

SecurityFocus Penetration: RE: SQL injection from within a table – is it possible?

RE: SQL injection from within a table – is it possible?

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/pen-test/2005-01/0052.html>

From: Scovetta, Michael V (*Michael.Scovetta_at_ca.com*)

Date: 01/07/05

Date: Fri, 7 Jan 2005 15:59:10 -0500

To: "Burnett, Robert" <burnettr@Fortrex.com>, "Peter Bair" <peterbair100@hotmail.com>, <pen-test@

It is interesting though, as to whether any databases interpreters parse incorrectly on subselects:

Uid_tbl:

uid

1; drop users;

2

3

select * from users where user_id = (select top 1 uid from uid_tbl)

I would assume that all parsers would parse the /entire/ sql query before executing anything, but if not, it would become:

select * from users where user_id = (select top 1 uid from uid_tbl)

select * from users where user_id = 1; drop users;

Michael Scovetta

Computer Associates

Senior Application Developer

-----Original Message-----

From: Burnett, Robert [mailto:burnettr@Fortrex.com]

Sent: Friday, January 07, 2005 10:38 AM

To: Peter Bair; pen-test@securityfocus.com

Subject: RE: SQL injection from within a table – is it possible?

Peter,

Building on what Kevin stated, there is a flaw in the scenario you presented in your post. Suppose your username was "bob", and it was stored in a table called "table", in a field called "name".

The query:

RE: SQL injection from within a table – is it possible?

SecurityFocus Penetration: RE: SQL injection from within a table – is it possible?

select name from table

will NOT become

select bob from table

It will just be: select name from table. "bob" will be the value returned.

Now, if you were querying for a specific username, you could do: select name from table where name = 'bob', but that requires you to know beforehand that you are looking for "bob".

Going back to your initial question about a "stored" SQL Injection exploit, I suppose that if an application was pulling a value from the database by doing a SELECT query and then blindly injecting that retrieved value into a subsequent query, then an exploit could be performed if that stored value was some maliciously crafted SQL code.

For example, if you somehow got the value ' OR 'a'='a into the name field, and then the application retrieved that value, stored it in the variable \$username, and then executed the query "SELECT * from table2 WHERE username='\$username';", the resulting query is:

```
SELECT * from table2 WHERE username=" OR 'a'='a';
```

And we get all the records in the table returned to us.

You get the idea.

Thanks.

Robert Burnett
Fortrex Technologies
5303 Spectrum Drive
Frederick, MD 21703
Toll Free: 1-877-367-8739
Fax: 301-947-3539
E-Mail: burnetr@fortrex.com

-----Original Message-----

From: Peter Bair [mailto:peterbair100@hotmail.com]
Sent: Thursday, January 06, 2005 8:00 PM
To: pen-test@securityfocus.com
Subject: SQL injection from within a table – is it possible?

Is it possible to store an SQL injection string into a MSSQL database table, so when the database performs an action like through a stored proc, the SQL injection attack takes place?

RE: SQL injection from within a table – is it possible?

SecurityFocus Penetration: RE: SQL injection from within a table – is it possible?

Not through the normal means of SQL injection via a web base means, but if you have the means of storing the data into the table directly.

Example:

An application has a users name in a table. Is it possible to assign the users name as the SQL injection string, something like
name from table; exec master.xp_cmdshell "ping me"; --

so when the database is running a stored procedure with a select clause like

```
select name from table
```

it really is becomes

```
select name from table; exec master.xp_cmdshell "ping me" ;-- from table
```

Of course using the SQL query analyzer on the database table, all this works ok.

But when I insert the SQL injection string into the table, as the name, and then query the table nothing happens.

Is it possible or have I missed the point here?

Thanks Peter.

Confidentiality Notice

The content of this communication, along with any attachments, is covered by federal and state law governing electronic communications and may contain confidential and legally privileged information. If the reader of this message is not the intended recipient, you are hereby notified that any dissemination, distribution, use or copying of the information contained herein is strictly prohibited. If you have received this communication in error, please immediately contact us by telephone at (301) 977-6966 or e-mail info@fortrex.com. Thank you.