JURNAL EMACS
[Engineering, MAthematics and Computer Science] Vol.7 No.2 May 2025: 129-138 DOI: 10.21512/emacsjournal.v7i2.13429
e-ISSN: 2686-2573

Combining Academia and Industry Approach for Secure Coding and Requirements Checklist in S-SDLC: Systematic Literature Review

Anderies[1*], Ika Dyah Agustia Rachmawati[2], Kenny Jingga[3], Calvin Linardy Candra[4]

[1,3]Computer Science Program, Computer Science Department, School of Computer Science,
[2] Cyber Security Program, Computer Science Department, School of Computer Science,
Bina Nusantara University,
Jakarta, Indonesia 11480

[4]Cyber Security Research, School of Information Technology,
Deakin University,
Waurn Ponds, Australia VIC 3216

anderies@binus.ac.id, ika.rachmawati001@binus.ac.id, kenny.jingga@binus.ac.id,
s224910144@deakin.edu.au

*Correspondence: anderies@binus.ac.id

Abstract *— Rapid progress of digital transformation has occurred across governments, organization and vendors around the world. where this rapid digital transformation is not linearly followed by the security protection of digital infrastructure and its application. For example, in Indonesia One of the largest banks was unable to operate its online and physical services for three consecutive days due to a cyber-attack. And many international organizations also experienced the same thing or even worse like bankruptcy. Because of this phenomenon the authors have performed a systematic literature review and identified there are two important phases namely requirement and coding in secure software development lifecycle (S-SDLC). In this study the authors compose 18 Secure Requirement practices (SREC) and 72 Secure Coding Checklist (SCOC) checklist based on Combining previous academia research study and international standard of open secure coding practices (OSCP) in which we target the security vulnerable most occurred to governments, organization and vendors around the world according to Open Web Application Security Project Foundation. This checklist can be embedded in the Quality Assurance process to check in sequence whether the Requirements and Coding that are produced are safe or not from the cyber-attack. Additionally, the checklist approach is simple to understand and can be implemented to a popular public consumer automation testing tools enabling faster software development while maintaining software security.*

Keywords: *Cyber Security; Secure Software Development Lifecycle; Software Engineering; Systematic Literature Review*

I. INTRODUCTION

Secure Software Development Lifecycle (S-SDLC) is currently neglected by most organizations, vendors and government and user itself. These parties are forgetting that incorrect environment is potentially resulting threats such as financial losses, operational disruptions and reputational damage to the parties itself (Humayun et al., 2022; Inggarwati et al., 2020; A. W. Khan et al., 2022). In May 2023, Bank Syariah Indonesia, one of the largest financial institutions in Indonesia, fell victim to a ransomware attack orchestrated by the cybercriminal group LockBit 3.0. The attackers claimed to have exfiltrated 1.5 terabytes of sensitive data, including records of 15 million customers, and threatened to disclose the information unless a ransom was paid. This cyberattack resulted in the disruption of BSI's banking services, rendering both mobile and physical transactions inaccessible to customers. Consequently, the incident inflicted substantial financial losses and severely compromised the institution's reputation, highlighting critical vulnerabilities in the cybersecurity framework of the banking sector (Fitriani et al., 2023).

Research claims by H. Sadler (Sadler, 2020) state that secure software development lifecycle environment or S-SDLC is necessary to avoid many external threats. It supports developers to build secure software applications while also enhancing their skill, competencies and productivity which impacted many aspects in software development (Saeed et al., 2025).

Advancements in Information and Communication Technology (ICT) have transformed various aspects of human life, from daily activities to critical sectors such as healthcare, finance, and other essential works. These activities often rely on software applications, making the security aspect of software applications important.

Additionally, software applications are typically interconnected with other applications, which underline the importance for governments, organizations and vendors to implement a Secure Software Development Lifecycle (S-SDLC) (de Vicente Mohino et al., 2019).

Due to lack of concern in S-SDLC that may threaten Confidentiality, Integrity, Availability and Valuable (CIAV) Resources for governments, organization and vendors. In this study the authors want to perform a Systematic Literature Review (SLR) on S-SDLC and inform what faces organization, vendors and firm if they are neglecting the S-SDLC (B. Kitchenham et al., 2009; B. A. Kitchenham, 2012). and introduce insight of comprehensive security checklist for S-SDLC that derived from systematic literature review and international security standard to mitigate security issues and cyber-attack to software application. The author also utilizes the Retrieval Augmented Generative (RAG) Artificial Intelligence product to perform searching, collecting and filtering the primary, review and SLR research study. In purpose of improving quality of articles collection and information to answer Research Questions (Ayemowa et al., 2024; Gwon et al., 2024).

We use checklist approaches because we aim for simplicity of practical uses and implementation purposes yet proven and beneficial on the field, several studies also use checklist approaches for their ICT Infrastructure in the healthcare industry to mitigate cyber-attack (Baz et al., 2023; Rajamäki et al., 2024).

The remainder of this paper is structured as follows: Section 2 provides details on phases of SLR conducted. Methods and research study to answer two research questions about security issues, cyber-attack and S-SDLC in Section 3. Section 4 is Result and Discussion which Introduce a minimalistic S-SLDC with checklist approach from international previous study and international standard. Section 5 of this study.

II. METHODS

A Systematic Literature Review a.k.a SLR was selected to be foundation of the research methodology on this study review, because SLR shown of credibility in the process of development article collection and reducing interpersonal-bias, However the authors are performing smaller enhancement for the SLR method rigorously and having alignment with the objective which is finding reliable, latest and trusted sources of the study. According to Kitchenham an SLR has three main phases (B. Kitchenham et al., 2009; B. A. Kitchenham, 2012). The first is planning, conducting and the third answering. see Table 1 for the details.

Table 1. SLR Phases

Phases	Sub Phases
Planning	● Research Question ● Study Sources ● Inclusion and Exclusion Criteria ● Search Strings ● AI Prompting String ● Study Selection by Matching Abstract with Authors keywords
Conducting	● Filter and Selecting the study ● Reading the study content
Answering	● Answering Research Question ● Making Responsible Insight and Comments

2.1 Planning Phase

2.1.1 Research Question.

The current study conducted a modified SLR, there are two research questions that were answered in this study:

- RQ1: What threat do organizations, firms, or vendors face if they neglect the Secure Software Development Life Cycle (S-SDLC) ?
- RQ2: What Secure Software Development Lifecycle practices should be implemented during requirement engineering and coding stages to mitigate the security threat ?

2.1.2 Study Sources.

In this study, the paper is gathered by manual search, there are total of six digital repositories were chosen, the following are scholarly digital sources that were chosen:

1. MDPI
2. Google Scholar
3. Science Direct
4. Springer Link
5. Wiley Online Library
6. ACM Digital Library

2.1.3 Search String. In this study the paper using search strings for searching in scholarly database as follows:

1. Secure Software Development
2. Software Development Lifecycle
3. Cyber Security Condition
4. Standard of Application System
5. Security Solution
6. Software Application
7. Global Software Development

2.1.4 AI Prompting String.

Generative Artificial Intelligence (GAI) has been transformed into Retrieval-Augmented Artificial Intelligence (RAG) which can perform a combination of operations like searching on the internet, thinking, and summaries. This RAG also showed potential in aiding systematic literature reviews according to

several current latest study in literature review (Ayemowa et al., 2024; Gwon et al., 2024).

For the objective of improving the quality of answering research questions the authors utilize this popular Retrieval Augmented Generative (RAG) to perform deep filtering, summarizing and gaining insight from primary and secondary study. The authors perform prompting string as follows:

1. "Make sure the information and reference are from popular journal or international conference indexed by Scopus"
2. "Can you reference the real example from popular open access journal?"
3. "Please give me the example of (keyword) from credible journal or sources ?"
4. "Can you find and answer these (keyword) scientifically using references from popular and credible journal ?"
5. "Search for me the 10-20 journal Scopus indexed related to (keyword)?"

2.2 Inclusion and Exclusion Criteria

2.2.1 Inclusion Criteria

For data Inclusion, we adopted the following guidelines-based parameters used by other researchers:

1. Papers must be written in English.
2. Papers were published between 2000 to 2024.
3. Articles related to the domain of Secure Software Development or Threat to Software Application.
4. Articles related to Cyber Security.
5. Articles related to Cyber Attack.

2.2.2 Exclusion Criteria

The authors followed the guidelines based on parameters used by other researchers:

1. Papers that don't deal with secure software development lifecycles.
2. Papers that don't mention any secure or software risk keywords
3. Publications are not peer-reviewed and do not conform to a complete book's abstract, an editorial, or a letter.
4. Paper is not written in English
5. Duplicate papers were not considered.

2.2.3 Conducting Phase

In the conducting phase we utilize several tools such as Preferred Reporting Items for Systematic Reviews and Meta Analyses (PRISMA) and Retrieval Augmented Generation (RAG) Searching and Filtering. PRISMA methodology is valuable tool to conduct systematic review and meta-analysis or SLR in different fields, including the computer science study and its branch knowledge (Javed et al., 2023).

Figure 1 shows the PRISMA process that authors perform there are four phases as follows:

A. Phase 1 : using search string and prompting string to find related articles.
B. Phase 2 : Perform Inclusion and Exclusion Criteria based on articles abstract and articles full-body.
C. Phase 3 : Perform full-abstract and skim reading on the articles body.
D. Phase 4 : Final collection of primaries study and Systematic Literature Review studies.

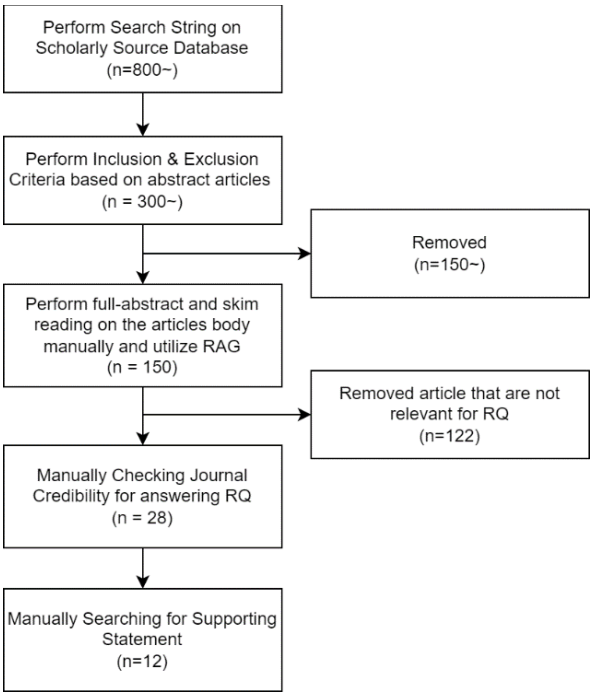

Figure 1. The PRISMA Process of relevant literature Review

Thanks to the advancement of Generative AI, the author is possible to perform phase 1 and 2 which consist of performing inclusion and exclusion criteria on the body because we have power of quick summarative to assist the author perform deep inclusion and exclusion criteria to find good quality of articles that match our research questions.

2.3 Answering Phase

In answering phase, the authors utilize the source from reputable international journal articles, international conference Scopus indexed and S-SDLC reputable security guidelines to provide information and insight for the author to answer research questions. in RQ 1 author's make a list of tables that cyber threat may occur to organizations if neglecting the S-SDLC, in RQ 2 the author utilizes many guideline and journal articles and translate it into actionable checklist table for requirement and coding stages. This research questions review isn't limited to S-SDLC within specific industries as a result, the insights derived are

more generalized and may apply to any organizations, firms and vendors.

III. RESULT AND DISCUSSION

RQ 1: What threat do organizations, firms, or vendors face if they neglect the Secure Software Development Life Cycle (S-SDLC)?

Ignoring Secure Software Development Lifecycle (S-SDLC) can expose companies to various threats, including data breaches, financial losses, operational disruptions and reputational damage. failure to compose SDLC with security concern can lead to several issues unpredicted problem, many organizations prioritize security as an afterthought using motto "patch and penetrate" strategy, resulting in increased cost and unpredictable timeline or even launch before it's ready (Humayun et al., 2022).

These various threats are categorized into issues by previous research (A. W. Khan et al., 2022), that claims there are 13 main cyber security issues and challenges faced by vendors and organizations from 67 research studies, the most common issues/challenges were related to 1) cyberattacks, 2) lack of right knowledge and 3) lack of management. In Table 2, it shows the three potential issues and its sub issues that organization will face if neglecting the secure software development lifecycle (S-SDLC) according to Khan et al. (2022) study. The author performs in-depth reviews with the potential impact to organization, firms and vendors.

The main issue / challenges faces is access of cyberattacks. The most sub issues of cyberattack frequent is injection type of attack such as SQL Injection surveyed by OWASP. It's the third ranked attack in 2021 and the number one cyberattack in 2020. When Software application gets SQL Injection, the impact of application is unpredictable.

Table 2. Top Ranking Cyber Security Issues

No	Issues	Sub-Issues
1	Cyber Attacks (Khan et al, 2022)	- SQL Injection - Broken Access Control - Distributed Denial of Service (DDoS)
2.	Lack of Knowledge (Khan et al, 2022)	- Third Party Integrating Vulnerabilities - Easier Reverse Engineering - Hardening Forensics/ Monitoring Failures - Cryptographic Failure and Encryption at rest Failure
3.	Lack of Management	- Insider Attack - Social Engineering - Misconfiguration Security

The hackers are able to retrieve all user data or even whole database, resulting in a data breach, which can be exacerbated by severe injection vulnerabilities. Additionally, the attacker may execute arbitrary changes using queries such as Insert, Alter, or even Drop query. If these actions occur, the potential financial losses for the company become highly unpredictable. Attackers could delete data, steal information, damage systems, and execute malicious commands, leading to significant financial, operational, or reputational damage to the company. several cyber-attacks that may occur to firms and organizations are Broken Access Control (Anas et al., 2024) dan Distributed Denial of Service (Karthikeyani & Karthikeyan, 2023; Singh & Gupta, 2022; Yuryna Connolly et al., 2020). and this cyber-attack is in line with OWASP 2021 report.

Broken Access Control (BAC) is a serious software application vulnerability stated by the previous research. BAC allows unauthorized users to bypass permissions and perform unauthorized actions leading to data breaches, breaking data integrity and privacy concerns. BAC enables user to be authenticated as another user or higher access user. lack of concern S-SDLC affecting software application has vulnerability in code or the weakness on user process causing user able to perform such actions (Anas et al., 2024). The Distributed Denial of Services (DDoS) technically cannot be eliminated, however it can be mitigated and reduced through various strategies that will be discussed in RQ2.

The second main issues of not implementing the Secure Software Development Life Cycle (S-SDLC): Lack of Knowledge, this is refers to a deficiency in understanding various aspect such as intellectual property rights, software products and third party application development domain which impacted to Unauthorized Access, Easier Reverse Engineering and Hardening Forensics / Monitoring Failures and Cryptographic Failure which reducing significant integrity of whole software products and it's organization.

The emerging of open-source libraries, components, software and application led some organizations, firms and vendors to rely on these services to build their software application efficiently and effectively. They also connect their software application to the service like online storage services, payment gateway services, and other services. These services are known as software, platform or code as services. However, they are unaware that the services they are integrating have security vulnerabilities in the integrating process that makes cyberattack scenario awaiting them. This security issue the authors refers to as Third Party Integrating Vulnerabilities. These vulnerabilities are sometimes fatal because unexperienced hacker can land cyber-attack without the organization knowing it and this becomes concern of some previous research in academia, technologies

founders and leading technology industry (Li et al., 2019; Zhan et al., 2021). Implementing S-SDLC may significantly reduce this Third-Party Integrating Vulnerabilities.

Reverse Engineering in context of software application, reverse engineering is a process of reconstruct and analysis an existing software application. This process is able to reconstruct the software structure, components, decoding source code, understanding the algorithms and documenting software key functions, therefore this technique could threaten an organization, firms and vendors that benefit their competitor to re-produce the software or crack the software which impacted to financial losses and integrity of respected organizations. Due to lack of knowledge, software applications become easier to perform reverse engineering, and the software application appears exposed to competitors and hackers. Reverse engineering with combination of security misconfiguration and S-SDLC may land fatal damage to an organization (Canavese et al., 2022).

Forensics in software engineering involves ensuring that software systems are capable of supporting digital investigations or in general forensics is a science process to investigate and verify claims to uphold the justice with non-repudation characteristics. Forensics activity in the context of software applications is performing logging, monitoring activities, ensuring data integrity and maintaining clear audit of spesific action in software application. These forensics in software application environment play a significant role to cyber-crime investigation, due to lack of knowledge, monitoring system and logging has become chaotic and hard for people to analyze and investigate the perpetrator and what kind of attack that prepetrator landing to organizations, firm and vendors, this investigation is essential for mitigating the issues occured. Some organizations are not even aware of this logging standardization in context of where, what kind of format and how long the logging is stored. worse than that this logging system doesn't even exist in software application which impacted to Hardening Forensics Process and Monitoring Failure or even worse, the attacker maintaining sustainable access to specific organization (Pasquale et al., 2018).

Cryptographic Failures, as mentioned by the Open Web Application Security Project (OWASP) refer to issues related to incorrect implementation or use of cryptographic systems that can lead to security vulnerabilities. These issues are included in the OWASP Top 10 list, which identifies the most critical and common security risks to software application. Cryptographic failures are caused by several events such as Insecure Cryptographic Storage, Outdated Algorithms, Improper Key Management, Encryption at Rest Failure, Insecure Transmission Protocol (Hazhirpasand & Ghafari, 2021). A previous study defines cryptographic failures as the incorrect usage of cryptography which can leave sensitive data vulnerable to exposure. The paper discusses instances such as the use of outdated cryptographic algorithms such as MD5 and SHA-1 which are known to be susceptible to attacks and highlights the need for using stronger alternatives like SHA-256 (Prasanna & Premananda, 2021). See Table 3 to prevent cryptographic failure issues.

Table 3. Cryptographic Failure Issues with its Prevention

Issues	Prevention
Cryptographic Failure	Encrypt Sensitive Data
	Proper Key Management
	Use Storing Hashing Functions such as bcrypt, scrypt and Argon2
	Avoid Deprecated Algorithms such as MD5 and SHA1
	Authenticated Encryption
	Disable Caching for Sensitive Data

Lack of Management refers to a critical challenge identified in Khan et al (2022) research indicating a deficiency in focusing requirements, managing issues, careless behavior of developers which related to insider threat.

Business Processes (BP) are considered cornerstone of organization and lack of management refers to a critical challenge identified in study by Khan et al (2022). These processes are often translated into software applications, and they are non-risk-free in terms of software security. An example is the attack of compromise of a business process, which takes advantage of system loopholes what is commonly known in cyber security Insider Attack, this loophole instead of being reported, however it's exploited.

In the study of Khan et al, they are analyze 121 studies and found 424 best practices that may help organization for developing a secure software application, one of the essential phase that may define a sofware application for organization is Requirement Engineering, therefore the authors filter the Secure Requirement Engineering Practice (SREP) from Khan systematic literature review study based on the most top frequency used in 121 primary study, which is identified as SREP1 ranked first, SREP2 ranked 2 and SREP 4 ranked 3 (R. A. Khan et al., 2022).

Table 4. Secure Requirement Engineering Practices.

SREP1	**Develop Threat Modelling (Freq : 25)**
SREP1.1	Perform STRIDE
SREP1.2	Include security requirements as part of defining functional requirements
SREP 1.3	Perform DREAD
SREP 1.4	Understand and Incorporate Compliance and Regulatory requirements
SREP2	**Security Requirement Elication Practices (Feq : 31)**
SREP2.1	Elicit and categorize safety and security requirements

SREP2.2	Take into consideration organizational and political issues
SREP2.3	Use scenarios to elicit sensitive data and communication in terms authentication, authorization, privacy, system maintenance
SREP2.4	Identifty Stakeholders.
SREP2.5	Identify the operationg environment of system.
SREP11	**Methods used in Security RE (Freq : 42)**
SREP11.1	UMLSec, SecureUML
SREP11.2	Secure Troops
SREP11.3	Abuse Cases
SREP11.4	Structure Object Oriented Formal Language
SREP11.5	Machine Learning Techniques
SREP11.6	Fuzz-Analytic Hierarchy Process
SREP11.7	Security Requirement Engineering Approach
SREP11.8	Problem Frames
SREP11.9	Tropos (i' framework)
SREP11.10	Create and describe Misuse Cases

In Table 4 there are SREP that the authors filter it by top frequency and select to retrieve the keyword STRIDE, DREAD, Identification and Threat, in reason that the authors want to convert SREP to actionable checklist that must be employed within SSDLC from SREP 1.1 to SREP 1.4 and Practice List (PL) and make an checklist shown in Table 5 and after that summarize and categorize it into Table 6 for easier implementation purpose.

Table 5. Secure Requirement Practice Checklist

No.	Checklist
PL1 SREP1.1-1.4	Have the organization's components/services been conceptually mapped along with other connected applications?
PL2 SREP1.1-1.4	Have all organizational assets or resources connected to the software application been identified and listed?
PL3 SREP1.1-1.4	Has threat modeling, including specific cyber-attack scenarios, been defined for the software application?
PL4 SREP2.1	Is the Security QA team aware of the system's security requirements?
PL5 SREP2.1	Is the product manager aware of the system's security requirements?
PL6 SREP2.2	Have the Security QA and Product Manager listed the types of risks, severity levels, nature, security priorities, etc.?
PL7 SREP2.1	Have all security requirements been defined and categorized?
PL8 SREP2.2, SREP2.4	Have the Security Quality Assurance and Product Manager prepared a report on potential cyber-attack scenarios and their consequences if security requirements are neglected?
PL9	Has stakeholder analysis (using a Power-Interest chart) been conducted to facilitate the implementation of security requirements?
PL10	Has the Product Owner been informed of the reports?
PL11	Have high-influence stakeholders been informed about the reports?
PL12	Have security requirements been revised to mitigate the identified risks?
PL13	Have the revised security requirements been incorporated into the functional requirements, including Security Acceptance Criteria?
PL14 SREP1 1.1 SREP1 1.10	Are personnel responsible for modeling and design implementing UMLSec, Secure UML, or SecureTroops?
PL15 SREP1 1.5 SREP1 1.16	Are personnel involved in modeling and coding aware of common AI/ML implementations to address cyber-attack scenarios?
PL16 SREP 11.17	Have security requirements been defined as functional requirements?
PL17 SREP 11.10 SREP1 1.13 SREP2.2 SREP2.4	Have the Security Quality Assurance and Product Manager finalized the report on potential cyber-attack scenarios and the consequences of neglecting each requirement?
PL18 SREP1 1.10 11.3	Does the report include misuse cases and abuse cases?

Table 6. Summary of Secure Requirement Checklist

1	Identification
SREC 1	Have the components/services of the organization been conceptually mapped along with other connected applications?
SREC 2	Have all organizational assets or resources connected to the software application been identified and listed?
2	**Reports**
SREC 3	Has threat modeling and its specific cyber-attack scenarios been defined for the software application?
SREC 4	Have the Security QA and Product Manager completed the listing of risks, including their type, severity level, nature, security priority, etc.?
SREC 5	Have the Security QA and Product Manager created a report on potential cyber-attack scenarios and the consequences of neglecting each defined security requirement?
SREC 6	Have the Security QA and Product Manager finalized the report on potential cyber-attack scenarios, along with the consequences of neglecting each defined requirement?

SREC 7	Does the report include misuse cases and abuse cases?
3	**Awareness and Informed**
SREC 8	Is the Security QA aware of the security requirements for the system?
SREC 9	Is the Product Manager aware of the security requirements for the system?
SREC 10	Has stakeholder analysis with a Power-Interest chart been performed to support the implementation of security requirements?
SREC 11	Has the Product Owner been informed about the reports?
SREC 12	Have high-influence stakeholders been informed about the reports?
SREC 13	Do the reports include frameworks such as UMLSec, SecureUML, or Secure Tropos?
SREC 14	Are individuals involved in modeling and coding aware of common AI/ML implementations to handle cyber-attack scenarios?
4	**Requirements**
SREC 15	Have all security pre-requirements been defined and categorized?
SREC 16	Have the security requirements been restructured to mitigate the identified risks?
SREC 17	Have the security requirements been incorporated into the overall requirements with Security Acceptance Criteria as the final functional requirement?
SREC 18	Have the security requirements been defined alongside functional requirements?

To answer the research question in Secure Coding Practice to avoid cyber-attack security we utilize Open Web Application Security Project (OWASP) guidelines in Secure Coding Practices.

In OWASP Secure Coding Practice there are 14 Section. 1) Input Validation, 2) Output Encoding, 3) Authentication and Password Management, 4) Session Management, 5) Access Control, 6) Cryptographic Practices, 7) Error Handling and Logging, 8) Data Protection, 9) Communication Security, 10) System Configuration, 11) Database Security, 12) File Management, 13) Memory, 14) General Coding Practices. From these 14 sections the authors determine to make checklist from Input Validation Secure Coding Checklist in Table 7, Output Encoding (OE) in Table 8, Authentication and Password Management Secure Coding Checklist in Table 9, Access Control Secure Coding Checklist Table 10, Cryptographic Secure Coding Checklist in Table 11.

Table 7. Input Validation Secure Coding Checklist

Sources	No.	Checklist
OSCP IV 1&2	1	Are all input fields in the web application's form tags validated on the front end?
OSCP IV 1&2	2	Is the web application's API hosted on a trusted server?
OSCP IV 1 & 2	3	Is the web application's API validated for each input submitted through form tags?
OSCP IV 1 & 2	4	Are all file upload forms in HTML strictly validated to permit only allowed file types and formats?
OSCP IV 3	5	Does the web application have a centralized file location to handle acceptable input formats?
OSCP IV 4, 5, 8	6	Are all input and output data in both the web API and front-end interface properly encoded in UTF-8?
OSCP 7	7	Is the input and output from the web application's API and interface encoded in UTF-8?
OSCP 6	8	Does the web API application use a centralized template for handling input rejection, complete with standardized messages (e.g., middleware or centralized file)?
OSCP IV 8	9	Are HTTP headers, such as Cookies, User-Agent, Referer, and others, validated and sanitized?
OSCP IV 8	10	Are URLs and parameters sanitized before being sent to the web API?
OSCP IV 8	11	Are URLs and parameters validated and sanitized upon receipt by the web API?
OSCP IV 9	12	Are all HTTP header responses to the API validated and restricted to the ASCII character set?
OSCP IV 10	13	Is redirect URL data validated before being processed or executed?
OSCP IV 11, 12, 13	14	Are all HTML form inputs validated for data type, range, and length prior to submission?
OSCP IV 14	15	Does the front end enforce correct input types using appropriate HTML form elements?
OSCP IV 15 & 16	16	If special or hazardous characters must be used as input, is there a specific function in a centralized file to handle them?

Table 8. Output Encoding (OE) Secure Coding Checklist

Sources	No.	Checklist
OSCP OE 1	18	Is all output encoding performed on the server side, ensuring that no encoding logic is delegated to the client side?
OSCP OE 2	19	Is there a standardized and well-tested encoding library used for all web application output encoding operations?
OSCP OE 3,5	20	Are encoding and decoding operations for both the web application's API and interface set to UTF-8?
OSCP OE 6	21	Have all inputs to database queries been properly sanitized?
OSCP OE 4, 5, 7	22	Have all inputs and outputs connecting the web application system to third-party applications been sanitized?

Table 9. Authentication & Password Management Secure Coding Checklist

Sources	No.	Checklist
OSCP APM 23, 26, 27	23	Are all authorized pages protected using a middleware-based strong authentication system?
OSCP APM 23	24	Does the authentication system in the middleware comply with security standards?
OSCP APM 23 & 24	25	Is the authentication API system hosted on a trusted server?
OSCP APM 23& 24	26	Is your server provider using secure communication protocols (e.g., HTTPS)?
OSCP APM 23	27	Is your server accessible?
OSCP APM 28, OSCP33	28	Does the web application API ensure that authentication failure messages do not reveal specific failure details?
OSCP APM 28	29	Does your web API properly clear all session data upon logout?
OSCP APM 28, OSCP41	30	Does your web API enforce an entry rate limitation for authentication attempts?
OSCP APM 30 & 31	31	Does the web API use strong hashing functions with a write-only method for storing sensitive data?
OSCP APM 32	32	Does the web API sequentially process all authentication inputs it receives?
OSCP APM 34	33	If your web API is connected to external systems, is there an authentication system in place for those connections?
OSCP APM 36	34	Are HTTPS POST requests used to transmit authentication credentials?
OSCP APM 37	35	If your system transmits temporary passwords, are they encrypted, transmitted over HTTPS, and accessible only to authorized users?
OSCP APM 38 & 39	36	Does your web API enforce specific password complexity requirements?
OSCP APM 40	37	Does your web interface obscure password entries on the user screen?
OSCP APM 53	38	Are user login and logout events logged into a temporary database system?
OSCP APM 53	39	Are user login and logout events logged with details such as a timestamp, user ID, IP address, and location?

Table 10. Access Control Secure Coding Checklist.

Sources	No.	Checklist
OSCP AC 1	40	Does your web application system rely on server-side objects or tokens to make access control decisions?
OSCP AC 1	41	Does your web application system sanitize all inputs used in authorization decisions at both the front-end and back-end?
OSCP AC 1	42	Is the web application framework utilizing built-in session management to ensure sessions are securely stored and transmitted?
OSCP AC 1	43	Is your web application system using a trusted and authorized built-in session management library?
OSCP AC 3	44	Does your web application system implement a "deny by default" policy?
OSCP AC 3	45	Does your web application log access control failures for monitoring and auditing purposes?
OSCP AC 4	46	Does your web application access control use exception handling to manage errors and prevent exposure of internal system details?
OSCP AC 4	47	Is there a monitoring system in place to detect configuration access failures?
OSCP AC 7	48	Does your web application prevent Path Traversal Attacks by properly validating and sanitizing file paths?
OSCP AC 7	49	Are file access restrictions enforced to protect sensitive files (e.g., .env, .git, config.php, package.json)?
OSCP AC 11	50	Does your web application follow the principle of least privilege, ensuring users and services only have the necessary access?
OSCP AC 12	51	Is sensitive data securely encrypted using AES for storage and bcrypt for password hashing?
OSCP AC 13	52	Does your web application implement secure data retention policies?
OSCP AC 14	53	Is your web application environment restricted to allowing admin panel and server access only to authorized personnel?
OSCP AC 15	54	Does your web application code enforce consistent access control between the back end and front end?
OSCP AC 15	55	Are critical user interface security actions enforced by server-side code?
OSCP AC 15	56	Is your access control logic centralized in a single, consistent logic file, avoiding duplication?
OSCP AC 18	57	Does your web application enforce rate limits for sensitive operations such as login attempts, financial transactions, or crucial API calls ?
OSCP AC 18	58	Does your web application implement CAPTCHA or MFA for high-risk actions to prevent automated attacks?
OSCP AC 18	59	Does your web application include logging and analytics to detect abnormal patterns, transactions, or activities?

OSCP AC 19	60	Does your web application use supplemental headers for authorization, such as CSRF protection and session validation?
OSCP AC 20	61	Does the web application system have a mechanism to force users to re-authenticate when their privileges change?
OSCP AC 20	62	Does your web application automatically log out users after a period of inactivity?
OSCP AC 22	63	Does your web application enforce session termination when an account is deactivated?
OSCP AC 22	64	Does your web application's logout feature properly invalidate session tokens to prevent reuse?

Table 11. Cryptographic Failures Secure Coding Checklist

Sources	No.	Checklist
OSCP CP 1	65	Should cryptographic operations be performed on a secure, trusted server rather than on the web application client side?
OSCP CP 1	66	Are cryptographic keys securely stored in a dedicated key management system instead of being embedded within application code or configuration files?
OSCP CP 2	67	Are your web application secrets stored securely and never hardcoded in the source code?
OSCP CP 2	68	Are secrets encrypted at rest using strong encryption algorithms such as AES-256?
OSCP CP 2	69	Are secrets encrypted during transit using TLS 1.2 or higher, including TLS 1.3?
OSCP CP 3	70	Does the web application implement secure error handling to prevent sensitive cryptographic errors from being exposed?
OSCP CP 3	71	Are system logs configured to capture cryptographic failures without exposing sensitive information?
OSCP CP 4	72	Are all random values (e.g., random numbers, GUIDs, filenames, session tokens) generated using a cryptographically secure random number generator (CSPRNG) rather than non-secure functions such as rand() in PHP or Math.random() in JavaScript?

IV. CONCLUSION

Recently organizations, government and vendors are evolving their information technology (IT) infrastructure and trying to utilize digital platform such as applications, artificial intelligence or automation process where this rapid development is a great things, however, this rapid development is not linearly followed by protection in IT application security, in this study the authors encourage utilization of secure software development where the authors emphasize security requirement practices (SREP) and security coding practices (SCP) that has been compose by author from international journal articles and international standard (Open Web Application Project), the authors also provide practical contribution by combining and summarizing the practices into actionable checklist. However, this study has certain limitations such as limitation applicability to specific regional area context, the actionable checklist hasn't been validated in real-world scenarios leaving room for further refinement and testing.

REFERENCES

Anas, A., Elgamal, S., & Youssef, B. (2024). Survey on detecting and preventing web application broken access control attacks. *International Journal of Electrical and Computer Engineering (IJECE)*, *14*(1), 772–781.

Ayemowa, M. O., Ibrahim, R., & Khan, M. M. (2024). Analysis of Recommender System Using Generative Artificial Intelligence: A Systematic Literature Review. *IEEE Access*.

Baz, A., Ahmed, R., Khan, S. A., & Kumar, S. (2023). Security risk assessment framework for the healthcare industry 5.0. *Sustainability*, *15*(23), 16519.

Canavese, D., Regano, L., & Lioy, A. (2022). Computer-Aided Reverse Engineering of Protected Software. *International Workshop on Digital Sovereignty in Cyber Security: New Challenges in Future Vision*, 3–15.

de Vicente Mohino, J., Bermejo Higuera, J., Bermejo Higuera, J. R., & Sicilia Montalvo, J. A. (2019). The application of a new secure software development life cycle (S-SDLC) with agile methodologies. *Electronics*, *8*(11), 1218.

Fitriani, R., Subagiyo, R., & Asiyah, B. N. (2023). Mitigating IT Risk of Bank Syariah Indonesia: A Study of Cyber Attack on May 8, 2023. *Al-Amwal: Jurnal Ekonomi Dan Perbankan Syari'ah*, *15*(1), 86–100.

Gwon, Y. N., Kim, J. H., Chung, H. S., Jung, E. J., Chun, J., Lee, S., & Shim, S. R. (2024). The Use of Generative AI for Scientific Literature Searches for Systematic Reviews: ChatGPT and Microsoft Bing AI Performance Evaluation. *JMIR Medical Informatics*, *12*, e51187.

Hazhirpasand, M., & Ghafari, M. (2021). Cryptography Vulnerabilities on HackerOne. *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, 18–27.

Humayun, M., Jhanjhi, N., Almufareh, M. F., & Khalil, M. I. (2022). Security threat and vulnerability assessment and measurement in secure software

development. *Comput. Mater. Contin*, *71*, 5039–5059.

Inggarwati, M. P., Celia, O., & Arthanti, B. D. (2020). Online single submission for cyber defense and security in Indonesia. *Lex Scientia Law Review*, *4*(1), 83–95.

Javed, Y., Khayat, M. A., Elghariani, A. A., & Ghafoor, A. (2023). PRISM: a hierarchical intrusion detection architecture for large-scale cyber networks. *IEEE Transactions on Dependable and Secure Computing*, *20*(6), 5070–5086.

Karthikeyani, R., & Karthikeyan, E. (2023). A Review on Distributed Denial of Service Attack. *Asian Journal of Research in Computer Science*, *16*(4), 133–144.

Khan, A. W., Zaib, S., Khan, F., Tarimer, I., Seo, J. T., & Shin, J. (2022). Analyzing and evaluating critical cyber security challenges faced by vendor organizations in software development: SLR based approach. *IEEE Access*, *10*, 65044–65054.

Khan, R. A., Khan, S. U., Khan, H. U., & Ilyas, M. (2022). Systematic literature review on security risks and its practices in secure software development. *Ieee Access*, *10*, 5456–5481.

Kitchenham, B. A. (2012). Systematic review in software engineering: where we are and where we should be going. *Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies*, 1–2.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and Software Technology*, *51*(1), 7–15.

Li, Y., Ma, L., Shen, L., Lv, J., & Zhang, P. (2019). Open source software security vulnerability detection based on dynamic behavior features. *Plos One*, *14*(8), e0221530.

Pasquale, L., Alrajeh, D., Peersman, C., Tun, T., Nuseibeh, B., & Rashid, A. (2018). Towards forensic-ready software systems. *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, 9–12.

Prasanna, S. R., & Premananda, B. S. (2021). Performance analysis of md5 and sha-256 algorithms to maintain data integrity. *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, 246–250.

Rajamäki, J., Wood, K., & Espada, B. (2024). LOCKing Patient Safety: A Dynamic Cybersecurity Checklist for Healthcare Workers. *European Conference on Cyber Warfare and Security*, *23*(1), 811–815. https://doi.org/10.34190/eccws.23.1.2072

Sadler, H. (2020). ER2C SDMLC: enterprise release r isk-centric systems d evelopment and maintenance life cycle. *Software Quality Journal*, *28*(4), 1755–1787.

Saeed, H., Shafi, I., Ahmad, J., Khan, A. A., Khurshaid, T., & Ashraf, I. (2025). Review of Techniques for Integrating Security in Software Development Lifecycle. *Computers, Materials & Continua*, *82*(1).

Singh, A., & Gupta, B. B. (2022). Distributed denial-of-service (DDoS) attacks and defense mechanisms in various web-enabled computing platforms: issues, challenges, and future research directions. *International Journal on Semantic Web and Information Systems (IJSWIS)*, *18*(1), 1–43.

Yuryna Connolly, L., Wall, D. S., Lang, M., & Oddson, B. (2020). An empirical study of ransomware attacks on organizations: an assessment of severity and salient factors affecting vulnerability. *Journal of Cybersecurity*, *6*(1), tyaa023.

Zhan, X., Fan, L., Chen, S., We, F., Liu, T., Luo, X., & Liu, Y. (2021). Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 1695–1707.