Bulletin of Electrical Engineering and Informatics
Vol. 9, No. 2, April 2020, pp. 565~572
ISSN: 2302-9285, DOI: 10.11591/eei.v9i2.2064

Security issues and framework of electronic medical record: A review

Jibril Adamu, Raseeda Hamzah, Marshima Mohd Rosli
Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia

Article Info

Article history:

Received Oct 30, 2019
Revised Dec 28, 2019
Accepted Feb 11, 2020

Keywords:

CodeIgniter security
CSRF
EMR security issues
Laravel security
SQL injection
Symfony security
Top vulnerabilities
XSS

ABSTRACT

The electronic medical record has been more widely accepted due to its unarguable benefits when compared to a paper-based system. As electronic medical record becomes more popular, this raises many security threats against the systems. Common security vulnerabilities, such as weak authentication, cross-site scripting, SQL injection, and cross-site request forgery had been identified in the electronic medical record systems. To achieve the goals of using EMR, attaining security and privacy is extremely important. This study aims to propose a web framework with inbuilt security features that will prevent the common security vulnerabilities in the electronic medical record. The security features of the three most popular and powerful PHP frameworks Laravel, CodeIgniter, and Symfony were reviewed and compared. Based on the results, Laravel is equipped with the security features that electronic medical record currently required. This paper provides descriptions of the proposed conceptual framework that can be adapted to implement secure EMR systems.

Corresponding Author:

Jibril Adamu,
Faculty of Computer and Mathematical Sciences,
Universiti Teknologi MARA,
40450 Shah Alam, Selangor, Malaysia.
Email: Jibrilsonghai@gmail.com

1. INTRODUCTION

E-health is a term commonly used in the medical environment to refer to the use of information and communication technologies in healthcare. There are different types of e-health systems. However, the two major e-health systems are Electronic Medical Record (EMR) and Electronic Health Record (EHR) [1]. There is frequent interchangeable use of the terms "Electronic Health Record" and "Electronic Medical Record" [2]. However, these terms portray entirely distinct ideas, both of which are essential to the achievement of local, regional and national objectives of improving patient safety, improving patient care quality and effectiveness, and reducing the cost of delivering healthcare [2, 3]. Some authors have chosen to refer to the EMR as a patient health record from a variety of sources related to patient care, diagnostic procedures, laboratory tests, medical history, drugs, and allergic conditions that can be retrieved from different sites within a single healthcare organization [4], whereas the EHR provides the opportunity to exchange patient medical data with other medical professionals and monitors patient medical information during the individual's different medical treatments. In order to handle EMR and EHR, secure and effective storage, collection and transformation of medical information must be embraced [5].

Hospitals implementation of e-health systems has increased rapidly over the previous several years. As the healthcare industry was digitized to keep abreast, the healthcare industry was unable to acquire electronic security features at the same speed, leading to vulnerabilities in e-health systems [6]. Violations of

health care data are likely to improve with the increased use of e-health systems [7, 8]. This can be ascribed to security approaches implemented by the healthcare industry that, while robust, are often less advanced compared to other sectors, such as the financial industry. Large databases comprising of medical reports are beneficial to cybercriminals as medical reports include "social security and loan card numbers, patient demographics, addresses, insurance identification numbers" as well as other health information [8]. Such data could be used for creating fake identifiers for the purchase of medications or for filing fake insurance claims. Securing medical data is therefore highly necessary [9].

Appropriate security policies can lead to reputational advantages, cost reductions, and lower reaction times for incidents. Failure to implement adequate security can, however, affect the competitive position of an organization [10]. The EMR system must implement and enforce sophisticated security policies to deny access to confidential data and activities such as secret tokens and diagnostic code against common security vulnerabilities, poor authentication, cross-site request forgery, cross-site scripting, and SQL injection [11]. Frameworks tend to offer many tools to assist in avoiding the common security vulnerabilities listed above. They may also provide an implementation for some security features, such as authentication and authorization alternatives [12].

The objective of this paper is to review and compare the security features of some popular and powerful PHP frameworks Laravel, CodeIgniter, and Symfony, leading to choosing the appropriate framework that ensures a much more secure EMR. The web application (e.g., the EMR) developed with the PHP framework will be much more dependable and secure [13]. The paper is organized as follows: Section 1 contains the introduction of the topic which includes problem statements and objectives. Section 2 is the literature review where the author reviewed and compared the selected PHP frameworks and the security challenges facing EMR. Section 3 covers the results and discussion part. Section 4 describes the conceptual framework for EMR based on Laravel. Section 5 is the conclusion and is the last section.

2. LITERATURE REVIEW

E-Health is the use of arising information and communication technology, particularly the Web, to enhance or support health and healthcare industry [14]. There are two major e-health systems which are electronic medical record (EMR) and electronic health record (EHR). Though the terms can sometimes be interchanged, these two systems are mildly distinct. EMR is a generic word used by healthcare professionals for a computerized record of patients [1]. EHR is also referred to as computerized health data with the ability to deliver significant advantages to physicians and patients. Unlike EMR, an EHR is not restricted to one care facility, institution, hospital network or state; it is completely interoperable and can also be accessed between many diverse healthcare stakeholders, allowing a healthcare professional to provide the patient with the standard of care much more efficiently [15]. To achieve the goals of using this modern technology, attaining security and privacy in e-health is extremely important. This is essential because digitizing and exchanging health-related data can result in various types of attacks [16]. Below are the security and privacy challenges facing EMR systems and frameworks related topics.

2.1. Security and privacy challenges in EMR

The protection of privacy and confidentiality of health data is of key importance; security leads to trust. Health data security is primarily concerned with privacy and confidentiality [17]. However, healthcare data breaches have become commonplace with 155 reported in the first five months of 2017. Healthcare organizations have remained vulnerable targets as cybercriminals are increasingly easily exploiting vulnerabilities, benefiting from the stolen data [18]. The current high-profile news-reported on EMR data breaches have indeed rendered the change to electronic format even harder for patients, regardless of its possible benefits. Their main concern was the privacy of their data as it is collected and passed across the healthcare system [19].

Institutions need adequate IT resources at the organizational level to introduce effective user authentication, powerful data encryption and regular software updates (for systems varying from EMR to operating systems) [7]. Secure access to e-health records needs three main steps. First, the identity of the user through entering a login username; authentication requesting the user to confirm their identity through passwords; and authorization that allows the user to use EMR [20].

Privacy challenges emerge from EMR suppliers applying their own views on defending the EMR's privacy. Furthermore, because the EMR suppliers are not usually the party responsible for a breach of privacy requirements, the issue will involve a federal judicial resolution. The judicial approach may involve creativity to urge suppliers of EMR to add security solutions that suit federal requirements [21].

In Jordan there is a program called "Hakeem", it is an electronic medical record. The design of the Hakeem program is built on a VistA system. It is used by many nations and was tailored to their

requirements. According to a research study carried out by [20], the Hakeem program restricts unauthorized staff from accessing the data stored in the EMR. A username and password are provided to every authorized employee. The username is made up of the first two alphabets of the name and work code of the employee. Employees can customize the password. There are digits and letters in the password. The purpose of this operation is to authenticate the individual authorized. For instance, staff in departments of medical records can access basic patient data. They can add the patient's email, phone number, and nationality and only view the electronic medical record, but they can't access the doctor's part or add any kind of disease or medical diagnostic tests.

2.2. Common security vulnerabilities in EMR

Current internet security policies and methods are not enough to safeguard the vulnerabilities to which e-health such as Myhealthatvanderbilt.com, Adjuvant. Health, Medical Web Experts, KPOnline and other associated healthcare systems are subjected to. Technically speaking, healthcare applications suffer from all common security vulnerabilities, such as poor authentication, cross-site scripting, SQL injection, cross-site request forgery, where traditional protections, such as SSL, firewall, have become weak, even worthless [22].

2.2.1. Authentication

User authentication plays a significant role in e-health systems to protect patient privacy and security. Single-factor authentication may be affected by attacks, it is not an appropriate method for a system storing sensitive data [23]. Healthcare organizations should develop a more effective multi-factor authentication (MFA) [23, 24].

2.2.2. SQL injection

SQL injection occurred in a database of electronic records and is still present even after years since it first occurred [25]. Most devices and applications (e.g., EMR and EHR) provide a database or data repository for that system to keep information, generally known as a back-end database. Most of these databases have a type of structured query language (SQL) and are extremely susceptible to SQL injection if not properly designed to sanitize input data [26].

A computer-based information system's front-end typically refers to the Graphical User Interface (GUI), which is where the user requests and gets feedback. The front-end, however, is often just a single component of the system and often a small portion of it. As far as EMR software is concerned, this front-end typically comprises of input fields that communicate instantly with the SQL database via web-based or web-enabled interface. If entries are not correctly sanitized in these input fields, they can communicate with the database to an extent that is restricted only by the expertise of the attacker on SQL injection attacks [27].

As for the login page, almost all the attackers will try to use brute force, meaning that it is regarded as a form of brute force to assume the password by attempting out any possibilities such as dictionary attack. Another operation that is very prevalent and commonly used is when the SQL injection intruder puts ' 1 ' OR '1 ' = ' 1' into the username and password. If the system has no SQL injection protection, when intruder inserts this code, the intruder can access the system [25].

2.2.3. Cross-site request forgery

A cross-site request forgery (CSRF) attack causes a victim's logged-on browser to send a fake HTTP request to a vulnerable web application, including the victim's session cookie and any other authentication data that is automatically included. Then the attacker can order the browser of the victim to create requests, the vulnerable application handles it as a legal request from the victim [28]. It may jeopardize the integrity and privacy of patient health records [29].

2.2.4. Cross-site scripting

Flaws in cross-site scripting (XSS) happens when an application gets untrusted data and passes it to a web browser without adequate validation or escape. XSS enables attackers to perform browser scripts that can hijack user sessions, damage websites, or redirect users to malicious sites [28]. Attackers may rob patient sensitive data, redirect users to phishing or malware sites [29].

2.3. History of web framework

Since the rebirth of Web 2.0, demands for development of online content have substantially increased, the users have been the dictate of the web contents and interactions. Web programmers faced quite

some challenges to meet up with the users demands, and then the web application framework was created to appease this issue [30].

Web programmers have been provided with an effective tool to easily and quickly create complex web applications, that have resolved the need to build each system component separately from the start [31]. With companies and firms determination to improve the application features, frameworks ensure the market demands are met [32]. There are currently a number of different framework technologies available and used by various websites, as shown in Figure 1, PHP is originally designed for web development and the most used sever-side scripting language for websites development.

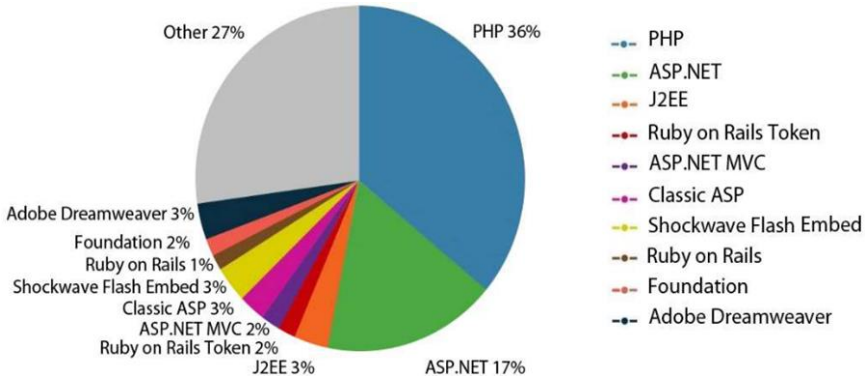

Figure 1. Frameworks usage statistics in the top 1 million websites [33]

PHP is considered one of the top server-side programming languages in the development of web applications because it is dynamic, flexible and easy to learn by new developers. However, sometimes developments with PHP often become repetitive processes due to the needs of basic web applications. The unorganized and messy codes often made by new developers leads to the difficulty in maintaining codes or adding new features to the web application. Developing web application by using a PHP framework help developers to organize the codes, speed up the development process and make web applications more secure [34].

All frameworks are not entirely different from each other, each having its own strength and lapses. Choosing a PHP framework must depend on the advantages that a developer will see, including ease of use, fast development/performance, popularity among other developers, powerful features, and support/ forums [33]. Being one of the most widely used scripting languages in web application development; a framework comparison will be done before selection. PHP frameworks have intuitive characteristics such as efficient execution, open-source code, cross-platform compatibility, and SQL support [13]. PHP framework provides the ability to develop a web application that is complex, secure, a well-rounded application that is faster than before [35]. There are many PHP frameworks available in market space with unique features such as Laravel, CodeIgniter, Zend, CakePHP, Symfony, Phalcon, FuelPHP and Yii that enhance the development life cycle. Despite the numerous frameworks with quality in-built features, choosing the best suitable framework for development to enhance and provide all the supports needed for a project, still, remain a challenge [13].

In this paper, the three most popular PHP frameworks CodeIgniter, Laravel, and Symfony [13, 33] are selected for security features comparison. Symfony and Laravel are the most used and the two PHP-based frameworks offering great feasible options for most PHP projects, with a full-stack web development environment for coders [33]. CodeIgniter on the other hand known as a "powerful framework with a very small print" [32, 36, 37].

2.3.1. Laravel

In the early days of June 2011, Taylor Otwell created a framework called Laravel with its first version 1 beta to solve the lack of essential functions such as user authentication in the CodeIgniter framework [36,37]. As at the time of this study Laravel current stable version is 6.5.0. It is open-source with rich features capable of boosting the speed of web development. For those who are familiar with core PHP and advance PHP, using Laravel makes their work a lot easier [37].

The established and proven web development patterns, convention over configuration that Laravel incorporates to make it ready for use out of the box, Model-View-Controller (MVC) application structure and

ActiveRecord powering its database wrapper adds to other numbers advantages of Laravel over hand-made codes or other similar frameworks. By using these conventions and patterns Laravel helps a developer to build a maintainable web application with easy to understand code separation and in a short time frame [38].

2.3.2. CodeIgniter

CodeIgniter is also an MVC based on PHP Framework created by Rick Ellis. As at the time of this study CodeIgniter current stable version is 3.1.11. What makes it stand out among other frameworks is its features such as no restrictive coding rules, no need to learn template language, small but comprehensive libraries, and thorough documentation. With these features, CodeIgniter is best for small and medium applications. CodeIgniter being introduced on February 28, 2006, with the purpose of assisting its developers to produce application faster than starting from scratch [39]. CodeIgniter allows its developer to publish their plugins to its library making it different from other frameworks [37].

2.3.3. Symfony

Symfony is a PHP framework with the latest version 4.3.6 that requires the support of PHP version 5.6.0 or higher. Symfony is another framework that is popular among developers. It was created in 2005 by Fabien Potencier [36]. Well internationally accepted, a complex and interesting PHP framework for large-scale projects [33].

Results from Table 1 indicate that all the three PHP frameworks except for CodeIgniter have in-built support for authentications. Blade in Laravel handles XSS attacks and CodeIgniter comes with a security filter for XSS, which looks for frequently used methods to cause JavaScript or other kinds of scripts that try to hijack cookies or do other malicious activities. If something is not allowed, the data is converted into character entities to make it safe. However, Symfony did not offer an in-built XSS protection, but it did support it through a template engine called Twig. The frameworks offer different methods of protection against SQL injection. By default, all the frameworks offered protection for CSRF.

Table 1. Common security features in the web frameworks

Parameters	Laravel	CodeIgniter	Symfony
Authentication module	Comes with simplified [33], ready-to-use authentication packages [40]. The authentication facilities of Laravel consist of "guards" and "providers." [41].	CodeIgniter by default does not include an authentication library [30].	User authentication and authorization components [36].
Cross-Site Scripting (XSS) protection	Blade {{}} statements are sent automatically to prevent XSS attacks [41].	CodeIgniter has an integrated XSS filter that is automatically initialized [37].	It did not provide default XSS protection only through Twig [42].
SQL injection protection	Query builder uses the PDO parameter which has built-in support for prepared statements [43].	It offers multiple methods to deal with malicious injection attempts in the database [30].	Integrated with the third party library "Doctrine ORM" can safeguard against SQL injections if used properly [42].
cross-site request forgery (CSRF) Protection	Laravel automatically generates a CSRF token for the user sessions of each application [44].	Inserts a hidden CSRF field into Web forms automatically [30].	By default, Symfony forms provide automatic CSRF protection [42].

3. RESULTS AND DISCUSSION

Strong security determines the effectiveness and overall accomplishments of any application. Regrettably, when it comes to security, many coders skimp because of the lack of knowledge or because development consists of many factors to consider. Frameworks have included many great and easy-to-use security modules to render the application developed by frameworks as secure as possible.

3.1. Authentication

One of the essential factors of the whole web-based project is still authentication. Frameworks include different ready-to-use authentication system that is built-in. Only the models, views, controllers and database migrations that need to be configured by a developer for the application of (e.g., Laravel) to function. If this feature is not included, coders may have to carry out a method of securing a web application in their own way which may sometimes lead to complete missed out of certain vital parts. As mentioned in Table 1, CodeIgniter by default does not include an authentication library [30]. Laravel has a more powerful in-build authentication which is even the motivation behind why the web framework was

developed. The reason for developing the framework was the absence of some vital features, such as user authentication in the CodeIgniter framework, according to Laravel's founder Taylor Otwell [36]. However, between symphony and Laravel, Laravel comes with a more simplified [33] yet, ready-to-use authentication packages [40], Laravel made authentication implementation quite easy. Almost everything is out of the box equipped for a user. The setup file for authentication is situated at config/auth.php, this includes several well-documented alternatives for adjusting authentication services behavior [41].

3.2. Cross-site Scripting

Symfony does not provide default XSS protection, only through a template engine referred to as Twig [42]. The Symfony framework enables coders to use third-party packages to extend their application. Despite its usefulness, third-party packages may bring security vulnerabilities into an application. In addition, the Symfony supports PHP templates (htmlspecialchars), however, this does not default protection for XSS because programmers need to implement clear user output filters, manual code addition is needed which is tedious and error-prone [42]. CodeIgniter has a built-in XSS feature which loads automatically [37]. Laravel provides native support that prevents applications from XSS attacks. By default, Blade{{}} comments are automatically sent via the htmlspecialchars feature of PHP to avoid XSS attacks [41].

3.3. SQL injection protection

In Laravel query builder utilizes PHP data object (PDO), which has built-in support for prepared statements that can safeguard a system against SQL injection attacks [43]. CodeIgniter offers multiple methods to deal with malicious injection attempts in the database [30] e.g. it provides a simplified query binding (also known as prepared data) to allow the system to be put together and to escape the developer's database queries by separating the database syntax and the data when preparing a statement. Symfony is integrated with the third-party library "Doctrine ORM" which can safeguard against SQL injections if the developer used it properly [42].

3.4. Cross-site request forgery

For each active user session controlled by Laravel application, Laravel automatically produces a CSRF "token." This token is used to confirm that the authenticated user is the one who genuinely makes demands to the application [41, 44]. CodeIgniter inserts a concealed CSRF field into web forms automatically. This prevents CSRF attacks by providing web forms a one-time-key token that is incredibly hard for an attacker to imagine when attempting to duplicate the behavior of the form [30]. Forms created with the Symfony form component include default CSRF tokens and Symfony automatically checks them so that a user does not have to do anything to protect against CSRF attacks. In a hidden field called token, Symfony provides the CSRF token by default, but this can be adjusted on a form-by-form basis [42].

4. PROPOSED CONCEPTUAL FRAMEWORK

The proposed framework in Figure 2 is based on common security vulnerabilities in EMR and Laravel features which are on security: authentication, cross-site scripting (XSS), cross-site request forgery (CSRF) and SQL injection as explained in the results and discussion section. Based on the above study, Laravel is integrated with all the security features that can protect against the most common security vulnerabilities in EMR and general web applications. Laravel is preferred against CodeIgniter because as a framework, CodeIgniter did not provide built-in support for authentication and authorization of a user which is vital in developing a secure EMR. Laravel comes with a more simplified and ready-to-use authentication system. Symfony relies on third-party packages which may bring security vulnerabilities. Laravel is the best choice for the implementation of a secure EMR.

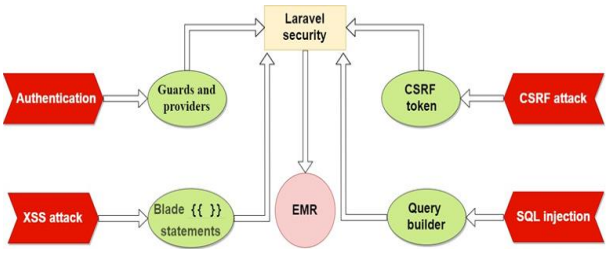

Figure 2. Proposed conceptual framework

5. CONCLUSION

The electronic medical record faces various challenges related to health data integration and interoperability, user interface and user experience design, health data visualization, security, etc. However, the security of EMR remained a top priority due to the highly classified nature of the records it stores that are beneficial to cybercriminals. This paper reviewed the common security vulnerabilities faced by the EMR and proposed a conceptual framework that would minimize these vulnerabilities through Laravel security features and best practices.

The security features of the three most popular and powerful PHP frameworks have been reviewed and compared, Laravel ensures a much more secure EMR by protecting against the various attacks mentioned in this paper through its best out-of-the-box security features, minimizing the attack surface area by default. However, by default, no framework can fully protect the EMR from the cybersecurity threats it faces. Developers need to adopt good security practices. At the end of the day, Laravel is a framework and depends on how developers use it.

ACKNOWLEDGEMENTS

The authors wish to thank the Research Management Centre under the MARA University of Technology's grant of 600-IRMI/REI 5/3 (010/2018).

REFERENCES

[1] M. Chidinma and U. Bright, "A Hybrid of Electronic Health Record (EHR) System and Health Maintenance Organizations (HMOs): A Sure Way to Improve Healthcare System in Nigeria," *International Journal of Computer Science and Mathematical Theory*, vol. 4, no. 3, pp. 14–22, 2018.

[2] Dave Garets and Mike Davis, "Electronic Medical Records vs. Electronic Health Records : Yes , There Is a Difference," *HIMSS AnalyticsTM White Paper*, pp. 1–14, 2006.

[3] A. O. Attah, "Implementing an electronic health record in a Nigerian secondary healthcare facility. Prospects and challenges," *UiT-The Arctic University of Norway Tromso*, Norway., pp. 1–82, 2017.

[4] G. Amirthalingam and H. Thangavel, "Multi-Biometric Authentication Using Deep Learning Classifier for Securing of Healthcare Data," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no.4,pp. 1340–1347, 2019.

[5] M. V. Vucha and A. L. Siridhara, "High Speed Cryptography Architecture for Health Information Exchange," *International Journal of Advanced Trends in Computer Science and Engineering Available*, vol. 8, no. 4, pp. 1443–1448, 2019.

[6] M. H. Gabriel, A. Noblin, A. Rutherford, A. Walden, and K. Cortelyou-Ward, "Data Breach Locations, Types, and Associated Characteristics Among US Hospitals," *American Journal of Managed Care*, vol. 24, no. 2, pp. 78–84, 2018.

[7] J. G. Ronquillo, J. Erik Winterholler, K. Cwikla, R. Szymanski, and C. Levy, "Health IT, hacking, and cybersecurity: national trends in data breaches of protected health information," *JAMIA Open*, vol. 1, no. 1, pp. 15–19, 2018.

[8] R. E. Moffit and B. Steffen, "Health Care Data Breaches: A Changing Landscape," *Maryland Health Care Commission*, December, pp. 1–19, 2017.

[9] N. Kagalwalla, T. Garg, P. Churi, and A. Pawar, "A Survey on implementing privacy in Healthcare: An Indian Perspective," *International Journal of Advanced Trends in Computer Science and Engineering Available*, vol. 8, no. 3, pp. 963–982, 2019.

[10] T. T. Smith and G. Velkova, "Examining data privacy breaches in healthcare," *ProQuest Dissertations and Theses*, pp. 1–176, 2016.

[11] B. Rotimi-williams, "Development of a Secured Web-Based Healthcare Portal," *Advance Academic Publisher*, pp. 1–91, 2016.

[12] K. Reintjes, "a Benchmark Approach To Analysis the Security of Web Frameworks," Master, Computer Science Radboud University Nijmegen, pp. 1–108, 2014.

[13] M. Laaziri, K. Benmoussa, S. K. Khoulji, and M. Larbi Kerkeb, "A Comparative study of PHP frameworks performance," Procedia Manufacturing, vol. 32, pp. 864–871, 2019.

[14] Oh H, Rizo C, Enkin M, and Jadad A, "What is eHealth?: a systematic review of published definitions," *Journal of Medical Internet Research*, vol. 41, no. 1, pp. 1–11, 2005.

[15] K. Rey, "Understanding the challenges faced by Central Valley California region ambulatory care practitioners when adopting electronic health records," *Ssrn*, pp. 1–175, 2015.

[16] N. A. Azeez and C. Van der Vyver, "Security and privacy issues in e-health cloud-based system: A comprehensive content analysis," *Egyptian Informatics Journal*, vol. 20, no. 2, pp. 97–108, 2018.

[17] E. K. Achampong, "Electronic Health Record (EHR) and Cloud Security: The Current Issues," *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, vol. 2, no. 6, pp. 417–420, 2014.

[18] A. Mcleod and D. Dolezel, "Understanding Healthcare Data Breaches: Crafting Security Profiles," AMICIS, no. June, pp. 1–9, 2018.

[19] D. K et al., "EMR Access and Confidentiality Based on Patient and Hospital Staff Perspectives," *The Open Public Health Journal*, vol. 11, no. 1, pp. 533–545, 2018.
[20] N. Innab, "Availability, Accessibility, Privacy and Safety Issues Facing Electronic Medical Records," *International Journal of Security, Privacy and Trust Management*, vol. 7, No 1, pp. 1–10, 2018.
[21] J. C. Dechene, "The challenge of implementing interoperable electronic medical records," Annals of health law / Loyola University Chicago, School of Law, Institute for Health Law, vol. 19, no. 1, pp. 195–203, 2010.
[22] X. Li and Y. Xue, "Protecting Web-based Patient Portal for the Security and Privacy of Electronic Medical Records," Conference: Proceedings of the 3rd USENIX conference on Health Security and Privacy, pp. 1–2, 2012.
[23] M. Jayabalan and T. O'Daniel, "A study on authentication factors in electronic healthrRecords," *Journal of Applied Technology and Innovation*, vol. 3, no. 1, pp. 7–14, 2019.
[24] R. Ganesan, "Patient Portal Identity Proofing and Authentication," Healthcare Information and Management Systems Society (HIMSS), pp. 2–36, 2016.
[25] M. A. Mohd Yunus, M. Zainulariff Brohan, N. M. Nawi, E. S. Mat Surin, N. Azwani Md Najib, and C. W. Liang, "Review of SQL Injection : Problems and Prevention," *JOIV : International Journal on Informatics Visualization*, vol. 2, no. 3–2, pp. 215–219, 2018.
[26] P. A. H. Williams and A. J. Woodward, "Cybersecurity vulnerabilities in medical devices: A complex environment and multifaceted problem," *Medical Devices: Evidence and Research*, vol. 8, pp. 305–316, 2015.
[27] L. Heinke, "The need for an investigation into possible security threats associated with SQL based EMR software," in *Proceedings of the 5th Australian Information Security Management Conference*, pp. 80–86, 2007.
[28] Q. Brian, "Breach Control: Best Practices in Health Care Application Security," *SANS Institute Information Security Reading Room*, pp. 2–22, Febuary 2016.
[29] R. Pankomera and D. Van Greunen, "Mitigating vulnerabilities and threats for patient-centric healthcare systems in low income developing countries," in *IST-Africa Week Conference, IST-Africa*, pp. 1–11, 2017.
[30] L. Forte, "Building a Modern Web Application Using an MVC Framework," Oulu University of Applied Sciences, pp. 3–65, 2016.
[31] J. Pater, "Modern Web Application Frameworks," *Annals of Dunarea de Jos University. Fascicle I : Economics and Applied Informatics*, vol. 21, no. 3, pp. 82–86, 2015.
[32] R. Das and L. P. Saikia, "Comparison of Procedural PHP with Codeigniter and Laravel Framework," *International Journal of Current Trends in Engineering & Research (IJCTER)*, vol. 2, no. 6, pp. 42–48, 2016.
[33] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi, and A. El Yamami, "A comparative study of laravel and symfony PHP frameworks," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 1, pp. 704–712, 2019.
[34] X. Li, S. Karnan, and J. A. Chishti, "An empirical study of three PHP frameworks," in 4th International Conference on Systems and Informatics, ICSAI, pp. 1636–1640, 2017.
[35] K. Benmoussa, M. Laaziri, S. Khoulji, K. M. Larbi, and A. El Yamami, "A new model for the selection of web development frameworks: application to PHP frameworks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 695–703, Feb. 2019.
[36] J. Samra, "Comparing Performance of Plain PHP and Four of Its Popular Frameworks," Linnaeus University, Sweden, pp. 1–22, 2015.
[37] N. Solanki, D. Shah, and A. Shah, "A Survey on different Framework of PHP," *International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS)*, vol. 6, no. 6, pp. 155–158, 2017.
[38] M. Surguy, "Laravel: my first framework," *Brimingham: Leanpub*, pp. 1–165, 2014.
[39] H. Hustinawati, A. Kurnia Himawan, and L. Latifah, "Performance Analysis Framework Codeigniter and CakePHP in Website Creation," *International Journal of Computer Applications*, vol. 94, no. 20, pp. 6–11, 2014.
[40] Q. H. Nguyen, "Building a web application with Laravel 5," Oulu University of Applied Sciences, pp. 1–35, 2015.
[41] T. Otwell, "Laravel Documentation - 6.x," *Driade Laravel-Book*, no. 7, pp. 1–495, 2020.
[42] S. Labs, "Symfony2 Docs Documentation," Read the Docs, no. January 10, pp. 1–644, 2016.
[43] H. Halimi and I. Jound, "Comparison of performance between Raw SQL and Eloquent ORM in Laravel," Faculty of Computing Blekinge Institute of Technology, Sweden, pp. 1–37, 2016.
[44] W. J. Gilmore, "Easy Laravel 5 A Hands On Introduction Using a Real-World Project," *Lean Publishing*, no. Febuary 16, pp. 2–289, 2018.