Developing Framework for Web Based e-Commerce: Secure-SDLC

Bambang Sugiantoro\textsuperscript{1}, Muhammad Anshari\textsuperscript{2} and Danang Sudrajat\textsuperscript{3}

\textsuperscript{1,3} Universitas Islam Negeri Sunan Kalijaga, Yogyakarta, Indonesia
\textsuperscript{2} Universiti Brunei Darussalam, Brunei Darussalam

Abstract. Software development usually follows the stages known as the Software Development Life Cycle (SDLC). Development begins with the requirements phase and then continues with the design, development of test plans, implementation (coding), testing, and deployment. However, the security aspect has not been explicitly reflected in SDLC. In fact security must be present at every stage of the SDLC. Application of security in the process of developing software, especially e-Commerce, one of which uses the Security Development Lifecycle-Agile (SDL-Agile) method. The advantage of the SDL-Agile method is the application development process which incorporates a safety factor in each application development process, which is concise, and emphasizes user satisfaction. This objective of this paper examine the deployment of SDL Agile Method for eCommerce website to measure its confidentiality, integrity, and availability of the systems.

1. Introduction
Security in an e-commerce system has a very pivotal role for business continuity and sustainability \cite{1, 2}. Whenever the system can be hacked by intruders it results tangible and intangible losses such as material, data, control towards the system, customer confidence and much more. There are many cases of hacking of e-commerce systems due to system weaknesses. One of a well-known hacking case in Indonesian’s e-commerce was hacking towards Tiket.com. It was reported that Tiket.com suffer losses of up to Rp 1.9 billion when the site was hacked \cite{3} and it is definitely has a major impact on customer confidence. E-commerce systems must be regularly monitored from development to maintenance \cite{4}.

Just like any other information systems, e-commerce development also follows certain steps or stages, known as the Software Development Life Cycle (SDLC) \cite{5}. Development begins with the requirements phase and then continues with the design, development of the test plan, implementation (coding), testing, and launching (deployment) \cite{6}. However, what has not been explicitly seen in SDLC above is the security aspect that should present at every stage of the SDLC.

One of the method or approach in the process of developing e-commerce system that accommodate security aspect in every steps is Security Development Lifecycle-Agile (SDL-Agile). SDL-Agile method is the application development process which includes security factors in each application development process. Therefore, it is necessary to design an e-commerce system that is able to take...
security procedures during the development processes. In this study, authors deployed SDL-Agile method to develop the secure system development life cycle for Small Medium Enterprises (SMEs).

2. Literature Review

Electronic commerce (e-commerce) is an activity related to the purchase, sale, marketing of goods or services by utilizing electronic systems such as the internet or computer networks [7]. Since e-commerce requires a very safe and secure online activities, there must be security aspect in all its processes. In securing e-commerce system, there are three aspects that become the objectives of a security. Firstly, e-commerce must have availability aspect of the system. This aspect relates to the availability of data stored on the system so that it can be accessed, retrieved or seen. For instance, whenever a threat either coming from outside or inside the system that might damage or eliminate existing data on the system such as viruses, spyware, etc. then they can have an impact on data availability. Likewise natural disasters like earthquakes can also result in loss of important data due to damaged data storage or hardware. Secondly, e-commerce must preserve confidentiality aspect. It relates to the confidentiality of important data stored on the system that cannot be accessed or used by unauthorized users. For instance, if there is an internal user who has a access the system but breach the trust by passing sensitive or confidential data to third parties or competitors which might results in losses to the company. It could also be, users who do not have permission and access but they try by force to enter the system in order to gain access towards data that is confidential. Finally, data Integrity is an aspect related to the consistency or integrity of the data stored on the system that must not be changed by unauthorized users. This aspect cannot be fulfilled if there is a change in data by people who are not entitled to make changes to the data. For example, an intruder attempts to make a profit by breaking a banking system and changing his savings data like adding the nominal for his savings account. According to the Open Web Application Security Project (OWASP) [8] that there are the 10 most critical risks in application security are as follows:

Table 1. Top 10 Security Threat in 2013 and 2017.

| OWASP Top 10-2013                          | OWASP Top 10-2017                          |
|------------------------------------------|------------------------------------------|
| A1 – Injection                           | A1 – Injection                           |
| A2 – Poor session authentication and management | A2 – Poor session authentication          |
| A3 – Cross-Site Scripting (XXS)           | A3 – Sensitive data exposed               |
| A4 – Unsafe direct object reference       | A4 – XML External Entities (XXE)          |
| A5 – Security configuration error         | A5 – Poor access control                 |
| A6 – Sensitive data exposed               | A6 – Security Configuration Error         |
| A7 – Lost level function access control   | A7 – Cross-Site Scripting (XXS)           |
| A8 – Cross-Site Forgery (CSRF)            | A8 – Unsecured decentralization           |
| A9 – Uses known vulnerable components     | A9 – Uses known vulnerable components     |
| A10 – Unvalidated redirections and forwards | A10 – Insufficient recording and monitoring |

2.1. Security Development Lifecycle

Security Development Lifecycle is a formal methodology for software development that implements security and privacy considerations throughout the phases of the software development process, helps developers build highly secure software, handles security compliance requirements, and reduces development costs [9].
The stages of developing software security are as follows:

**Training**: Security is everyone's business task. Developers, service engineers, and program and product managers must understand the basics of security and know how to build security into software and services to make products safer while still meeting business needs and providing user's value. Effective training will complement and re-enforce security policies, SDL practices, standards and software security requirements, and be guided by insights gained through data or technical capabilities that are newly available. Although security is everyone’s task, it is important to remember that not everyone needs to be a security expert or try to become a skilled penetration tester. However, making sure everyone understands the attacker's perspective, and their goals will likely help in developing everyone’s awareness and the urgency of security for organization.

**Requirements**: the need to consider security and privacy is a fundamental aspect of developing highly secure applications and systems and regardless of the development methodology used, security requirements must be constantly updated to reflect changes in the functionality required and changes in the threat landscape. The optimal time to define security requirements is during the initial design and planning stages because this allows the development team to integrate security to minimize disruption. Factors that affect security requirements include (but are not limited to) legal and industry requirements, internal standards and coding practices, reviewing prior incidents, and known threats. These requirements must be tracked through work tracking systems or through telemetry derived from engineering pipelines.

**Design**: SDL helps developers to implement safe features, because they are well engineered with regard to safety. To achieve this, developers will usually rely on security features, such as cryptography, authentication, logging, and more. In many cases, the selection or implementation of security features has proven so complex that design or implementation choices tend to produce vulnerabilities. Therefore, it is very important that this is applied consistently and with a consistent understanding of the protection they provide.

**Implementation**: Define and publish approved tool lists and related security checks, such as compiler / link options and warnings. Developers should use the latest version of approved tools, such as compiler versions, and to take advantage of the functionality and protection of new security analyses.

**Release**: Preparing an Incident Response Plan is very important step to overcome new threats that can arise over time. This must be made in coordination with the Product Safety Incident Response Team (PSIRT) dedicated to the system. The plan should include who to contact in the event of a security emergency, and establish protocols for security services, including plans for codes inherited from other groups in the organization and for third party codes. The incident response plan must be tested before it is needed.

**Response**: Implement an incident response plan that has been tested after the systems is released.

### 3. Methodology

The system development method used in this study is one of the secure system development lifecycle (Secure-SDLC), namely Security Development Lifecycle-Agile (SDL-Agile) [10]. The SDL-Agile method is suitable for the development of small-scale systems, which more quickly adapt to changing requirements. In this method there are three categories of SDL-Agile requirements; First, Every
Sprint Requirement - This category consists of security needs that are very important for security so software must not be released without requirements being met. Some examples of sprint requirements such as update the threat model, fix all problems identified by the code analysis tool for unmanaged code, follow the input validation and output coding guidelines to defend against cross-site scripting attacks. Second, Bucket Requirements - this category consists of tasks that must be carried out regularly during the life of the project but which are not so important to be mandated for each sprint. Within the bucket requirements there are three separate task buckets; verification task, the review task, and the planning task. Examples of task buckets are security verification tasks (running fuzzing tools, manual and automatic codes, and attack surface analysis), review tasks such as design review and privacy review, planning tasks such as determining security bugs and creating supporting documents. Thirdly, One-time Requirements - The one-time category is an SDL requirement that must be met when first starting a new project with SDL-Agile or when starting to use SDL-Agile with an existing project. This is usually a one-time project that does not need to be repeated after completion. Examples of one-time requirements are configure system bug tracking, security identification, and basic threat models.

4. Security Analysis
In this section we deployed OWASP approach to analyse as follows: Firstly, identifying risks is the process of gathering information about the threat agents involved, the attacks that will be used, the vulnerabilities involved, and the impact of successful exploitation on business. There may be several possible groups of attackers, or even some possible business impacts. In practice, it is also providing the worst case option, because that will result in the highest overall risk. Secondly, factors for estimating possibilities - risk assessment based on the OWASP framework is done through a scoring system that is prepared based on threat agent factors (actors / potentially threatening parties), vulnerability factors (system weakness points), and impacts (potential losses / impacts of such attacks). Threat agent factor measures how big the chance of a risk threat is based on the capacity of the agent's ability to attack the system. Threat agents are divided into skill level, motive, opportunity and size (Table 2). Each threat agent for each parameter is given a score. Vulnerability factor measures how easily these weaknesses are found, which can be identified through several components: Ease of discovery (how difficult the weakness can be found), Ease of exploit (how easily the threat agent exploits this weakness), awareness (what level of concern the system manager against these types of attacks), and intrusion detection (the extent to which attempts to detect such attacks are applied to the system).

| Parameter      | Condition                        | Score |
|----------------|----------------------------------|-------|
| Expertise Level| Security penetration skills       | 1     |
|                | Network and programming skills    | 3     |
|                | Advanced computer user           | 5     |
|                | Technical expertise (in average)  | 6     |
|                | No technical expertise           | 9     |
| Motivation     | No rewards                       | 1     |
|                | Possible rewards                 | 4     |
|                | High rewards                     | 9     |
| Opportunity    | Full access / Expensive          | 0     |
|                | Required access / special resources | 4   |
|                | Some access or resources are needed | 7   |
|                | No access or resources required   | 9     |
|                | Developer                        | 2     |
To measure the technical impact of an attack or disturbance on the system, it uses two aspects that are technical aspects and business aspects (Table 3).

| Aspect           | Parameter                              | Condition                        | Score |
|------------------|----------------------------------------|----------------------------------|-------|
| Lost confidentiality | Minimal Data / not sensitive          | 2                                |       |
|                  | Minimal Critical Data Disclosed        | 6                                |       |
|                  | Extensive non-sensitive data disclosed | 6                                |       |
|                  | Important data widely disclosed        | 7                                |       |
|                  | All data disclosed                     | 9                                |       |
| Loss of integrity | Minimal data minor corrupt            | 1                                |       |
|                  | Minimal Data Critical Corrupt          | 3                                |       |
|                  | Corrupt data is rather extensive       | 5                                |       |
|                  | Seriously extensive corrupt data       | 7                                |       |
|                  | All data is corrupt                    | 9                                |       |
| Loss of availability | Secondary services are minimally disrupted | 1                              |       |
|                  | Primary services are minimally disrupted | 5                             |       |

**Figure 2.** Results of Assessment of Risk Theft / Loss of Databases.

Figure 2 shows the results of assessment of risk for database. It indicates possible the security risks’ level of possibility of security breach. It can be seen for the possibility of assessing the risk of theft / loss of data, the threatening factors and system weaknesses have a value of 4,500, which means that there is a risk at the medium level. Based on these results the authors take one of the 10 security risks according to OWASP of data theft that is usually caused by SQL-Injection.

| Table 3. Technical and Business Aspect Parameters |
5. System Testing
System testing is a stage to find errors and deficiencies in the system being built, so it can be seen whether the system has met the criteria in accordance with the objectives or not. The testing method used to test the security system that has been built is a manual testing method.

5.1. SQL Injection Testing
Test manually SQL injection in the system login form by entering “’or’ 1 =’ 1” in the username and password. If the user manages to log in without using the username and password stored in the database, the system is vulnerable to SQL injection attacks and if the user cannot enter the system is safe from SQL injection attacks. SQL injection testing on the system is also done by finding the error message that is displayed by modifying the url on the system. A secure system does not display an error message so that the location of the file and its database are unknown (Figure 3).

![Figure 3. SQL Injection Testing](image)

5.2. Password Testing
The purpose is to find out the password that has been stored in the database can be decrypted or not, password decryption testing can use the online application through the site https://hashkiller.co.uk on the hash cracker menu, the password storage encryption type is selected. After that, enter the password that will be decrypted in your hashes field, then click the button Crack my Hashes and the results of the decrypt will appear in the Cracker Results column. If successful, it will bring up the decrypted password and if the password cannot be decrypted the notification will not find your hashes, which means encrypting the password in the secure database only members will know. The results of testing the confidentiality aspects of the website application using SDL-Agile method.

| No. | Statements                                                                 | Yes | No |
|-----|-----------------------------------------------------------------------------|-----|----|
| 1.  | Sensitive data sent over the network must also be encrypted                 | 20  | 0  |
| 2.  | The password when displayed on the screen, must not be in the form that can be read | 20  | 0  |
| 3.  | The password must not be logged.                                            | 20  | 0  |
4. The user cannot see transactions (data) from the user the other. 20 0  

Total 80 0

5.3. Integrity Testing
The results of testing the aspects of integrity website application using the SDL-Agile method (Table 5).

| No. | Statement                                                                 | Yes | No |
|-----|---------------------------------------------------------------------------|-----|----|
| 1.  | Apply strong encryption to data storage and transmission media.            | 20  | 0  |
| 2.  | Apply strong authentication and validation to every file access / account | 20  | 0  |
|     | login / action that is applied.                                           |     |    |
| 3.  | Applying strict access control to the system, i.e. every existing account | 20  | 0  |
|     | must have limited access rights.                                          |     |    |
|     | Total                                                                      | 60  | 0  |

5.4. Availability Testing
The results of testing the availability aspect website application using SDL-Agile method (Table 6).

| No. | Statement                                                                 | Yes | No |
|-----|---------------------------------------------------------------------------|-----|----|
| 1.  | Applications must be available during business hours.                     | 20  | 0  |
|     | Total                                                                      | 20  | 0  |

5.5. Injection SQL Testing Results
The results of testing two website applications that use the SDL-Agile method and do not use SDL-Agile (Table 7).

| No. | Statement                                                                 | Yes | No |
|-----|---------------------------------------------------------------------------|-----|----|
| 1.  | Can overcome SQL Injection attacks by entering SQL commands 'or' 1         | 20  | 0  |
|     | '=" 1 in the username and password                                         |     |    |
|     | Does not display an error message when modifying parameters sent via       | 20  | 0  |
|     | url with a special character that is single quote (' )                     |     |    |
| 2.  |                                                                           |     |    |
|     | Total                                                                      | 40  | 0  |
Table 8 shows the full testing results by comparing the website using SDL-Agile testing and non SDL-Agile and the result shows promising security with SDL-Agile.

6. Conclusion
The design process of e-Commerce applications using the Agile secure-SDLC method has three stages namely analysis, implementation and evaluation. System testing is a stage to find errors and deficiencies in the system being built, so it can be seen whether the system has met the criteria in accordance with the objectives or not. The testing method used to test the security system that has been built is a manual testing method. Based on the test results, it can be concluded that the e-Commerce application that has been made using the SDL-Agile method is able to overcome SQL Injection attacks and can store member passwords securely without decrypting. However, there is a need for further system development so that the system can overcome other hacking attacks.

References

[1] Anshari, M., Almunawar, M. N., Masri, M., & Hamdan, M. (2019). Digital marketplace and fintech to support agriculture sustainability. Energy Procedia, 156, 234-238.
[2] Anshari, M., Alas, Y., & Sulaiman, E. (2019). Smartphone addictions and nomophobia among youth. Vulnerable Children and Youth Studies, 14(3), 242-247.
[3] Prihadi, S. D. (2017) Begini Cara Hacker Bobol Situs Tiket.com, CNN Indonesia. Available at: https://www.cnnindonesia.com/teknologi/20170331145137-185-204065/begini-cara-hacker-bbol-situs-tiketcom (Accessed: 22 February 2019).
[4] Anshari, M., Alas, Y., & Guan, L. S. (2016). Developing online learning resources: Big data, social networks, and cloud computing to support pervasive knowledge. Education and Information Technologies, 21(6), 1663-1677.
[5] Hermawan, A. F., & Anshari, M. (2016). Performance Measurement Of Project Management By Using Fanp Balanced Scorecard. Journal of Theoretical and Applied Information Technology (JATIT), 83(2), 2005-2016.
[6] Low, K. C. P., & Anshari, M. (2013). Incorporating social customer relationship management in negotiation. International Journal of Electronic Customer Relationship Management, 7(3/4), 239-252.
[7] Wijaya, E. (2015) Pengertian E-Commerce, Manfaat Serta Keuntungan E- Commerce. Available at: http://www.patartambunan.com/pengertian-e-commerce-manfaat-serta-keuntungan-e-commerce/ (Accessed: 23 February 2019).
[8] OWASP (2017) ‘OWASP Top 10-2017 The Ten Most Critical Web Application Security Risks’, p. 5. Available at: https://owasp.org.
[9] Microsoft (2013) ‘Microsoft Security Development Lifecycle’, p. 2.
[10] Iqbal, M. M. (2016) Rancang Bangun Sistem Penyedia Konten Menggunakan Metode Agile. UIN Sunan Kalijaga.