WEB SECURITY: CLICKJACKING

CMSC 414

FEB 27 2018

Misleading users

- Browser assumes that clicks and keystrokes = *clear indication* of what the user wants to do
  - Constitutes part of the user's *trusted path*
- Attacker can meddle with integrity of this relationship in all sorts of ways

Misleading users

- Browser assumes that clicks and keystrokes = *clear indication* of what the user wants to do
  - Constitutes part of the user's *trusted path*

- Attacker can meddle with integrity of this relationship in all sorts of ways
- Recall the power of Javascript
  - **Alter page contents (dynamically)**
  - **Track events (mouse clicks, motion, keystrokes)**
  - Read/set cookies
  - Issue web requests, read replies

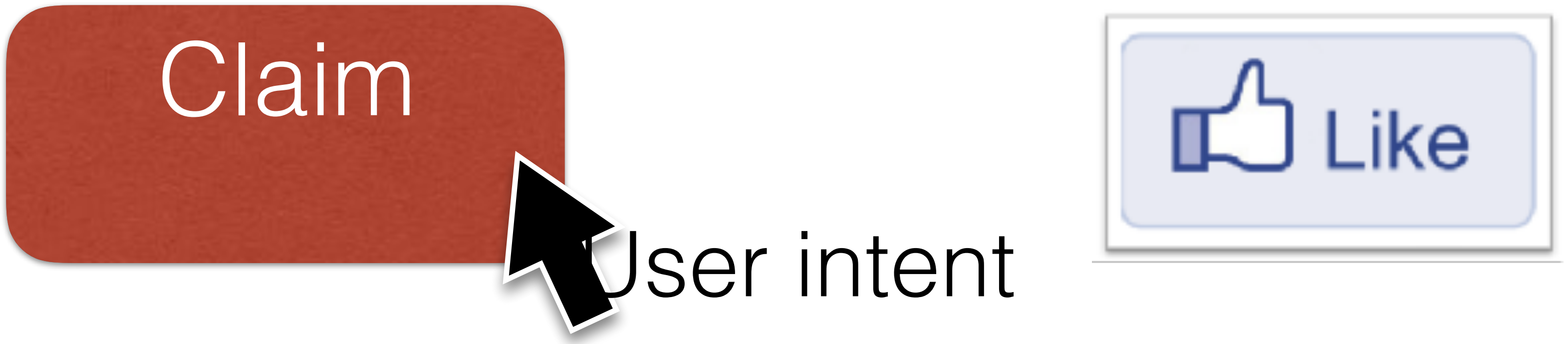
Using JS to Steal Facebook *Likes*

Bait and switch

User tries to claim their free iPad, but you want them to click your Like button

(Many of these attacks are similar to TOCTTOU vulnerabilities)

Using JS to Steal Facebook *Likes*



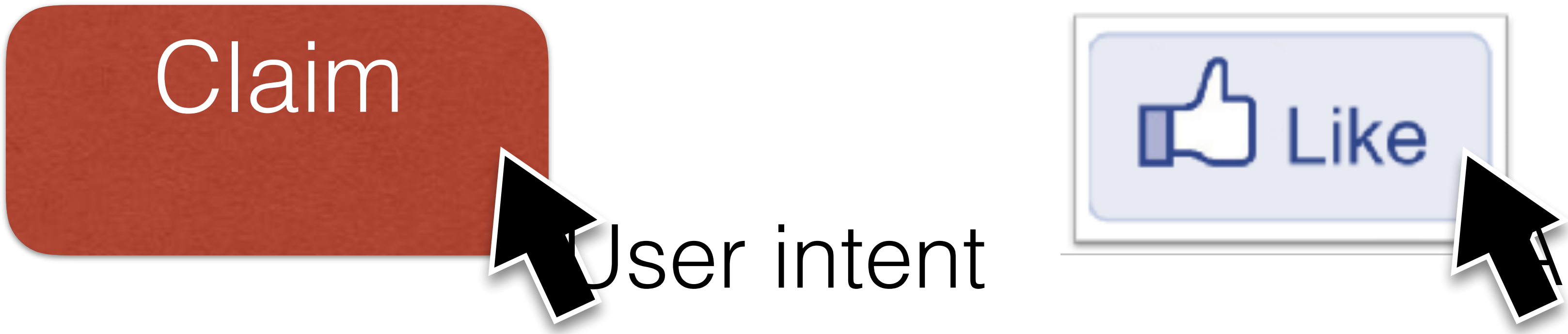
User intent

Bait and switch

User tries to claim their free iPad, but you want them to click your Like button

(Many of these attacks are similar to TOCTTOU vulnerabilities)

Using JS to Steal Facebook *Likes*



User intent

Actual outcome

Bait and switch

User tries to claim their free iPad, but you want them to click your Like button

(Many of these attacks are similar to TOCTTOU vulnerabilities)

Clickjacking

When one principal tricks the user into interacting with UI elements of another principal

An attack application (script) compromises the ***context integrity*** of another application's User Interface when the user acts on the UI

Clickjacking

When one principal tricks the user into
interacting with UI elements of another principal

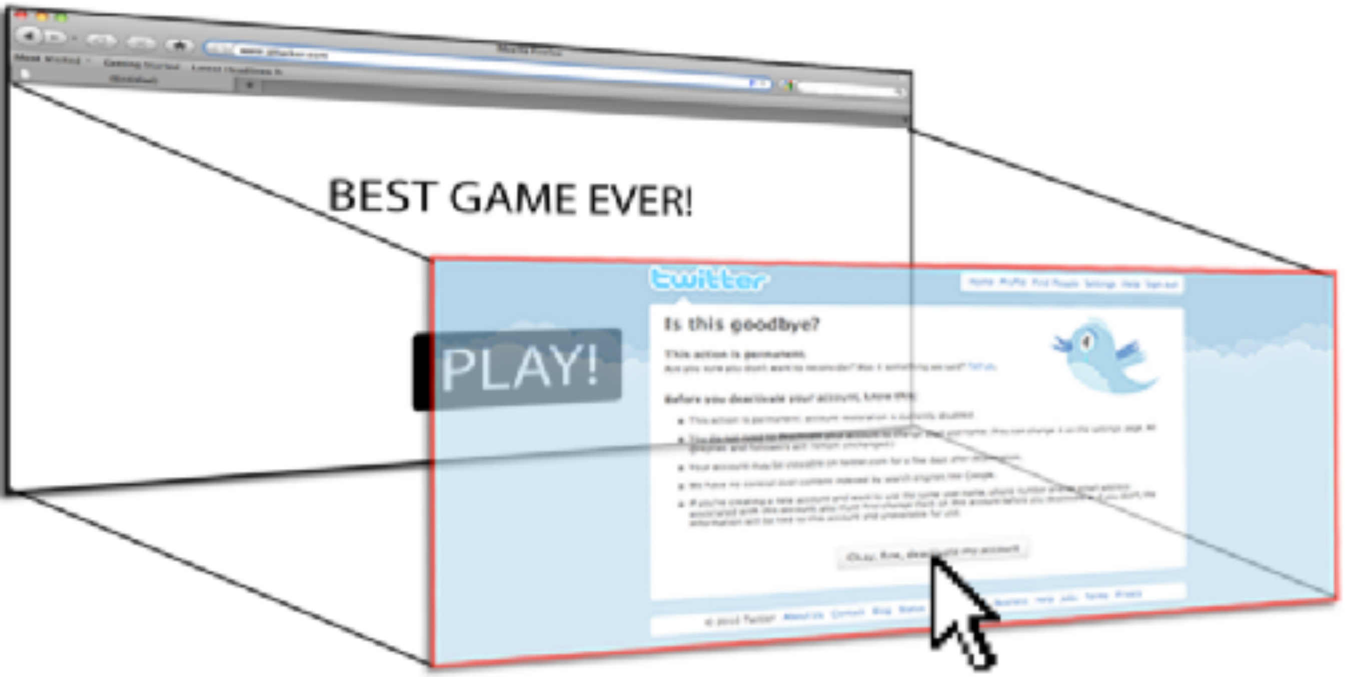
An attack application (script) compromises the ***context integrity*** of another application's User Interface when the user acts on the UI

Context Integrity

1. **Visual context**: what a user should see right bef
   the sensitive action. Ensuring this = the sensit
   UI element and the cursor are both visible
2. **Temporal context**: the timing of a user action. En
   this = the user action at a particular time is wh
   the user intended

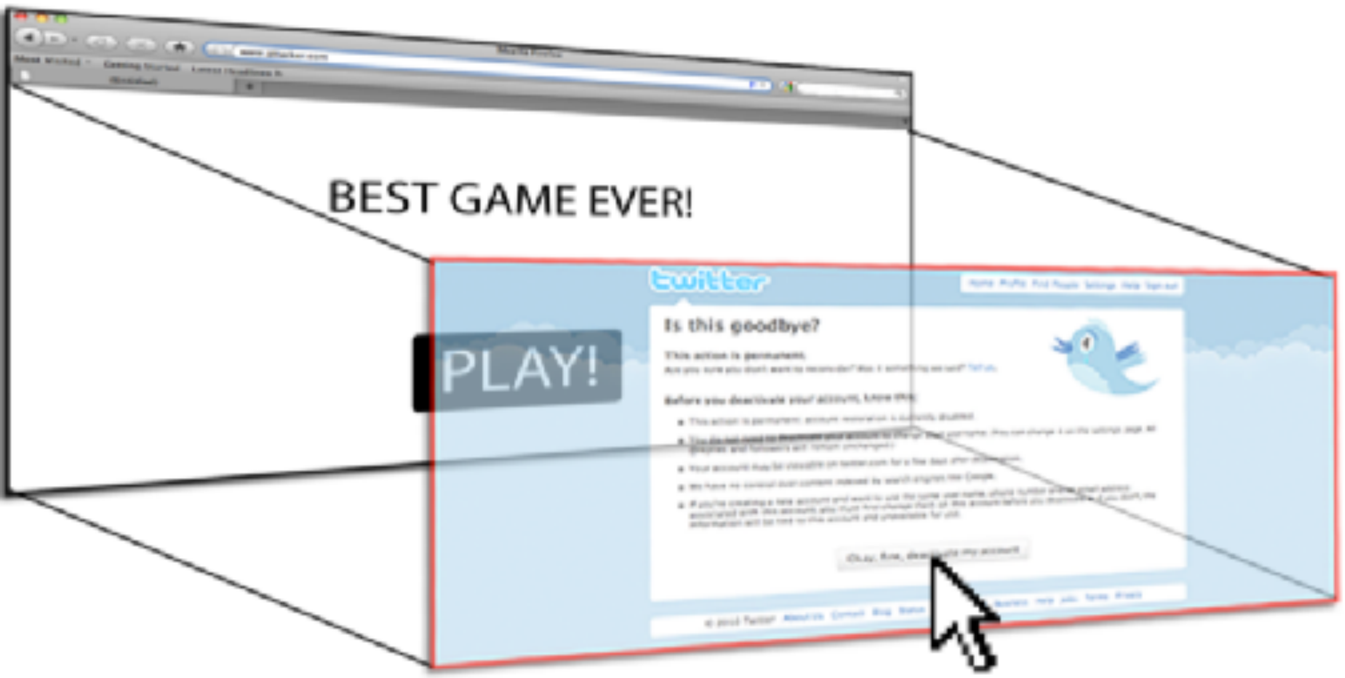
Compromising visual integrity of the *target*

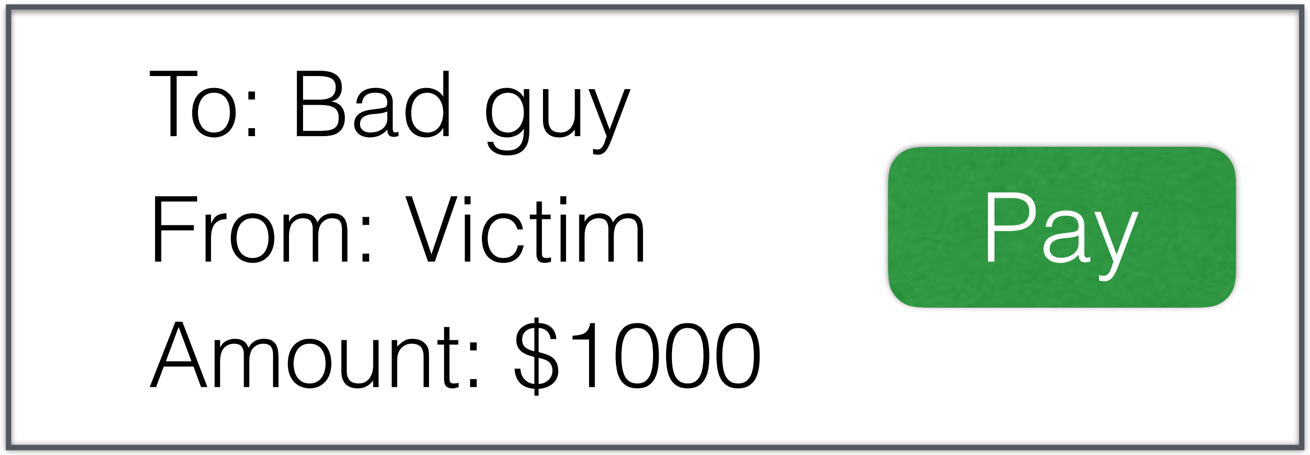
- Hide the target element
  - CSS lets you set the opacity of an element to zero (clear)


BEST GAME EVER!
PLAY!
twitter
Is this goodbye?
This action is permanent.
Before you deactivate your account, know this:
Okay, fine, deactivate my account

Compromising visual integrity of the *target*

- Hide the target element
  - CSS lets you set the opacity of an element to zero (clear)
- Partially overlay the target
  - Or *crop* the parts you don’t wa



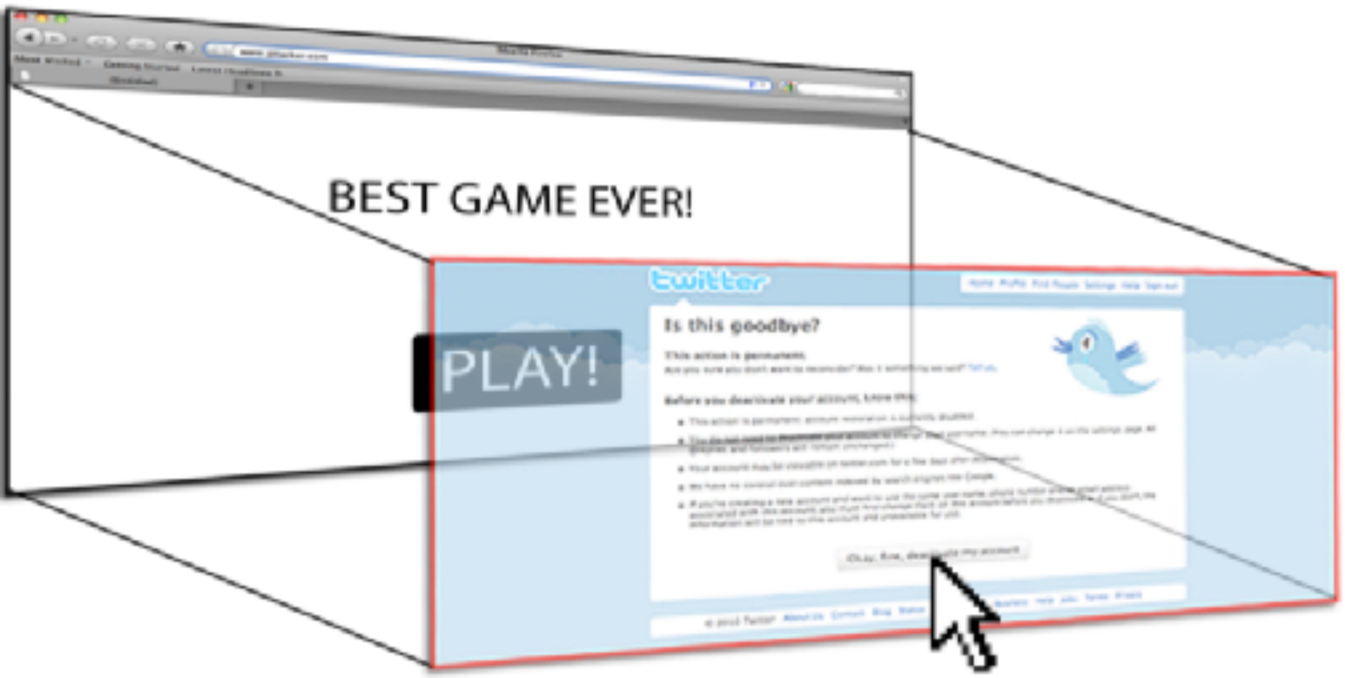

To: Bad guy

From: Victim

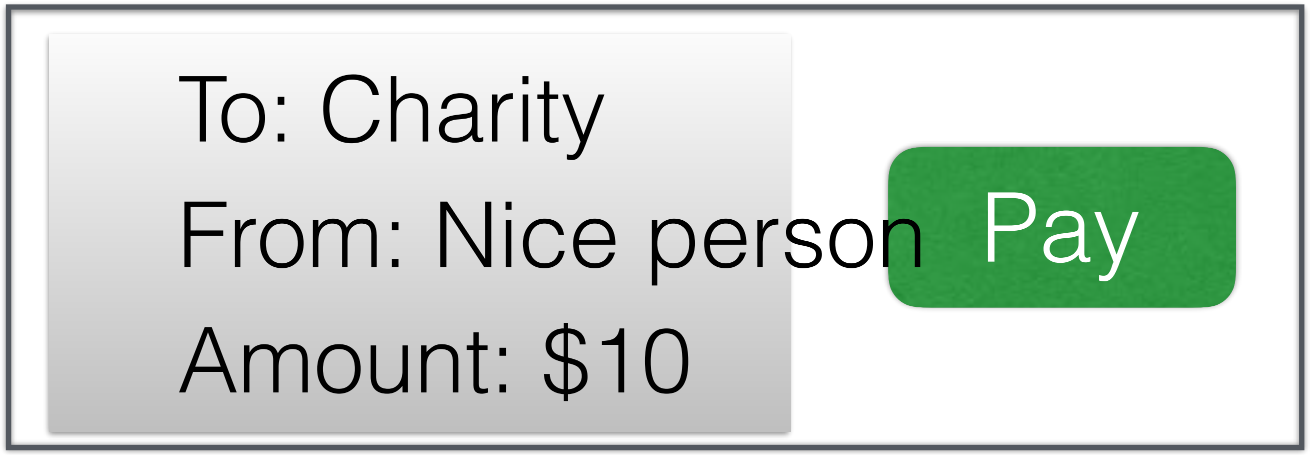
Amount: $1000

Compromising visual integrity of the *target*

- Hide the target element
  - CSS lets you set the opacity of an element to zero (clear)



- Partially overlay the target
  - Or *crop* the parts you don't wa



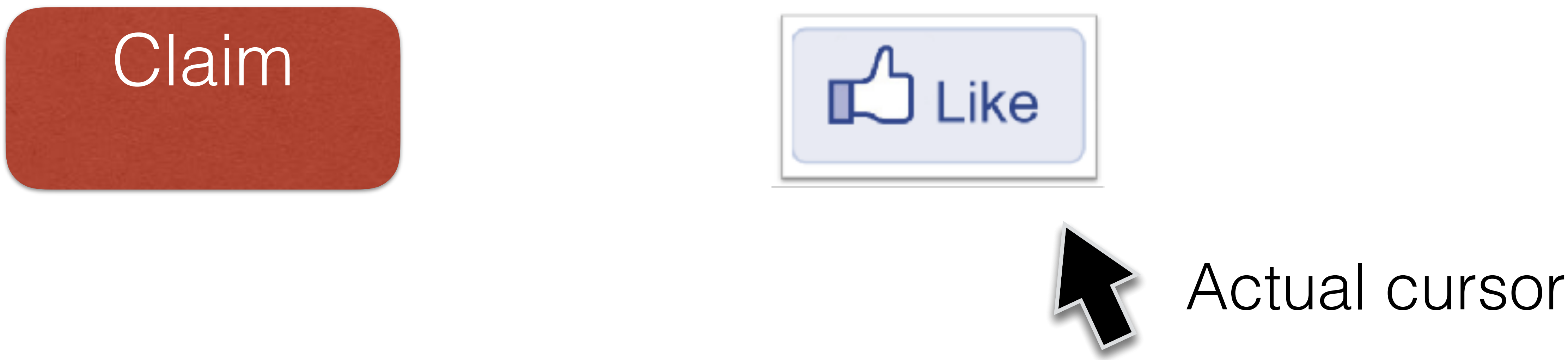
To: Charity

From: Nice person

Amount: $10

Pay

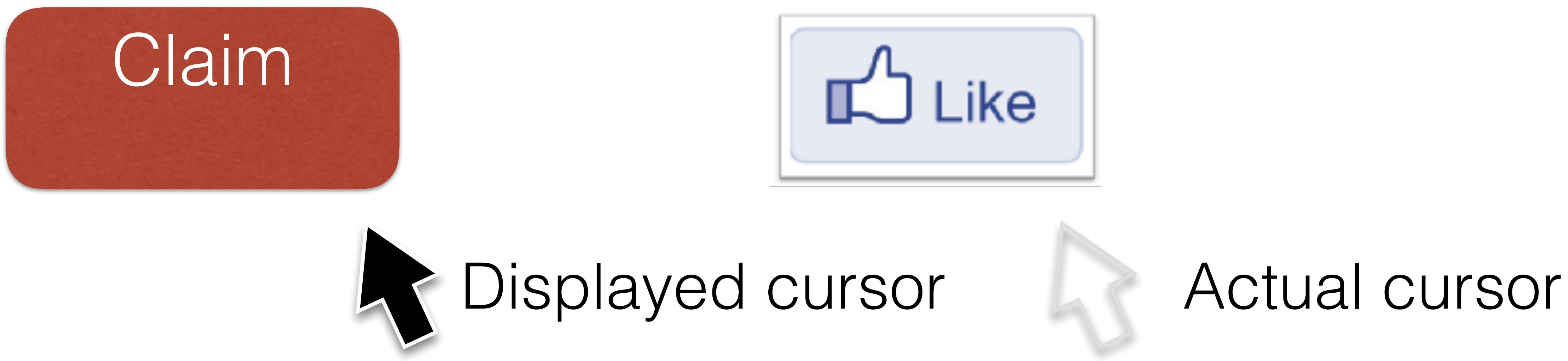
Compromising visual integrity of the *pointer*



Actual cursor

- Manipulating cursor feedback

Compromising visual integrity of the *pointer*

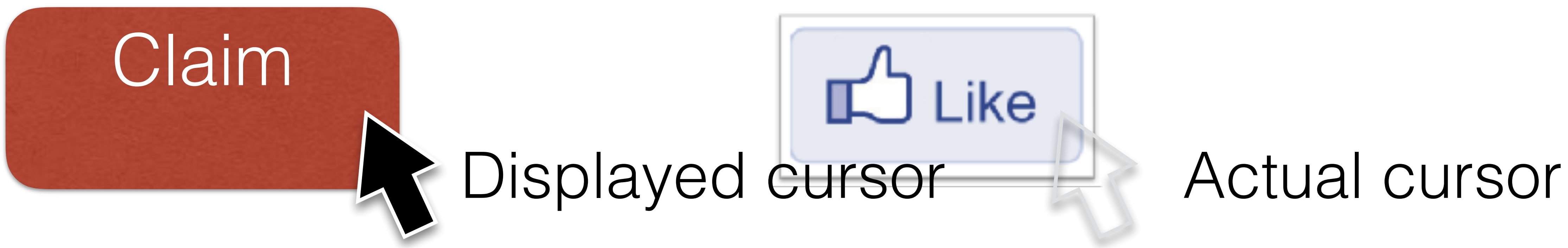


Displayed cursor

Actual cursor

- Manipulating cursor feedback

Compromising visual integrity of the *pointer*

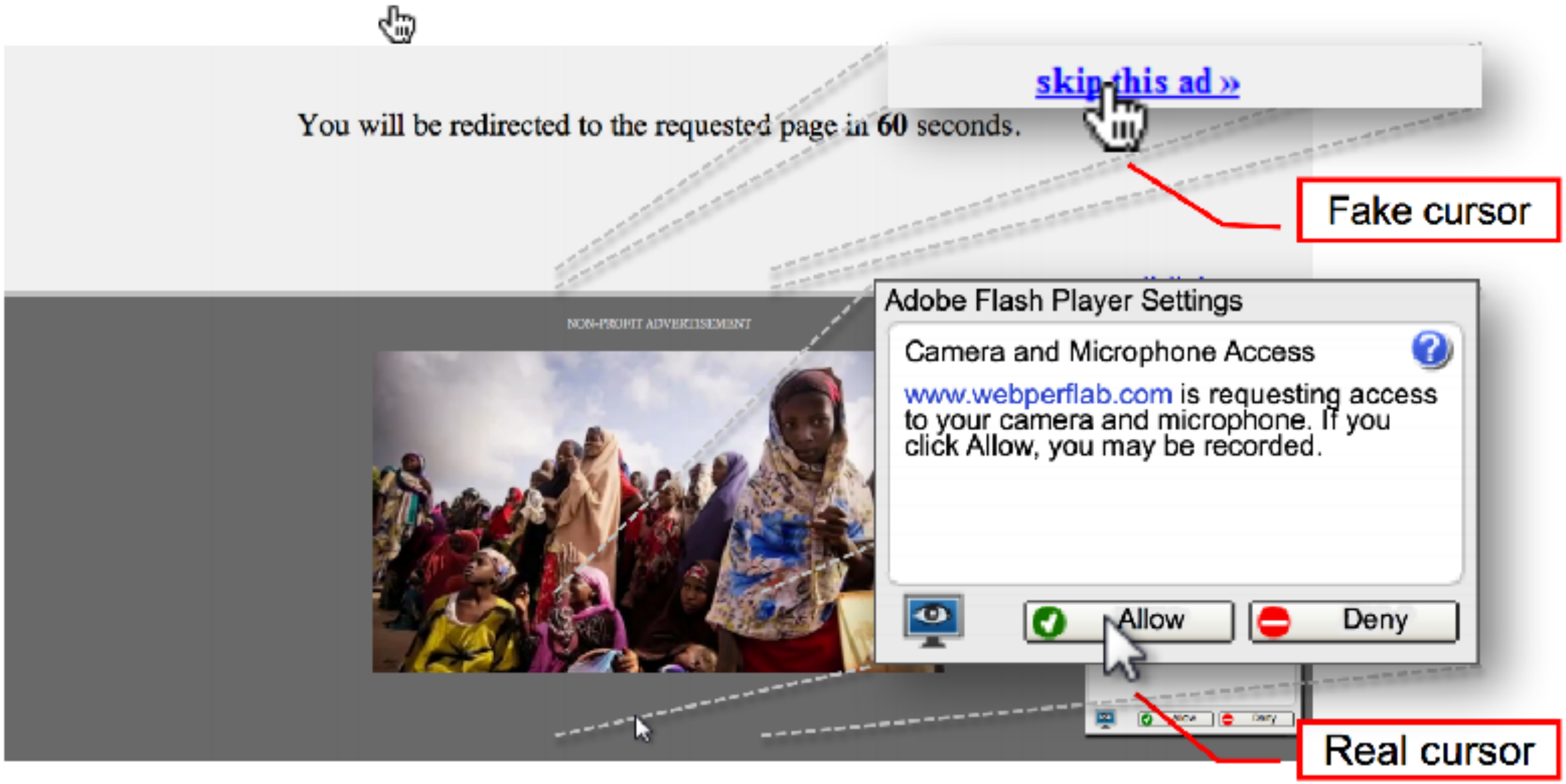


Displayed cursor

Actual cursor

- Manipulating cursor feedback

Clickjacking to access a user's webcam



You will be redirected to the requested page in **60** seconds.

Adobe Flash Player Settings

Camera and Microphone Access

www.webperflab.com is requesting access to your camera and microphone. If you click Allow, you may be recorded.

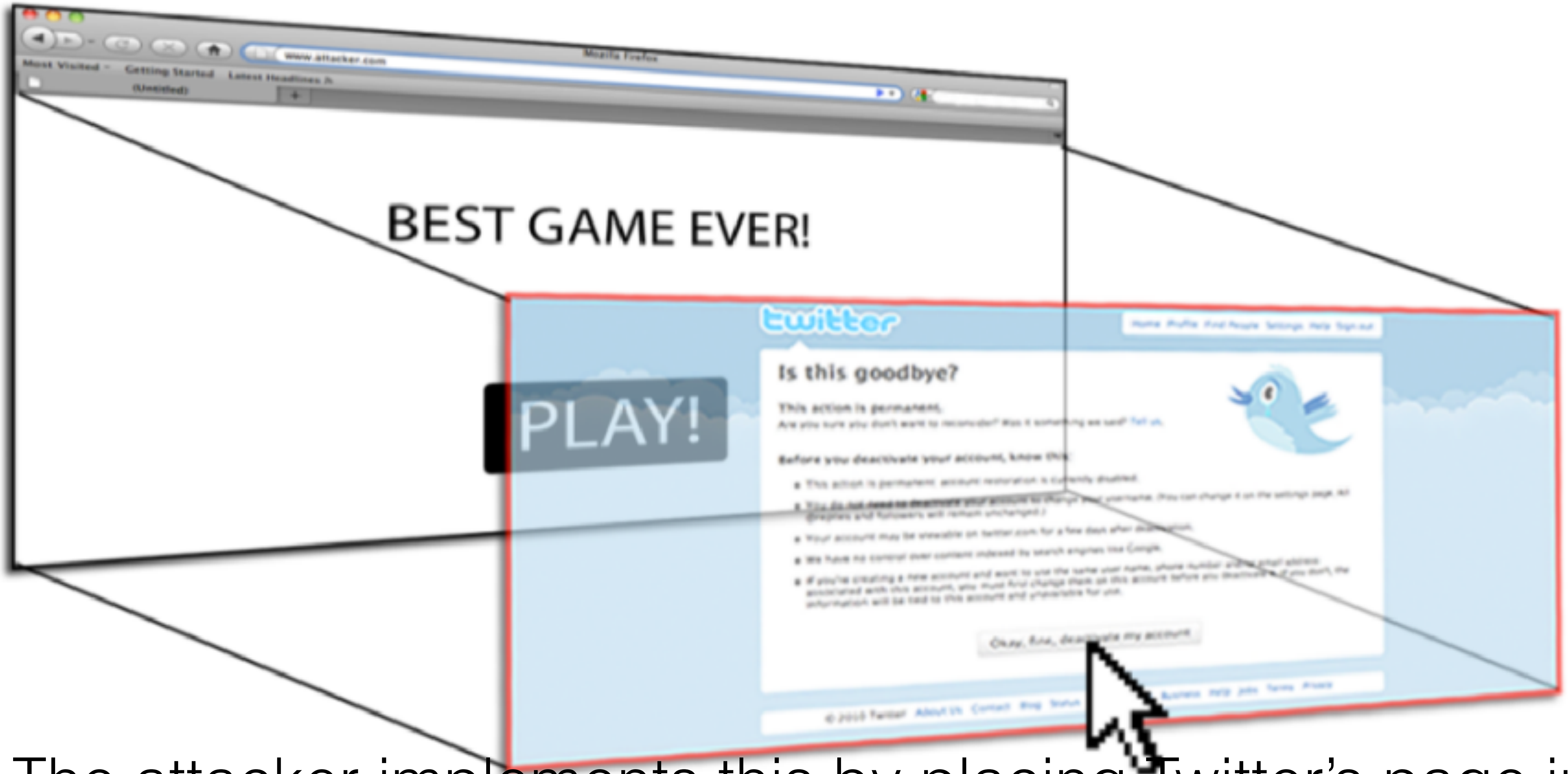
Fake cursor

Real cursor

Some clickjacking defenses

- Require confirmation for actions
  - Annoys users

- **Frame-busting**: Website ensures that its "vulnerable" pages can't be included as a *frame* inside another browser frame
  - So user can't be looking at it with something invisible overlaid on top…
  - …nor have the site invisible above something else



The attacker implements this by placing Twitter’s page in a “Frame”
inside their own page, otherwise they wouldn’t overlap

Some clickjacking defenses

- Require confirmation for actions
  - Annoys users

- **Frame-busting**: Website ensures that its “vulnerable” pages can’t be included as a *frame* inside another browser frame
  - So user can’t be looking at it with something invisible overlaid on top…
  - …nor have the site invisible above something else

- Conceptually implemented with Javascript like

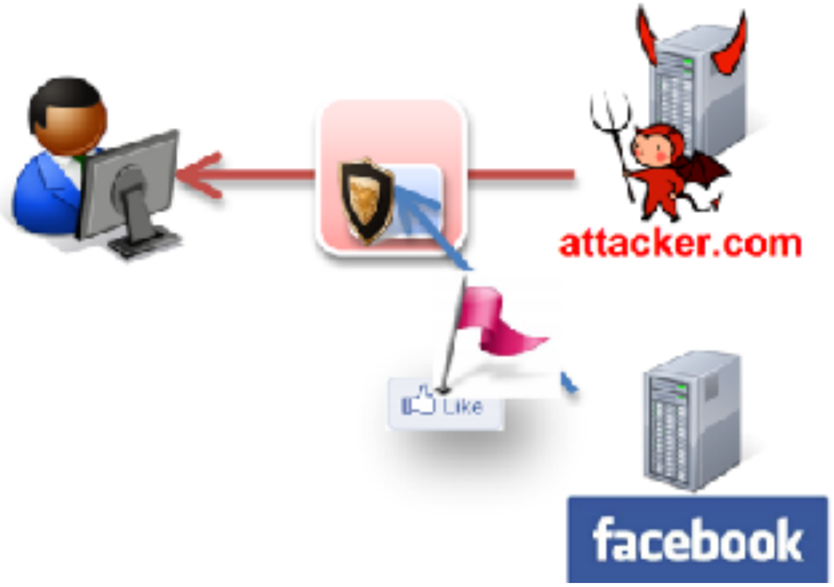
  ```
  if(top.location != self.location)
      top.location = self.location;
  ```

  (actually, it’s quite tricky to get this right)
- Current research considers more general approaches

InContext Defense (recent research)

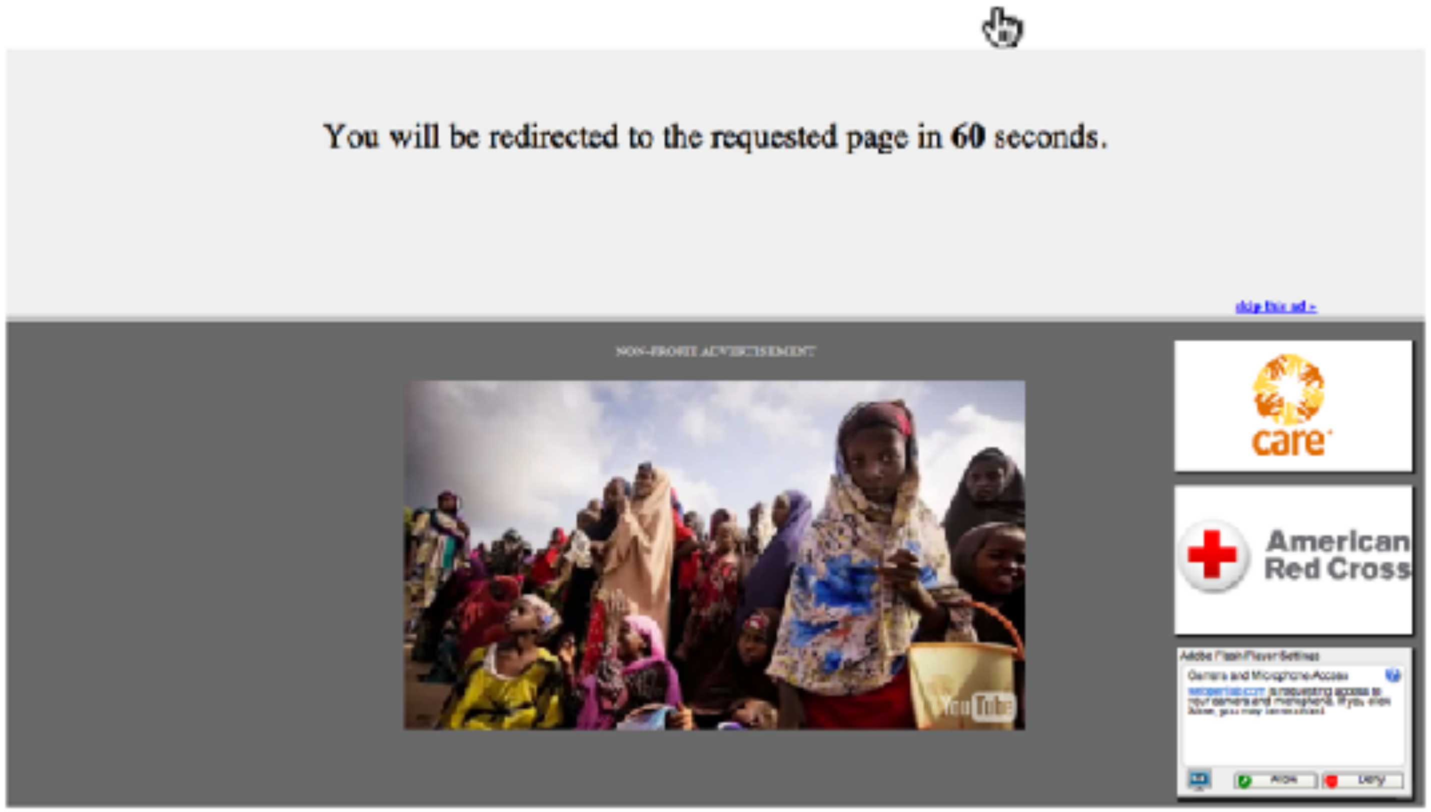
- A set of techniques to ensure context integrity for user actions

- Servers opt-in
  - Let the websites *indicate* their sensitive UIs
  - Let browsers *enforce* (
    when users act on the



Ensuring visual integrity of pointer

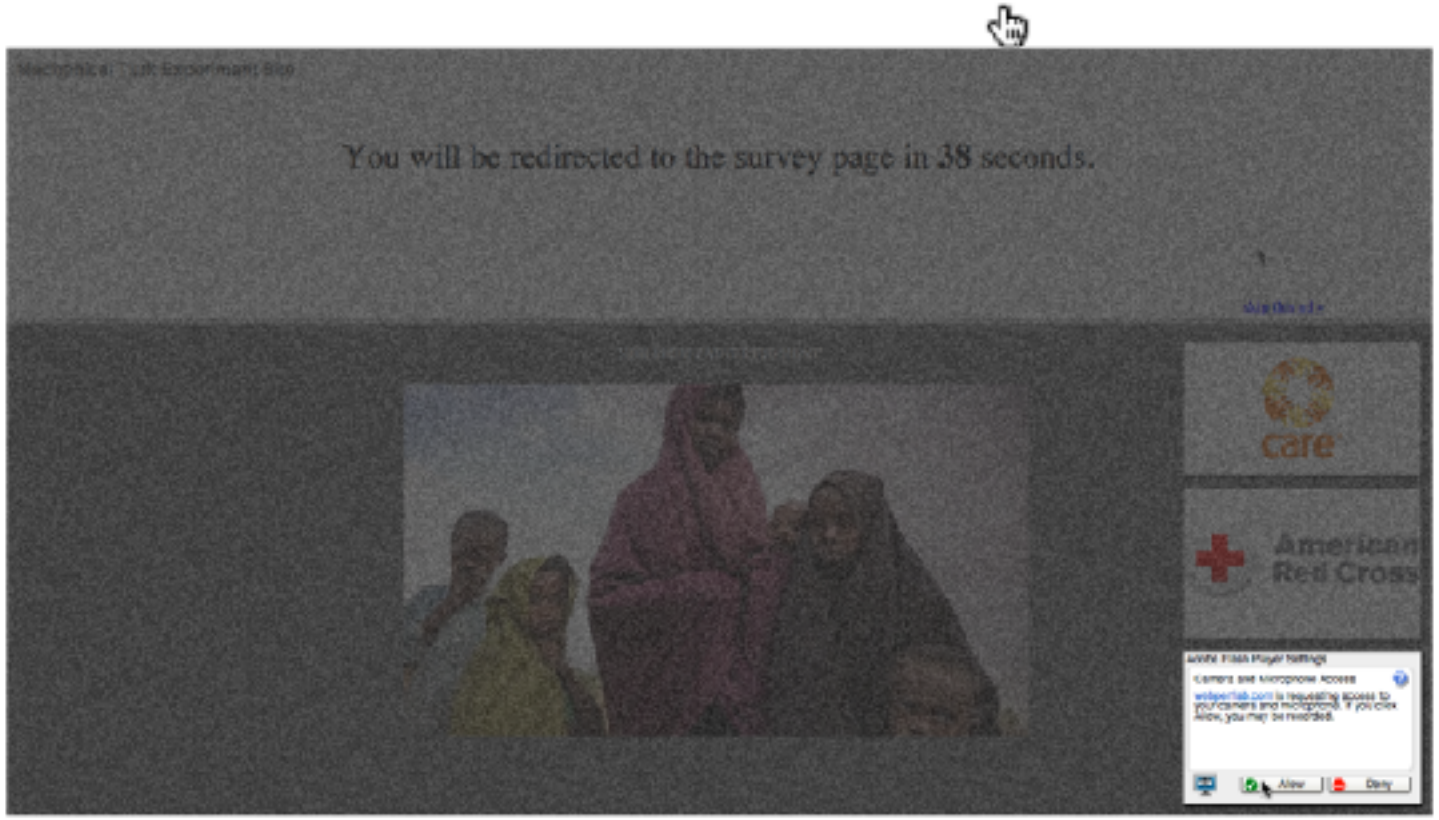
- Remove cursor customization
  - Attack success: 43% -> *16%*



You will be redirected to the requested page in **60** seconds.

[illegible] is requesting access to your camera and microphone. If you click Allow, you may be recorded.

Ensuring visual integrity of pointer

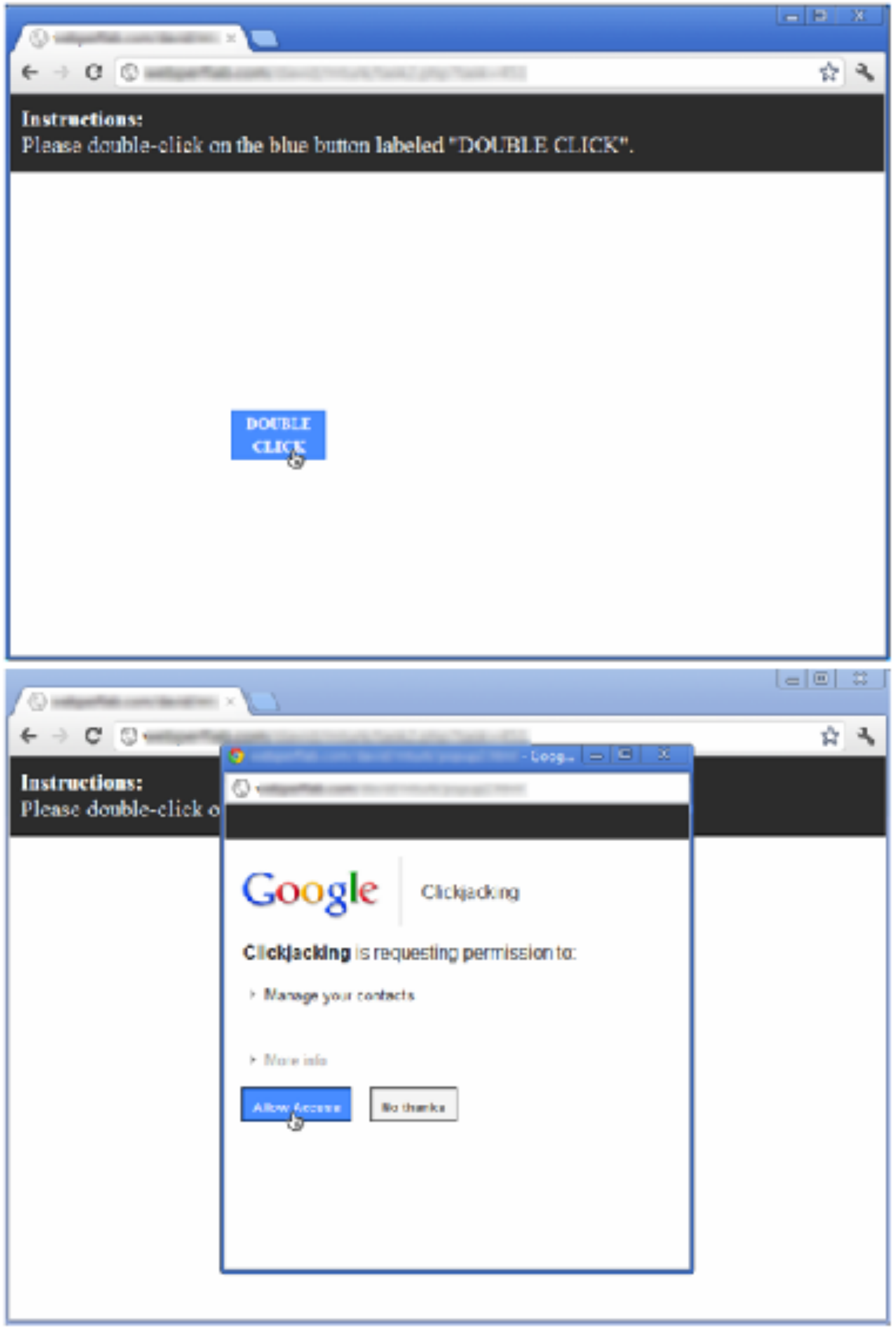
- Lightbox effect around target on pointer entry
  - 

You will be redirected to the survey page in 38 seconds.

Adobe Flash Player Settings

Camera and Microphone Access

webperflab.com is requesting access to your camera and microphone. If you click Allow, you may be recorded.


Instructions:
Please double-click on the blue button labeled "DOUBLE CLICK".

Instructions:
Please double-click o

Clickjacking is requesting permission to:

- Manage your contacts

Enforcing temporal integrity

- UI delay: after visual changes on target or pointer, invalidate clicks for a few milliseconds
- Pointer re-entry: after visual changes on target, invalidate clicks until pointer re-enters target

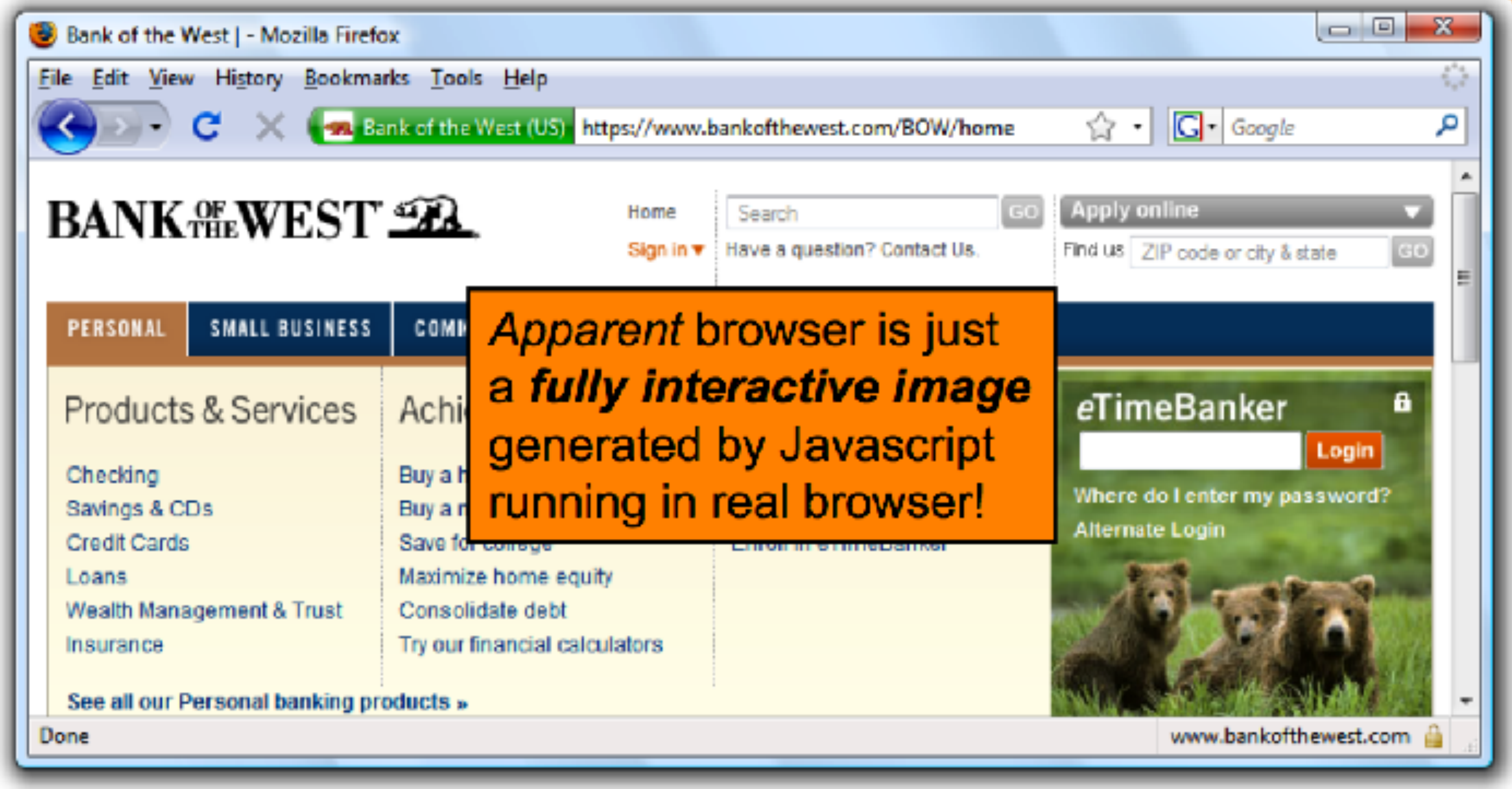
Other forms of UI sneakiness

- Along with stealing events, attackers can use the power of Javascript customization and dynamic changes to mess with the user’s mind

- For example, the user may not be paying attention, so you can swap tabs on them

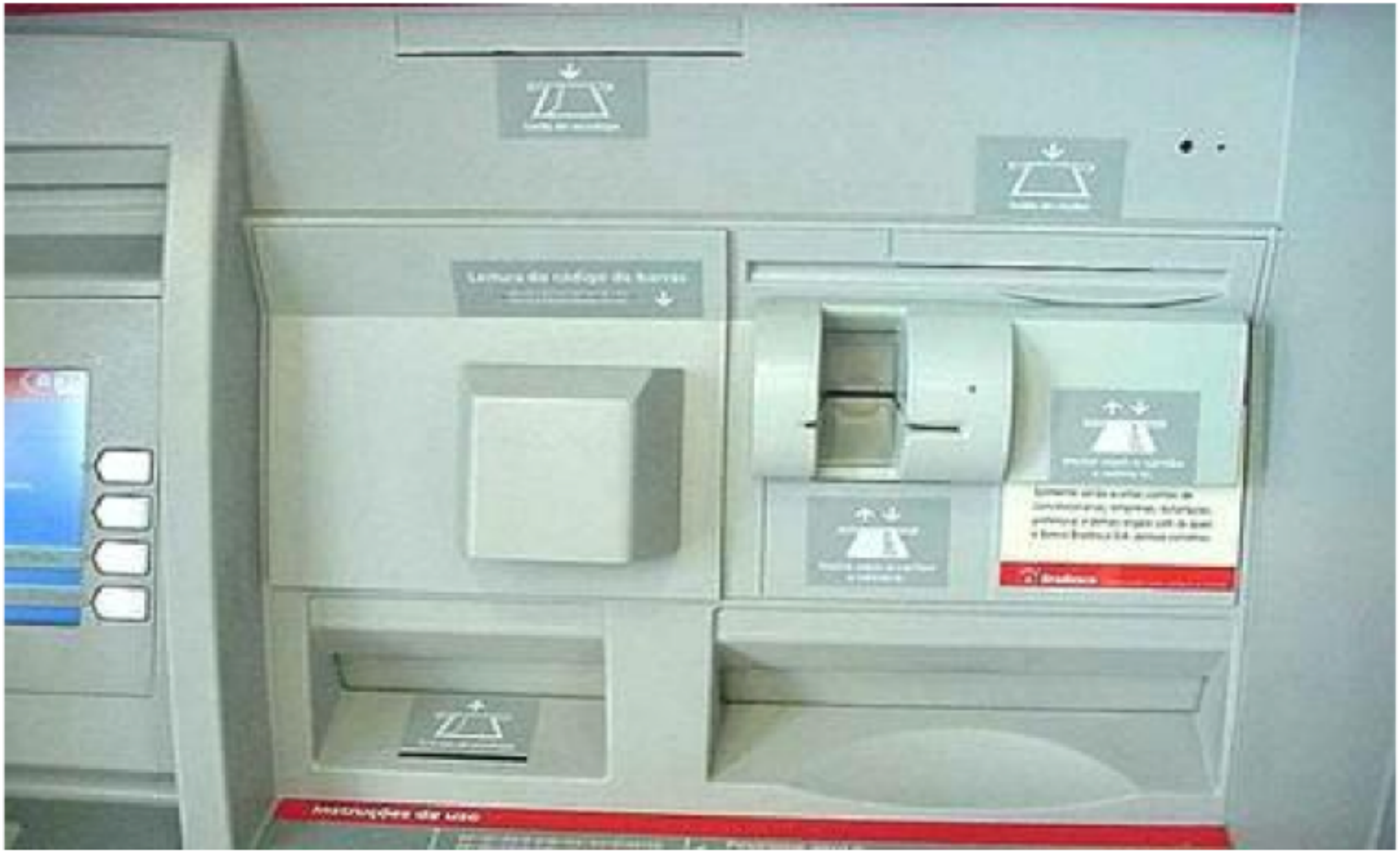
- Or they may find themselves “eclipsed”

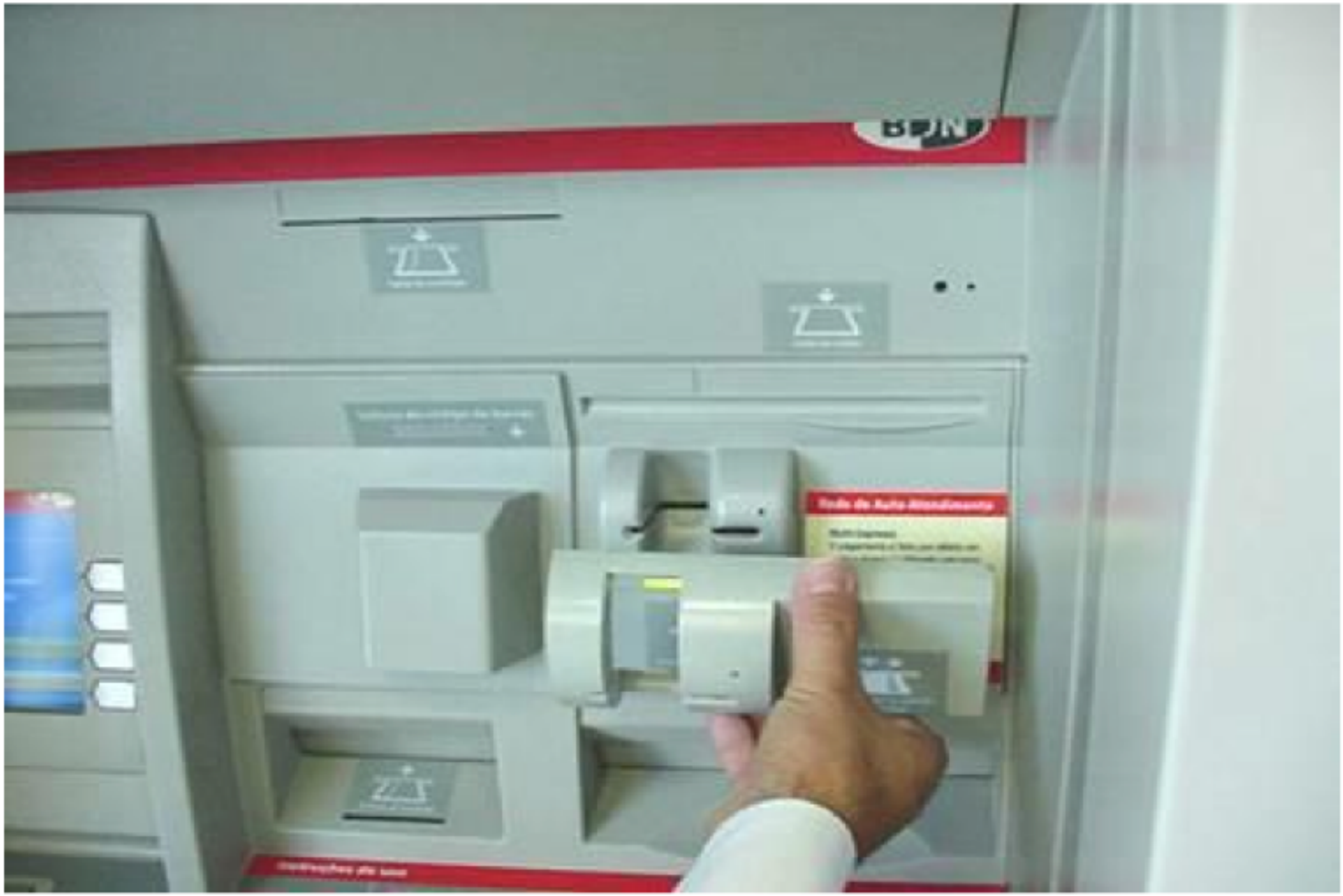
Browser in browser



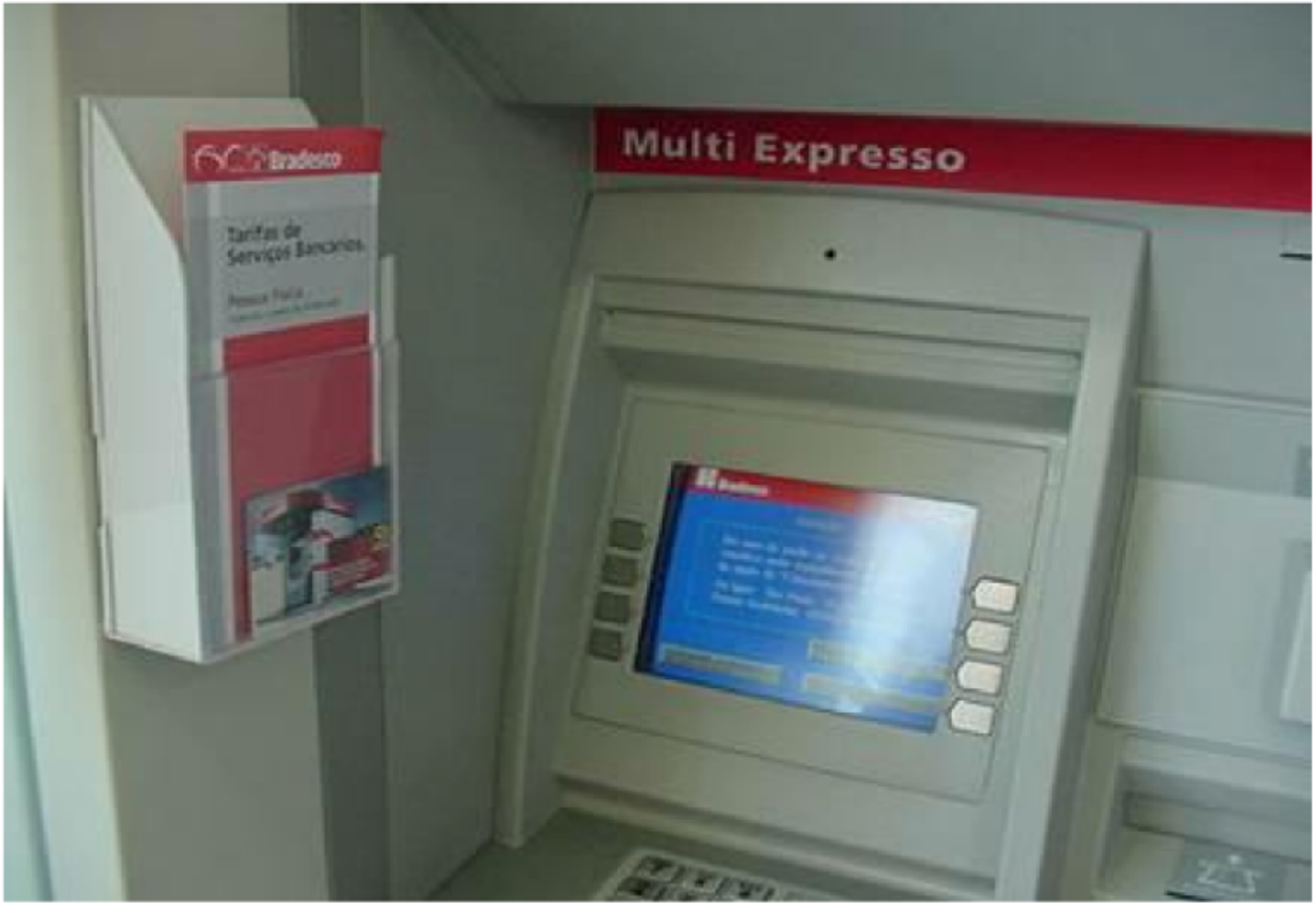
Apparent browser is just a ***fully interactive image*** generated by Javascript running in real browser!

WHAT IS UNTRUSTWORTHY HERE?



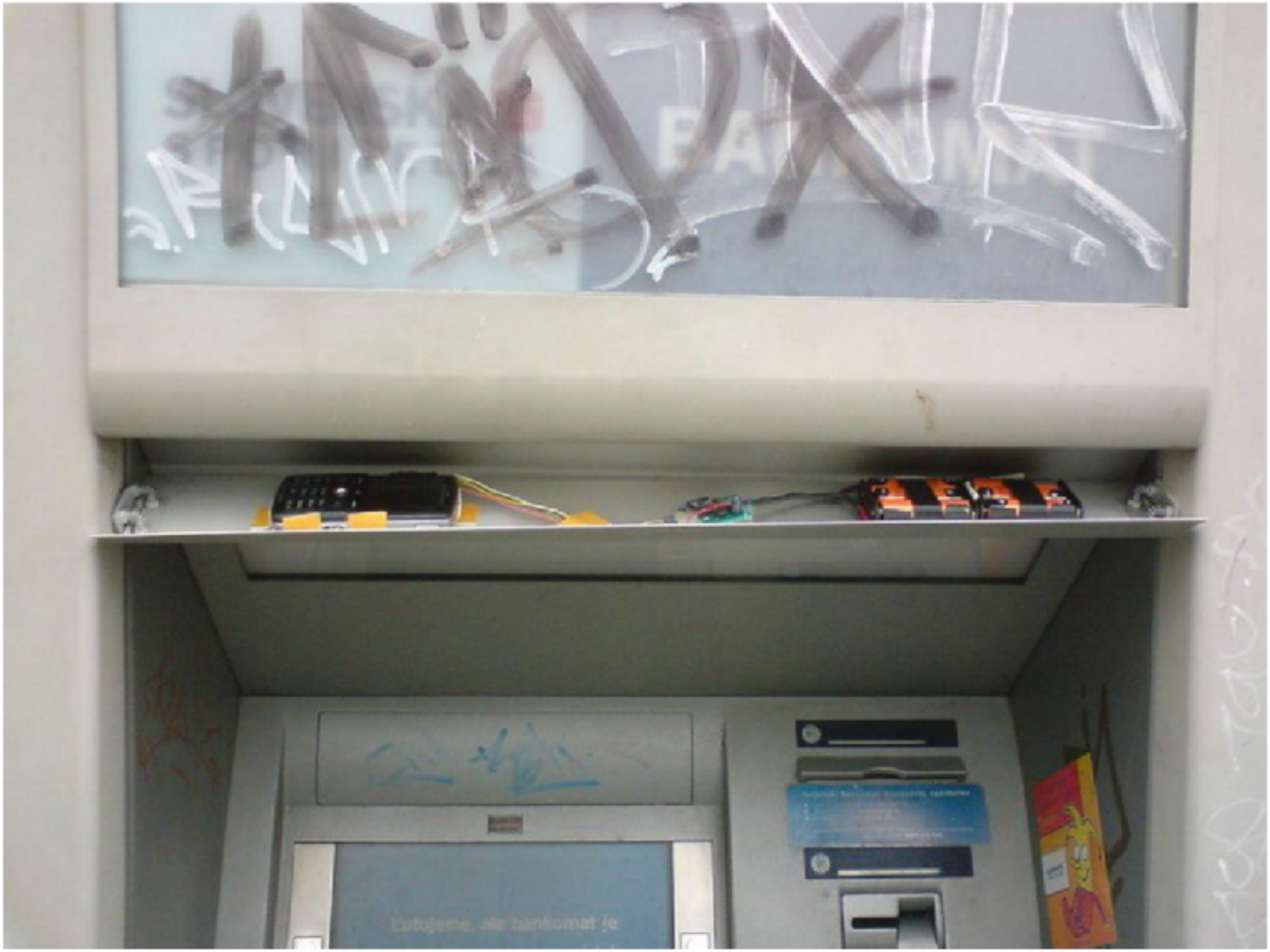


WHAT IS UNTRUSTWORTHY HERE?



MU
Bradesco
Tarifas de
Serviços Bancários
Pessoa Física
Mul

widelec.org


BANKOMAT
Ľutujeme, ale bankomat je

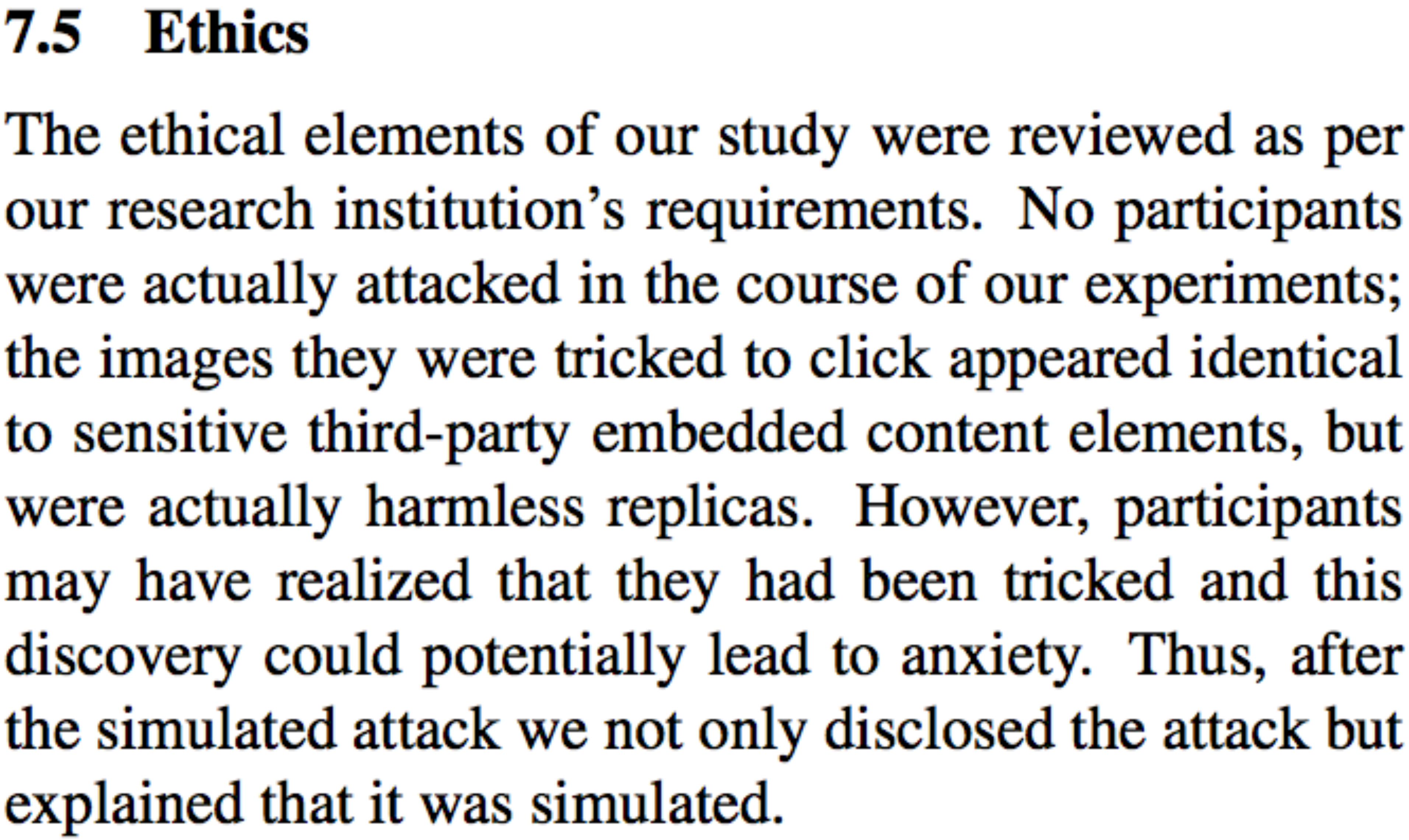
CLICKJACKING: EXPERIMENTS

- Mechanical Turks
  - $0.25 per participant to “follow the on-screen instructions and complete an interactive task.”
  - Simulated attacks, simulated defenses
  - 3251 participants
  - Note: You must control for sloppy participation
    - Excluded 370 repeat-participants

CLICKJACKING: EXPERIMENTS

- Control group 1
    - "Skip ad" button
    - No attack to trick the user
    - Purpose: To determine the click rate we would hope a defense could achieve in countering an attack
    - 38% didn't skip the ad
- Control group 2
    - "Allow" button to skip ad
    - Purpose: An attempt to persuade users to grant access without tricking them
    - 8% allowed (statistically indistinguishable from group 1)

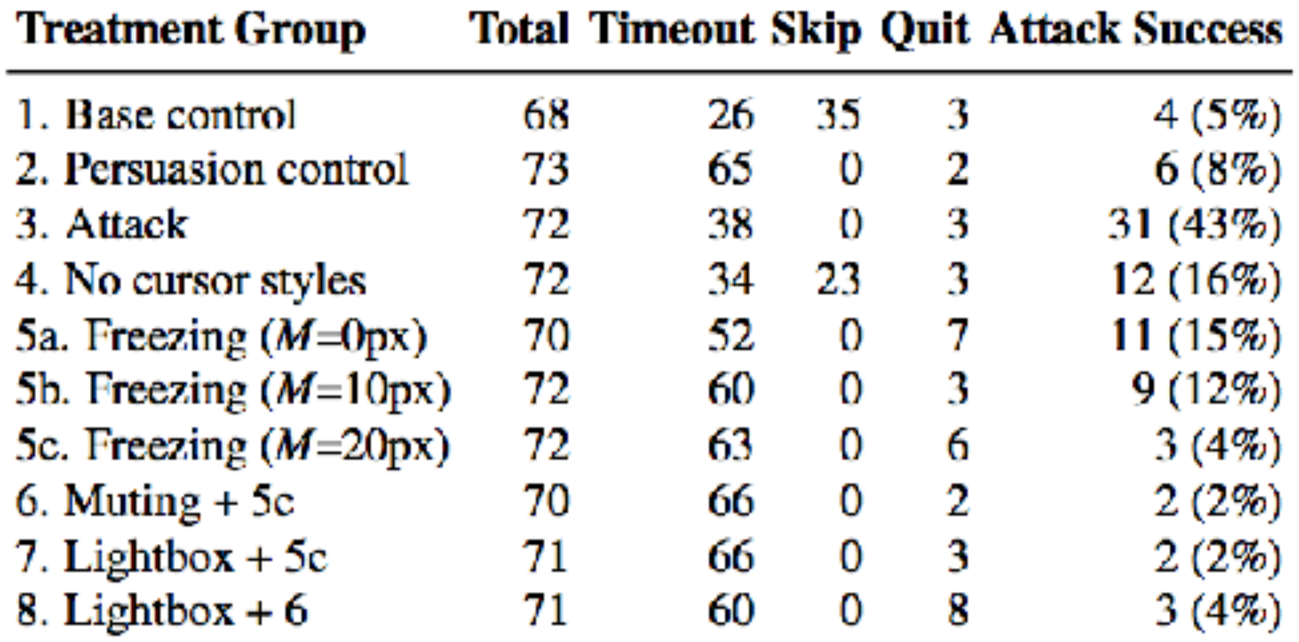
CLICKJACKING: EXPERIMENTS



7.5 Ethics

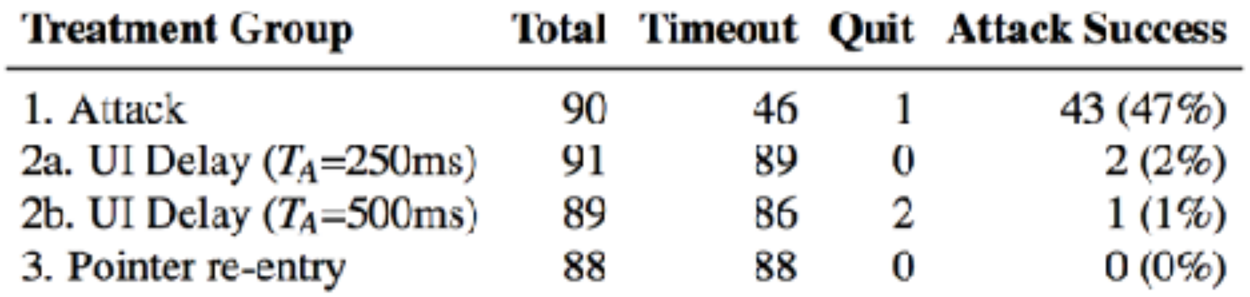
The ethical elements of our study were reviewed as per our research institution's requirements. No participants were actually attacked in the course of our experiments; the images they were tricked to click appeared identical to sensitive third-party embedded content elements, but were actually harmless replicas. However, participants may have realized that they had been tricked and this discovery could potentially lead to anxiety. Thus, after the simulated attack we not only disclosed the attack but explained that it was simulated.

CLICKJACKING: EXPERIMENTS



Treatment Group	Total	Timeout	Skip	Quit	Attack Success
1. Base control	68	26	35	3	4 (5%)
2. Persuasion control	73	65	0	2	6 (8%)
3. Attack	72	38	0	3	31 (43%)
4. No cursor styles	72	34	23	3	12 (16%)
5a. Freezing (M=0px)	70	52	0	7	11 (15%)
5b. Freezing (M=10px)	72	60	0	3	9 (12%)
5c. Freezing (M=20px)	72	63	0	6	3 (4%)
6. Muting + 5c	70	66	0	2	2 (2%)
7. Lightbox + 5c	71	66	0	3	2 (2%)
8. Lightbox + 6	71	60	0	8	3 (4%)

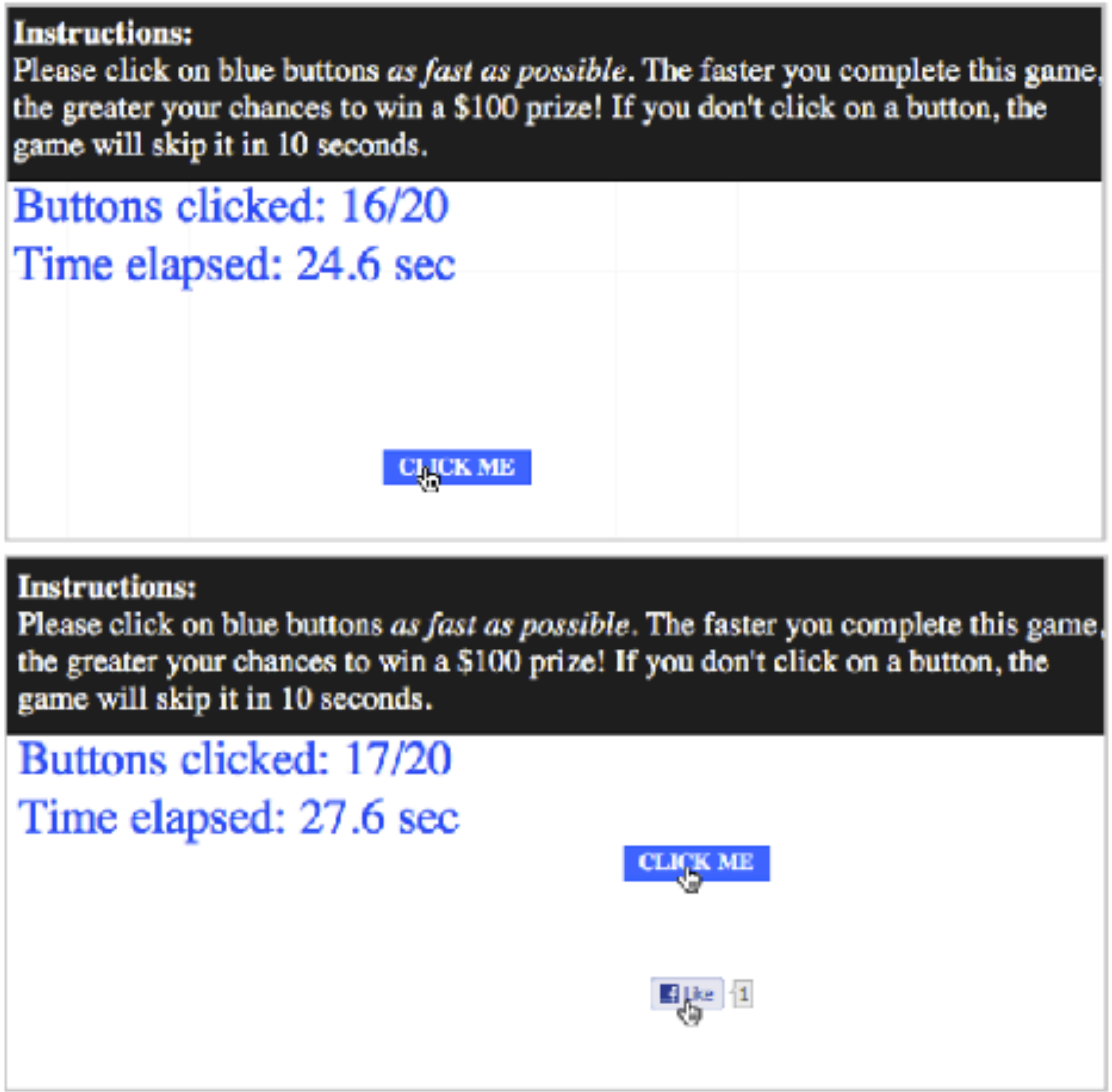
Table 2: **Results of the cursor-spoofing attack.** *Our attack tricked 43% of participants to click on a button that would grant webcam access. Several of our proposed defenses reduced the rate of clicking to the level expected if no attack had occurred.*



Treatment Group	Total	Timeout	Quit	Attack Success
1. Attack	90	46	1	43 (47%)
2a. UI Delay (T_A=250ms)	91	89	0	2 (2%)
2b. UI Delay (T_A=500ms)	89	86	2	1 (1%)
3. Pointer re-entry	88	88	0	0 (0%)

Table 3: **Results of double-click attack.** 43 *of* 90 *participants fell for the attack that would grant access to their personal Google data. Two of our defenses stopped the attack completely.*

CLICKJACKING: EXPERIMENTS



Instructions:
Please click on blue buttons *as fast as possible*. The faster you complete this game, the greater your chances to win a $100 prize! If you don't click on a button, the game will skip it in 10 seconds.

Buttons clicked: 16/20

Time elapsed: 24.6 sec

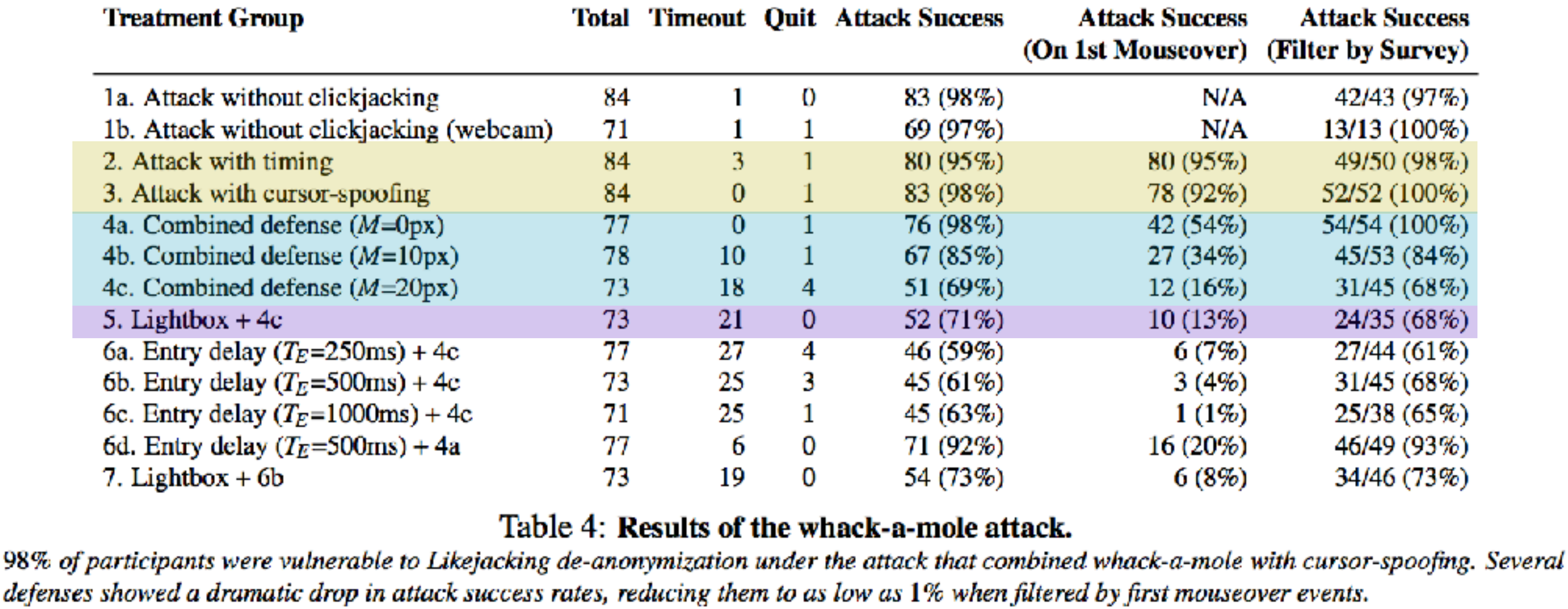
Instructions:
Please click on blue buttons *as fast as possible*. The faster you complete this game, the greater your chances to win a $100 prize! If you don't click on a button, the game will skip it in 10 seconds.

Buttons clicked: 17/20

Time elapsed: 27.6 sec

Figure 3: **Whack-a-mole attack page.** *This is a cursor spoofing variant of the whack-a-mole attack. On the 18th trial, the attacker displays the target Like button underneath the actual pointer.*

CLICKJACKING: EXPERIMENTS



Treatment Group	Total	Timeout	Quit	Attack Success	Attack Success (On 1st Mouseover)	Attack Success (Filter by Survey)
1a. Attack without clickjacking	84	1	0	83 (98%)	N/A	42/43 (97%)
1b. Attack without clickjacking (webcam)	71	1	1	69 (97%)	N/A	13/13 (100%)
2. Attack with timing	84	3	1	80 (95%)	80 (95%)	49/50 (98%)
3. Attack with cursor-spoofing	84	0	1	83 (98%)	78 (92%)	52/52 (100%)
4a. Combined defense (M=0px)	77	0	1	76 (98%)	42 (54%)	54/54 (100%)
4b. Combined defense (M=10px)	78	10	1	67 (85%)	27 (34%)	45/53 (84%)
4c. Combined defense (M=20px)	73	18	4	51 (69%)	12 (16%)	31/45 (68%)
5. Lightbox + 4c	73	21	0	52 (71%)	10 (13%)	24/35 (68%)
6a. Entry delay (T_E=250ms) + 4c	77	27	4	46 (59%)	6 (7%)	27/44 (61%)
6b. Entry delay (T_E=500ms) + 4c	73	25	3	45 (61%)	3 (4%)	31/45 (68%)
6c. Entry delay (T_E=1000ms) + 4c	71	25	1	45 (63%)	1 (1%)	25/38 (65%)
6d. Entry delay (T_E=500ms) + 4a	77	6	0	71 (92%)	16 (20%)	46/49 (93%)
7. Lightbox + 6b	73	19	0	54 (73%)	6 (8%)	34/46 (73%)

Table 4: **Results of the whack-a-mole attack.**

98% of participants were vulnerable to Likejacking de-anonymization under the attack that combined whack-a-mole with cursor-spoofing. Several defenses showed a dramatic drop in attack success rates, reducing them to as low as 1% when filtered by first mouseover events.