Database Security Request for Proposal: Defending both Internal and Public Facing Databases against Known Vulnerabilities

Emergent Research Forum papers

Thomas Arthur Talmadge
University of Maryland Global Campus
rickshaw98@hotmail.com

Executive Summary:

Open Web Application Security Project (OWASP) vulnerabilities are important for business leaders to know and understand. The issues are well known and understood, as are the mitigation and fix actions to harden database vulnerabilities. Database vulnerabilities remain persistent because of the proliferation of databases in many aspects of data storage and web applications, where the perceived value of the information in the database is high and the attack mechanism is easy. There are actually several database vulnerabilities, depending on the database employment, which expands the targets and, thus, the vulnerabilities. Individual databases holding organizational data have vulnerabilities, as do web application databases— with web application SQL injection being the most well-known vulnerability.

1) The different database vulnerabilities and attacks can be analyzed through the differentiation between information security and cybersecurity vulnerabilities to better understand the taxonomy and operational semantics for targeted mitigation.

2) The root cause of database vulnerabilities remains poor design and coding. Ultimately, there are multiple SQL injection attack detection methodologies being researched, but prevention through proper design and coding is the best defense.

3) Proper training and education are a key factor in combating these vulnerabilities.

Thus, while a defense in depth is required, when implementing a database solution one of the most important aspects is creating and implementing a robust functional database vulnerability test plan within the request for proposal to ensure that the root cause is mitigated.

Keywords

Cybersecurity, Software Testing, Database, Request for Proposal (RFP), SQL.

Introduction:

The ranking of the injection vulnerability, of which SQL Database injection is one of the most common targeted vulnerabilities, has remained at the top of the Open Web Application Security Project (OWASP) list of "The Ten Most Critical Web Application Security Risks" since 2010, and on the regularly updated list since at least 2004 (OWASP, 2004, 2010, 2017). Whitman and Mattord (2017) refer to SQL Injection vulnerabilities as one of the "Deadly Application Sins" because this software vulnerability makes software so vulnerable in the hostile internet environment, yet are rated as "easy" to discover when examining code. These vulnerabilities exist because SQL databases provide extensive backside support to website applications, which are then tricked into providing additional data to the attacker. But, these vulnerabilities also exist on internal databases that are commonly used to store sensitive organizational data. More and more database technology is being implemented for the gathering, sharing, and analysis of internal organizational data- which rely on Database Management Systems (DBMS) (Fuller et al., 2017). However, given the internal nature of these organizational DBMS, the threats and vulnerabilities are different. Whenever the perceived value of information increases or the simplicity of the attack decreases, there will be additional attacks (Horner and Hyslip, 2017).

There are multiple examples of the successful database attacks and the high costs of these successful attacks to the victim (Dorai and Kannan, 2011; Perkel, 2010; Horner and Hyslip, 2017). Because of the different attack mechanisms and vectors between Web Application database attacks and internal data storage attacks, mitigation techniques are different. Whitman and Mattord (2017) define information security as: 'the safeguarding of information asset confidentiality, integrity, and availability, whether in storage, processing, or transmission, via the application of policy, education, training and awareness, and technology.' Thus, the Confidentiality, Integrity, and Accessibility (CIA) model for information security is our working model. However, Whitman and Mattord (2017) accept that, at present, the term "cybersecurity" remains ambiguous with a final definition to be agreed at a later date.

Von Solms and van Niekert (2013) build on the CIA model from Whitman and Mattord and define information security as the preserving the CIA of information, but then expand the discussion and specifically define Information and Communication Technology (ICT) security and cybersecurity with additional taxonomy:

- ICT Security: Protecting the technology-based systems that store and process information.
- Cybersecurity: Protecting information and non-information assets from risks stemming from interaction with cyberspace.

Thus, von Solms and van Niekert (2013) include the requirement for interaction with cyberspace, but also includes non-information assets (i.e. cyberbullying, organizational reputation, digital media, cyberterrorism, etc.). It is primarily the differentiation of requiring interaction with cyberspace that is applicable for this discussion. This is an important divergence classification, as it relates directly to the specific database vulnerabilities and gives a useful way to discuss internal database information security and public facing database cybersecurity more granularly.

Database Information Security Vulnerabilities:

There are five basic threats to the confidentiality, integrity, and accessibility of database information security (Vijayan and Madhusudhanan, 2017):

1- Excessive Privilege Abuse: Granted access above requirements and using this access for malicious purposes.

2- Legitimate Privilege Abuse: Legitimate users, but unauthorized purposes.

3- Platform Vulnerabilities: Underlying operating system vulnerabilities.

4- SQL Injection: Taking advantage of poor design/coding to insert unauthorized database statements.

5- Denial of Service: Data access denied to users.

Authorized users taking advantage of elevated or legitimate privileges for malicious purposes are the most common issue for database information security, with 80% of attacks on organizational data coming from employees/ex-employees (Vijayan and Madhusudhanan, 2017). Granting excessive privileges simplifies administration and cuts down on requirements for IT professionals to granularly track and update permissions and access, but also creates major threat vulnerabilities. This is a challenging aspect of organizational information security because it relies on the user intent to act maliciously, although not specifically a database issue.

Guarnieri, Marinovic, and Basin (2016) argue that there is currently no specific taxonomy and model that can be used to certify security of database information security. Thus, creating both confidentiality and integrity attack models to be tested against system models under test to create "provably secure" database designs. This would address: 4. SQL Injection and 5. Denial of service threats. However, the primary defense of the greatest CIA vulnerabilities of database information security is a robust mechanism for access control of important database information assets and a process for granting access/privileges -- but also for revoking access/privileges in a timely manner (Vijayan and Madhusudhanan, 2017). Further, there is also a risk of inadvertent data leaks or incidents due to user's lack of knowledge or understanding. Perkel (2010) expands the discussion on vulnerabilities that may be unknowingly introduced (i.e. a well-meaning professor moving data to a new database, without realizing the underlying vulnerabilities). Thus, the importance of addressing the application of policy, education, and training and awareness for to combat the most common database information security vulnerabilities.

Database Cybersecurity:

As discussed, while the primary solution for SQL database vulnerability is proper coding to avoid creating vulnerabilities in the first place. Basically, understanding that the database may be targeted, thus all possible inputs must be understood and planned for vice limiting the database design and coding to only accepted and proper queries (Patwari and Bhurani, 2012). However, with the continuing SQL database injection attacks after over 19 years of understanding the attack, there is extensive research into creating a second layer of defense via detection techniques. Database cybersecurity vulnerabilities predominately take advantage of poorly coded SQL databases, via SQL database injection attacks, to transmit commands directly to a database to gain access access and copy, edit, modify, etc. data as desired with a malicious intent (Horner and Hyslip, 2017).

The prevalence of databases providing backside functionality for websites make this a simple attack that is well understood. In the case of database attacks, and specifically SQL injection attacks, the perceived value of the information has increased while the ease of attack has decreased—which has led to this being one of the most common malicious attack types. There are even automated applications designed to search variations of website addresses and returns the associated data, thus even low skilled attackers with malicious intent can take advantage of the SQL injection vulnerability (Horner and Hyslip, 2017). The SQL injection is best understood by analyzing the process via the three tiered web application model (Kharche et al., 2015):

- Tier 1: Presentation Tier with user interface.
- Tier 2: Server Script Tier sits between the user interface and the database tier for processing.
- Tier 3: Database Tier stores and manages the processed data

Specifically, the database tier manages access of authorized users. Thus, frameworks for using data mining of database logs and pattern matching algorithms analyzing user SQL queries have been proposed and tested (Bertino and Ghinita, 2011). Thus, efforts to address SQL database vulnerabilities at both the Tier 1: Presentation and the Tier 3: Database layers. Additionally, in line with intrusion detection methodologies, anomaly-based detection and mitigation methodologies that focuses on SQL and web access query anomaly detection methodology (Tier 1: Presentation) (Vigna et al, 2009). As with intrusion detection technologies, anomaly-based SQL detection methodologies suffer from excessively high false positive results.

Thus, there are multiple SQL injection attack detection methodologies that are being proposed that focus on the different web access tiers, but suffer from different levels of successful identification of actual attacks and false positive attacks. This research is important because it shows the level of effort to work around known SQL vulnerabilities and shows a general acceptance of the underlying root cause of database vulnerabilities—poor coding and design. Dorai and Kannan (2011) recommend addressing the SQL vulnerabilities by introducing web application firewalls at the web server level. The requirement for defense in depth for database cybersecurity is, thus, well established—whether through firewalls applications or database specific SQL attack detection methodologies.

There are numerous forms of SQL injection vulnerabilities, all should be proactively avoided through proper coding and design, but must be understood to be addressed (Dorai and Kannan, 2011):

- Incorrectly filtered escape characters
- Incorrect type handling

- Vulnerabilities inside the database server
- Blind SQL Injection

Each of these attacks is unique and should be included specifically in any mitigation strategy. Horner and Hyslip (2017) state that the best way to overcome these vulnerabilities is through whitelisting, accepting only predefined characters, prepared statements, to properly structure data requests, and stored procedures, to keep SQL code within the SQL database. Thus, coding solutions.

Conclusion:

At first glance it appears that a vulnerability which has been understood since 1998, such as the database attacks, should have been conclusively mitigated years ago. However, the multiple injection vulnerabilities, web application tiers, and other database vulnerabilities with both information security and cybersecurity discussed here show that this is a broad and continually changing challenge. Further, databases are frequently customized, and with each customization comes another possible vulnerability vector. The primary mitigation techniques for both database information security and cybersecurity are designing security into the database and ensuring proper coding for security. But, a defense in depth is required for database cybersecurity and additional policy and database management processes are required for database information security vulnerabilities. Horner and Hyslip (2017) identify properly trained, security minded programmers as a primary database cybersecurity countermeasure. They further point out that the cost of these countermeasures is far less than the cost of cleaning up the attack ramifications.

While Tanenbaum (2016) discusses using the Request for Proposal (RFP) as a primary IT security defense for healthcare specific IT service procurement, this is applicable to all organizations with sensitive data. Thus, using this methodology of codifying security into RFPs for IT services, whenever a database is required, either from an internal or external source, this initial design phase should include the specific test requirements for a hardened database solution to mitigate known vulnerabilities. A detailed test plan requirement within the RFP, which addresses both information security and cybersecurity related database issues and known SQL vulnerabilities, is required.

REFERENCES:

Bertino, E., & Ghinita, G. (2011). Towards mechanisms for detection and prevention of data exfiltration by insiders: keynote talk paper. In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (pp. 10-19). ACM.

Dorai, R., & Kannan, V. (2011). SQL injection-database attack revolution and prevention. J. Int'l Com. L. & Tech., 6, 224.

Fuller, B., Varia, M., Yerukhimovich, A., Shen, E., Hamlin, A., Gadepally, V., ... & Cunningham, R. K. (2017). SoK: Cryptographically Protected Database Search. arXiv preprint arXiv:1703.02014.

Guarnieri, M., Marinovic, S., & Basin, D. (2016, March). Strong and provably secure database access control. In Security and Privacy (EuroS&P), 2016 IEEE European Symposium on (pp. 163-178). IEEE.

Horner, M., & Hyslip, T. (2017). SQL Injection: The Longest Running Sequel in Programming History. Journal of Digital Forensics, Security and Law, 12(2), 10.

Kharche, S., Gohad, K., & Ambetkar, B. (2015). Preventing SQL Injection attack using pattern matching algorithm. arXiv preprint arXiv:1504.06920.

OWASP, T. (2010). Top 10- 2010. The Ten Most Critical Web Application Security Risks.

OWASP, T. (2013). Top 10- 2013. The Ten Most Critical Web Application Security Risks.

OWASP, T. (2013). Top 10- 2017. The Ten Most Critical Web Application Security Risks.

Patwari, N., & Bhurani, P. (2012). Framework of SQL Injection Attack. arXiv preprint arXiv:1207.1542.

Perkel, J. (2010). Cybersecurity: how safe are your data?. Nature News, 464(7293), 1260-1261.

Tanenbaum, W. A. (2016). IT Systems Put Security into Health Care Cybersecurity. Journal Of Health Care Compliance, 18(4), 21-26.

Von Solms, R., & Van Niekerk, J. (2013). From information security to cyber security. computers & security, 38, 97-102.

Vigna, G., Valeur, F., Balzarotti, D., Robertson, W., Kruegel, C., & Kirda, E. (2009). Reducing errors in the anomaly-based detection of web-based attacks through the combined analysis of web requests and SQL queries. Journal of Computer Security, 17(3), 305-329.

Vijayan, S., & Madhusudhanan, S. (2017). Prevention of Insider Attack Against Database Access Control Mechanism. Journal Of Network & Information Security, 5(1), 20-23

Whitman, M., & Mattord, H. (2017). Management of information security. Nelson Education.