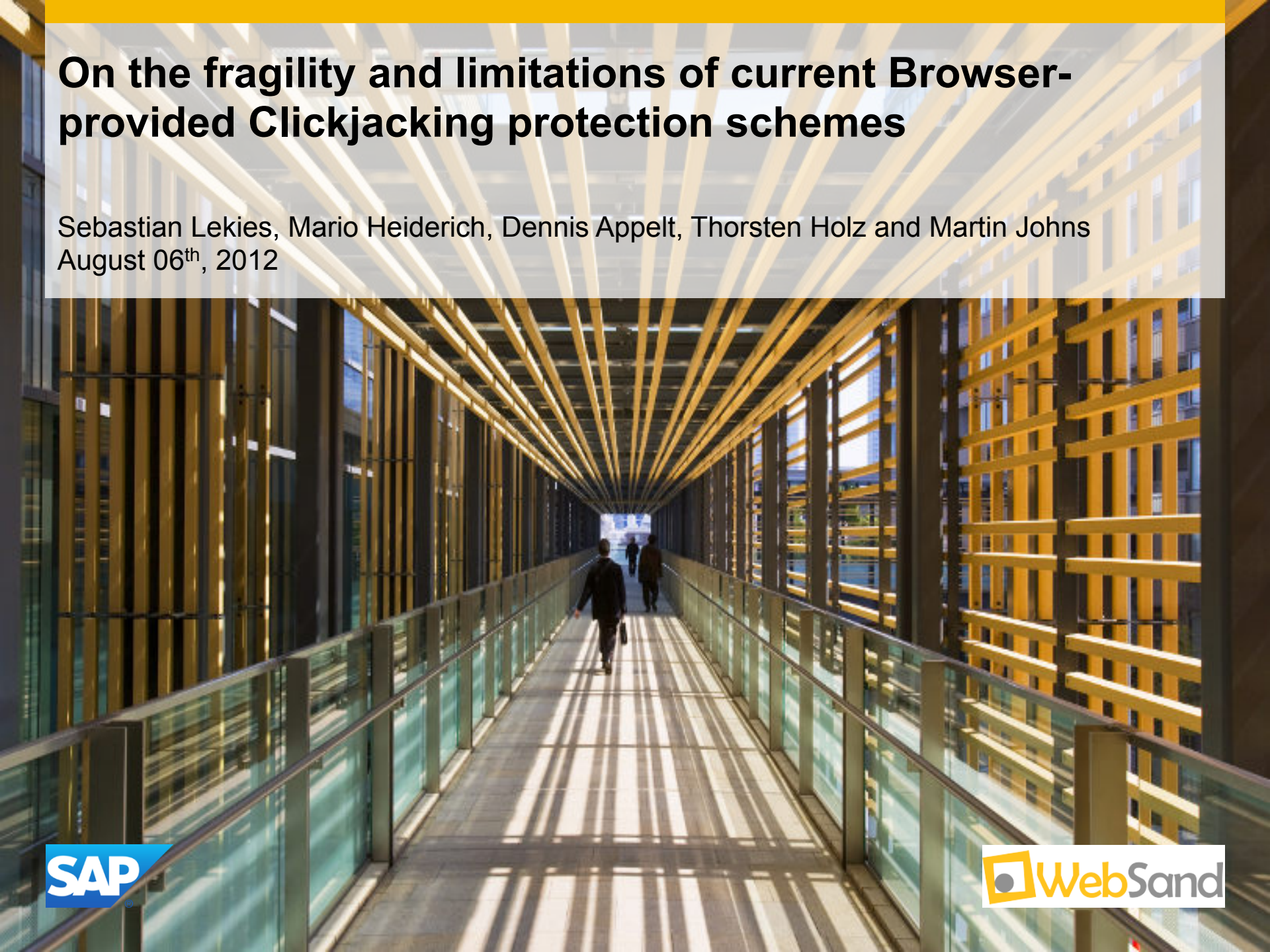
On the fragility and limitations of current Browser-provided Clickjacking protection schemes

Sebastian Lekies, Mario Heiderich, Dennis Appelt, Thorsten Holz and Martin Johns
August 06th, 2012



Agenda

Technical Background

- What is Clickjacking?

Current Defense Techniques

- Client-side approaches
- Server-side approaches

Open Issues

- Non-applicability of defense techniques
- Vulnerabilities

Empirical Study

- Methodology
- Results

Conclusion

Technical Background

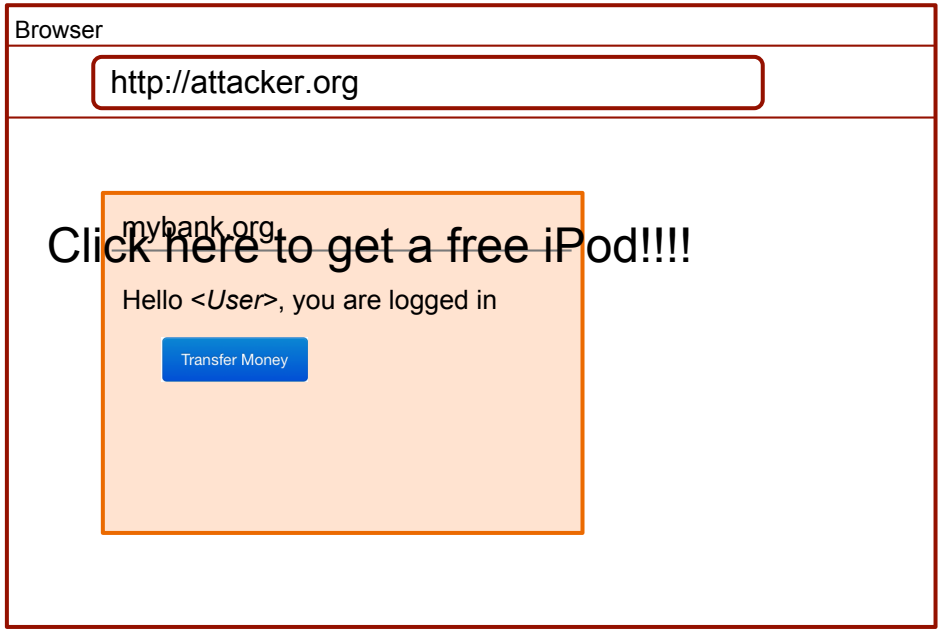
What is Clickjacking?

> *Clickjacking* (also called *UI redressing*) is an attack that lures an unsuspicious user into clicking on an element that is different to what the user perceives to click on.

Technical Background
What is Clickjacking?

Controllable Container
- Frames
- Object, Embed
- Popup windows

Disguising the UI
- Covering it with other elements
- Reducing its size
- Displaying it only for a very short amount of time
- Making it transparent




mybank.org


User

Agenda

Technical Background

- What is Clickjacking?

Current Defense Techniques

- Client-side approaches
- Server-side approaches

Open Issues

- Non-applicability of defense techniques
- Vulnerabilities

Empirical Study

- Methodology
- Results

Conclusion

Current Defense Techniques
Overview

Client-side approaches:

- Protect a single user against attacks on all web applications
- Deployed within the user's browser
  - E.g.: NoScript Clearclick
  - Further countermeasures: GuardedId, Gazelle, OP, Secure Web Browser

Server-side approaches:

- Protect all users of single web application against attacks
- Deployed on the server-side, but enforcement happens on the client-side
  - Frame Busting
  - X-Frame-Options Header
  - Earlier versions of Content-Security Policy

Current Defense Techniques
Client-Side

No-Script Clearclick

- Basic Idea: Prevent clicks on obfuscated cross-domain iframes, objects, embeds
- Malicious Situation:
  - frame, object or embed element overlaps with elements that could potentially receive mouse or keyboard events
  - Opacity of the frame, object or embed element reaches a value below 0.3
- When detecting such a potentially malicious situation a warning is shown to the user
- Two confirmations needed to carry out the potentially malicious action

Alternative Approaches

- GuardedID
- Gazelle
- OP Web browser, Secure Web browser

Current Defense Techniques
Server-Side

Frame Busting

- Basic Idea: Avoid unauthorized framing of a web page
- Implementation: A small snippet of JavaScript code checks if page is framed. If so, it navigates the top frame towards the framed page

```
<script>
  if (parent!= self)
    parent.location = self.location;
</script>
```

Several ways exist to circumvent this protection:

- Prevent JavaScript execution
  - Misusing modern XSS filters
  - Using sandboxed iframes
- Prevent redirect
  - 204 flushing
  - Double framing
  - By asking the user nicely (onbeforeunload event)

It is possible to build secure frame busters. However, the knowledge about it is not widely spread

Current Defense Techniques
Server-Side

X-Frame-Options Header

- Approach introduced by Microsoft to counter Clickjacking attacks
- Similar to frame busting: Avoid unauthorized framing of a page
- Implementation:
  - Non-JavaScript solution
  - Based on an HTTP Response header
  - Browser enforces the Web server's desired behavior

The X-Frame-Options header values:

- SAMEORIGIN: Same-Origin framing only
- DENY: Framing is forbidden
- (IE only: FROMORIGIN: Allows one specific origin to frame the marked page)

Current Defense Techniques
Server-Side

Content-Security Policy

- Earlier revisions of CSP contained a directive called *frame-ancestors*
  - Allows to control framing behavior similar to X-Frame-Options: ALLOWFROM origin
  - Not present anymore in the current revision

Agenda

Technical Background

- What is Clickjacking?

Current Defense Techniques

- Client-side approaches
- Server-side approaches

Open Issues

- Non-applicability of defense techniques
- Vulnerabilities

Empirical Study

- Methodology
- Results

Conclusion

Open Issues

Non-applicability of defense techniques

The current defense mechanisms have one thing in common

- Focus on Framing: prevent unauthorized framing to protect against Clickjacking
- Authorized framing = same-domain framing
- Unauthorized framing = cross-domain framing

How to protect against Clickjacking if cross-domain framing is required?

- No possibility to protect against Clickjacking in this case
- E.g. in Corporate-Portal environments
- E.g. Ad providers

Clickjacking is not limited to frames

- Double Clickjacking
- Clickjacking via History navigation

Open Issues
Non-applicability of defense techniques

Double Clickjacking

Technique developed by Huang and Jackson

- Based on popups instead of frames

Procedure

1. Open target page as a popup window behind the current browser window
   - The opening page receives a window handle = navigational control
2. Lure the victim into double clicking on the current browser window
   - First click: hits current window + triggers focus to the popup
   - Second click: hits the popup + triggers state changing action
3. Close the popup shortly after the first click

E.g. Google OAuth authentication popup was vulnerable to this attack

Open Issues
Non-applicability of defense techniques

Clickjacking via History Navigation (Caching)

Technique developed by Michael Zalewski

- Also based on popups instead of frames

Procedure

1. Open target page as a popup window
   - Page is cached by the browser
2. Immediately navigate the popup to an attacker controlled site
3. Lure user into clicking on the page
4. Shortly before the click: call history.back()
   - Target page is loaded from cache: It is immediately rendered
   - Click hits the target page: Navigate away or close the window afterwards

Open Issues
Vulnerabilities

NoScript Clearclick

- Clearclick uses two basic rules to detect a Clickjacking attack

The user clicks on an embedded cross-domain element that is

1. …(almost) invisible
2. …and overlaps with clickable elements

➢ **By circumventing one of these rules, an attack can be conducted**

Open Issues
Vulnerabilities

Circumventing NoScript Clearclick

1. Cursor jacking (Avoid overlapping of elements)
   - Developed by Kotowicz et al.
   - The real cursor is hidden via CSS (cursor:none)
   - A fake cursor is shown somewhere else

2. Using invalid SVG filters (Avoids transparency detection)
   - When specifying an invalid SVG filter an element becomes invisible (filter: url(invalid);)
   - Element appears to be perfectly visible

3. Drag & Drop of style declarations (Avoids transparency detection)
   - Works by dragging and dropping a style declaration into a frame
   - Makes content of the iframe invisible instead of the iframe itself
   - Content editable div needed: Often used in rich text editors (e.g. used by Webmail applications)

Open Issues
Vulnerabilities

Removing HTTP response headers in Safari

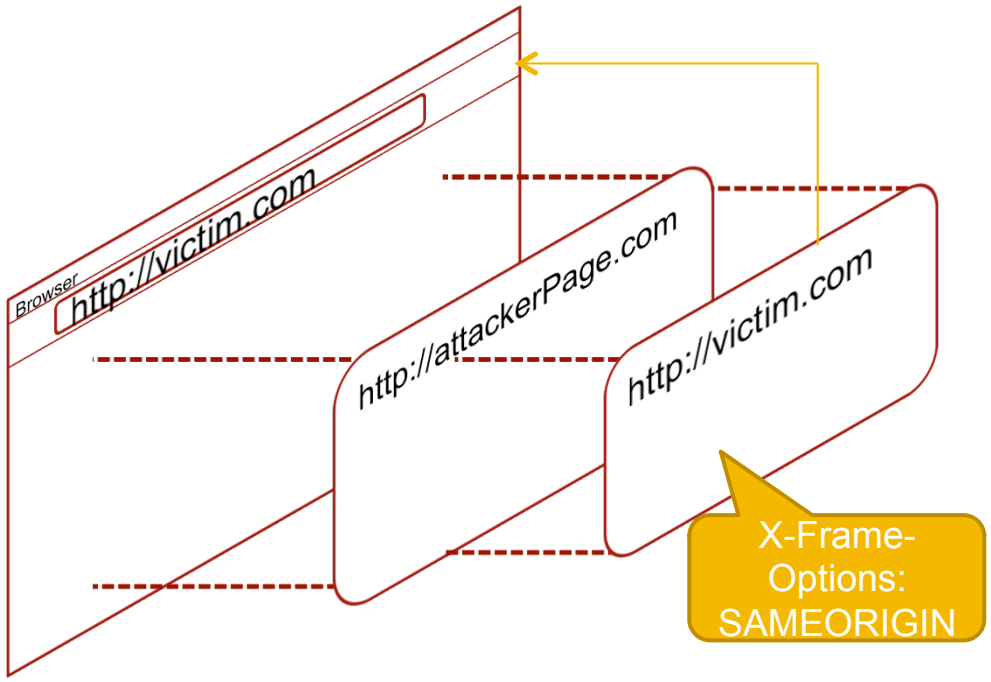
- Vulnerability within the HTML5 Offline Application Cache
- Offline App Cache allows to store HTML documents for offline usage
- Websites are always retrieved from Cache first (even if a connection is available)
- The App Cache does not store HTTP response headers
  - Hence web pages stored/read from cache cannot be protected via X-Frame-Options

Risk Assessment

- HTML5 Offline Applications are often used in mobile environments
  - Mobile version of GMAIL
  - Mobile version of Hotmail
- Given Apple's (and therewith Safari's) market leadership this is a serious vulnerability for iOS users

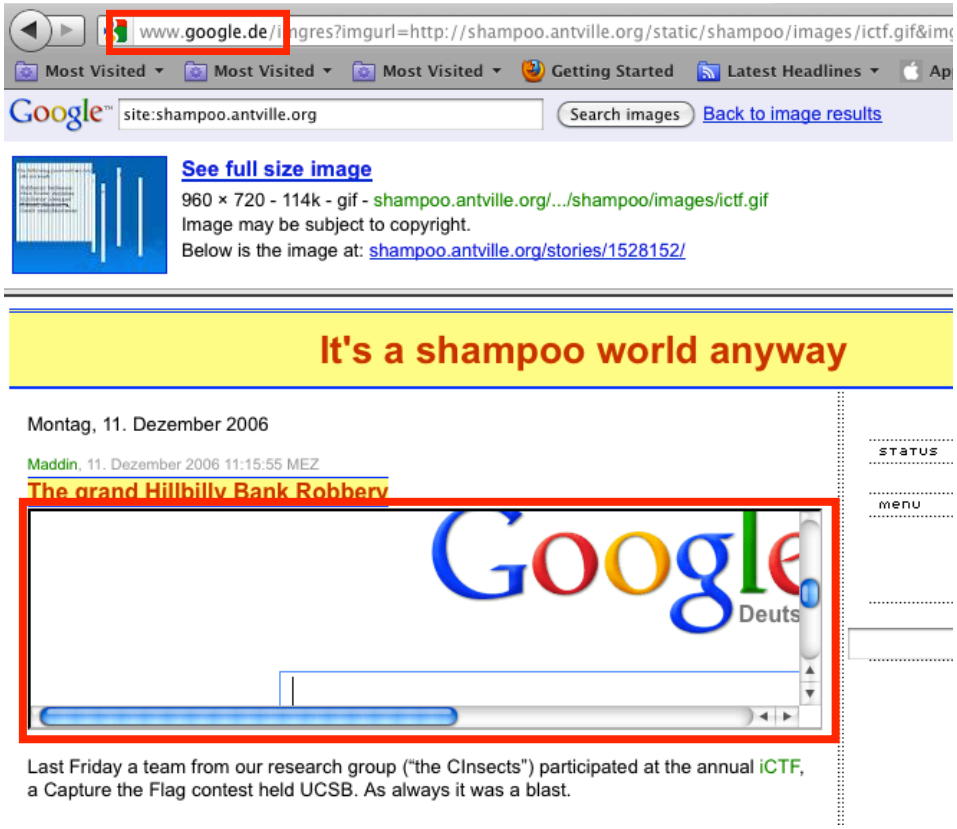
Open Issues
Vulnerabilities

Nested Clickjacking



Open Issues
Vulnerabilities

Nested Clickjacking



Agenda

Technical Background

- What is Clickjacking?

Current Defense Techniques

- Client-side approaches
- Server-side approaches

Open Issues

- Non-applicability of defense techniques
- Vulnerabilities

Empirical Study

- Methodology
- Results

Conclusion

Empirical Study
Methodology

Scope

- Crawl of the Alexa 20.000 + first level subpages
- 2,039,679 million web pages (1,900,463 successful)

Research Questions

1. How many websites make use of frame busters, X-Frame-Options, CSP
2. How many websites have cross-domain frames (and hence are potentially vulnerable to Nested Clickjacking)
3. How many websites are framed by cross-domain websites and are thus not able to deploy X-Frame-Options headers

Empirical Study
Results

General Overview

- 2,975 Web sites (14.88 %) utilized a protection mechanism

Mechanism	Pages	Websites	% Sites
X-Frame-Options	10,982	972	4.86 %
Frame-Busting	87,685	2,230	11.15 %
CSP	13	2	0.01 %

Interesting Observation:

- Clickjacking protection is often deployed only on subpages:
  - Frame-Busting:
    - 4.61% of all pages are protected while in total 11.15 % of all Web sites utilize frame busting
    - 899 Web sites use frame busting on the main page + subpages
    - 1,331 Web sites use frame busting only on some subpages
  - X-Frame-Options:
    - 265 Web sites utilize X-Frame-Options on the main page + subpages
    - 707 Web sites only on some subpages

Empirical Study
Results

Framing Behavior

- 14.449 (72.25 %) of the investigated Web sites make use of frames

- 4,007,176 iFrame elements
  - On average…
    - 200.35 frames per Web sites
    - 2.1 frames per Web page

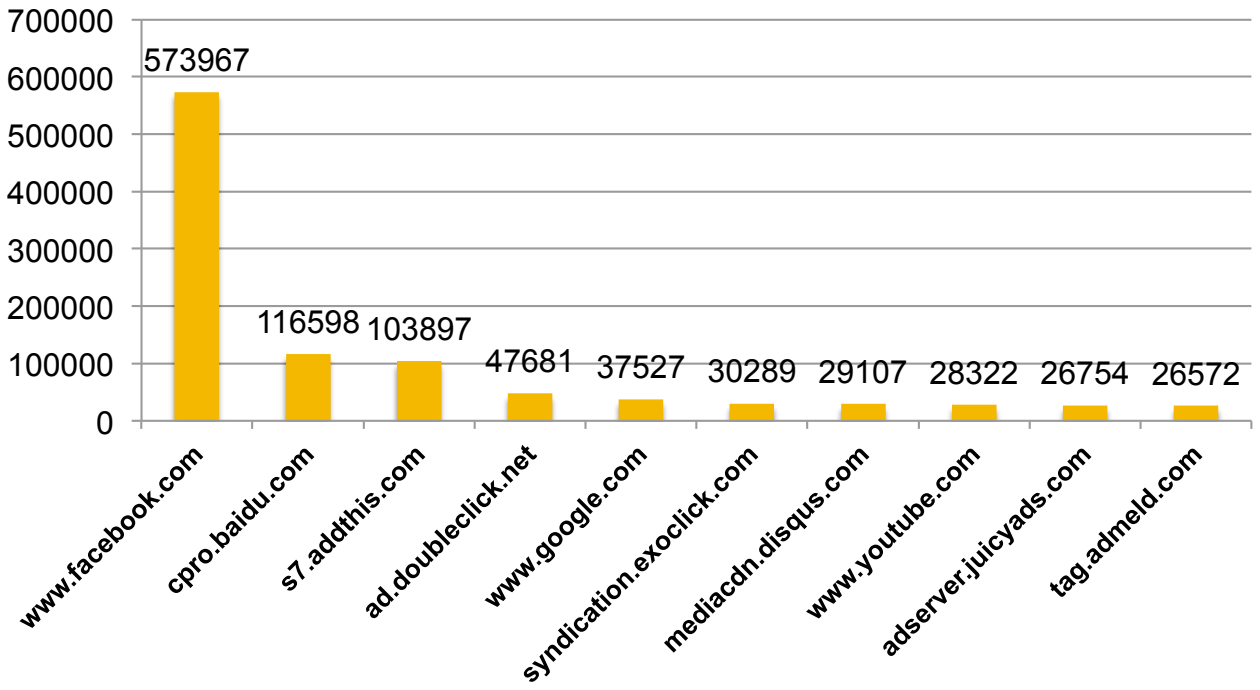
- 2,812,274 (~ 70%) pointing to cross-domain resources
  - Potentially vulnerable to nested Clickjacking
  - Nested Clickjacking only applies to *X-Frame-Options: Same-Origin*

Value	Pages
SAME-ORIGIN	7,906 (72 %)
DENY	3,076 (28 %)

Empirical Study
Results

Applicability of frame-based countermeasures

- 17,496 unique domains are framed across-domain boundaries
  - not able to deploy frame-based anti-clickjacking solutions



Agenda

Technical Background

- What is Clickjacking?

Current Defense Techniques

- Client-side approaches
- Server-side approaches

Open Issues

- Non-applicability of defense techniques
- Vulnerabilities

Empirical Study

- Methodology
- Results

Conclusion

Conclusion

Clickjacking

- Attack that hijacks clicks from an unsuspicious user

Protection Approaches

- Client-side: Clearclick or alternative browser designs
- Server-side: X-Frame-Options, Frame busting, CSP
- All techniques focus on preventing framing
- Techniques are not applicable in many cases
- There are vulnerabilities: Nested Clickjacking, Clearclick circumvention

Empirical Study

- Clickjacking protection is only applied to neuralgic points
  - Possible interpretation: Current mechanisms are not flexible enough

The nature of Clickjacking is complex

- Frame-based solutions do not solve all problems
- Instead of an X-Frame-Options-Header we rather need X-Viewport-Options

Thank you

Contact Information:
Sebastian Lekies
SAP Research
Sebastian.lekies@sap.com
Twitter: @sebastianlekies