Hijacking Innocent Frames

Hijacking Innocent Frames - pdp, gnucitizen.org.

- Published: date not stated
- Original: <https://www.gnucitizen.org/blog/hijacking-innocent-frames/>
- Preserved from: https://www.gnucitizen.org/blog/hijacking-innocent-frames/ (stored) on 2026-08-11
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Hijacking Innocent Frames

Thu, 11 Dec 2008 21:39:16 GMT

by pdp

Magic tricks are all about suggestion, psychology, misdirection and showmanship (see Tricks of the Mind), or as Cutter) perhaps will say, every magic trick has tree parts: the pledge (where the magician shows you something ordinary), the turn (where the ordinary becomes something extraordinary), and the prestige (where the extraordinary turns into something you have never seen before).

In a similar way, real world information security breaches are combination of the characteristics you will often find in the performance of skillful magicians. Therefore, allow me introduce you to a simplistic form of an attack, perhaps so simple that in fact it may work far more often than we would like to admit, which skillfully uses suggestion, psychology, misdirection and a great doze of showmanship.

So, we've all heard of clickjacking and we know that it is a design bug and therefore it is very hard to deal with. However, are there other flawed areas of modern browsers design which can be abused? Of course there are. It just takes time to find them all because they are often well hidden underneath our common believes, ignorance and prejudice. Here is some code:

```
<html>
  <body>
    <script>
      function clickme() {
        var w = window.open('http://www.google.com');

        setTimeout(function () {
          w.location = 'http://www.gnucitizen.org';
        }, 5000);
      }
    </script>
    <input type="button" value="click me" onclick="clickme(this)"/>
  </body>
</html>
```

"Quite boring! I agree." First of all the user clicks on a button/link. Then a new tab/window opens which loads the content of `http://www.google.com`. Five seconds later, the newly created tab is preloaded with the content of `http://www.gnucitizen.org`. Do you find this code disturbing? I do. It is disturbing because it breaks the trust relationship that is going on between the user and `google.com` in this specific example. Call it **surfjacking**, **framejacking**, **tabjacking** or whatever you want to call it, but at the end of the day, I believe that this is just yet another form of bad design.

Here is another example. You browse the web, you click to digg a story, you get redirected to `digg.com` to login. SSL looks fine. The browser lights up all green. It is OK to type your username/password and you do. In the background, the page which initially took you to `digg.com` waits for you to login. It subsequently queries the `digg.com` login page for changes in the DOM structure by using script tags and error handlers to capture different error code offsets (check AttackAPI), and as such it tries to detect when you are fully logged on. It does these checks every half a second. Once a successful login is detected, it simply fires `w.location = "some evil url here";` which will force the browser to render something else, perhaps something malicious, instead of the page that should have came after a successful authentication. Perhaps, the evil caller could even fire just a simple `alert('Hey there!');` message as a form of misdirection and than return back the control with another `w.focus()`.

Would you check the address bar again? Perhaps not, because the page which was forced onto you now contains similarly looking `digg.com` login page accompanied with some red and quite scary looking text which tells you that your login was unsuccessful. This is the psychology. The attacker uses the red color to distract your from the address bar so that you put all of your attention into the login form. You cannot escape your instincts. The forms screams at you that all you have to do is to fill in your username and password and everything will be fine again. You rush to fill in your credentials again. Your request is recorded. A 302 redirect fires back and the browser redirects you to your `digg.com` account like nothing has ever happened. This is the prestige.

As far as I know, although I might be wrong, this form of an attack is **new**. It is definitely not devastating and it wont break the Web. However, my honest opinion is that it does break a lot of things. For example, it breaks the user's normal surfing experience. The good news is that there is

an easy fix. Simply put, do not allow pages to redirect windows which are preloaded with content from a different origin! We fix this, we save the Web again.
