New Insights into Clickjacking

Marco `embyte` Balduzzi
iSecLab @ EURECOM
embyte@iseclab.org

OWASP
AppSec Research 2010

Joint work with Egele, Kirda, Balzarotti and Kruegel

Clickjacking

Recent web threat, introduced by Robert Hansen and Jeremy Grossman in September 2008

Construct a malicious web-page (benign site with a XSS vulnerability) to trick the user into performing unintended clicks that are advantageous for the attacker

Propagate worms, steal confidential information (passwords, cookies), send spam, delete personal e-mails, etc...

Attracted a broad attention by the security industry and the web community

Objectives

Determinate the prevalence of clickjacking on the Internet

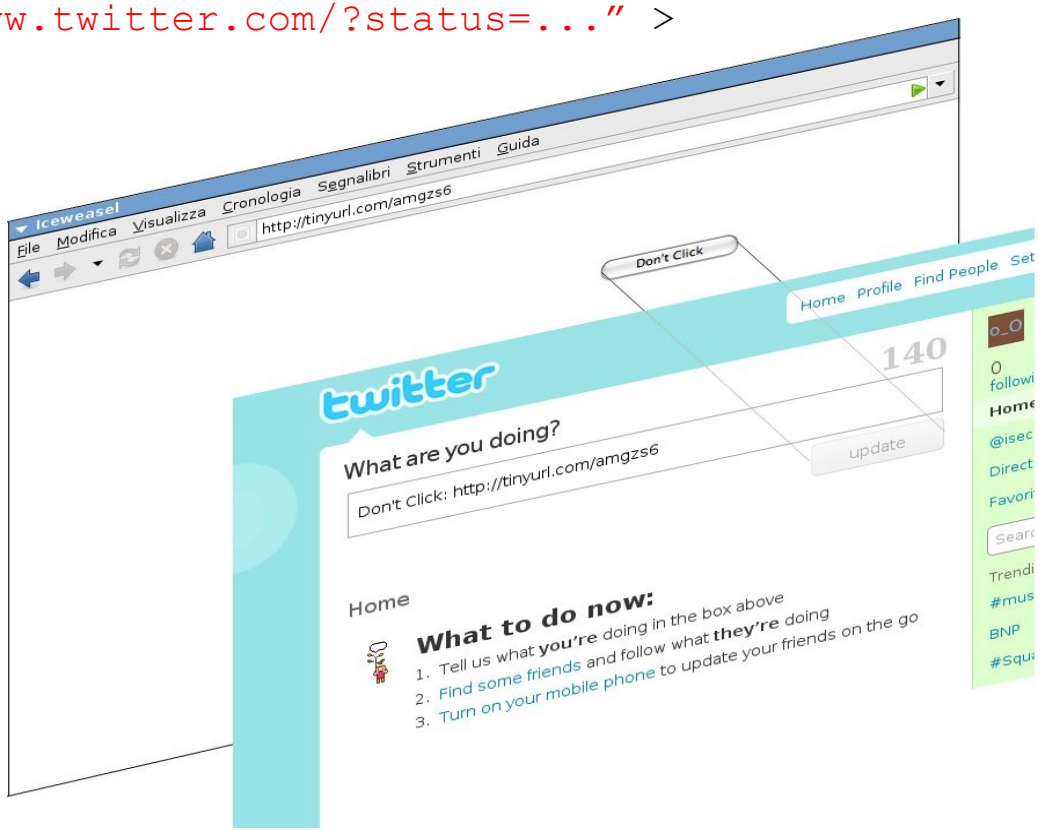
The “Twitter bomb”

Self-replicating message that is twitter via Clickjacking

Abuse of some HTML/CSS features (transparent IFRAMEs)

```
<IFRAME style={z-index:2; opacity:0; filter:alpha(opacity=0); }
   scrolling=”no” src=”http://www.twitter.com/?status=...” >
```

The same attack can be reused to spread malware through drive-by-download sites, to send spam messages or to steal confidential information



→ February 2009, Mahemoff, Explaining the “Don't Click” Clickjacking Tweetbomb
http://softwareas.com/explaining-the-dont-click-clickjacking-tweetbomb

Approach

All-in-one solution

- Combine a *testing unit* with a *detection unit*

Automated

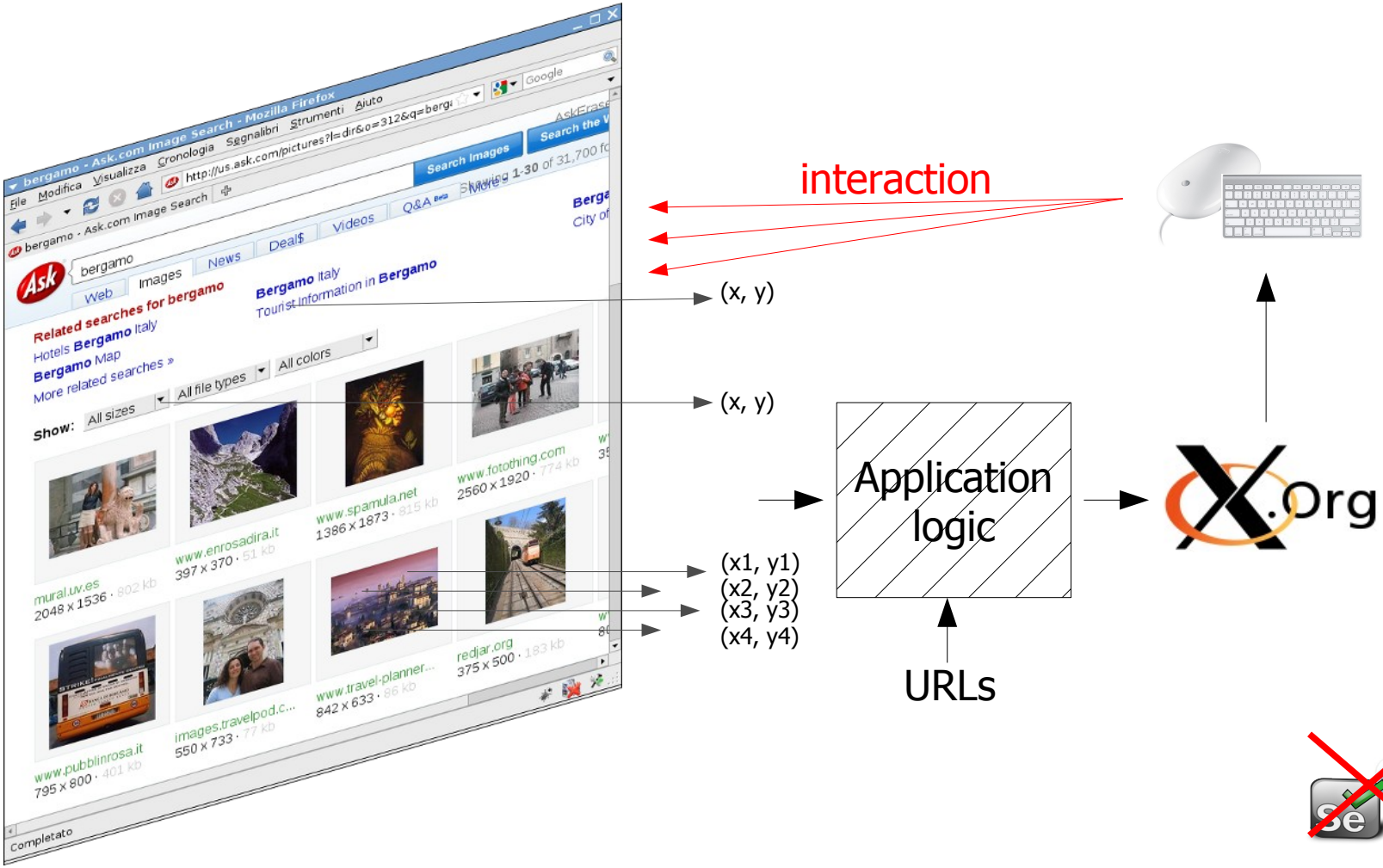
- Instruct a browser to simulate user-real actions (clicks, scroll)
- Automate the testing on multiple sequential pages

Efficient detection

- Analyze the clicks with two independent browser plug-ins
- Detect possible clickjacking attacks

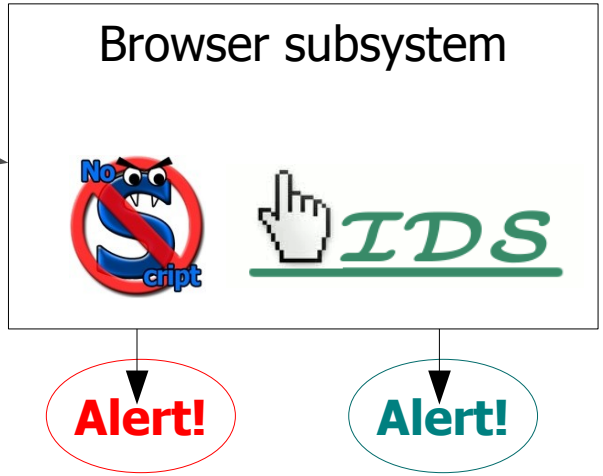
Collect statistics on the visited pages

Testing


interaction
(x, y)
(x, y)
(x1, y1)
(x2, y2)
(x3, y3)
(x4, y4)
Application logic
URLs
X.Org

Detection

clicks



click is discarded

Experiments [1/2]

Validation of the tool on 5 test cases

Initial seed of 70,000 unique URLs:

- Popular: Alexa's Top 1000
- Social-networks: 20.000 MySpace public profiles
- Google and Yahoo queries for malicious keywords (download warez, free ringtones, porn, etc...)
- Phishing URLs from *PhishTank*
- Malicious domains for *MalwareDomains*
- Sites accessed by *Anubis's malwares*

Fed into a crawler that generates:

- 1,065,420 online Internet pages
- 830,000 unique domains

Experiments [2/2]

10 Linux Virtual Machines

2 months (71 days) → 15,006 pages/day

92% of the visited pages embeds elements such as links and forms

143 million clickable elements

Frame statistics:

- 3.3% standard Frames
- 37.3% Iframes
- Only 0.16% were transparent

Discussion – True Positives

Identified two real-world clickjacking attacks

1) Click fraud: Tricks users into clicking on a transparent Iframe that contain a concealed banner

2) Twitter attack:

- anti-clickjacking defense in place (if iframed → substitute with empty content)

Examples posted on security-related sites

Not aware of them. Detected automatically.

Detection	Total	True Positives	Borderlines	False Positives
ClickIDS	137	2	5	130
NoScript	535	2	31	502
Both	6	2	0	4

Discussion – False Positives

NoScript:

1. Pop-ups that appear in response to particular events
2. Iframed banners in the proximity of the click
3. Hidden Iframes located outside the page margins

ClickIDS:

1. Visible Iframes that overlap and contain clickable elements

   Observed multiple sites that were "Frame-defaced": A javascript loads the attacker page and displays it fullscreen (→ Clickjacking through a stored-XSS?)

Detection	Total	True Positives	Borderlines	False Positives
ClickIDS	137	2	5	130
NoScript	535	2	31	502
Both	6	2	0	4

Discussion of Borderline Cases

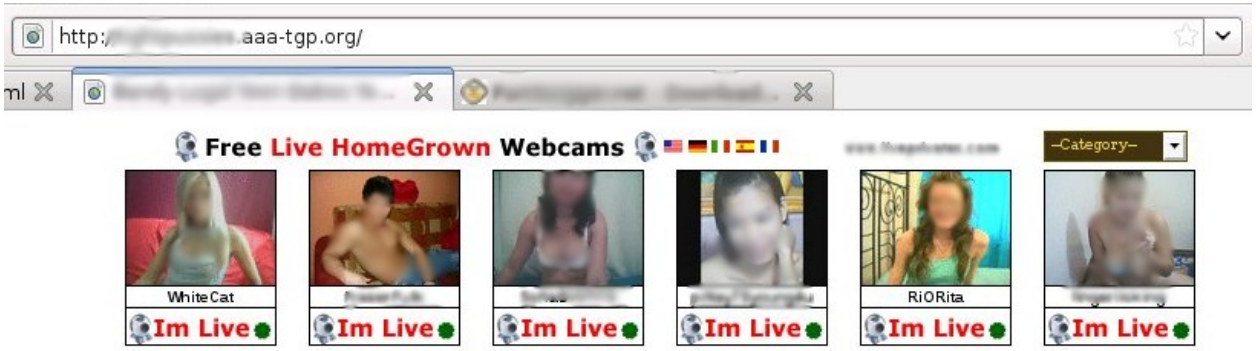
Reverse Clickjacking

A cross-domain Iframe is encapsulated into a link tag:

```
<A href="http://evil.com"><IFRAME src="http://site.com"/></A>
```

Users interact with the framed page *site.com,* but the clicks are grabbed by the link tag and sent to *evil.com*

505 Frame



Iframe with CSS-transparent background

```
Allowtransparency: true & background-color: transparent
```

Normally employed for banner or blogging systems

What have we learned?

Iframes are largely adopted on the Internet and it seems that have overcome traditional frames

→ a new attack vector?

Very few transparent Frames (~3%)

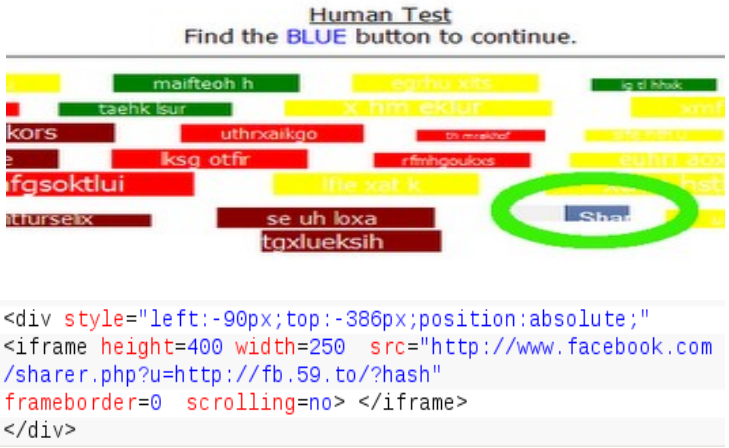
Despite of the wide media coverage we observed very few clickjacked pages and a bunch of borderline cases

Clickjacking is not among the preferred attack vector adopted by miscreants on the Internet

It is complicated to setup and is not easily portable (different browsers / configurations render the page differently)

Looking at the future [1/2]

Facebook worms that use clickjacking (11/09 and 05/10)




```
<div style="left:-90px;top:-386px;position:absolute;"
<iframe height=400 width=250  src="http://www.facebook.com
/sharer.php?u=http://fb.59.to/?hash"
frameborder=0  scrolling=no> </iframe>
</div>
```

Propagation on the own profile

References:

→ [A] Krzysztof Kotowicz, New Facebook clickjacking attacks on the wild
http://blog.kotowicz.net/2009/12/new-facebook-clickjagging-attack-in.html

→ [B] Joey Tyson, Facebook worm uses clickjacking in the wild
http://theharmonyguy.com/2009/11/23/facebook-worm-uses-clickjacking-in-the-wild

→ [C] May 2010 Worms, Attack spreading through "likes"
http://mashable.com/2010/05/31/facebook-like-worm-clickjack/

Looking at the future [2/2]

Use of javascript to position the hidden Iframe

Use of *URL fragment identifiers* to accurately align the frame content

Inject controlled text into a form field using the browser's drag-and-drop API (HTML5)

→ same-origin policy does not applied here

→ Java allow to override the default behavior → initiate the drag with a simple click

Steal the content (and HTML) of a cross-domain page

→ Stone, BH Europe 2010, Next generation clickjacking:

http://contextis.co.uk/resources/white-papers/clickjacking/Context-Clickjacking_white_paper.pdf

Some mitigation techniques

The HTTP X-FRAME-OPTIONS header (proposed my Microsoft and adopted by IE8, Chrome, Opera, Safari, NoScript)

The use of *frame-busting:*

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

- Thwarted by forcing IE to treat the site as restricted (javascript disabled)
- Other variants go around this issue [1]
- A recent paper discusses this problem in detail [2]

The *ClearClick* feature offered by NoScript or *ClickIDS*, or both :-)

Server-side: CAPTCHAs to protect sensitive actions

More references

→ [1] Preventing Frame Busting and Click Jacking (UI Redressing)
http://coderrr.wordpress.com/2009/02/13/preventing-frame-busting-and-click-jacking-ui-redressing/

→ [2] Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites
http://w2spconf.com/2010/papers/p27.pdf

→ A Solution for the Automated Detection of Clickjacking Attacks,
http://www.iseclab.org/people/embyte/papers/asiaccs122-balduzzi.pdf

→ The International Secure System Lab:

- 3 Location: Vienna, Santa Barbara (CA), South-France Riviera
- Applied research in:
  - Web Security
  - Web 2.0 Privacy (Social-Networks)
  - Malware Analysis
  - Botnets Detection

Summary

Motivations:

- Analyze a recent web threat that has received wide media coverage

Approach:

- All-in-one solution for an automated testing and detection of clickjacking attacks

Experiments:

- One million live Internet sites
- Found 2 real cases and some borderline attacks

Is currently Clickjacking posing an important threat for the Internet users?

Thanks!