Lecture Network Security
Chapter 3 – Attack Vector SQL Injection

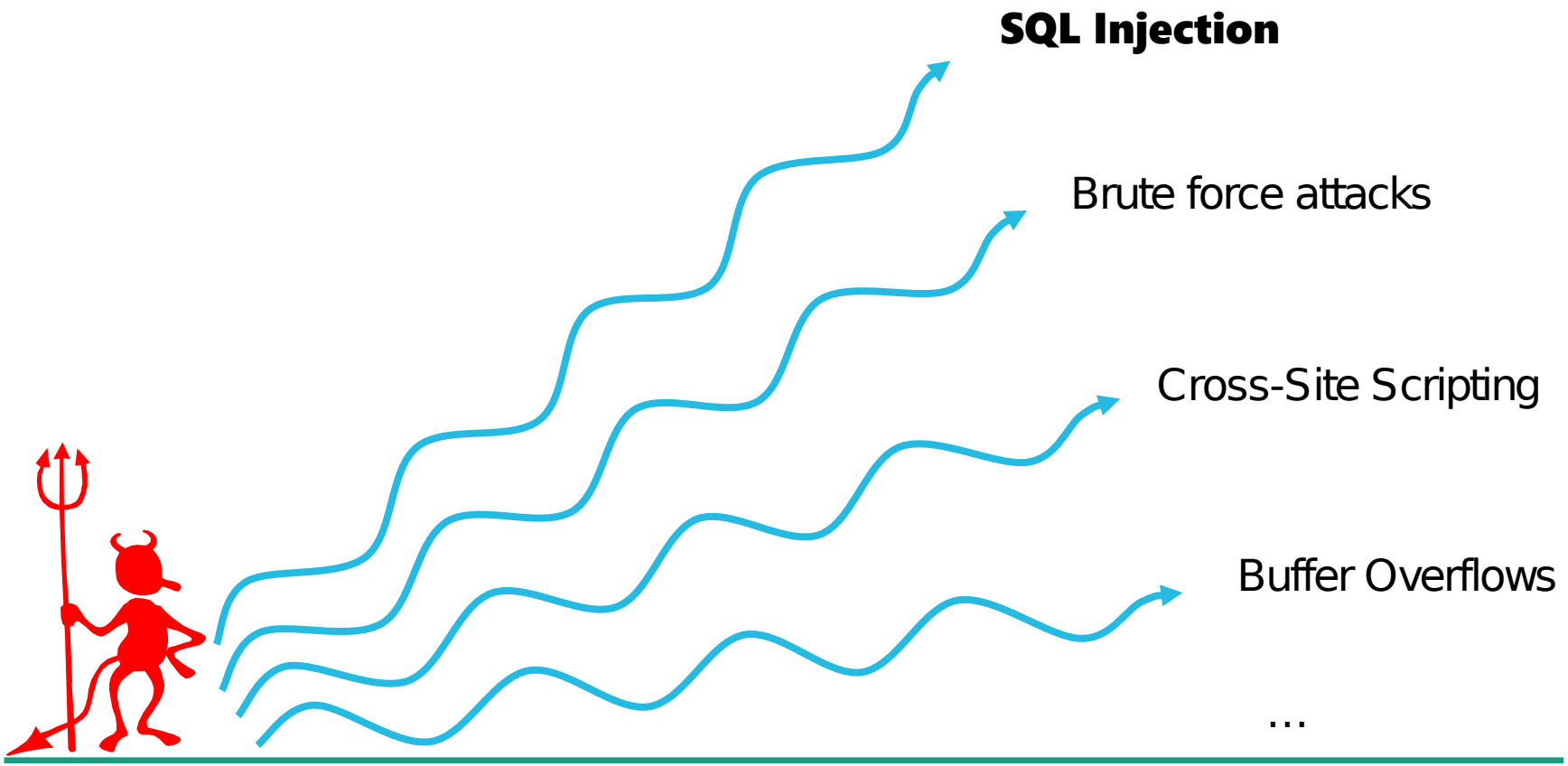
University of Bonn, Institute of Computer Science IV, Summer 2016

universitätbonn

Fraunhofer FKIE

Attack Vectors

There are several categories of methods to attack computer systems.
They are called **Attack Vector** or **Injection Vector**.



universitätbonn

Fraunhofer FKIE

SQL – Structured Query Language

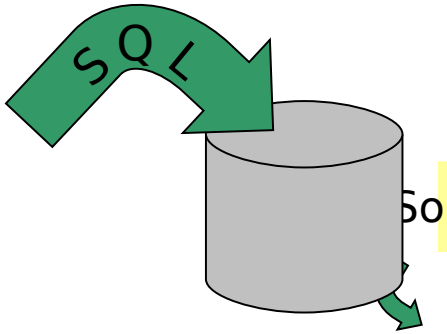
Designed in 1974 by Donald D. Chamberlin and Raymond F. Boyce (IBM)

Several newer releases exist

Standard language for communication with relational database management systems (RDBMS)

Used to fetch/add/delete/modify *data* in tables (and sometimes the database structure)

Database **query** = sending an SQL string to the database



Some queries return result data sets

Applications access RDBMS using SQL and API calls

SQL Basics – SELECT Statement

SELECT – Statement: Select a specific set of data

Simplified Syntax:
SELECT Columns FROM Table WHERE options

Query: String

```
SELECT Id, Username FROM SystemUsers WHERE Valid='Yes'
```

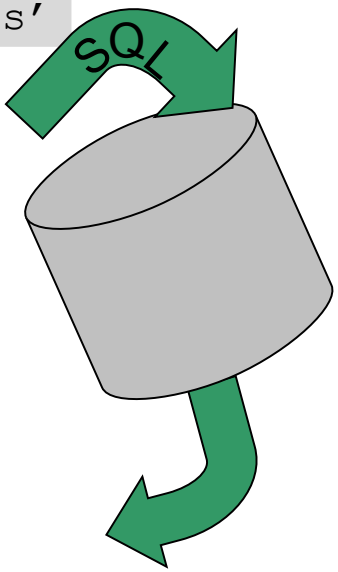


Table: SystemUsers

Id	Username	Password	Valid
1	Jens	Chief	Yes
4711	Wolfgang	MrMagic	No
666	Elmar	Evil	Yes
007	Felix	Nuts	No

Result data set

Id	Username
1	Jens
666	Elmar

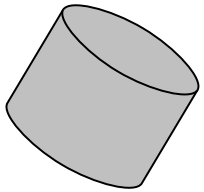
SQL Basics – Insert Statement

Insert – Statement: Insert new data set into table

Simplified Syntax:
INSERT INTO Table (Columns) VALUES Values

Table: SystemUsers – before query

Id	Username	Password	Valid
1	Jens	Chief	Yes

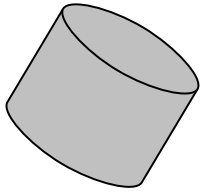


Query: String

```
INSERT INTO SystemUsers (Id, Username, Password, Valid)
VALUES (4711, 'Wolfgang', 'MrMagic', 'Yes')
```

Table: SystemUsers – after query

Id	Username	Password	Valid
1	Jens	Chief	Yes
4711	Wolfgang	MrMagic	No

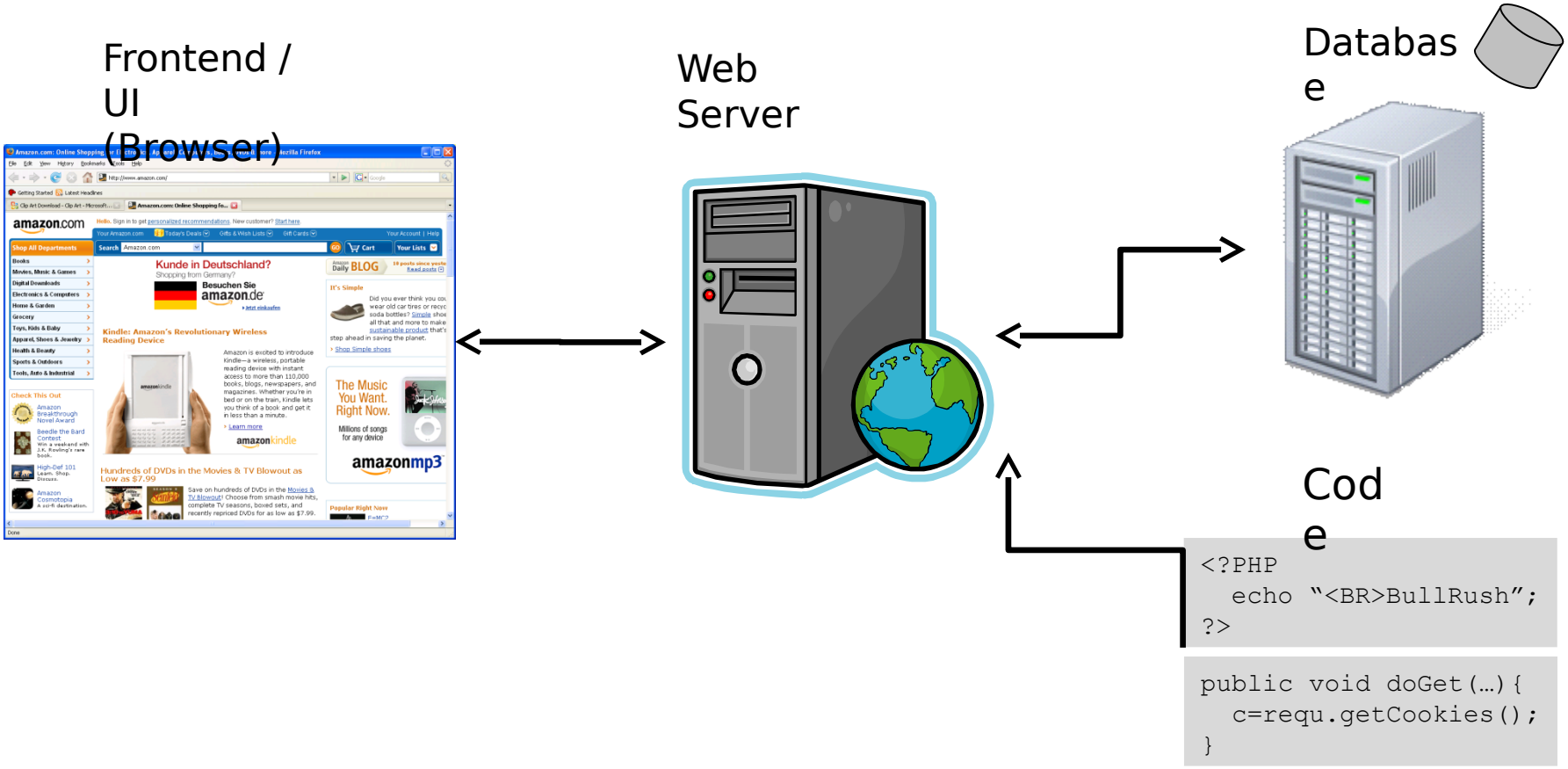


universitätbonn

Fraunhofer FKIE

Typical Application Architecture

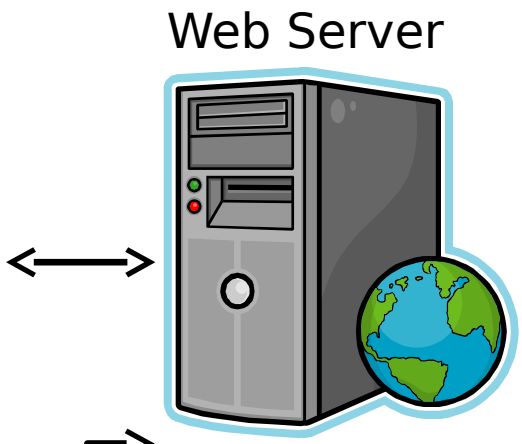
Most (web-)applications have the following architecture…



universitätbonn

Fraunhofer FKIE

Web-page login




```
user_var = $email
pass_var = $password

sql = "SELECT Balance FROM Users
       WHERE User='%s' AND Password='%s'", user_var, pass_var

results = database.query(sql)

if count(results) > 0 then
  //user logged in
  print "Your account balance is: %s", results.Balance
else
  print "Access denied"
```

Web-page login (2)

Sign In

What is your e-mail address?

My e-mail address is leder@cs...

Do you have an Amazon.com password?

No, I am a new customer.

Yes, I have a password: Nuts

Sign in using our secure server

Forgot your password? Click here

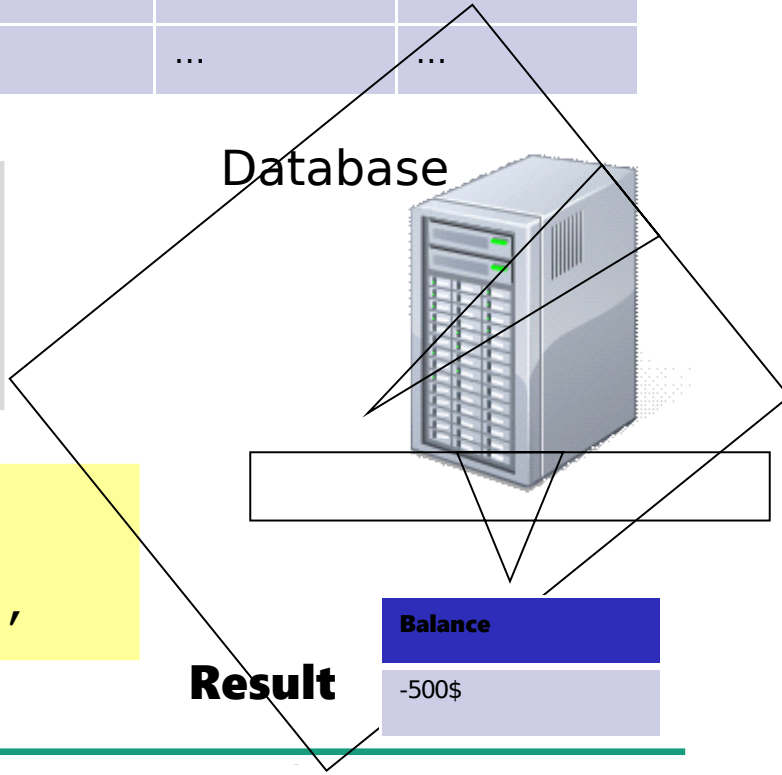
Has your e-mail address changed since your last order?

Table: Users

User	Password	Balance
moll@cs...	MrMagic	1234$
leder@cs...	Nuts	-500$
...	...	...

```
sql = "SELECT Balance FROM Users
       WHERE User='%s' AND Password='%s'",
       user_var, pass_var

results = database.query(sql)
```



Query sent to database:

```
SELECT Balance FROM Users WHERE
User='leder@cs…' AND Password='Nuts'
```

Result

Balance
-500$

universitätbonn

Fraunhofer FKIE

SQL Injection Attack on the Login Process

Query sent to database:

```
SELECT Balance FROM Users WHERE User='leder@cs…' AND
Password='xxx' OR User='moll@cs…'
```

Table: Users

User	Password	Balance
moll@cs...	MrMagic	1234$
leder@cs...	Nuts	-500$
…	…	…

Result

Balance
1234$

universitätbonn

Fraunhofer FKIE

Countermeasures

Properly check every input into your application !!!

Never trust any **external** data

(user input, URL-parameters, files, cookies, configuration, …)

Escape critical characters OR use provided escaping function

Note: The characters that require escaping differ from database to database

Example: mysql_real_escape_string( )

Network intrusion detection systems can detect suspicious patterns in network traffic

Some databases have detection modules for typical SQL injection patterns

SQL Injecting escaped Strings

Escaped: ‘ to \’

```
SELECT … WHERE … AND Password='xxx\' OR User=\'moll@cs…'
```

…can be tricked by password “xxx\’ OR 1=1 /*comment:” …

Query sent to database:

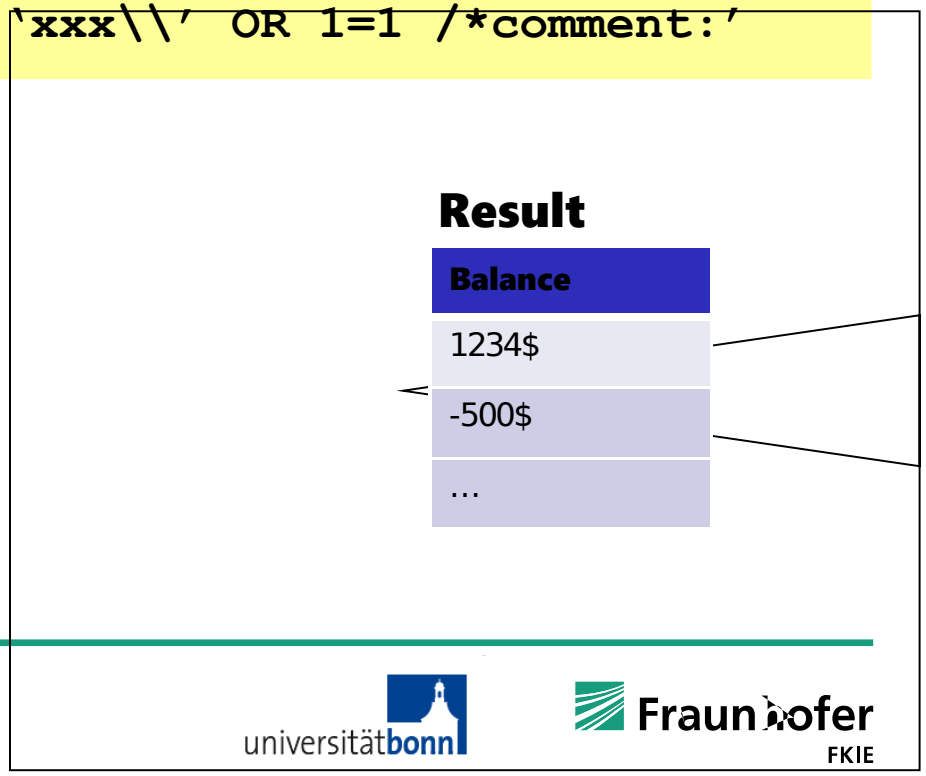
```
SELECT … WHERE … AND Password='xxx\\' OR 1=1 /*comment:'
```

Table: Users

User	Password	Balance
moll@cs…	MrMagic	1234$
leder@cs…	Nuts	-500$
…	…	…

Result

Balance
1234$
-500$
…



universitätbonn

Fraunhofer FKIE

SQL Injecting escaped Strings (2)

Even worse: Password "xxx\'; INSERT INTO Users (User, Password, Balance) VALUES("bl@bla...", "hijacked, 10.000) /*" …

Query sent to database:

```
SELECT … WHERE … AND Password='xxx\\';
INSERT INTO Users (User, Password, Balance)
VALUES( "bl@bla...", "hijacked, 10000) /*'
```

Two queries are sparated by "; "

Table: Users

User	Password	Balance
moll@cs...	MrMagic	1234$
leder@cs..	Nuts	-500$
…	…	…

Table: Users

User	Password	Balance
moll@cs...	MrMagic	1234$
leder@cs..	Nuts	-500$
bl@bla...	Hijacked	10000$
…	…	…

Think, think, think…

Properly check every input into your application !!!

Think about all possible escaping problems and escape other critical characters as well:

\ to \\

\n to \\n

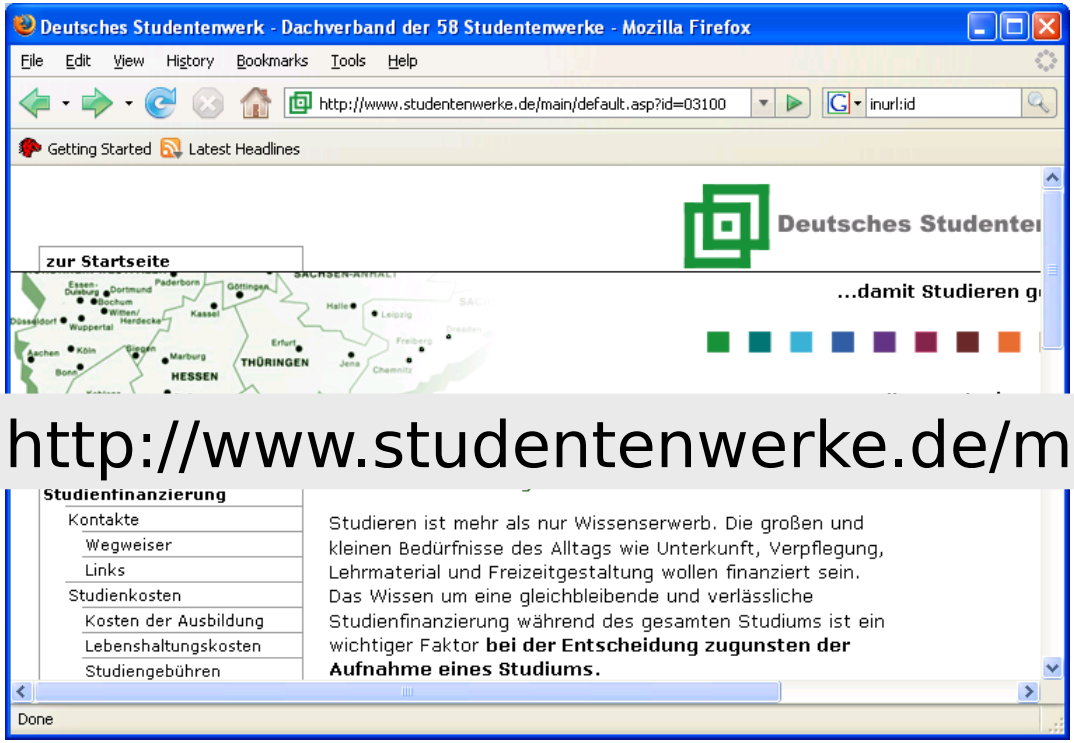
\r to \\r

```
SELECT … WHERE … AND Password='xxx\\\' OR 1=1'
```

One single string

and **use available escaping functions**

Pitfall: Untyped languages



http://www.studentenwerke.de/main/default.asp?**id=03100**

Often string parameters are well escaped

But developers forget that untyped variables may contain strings instead of numbers

```
Query = "SELECT … WHERE … AND id=", $id
```

Pitfall Untyped languages (2)

http://www…./default.asp?**id=03100 OR 1=1 OR user='root'**

Query sent to database:

```
SELECT … WHERE … AND id=03100 OR 1=1 OR user='root'
```

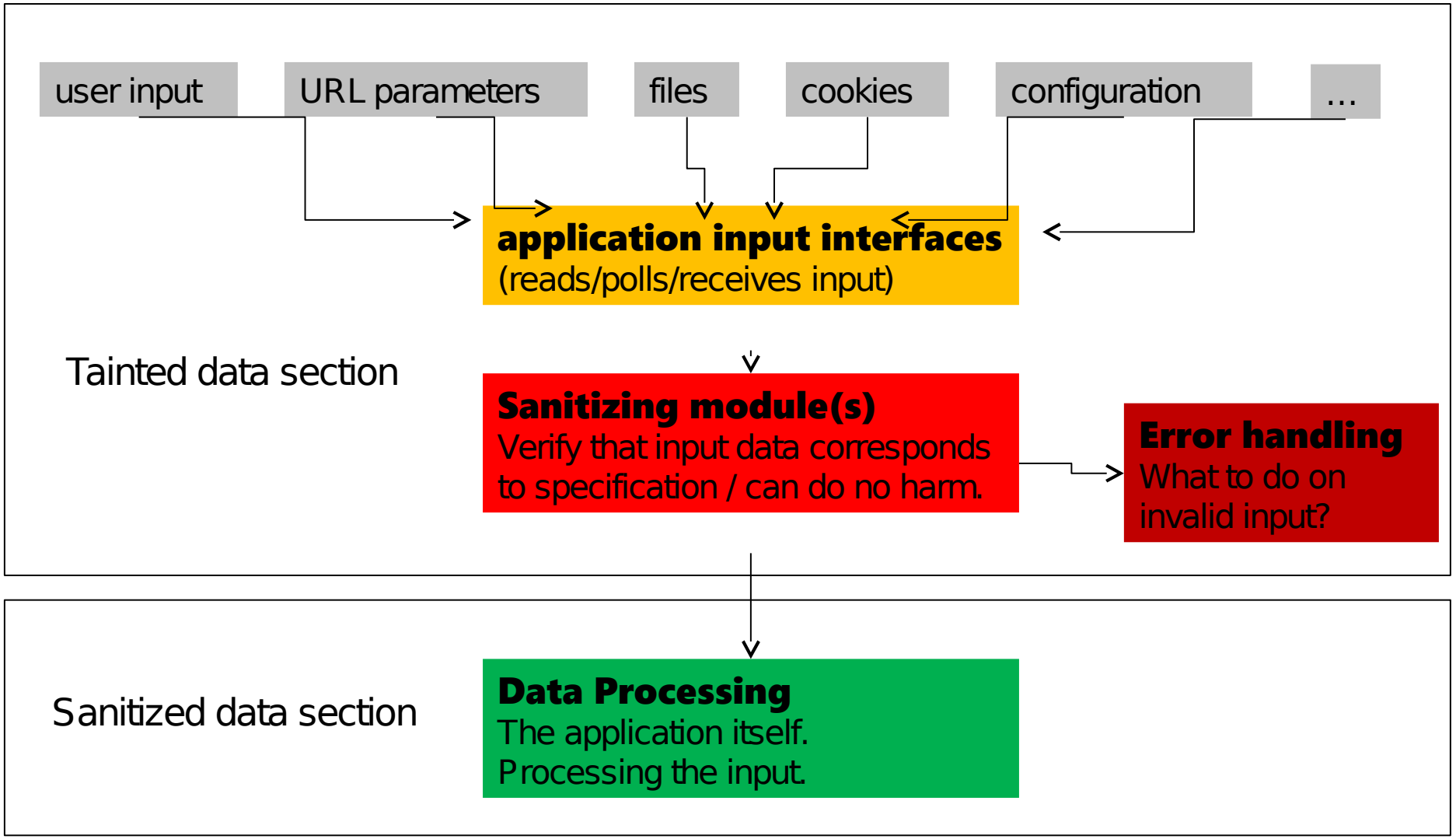
Properly check every input into your application !!!
(user input, URL-parameters, files, cookies, configuration, registry, …)

```
If isNumeric($id) then
  Query = "SELECT … WHERE … AND id=", $id
Else
  …
```

Conclusions:
Securing Applications against Attack Vectors

- Many flaws can be avoided at design and development time
- Never trust any external data
  (not even other applications - they may be modified)
- Properly check every input into your application
- Design a proper architecture to not lose overview and important details

Secure Software Architectures



Countermeasures

Never

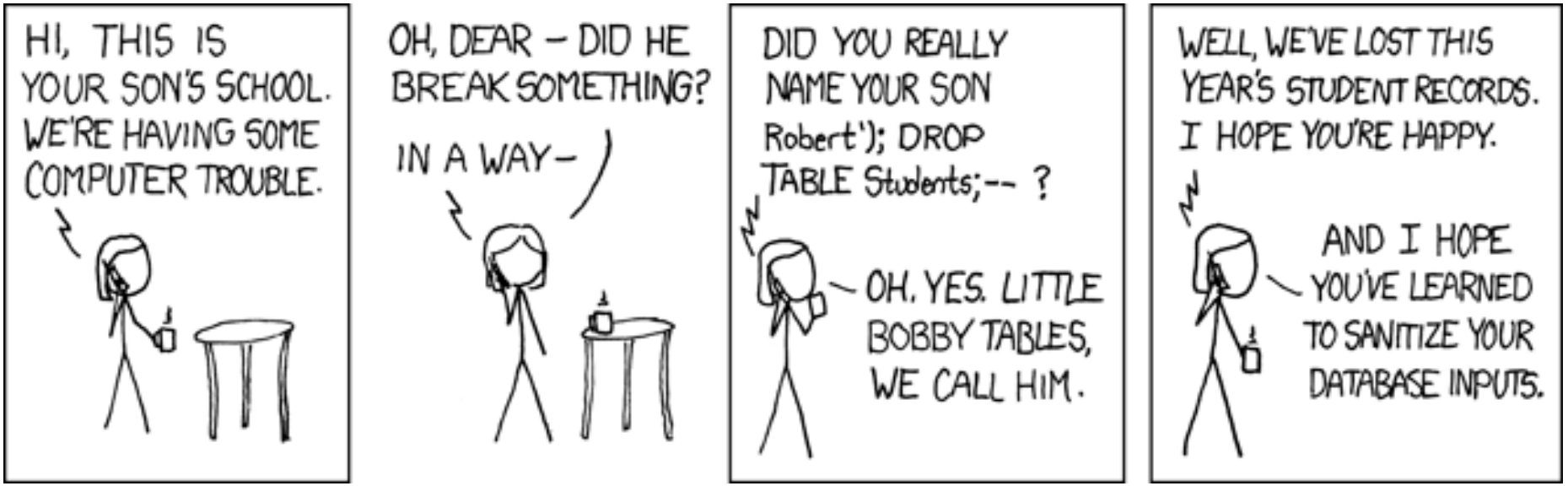
trust any

external data!

Check every input! (user input, URLs, parameters, files, cookies, configuration, …)

Use *Escape Sequences* – (but do it right!), look for suspicious patterns,…

Summary: xkcd.org



http://xkcd.org/327/