TLP:CLEAR

Secure by Design Alert
Eliminating SQL Injection Vulnerabilities in Software

Malicious Cyber Actors Use SQL Injection Vulnerabilities to Compromise Systems

SQL injection—or SQLi—vulnerabilities remain a persistent class of defect in commercial software products.[1] Despite widespread knowledge and documentation of SQLi vulnerabilities over the past two decades, along with the availability of effective mitigations, software manufacturers have continued to develop products with this defect, which puts many customers at risk.[2] CISA and the FBI are releasing this Secure by Design Alert in response to a recent well-publicized malicious threat actor campaign that exploited SQLi defects in a managed file transfer application to target and compromise users of that application—impacting thousands of organizations. CISA and the FBI urge senior executives at technology manufacturers to mount a formal review of their code to determine its susceptibility to SQLi compromises and encourage all technology customers to ask their vendors whether they have conducted such a review. If they discover their code has vulnerabilities, senior executives should ensure their organizations' software developers immediately begin implementing mitigations to eliminate this entire class of defect from all current and future software products.[3] Building security into products from the beginning can eliminate SQLi vulnerabilities.

The software industry has known how to eliminate these defects at scale for decades. MySQL introduced prepared statements, which can eliminate SQL injection vulnerabilities, in 2004.[2]

Secure by Design Lessons to Learn

Secure by Design means that manufacturers design and build their products in a way that reasonably protects against malicious cyber actors successfully exploiting product defects. Incorporating this mitigation at the outset—beginning in the design phase and continuing through development, release, and updates—reduces the burden of cybersecurity on customers and risk to the public. Vulnerabilities like SQLi have been considered by others an 'unforgivable' vulnerability since at least 2007. Despite this finding, SQL vulnerabilities (such as CWE-89) are still a prevalent class of vulnerability. For example, CWE-89 is on top 25 lists for both the most dangerous and stubborn software weaknesses in 2023.[4]

What Are SQL Injection Vulnerabilities?

SQL injection vulnerabilities involve the insertion of user-supplied input directly into a SQL command, allowing threat actors to execute arbitrary queries (see CWE-89: Improper Neutralization of Special Elements used in an SQL Command ['SQL Injection']). SQLi vulnerabilities are caused by software developers' inattention to security best practices, resulting in the co-mingling of database queries and user-supplied data. The impact of successful SQLi exploitation can be devastating as it challenges the confidentiality, integrity, and availability of a database and its information. Specifically, SQLi vulnerabilities can allow malicious cyber actors to steal sensitive information, tamper with, delete, or render information unavailable in a database. SQL injections succeed because **software developers fail to treat user-supplied content as potentially malicious.**

---

[1] OWASP Foundation. "SQL Injection." n.d. https://owasp.org/www-community/attacks/SQL_Injection.
[2] In 2004, MySQL introduced a technique called "prepared statements" that separate database commands from untrustworthy data, thereby eliminating SQL injection vulnerabilities. See: MySQL. "Changes in release 4.1.3 (28 Jun 2004: Beta)." June 28, 2004. https://web.archive.org/web/20060422175612/http://dev.mysql.com/doc/refman/4.1/en/news-4-1-3.html.
[3] OWASP Foundation. "SQL Injection Prevention Cheat Sheet." n.d. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html.
[4] "2023 CWE Top 25 Most Dangerous and Stubborn Software Weaknesses in the CWE Top 25." MITRE's CWE Top 25, 2023. https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html,https://cwe.mitre.org/top25/archive/2023/2023_stubborn_weaknesses.html

TLP:CLEAR

How Can Software Manufacturers Prevent SQL Injections?

During the design and development of a software product, developers should use parameterized queries with prepared statements to separate SQL code from user-supplied data to prevent this class of vulnerability. This separation ensures the system treats user input as data and not executable code, thereby eliminating the risk of malicious user input being interpreted as a SQL statement. Software manufacturers should systemically eliminate SQLi vulnerabilities by enforcing the use of parametrized queries across their applications. **Note:** Some developers attempt to use input sanitization techniques to prevent SQLi vulnerabilities. While input sanitization may prevent some attacks, those techniques are brittle, difficult to enforce at scale, and frequently can be bypassed. Parameterized queries therefore better embody a secure by design approach.

CISA and the FBI recommend software manufacturers research the causes and widely known solutions to this predictable and commonly exploited vulnerability. Additionally, CISA and the FBI encourage manufacturers to review the following three principles of the joint guidance, Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software.

Principle 1: Take Ownership of Customer Security Outcomes

There are key security areas manufacturers should invest in to protect their customers as well as the public. These include providing safe building blocks for their software developers to ensure that a single developer error does not compromise data of millions of users. Software manufacturers should adopt the use of prepared statements with **parametrized queries** as a standard practice in software development. This should be enforced in their development environments via, for instance, development libraries that make the secure route the default one for developers and checks at the time of pull requests.

Additionally, senior executives at software manufacturers must take accountability for the security of their customers starting by conducting formal reviews of their code to determine their susceptibility to attack. A simple code review would reveal the prevalence of this well-known class of vulnerability—with clear and effective mitigations readily and publicly available. Manufacturers and developers should take ownership of securing products by defaulting to parametrized queries, thereby eliminating an entire class threat.

Principle 2: Embrace Radical Transparency and Accountability

Manufacturers should lead with transparency when disclosing product vulnerabilities. To that end, manufacturers should track the classes of vulnerability associated with their software and disclose them to their customers via the CVE program. Manufacturers should ensure that their CVE records are correct and complete. It is especially important that manufacturers supply an accurate CWE so the industry can track classes of software defect, not just individual CVEs, and customers can understand areas where a given vendor's development practices may require improvement.[5] They should also identify and document the root causes of those vulnerabilities and declare it a business goal to work toward eliminating entire classes of vulnerability.

Principle 3: Build Organizational Structure and Leadership to Achieve These Goals

Just as software and hardware manufacturing executives care about cost, features, and customer experience, they should also prioritize the security of their products. Leaders must consider the full picture: that customers, our economy, and our national security are currently bearing the brunt of business decisions to not build security into their products—as the threat actor campaign described earlier in this Alert clearly reflects. Moreover, directing the business toward secure by design software development often reduces financial and productivity costs as well as complexity. Leaders should make the appropriate investments and develop the right incentive structures that promote security as a stated business goal.

[5] Common Weakness Enumeration (CWE) classification identifies classes of software/hardware weaknesses (including vulnerabilities and defects); Common Vulnerabilities and Exposures (CVE) classification identifies and labels unique vulnerabilities in specific software/hardware products.

Leaders should highlight the importance of rooting out entire classes of vulnerabilities rather than addressing them on a case-by-case basis. Additionally, leaders should establish organizational structures that prioritize proactive measures, such as adopting secure coding practices like parametrized queries, to create enduring security and reduce reliance on reactive responses. Senior executives should also ensure their organization conducts reviews to detect common and well-known vulnerabilities, like SQLi, to determine their susceptibility, and implement the existing effective and documented mitigations. These reviews should be continually conducted to root out classes of vulnerability, as some vulnerabilities may change or develop over time.

Action Item for Software Manufacturers

Although this Secure by Design Alert focuses on approaches to mitigate SQL injections as a class of defect, it is just one part of a more comprehensive set of secure by design practices. To protect their customers from a wide range of malicious cyber activity, manufacturers should fully implement the principles and practices touched upon in this alert by reviewing Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software. Further, CISA and the FBI urge manufacturers to publish their own secure by design roadmap to demonstrate that they are not simply implementing tactical controls but are strategically rethinking their responsibility in keeping customers safe.