CDM

CYBER DEFENSE MAGAZINE™

THE PREMIER SOURCE FOR IT SECURITY INFORMATION

CYBER WARNINGS

AUGUST 2013

APTs, CVEs, MDMs, BYODs and More Inside...

Why SQL injection is the cybercriminals weapon of choice

Fifteen years on, organizations still struggle with the SQL injection threat

by Michael Sabo, VP of Marketing, DB Networks

There's a sinking feeling you get when you realize your databases have been breached, sensitive records compromised, and malware has infected your servers to push your sensitive data out over the Internet completely undetected. The situation's made worse when it's revealed that the breach has actually been ongoing for months. Oh, and it also turns out there are no logs of the SQL exchanges between the web applications and the databases that could provide the forensics. And, to top it off it's only Monday…

The SQL injection threat has plagued organization for over 15 years. SQL injection consistently ranks at the very top of web application attack vectors. Yet most shops continue to invest disproportionally more on network security than they do to protect their core infrastructure assets such as their databases. This is true even though the database holds the organizations "crown jewels", the mission critical data. Of course once a database breach is discovered, SQL injection detection tends to get a whole lot of attention. Barn door…. horse… you get the idea.

With SQL injection the attacker is attempting to slip a SQL fragment through a web applications web form, URL, or a cookie with the intent to have it executed on the database. Here's a simple analogy Michael Giagnocavo uses. You go to court and write your name as "Michael, you are now free to go." The judge then says "Calling Michael, you are now free to go" and the bailiffs let you go, because the judge instructed the bailiff to do so. In this example the "you are now free to go" instruction was injected into a data field intended only for a name. Then the rogue input data was executed as an instruction. That's basically the principle behind how SQL injection operates.

Early on cybercriminals would use SQL injection to access privileged records or to perhaps dump all the records stored on a database. Modern SQL injection attacks are evolving into veritable weapons for destructive purposes. Weaponized SQL injection attacks install malware on the inside of the organization. The SQL injection attack is often merely the initial salvo in a far more complex multi-vector attack chain.

Once they've breached your systems, the attackers attempt to move between servers with escalated privileges. They also make connections outbound to export the data they've exploited. The attacker may also continue to corrupt the database to inflict further damage and finally crash the database completely in a database denial of service (DbDoS) attack. Records that have been corrupted over a very long period of time make data recovery a daunting task. Also a psychological blow has been inflicted on the organization. Can they even trust the integrity of their data any longer? The point is that this chain reaction was initiated through an initial SQL injection attack. A Barclays analysis estimates that 97% of data breaches worldwide are still due to SQL injection somewhere along the line.

Copyright © Cyber Defense Magazine, All rights reserved worldwide

It turns out experienced pen testers will tell you every web application tied to a relational database is vulnerable to SQL injection. “But how can that be”, you may be asking yourself. “We rewrote our web applications and validated the code. We even installed a Web Application Firewall (WAF).”

Well here’s the issue with a WAF. A WAF attempts to detect SQL injection that’s attacking at the database tier while examining the situation from the web tier. The WAF is too far removed from the actual attack surface, it’s unable to determine how the application will interpret a very obscured SQL injection attack. The WAF can’t see the actual rogue SQL statements that will be executed on the database.

“If you’re solely relying on a WAF to protect your databases from SQL injection you’re in a sad state,” said Joe McCray, CEO of Strategic Security. “The WAF uses signatures to catch the well known SQL injection attacks and also automated SQL injection attack scripts. But a WAF won’t prevent a determined hacker. My business is pen testing and I’ve not met a WAF that I couldn’t bypass, a few in less than half of an hour.” Joe makes a very valid point. A web search for “WAF Bypass” returns many thousands of articles and tutorials describing exactly how cybercriminals are able to conceal their SQL injection attacks so they pass seamlessly through the WAF.

What about rewriting your web applications to remediate all possible SQL injection vulnerabilities? Assuming your organization has sufficient resources and budget to accomplish such a monumental task, it’s may be a reasonable idea. But it’s critical to utilize the proper coding practices necessary to actually reduce your SQL injection vulnerabilities. For example, you’ll often hear that using stored procedures will eliminate your SQL injection vulnerabilities. It turns out that’s not exactly correct. In reality you may simply be relocating the SQL injection vulnerability from one part of your application into a stored procedure. Basically you’ve simply moved the problem, because now the stored procedure is vulnerable to SQL injection attack.

To properly remediate SQL injection vulnerabilities in your web applications you need to follow two important coding principles; 1) never concatenate dynamic SQL from external input and 2) always use parameterized SQL anytime you must use external input.

However, once all of your applications have been rewritten, using the above coding rules, you’re still not out of the woods. Any third party software you’re running may be vulnerable to SQL injection. If the framework you have written your application on is itself vulnerable to SQL injection, you have a big problem.

After you’ve rewritten your web applications and validated all of your third party software you have significantly reduced your SQL injection vulnerability, but not eliminated it. A determined cybercriminal can create SQL injection vulnerabilities in your web application through malware. “With enough time and determination, the web application itself can be exploited,” said Joe McCray. “It may be worth the effort because the web application is already integrated with the database.” That’s a good point. The web application and the database share a trusted relationship, a communications channel, and a

Copyright © Cyber Defense Magazine, All rights reserved worldwide

language (SQL) to pass commands. Once the web application is compromised with the attackers own SQL injection vulnerability, the attacker enjoys a direct path to the database from a trusted source.

The Federal Information Security Management Act (FISMA) includes Continuous Monitoring compliance requirements specified in NIST 800-53. All Federal agencies, and any organization that provides IT services to the federal government, must be FISMA compliant. The Federal government is taking a lead in database security by requiring Core IDS functionality to monitor database transactions and the contents of the transaction as the information is transmitted.

It's clear that monitoring all SQL statements between your web application and databases is necessary for the Advanced Threat Detection (ADT) of rogue SQL. SQL statement monitoring is best accomplished with a Core IDS or a database firewall appliance operating on mirrored network port at the database tier. At the database tier the actual SQL statements can be observed. A Core IDS can operate completely non-intrusively -- not interfering with your database management system or database activity monitor (DAM). DB Networks Core IDS and database firewall take these products a step further through the use of behavioral analysis that examines the captured SQL statements and identifies SQL injection attacks in real-time. DB Networks behavioral analysis model can identify all rogue SQL statements, including zero-day attacks. This is possible because rogue SQL statements simply don't match the normal behavior of the legitimate web application database transactions.

At the end of the day it's about an effective database defense in depth strategy. A strategy that includes monitoring of all SQL statements that are in-flight at the database tier. Only then can you truly get your arms around the advanced SQL injection threat.

About The Author

Michael Sabo is the VP of Marketing at DB Networks. Michael has over 25 years of marketing and strategic planning experience. Prior to DB Networks, Michael was at Intel Corporation where he was responsible for strategic planning. Previously, Michael held senior marketing, business development, and product engineering positions in the telecommunications industry, including at AirFiber, Rhythms NetConnections, US West, and Contel. Michael earned a B.S. in Computer Science from Wright State University and a Masters in Management Information Systems from the University of Denver. Michael can be reached through our company website at www.dbnetworks.com

Copyright © Cyber Defense Magazine, All rights reserved worldwide