MODERN
WEB SECURITY

GRAD SEC

SEP 21 2017

TODAY'S PAPERS

Clickjacking: Attacks and Defenses

Lin-Shung Huang
Carnegie Mellon University
linshung.huang@sv.cmu.edu

Alex Moshchuk
Microsoft Research
alexmos@microsoft.com

Helen J. Wang
Microsoft Research
helenw@microsoft.com

Stuart Schechter
Microsoft Research
stuart.schechter@microsoft.com

Collin Jackson
Carnegie Mellon University
collin.jackson@sv.cmu.edu

Abstract

Clickjacking attacks are an emerging threat on the web. In this paper, we design new clickjacking attack variants using existing techniques and demonstrate that existing clickjacking defenses are insufficient. Our attacks show that clickjacking can cause severe damages, including compromising a user's private webcam, email or other private data, and web surfing anonymity.

We observe the root cause of clickjacking is that an attacker application presents a sensitive UI element of a target application *out of context* to a user (such as hiding the sensitive UI by making it transparent), and hence the user is tricked to act out of context. To address this root cause, we propose a new defense, *InContext*, in which web sites (or applications) mark UI elements that are sensitive, and browsers (or OSes) *enforce context integrity* of user actions on these sensitive UI elements, ensuring that a user sees everything she should see before her action and that the timing of the action corresponds to her intent.

We have conducted user studies on Amazon Mechanical Turk with 2064 participants to evaluate the effectiveness of our attacks and our defense. We show that our attacks have success rates ranging from 43% to 98%, and our InContext defense can be very effective against the clickjacking attacks in which the use of clickjacking is more effective than social engineering.

1 Introduction

When multiple applications or web sites (or OS principals [44] in general) share a graphical display, they are subject to *clickjacking* [13] (also known as *UI redressing* [28, 49]) attacks: one principal may trick the user into interacting with (e.g., clicking, touching, or voice controlling) UI elements of another principal, triggering actions not intended by the user. For example, in *Likejacking* attacks [46], an attacker web page tricks users into clicking on a Facebook "Like" button by transparently overlaying it on top of an innocuous UI element, such as a "claim your free iPad!" button. Hence, when the user "claims" a free iPad, a story appears in the user's Facebook friends' news feed stating that she "likes" the attacker web site. For ease of exposition, our description will be in the context of web browsers. Nevertheless, the concepts and techniques described are generally applicable to all client operating systems where display is shared by mutually distrusting principals.

Several clickjacking defenses have been proposed and deployed for web browsers, but all have shortcomings. Today's most widely deployed defenses rely on *framebusting* [21, 37], which disallows a sensitive page from being framed (i.e., embedded within another web page). Unfortunately, framebusting is fundamentally incompatible with embeddable third-party widgets, such as Facebook Like buttons. Other existing defenses (discussed in Section 3.2) suffer from poor usability, incompatibility with existing web sites, or failure to defend against significant attack vectors.

To demonstrate the insufficiency of state-of-the-art defenses, we construct three new attack variants using existing clickjacking techniques. We designed the new attack scenarios to be more damaging than the existing clickjacking attacks in the face of current defenses. In one scenario, the often-assumed web-surfing-anonymity can be compromised. In another, a user's private data and emails can be stolen. Lastly, an attacker can spy on a user through her webcam. We have conducted the first clickjacking effectiveness study through Amazon Mechanical Turk and find that the aforementioned attacks have success rates of 98%, 47%, and 13%, respectively.

Learning from the lessons of existing defenses, we set the following design goals for our clickjacking defense:

- Widget compatibility: clickjacking protection should support third-party widgets.
- Usability: users should not be prompted for their actions.
- Backward compatibility: the defense should not break existing web sites (e.g., by disallowing exist-

COVER FEATURE **CYBERSECURITY**

Protecting Websites from Attack with Secure Delivery Networks

David Gillman, New College of Florida and Akamai Technologies
Yin Lin, VMware
Bruce Maggs, Duke University and Akamai Technologies
Ramesh K. Sitaraman, University of Massachusetts Amherst and Akamai Technologies

Secure delivery networks can help prevent or mitigate the most common attacks against mission-critical websites. A case study from a leading provider of content delivery services illustrates one such network's operation and effectiveness.

The Web has become an indispensable medium for conducting business, performing financial transactions, accessing news and entertainment, playing games, and interacting with government and other types of services. We have come to expect the websites that facilitate these activities to be available at all times and to perform well.

However, threats against such sites have never been greater. As Akamai Technologies, a leading provider of content delivery services, reports in its *State of the Internet* for the third quarter of 2014,[1] distributed denial-of-service (DDoS) attacks against its customers are increasing in terms of both the bandwidth and the number of requests generated by the attackers. From 2009 to 2014, the size of the largest attack, in gigabits per second, grew year by year from 48 to 68 to 79 to 82 to 190 to 321. At the same time, the number of packets per second in the largest attack grew from 29 million to 109 million. The number of DDoS attacks detected and mitigated has also more than doubled over the past two years, reaching 5,434 in 2014.

Beyond DDoS, online theft of data such as credit card numbers, personally identifiable information, business secrets, and login credentials has also grown rapidly. In fact, Web application attacks are the most common cause of data breaches today.[2] Not surprisingly, the financial cost of DDoS attacks and other forms of cybercrime is increasing rapidly. In a survey of 277 companies in 16 industry sectors,[3] the Ponemon Institute found that the average financial cost of cybercrime to survey participants was US$11.6 million in 2013. That cost is expected to grow rapidly in succeeding years as attacks increase in size, frequency, and sophistication.

Attacks impact every segment of the Internet ecosystem. In Q3 2014, those against Akamai customers spanned every segment of online services. The

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*

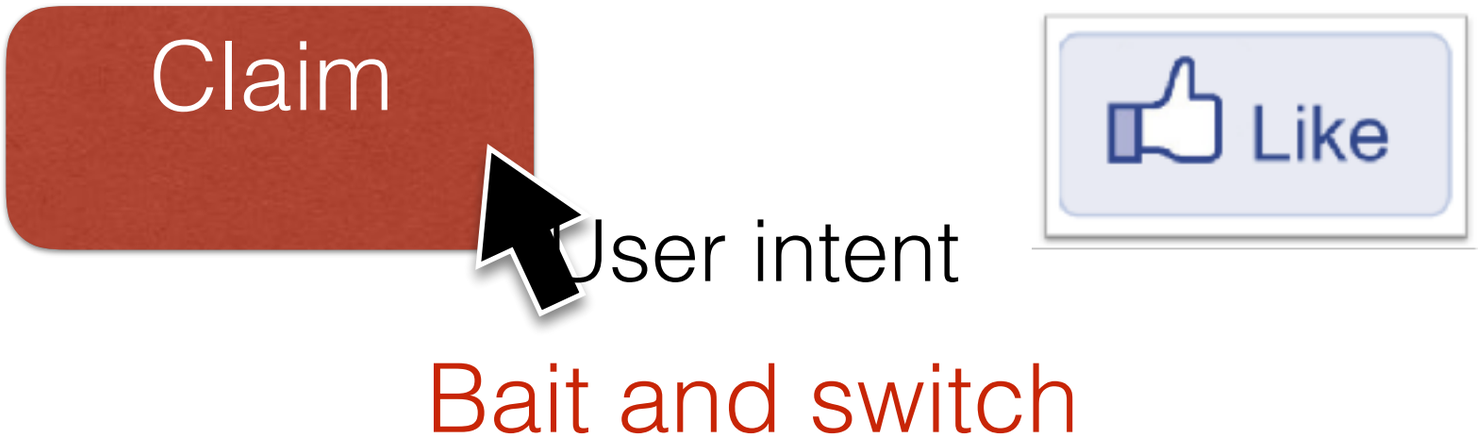
Claim

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*

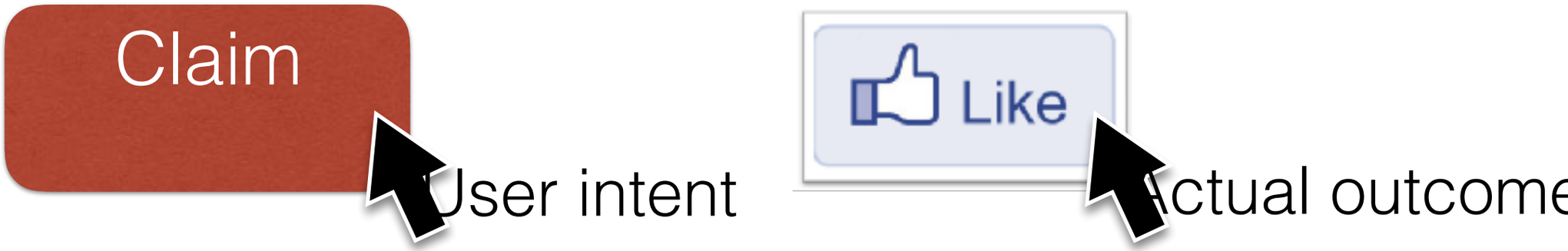


Bait and switch

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*



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)

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

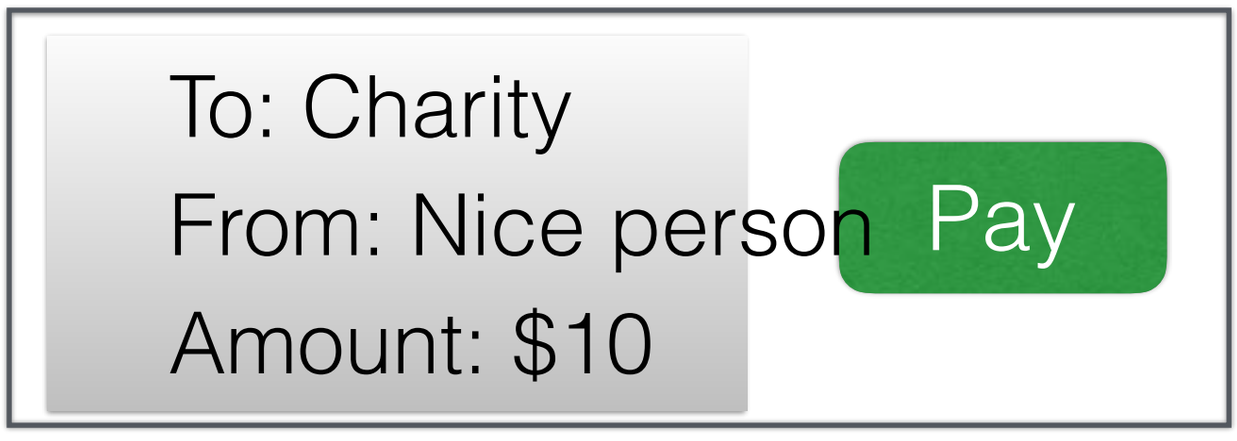
To: Bad guy
From: Victim
Amount: $1000

Pay

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



Compromising visual integrity of the *pointer*

Claim

Actual cursor

- Manipulating cursor feedback

Compromising visual integrity of the *pointer*

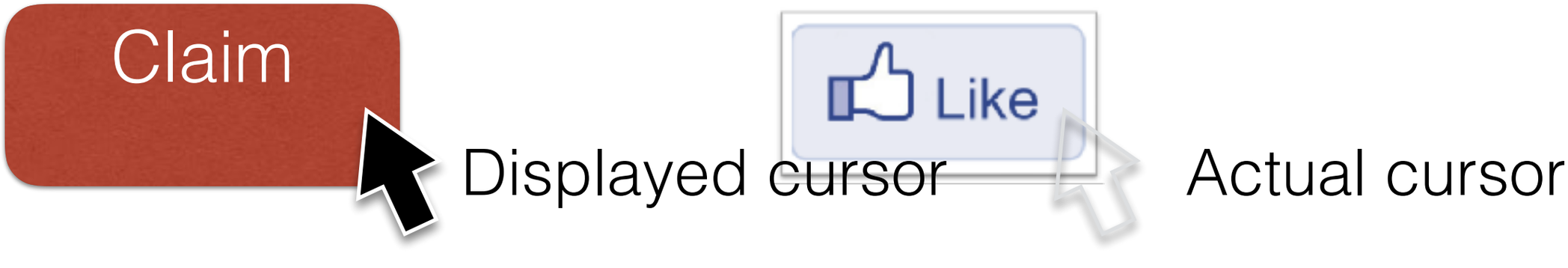
Claim

Displayed cursor

Actual cursor

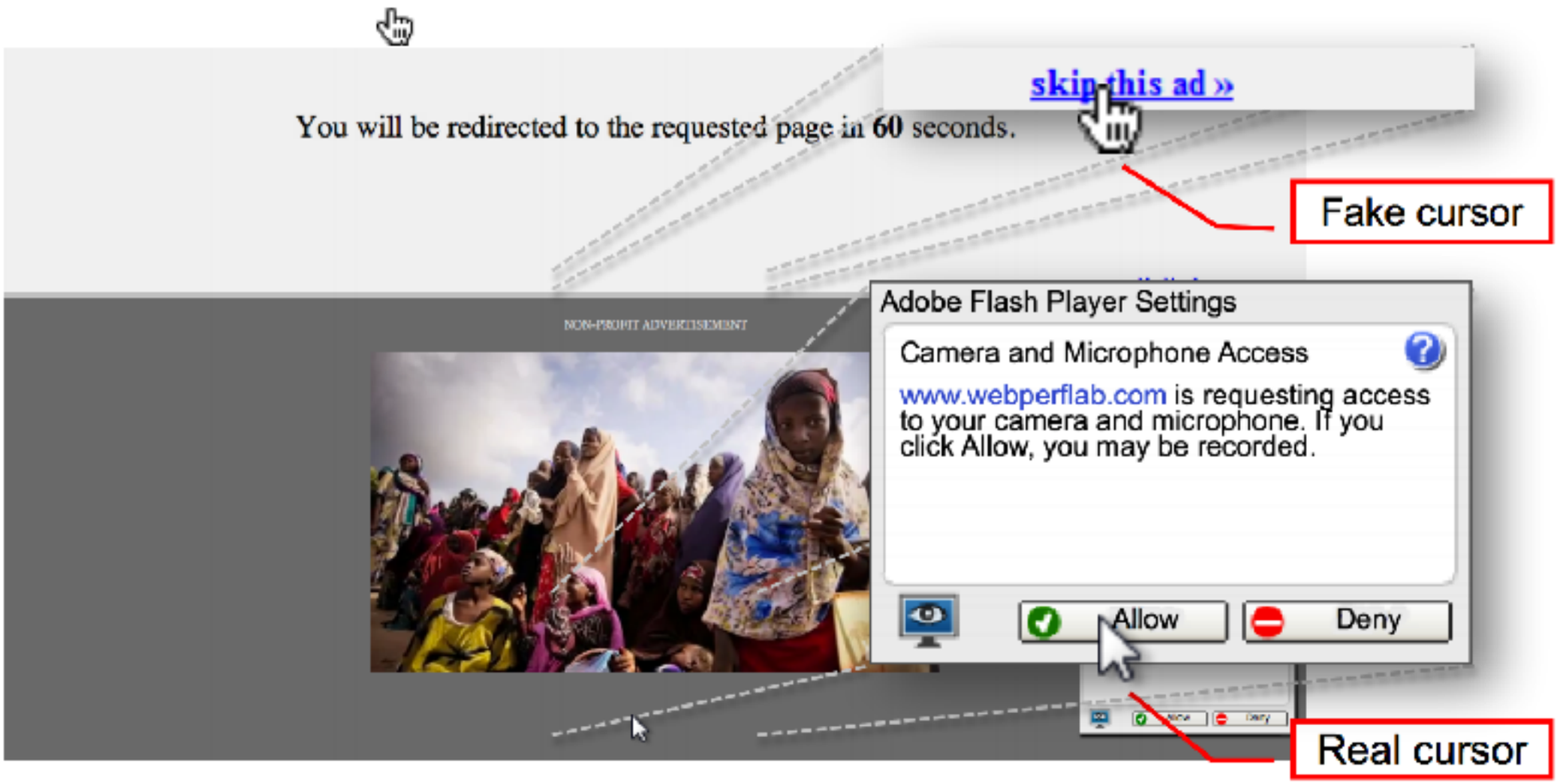
- Manipulating cursor feedback

Compromising visual integrity of the *pointer*



- Manipulating cursor feedback

Clickjacking to access a user's webcam



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

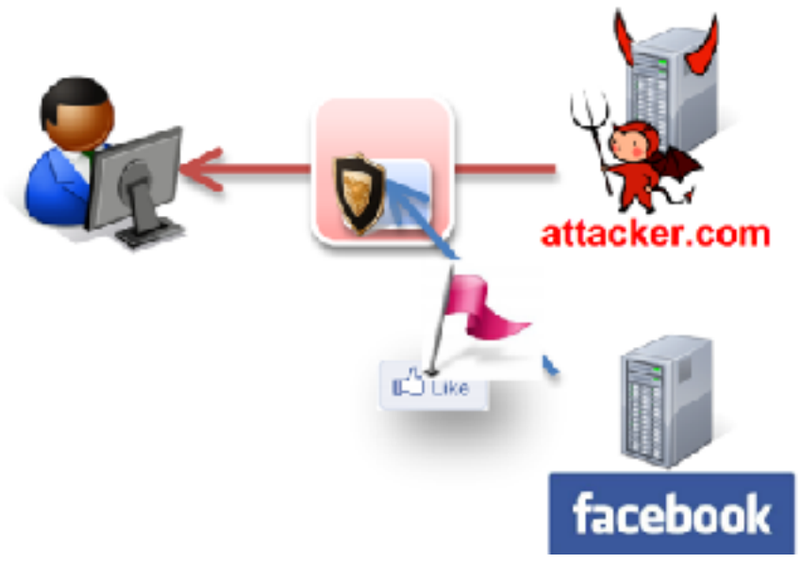
- Conceptually implemented with Javascript like

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

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

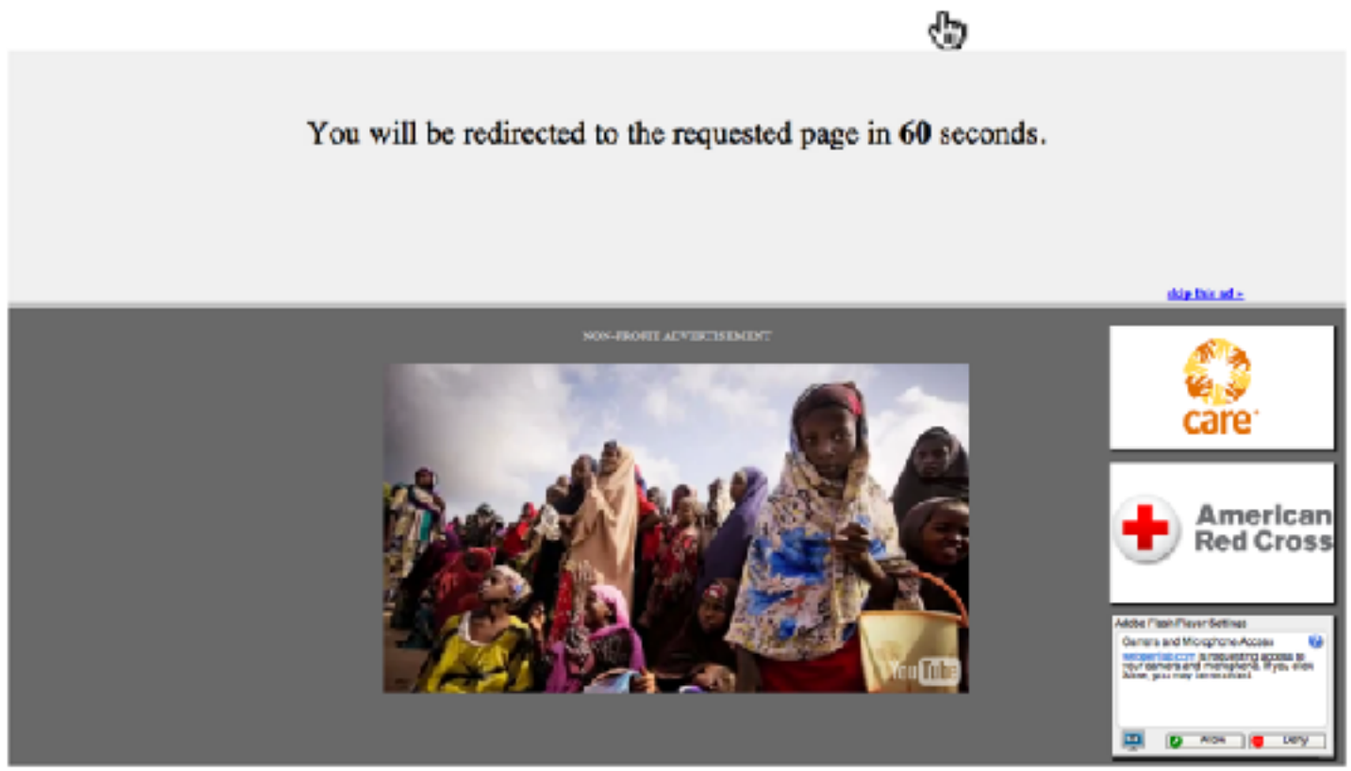
InContext Defense (recent research)

- A set of techniques to ensure context integrity for user actions
- Servers opt-in
  - Let the websites *indicate* their sensitive UIs
  - Let browsers *enforce* c
    when users act on the



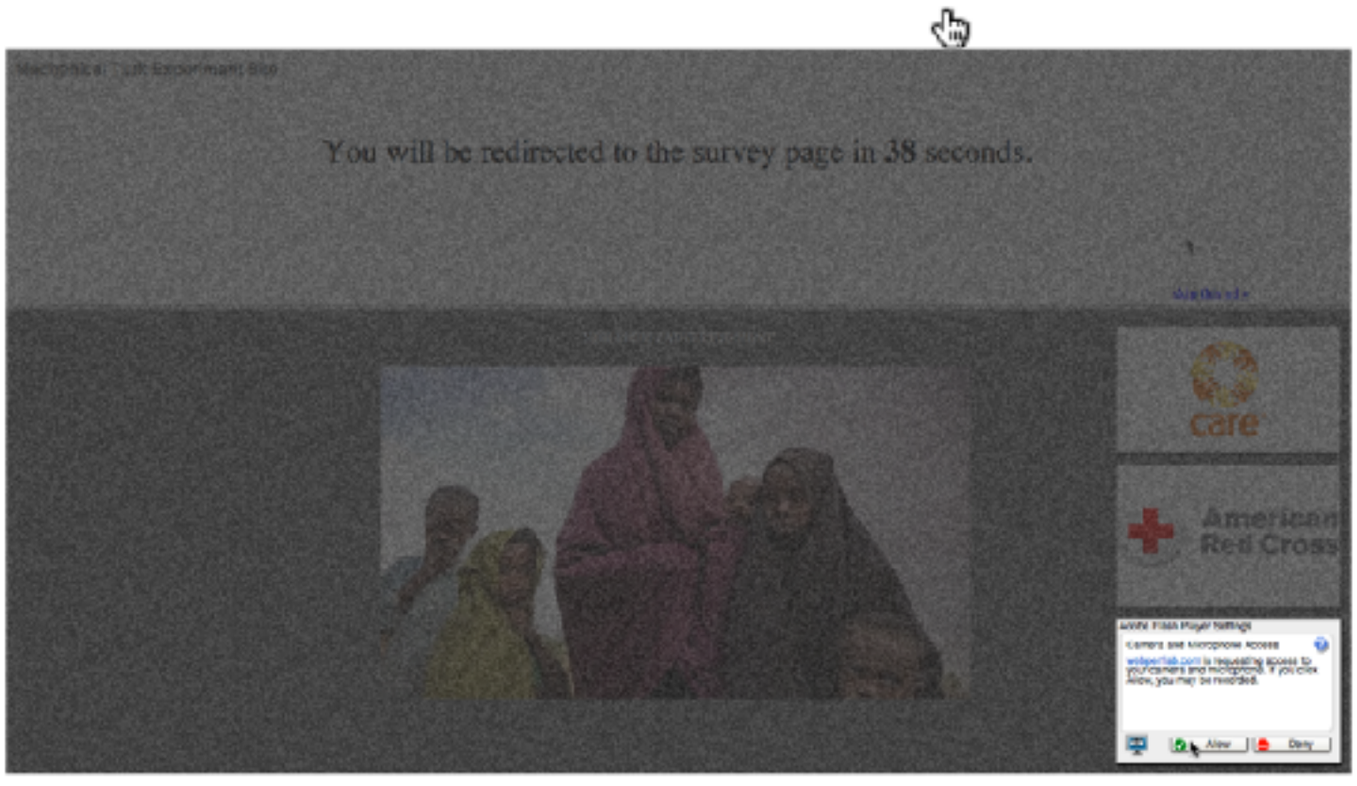
Ensuring visual integrity of pointer

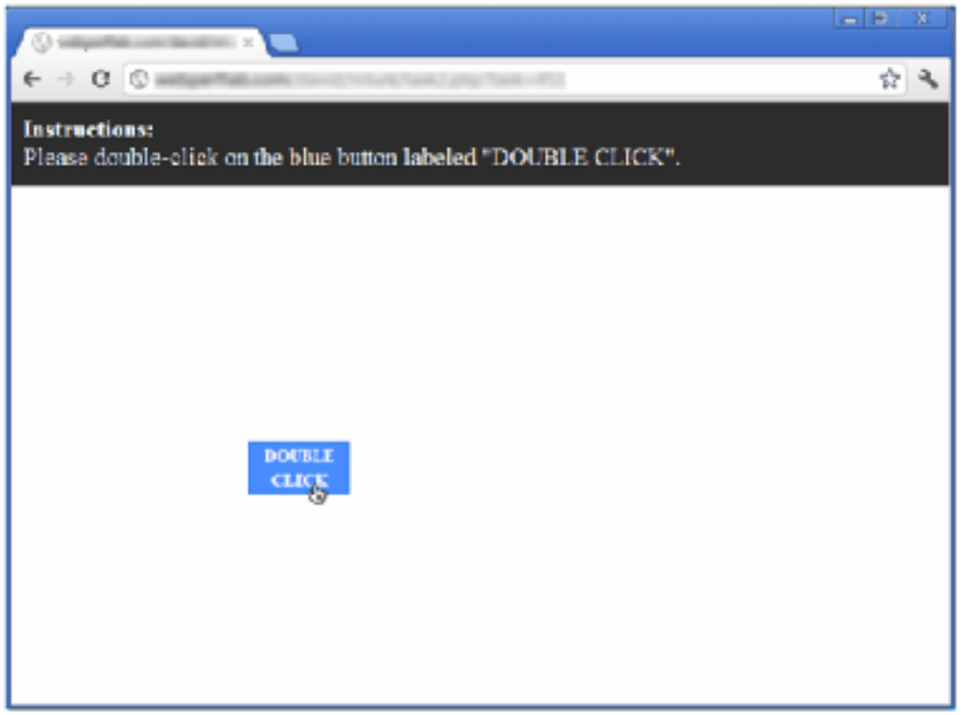
- Remove cursor customization
  - Attack success: 43% -> *16%*



Ensuring visual integrity of pointer

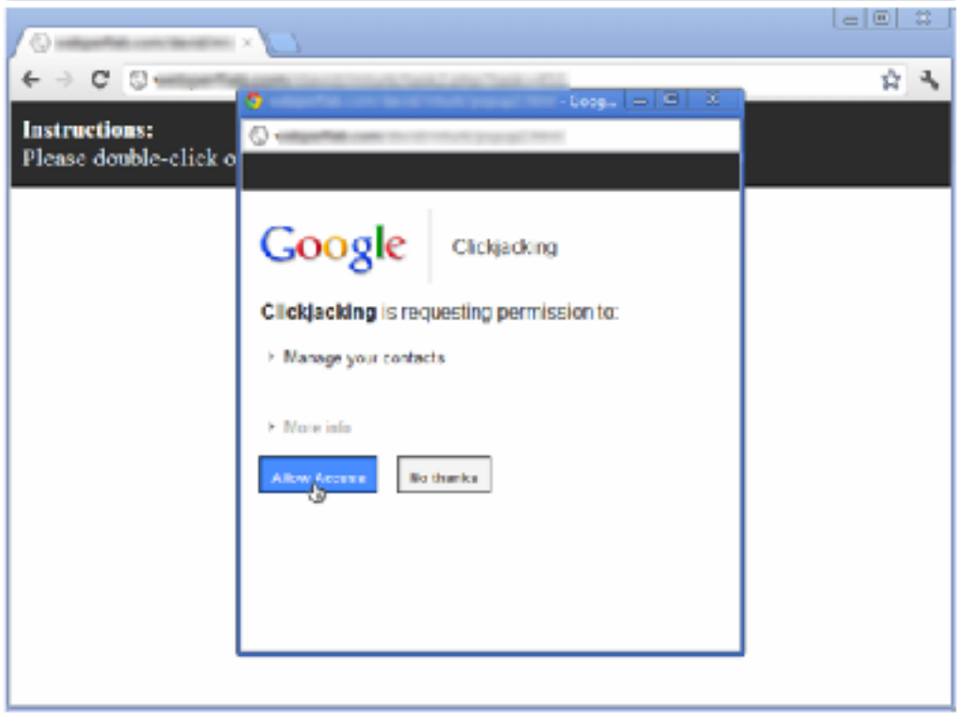
- Lightbox effect around target on pointer entry
  -





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

WHAT IS UNTRUSTWORTHY HERE?

WHAT IS UNTRUSTWORTHY HERE?


Bradesco
Tarifas de
Serviços Bancários
Pessoa Física


Cutujeme, 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

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.

YOUR THOUGHTS: CLICKJACKING

- I liked the very thorough and systematic approach this paper took to defining and sub-classifying clickjacking attacks.

- Shortcomings:
  - it requires websites to identify sensitive elements
  - does not defend against attacks where visibility and temporality are maintained
  - Much of their approach in defending against clickjacking seems like overkill
- Evaluation with Mechanical Turks
  - most fascinating portion of the paper… [MT] seems perfect for recruiting many users to participate in a lightweight study
- touched nicely on the overlap between technical and user problems in security