Should I reject URLs longer than what is expected?

Asked 16 years, 3 months ago Modified 9 years, 5 months ago

Viewed 590 times

5

I am developing an application, and have URLs in the format `www.example.com/some_url/some_parameter/some_keyword` . I know by design that there is a maximum length that these URLs will have (and still be valid). Should I validate the URL length with every request in order to protect against buffer overflow/injection attacks? I believe this is an obvious yes but I'm not a security expert so perhaps I am missing something.

web-services rest security http url

Share

Improve this question

Follow

edited Jul 22, 2015 at 12:55

thecoshman

8,650 ● 8 ● 56 ● 77

asked Sep 18, 2008 at 13:34

Ethan Post

3,050 ● 3 ● 28 ● 28

Similar Techniques are used by Web Application Firewalls if you're interested in more research. – Tom Ritter Sep 18, 2008 at 13:41

13 Answers

Sorted by: Highest score (default)

5

If you are not expecting that input, reject it.

You should always validate your inputs, and certainly discard anything outside of the expected range. If you already know that your URL's honestly won't be beyond a certain length then rejecting it before it gets to the application seems wise.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:37

anonymous

5

Defence in depth is a good principle. But false security measures are a bad principle. The difference depends on a lot of details.

If you're truly confident that any URL over N is invalid, then you may as well reject it. But if it's true, and if the rest of your input validation is correct, then it will get rejected later anyway. So all this check does is potentially, maybe, mitigate the damage caused by some other bug in your code. It's often better to spend your time thinking how to avoid those bugs, than thinking about what N might be.

If you do check the length, then it's still best not to rely on this length limit elsewhere in your code. Doing that couples the different checks more tightly together, and makes it harder to change the limit in the next version, if you change the spec and need to accept longer URLs. For example if the length limit becomes an excuse to put URLs on the stack without due care and attention, then you may be setting someone up for a fall.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:47
Steve Jessop
279k ● 40 ● 469 ● 709

1

how are you so sure that *all* URL longer than N is invalid? If you can be sure, then it shouldn't hurt to limit it just as a sanity check - but don't let this fool you into thinking you've prevented a class of exploit.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:36

community wiki
Chii

1

The only thing I can see that could cause issues is that while today your URL will never exceed N, you cannot guarantee that that won't be the case forever. And in a year, when you go back to make an edit to allow for a url to be N+y in length, you may forget to modify the url rejection code.

You'll always be better off verifying the URL parameters prior to using them.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:41

Stephen Wrighton
37.8k ● 6 ● 70 ● 87

1

Safari, Internet Explorer, and Firefox all have different max lengths that it accepts.

I vote go for the shortest of all three.

http://www.boutell.com/newfaq/misc/urllength.html

Pulled from link -

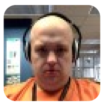
"**Microsoft Internet Explorer (Browser)** - 2,083 characters

Firefox (Browser) - After 65,536 characters, the location bar no longer displays the URL in Windows Firefox 1.5.x. However, longer URLs will work. I stopped testing after 100,000 characters.

Safari (Browser) - At least 80,000 characters will work."

Share Improve this answer Follow

answered Sep 18, 2008 at 15:01



Scott Gowell

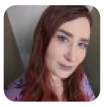
485 ● 1 ● 5 ● 14

0

I think this may give you some modicum of safety and might save you a little bandwidth if people do send you crazy long URLs, but largely you should just validate your data in the actual application as well. Multiple levels of security are generally better, but don't make the mistake of thinking that because you have a (weak) safeguard at the beginning that you won't have issues with the rest.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:37



jamuraa

3,419 ● 27 ● 29

0

I'd say no. It's just false security. Just program well and check your requests for bad stuff. It should be enough.

Also, it's not future proof.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:37

Giuda

7

Almost. Check your requests for *good* stuff, not bad stuff. There will always be bad things you haven't thought of, it's a lot easier to only allow the good things (which are nearly always easier to define). – Doug McClean Jun 25, 2009 at 23:36

0

Yes. If it's too long and you're sure then reject it as soon as possible. If you can, reject it before it reaches your application (for example IISLockdown will do this).

Remember to account for character encoding though.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:38

blowdart
56.5k ● 12 ● 118 ● 151

0

Better than checking length, I think you should check content. You never know how you're going to use your URL schema in the future, but you can always sanitize your inputs. To put a very complex thing very simply: Don't trust user-supplied data. Don't put it directly into DB queries, don't eval() it, don't take anything for granted.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:38

Lucas Oman
15.9k ● 2 ● 46 ● 45

0

If you know valid URLs can't be over *N* bytes then it sounds like a good way to quickly reject cross-site-scripting attempts without too much effort.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:38

pdc
2,394 ● 21 ● 28

0

It's better to validate what is **in the request** than validate URL length.

Your needs may change in the future, at which point you'll have to remove or change the URL length validation, possibly introducing bugs.

If it does end up as a proven security vulnerability, then you can implement it.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:45

Seibar
70.2k ● 38 ● 93 ● 100

0

Ok, let's assume such an N exists. As onebyone pointed out, a malformed URL that is longer than N characters will be rejected by other input validation anyway. However, in my eyes, this opens up a whole new thing to think about:

Using this constant, you can validate your other validation. If the other validations have been unable to detect a certain URL as invalid, however, the URL is longer than N characters, then this URL triggers a bug and should be recorded (and maybe the whole application should shut down, because they might create an invalid URL that is short enough).

Share Improve this answer Follow

answered Sep 18, 2008 at 14:55

Tetha
4,904 ● 1 ● 18 ● 17

0

Oh my, lots of answers, lots of good points, so spread out though, so let me attempt to consolidate all this. **tl;dr** imo, this is too low level a concern for application layer code.

Yes, the URL could be of *any* length, but in practice browsers have a limit. Of course though, that only protects you from browser based attacks by people willing to limit them selves to those vectors, so you do need some way of handling the active attack attempts.

Ok, it can protect against buffer overflows. Well, only if you are working at a low level and not thinking about such concerns. Most languages these days support strings rather well and will not allow them to just overflow. If you were dealing with some very low level system, actually reading the data as bytes and putting it into a 'string' type, then sure, you should have some way of detecting and handling this, but it's not that hard to allocate memory, and transfer known amounts at a time, just keep track of how much memory you set aside. Frankly if you are dealing with that low level, you really should use something else.

Well ok, what about just rejecting based on string length? The major draw back to this is the potential for a false sense of security. That is to say, some areas of the code might get 'sloppy' and be vulnerable to the very exploits you are trying to avoid. You clearly have to be careful to make sure that this 'global' limit actually is sufficient, but considering your URI format, you might be able to have those 'parts' report back what their max length is and central the length checking (for both the entire string, and the components of it); at least this way, if one part needs to allow a longer string, it's easier to handle the change.

This does of course have some advantages to it, for one, it's very quick to be able to compare the length of a string and reject the request right away... but don't forget to be a 'well behaved' site you should be sending back a proper response explaining why the server is rejecting this. In practice though, do you really think you are going to have to handle that many of these types of 'wrong' URL, surely they would be wrong in so many other ways.

For some reason, you felt like not saying what language you are using. High level languages like Java or Python have some very good libraries for dealing with 'web stuff'. Java will let you specify patterns for the URI, including the use of regex for that pattern, so if you wanted a name in the URL, you could have something like

`@Path("/person/(.{0..100}")` to limit the parameter to 100 characters. I'd be surprised if the likes of Ruby or Python didn't have equivalent, they like to promote themselves as nice 'webby' languages.

Finally, regardless of length, there are *many* things that you will need to validate, not just length. Having to worry about the length of the URI causing a buffer overflow is a very low level thing, and would need to be very generic, ie need to handle any request, even one with a 1GB URI potentially; note I said 'handle' not 'accept it an pass it up to the application layer', it could reject it at that low level, also triggering system events maybe.

Share Improve this answer Follow

answered Jul 22, 2015 at 15:16

thecoshman

8,650 ● 8 ● 56 ● 77