

Direct Database SQL Injection Attacks – an Interview with Amichai Shulman, CTO of Imperva

Listen to Podcast [here](#).

Brian Contos: Welcome to the Imperva Security podcast. I'm Brian Contos, chief security strategist for Imperva. If you would like to learn more about this subject and Imperva, visit [Imperva.com](https://www.imperva.com), check out our [blog](#), follow us on [Twitter](#), or send us an e-mail at

Brian Contos: Well, welcome to the show, Amichai.

Amichai Shulman: It's a pleasure to be here, Brian.

Brian Contos: So, Amichai, [our last podcast together](#), we were discussing more traditional SQL injection attacks, those that take place over a web application. Today, while we're talking about SQL injection again, we're actually talking about direct database SQL injection attacks. Could you give us a little bit of background of what exactly these attacks are, and how they're different from our previous podcast?

Amichai Shulman: Well, sure, Brian. I think we have to clear up before anything to direct SQL injection into database, first SQL injection into the database is usually through a construct that is called stored procedures. Now, stored procedures are pieces of software that are stored and executed on the database server. They are written using an extended SQL syntax that allows combining of standard SQL queries together with common programming constructs, such as variables, control structures, loops, if statements, and so on.

And they are considered one of the tools that application programmers can use when creating a database application. Now, these stored procedures can, of course, take parameters from the common uses, just like any other procedure.

The interesting thing about stored procedures is that they are executed by one user under the security context of the owner of the stored procedure. The owner of the stored procedure, the creator of the stored procedure can pack together a number of statements that affect a number of tables and then grant execution privileges on that stored procedure to different users without giving them direct privileges over those tables. So, it is a very important construct in terms of security and in terms of integrity of transactions.

As a consequence, if someone is able to manipulate the execution of a stored procedure, then that someone would actually execute code under the security context of the stored procedure owner, which is usually a highly privileged user in the database server. The fact that SQL injection occurs within stored procedures is related to the use of dynamic SQL statements within the stored procedure. These are statements that are not created at the time of stored procedure creation or compilation but rather are created using literal, taken from user input at the time of execution, in a very similar way to traditional SQL injection.

Direct Database SQL Injection Attacks with Amichai Shulman

So, if a stored procedure contains or uses dynamic SQL, and that stored procedure takes parameters from its common user without sanitizing them, and uses those parameters to construct the dynamic SQL statement, then that procedure would be vulnerable to SQL injection, which means that a lower privileged user can execute SQL statements under the security context of a highly privileged user in a database server.

Brian Contos: I see so for something like this, would an organization be able to leverage a vulnerability scanner, for example, to scan their databases for this type of risk, or is this something that requires more detailed code review analysis?

Amichai Shulman: Well, it really depends on what are the vulnerable stored procedures. There are a lot of built-in stored procedures in today's commercial databases that provide a lot of functionality around the core database functionality. And usually, if there are vulnerabilities in those procedures, then they become public by the vendor, and you can then find out whether your version of database server is vulnerable to specific issues in specific stored procedures. If you are something involved in proprietary code, then you probably need to dig more into your code in order to look for those stored procedures that might be vulnerable.

Brian Contos: So, when we're talking about SQL injection attacks launched over a web application, most people associate that with external attacks. But with the direct database SQL injection attack, is that something that is more geared to insider abuse, or is that also something that somebody could leverage from outside of the organization? It seems like it's more of an internally focused issue.

Amichai Shulman: It seems like. And, indeed, this is the kind of issue that can be exploited internally with almost any desktop within an organization, using very simple and available tools on default workstations within enterprises. However, this same problem can be exploited externally through web applications when the application itself is not necessarily vulnerable to SQL injection. In many applications interface between the web application and the database is not necessarily using SQL statements created on the fly, but rather using stored procedures within the database server.

Take, for example, a lot of the Oracle web products actually rely on stored procedures in their communication between the application server and the database server. And if the web server is invoking a vulnerable stored procedure with user input without sanitizing that user input, then that user can very easily take advantage or abuse SQL injection vulnerability within that stored procedure.

Beautiful example provided by David Litchfield. I think it was a year or so ago where the same patch released by Oracle, there was one SQL injection vulnerability in stored procedure, and that stored procedure was exposed by a default through the Oracle portal product, basically allowing an unauthenticated Internet user to take control over a database server.

Brian Contos: So, Amichai, how do people protect from this type of attack? Is it leveraging a database firewall, is it leveraging a web application firewall, a combination, other technologies? What can they do to address this?

Amichai Shulman: So, I think that the first layer of defense would be the database firewall, where you usually would do two things. One is, as it turns out, a lot of the vulnerabilities within built-in stored procedures occur in esoteric built-in stored procedures. They are either very seldom used or they are actually internal stored procedures that are

Direct Database SQL Injection Attacks with Amichai Shulman

used by other stored procedures within the database server. So in that case, you definitely would like to block any access to those vulnerable stored procedures.

And the other thing is for those stored procedures that are required usually for administrative work in database server, or other tasks, you would like to make sure that only privileged users are accessing those stored procedures. So, although they remain vulnerable, abusing these vulnerabilities has no value for those users who have access to them, because they are already privileged.

So this is the first line of defense. And, of course, for exploiting such vulnerabilities through a web application, I would always recommend implementing a web application firewall to protect against any kind of SQL injection, including the direct database SQL injection attacks.

Brian Contos: Good advice. Well, Amichai, I think this was a great addition to our previous SQL injection podcast, and I think it certainly expands upon the details around SQL injection. Thanks so much for joining us today.

Amichai Shulman: Thank you, Brian.

Brian Contos: If you would like to learn more about this subject and Imperva, visit Imperva.com, check out our [blog](#), follow us on [Twitter](#), or send us an e-mail at blog@imperva.com.



North America Headquarters
3400 Bridge Parkway
Suite 101
Redwood Shores, CA 94065
Tel: +1-650-345-9000
Fax: +1-650-345-9004

International Headquarters
125 Menachem Begin Street
Tel Aviv 67010
Israel
Tel: +972-3-684-0100
Fax: +972-3-684-0200