BUG TRACKER

Ticket #9521 (closed bug: fixed)

XSS WITH $(LOCATION.HASH) AND $(#<TAG>) IS NEEDED?

Opened 3 years ago

Last modified 17 months ago

Reported by:	jquery@…	Owned by:	dmethvin
Priority:	blocker	Milestone:	~~1.7~~
Component:	core	Version:	1.6.1
Keywords:		Cc:	
Blocking:		Blocked by:	

Description

I found and reported this xss pattern in evernote.com, skype.com and many famous websites.

see http://ma.la/jquery_xss/

and many jQuery plugin has this XSS pattern.

https://github.com/rodbegbie/threequarters/blob/master/htdocs/design/threequarters.js#L4-5
https://github.com/vitch/jScrollPane/blob/master/script/jquery.jscrollpane.js#L1013-1016
https://github.com/kastner/audio-sinner/blob/master/public/javascripts/app.js#L19
https://github.com/steadicat/labels/blob/master/tabs.js#L5-7

"$(location.hash)" expected CSS selector in many case, but this code also can create html element.

the quick patch by jquery is here

```
-       quickExpr = /^(?:[^<]*(<[\w\W]+>)[^>]*$|#([\w\-]*)$)/,
+       quickExpr = /^(?:[^#<]*(<[\w\W]+>)[^>]*$|#([\w\-]*)$)/,
```

CHANGE HISTORY

Changed 3 years ago by dmethvin comment:1

- **Priority** changed from *undecided* to *high*
- **Status** changed from *new* to *open*
- **Version** set to *1.6.1*
- **Component** changed from *unfiled* to *core*

Yep: http://jsfiddle.net/UyuBx/

However, I am not sure how we can generally protect against this. The hash symbol is just text, and the string is valid HTML. Users should not be passing untrusted input to `$()` that could contain script.

I guess the question is whether this is such a common bone-headed move that we need to prioritize an id selector over HTML for this case. I could be convinced.

In any case, if people wait for us to plug the holes then the problem won't be fixed until we release 1.7 AND all the vulnerable web pages upgrade.

Changed 3 years ago by jquery@… comment:2

jQuery Mobile is also vulnerable

https://github.com/jquery/jquery-mobile/pull/1789

https://github.com/jquery/jquery-mobile/blob/1.0a3/js/jquery.mobile.navigation.js#L474

Changed 3 years ago by jquery@… comment:3

Hi again, jQuery Mobile's case is $(":jqm(url=' + location.hash.replace(/#/,"") + ')")

so my quick patch can't fix XSS in this case.

I suggest these

```
quickExpr = /^(?:\s*(<[\w\W]+>)[^>]*$|#([\w\-]*)$)/,
$("   \t  <img>") is OK
$("anychar<img>") is NG
```

and

```
if( $.safetyMode == true ) { // or more cute name
       // code for disable create html with $ function
}

$("<img>"); // throw error
$.html("<img>"); // create Element
```

This changes clarifies what we are doing. And it will become easy to inspect DOM XSS.

Of course upgrade jQuery is needed.

Thanks.

Changed 3 years ago by dmethvin comment:4

- **Priority** changed from *high* to *blocker*
- **Milestone** changed from *1.next* to *1.7*

We're going to change this for 1.7

Changed 3 years ago by rwaldron comment:5

~~#9776~~ is a duplicate of this ticket.

Changed 3 years ago by ajpiano comment:6

FYI, We're aware that this isn't a "dupe" of 9776, per se, but fixing 9521 will address this problem.

Changed 3 years ago by john comment:7

- **Owner** set to *john*
- **Status** changed from *open* to *assigned*

Changed 3 years ago by john comment:8

- **Milestone** changed from *1.7* to *1.6.3*

Changed 3 years ago by dmethvin comment:9

- **Owner** changed from *john* to *dmethvin*

Changed 3 years ago by Dave Methvin comment:10

- **Status** changed from *assigned* to *closed*
- **Resolution** set to *fixed*

Merge pull request #474 from dmethvin/fix-9521-xss-hash

Fixes ~~#9521~~. Prioritize #id over <tag> to avoid XSS via location.hash.

Changeset: db9e023e62c1ff5d8f21ed9868ab6878da2005e9

Changed 3 years ago by mikeycgto@… comment:11

Saw the jQuery 1.6.3 update in JSWeekly. I too have noticed the potential XSS issue with selectors. I wrote about it a few months ago here and dubbed it "jQuery Selector Injection": http://www.mjcblog.net/2011/06/jquery-selector-injection

As you likely know, the potential for XSS is still present if the selector is not #id based. See my second PoC at the end of that blog post.

My post just tired to emphasize the need, as a developer, to be mindful of user-input and security considerations. It did not suggest any solutions.

Thanks!

Changed 3 years ago by dmethvin comment:12

We've done the best we can do with a generic solution for the very common but unwise case of `$(location.hash)`. Other exploits are possible but not as widespread.

I agree that developers need to be aware of any case where they are passing untrusted input to methods that can create HTML, such as `$()` or `$().html()` or the DOM's `.innerHTML` property for that matter. We cannot solve the problem inside jQuery because we cannot judge the trustworthiness of the string being passed.

Changed 3 years ago by mikeycgto@… comment:13

Couldn't agree more, no sense in sacrificing performance or functionality in order to solve every possible attack vector.

Perhaps the documentation should have some information regarding this, such as some sort of security notes section. Regardless though, you can't secure against stupid ;)

Changed 3 years ago by jquery@… comment:14

FYI

http://twitter.com/#!/kkotowicz/status/113226491373961216

http://twitter.com/#!/DOMXss/status/113208406986342400

Please consider again

```
quickExpr = /^(?:\s*(<[\w\W]+>)[^>]*$|#([\w\-]*)$)/
```

and remember ~~#9776~~

Changed 3 years ago by jdalton comment:15

Ok so there may be an issue with 1.7: http://twitter.com/bulkneets/statuses/156620076160786432

The `location.hash` in the example is `#p=<img src%3D/%20onerror%3Dalert(1)>` which may make it less of a concern as `@dmethvin` points out, but I figured I would log the reported issue.

A reduced example would be:

```
var param = unescape((/[#&]p=([^&]+)/.exec(location.hash) || 0)[1]);
$('.sub a[href*="' + param + '"]:first'); // returns [<img ...>] and alerts 1
```

The old patch:
https://github.com/jquery/jquery/commit/749dbad981f040bd65cbb50c10e9aa6e44bd26ff

Last edited 3 years ago by jdalton (previous) (diff)

Changed 3 years ago by dmethvin comment:16

> Here is #XSS caused by `$('.sub a[href*=" $query_param "]:first')`

The practice of blindly feeding untrusted strings to jQuery really has to be fixed at the site level. If you can control $query_param, why not close the string and inject arbitrary Javascript directly?

Changed 3 years ago by mathias comment:17

@dmethvin It's not actually PHP or anything. The string is concatenated via JS.

Here's the uncompressed JS: http://jqapi.com/js/main.js

And here's the insecure part:

```
    $(window).bind('hashchange', function(event) {
      var state = event.getState();
      if(state.p) loadPage($('.sub a[href*="/' + state.p + '/"]:first'));
    }).trigger('hashchange');
```

Note that `event.getState()` is an addition by jQuery BBQ.

Last edited 3 years ago by mathias (previous) (diff)

Changed 3 years ago by dmethvin comment:18

@mathias, OK, that makes more sense. But the problem is still that the caller is using untrusted data. Am I missing some simple fix we can apply at the jQuery/Sizzle level to fix this in a general way?

Changed 3 years ago by jdalton comment:19

`@dmethvin` The bug is that passing `$(someSelectorNotHtml)` will create a new element when it isn't intended to.

Last edited 3 years ago by jdalton (previous) (diff)

Changed 3 years ago by dmethvin comment:20

- **Status** changed from *closed* to *reopened*
- **Resolution** *fixed* deleted
- **Milestone** changed from *1.6.3* to *1.8*

@jdalton and I talked about this, and I don't think it can be solved in a general way unless we make breaking changes to the "looks like HTML" rule.

Right now we look deep into the string for tags using a regexp (quickExpr), which means `"'.sub a[href*="<img src=/ onerror=alert(1)>"]:first'` looks like HTML with some other text nodes around it. Since top-level text nodes are generally discarded in a jQuery collection it only leaves the image.

The proposed solution would be to only allow strings beginning (and ending?) in angle brackets to be recognized as HTML.

An alternative solution would be to create some sort of "looks like a selector" rule but that would seem to be a lot more troublesome.

In any case this is no doubt a big breaking change to some poorly-written code out there.

Thoughts?

Changed 3 years ago by jquery@… comment:21

@dmethvin, that is this.

```
quickExpr = /^(?:\s*(<[\w\W]+>)[^>]*$|#([\w\-]*)$)/
```

that means create html elements must start with "<" or "\s*<".

$("sometext<img>") is useless, that create just <img> tag, same as $("<img>")

The patch will make very very few incompatibility that can ignore.

Changed 3 years ago by dmethvin comment:22

- **Status** changed from *reopened* to *open*

Why bother with this complex regexp at all? It seems like we are saying the string must have optional leading spaces and a < to be considered HTML.

> The patch will make very very few incompatibility that can ignore.

If there is one thing I have learned over 5 years on this project, it is that one man's bug is another man's feature. Any change in behavior, documented or not, causes stuff to break.

Changed 3 years ago by mala <jquery@…> comment:23

Hi @dmethvin, I agree and I know how backward-compatibility is important.

```
var tmpl = $("#template_element").val(); // textarea, script, etc.
var element = $(tmpl); // "\s*<" will accept "\n\t\s<tags>text</tags>"
```

If this change will break someone's code, that is already wrong code or vulnerable code. I've never seen $("sometext<tag></tag>") snippets over 5 years.

I found many XSS on famous web-sites or web-applications caused by this issue. All of them expected that "wrong css selector will return empty element", so I finally think this is a jQuery's vulnerability.

"wrong css selector" is just a bug, but "create unexpected html tag" is a vulnerability. Dangerous operation must be performed explicitly. It will be clear which operation should have priority.

I would like to also hear other people's opinions.

Thanks.

Changed 3 years ago by mala <jquery@…> comment:24

To throw away complex quickExpr, it's time to refactoring.

```
if (/^#[\w\-]*$/.test(selector)) { // HANDLE: $("#id")
    ...
} else if (/^\s*<[\w\W]>.*/.test(selector)) { // HANDLE: $(html)
    ...
} else { // HANDLE: $(expr)
    ...
}
```

How about this? Full section is here https://gist.github.com/1599411

Two test was failed.

```
equal( jQuery(" a<div/>b ").length, 1, "Make sure whitespace and other characters are t
equal( jQuery(" a<div>" + long + "</div>b ").length, 1, "Make sure whitespace and other
```

and other test was passed.

For more loose rule for html is

```
 /^\s*<[\w\W]>.*|^\s+[\w\s]*<[\w\W]>.*/
```

space and alphabet chars will be trimmed,

```
$(" a <img>") // to create <img>
$("a <img>") // throw error.
```

Then all test was passed.

Changed 2 years ago by vidmich@… comment:25

you will not be able to save humans from own stupidity and you definitely should not increase code complexity and decrease performance of jQuery because or them.

Changed 2 years ago by anonymous comment:26

What about splitting the functionality of $() into two pieces? One that has the current behavior, and a new method whose sole purpose is to accept a selector. And this new method would NOT generate a new HTML element no matter what is passed in. That way, developers would have an alternate safe method they could use instead, and this would also not break backward compatibility since it wouldn't change the behavior of $().

The real problem here, in my opinion, is that the $() function is trying to do too many things at once, and since one of those things is unsafe when provided untrusted user input, all other uses of $() are unsafe too, even though those uses should normally be safe no matter what input is provided.

An additional change that could be considered would be to add some kind of optional 'safe' mode, where when it was on, if you passed a selector to $() it would throw an error and tell you that you should use the new 'safe' selector method. This would help people quickly identify unsafe uses of the $() with selectors so they could port their applications to the new safer model. But since its off by default, it wouldn't affect existing jQuery users.

Dave Wichers

Changed 2 years ago by timmywil comment:27

- **Status** changed from *open* to *closed*
- **Resolution** set to *fixed*
- **Milestone** changed from *1.8* to *1.7*

We've already fixed the issue described in this ticket. The comments bring up a separate issue, which is a duplicate of ~~#11290~~.

Changed 2 years ago by mala <jquery@…> comment:28

@timmywil Good work, but some XSS will still work like that

http://jsfiddle.net/VqFWJ/

```
var param = "]<img src='/' onerror='alert(1)'>[";
$("a[rel=" + param + "]")
```

How do you think?

Changed 2 years ago by timmywil comment:29

True, but I think we are mainly concerned with valid input. There is still an issue open to address the remaining xss issues and that is here:

http://bugs.jquery.com/ticket/11617

Changed 17 months ago by augustd comment:30

Also affects jQuery v1.4.2

Download | Documentation | Tutorials | Bug Tracker | Discussion