A new paradigm is among us beyond the “Cloud” or mobile devices, and that is with the use of Distributed Database Systems (DDS). Though they have been around for the better part of a decade, the popularity of this technology has grown to the point where information systems professionals should become familiar with not only its architecture and use, but how to assess their risk posture, and how to secure them. This article will provide an overview of this technology, an idea of the security vulnerabilities that can be found within this technology, and how to remediate those vulnerabilities.

Once thought of as a technology solely for academic purposes, DDS systems are now reaching critical mass in industry as leading technology service providers (e.g. Facebook, Amazon, Adobe, Google, Twitter) leverage them, and as individuals and companies consume those provider offerings. The genesis for this adoption is the open-source nature of these platforms, leading to cost savings experienced from not having to invest in traditional relational database software licensing and/or local hardware. Also, organizations can expect to experience enhanced elasticity, modularity and interoperability between DDS platforms and Web 2.0 technologies, such as Ruby on Rails or web services/Service-Oriented Architecture (SOA). DDS systems are here to stay and it is important to understand how they defer from a Relational Database Management System (RDBMS).

RDBMS systems have been around awhile in the industry and they will not go away any time soon; however, DDS systems are becoming the preferred database architecture for organizations leveraging Web 2.0 for the reasons mentioned above. DDS systems are distributed
in nature and that is the major difference as data in a DDS is distributed across multiple computers. Furthermore, data is Created, Read, Updated and/or Deleted (CRUD) from/to a DDS through an Application Programming Interface (API) call versus a database connection (e.g. Open Database Connectivity: ODBC, Java Database Connectivity: JDBC). A DDS also differs from a RDBMS in the way that it treats data as tables in a DDS are referred to as domains. Also, in a DDS system Data Definition Language (DDL) or metadata is not as easily queried. It should also be noted that most DDS systems have moved away from using Structured Query Language (SQL) for Data Manipulation Language (DML) calls, specifically leveraging a DML platform called NoSQL. Finally, a DDS system requires an operating API service to be running versus a database server instance, which often leads to a considerably lower Operating Expense (OpEx).

By leveraging an API service DDS users can save on Capital Expenses (CapEx), OpEx and easily build applications leveraging this technology, and this is what technology service providers are counting on. Cloud Service Providers (CSPs) like Amazon, Google and others now have DDS offerings available and companies are now leveraging them for what is predominantly their Web 2.0 applications that are hosted in the cloud. The most prevalent DDS systems include: CouchDB, FlockDB, Hive, MongoDB, RavenDB and SimpleDB. Though DDS systems are becoming increasingly popular they have their vulnerabilities.

The largest vulnerability found in a DDS system is its distributed nature, which in today’s highly regulated and litigious society can be a burden too high to bear for most organizations. Data that is distributed across multiple jurisdictions is a problem in itself, and this issue can be exacerbated when the location of a specific record cannot be determined for audit, electronic discovery (eDiscovery), or forensic purposes. There are several other vulnerabilities found within a DDS system, and these include: a lack of uniform standards across DDS systems.
affecting portability, a lack of supported native encryption/hashing methods, limited consumer-level logging/monitoring support, limited support for dynamic application security testing (DAST) by vulnerability assessment tools, and a belief that NoSQL DDS implementations are more secure than RDBMS systems.

DDS systems leveraging NoSQL are still susceptible to input validation vulnerabilities, such as SQL-Injection. Therefore, it is important that the source code check for input validation at both the client and server-side as most DAST tools will not pick up the nuances of SQL-Injection-based vulnerabilities for a DDS. Organizations that are obligated to comply with PCI DSS, HIPAA, etc. should also be cognizant of the fact that DDS systems provide limited logging for auditing and accounting purposes as the logs are built predominantly for debugging. Encryption/hashing for any critical data should be done prior to insertion into a DDS; however, an organization should be cognizant of the stored encrypted value for querying purposes. Finally, as there is no standard for DDS systems one should know that there will be limited portability of the implementation between DDS providers.

Beyond the known portability limitations, there are additional vulnerabilities that can be found from a process standpoint. These are commonly from a lack of a documented and/or implemented security review tasks within the Software Development Lifecycle (SDLC), or due to a lack of a separation/segregation of duties. A large part of this problem stems from the ability of a developer to sidestep the traditional role and/or need of a Database Administrator (DBA) to design and build a logical/physical RDBMS. The cause of this is the reliance on the API calls versus a pre-built database to connect to via ODBC/JDBC. Additionally, as small start-ups are the most prevalent users of this technology there are often control gaps from a security standpoint. With their often limited budgets and staff these entities often lack a
documented and secure software development lifecycle (SDLC), leading to increased risk. However, with some education and awareness it has been found that these vulnerabilities can be remediated relatively quickly.

Whether an organization adapts its processes to remediate known vulnerabilities, or it implements technological solutions, there are ways to mitigate the risks found in implementing a DSS system. With that said, the largest remediation strategy should be to do a risk assessment before using a DDS. Once completed, a go or no-go decision can be made to determine the risk appetite for a DDS, or for using the cloud at all for that matter. This assessment should also decide what data, if any, may be stored in a DDS. For many, that data may only be log data from another server/system. Such a solution would be used as a temporary/intermediate data store for log data until it can be analyzed. For those organizations that are looking to leverage technical controls/safeguards for additional protection they may want to consider implementing a Web Application Firewall (WAF) and/or an XML Firewall as a preventive control while using a DSS. In the end, an organization needs to do a Cost-Benefit Analysis (CBA) on using a DSS system while incorporating the necessary controls to safeguard the data.

By completing a CBA, conducting a risk assessment, and by understanding the vulnerabilities found within this platform; an organization can make an informed decision on whether or not to use a DSS system. This decision is more important than ever now as organizations must realize that data is the lifeblood of their organization, and as organizations collect data in staggering quantities it is crucial that they keep their options open for the best system to use when storing that data. As DSS systems come closer to hitting critical mass it is important to know how they work, how they can be used within the organization, how they differ from RDBMS platforms, and who is best suited to use them for collecting and processing data.
References


Biography

Steve Markey is the Principal of nControl, a consulting firm based in Philadelphia. He is also an Adjunct Professor, and he is the current president of the Delaware Valley (Greater Philadelphia) chapter of the Cloud Security Alliance (CSA). Steve holds multiple certifications, degrees and has over ten (10) years of experience in the technology sector. He frequently presents on: information security, information privacy, cloud computing, project management, eDiscovery and information governance.