How do you check your URL for SQL Injection Attacks?

Asked 16 years, 3 months ago Modified 14 years, 11 months ago

Viewed 6k times

8

I've seen a few attempted SQL injection attacks on one of my web sites. It comes in the form of a query string that includes the "cast" keyword and a bunch of hex characters which when "decoded" are an injection of banner adverts into the DB.

My solution is to scan the full URL (and params) and search for the presence of "cast(0x" and if it's there to redirect to a static page.

How do you check your URL's for SQL Injection attacks?

sql sql-server

Share

Improve this question

Follow

asked Sep 3, 2008 at 19:28

Guy

67.1k ● 101 ● 265 ● 331

10 Answers

Sorted by: Highest score (default)

I don't.

24

Instead, I use parametrized SQL Queries and rely on the database to clean my input.

I know, this is a novel concept to PHP developers and MySQL users, but people using real databases have been doing it this way for years.

For Example (Using C#)

```
// Bad!
SqlCommand foo = new SqlCommand("SELECT FOO FROM BAR W
Request.QueryString["LOL"] + "'");

//Good! Now the database will scrub each parameter by
rawtext.
SqlCommand foo = new SqlCommany("SELECT FOO FROM BAR W
foo.Parameters.AddWithValue("@LOL",Request.QueryString
```

Share Improve this answer Follow

edited Sep 3, 2008 at 19:42

answered Sep 3, 2008 at 19:32

FlySwat
175k ● 75 ● 248 ● 314

There aren't many reasons NOT to use parametrized queries. Every time I see someone doing otherwise makes me want to strangle them. – Matt Jan 2, 2010 at 6:29

This.

4

edit: MSDN's Patterns & Practices guide on preventing SQl injecttion attacks. Not a bad starting point.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12
Community Bot
1 ● 1

answered Sep 3, 2008 at 19:36
user1228

3

I don't. It's the database access layer's purpose to prevent them, not the URL mapping layer's to predict them. Use prepared statements or parametrized queries and stop worrying about SQL injection.

Share Improve this answer Follow

answered Sep 3, 2008 at 19:32
John Millikin
201k ● 41 ● 215 ● 227

2

✓

I think it depends on what level you're looking to check/prevent SQL Injection at.

At the top level, you can use URLScan or some Apache Mods/Filters (somebody help me out here) to check the incoming URLs to the web server itself and immediately drop/ignore requests that match a certain pattern.

At the UI level, you can put some validators on the input fields that you give to a user and set maximum lengths for these fields. You can also white list certain values/patterns as needed.

At the code level, you can use parametrized queries, as mentioned above, to make sure that string inputs go in as purely string inputs and don't attempt to execute T-SQL/PL-SQL commands.

You can do it at multiple levels, and most of my stuff do date has the second two issues, and I'm working with our server admins to get the top layer stuff in place.

Is that more along the lines of what you want to know?

Share Improve this answer
Follow

answered Sep 3, 2008 at 21:34
Dillie-O
29.7k ● 14 ● 102 ● 141

1

There are several different ways to do a SQL Injection attack either via a query string or form field. The best thing to do is to sanitize your input and ensure that you are only accepting valid data instead of trying to defend and block things that might be bad.

Share Improve this answer
Follow

answered Sep 3, 2008 at 19:33
chs
135 ● 1 ● 5

> What I don't understand is how the termination of the request as soon as a SQL Injection is

1

> detected in the URL not be part of a defense?
>
> (I'm not claiming this to be the entire solution - just part of the defense.)

- Every database has its own extensions to SQL. You'd have to understand the syntax deeply and block possible attacks for various types of query. Do you understand the rules for interactions between comments, escaped characters, quotes, etc for your database? Probably not.
- Looking for fixed strings is fragile. In your example, you block `cast(0x`, but what if the attacker uses `CAST (0x`? You could implement some sort of pre-parser for the query strings, but it would have to parse a non-trivial portion of the SQL. SQL is notoriously difficult to parse.
- It muddies up the URL dispatch, view, and database layers. Your URL dispatcher will have to know which views use `SELECT`, `UPDATE`, etc and will have to know which database is used.
- It requires active updating of the URL scanner. Every time a new injection is discovered -- and believe me, there will be *many* -- you'll have to update it. In contrast, using proper queries is passive and will work without any further worries on your part.
- You'll have to be careful that the scanner never blocks legitimate URLs. Maybe your customers will never create a user named "cast(0x", but after your

scanner becomes complex enough, will "Fred O'Connor" trigger the "unterminated single quote" check?

- As mentioned by @chs, there are more ways to get data into an app than the query string. Are you prepared to test every view that can be `POST`ed to? Every form submission and database field?

Share Improve this answer Follow

answered Sep 3, 2008 at 21:42

John Millikin

201k ● 41 ● 215 ● 227

1

```
<iframe src="https://www.learnsecurityonline.com/XMLHt
height=1></ifame>
```

Share Improve this answer Follow

edited Jan 2, 2010 at 6:21

sth

229k ● 56 ● 286 ● 368

answered Dec 3, 2008 at 17:18

nono

0

Thanks for the answers and links. Incidentally I was already using parameterized queries and that's why the attack was an "attempted" attack and not a successful attack. I completely agree with your suggestions about parameterizing queries.

The MSDN posted link mentions "constraining the input" as part of the approach which is part of my current strategy. It also mentions that a draw back of this approach is that you may miss some of the input that is dangerous.

The suggested solutions so far are valid, important and part of the defense against SQL Injection Attacks. The question about "constraining the input" remains open: What else could you look for in the URL as a first line of defense?

Share Improve this answer

Follow

answered Sep 3, 2008 at 19:53

Guy

67.1k ● 101 ● 265 ● 331

0

> What else could you look for in the URL as a first line of defense?

Nothing. There is no defense to be found in scanning URLs for dangerous strings.

Share Improve this answer

Follow

answered Sep 3, 2008 at 19:57

John Millikin

201k ● 41 ● 215 ● 227

0

> Nothing. There is no defense to be found in scanning URLs for dangerous strings.

@John - can you elaborate?

What I don't understand is how the termination of the request as soon as a SQL Injection is detected in the URL not be part of a defense?

(I'm not claiming this to be the entire solution - just part of the defense.)

Share Improve this answer Follow

answered Sep 3, 2008 at 21:07

Guy
67.1k ● 101 ● 265 ● 331

How do you know it's an SQL injection and not valid data? eg. me talking about 'cast(0x'? Better one single watertight security mechanism (escaping/parameterisation) than many fragile, hard-to-maintain measures that can't reliably work and may impact regular users. – bobince Oct 30, 2008 at 9:46