Critical Evaluation of MEAN and LAMP  
as Technology Stacks

Introduction

Choosing the right technology stack when starting a new software project is an important decision which can lead to success or failure of the project. Changing the stack becomes increasingly difficult and expensive the more development is made on the project (Martinsson & Svanqvist, 2022). Websites were historically made only using HTML, CSS and JavaScript, but these are limited in their capabilities and no longer suitable for business websites (Aggarwal & Verma, 2018). Technology stacks are sets of technologies used to develop applications which most commonly include, but not limited to, programming languages, frameworks, libraries and databases (Martinsson & Svanqvist, 2022). Different technology stacks have different technologies and can increase or decrease the difficulty in developing certain apps (Martinsson & Svanqvist, 2022). This paper will compare the two technology stacks MEAN, (MongoDB, Express.js, AngularJS and Node.js) and LAMP (Linux, Apache, MySQL and PHP) (Karanjit, 2016).

MEAN as a Technology Stack

The MEAN stack uses MongoDB as the NoSQL database, Express as a framework for server-side scripting, Angular for client-side scripting, and Node as the server built in JavaScript (Martinsson & Svanqvist, 2022). Mean is the most popular full stack JavaScript framework used in industry (Adhikari, 2016). All components of MEAN are open source, relatively lightweight and easy to use (Karanjit, 2016). MEAN only uses JavaScript as its programming language for the server, client-side scripting and database connectivity, making connectivity between programmes much easier and provides the added benefit of every developer using is able to understand (Karanjit, 2016).

MEAN uses MongoDB as a NoSQL database management system, a non-relational system (Damodaran et al, 2016). MongoDB has proven to be effective in the development of new applications and its ability to replace existing relational infrastructure (Damodaran et al, 2016). Based on experimental data conducted by Adhikari (2016), MEAN stack is one of the best frameworks when developing web applications. It has proven very effective at dealing with large amounts of data due to the structure of the non-relational system (Adhikari, 2016). Security has proven to be a drawback for MongoDB and therefore is not suitable for applications which require a high level of security, and also lacks the robustness of pre-existing technologies (Adhikari, 2016).

MEAN uses NodeJS which runs quickly, is being continuously expanded on, has the ability to support the development of real-time applications and enhances overall productivity (Aggarwal & Verma, 2018). NodeJS is limited by a poor performance with large scale applications, CPU intensive queries, does not perform well with relational databases and has some inconsistency issues in the API (Aggarwal & Verma, 2018).

LAMP as a Technology Stack

The LAMP stack uses Linux as the operating system, Apache as the web server, MySQL as the database system and PHP for server-side scripting (Martinsson & Svanqvist, 2022). LAMP has been a successfully implemented framework for decades and continues to be used widely in industry (Karanjit, 2016). LAMP is predominantly used in the development of websites and web applications (Singh, 2022).

LAMP is run on the Linux operating system but does have a version called WAMP which runs on windows (Singh et al, 2022). Linux is a fast system with minimal hardware requirements, a high level of security and access to remote administration (Rosebrock & Filson, 2006). Another advantage using Linux is it can run with or without a graphical user interface (Rosebrock, & Filson, 2006). Linux is limited when it comes to diagnosing and fixing problems (Denton & Peace, 2003).

LAMP uses MySQL as the database management, a relational system (Rosebrock & Filson, 2006). MySQL is able to store a wide variety of data types due to its easy connection with PHP and therefore is able to carry out a vast number of features which includes, but not limited to, data replication, query limiting, table locking, multiple databases, user accounts, persistent connections and many more (Rosebrock & Filson, 2006). MySQL is advantageous due to its low cost, ease of use and high connectivity (Denton & Peace, 2003). The use of PHP is efficient due to it being the fastest server-side scripting program currently used (Rosebrock & Filson, 2006).

Conclusion

In conclusion, MEAN and LAMP run on different operating systems with different database management systems. MEAN displays live data without requests from client to server whereas LAMP sends continuous requests to the server for updating the data (Karanjit, 2016). MEAN uses a non-relational database which can store a variety of data types in Json-like files whereas LAMP uses a relational database structure which is limited as each item can only contain single attributes (Damodaran et al, 2016). Non-relational databases are limited as it cannot create key-value pairs and therefore unable to use relationships between data items, which is not a problem in relational databases due to the use of PHP (Damodaran et al, 2016). Each is efficient at producing different types of applications and is important to consider the right stack for the right job.

Evaluating Various Approaches to Securing  
User Credentials in Web Applications

Introduction

Almost modern web applications, including web browsers, store sensitive data about individuals for various reasons. The most common form of data stored is password protected logins which authenticate a user’s identity (Ahmed & Khay, 2017). Passwords are used as the form of protection against exploitation, but storing this data as plaintext has proven to be not secure and easily exploited (Hossain *et al*, 2020). Data needs to travel between applications which can pose a security threat (Clark & Van Oorschot, 2013). To increase data protection further, supplementary forms of authentication can be used such as multi-factor authentication and biometric authentication.

Storage Credentials

Storing passwords as a plaintext has proven to be a weak form of protection (Ahmed & Khay, 2017). Encryption is a strong form of securing stored data by making the data unintelligible and stored in this form which can be used to protect this form of data (Ciriani *et al*, 2007). Encryption is used frequently in data protection but is not secure enough to use by itself and should always be paired with a method such as hashing (Newman & Shi, 2017). Hashing is where an algorithm such as MD5, SHA1, bcrypt or many others, is able to change information of any size into a string of information of a settled size (Kale & Dhamdhere, 2018). Hashing is irreversible due to the one-way function being applied, therefore obtaining the original password is impossible making it a very secure method of storing the data (HJossain *et al*, 2020). Hashing is an easy method for developers to employ for security and is usable for web applications and web browsers (Ahmed & Khay, 2017). Not all hashing algorithms are resistant to brute force attacks and the hashed password requires safe storage (Kale & Dhamdhere, 2018).

Transmission of Credentials

For applications to function properly, all data types need to be moved between systems which poses a risk for the data to be intercepted (Clark & Van Oorschot, 2013). HTTPS is a method used to secure the data while it’s being transferred back and forth by encrypting the data before transmission (Clark & Van Oorschot, 2013). This encryption is applied for all communications and is a method that prevents eavesdropping from a middle-man attack (Fung & Cheung, 2010). HTTPS is limited by the need for encryption and decryption which adds computational overhead, slowing the performance speed of the web application (Rodriguez, 2018). HTTPS also requires regular updates and renewals of SSL/TLS certificates (Fung & Cheung, 2010). Secure authentication protocols such as OAuth and OpenID Connect can be used for protection of data being transmitted where data is integrated with identity providers and proper validation is implemented (Fett *et al*, 2016). This is advantageous as it enhances security by offloading authentication to trusted providers but is limited by requiring libraries and dependencies to be up-to-date and is susceptible to security flaws if implemented incorrectly (Fett *et al*, 2016).

Supplementary Forms of Authentication

The storage of data is important for verification of individual users, which is why password protected logins are used. To further secure data beyond encryption and hashing, a number of supplementary forms of authentication can be used. The first is multi-factor authentication where additional data is required to verify individuals and uses combinations of “something you know”, “something you have”, “something you are” and “somewhere you are” / “someone you know” (Abhishjek *et al*, 2013). This provides a stronger level of protection by requiring more sensitive individual data and is therefore stronger than standard single-factor authentication (Abhishjek *et al*, 2013). (Ometov *et al*, 2018). Whilst this does increase security, user experience is less favourable as it is more inconvenient, and is vulnerable to phishing (Ometov *et al*, 2018).

Biometric authentication is becoming an increasingly popular method of security with the use of thumbprints, face scanners, iris scanners and ear geometry, with increasing popularity in voice recognition, keystroke dynamics and gait analysis (Weaver, 2006). Biometric data is secure as it only has to be stored and not transmitted, removing the threat of middle-man attacks, and is an easy and convenient method of security (Weaver, 2006). Biometric authentication is limited for users unable to provide such data and requires fallback methods, and there are a number of privacy concerns over obtaining user consent for such data (Bhattacharyya *et al*, 2009).

Conclusion

Whilst there are many forms of data protection for online web applications, the most secure method is to implement several of these methods. Although this increases security, it vastly increases workload for developers and slows the web applications considerably the more methods applied.

References

Abhishek, K., Roshan, S., Kumar, P., & Ranjan, R. (2013). A comprehensive study on multifactor authentication schemes. *Advances in Computing and Information Technology: Proceedings of the Second International Conference on Advances in Computing and Information Technology,* 561-568

Adhikari, A. (2016). Full Stack JavaScript: Web Application Development with MEAN. *Springer Link*.

Ahmed, A. A., & Khay, L. M. (2017). Securing user credentials in web browser: Revier and suggestion. *IEEE Explore*, 67-71

Aggarwal, S., & Verma, J. (2018). Comparative analysis of MEAN stack and MERN stack. *International Journal of Recent Research Aspects*, 5, 127-132

Bhattacharyya, D., Ranjan, R., Alisherov, F., & Choi, M. (2009). Biometric authentication: A review. *International Journal of u-and e-Service, Science and Technology*, *2*(3), 13-28

Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., & Samarati, P. (2007). Fragmentation and encryption to enforce privacy in data storage. *Computer Security–ESORICS 2007: 12th European Symposium On Research In Computer Security,* 171-186

Clark, J., & Van Oorschot, P. C. (2013). SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. *IEEE Symposium on Security and Privacy*, 511-525

Damodaran, D., Salim, Shirin, S., & Vargese, S. M. (2016). Performance Evaluation of MySQL and MongoDB Databases. *International Journal on Cybernetics & Informatics*, 5(2), 387-394

Denton, J. W., & Peace, G. A. (2003). Selection and Use of MySQL in a Database Management Course. *Journal of International Education*, 14(4), 401-407

Fett, D., Küsters, R., & Schmitz, G. (2016). A comprehensive formal security analysis of OAuth 2.0. *Conference on computer and communications security*, 1204-1215

Fung, A. P., & Cheung, K. W. (2010). HTTPSLock: Enforcing HTTPS in unmodified browsers with cached JavaScript. In *2010 Fourth International Conference on Network and System Security*, 269-274

Hossain, A., Rahaman, H., Jamil, A., & Khan, M. A. (2020). An Algorithm for Securing User Credentials by Combining Encryption and Hashing Method. *International Journal of Electrical Engineering and Applied Sciences*, 3(2), 35-42

Kale, A. M., & Dhamdhere, S. (2018). Survey paper on different type of hashing algorithm. *International Journal of Advance Scientific Research Algorithm*, *3*(2)

Karanjit, A. (2016). Culminating Projects in Computer Science and Information Technology. *St Cloud State University*, 11

Martinsson, H., & Svanqvist, V. (2022). Technology stack selection: Guidelines for organisations with multiple development teams. *Digitala Vetenskapliga Arkivet*, 64

Newman, M., & Shi, Y. (2017). Limitations on transversal computation through quantum homomorphic encryption. *arXiv preprint*

Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T., & Koucheryavy, Y. (2018). Multi-factor authentication: A survey. *Cryptography*, *2*(1), 1.

Rodriguez, M. (2018). HTTPS everywhere: Industry trends and the need for encryption. *Serials Review*, *44*(2), 131-137.

Rosebrock, E., & Filson, E. (2006). Setting up LAMP: Getting Linux, Apache, MySQL, and PHP Working Together. *John Wiley & Sons*.

Singh, G., Javed, M., & Dhaliwal, B. K. (2022). Full Stack Web Development: Vision, Challenges and Future Scope. *International Research Journal of Engineering and Technology*, 09(44), 3083-2089

Weaver, A. C. (2006). Biometric authentication. *Computer*, *39*(2), 96-97