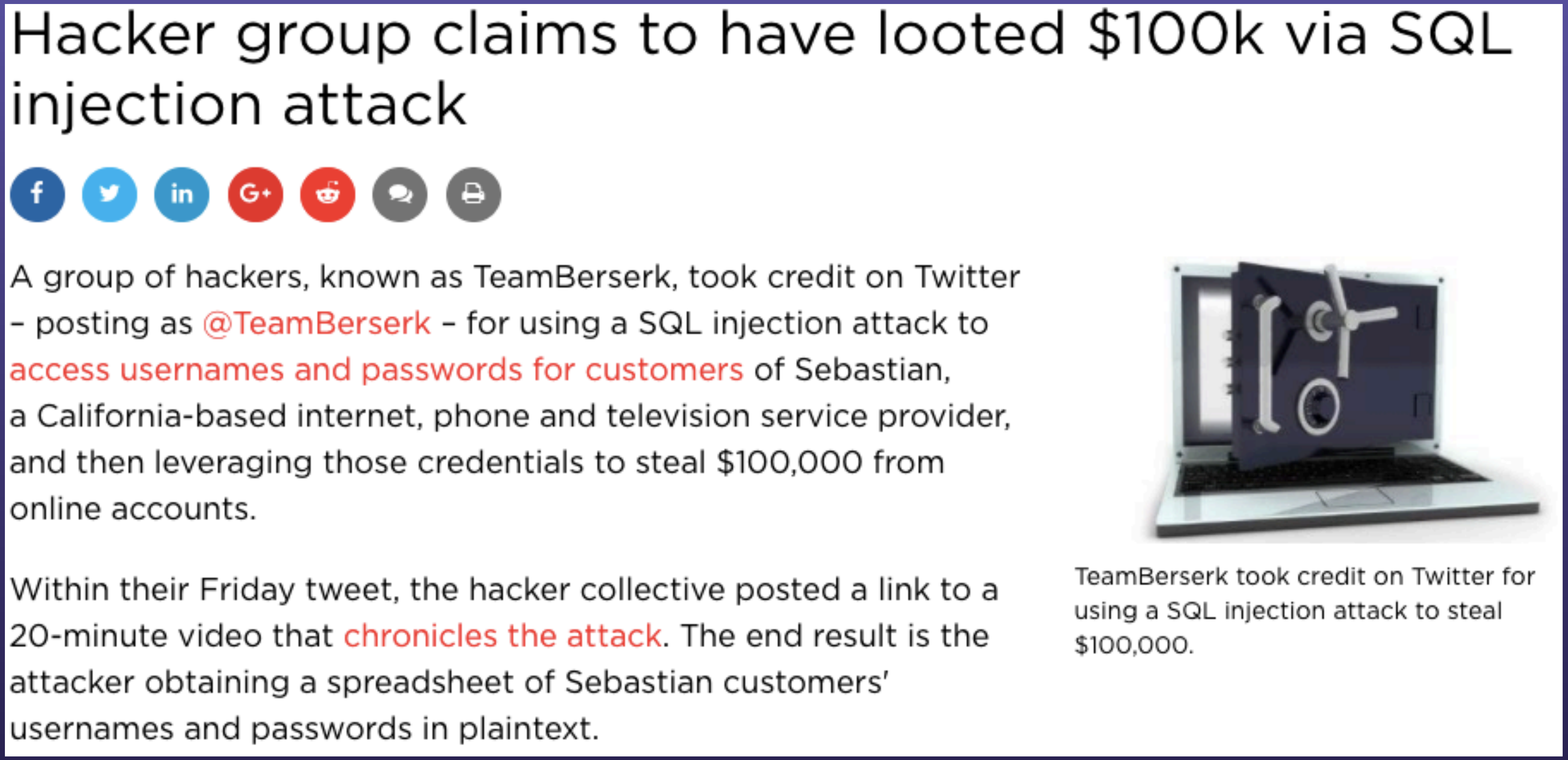
The dump, posted on a public website by a hacking collective known as D33Ds Company, said it penetrated the Yahoo subdomain using what's known as a union-based SQL injection. The hacking technique preys on poorly secured Web applications that don't properly scrutinize text entered into search boxes and other user input fields. By injecting powerful database commands into them, attackers can trick back-end servers into dumping huge amounts of sensitive information.

To support their claim, the hackers posted what they said were the plaintext credentials for 453,492 Yahoo accounts, more than 2,700 database table or column names, and 298 MySQL variables, all of

http://arstechnica.com/security/2012/07/yahoo-service-hacked/

SQL injection in the wild (2013)

- SQL injection used to retrieve user names and passwords
- Several users had same passwords on their Paypal accounts



Hacker group claims to have looted $100k via SQL injection attack

A group of hackers, known as TeamBerserk, took credit on Twitter – posting as @TeamBerserk – for using a SQL injection attack to access usernames and passwords for customers of Sebastian, a California-based internet, phone and television service provider, and then leveraging those credentials to steal $100,000 from online accounts.

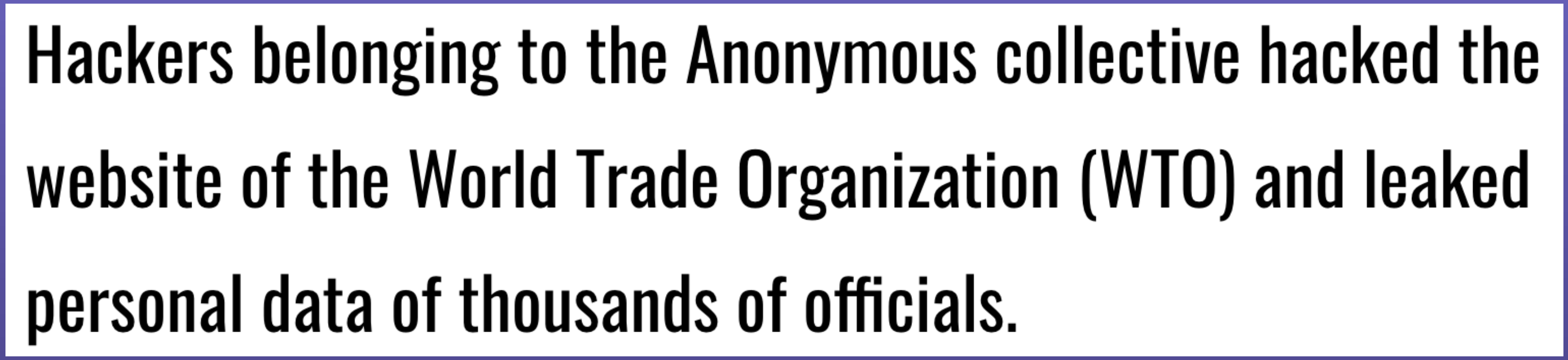
Within their Friday tweet, the hacker collective posted a link to a 20-minute video that chronicles the attack. The end result is the attacker obtaining a spreadsheet of Sebastian customers' usernames and passwords in plaintext.

TeamBerserk took credit on Twitter for using a SQL injection attack to steal $100,000.

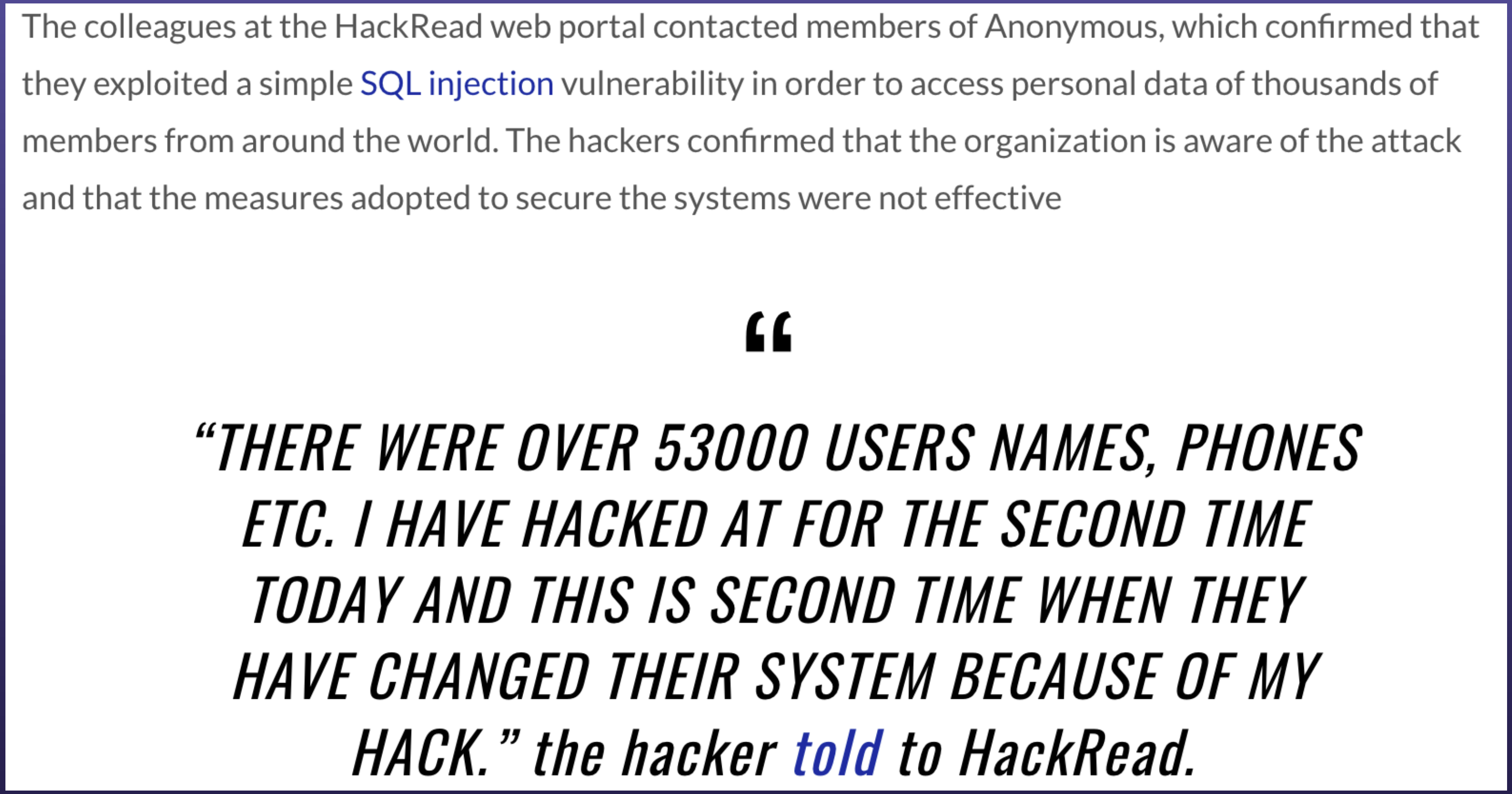
www.scmagazine.com/hacker-group-claims-to-have-looted-100k-via-sql-injection-attack/article/542609/

SQL injection in the wild (2015)

- Personal details of staff at the WTO were leaked due to an SQL injection attack



Hackers belonging to the Anonymous collective hacked the website of the World Trade Organization (WTO) and leaked personal data of thousands of officials.



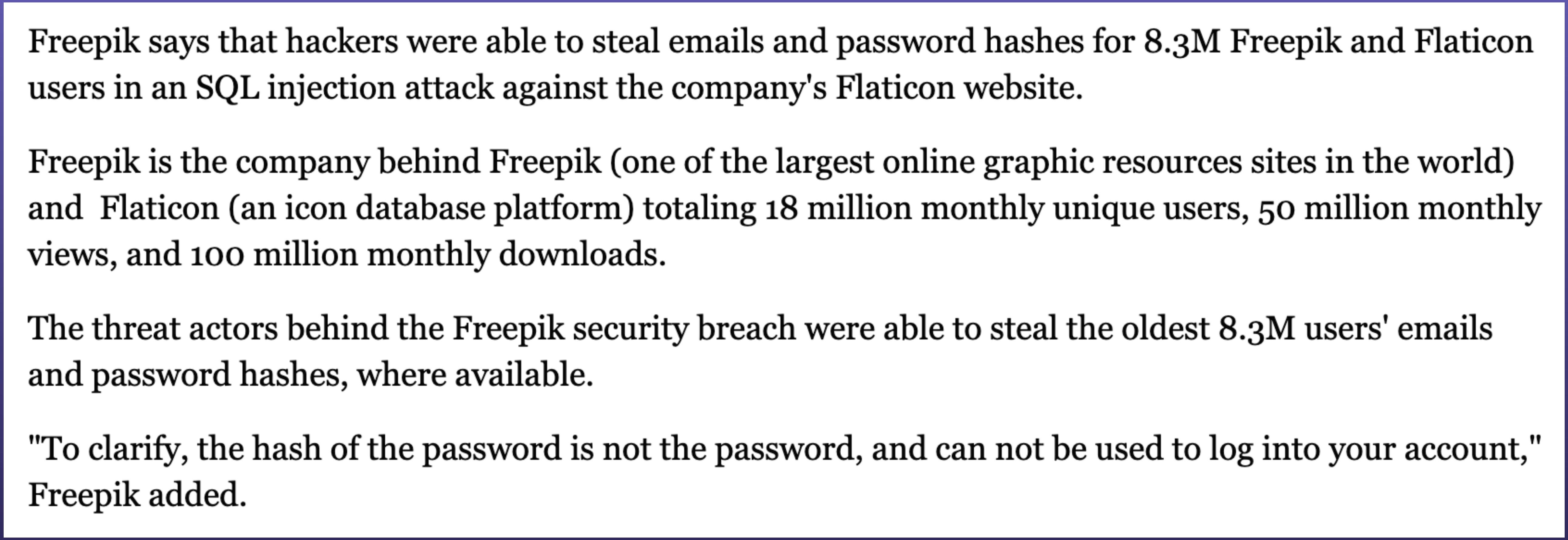
The colleagues at the HackRead web portal contacted members of Anonymous, which confirmed that they exploited a simple SQL injection vulnerability in order to access personal data of thousands of members from around the world. The hackers confirmed that the organization is aware of the attack and that the measures adopted to secure the systems were not effective

❝

"THERE WERE OVER 53000 USERS NAMES, PHONES ETC. I HAVE HACKED AT FOR THE SECOND TIME TODAY AND THIS IS SECOND TIME WHEN THEY HAVE CHANGED THEIR SYSTEM BECAUSE OF MY HACK." the hacker told to HackRead.

http://securityaffairs.co/wordpress/36528/hacking/anonymous-breached-wto-db.html

SQL injection in the wild (2020)



Freepik says that hackers were able to steal emails and password hashes for 8.3M Freepik and Flaticon users in an SQL injection attack against the company's Flaticon website.

Freepik is the company behind Freepik (one of the largest online graphic resources sites in the world) and Flaticon (an icon database platform) totaling 18 million monthly unique users, 50 million monthly views, and 100 million monthly downloads.

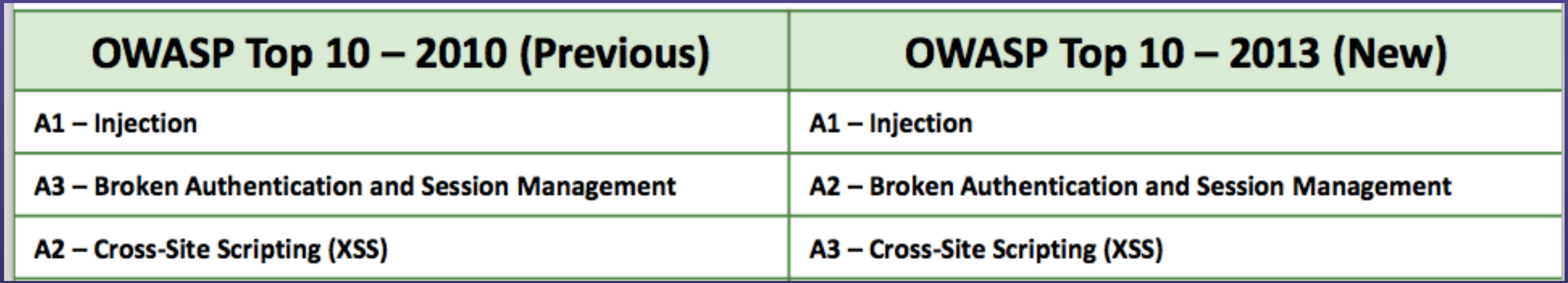
The threat actors behind the Freepik security breach were able to steal the oldest 8.3M users' emails and password hashes, where available.

"To clarify, the hash of the password is not the password, and can not be used to log into your account," Freepik added.

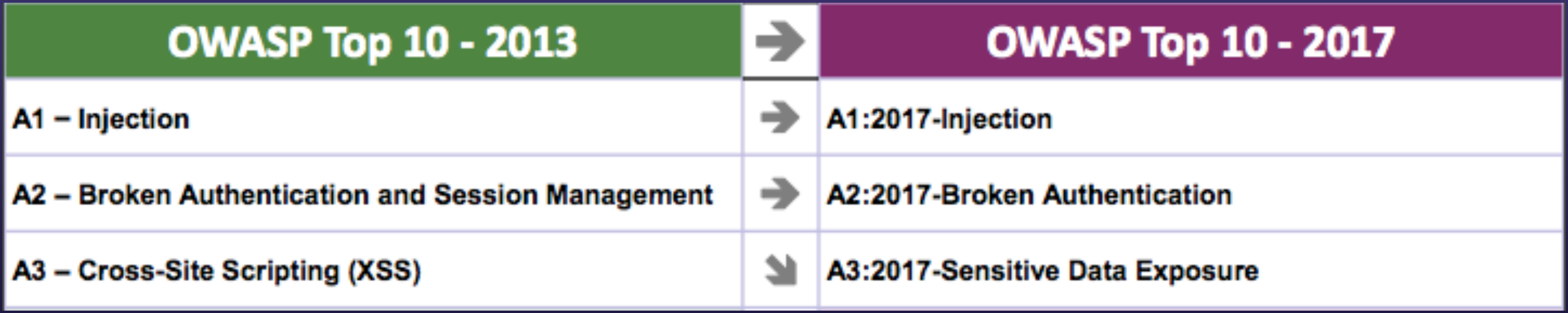
SQL injection still major weakness

- The dangers of SQL injection has been well known for more than 20 years

- SQL injection still #1 on the OWASP top ten list of web security risks



OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)



OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure

Defense

- Defending against SQL injection attacks is not hard, and can be done in several ways:
    - Input validation
    - Prepared statement
    - Restrict database permissions
    - HEX conversion

Input validation

- Check data coming from a web user before inserting it into an SQL query
- May create an application on the web server that filters all input from users
- Any data supplied by the user must be accepted by the filter before being inserted into the SQL string

Input validation

- Expecting a number?
    - make sure input really is a number, otherwise reject
- Expecting an alphanumerical string?
    - Make sure there are only letters and numbers in input string and no special characters
- Never allow the ‘-sign or other special characters in the input!

Problems with input validation

- If input field is a password, special characters should be allowed
- Hard to verify 100% that your filter will catch *all* potentially harmful inputs

Prepared statements

- Most databases supports compiling or parsing SQL queries with unknown parameters
- A parsed query is stored in the database for later execution
- A web server needs only pass some actual values of the parameters to execute a query, and not submit a whole query to be interpreted and executed

Prepared statements

- A prepared statement is not vulnerable to SQL injection

- If SQL code is supplied as a parameter, it will not be interpreted and executed

SELECT * FROM users WHERE UserName=‘victim’; - -’ AND Password=‘’;

The statement has already been interpreted earlier.
The database engine will look for records where the value in the UserName column is victim’; - - and the password column is empty (and will probably not find anything)

Restrict permissions

- Most database systems support restrictions on queries some server can make
- A web server may be denied the permission to delete tables using the DROP command
- A web server may be given *only* the permission to execute prepared statements
- Permission restriction gives good defence in depth, protecting against future (or unknown) security holes on the web server

HEX conversion

- Convert any user input into its hexadecimal representation at the web server
- Hexadecimal conversion supported by almost any web or software technology
- The hard-coded SQL statement must include UNHEX() to be executed correctly

```
SELECT * FROM users WHERE UserName=UNHEX('str2hex(input1)') AND
Password=UNHEX('str2hex(input2)');
```

HEX conversion

- At the time the SQL statement is parsed and interpreted, it may look like

```
SELECT * FROM users WHERE UserName=UNHEX(‘12A5B7640E’) AND
Password=UNHEX(‘20FC5689A1’);
```

- Only numbers 0-9 and letters ABCDEF can occur in user input at the time SQL statement is interpreted

HEX conversion

- Protection is similar to prepared statement
- SQL code in user input is not interpreted by database

If UNHEX(‘12A5B7640E’) = victim’; - - then database will search for the string victim’; - - in the column with user names (and probably not find any matching records)

Summary

- SQL injection is well-known, but still one of the most dangerous attacks on the internet
- Very large amounts of sensitive information may leak, or a whole database be deleted, in one successful attack
- Protecting against SQL injection is not hard, but awareness of the vulnerability is needed