Security in Hop Web Applications

T. Rezk

INDES team

INRIA Sophia Antipolis –Méditerranée

INRIA-MSR Joint Lab

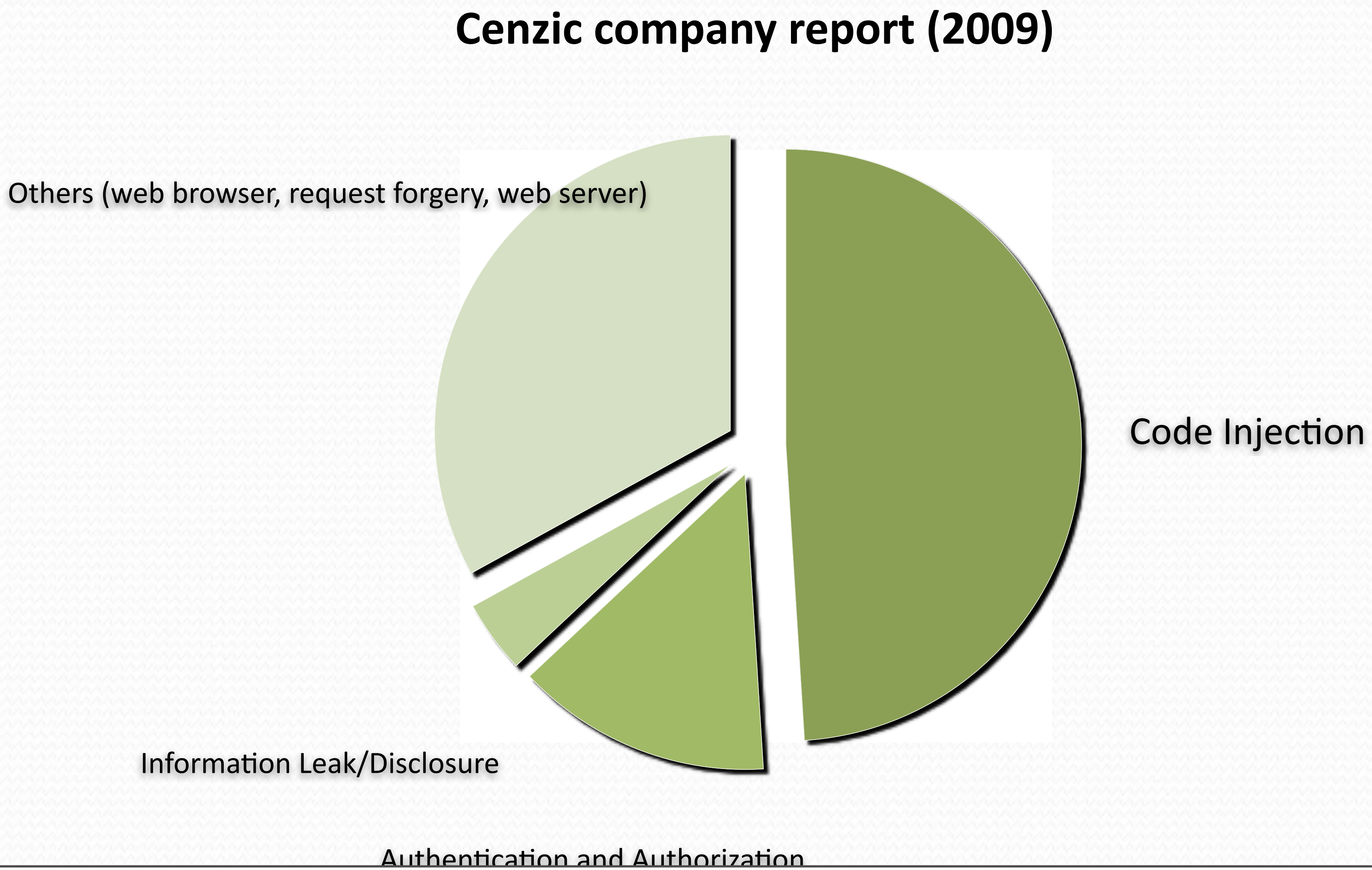
JOINT WORK (in progress):

G.Boudol, Z.Luo, M.Serrano

29/01/2010

PARSEC meeting

Reports on web vulnerabilities



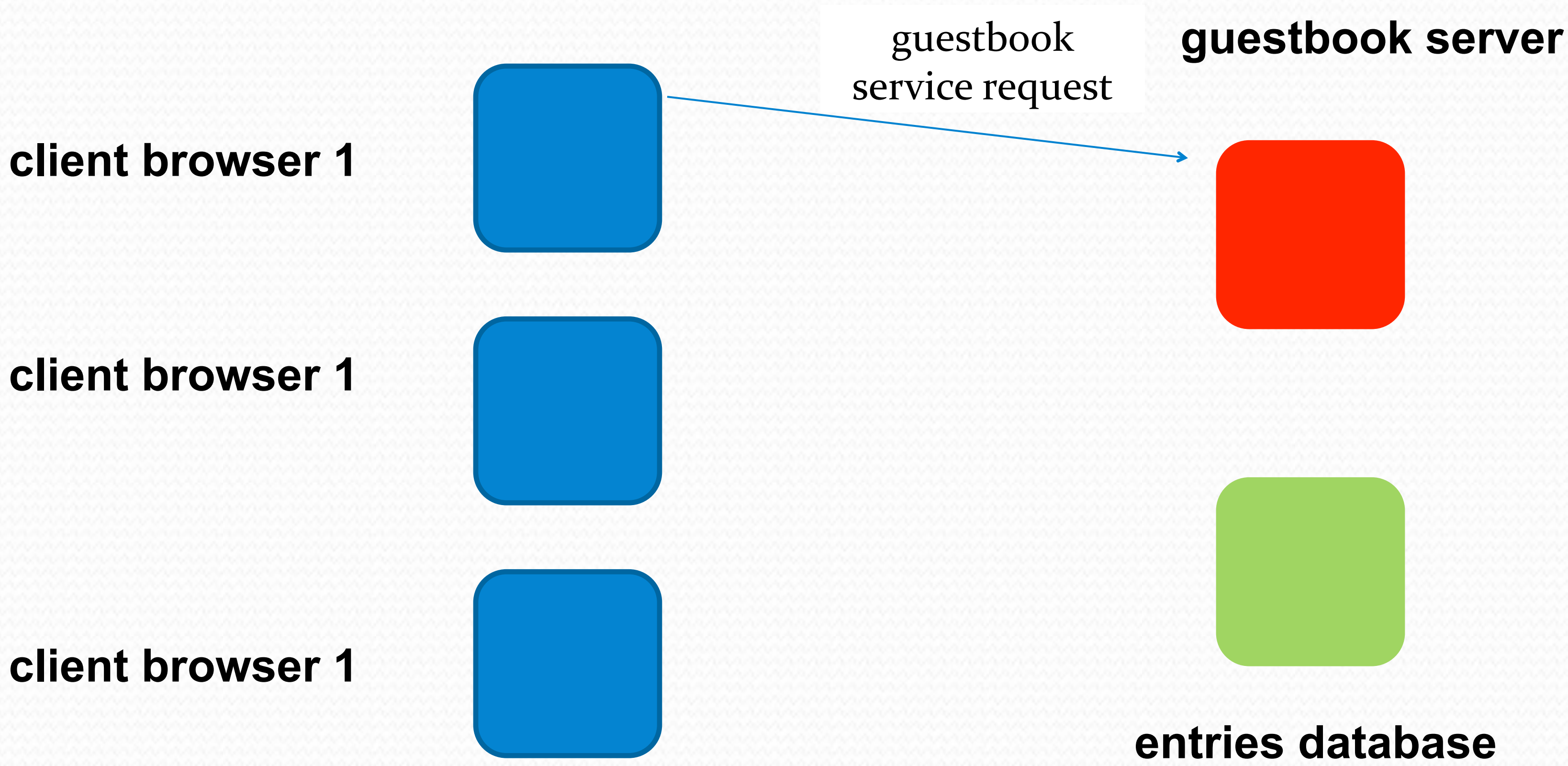
Code Injection

An untrusted source introduces code to change the course of "normal" execution

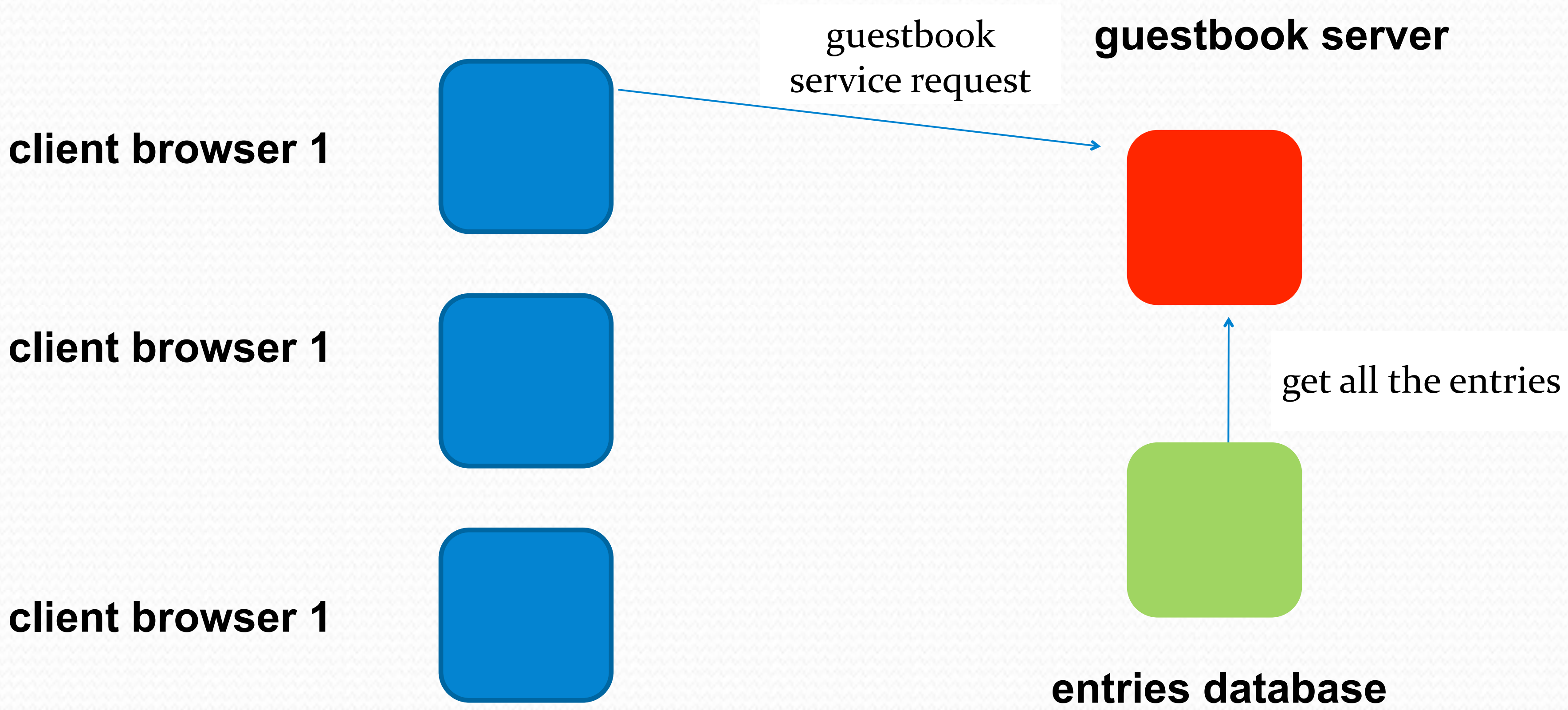
A code injection example

```
<script>hop_innerHTML_set
  ((document.getElementById( "allentries")),"You
  can only see this entry!!")</script>
```

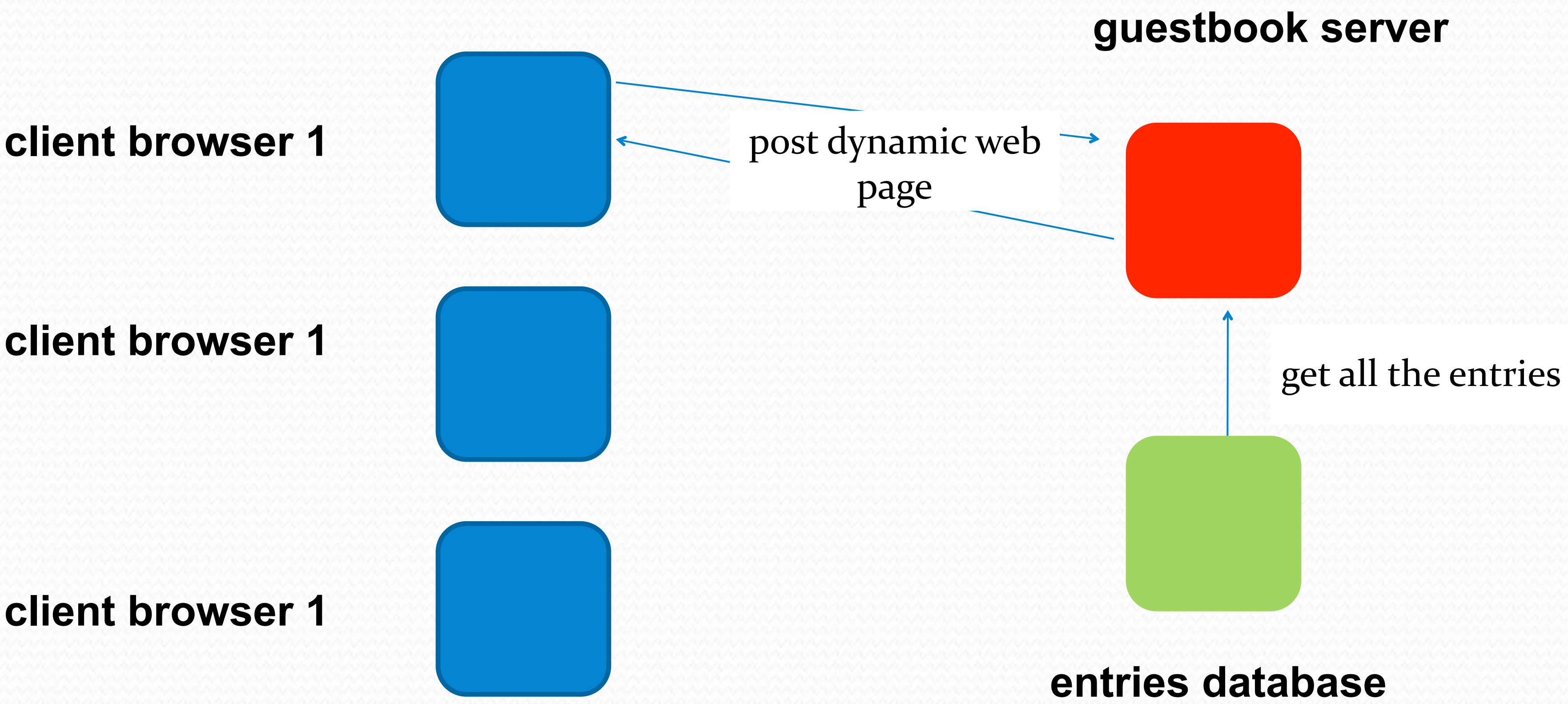
A code injection example



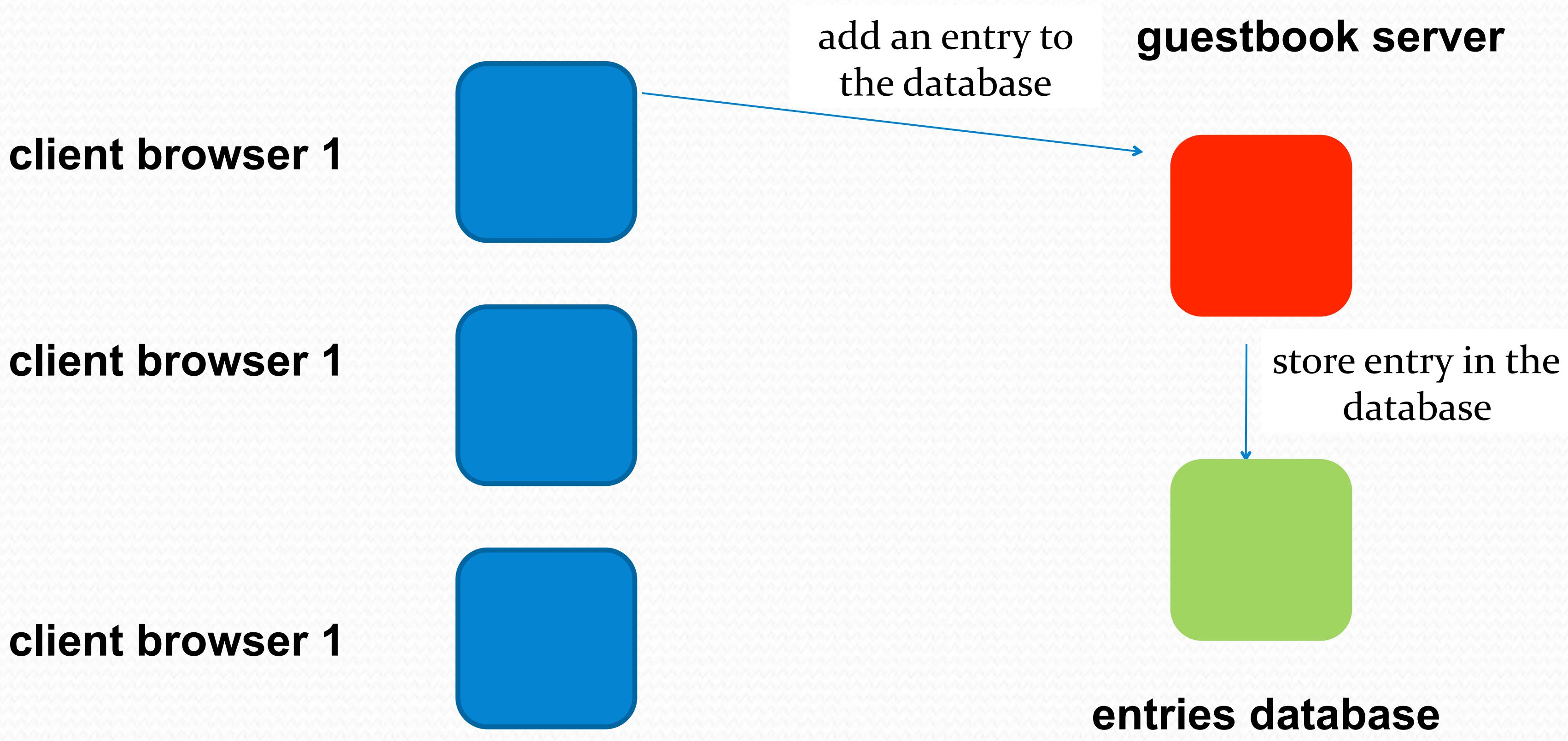
A code injection example



A code injection example



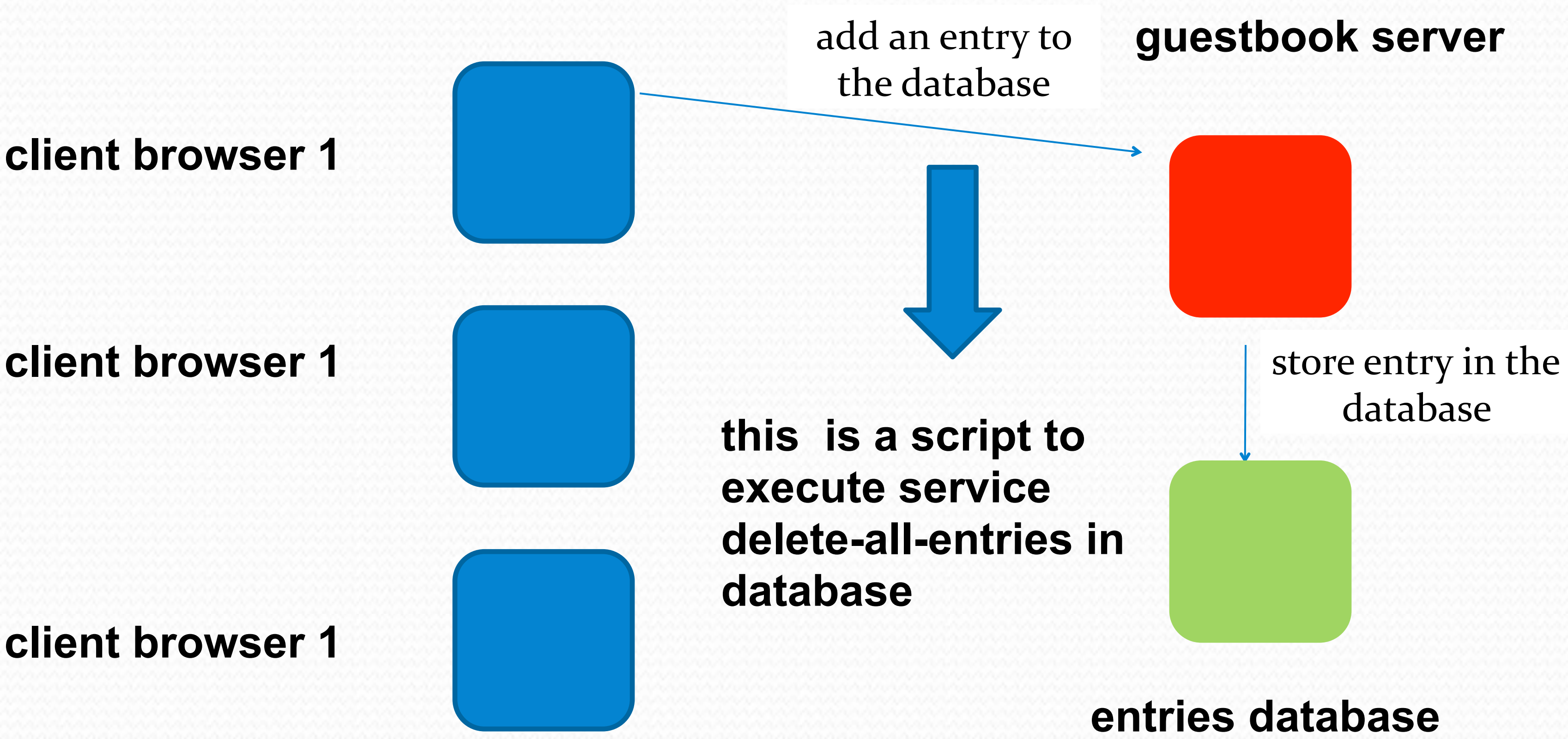
A code injection example



Cross Site Scripting (XSS)

Untrusted code in injected in a dynamic generated web page that is executed in a trusted environment

XSS example

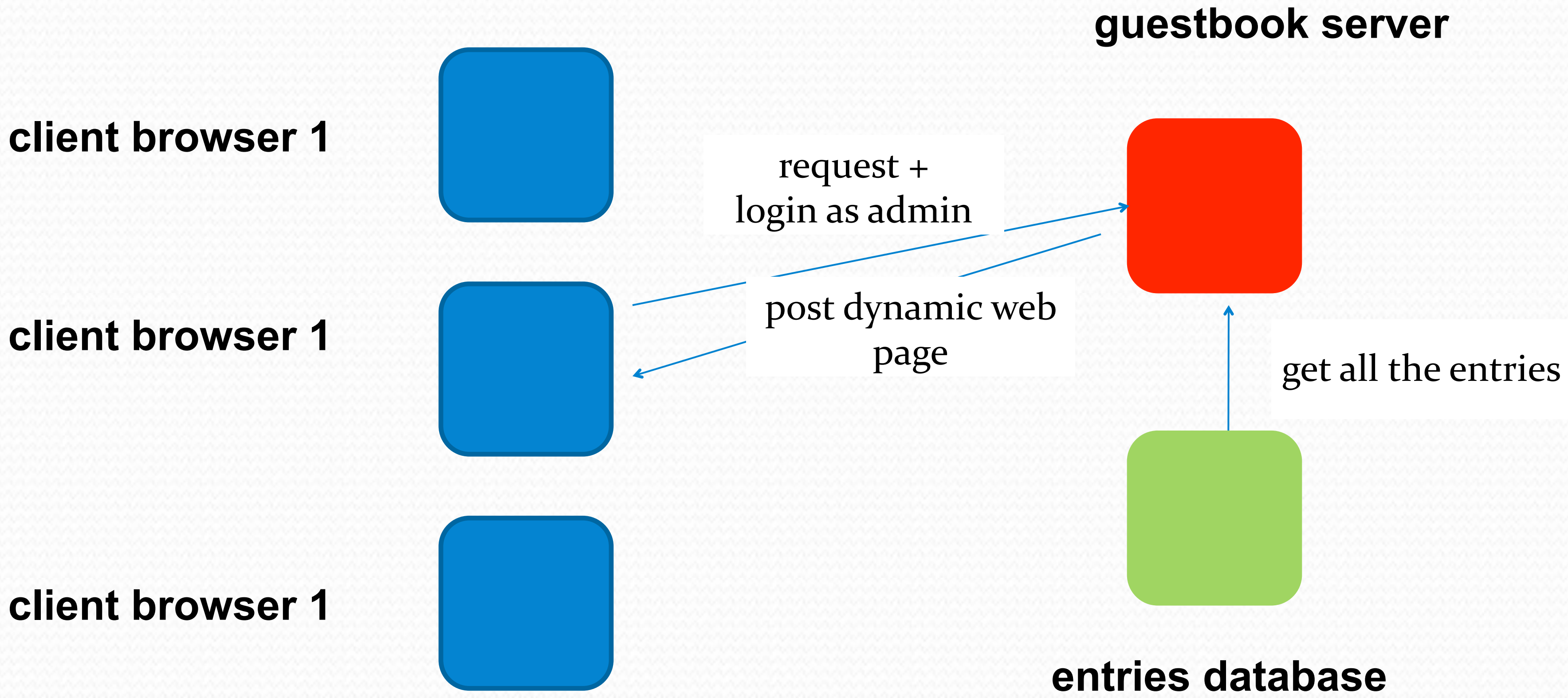

add an entry to the database
guestbook server
client browser 1
client browser 1
client browser 1
store entry in the database
this is a script to execute service delete-all-entries in database
entries database

XSS example

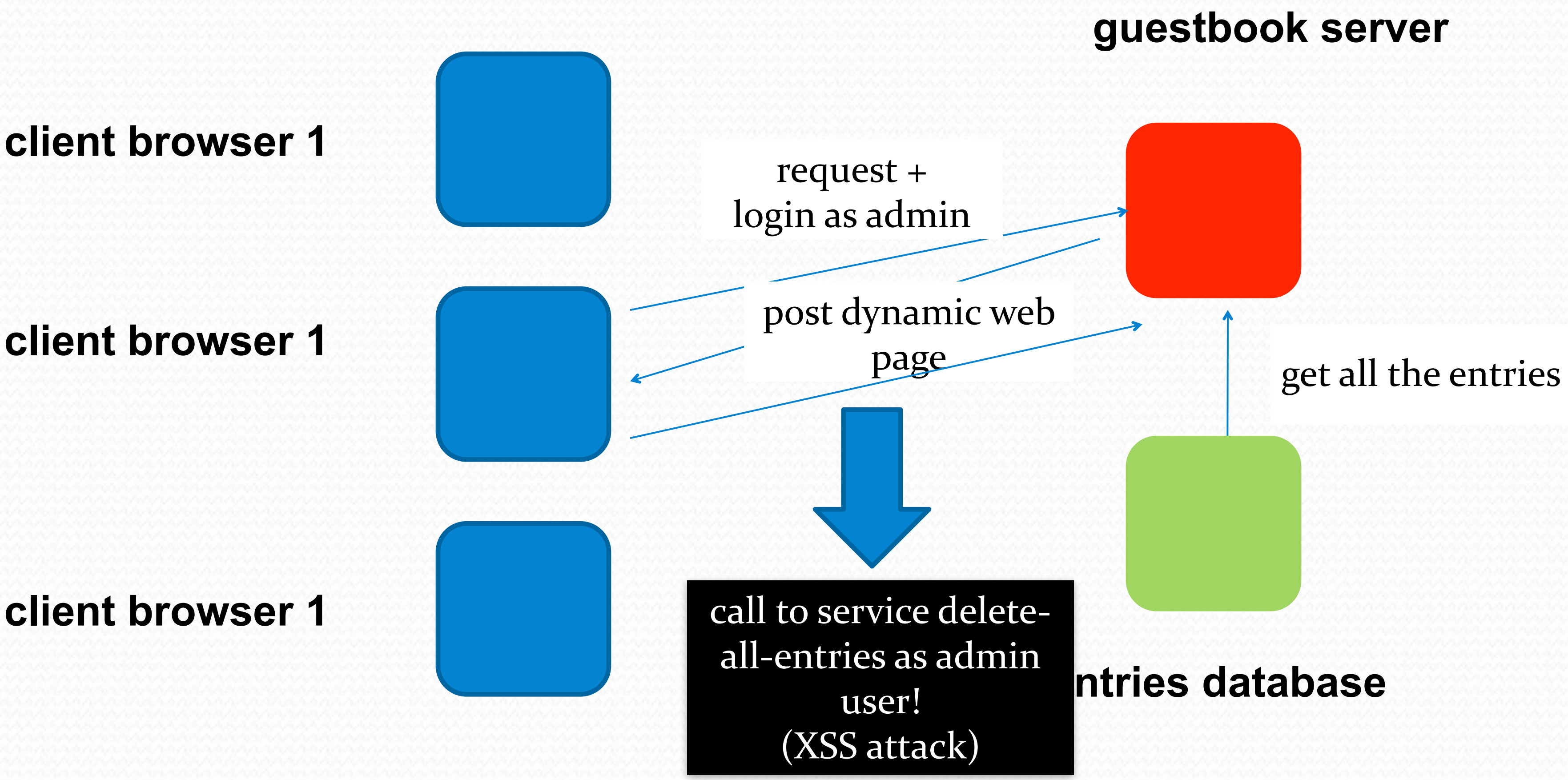
- The script to execute service delete-all-entries is stored in the database (but it cannot be executed without admin rights)

- The next user to log in : the admin user

XSS example


guestbook server
client browser 1
request +
login as admin
client browser 1
post dynamic web
page
get all the entries
client browser 1
entries database

XSS example



guestbook server
client browser 1
client browser 1
client browser 1
request +
login as admin
post dynamic web
page
get all the entries
call to service delete-
all-entries as admin
user!
(XSS attack)
ntries database

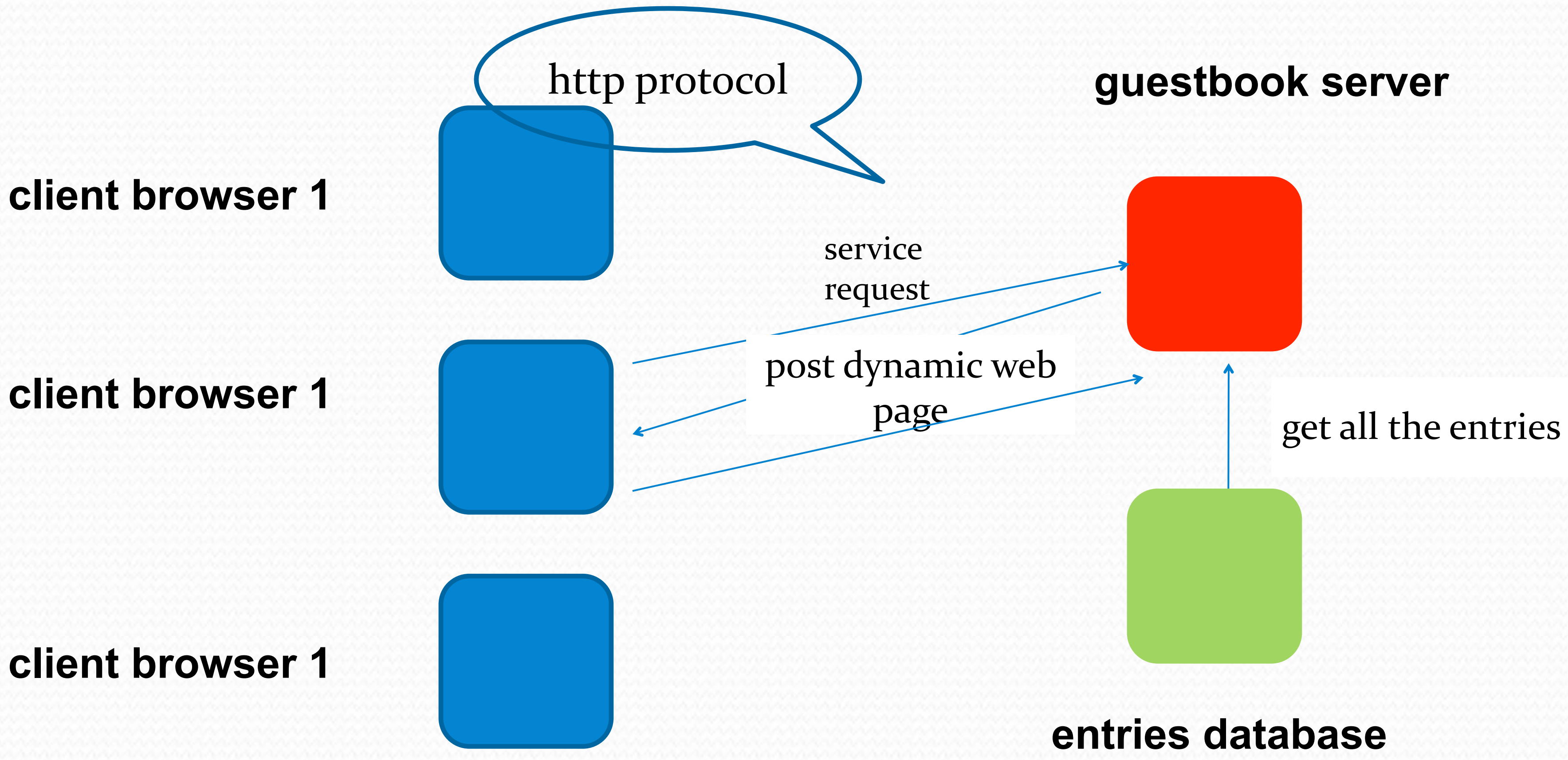
Code injection: more formally

The dynamically generated web page should behave as the specification.

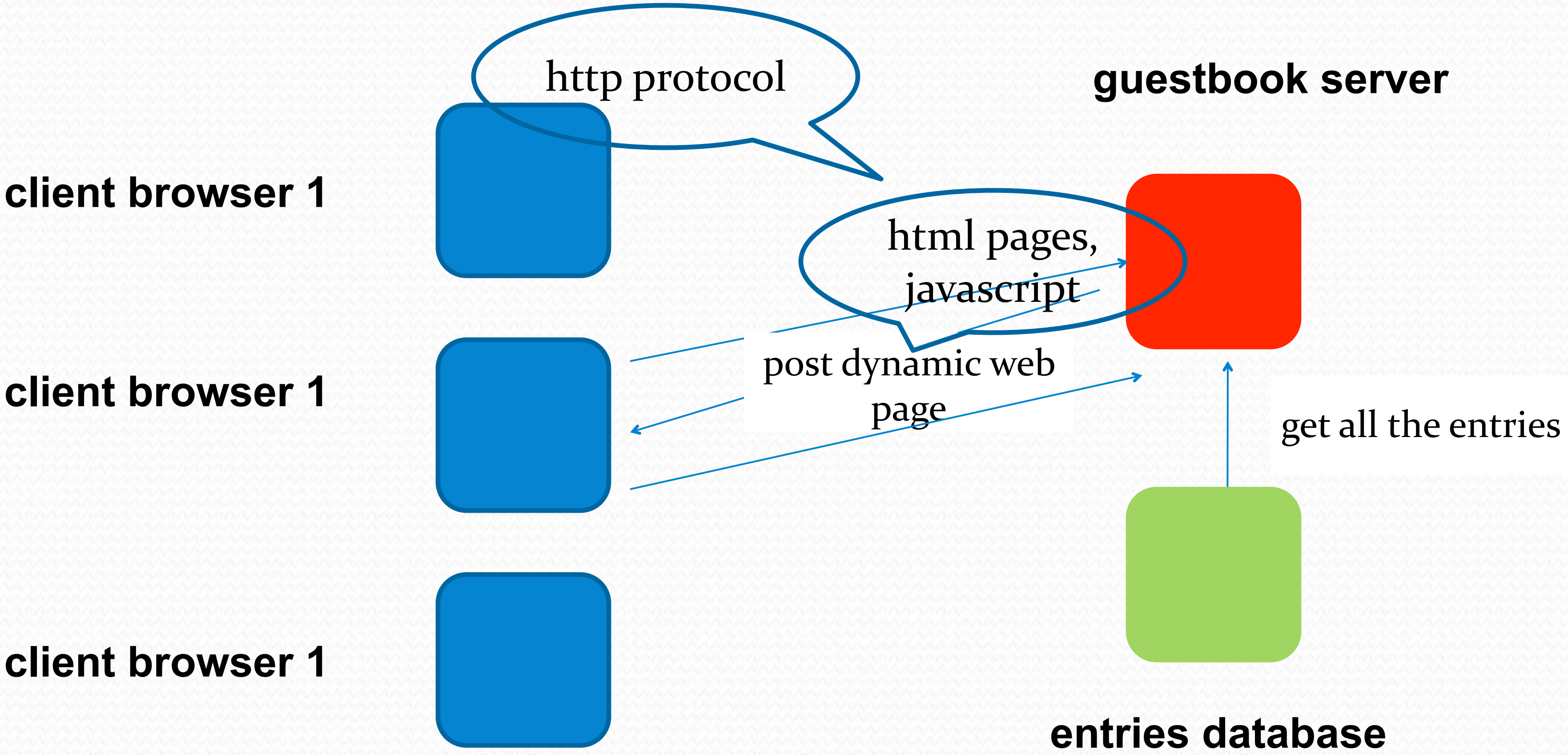
Security against code injection definition:

- a formal specification of the expected behaviour of dynamically generated web pages
- an application is secure if it always generates web pages that are observational equivalent to their specifications

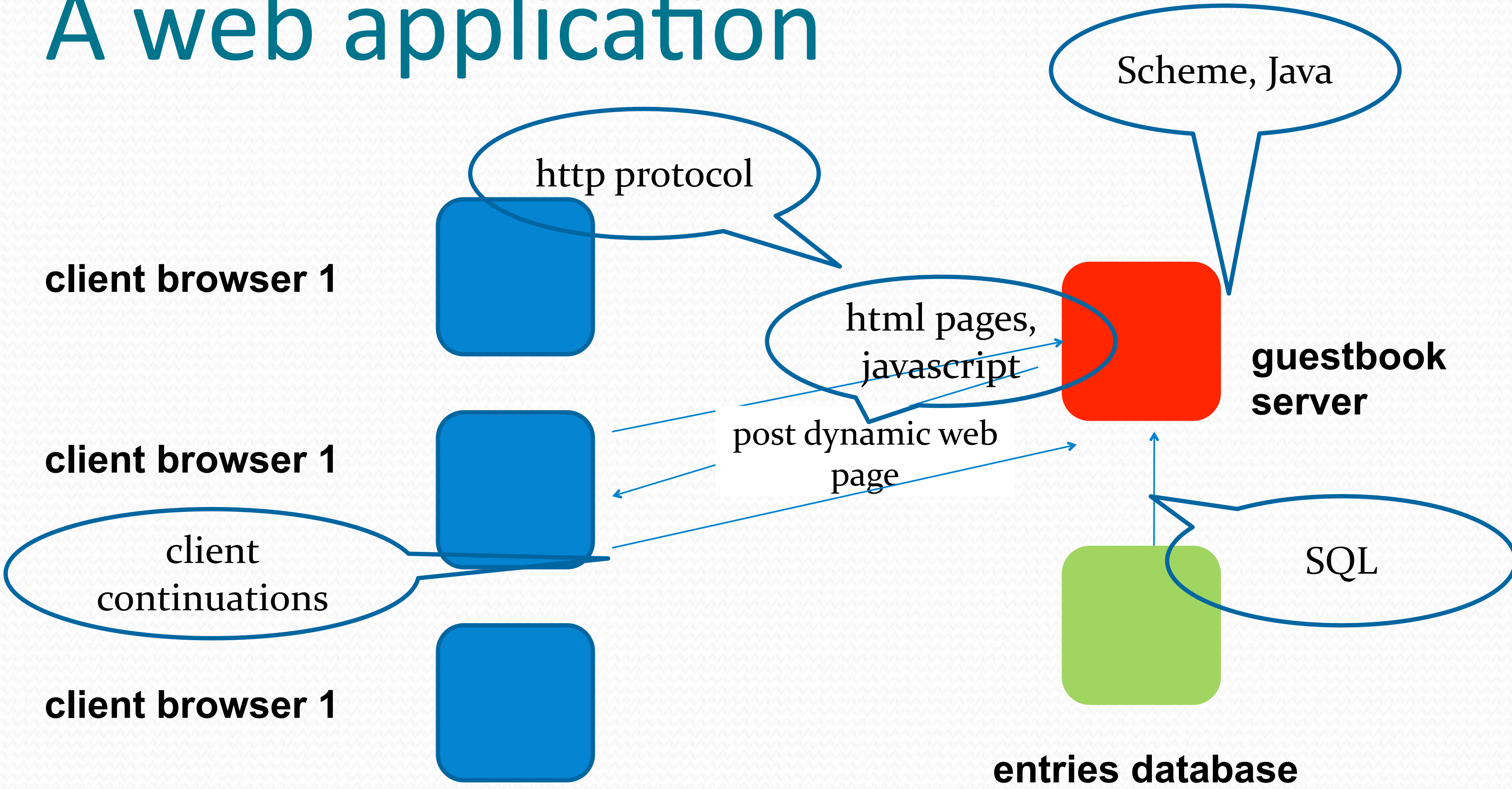
A web application



A web application



A web application



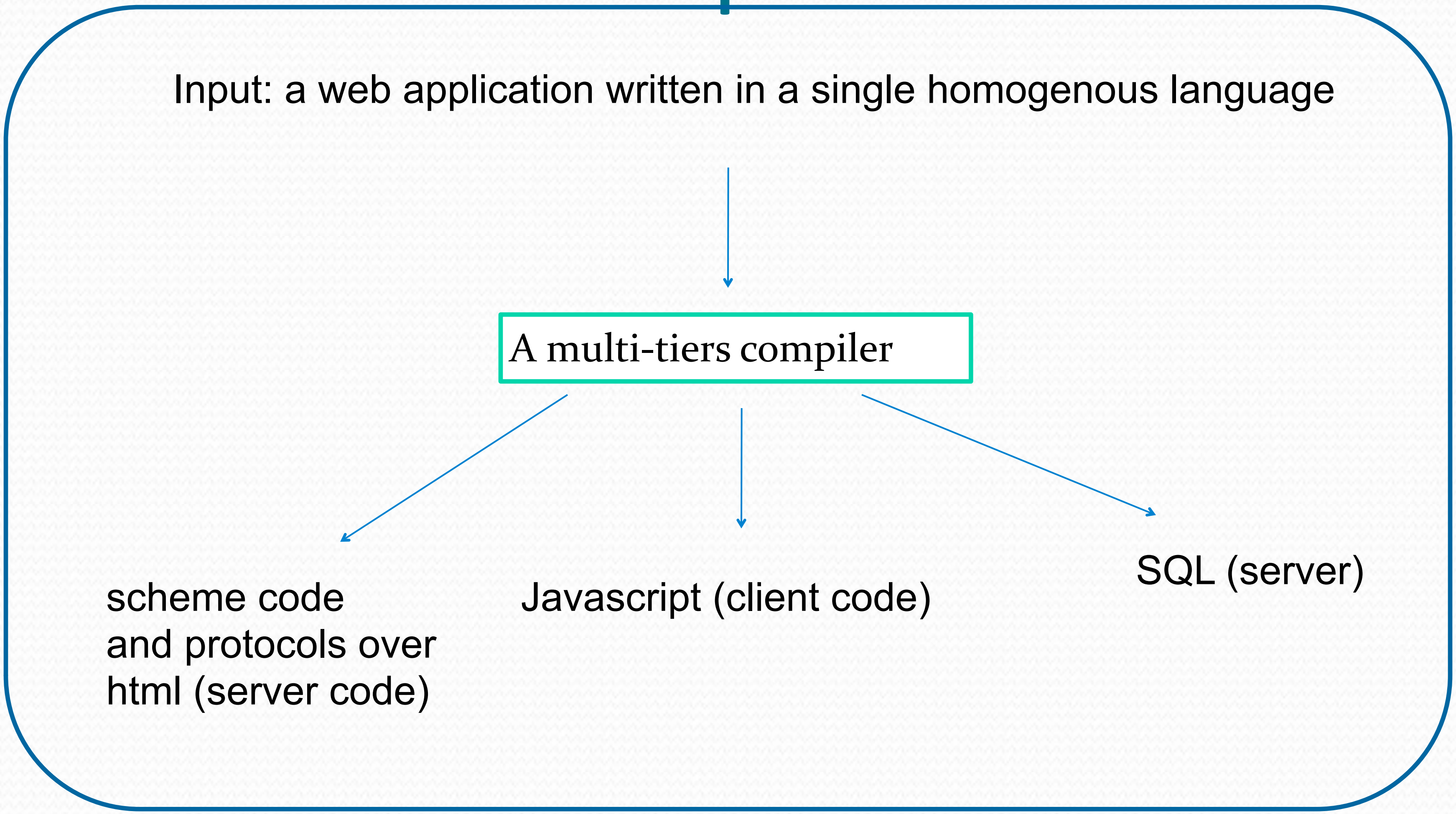
How to formally define code injection?

Unless there is a prior, generally-accepted mathematical definition of a language at hand, who is to say whether a proposed implementation is correct? (Dana Scott 1969)

Formal specification of a web application might be complex since it involves several heterogeneous languages and technologies.

A solution → **Multi-tiers compilers**

Multi-tiers compilers



Multi-tiers compilers

Examples:

Links: Philip Wadler et al
Hop: Manuel Serrano and Florian Loitsch
Google Web Toolkit: Google
Swift: Andrew Myers et al

Hop (simplification)

Syntax

$$
\begin{array}{rcl}
e^{\mathsf{s}} & ::= & e \\
 & | & (tag\ e^{\mathsf{s}} \ldots) \\
 & | & \tilde{}\,e^{\mathsf{cs}} \\
tag & ::= & \mathsf{html} \mid \mathsf{head} \mid \mathsf{div} \mid \mathsf{canvas} \ldots \\
v^{\mathsf{s}} & ::= & v \mid \tilde{}\,e^{\mathsf{c}} \mid @p
\end{array}
\qquad
\begin{array}{rcl}
e^{\mathsf{cs}} & ::= & e \\
 & | & \$e^{\mathsf{s}} \\
 & & \\
 & & \\
p & ::= & pointers
\end{array}
$$

Configuration

$$\boxed{(\,id\,,\, m,\, h,\, e)}$$

expression

id for client
or server

environment

heap

Hop (simplification)

small step relation

p is a fresh pointer

(S , m, [], (C[(html v)])) →   (S, m, [p -> (html v), nil],C[@p])

(m,e) → *(m’,e’)*

--

(S , m, [], (C[(html e)])) →   (S, m, [],C[(html e’)])

$\boxed{(\ id\ ,\ m,\ h,\ e)}$

expression

id for client or server

environment

heap

Hop execution

$(S, m_0, h_0, e_0) \rightarrow (S, m_1, h_1, e_1) \ldots (S, m_k, h_k, e_k) \rightarrow (C, m_{k+1}, h_{k+1}, e_{k+1}) \rightarrow \ldots$

Hop execution

$(S , m_0, h_0, e_0) \rightarrow (S , m_1, h_1, e_1) \ldots (S , m_k, h_k, e_k) \rightarrow (C , m_{k+1}, h_{k+1}, e_{k+1}) \rightarrow \ldots$

compilation of ek+1
and its
environment to
javascript

HTML + Javascript
configuration

$(m_{k+1}, TREE, e_{JS}) \rightarrow \ldots$

Hop execution

$(C, m_{k+1}, h_{k+1}, e_{k+1}) \rightarrow \quad \ldots$

The compiler is NOT correct if a string in the hop expression

<script> do sth evil </script>

is interpreted as a javascript node in the target expression

compilation of ek+1 and its environment to javascript

$(m_{k+1}, TREE, e_{JS}) \rightarrow \quad \ldots$

How to detect strings that will change the behaviour?

standard practice

- standard practice: filter or validate tainted input
  - advantage: efficiency
  - disadvantage: each language requires a different filter, and this kind of problems: <script> tainted string here </script> is not detected

- advantange: the same technique applies to different languages
- disadvantage: it does not help against the insecure javascript practice (document.write, eval)

A tree comparison technique

(C , mk+1, hk+1, ek+1) → ...

Does the abstract syntax tree of ek+1 correspond to the AST of eJS?

compilation of ek+1 and its environment to javascript+html

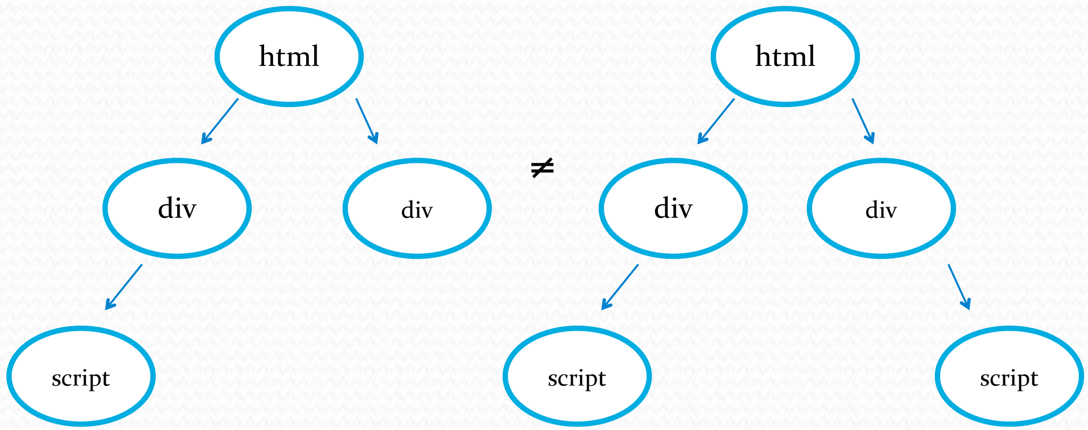
(mk+1, TREE, eJS) → ...

A tree comparison technique

(C , mk+1, hk+1, ek+1) → ...

compilation of ek+1
and its
environment to
javascript+html

Does the abstract syntax tree of ek
+1 correspond to the AST of eJS?



(mk+1, TREE, eJS) → ...

Conclusions

- We have:
    - a theorem for a toy Hop language that does not include any DOM operations such as “dom-append-child(node, tree)”
    - a Hop implementation of the technique (-s2 flag) that disallows insecure javascript (eval, (SCRIPT ..) hop constructor, document.write, call to unknown javascript functions)
- Our wish list:
    - understand how to prove compiler correctness with DOM operations (innerHTML)
    - understand whether we are being too restrictive for the disallowed JS