

## SQL Injection – 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 [blog@imperva.com](mailto:blog@imperva.com).

Welcome back to the show Amichai.

**Amichai Shulman:** Thank you, Brian. It's great to be here, always.

**Brian Contos:** So, Amichai, today's topic is SQL injection, and this is sort of the great-grandfather of the web application attack. Could you give our audience a little bit of background on what exactly SQL injection is?

**Amichai Shulman:** You always make me talk again and again about SQL injection and people say, well, that's so '90s. And I sometimes have this feeling myself, but as you mentioned, it is one of the oldest tricks in the book of web application attacks. SQL injection is a vulnerability in the application in which user input is taken directly without any form of validation and sanitation, and is used to construct the text of SQL queries. So it's not just used as a parameter in a SQL query or some constraint on an SQL query, but it is actually taken and used to construct the text of the SQL query.

And when that happens, an attacker can craft the input in a way that would change the entire meaning of the query sent by the application to the database. It could be a small change such as affecting the result set that is returned for the database servers, so instead of retrieving one's own records, you can access records for all users in the system.

It could be a larger change, making the application server retrieve information from a totally different table from the database server. And in some cases it could be even changing in the functionality in terms of inserting records into the database instead of just taking information out or doing other changes to the database server.

**Brian Contos:** So, Amichai, you made a really interesting point at the end there which is, SQL injection is more than just what people sometimes think it is in terms of just extracting data -- whether that be credit card numbers, user account names, passwords, what have you -- but you could also use it to modify tables, to delete tables, to shut down a database, to do a whole number of things way above and beyond simply extracting data.

**Amichai Shulman:** Well, exactly. I have seen cases in which SQL injection was used to tamper with the amount of money in prepaid cards. We have all seen SQL injection being abused for site defacement. That was very popular in 2008. And we have also seen a lot of SQL injection being used for drive-by download attacks, where the actual change to the

content was to introduce a script attack that would drive the innocent visitors to a server that delivers malware.

**Brian Contos:** Of course, and for those listeners who want to hear a little more about drive-by download, we do have a dedicated podcast on that subject that we did just a couple months ago. So, Amichai, my question is this: SQL injection has been around for a very long time. There's been guides to SQL injections attacks, there's been guides to preventing SQL injection attacks, good coding practices, security development life cycles, all these different things.

But still the threat remains and there are still systems that are vulnerable to it, including a recent case that we heard about at Information Week just a couple days ago about the US Army being subject to a SQL injection attack.

Why is it still so pervasive? Is it just one of those things that here, and here to stay?

**Amichai Shulman:** Are bugs going to be extinct any time? Are software bugs going to disappear as a consequence of having larger QA teams and better QA tools? The answer is no. And eventually, SQL injection is bad code. It's a software bug. And we can do a lot in order to try and eliminate those kinds of bugs from our software, but we have to remember, almost all software today is accessing databases. So we have a lot of code around accessing databases, and statistically some of this code is poorly written. So I do not expect SQL injection to go away.

I must give you an example. In one example they had SQL injection vulnerabilities all over it. We told them, OK, you need to use prepared statements. So, they go to their programmers, they go back to use after a week, and they're saying, OK, we've used a prepared statement. Go to the system.

SQL injection all over again. We looked at their code, and what they did was to steal user input in the construction of the query text. But then they took that query text and sent it to the database through a prepared statement. You've got a million reasons for having bad code. Some of the bad code is security-related. Some of it, SQL injection.

**Brian Contos:** I think anybody that's played around with SQL injection attacks -- for research purposes or what have you -- very quickly discovers how creative you can really be in a SQL injection attack. For example, there are literally millions, if not billions or more, ways to say the same thing, to call a script tag, for example, or to use encoding or multiple flavors of encoding. A lot of people, when they look at SQL injection, they feel that perhaps signature matching is the way to go. From what I've seen, I think you might have a subset of approaches to prevent it with signatures. But really it has to go way above and beyond what you could ever do with signature detection.

What are some of the other ways to combat SQL injection?

**Amichai Shulman:** Using plain signatures is indeed not a very effective way to mitigate SQL injection. This is a topic that has been well-discussed. We have presented this topic a few years ago in RSA, and we do have a whitepaper in the presentation on our website regarding this. Basically, as you mentioned, the number of potential combinations that can be used for saying the same thing is huge. In order to effectively detect SQL injection, you must be able to combine some positive security model with some negative security model in terms of detecting first when an input is not normal. Then when you have the suspicion that an input is not normal, you can then start to evaluate whether it looks like SQL injection.

## SQL Injection with Amichai Shulman

In order to do that effectively, you need to try and think like or emulate the thought-line of an attacker in terms of what would an attacker try to achieve, and how would a SQL injection look like if an attacker is trying to achieve data tampering, defacement, extraction of information from the same table or extraction of information from a different table.

Then, when you have these lines of thought, you can formalize them in the forms of regular expressions or more complex conditions on the input, and be able to come up with a sensible conclusion.

Of course, I forgot to mention that prior to all this, you will probably have to do some normalization on the input in order to avoid different evasion techniques such as comments and character-code-page-related evasion techniques.

**Brian Contos:** Amichai, for your parting comments here, I'd just like it if you could just hit on one more item, which is SQL injection attacks that are launched directly against a database server. Thus far we've been talking about going through a web application, perhaps mitigating the threat with a web application firewall and some other solutions that use a variety of techniques above and beyond simple signature detection -- which we know doesn't work -- Oracle stored procedures, for example, and some of these other examples of how SQL injection attacks can be launched directly against a database server. Maybe spend 60 seconds on that.

**Amichai Shulman:** Yes, definitely. It turns out then that SQL injection can be used for privilege elevation in databases using direct access. So an unprivileged user with execute privileges on a stored procedure that is vulnerable to SQL injection can have SQL code running under the privileges of the owner of the stored procedure, which is usually a higher-privileged user. We have seen that a lot with Oracle stored procedures. But we have seen that with MSSQL server stored procedures. We have seen that with DB2 stored procedures. So this is an across-the-board issue, let's dedicate a separate podcast to it.

**Brian Contos:** Yes, I think that might be an interesting topic for our audience. Well, if anyone's interested in actually seeing what SQL injection looks like, sometimes these things are easier seen than heard, please go to the Imperva channel on YouTube. We have a number of videos dedicated to this topic, including basic SQL injection, blindfolded SQL injection, SQL injection with signature evasion, and a number of other attack examples for education purposes. So please be sure to check that out.

Amichai, it was a great pleasure having you on today's podcast.

**Amichai Shulman:** It's always a pleasure to be here.

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