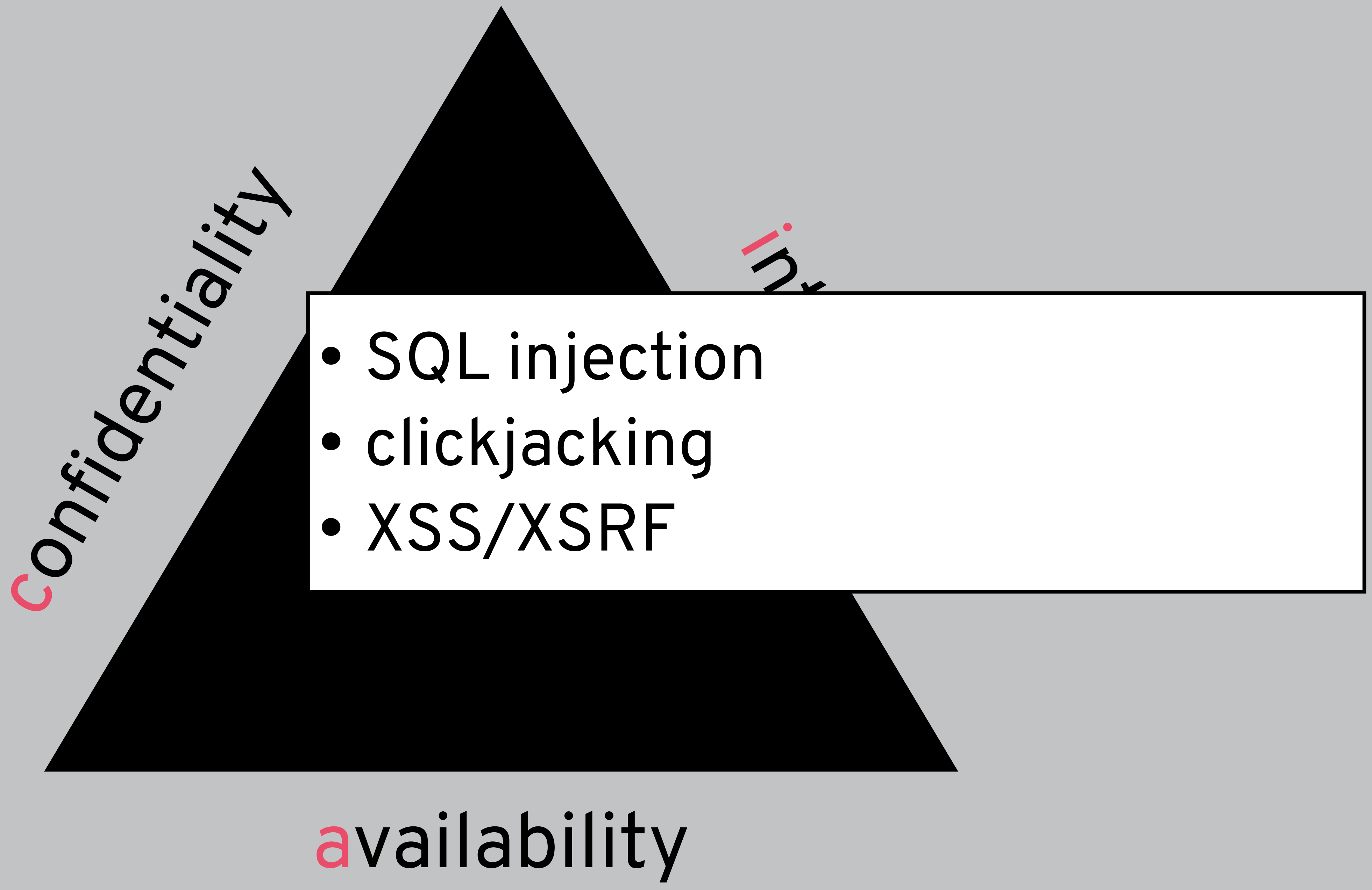
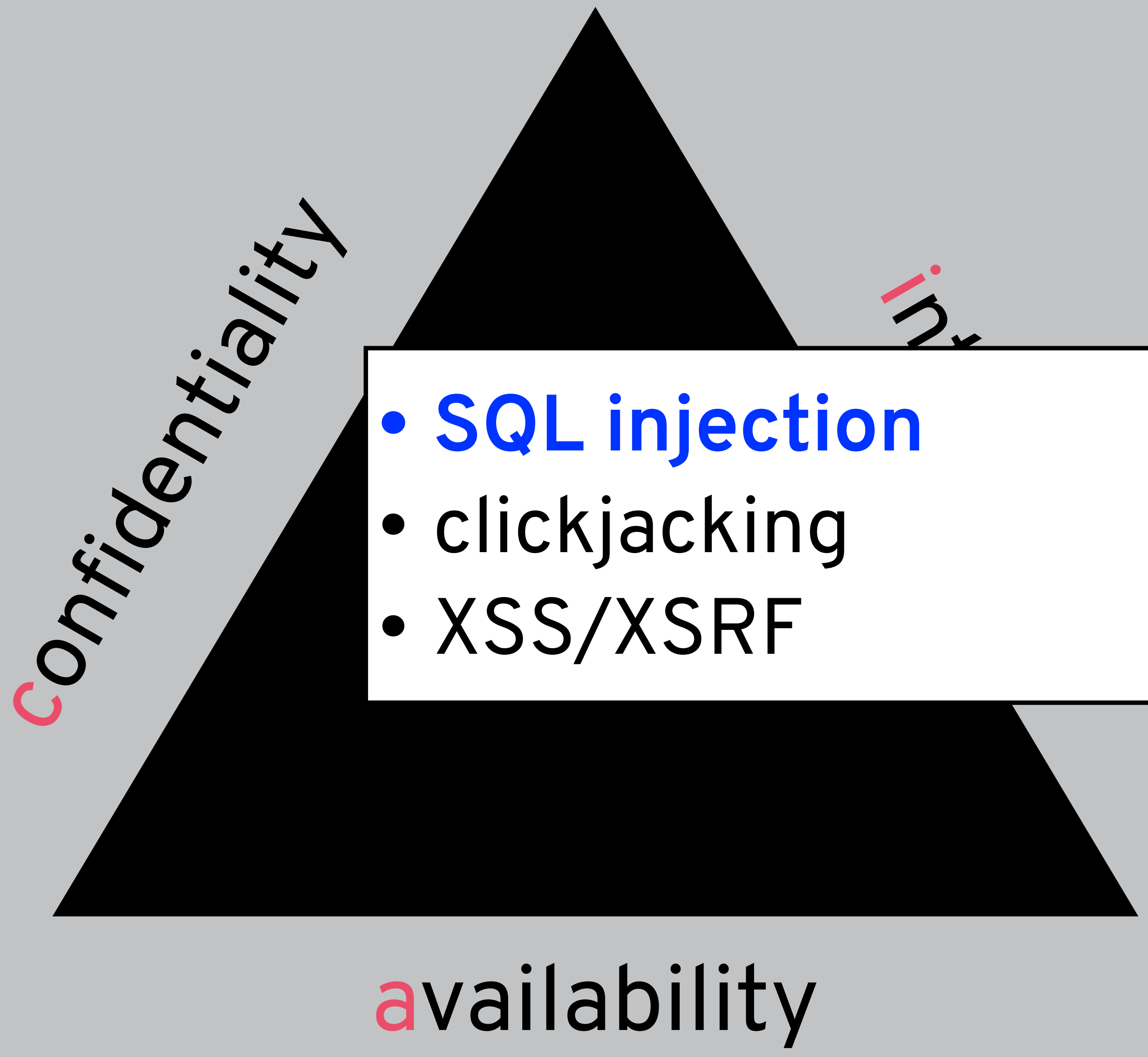
SECURITY (COMP0141): ATTACKS ON INTEGRITY

INTEGRITY, REVISITED



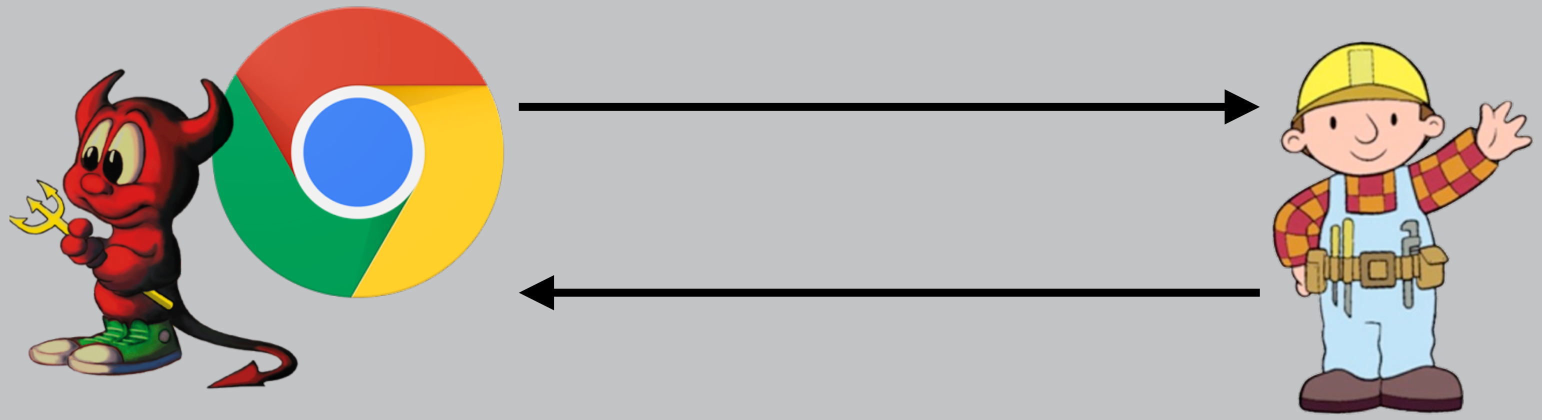
- SQL injection
- clickjacking
- XSS/XSRF

INTEGRITY, REVISITED



- **SQL injection**
- clickjacking
- XSS/XSRF

THREAT MODEL



Is the user trusted by the server? or the browser?

- SQL injection
- Click fraud

SQL BASICS

SQL = Structured Query Language

```
SELECT * FROM shop WHERE price < 10 ORDER BY type;
```

Logical expressions: `AND, OR, NOT`

Comment: `--`

Statement terminator: `;`

SQL INJECTION

Server-side applications generate SQL queries based on arbitrary user input

```
query = “SELECT count(*) FROM users WHERE
user_name =‘“ + req.getParam(“user”) + “‘“ + “ AND
user_pass =‘“ + req.getParam(“pass”) + “‘“;
```

What’s the big deal?

SQL INJECTION

Server-side applications generate SQL queries based on arbitrary user input

```
query = “SELECT count(*) FROM users WHERE
user_name =‘“ + req.getParam(“user”) + “‘“ + “ AND
user_pass =‘“ + req.getParam(“pass”) + “‘“;

“user” = alice’ ;--
query = “SELECT count(*) FROM users WHERE
user_name = ‘alice’ ;-- ...”

“user” = alice
“pass” = foo’ OR 1=1 ;--
query = “SELECT count(*) FROM users WHERE
user_name = ‘alice’ AND user_pass = ‘foo’ OR 1=1;”
```

SQL INJECTION

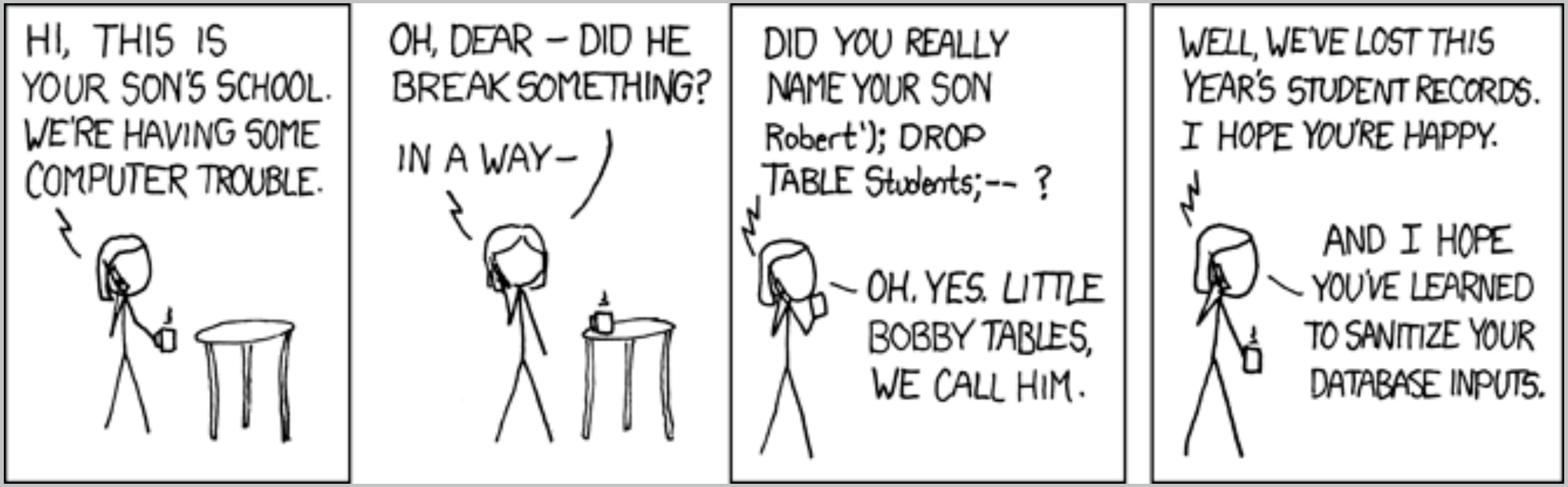
Server-side applications generate SQL queries based on arbitrary user input

```
query = “SELECT count(*) FROM users WHERE
user_name =‘“ + req.getParam(“user”) + “‘“ + “ AND
user_pass =‘“ + req.getParam(“pass”) + “‘“;
```

More generally, can execute any SQL command

```
“user” = alice’; DROP TABLE users;--
query = “SELECT count(*) FROM users WHERE
user_name = ‘alice’; DROP TABLE users;– ...”
```

LITTLE 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.

SQL INJECTION MITIGATIONS

Server-side applications generate SQL queries based on arbitrary user input

Solution? Don't accept arbitrary user input!

Parameterised queries: pre-compiled queries that separate commands from input

Input sanitisation: make sure only safe input is accepted

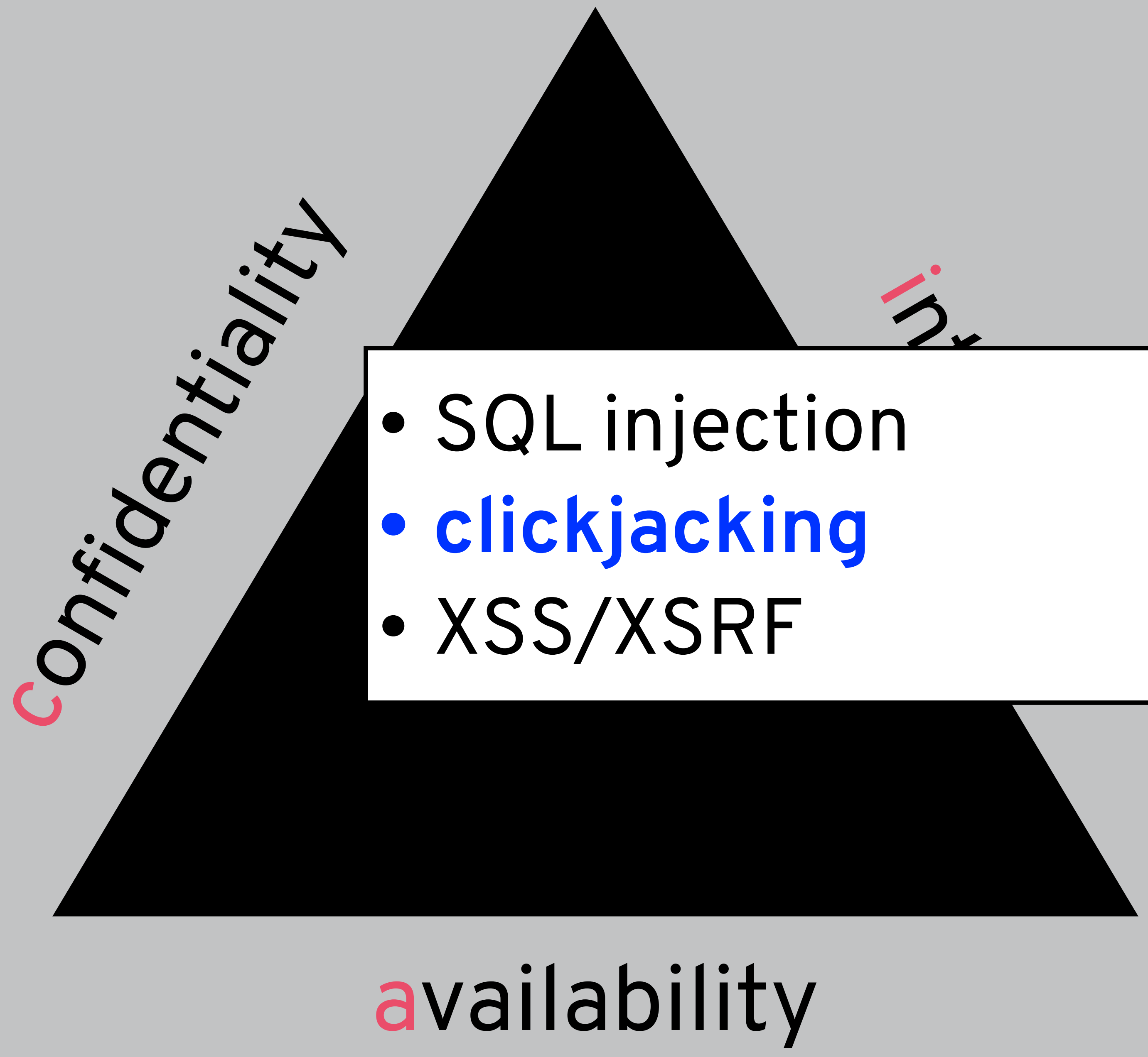
What is unsafe?

- Single quote? Dashes? But these could be legitimate

Sanitise on the client or the server?

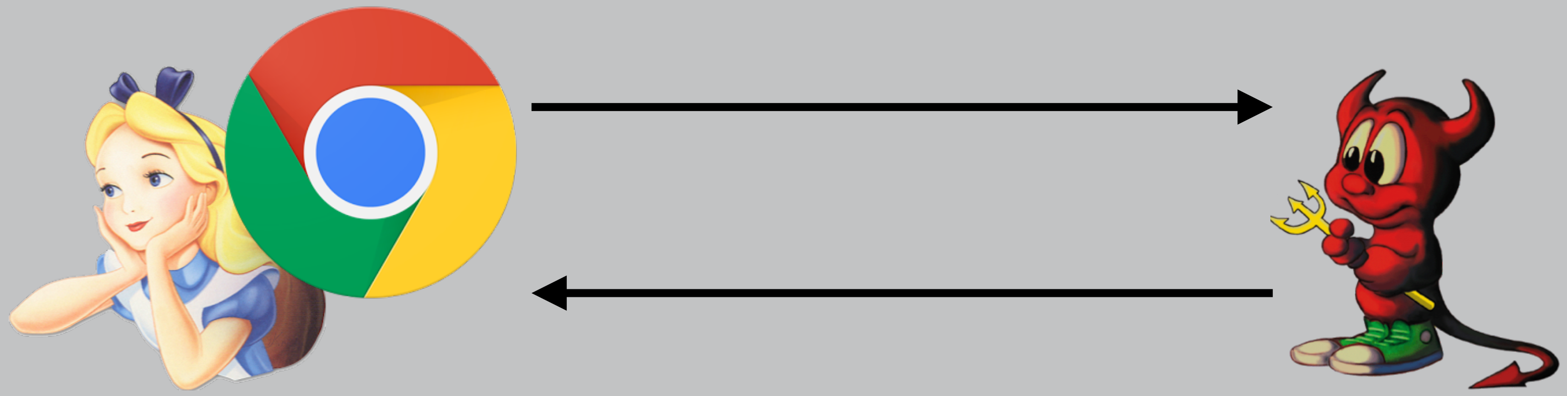
Use proper escaping/encoding?

INTEGRITY, REVISITED



- SQL injection
- **clickjacking**
- XSS/XSRF

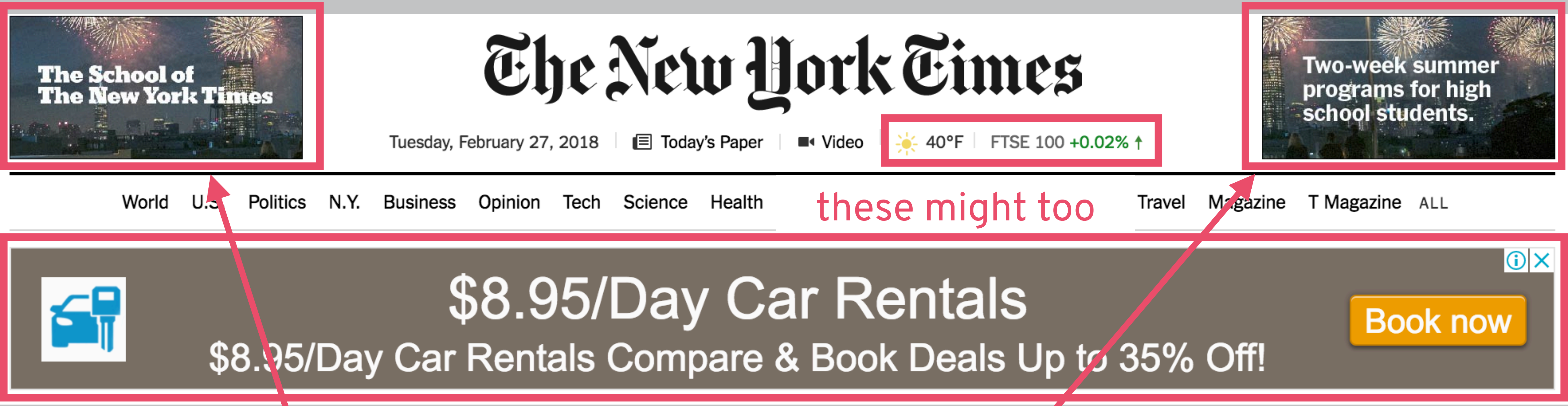
THREAT MODEL



Is the server trusted by the browser? or the user?

- Browser fingerprinting
- Forward secrecy / revocation
- Typosquatting / pharming
- Clickjacking

HOW DOES THE MODERN WEB WORK?



The School of The New York Times

The New York Times

Tuesday, February 27, 2018

Two-week summer programs for high school students.

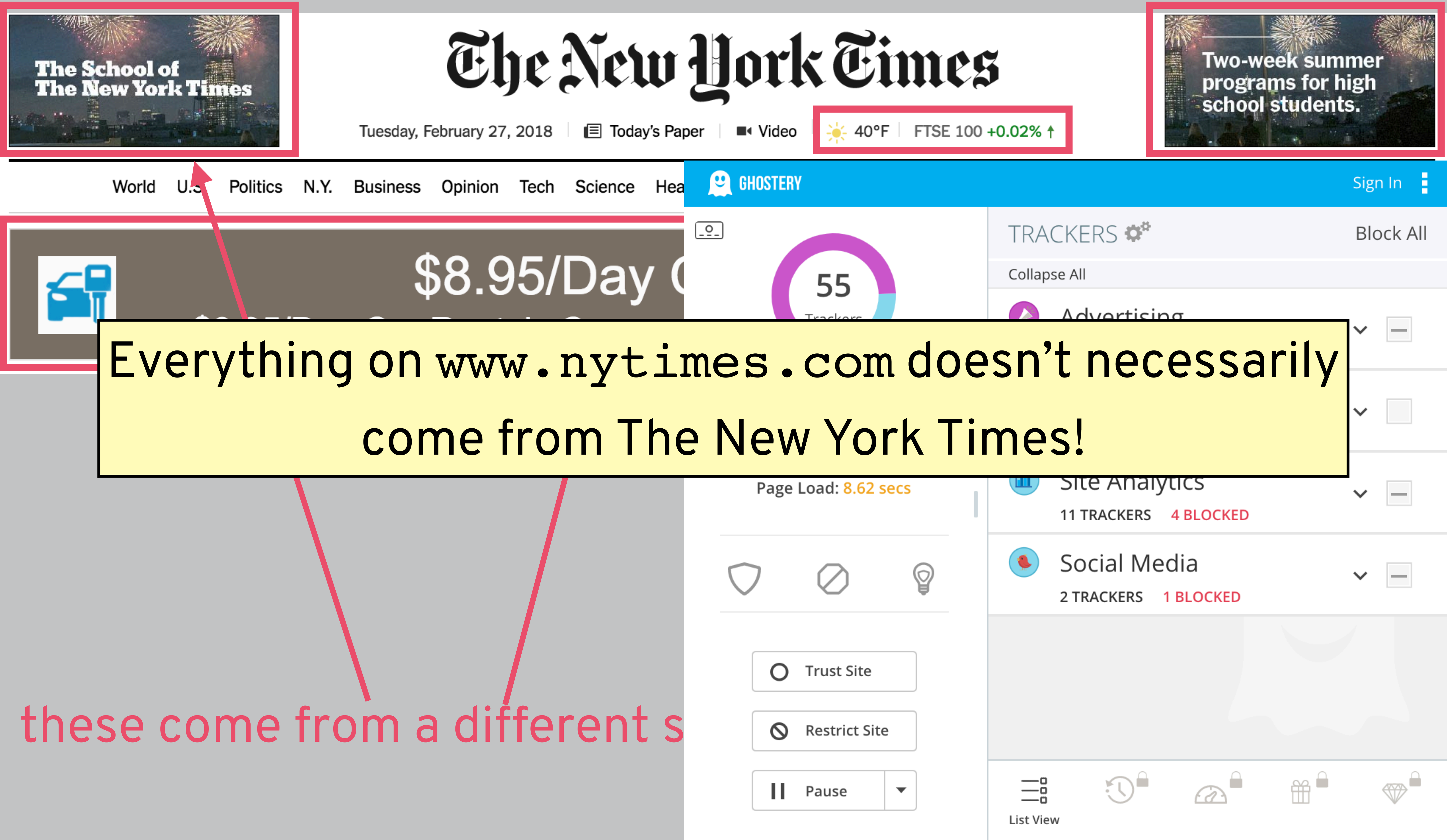
$8.95/Day Car Rentals

$8.95/Day Car Rentals Compare & Book Deals Up to 35% Off!

these might too

these come from a different site

HOW DOES THE MODERN WEB WORK?



Everything on www.nytimes.com doesn’t necessarily come from The New York Times!

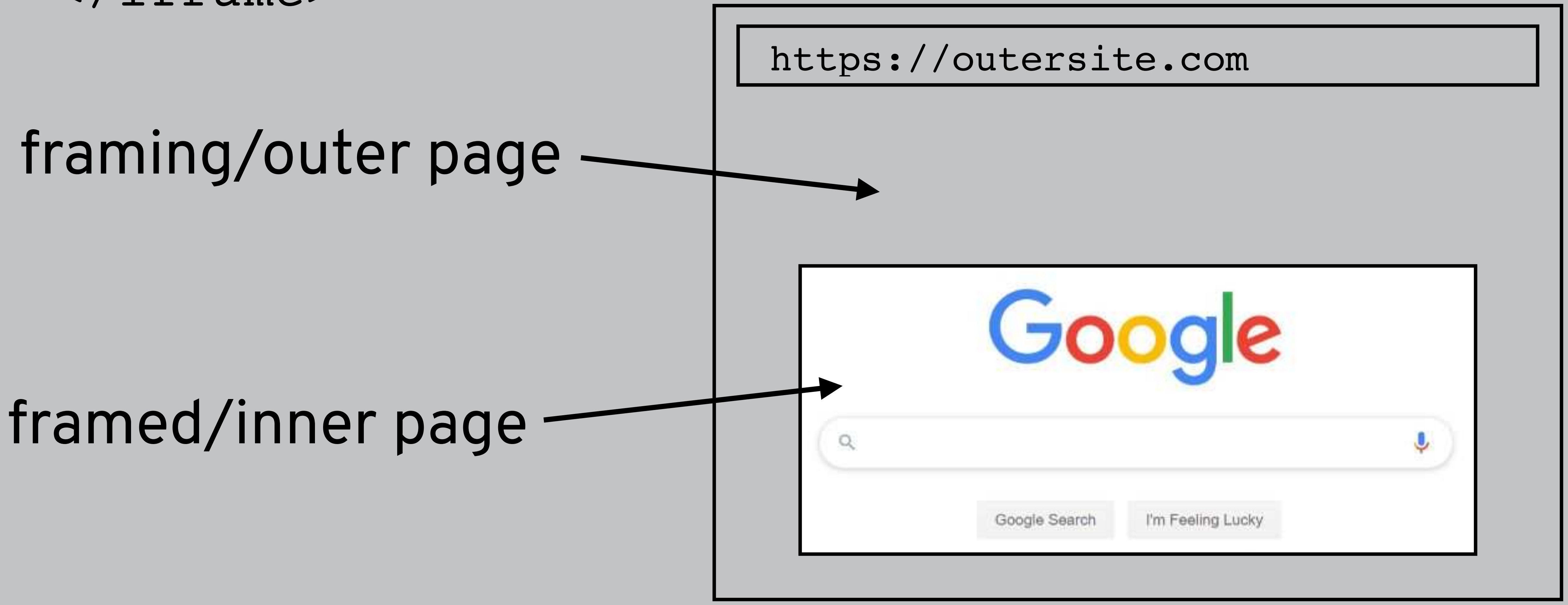
these come from a different s

IFRAMES

Content from one site is embedded into another using **iframes**

Example:

```
<iframe src=“https://www.google.com”>
</iframe>
```



framing/outer page

framed/inner page

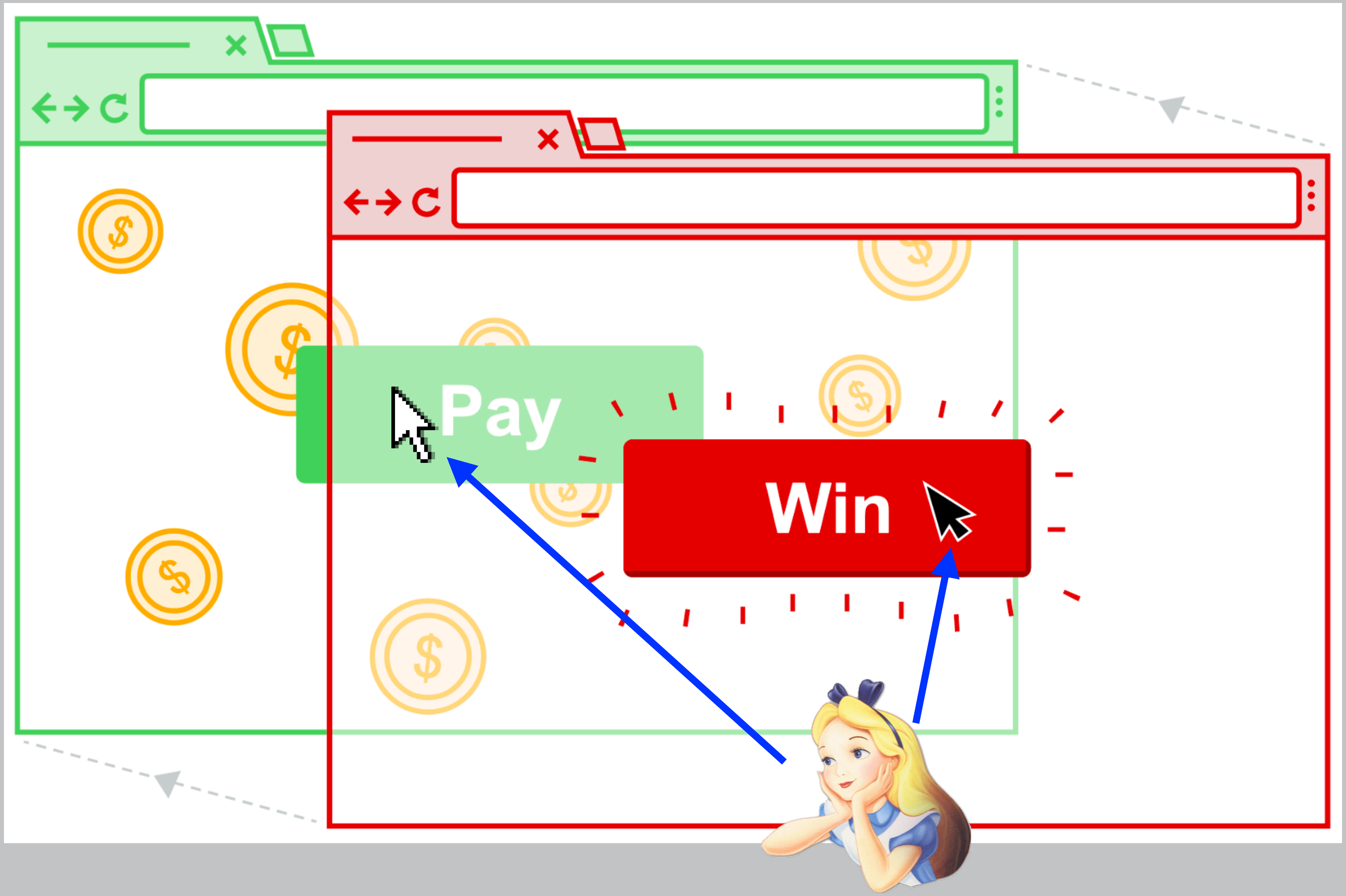
IFRAMES

Content from one site is embedded into another using **iframes**

Outer page can set the width and height of the frame

Only inner page can draw within its frame

CLICKJACKING

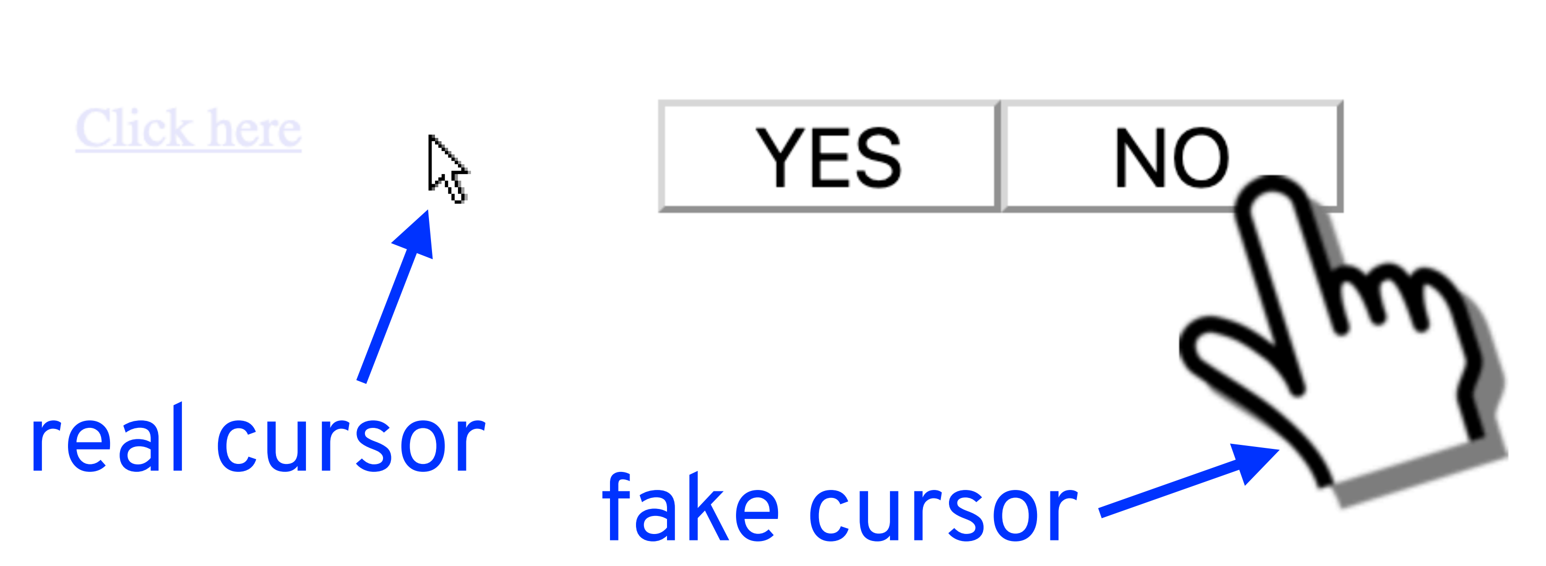


CLICKJACKING

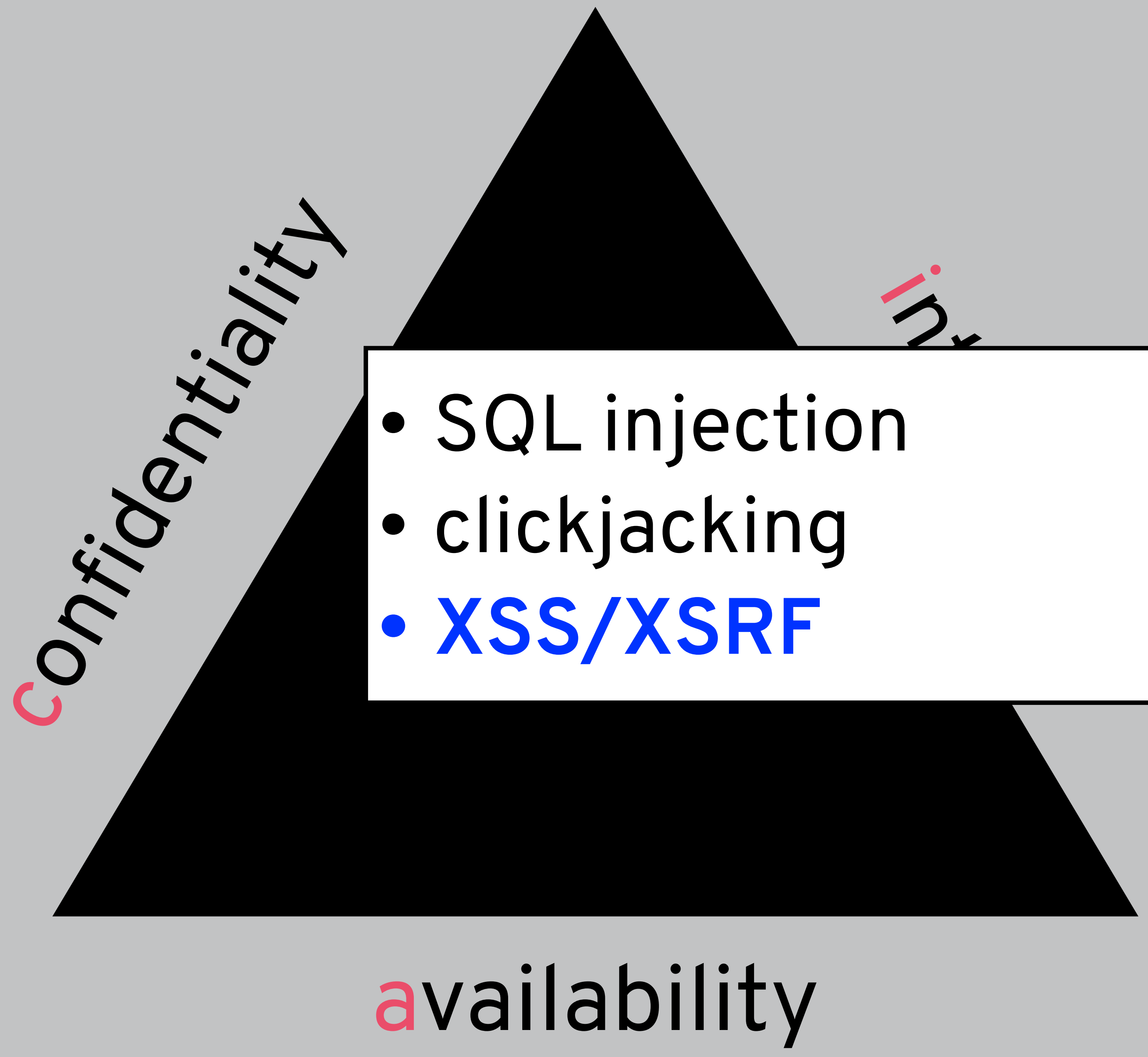
Attacker site frames good site and covers part of it to look different / create unintended interactions

Likejacking is when Facebook users are tricked into clicking the 'like' button

This is often achieved using **cursorjacking**



INTEGRITY, REVISITED



- SQL injection
- clickjacking
- **XSS/XSRF**