Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 9, 2025

# INTEGRATING FORMAL METHODS INTO CLOUD BASED LEARNING MANAGEMENT SYSTEM SECURITY: A LIGHTWEIGHT FORMAL DEVELOPMENT FRAMEWORK

**Shafiq Hussain[1], Muhammad Jamil[*2], Imran Shehzad[3], Azka Sarfraz[4]**

[1]*Department of Computer Science, University of Sahiwal, Pakistan (drshafiq@uosahiwal.edu.pk)*
[*2,3]*Department of Computer Science, COMSATS University Islamabad, Sahiwal Campus, Pakistan*
[4]*Department of Computer Science, Lahore College for Women University, Lahore, Pakistan*

[*1]jamil138.amin@gmail.com, [2]drshafiq@uosahiwal.edu.pk, [3]maniedulive@gmail.com, [4]azkasarfraz81@gmail.com

DOI:https://doi.org/10.5281/zenodo.17311145

**Keywords**
*Cloud LMS Security*
*Threat Analysis Framework*
*Formal Methods*
*STRIDE model*
*DREAD model*
*Event-B*

**Article History**
*Received: 07 July 2025*
*Accepted: 07 September 2025*
*Published: 22 September 2025*


**Corresponding Author: ***
**Muhammad Jamil**

**Abstract**
*The focus of Cloud LMS security extends beyond external protective measures, emphasizing the need for robust internal security within Cloud LMS applications, where vulnerabilities often originate. Many securities issues stem from flaws in software design, making internal security more critical than external defenses. Numerous techniques have been developed to tackle these internal security challenges, with threat modelling being a prominent approach. Threat modelling, conducted in the early stages of software development, identifies potential threats to software systems and suggests mitigation strategies. Existing methodologies, such as the Secure Development Lifecycle (SDL) and Threat Analysis Framework (TAM), often use informal or semi-formal approaches, which can introduce inconsistencies and ambiguities in system design. Formal methods, grounded in mathematics, are employed for the specification, analysis, and design of software systems, enabling precise analysis and verification at the design level. This leads to the creation of more accurate, reliable, and consistent software systems. This paper introduces a novel framework based on the lightweight application of formal methods using the STRIDE model and DREAD models, which integrates threat modelling to identify security requirements early in the development process and incorporates these security properties into the software design. The design is then analyzed, verified, and validated using formal methods including Event-B and RODIN.*

## INTRODUCTION

The swift development in Cloud LMS computing has led to an increased risk of security breaches as software applications become more vulnerable to attacks from increasingly professional security attackers [1]. The dependence of devices on computers, networks, and internet has made security an important part of the system development [2]. Traditionally, security is something that is an afterthought, not being considered an integral part of the development process and with the main focus still being on the functional aspects of systems. Furthermore, the efforts at security focused less on protective mechanisms and

more on what should be viewed as basic requirements [3]. But the modern sophistication of cyber threats and the increasing importance of software to our daily lives now mandate that security properties are baked in at design time. The root cause of many security issues is vulnerabilities in the software design, which is why internal security becomes paramount over external defenses. There have been several ways to overcome these internal security threats and threat modelling is one of such means [4]. Threat modelling is a process of identifying all possible threats that software systems are facing. The software's design can expose it to vulnerabilities which are a threat to it [5]. If threat modelling is conducted early, all the potential threats as well as the ways with which they can be addressed are identified. Instead of reining them, it takes a proactive approach to handle security issues before they get to them rather than trying to address them after the system is out there.

Most of the existing methodologies regarding threat modelling like Security Development Lifecycle (SDL) or Threat Analysis Framework (TAM) go up to using informal or semi-formal approaches for threat modelling, causing inconsistencies and ambiguities in the system design process [6]. Formal methods are based on math and thus take a more rigorous look at the specification, analysis and design of software systems which can be precisely analyzed and verified at the level of design [7]. So, we get more accurate, reliable, and consistent software systems. In this paper, the authors presented a novel framework that uses the lightweight use of formal methods with the STRIDE model and DREAD models including threat modelling to establish security requirements early in the development process and incorporate these security properties into the software design. Formal methods such as Event-B and RODIN are then used to analyze, verify and validate the design. The rest of the paper is organized as follows: Section 2 presents the related work, Section 3 introduces formal methods for securing Cloud LMS, Section 4 provides a summary of the security properties of Cloud LMS, Section 5 elaborates on the security models of Cloud LMS, Section 6 discusses the proposed framework, and Section 7 concludes the paper.

## 2. Related Work

With the fast development and broad adoption of Cloud LMS computing, new security issues have arisen beyond the traditional security approaches focused on just the perimeter [8]. Data and applications in the Cloud LMS run on remote servers and are hosted over the Internet and can be threatened by a lot of cyber-attacks [9]. Cloud LMS security must transcend picking locks from the outside and must be about protecting the inside of Cloud LMS where vulnerabilities are proximate most typically [10]. Fundamentally, many security problems are caused by bad software design and so the Cloud LMS application security needs to focus both on external and internal threats [11]. Too many of the security challenges in software development have been addressed with the Secure Development Lifecycle (SDL) [12] and the Threat Analysis Framework (TAM) [13]. An SDL is a process in which security practices are injected at several stages of the software development life cycle, from requisition to deployment and preservation [14]. TAM, however, is a framework dedicated to describing threat modelling, a technique to identify and remove threats to software systems [15]. Although these approaches have been shown to be effective, they are often imprecise and require the usage of informal or semi-formal methods that introduce an indirect use of metrics and hence an incorrectness in the system design [16].

The specification, analysis and design of software systems is performed with a more rigorous approach using formal methods, that are based on mathematics [17]. These techniques allow for accurate analysis and verification of the properties of the software at the design level and hence result in the creation of more accurate, reliable and consistent software systems. As a solution to the limitations of present methodologies such as the SDL and TAM, the formal methods application in the software security domain can assist in the formal modelling of threats and the inclusion of security requirements into the software design [18]. Two traditional threat modelling frameworks in the industry and academia are Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (STRIDE) and Damage potential, Reproducibility, Exploitability, Affected users, Discoverability (DREAD) [19]. Classification of threats in software systems is among

other already mentioned needs and the proposed threat classification scheme, i.e. the STRIDE model, helps determine potential threats to a software system and the proposed threat classification scheme, i.e. the STRIDE model, provides a structured way to quantify and prioritize such threats [20]. The formal development methodology provides a means of integrating these threat modelling frameworks to help identify, analyse, and mitigate security vulnerabilities in Cloud LMS [21]. The STRIDE Average Model is an enhancement to the STRIDE model which divides software applications into five categories and rates them. The type of data software works with dictates how much information criticality they require. The potential security risk is then calculated by taking the everywhere average (STRIDE average) of the above items. Nevertheless, no automated tool support is available for this approach [22].

One of the other threat modelling techniques is attack trees, whereby attacks are structured as trees, in which leaves are different attack paths, while nodes represent goals. The leaf nodes can be given values to evaluate the security of the goal [23]. The trees can be extended to include other attributes (i.e. time operational cost). Fuzzy logic-based threat modelling uses fuzzy set theory and fuzzy inference engine, to process input variables from the STRIDE model and identify related threats [24]. SDL and TAM threat modelling tools, from Microsoft based on STRIDE and DREAD, take a DFD as input and produce reports about threats and strategies for response [25]. Correctness-by-Construction goes beyond formal methods and uses them in application to the more critical and earlier parts of software engineering: to clarify and specify the system, as well as design the system. This approach does not advise the use of formal methods in all phases, but it does recommend their use on security-critical components [26].

### 3. A Cloud LMS -based LMS (Learning Management System)

A Cloud LMS -based LMS (Learning Management System) makes it possible to create, administer, deliver, and track training or educational information via the internet. It does away with the requirement for in-house IT management, complicated infrastructure, or local installations. This architecture facilitates remote learning and international collaboration by enabling users—students, instructors, and administrators—to access the system at any time, location, and on any internet-connected device. Compared to conventional on-premise systems, Cloud LMS -based learning management systems provide many benefits. However, creating such learning management systems is difficult and calls for cutting-edge tools like formal methods. This study presents a novel approach to the design and development of Cloud LMS -based learning management systems that integrates formal approaches.

### 4. Formal Methods in Cloud LMS

In other words, formal methods are a rigorous, mathematically grounded set of techniques for software system specification, analysis, design, testing, and implementation [27]. Formal methods use precise unambiguous mathematical language to develop applications which are not contain inconsistency or errors. In the context of Cloud LMS computing where data and applications are enabled in remote hosting on shared infrastructure, this is particularly important as it goes beyond traditional perimeter-based defenses [28]. Once such models have been constructed, they may be analyzed, validated, and verified in great detail, using the support of highly developed tools like theorem provers and model checkers [26]. This makes specification flaw correction before the implementation phase possible and the systematic checking of consistency of specifications possible. The specific requirements of the system determine the use of: Theorem provers and model checkers, individually or in combination. Compared to conventional testing techniques, formal methods provide a major advantage through their ability to analyze applications under a large spectrum of data scenarios that will otherwise not be possible due to exhaustive property testing [25]. Moreover, formal methods provide a strong approach to verifying the correctness and robustness of software systems used in Cloud LMS deployed applications, against security issues [29].

However, formal methods rely on a significant amount of expertise, which has prevented them from being widely applied to all domains except perhaps some critical systems in the computer science domain. Example formal methods include Event-B, Z, VDM++, RODIN, Alloy, Z/EVES, Isabelle and Coq.

These techniques can be broadly categorized into three main groups: This work focuses on formal specification languages, model checkers, and theorem provers. System requirements and design as well as the generation of test cases are formalized in a formal specification language, typically Z. The use of model checkers is to validate and verify properties and in this case to spot deadlocks. In contrast, Theorem provers are used to formally prove that properties such as syntax, type checking, domain checking, invariant check, and proof obligations are all ok. Because of the severe consequences of software flaws in the Cloud LMS, formal methods are a tool of great rigor and mathematics that can be effectively used to develop secure and reliable Cloud LMS applications.

**4. Security Properties of Cloud LMS Apps**

In the case of Cloud LMS computing, where data and applications are being run on remote, shared infrastructure, the security properties of Cloud LMS - based software system become more important than ever. The vulnerabilities in Cloud LMS application can be used by attackers to exploit the weaknesses to control valuable system resource and cause the critical operations to fail.

Measures implemented to protect against exploitation of vulnerabilities, are known as security properties. A vulnerability is a weakness in the system that can be exploited by the attacker in order to compromise some of the system resources. System resources that include valuable (sensitive data, computing power, etc.) entities that must be protected in such a manner to ensure the secure operations.

A threat refers to a potential attack that could occur, and security properties are generally classified into six key categories: Authentication, authorization, confidentiality, integrity, availability and non-repudiation. The User or Process identification is stored securely and user identification is confirmed through the authentication process. Authorization determines what parts of the system a user has access to, but not how to access them. Read access to data is governed by the confidentiality. Write access is managed by integrity and it is verified that only specific authorized individuals should have access to write data appropriately. Availability makes sure system resources are available to authorized users when they need them, and the Cloud LMS based application works as expected. Non repudiation prevents users from denying their actions, which relies on robust logging mechanism to track these events.

Other security properties and mechanisms are available in the industry but in this paper specifically, we focus on the core properties listed above which are also necessary to the development of secure and robust Cloud LMS that can cope with the evolving threat landscape.

**5. Security Models for Cloud LMS**

This research is grounded in fundamental security models that have been developed and refined by experts over the years. Among these, the STRIDE and DREAD models are the most widely utilized.

**5.1 STRIDE Model**

The STRIDE model categorizes six major types of threats that an application might encounter: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. Spoofing occurs when an unauthorized user obtains another user's identification details and impersonates them to gain access to system resources. Tampering involves the unauthorized alteration of system data. Repudiation refers to the denial of an action performed by an authorized user, often due to weak or non-existent logging systems. Information disclosure happens when confidential data is exposed to users who are not authorized to access it. Denial of Service (DoS) attacks make system resources unavailable to legitimate users. An elevation of Privilege is the gain in unauthorized access to higher level resources by a user that has limited access.

**5.2 DREAD Model**

In this model, the risk levels of the threats found by STRIDE model are assessed using DREAD model. DREAD evaluates threats across five categories: Damage Potential, Reproducibility, Exploitability, Affected Users, and Discoverability. The system designer or user assigns these parameters values and the DREAD algorithm works out the risk level. Based on that, threats are classified as High or Medium or Low. Control must be robust for high-risk threats, moderate for medium risk threats, and simple for low-risk threats. And in some cases, there are no controls on what is acceptable as a threat.

## 6. Proposed Framework

A new framework based on lightweight application of formal methods is composed of four phases. The structure of this framework is illustrated in Fig.1. It employs a lightweight approach to formal methods, under which formal methods are applied selectively to certain activities within the framework rather than across all phases. The core idea of this framework is to make formal methods accessible to software engineers for the development of secure Cloud LMS applications. In this framework, Event-B is used as specification languages for modelling, while the RODIN platform is utilized for the analysis, verification, validation, and formal proofs of the models. The ProB model checker is used to ensure invariant preservation and deadlock-freeness in the system models. The framework encompasses seven phases: requirements gathering, abstract formal models of security properties, threat modelling, system development, verification of secure models, model checking, and validation of secure models.

### 6.1 Requirements Engineering

#### 6.1.1 System Requirement Specification

During the requirement specification phase, the system under development is thoroughly analyzed. This involves the identification of system entities, outlining assumptions, and documenting both functional and non-functional requirements in English. Additionally, the necessary properties for validation are also defined during this phase.

#### 6.1.2 System Data Flow Diagram

A comprehensive Data Flow Diagram (DFD) of the system is created in this phase. This DFD serves as the input for the SDL Threat Modelling tool, which uses it to generate a list of potential threats to the system.

### 6.2 Threat Modelling

#### 6.2.1 Threat Modelling using STRIDE Model

The STRIDE model is applied to the system's Data Flow Diagram (DFD). This automated process uses the SDL Tool, which integrates the STRIDE model to evaluate each element of the DFD, resulting in a comprehensive list of potential threats to the system.

#### 6.2.2 Threat Assessment using DREAD Model

The DREAD model is then used to assess the threats identified. Each threat is rated for severity using the DREAD algorithm, categorizing them as high, medium, or low risk.

#### 6.2.3 Threat Assessment using DREAD Model

In this step, the threats identified and rated in the previous phases are reviewed. Any threats based on assumptions are eliminated, resulting in a refined threat model for the system.

### 6.3 Formal Models of Security Properties

Since as this framework relies upon formal procedures to ensure that Cloud LMS are securely created, a formal model of security properties in the way of Event-B is developed at the high level of abstraction. Essential security properties are modelled as abstraction in this abstract model without which attacks based on system vulnerabilities will not be possible. The framework addresses the security properties of confidentiality, integrity, availability, authentication, authorization, and non-repudiation respectively.

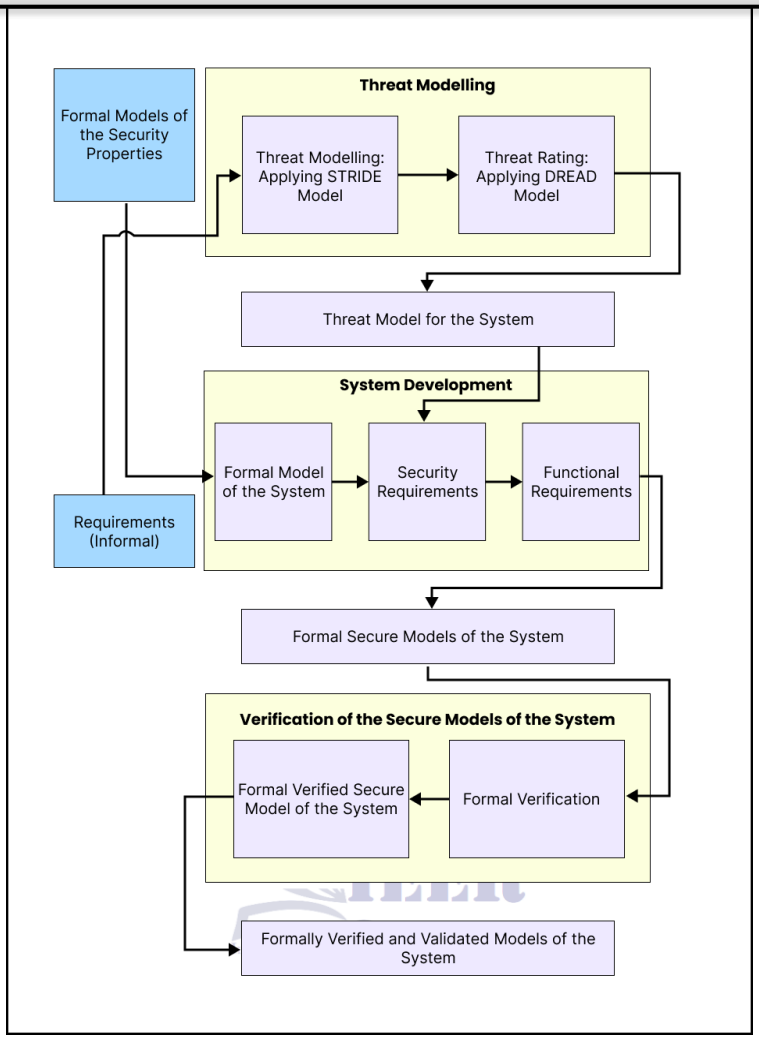


**Fig. 1. Proposed Framework**

**System Development**

**Basic System Models in Event-B:** In this phase, a basic formal model of the system is developed using Event-B. The system requirements along with the abstract formal models of security properties, serve as inputs. The essential functionality needed to implement both security and functional properties is incorporated.

**First Refinement: Incorporating Security Properties:** Next, the basic formal model is refined to include additional functionality. This refinement integrates security properties derived from the threat model developed. The goal of refinement is to begin with simple operations and gradually introduce more complexity in subsequent stages.

**Second Refinement: Integrating Functional Properties:**

In this phase, the model refined is further refined by adding the functional properties. These functional properties describe the system's behavior.

**Final Secure Formal Models in Event-B:** At this stage, the secure formal models of the system are completed in Event-B, incorporating all necessary security and functional properties based on the refinements made in previous phases. For this framework, two refinements are typically sufficient, although further refinements may be required depending on the complexity of the system. For a simple Cloud LMS application with minimal functional requirements, these two refinements are usually adequate.

**Verification of the Secure Models of the System**
In this phase, the secure formal models are verified using theorem provers. For this framework, Altelier B theorem provers, integrated with the RODIN tool, are employed to verify the secure models. These provers generate proof obligations that must be resolved for the system to function correctly. The user initiates the Altelier B provers, which automatically resolve most proof obligations in the initial run. If some obligations remain unresolved, additional strategies are applied, including modifications to the formal models based on the system requirements.

**Model Checking**
During this phase, the secure formal models developed are subjected to model checking using the ProB model checker, integrated with the RODIN tool. This framework focuses on checking for invariant preservation and deadlock-freeness within the system models. If the system successfully passes both tests, it is deemed acceptable. Otherwise, modifications are made to the formal models based on the system's requirements.

**Validation of Secure Models of the System**
In the final phase of this framework, the properties defined in the requirements phase including both security and functional properties are validated. Validation is performed using the AnimB animator, another tool for the RODIN platform. The constants and variables in the formal models of the secure system are assigned values to make them animable by AnimB. Once AnimB is initiated, the system's behavior is observed. If the behavior aligns with the properties specified in the requirements phase, the models are considered validated. If discrepancies arise, the formal models are revised in consultation with the system requirements.

## 7. Implementation and Results

To demonstrate the effectiveness of the proposed lightweight formal development framework, we implemented a prototype Cloud LMS -based file-sharing application incorporating essential security properties. The implementation followed all phases of the framework, from requirement specification to model validation, utilizing the Event-B formal method, RODIN platform, and associated tools like ProB and AnimB.

### 7.1 Use Case Description

The selected use case was a secure file-sharing system allowing authenticated users to upload, download, and manage documents. The application enforces access control policies and logs user activities. This system is representative of Cloud LMS dealing with sensitive data and user roles.

### 7.2 Threat Modelling Outcomes

Using the STRIDE model integrated into the SDL threat modelling tool, we generated a threat list based on a Data Flow Diagram (DFD) of the system. The threats identified were as follows:
**Spoofing**: Unauthorized impersonation of valid users
**Tampering**: Modification of uploaded files
**Repudiation**: Users denying file operations
**Information Disclosure**: Unauthorized file access
**Denial of Service**: File system flooding
**Elevation of Privilege**: Gaining admin-level access
Each threat was then rated using the DREAD model. For instance, "Information Disclosure" and "Elevation of Privilege" received high-risk scores, triggering the implementation of stricter access control and encryption.

### 7.3 Formal Modelling

Formal models were developed in Event-B with the following components:
**Abstract Model**: Described system operations (e.g., login, upload) and security invariants (e.g., only authenticated users can access files).
**First Refinement**: Incorporated security properties such as authentication, confidentiality, and access logging.
**Second Refinement**: Added full functional behavior of file upload/download, including user role handling.
7.4 Verification and Model Checking
**Verification**: The models were verified using Atelier-B provers in RODIN. Out of 60 proof obligations generated, 58 were discharged automatically. Two were resolved by refining event guards.
**Model Checking**: Using ProB, the system passed checks for invariant preservation and deadlock-freeness. No critical flaws were found.

### 7.5 Validation

With AnimB, we animated the model using sample input values. The application responded to user operations as expected.

### 7.6 Performance Summary

| Phase | Result |
|---|---|
| Threats Identified | 12 total (5 High, 4 Medium, 3 Low) |
| Proof Obligations | 60 total (100% resolved) |
| Deadlock-Freeness | Verified (No deadlocks) |
| Invariant Preservation | Verified |
| Validation via AnimB | Success (Behaviour matched specs) |

The prototype demonstrates that the proposed framework is effective in identifying, modelling, and validating both security and functional requirements in a Cloud LMS, without the overhead of full-scale formal development.

## 7. Conclusion

This paper introduces a new framework for developing secure Cloud LMS applications, which is grounded in the selective application of formal methods. The paper includes a brief review of relevant literature, an overview of formal methods, and a discussion of their role in the specification, verification, and validation of software systems. Security properties are essential to the success of any software system, and this paper highlights key properties such as authentication, authorization, confidentiality, non-repudiation, denial of service, and elevation of privilege. Additionally, the basic security models, STRIDE and DREAD, are introduced. The proposed framework consists of 16 phases, ranging from requirements gathering to the creation of implementation-ready models. The research is implemented for a secure file sharing use case. Future work will involve applying this framework to a real-world case study involving a simple Cloud LMS -based application and validating the approach. Further investigation will also explore the possibility of automatically converting formal models into code.

## 5. REFERENCE

Admass, W. S. (2024). Cyber security: State of the art, challenges and future directions. *Cyber Security and Applications, 2*, 100031.

Ahn, H. a. (2021). Formal Approach to Workflow Application Fragmentations Over Cloud Deployment Models. *Computers, Materials & Continua, 67*(3).

Akbar, H. a. (2023). The security issues and challenges in cloud computing. *International Journal for Electronic Crime Investigation, 7*(1), 13–32.

Awaysheh, F. M. (2021). Security by design for big data frameworks over cloud computing. *IEEE Transactions on Engineering Management, 69*(6), 3676–3693.

Bernsmed, K. a. (2022). Adopting threat modelling in agile software development projects. *Journal of Systems and Software, 183*, 111090.

Bordis, T. a. (2020). Correctness-by-construction for feature-oriented software product lines. *Proceedings of the 19th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, 22–34.

Butt, U. A. (2023). Cloud security threats and solutions: A survey. *Wireless Personal Communications, 1*, 387–413.

Coskun, M. a. (2022). Fuzzy Logic-Based Threat Assessment Application In Air Defense Systems. *IEEE Transactions on Aerospace and Electronic Systems, 59*(3), 2245–2251.

Dankov, Y. a.-P. (2023). Towards Analysis of Threat Modeling of Software Systems According to Key Criteria. *International Conference on Intelligent Systems Design and Applications*, 98–106.

Davis, R. a. (2024). Cyber Threat Modeling for Water and Wastewater Systems: Contextualizing STRIDE and DREAD with the Current Cyber Threat Landscape. *2024 Systems and Information Engineering Design Symposium (SIEDS)*, 301–306.

Ferrari, A. a. (2022). Formal methods in railways: a systematic mapping study. *ACM Computing Surveys, 55*(4), 1–37.

Friha, O. a.-Z. (2024). Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. *IEEE Open Journal of the Communications Society*.

George, S. M.-K. (2021). A Survey on Software Test Specification Qualities for Legacy Software Systems. *Research and Evidence in Software Engineering*, 59–74.

Gleirscher, M. a. (2023). A manifesto for applicable formal methods. *Software and Systems Modeling, 22*(6), 1737–1749.

Gobov, D. (22). Practical study on software requirements specification and modelling techniques. *International Journal of Computing, 1*(78–86), 2023.

Granata, D. a. (2024). Systematic analysis of automated threat modelling techniques: Comparison of open-source tools. *Software Quality Journal, 32*, 125–161.

Joshi, M. a. (2021). Analytical review of data security in cloud computing. *2021 2nd International conference on intelligent engineering and management (ICIEM)*, 362–366.

Lallie, H. S. (2020). A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review, 35*, 100219.

Lange, F. a. (2024). Evolution of secure development lifecycles and maturity models in the context of hosted solutions. *Journal of Software: Evolution and Process*, e2711.

Luckcuck, M. (2023). Using formal methods for autonomous systems: Five recipes for formal verification. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 237*(2), 278–292.

Nag, A. a. (2024). Exploring the applications and security threats of Internet of Thing in the cloud computing paradigm: A comprehensive study on the cloud of things. *Transactions on Emerging Telecommunications Technologies, 35*(4), e4897.

Omolara, A. E. (2022). The internet of things security: A survey encompassing unexplored areas and new insights. *Computers & Security, 112*, 102494.

Riad, R. a. (2022). Learning strides in convolutional neural networks. *arXiv preprint arXiv:2202.01653*.

Sharma, R. a. (2020). *Cloud computing–security, issues, and solutions*. Springer.

Sheikh, Z. A. (2022). A Hybrid Threat Assessment Model for Security of Cyber Physical Systems. *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 582–587.

Umeugo, W. (2023). SECURE SOFTWARE DEVELOPMENT LIFECYCLE: A CASE FOR ADOPTION IN SOFTWARE SMES. *International Journal of Advanced Research in Computer Science, 14*(1).

Yeng, P. a. (2020). *Comparative analysis of threat modeling methods for cloud computing towards healthcare security practice*. SAI, The Science and Information Organization.

Zaki-Ismail, A. a. (2021). Requirements formality levels analysis and transformation of formal notations into semi-formal and informal notations. *International Conference on Software Engineering and Knowledge Engineering 2021*, 303–308.

Zografopoulos, I. a. (2021). Cyber-physical energy systems security: Threat modeling, risk assessment, resources, metrics, and case studies. *IEEE Access, 9*, 29775–29818.