UI Redressing Attacks on Android Devices

Marcus Niemietz
Ruhr-University Bochum
Horst Görtz Institute

German Book
  - Clickjacking und UI-Redressing

WebAppSec:
  Trainings, Pentests

Speaker at BlueHat, CONFidence, German OWASP Day, PHDays, ...

Twitter: @mniemietz
This talk is based on the – for the Black Hat conference written – paper

UI Redressing Attacks on Android Devices

Authors

Marcus Niemietz, Jörg Schwenk
Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
1. Introduction
2. Related work
3. Porting UI redressing to Android devices
4. New browserless attacks
5. Mitigation techniques
6. Conclusion and outlook
1. Introduction
UI redressing is a known problem since 2002

Security problem has been overlooked until 2008 ➔ Clickjacking

Clickjacking ⊆ UI redressing

- The subclass consists of attacks like cursorjacking, filejacking, tabnabbing, and inter alia tapjacking
- In essence, all of these attacks need a Web browser to be executed
Considering the given attack vectors on desktop-based Web browsers, we pose the following question:

Can UI redressing attacks be ported to smartphone-based systems?
We focus on the Android operating system

Source: Gartner (November 2011)
We focus on the Android operating system

Source: Android.com; 14-day period data - February 1, 2012
Introduction

ANDROID 2.3.3

ANDROID 4.0
This talk focuses on two points

1. Attacks and countermeasures for desktop-based Web browsers available for Android
2. A tapjacking attack technique which does not need a Web browser to execute
2. Related work
2.1. Desktop-based UI Redressing Techniques
2.2. Browserless UI Redressing Attacks
2.1. Desktop-based UI Redressing Techniques
Related work – UI redressing

- Clickjacking
- Strokejacking
- Drag-and-drop operations
- Content extraction
- Event-recycling
- SVG masking
Related work – Clickjacking

- Classic clickjacking
- Likejacking and sharejacking
- Nested clickjacking, double clickjacking
- Cookiejacking, filejacking
- Eventjacking, classjacking
- Cursorjacking, tabnabbing
- Combinations with CSRF, XSS, and CSS
Related work – Clickjacking
Related work – Clickjacking
Related work – Clickjacking
Related work – Clickjacking
Related work – Clickjacking
Related work – Clickjacking

<h1>Funny pictures</h1>
<img src="lol.gif">
<button>Click me</button>
<img src="lol.gif">
<iframe style="position:absolute; z-index:1; opacity:0.0; filter:alpha(opacity=0); left:-120px; top:95px;" width="300" height="200" src="http://www.bing.com">
</iframe>
2.2. Browserless UI Redressing Attacks
Is it possible to perform browser-like UI redressing attacks on mobile devices without using a Web browser or, at the very least, without using it directly?
Related work

David Richardson in 2010 about the Android trust model

- An application is allowed to programmatically open a dialog but not to interact with it

Idea

- Use a toast-view to show a quick little message to the user
  - Basic idea: Be as unobtrusive as possible
Jack Mannino published a proof of concept of a tapjacking attack in 2011

- Scaling the usually small notification message to the entire display of the mobile device
- Subsequent usage of the default constant LENGTH LONG
Crucial point

A touch gesture on such a message or notification will be passed through to the underlying application

Similar to Clickjacking

Idea

Create a notification message, which looks like a normal application
Related work

- LOL
  Funny pictures
  YES
- Awesome App for 200 €
3. Porting UI redressing to Android devices
Porting UI redressing

- Classic clickjacking, classjacking, strokejacking
  - Requires a Web browser supporting frames, CSS, JavaScript, and HTML5
- Nested clickjacking, filejacking, tabnabbing, content extraction, event-recycling, and SVG masking
  - Additional features in desktop-based Web browsers
Nowadays, any Web browser one requires can be downloaded via Google Play

Not transferable attacks

- Cursorjacking
- Cookiejacking
- Double clickjacking and pop-up-blocker bypasses
4. New browserless attacks
In addition to the attack described by Jack Mannino we can do:
- Contact data manipulation
- Native browser utilization
- Touch gestures logging
- Predefined phone calls
- Installing applications in the background
All of these attacks are using the same technique

1. There is a visible attacker's application in form of a notification in the foreground
2. There is a target application in the background
New browserless attacks
New browserless attacks

- There is a limited number of operations like opening the phone call application
- Solution: Unauthorized home screen navigation attack
  - Substantially extend the limited set of attacks
  - An attacker needs more touch gestures of a victim
5. Mitigation techniques
Mitigation techniques

5.1. Browser-Based UI Redressing
5.2. Tapjacking Defense Mechanisms
5.1. Browser-Based UI Redressing
Mitigation techniques

■ Frame Buster
  ■ Consists of a conditional statement and a counter-action
    
    ```
    if (top.location != location)
        top.location = self.location;
    
    ■ Busting frame busting is possible
    ■ August Detlefsen et al. published the most attack-resistant countermeasure against busting frame busting techniques
Mitigation techniques

```javascript
if (self === top) {
    var antiClickjack = document.getElementById("antiClickjack");
    antiClickjack.parentNode.removeChild(antiClickjack);
} else {
    top.location = self.location;
}
```
Mitigation techniques

- X-Frame-Options
  - HTTP header developed by Microsoft in 2008
  - Checks if a website should be loaded in a frame or not
    - DENY
    - SAMEORIGIN
    - ALLOW-FROM origin
  - Restricted to modern browsers such as Firefox ≥3.6.9, Opera ≥10.5, and IE ≥8.
Mitigation techniques

- Content Security Policy
  - Old CSP: Aside from the framing protection, one can also identify other targets, such as preventing data injection attacks or cross-site scripting (*frame-ancestors*)
  - New CSP: Focus on sandboxing and source specification of style sheets, script files and similar issues
## Mitigation techniques

<table>
<thead>
<tr>
<th>Browser</th>
<th>Engine</th>
<th>XFO</th>
<th>oCSP</th>
<th>nCSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android – 4.0.3</td>
<td>WebKit</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Dolphin – 8.7.0</td>
<td>WebKit</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Firefox – 4.0.3</td>
<td>Gecko</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Opera Mini – 7.0</td>
<td>Presto</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Opera Mobile – 12.00</td>
<td>Presto</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
5.2. Tapjacking Defense Mechanisms
Mitigation techniques

- **Android touch filter**
  - Blocks touch gestures received whenever view's window is obscured
  - `setFilterTouchesWhenObscured()`
  - Or, alternatively, with the attribute `android:filterTouchesWhenObscured`
  - Not enabled by default and they are only available in Android versions higher than 2.2
Tapjacking Security Layer (TSL)

- Should be implemented by the Android team into the kernel in the near future
- It opens automatically once a user fires an application
- It is always in the background and remains opened until the application in its forefront gets closed
- A touch gesture on the TSL will be blocked
Mitigation techniques
6. Conclusion and outlook
Most of the existing UI redressing attacks can be used with very little effort.

There are a lot of countermeasures: Frame Buster, XFO, and the CSP.

We have introduced a browserless UI redressing attack and a new security layer against tapjacking attacks.

Conclusion and outlook
We must recommend that vendors of security software urgently implement our TLS HTML5 and CSS3 drafts are partially implemented in Web browsers

- The field of attacks will continuously expand
- Long-lasting 'cat and mouse game'
Thank you for your attention.

Any questions?