injection attacks

Daniel Jackson

For now, all rights reserved. Daniel Jackson, 2013.
what is injection?
a JavaScript injection

lethal combination
› strings everywhere
› eval command

...<FORM NAME="Calc">
  <INPUT TYPE="text" NAME="Input" Size="16">
  <INPUT TYPE="button" NAME="one" VALUE="  1  "
    OnClick="Calc.Input.value += '1'">
  <INPUT TYPE="button" NAME="three" VALUE="3"
    OnClick="Calc.Input.value += '3'">
  ...
  <INPUT TYPE="button" NAME="plus" VALUE="+
    OnClick="Calc.Input.value += '+'">
  <INPUT TYPE="button" NAME="DoIt" VALUE="=
    OnClick="Calc.Input.value = eval(Calc.Input.value)">
</FORM>

A Javascript/HTML calculator

what is injection?

interpreters
› eg, eval (JavaScript), execute (SQL)
› turn data into code
› very useful, very dangerous

JavaScript injection
› in itself, no big deal (unless JS runs server side)
› but can lead to XSS and CSRF
SQL injection
a SQL injection attack

show items ordered

enter year

query = "SELECT date, item FROM orders WHERE user=" 
+ session[‘user_id’] 
+ " AND year=" + request.form[‘year’] 
execute(query)
an injection attack

suppose user makes a modified HTTP request
› https://www.store.com/orders?year=0%20OR%201%3D1

`SELECT date, item FROM orders
WHERE user=126 AND year=0 OR 1=1`

effect
› sets year variable to 0 OR 1=1
› shows all orders in the database
worse

user generates this query:

```
SELECT date, item FROM orders
WHERE user=126 AND year=0
UNION
SELECT cardholder, number, exp_date FROM creditcards
```

reveals credit card database!
even worse

user generates this query:

```
SELECT date, item FROM orders 
WHERE user=126 AND year=0 
; DROP TABLE creditcards
```
and even worse...

user generates this query

```
SELECT date, item FROM orders
WHERE user=126 AND year=0
; INSERT INTO admin VALUES ('hacker', ...)
```

user takes over machine!
revenge on traffic cameras?

Bobby Tables

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.

from http://xkcd.com/327/
We have an employee whose last name is Null. He kills our employee lookup app when his last name is used as the search term (which happens to be quite often now). The error received (thanks Fiddler!) is

```xml
<soapenv:Fault>
  <faultcode>soapenv:Server.userException</faultcode>
  <faultstring>coldfusion.xml.rpc.CFCInvocationException: [coldfusion.runtime.Mis
```

Cute, huh?

The parameter's type is string. Using WSDL (SOAP). Flex 3.5 Actionscript 3 ColdFusion 8

Note that the error DOES NOT occur when calling the webservice as an object from a coldfusion page.
shell injection
secure voting site?

Hacker infiltration ends D.C. online voting trial
Last week, the D.C. Board of Elections and Ethics opened a new Internet-based voting system for a weeklong test period, inviting computer experts from all corners to prod its vulnerabilities in the spirit of ‘give it your best shot.’ Well, the hackers gave it their best shot -- and midday Friday, the trial period was suspended, with the board citing ‘usability issues brought to our attention.’

Here's one of those issues: After casting a vote, according to test observers, the Web site played *The Victors* -- the University of Michigan fight song.

Washington Post, Oct 4, 2010
uploading completed PDF ballot
shell injection vulnerability

uploaded ballot saved like this:

```bash
run ('gpg', '−−trust−model always −o "#{File.expand_path(dst.path)}" −e −r "#{@recipient}" "#{File.expand_path(src.path)}"')
```

so attacker uploaded file with name

```bash
> myfile.$(command)
```

see Wolchok et al. Attacking the Washington, D.C. Internet Voting System


Unix command substitution: execute command and replace expr by result
even got control of camera!

see Wolchok et al. Attacking the Washington, D.C. Internet Voting System
preventing injection attacks

best strategy
 › never call an interpreter!

if you must make commands on the fly
 › build them with expression objects, not strings

for database injections
 › use an ORM: no SQL queries
 › use parameterized queries

bad:
Client.where("city = #{params[:city]}")

better:
Client.where("city = ?", params[:city])