Noname manuscript No.
(will be inserted by the editor)

Implicit Security Requirements Classification with Large Language Models Using the OWASP Application Security Verification Standard: A Shift-Left Approach.

Yusuf Gür · Tuğba Taşkaya Temizel · Banu Günel Kılıç

Received: date / Accepted: date

Abstract Cybersecurity threats require early integration of security, starting from the requirements analysis phase of the Software Development Life Cycle (SDLC). However, security requirements in Software Requirements Specification (SRS) documents are often implicitly embedded, making their manual identification time-consuming, error-prone, and reliant on specialized expertise. The accurate classification of security requirements (SR) is important for effective resource allocation and risk management in software development. Automated tools to extract implicit security requirements are lacking, largely due to the scarcity of large, annotated datasets in Security Requirements Engineering (SRE). This paper proposes a data-driven methodology to automate the classification of implicit security requirements in SRS documents, supporting the early and systematic integration of security into software systems. We introduce a novel multi-label corpus, the Agency Security Requirements Dataset (ASRD), derived from 2,652 real-world requirement statements from six diverse documents and annotated using a high-granularity taxonomy based on the OWASP Application Security Verification Standard (ASVS) V2-V13 and the MATTER cycle annotation framework by three cybersecurity experts. Using this dataset, we evaluate both supervised fine-tuned BERT variants (such as SecureBERT) and general-purpose large-language models (LLMs) including Gemma, GPT, DeepSeek, Meta Llama, and Gemini under zero-shot and few-shot settings. We conduct an empirical comparison between tradi-

Yusuf Gür
Graduate School of Informatics, Middle East Technical University, Ankara 06800, Türkiye
E-mail: yusuf.gur@metu.edu.tr

Tuğba Taşkaya Temizel
Graduate School of Informatics, Middle East Technical University, Ankara 06800, Türkiye
E-mail: ttemizel@metu.edu.tr

Banu Günel Kılıç
Graduate School of Informatics, Middle East Technical University, Ankara 06800, Türkiye
E-mail: bgunel@metu.edu.tr

tional fine-tuned transformer models and contemporary Large Language Models (LLMs) employing few-shot and zero-shot prompt engineering strategies. The results show that a few-shot prompting with Gemini 2.0 achieves a macro-average F1 score of 0.941, directly comparable to the fine-tuned BERT model's 0.942. This study culminates in two primary findings: first, the validation and publication of the ASRD, a high-granularity, multi-label dataset for implicit security requirements based on OWASP ASVS V2-V13; and second, the direct comparison demonstrating that few-shot Large Language Models (LLMs) achieve competitive multi-label classification performance (Macro-F1 0.941) nearly equal to resource-intensive fine-tuned transformer models (Macro-F1 0.942). This confirms that LLMs represent a highly practical and resource-saving strategy for automating the identification of embedded (implicit) security requirements for software security in industrial SRS documents

Keywords Cybersecurity requirement elicitation · OWASP based security requirement classification · OWASP ASVS-based Requirement Labeling · LLM based classification

1 Introduction

Requirements Engineering (RE) concerns the identification, documentation, and management of software requirements (Sommerville and Sawyer, 1997). The increasing interconnectivity of information systems forces organizations to provide secure services in cyberspace. Integrating security during initial requirements analysis is essential to proactively mitigate risks, reduce vulnerabilities, and reduce the high costs associated with downstream remediation (Villamizar et al., 2018). This proactive approach is central to the "Shift-Left" security paradigm, which advocates for the identification and mitigation of vulnerabilities as early as possible in the Software Development Life Cycle (SDLC).

Although Security Requirements Engineering (SRE) has matured as a discipline, its effective application in practice remains challenging. The issue is not the absence of security considerations, but the difficulty of ensuring that security requirements are properly validated and verified. An empirical investigation that combines a systematic literature review with a survey of 58 industry professionals highlights this gap (Alam et al., 2025). While practitioners reported a moderate understanding of SRE practices, Security Requirements Assurance (SRA) was found to be poorly understood and rarely applied in industrial contexts. The study concludes that validating and ensuring security requirements remains a major and unresolved challenge in practice.

This assurance gap persists due to the documented lack of specialized security training for developers in early-phase RE (Andrade et al., 2023). The task relies heavily on scarce specialized expertise and time-intensive manual analysis. The main challenge is that critical security requirements are often not explicitly stated. Instead, they are implicitly embedded within standard functional descriptions. For example, a seemingly simple functional requirement,

"The system will allow the user to print documents," carries numerous implicit security needs, including verifying user authorization, maintaining a valid session, and recording the action for accountability. These concerns correspond to common security control areas, such as access control, session management, and logging. This complexity requires substantial guidance. Manually identifying and validating this network of implicit security dependencies across thousands of requirements is error prone, inconsistent, and not scalable (Ye et al., 2025).

Given the limitations of manual SRA, automation through Artificial Intelligence (AI) based Natural Language Processing (NLP) methods, offers a promising path forward. Although AI based approaches for RE have received growing attention (Abbasi et al., 2025; Zadenoori et al., 2025), recent evidence shows a gap between experimental advances and validated industrial solutions. For example, an analysis of 74 primary studies found that most AI based tools are evaluated in controlled settings, with limited industrial adoption and weak integration into real workflows (Bolanos et al., 2024). Similarly, another study reports a clear mismatch between expectations and practical outcomes in SRE in real world use (Karhu et al., 2025).

The main problem is the lack of suitable data, not the availability of effective modeling approaches. The development and validation of data driven approaches is constrained by the scarcity and limited quality of domain specific datasets (Wang et al., 2024; Zadenoori et al., 2025). This concern extends beyond SRE, as leading NLP venues highlight dataset limitations that make reliable evaluation difficult (Muresan et al., 2022; Goldberg et al., 2022; Christodoulopoulos et al., 2025).

This challenge is especially evident in SRE. Identifying implicit security requirements relies on large collections of functional requirements annotated by security experts to reflect underlying security concerns. Current datasets do not meet this need. For instance, PROMISE exp (Lima et al., 2019) includes only a single, high-level security category; DOSSPRE (Kadebu et al., 2023) is based on student projects and lacks industry realism; and the Healthcare dataset (Riaz et al., 2014) is limited to a narrow domain and uses coarse-grained labels. As a result, existing resources offer limited support for developing and evaluating methods aimed at uncovering implicit security requirements.

The OWASP Application Security Verification Standard (ASVS) [1] offers a standardized, systematic, and verifiable framework for defining and evaluating application security controls. ASVS emphasizes the principle of "security by design" by facilitating the integration of security considerations early in the software development lifecycle. Its comprehensive set of requirements can be directly mapped to verifiable test cases and secure coding practices, ensuring traceability throughout the development process. This approach enables the early identification and remediation of vulnerabilities, which is both more effi-

[1] https://owasp.org/www-project-application-security-verification-standard/

cient and cost-effective compared to addressing security issues at later stages of development ((Khan et al., 2024).

This paper proposes a machine learning-based system to automatically analyze SRS documents and classify security requirements according to OWASP ASVS. Its main contribution is the Agency Security Requirements Dataset (ASRD), a publicly available corpus created to support research in automated SRA. The ASRD comprises:

- Real-World Data: 2,652 Software Requirements Specification (SRS) sentences extracted from six diverse, real-world industry projects,
- Expert Annotation: Meticulously annotated by three cybersecurity experts, each with over 15 years of industry experience,
- Rigorous Methodology: Developed using the iterative MATTER cycle annotation framework (Pustejovsky and Stubbs, 2012),
- High Granularity and Multi-label Structure: Mapped to 11 distinct, actionable security categories from the industry-standard OWASP Application Security Verification Standard (ASVS, $V2$–$V13$), allowing each requirement to be associated with multiple categories.

The ASRD is intended as a shared research resource and benchmark. Using this dataset, the study conducts an empirical analysis of automated classification approaches and addresses the following research questions:

1. To what extent can fine-tuned transformer models accurately perform multi-label classification of implicit security requirements into OWASP ASVS categories using the ASRD?
2. How does the performance of prompt-based approaches (zero-shot and few-shot) using modern LLMs compare to fine-tuned BERT-based models and naive baselines for this task?

The results show that few-shot prompting with a modern language model (Gemini 2.0) achieves a macro-average F1 score of 0.941, closely matching the best fine-tuned BERT-based model (0.942). This demonstrates that example-guided inference can reach performance comparable to supervised fine-tuning while requiring substantially less annotated data and model retraining. The evaluation compares fine-tuned transformer models (e.g., SecureBERT) with prompt-based methods in zero-shot and few-shot settings across multiple contemporary models, demonstrating that prompt-based approaches provide a practical and resource-efficient alternative for identifying implicit security requirements in industrial SRS documents.

The remainder of this paper is organized as follows. Section 2 reviews related work on NLP-based requirements classification, security frameworks, and existing security datasets. Section 3 describes the curation and expert annotation of the ASRD and summarizes its key characteristics. Section 4 outlines the experimental setup and classification methods, including fine-tuned BERT models and prompt-based approaches. Section 5 reports the experimental results. Section 6 discusses the findings and their implications, with a comparative analysis of fine-tuning and prompting strategies. Section 7 examines

threats to validity, and Section 8 concludes with a summary of contributions and directions for future work.

2 Background

This section reviews prior work on NLP-based techniques for requirements classification, followed by an overview of relevant security frameworks. It concludes with a discussion of existing datasets used in security requirements research.

2.1 Related Work on Requirements Classification using NLP Techniques

Software requirements (SR) classification supports effective project management by enabling prioritization and risk assessment (Batool et al., 2025). The task typically involves distinguishing functional requirements (FRs) from non-functional requirements (NFRs) and further categorizing NFRs into classes such as security, usability, and performance. Security requirements may appear as explicit system behaviors or as quality attributes addressing confidentiality, integrity, and availability—the security triad—which are often abstract and difficult to specify precisely in practice (Anwar Mohammad et al., 2019). This inherent ambiguity complicates the development of clear guidelines for separating security-related requirements from non-security ones.

Early studies applied supervised learning techniques, including Bayesian classifiers(Knauss et al., 2011), decision trees (Jindal et al., 2016), and Support Vector Machines (SVMs) (Dalpiaz et al., 2019), but these approaches depended heavily on manual feature engineering and rigid linguistic patterns. To address these limitations, later work adopted neural models such as convolutional neural networks (CNN) combined with Word2Vec embeddings (Dekhtyar and Fong, 2017). The adoption of transformer-based models such as BERT enabled improved classification accuracy through transfer learning and fine-tuning, eliminating the need for handcrafted features (Devlin et al., 2019; Subahi, 2023). More recent work explores domain-adapted transformer models, such as NoRBERT (Hey et al., 2020) and SecureBERT (Aghaei et al., 2022), to better capture domain-specific semantics and mitigate overfitting. These models have shown improved capability in identifying implicit non-functional requirements, a particularly challenging task in security-critical settings (Necula et al., 2024). However, further progress in SRE is limited by the availability of large, high-quality, and domain-specific annotated datasets.

The subjective nature of defining security requirements further complicates reliable classification (Riaz and Williams, 2012). Security Requirement Elicitation highly depends on domain-relevant expertise which includes various costly manual steps such as creating security requirement templates, determining security classification levels according to natural language artifacts, and mapping the class labels to related requirement templates by identifying entities.

Beyond requirements classification, security research has also turned to task-specific models for narrowly defined problems. For instance, deep learning approaches such as BiLSTMs have been used to detect re-entrancy vulnerabilities in smart contracts, where generic models are insufficient (Maturi et al., 2025). This work highlights the importance of explainability, as security decisions must be understandable to support assurance and auditing. Although this line of research addresses a different problem, it reflects a shift toward domain-aware and interpretable models.

Transformer-based models have enabled new capabilities in automated text understanding(Zhou et al., 2025). They possess emergent abilities like in-context learning, instruction following, and step-by-step reasoning (Zhu et al., 2025), which allow them to perform complex tasks, including requirements classification, with minimal or no labeled data (Liu et al., 2023b; Zhu et al., 2025). Recent trends show a growing interest in applying LLMs to RE tasks, including requirements classification, prioritization, and traceability (Rajbhoj et al., 2024; White et al., 2024). LLMs can perform zero-shot and few-shot learning, reducing the need for large annotated datasets and facilitating adaptation to specialized domains like SRE (Labrak et al., 2023). Techniques such as prompt engineering and integration with external knowledge bases (e.g., Retrieval-Augmented Generation) further improve their performance and contextual relevance (Masoudifard et al., 2024).

LLM-assisted security solutions typically follow a three-step process: preprocessing (extracting context like documentation or target units), prompt generation (using instructions and unfilled slots for the model to complete with artifacts like fuzz drivers or patches), and post-processing (validating outputs, e.g., checking if generated code successfully compiles) (Zhu et al., 2025).

While these models provide powerful automation, they can also introduce new risks and errors if applied without careful oversight. This underscores the need for realistic, high-quality datasets and standardized evaluation frameworks to reliably assess automated security requirement classification, ensuring both effectiveness and safety in practice.

The use of automated models in security highlights both their potential and the risks they can introduce. Ensuring accurate and reliable classification of security requirements requires evaluation against high-quality, domain-specific benchmarks (Jeong, 2024). This underscores the need for realistic datasets and standardized evaluation frameworks tailored to SRE.

2.2 Related Work on Security Frameworks

An appropriate security framework is necessary to ensure systematic and consistent identification and classification of security requirements, particularly in automated settings. This section reviews widely adopted security frameworks and evaluates their suitability for requirement analysis.

Several established frameworks, while influential, are not designed for this purpose. MITRE ATT&ACK [2] framework is a threat-based model of adversary behaviors, not a taxonomy of software requirements. Microsoft's STRIDE operates at a high level of abstraction (e.g., "Spoofing", "Tampering") which is useful for threat modeling but lacks the granularity needed for requirement classification (Khan et al., 2017). Similarly, the NIST Secure Software Development Framework (SSDF) defines high-level secure process practices rather than atomic requirement categories (Souppaya et al., 2022), and the Common Criteria (CC) [3] is a template-based framework designed for formal certification, making it impractical for fine-grained, natural-language analysis (Infrastructure and Profile, 2002).

In contrast, the Open Web Application Security Project (OWASP)[4] Application Security Verification Standard (ASVS)[5] is well suited to the objectives of this study. ASVS provides a detailed, verifiable, and developer-oriented taxonomy explicitly intended for requirement-level security analysis. Its organization into distinct, actionable security categories supports direct mapping to functional and non-functional security controls. Previous studies have successfully applied ASVS in requirements and design contexts (Łukasiewicz and Cygańska, 2019; Tan et al., 2021; Wen and Katt, 2023). Its consistency with standards such as NIST 800-63B supports its use in both academic research and industrial settings.

2.3 Related Work on Security Related Requirement Datasets

High-quality benchmark datasets are essential for developing automated classification models, ensuring research reproducibility, and assessing model generalizability. In response to this need, the research community has developed a limited number of foundational datasets for security requirements analysis, each offering distinct advantages and limitations. The three most representative datasets are briefly described below, and their key characteristics are summarized in Table 1.

2.3.1 PROMISE_Exp

Introduced by Lima et al. (Lima et al., 2019), to expand the well-established tera-PROMISE repository, this dataset resulted from a systematic curation of publicly available SRS documents. The curation process relied on manual extraction and expert consensus to ensure high-quality annotations. Although it offers significant project diversity compared to its predecessor, its primary challenge for modern research is its "High Dimension, Low Sample Size" (HDLSS) nature. The high feature-to-instance ratio and inherent class

[2] https://attack.mitre.org/

[3] https://www.commoncriteriaportal.org/cc/

[4] https://owasp.org/

[5] https://owasp.org/www-project-application-security-verification-standard/

Table 1: Comparative Analysis of Security Requirement Datasets

Characteristic	**PROMISE_exp**	**DOSSPRE**	**Healthcare Dataset**	**ASRD (Ours)**
Total Requirements	969	1,317	10,963	2,652
Data Source	Public SRS documents (expanded to 49 projects)	105 Student project documents	6 Official Canadian and U.S. SRS documents	6 Real-world SRS docs from a gov't agency
Domain	General / Cross-domain	General / Academic	Electronic Healthcare	Multi-domain: Legal, Const., Edu., etc.
Primary Task	FR vs. NFR, Multi-class NFR	SR vs. NSR, Multi-class SR	Multi-label Security Objectives	Multi-label Implicit Security Class.
Security Granularity	Single SE class	10 security classes	6 security objectives	11 OWASP ASVS classes
Language	English	English	English	Turkish
Annotation Methodology	Expert Consensus: Manual extraction and validation	Author Compilation: Categorized by authors only, limited rigorous validation	Multi-Stage Consensus: 2 researchers + mediator, high agreement	MATTER Cycle: 3 SMEs with adjudication, iterative refinement and abjudication
Key Strength	Established benchmark and project diversity	Fine-grained security-specific taxonomy	Large scale and domain authenticity	Industrial complexity + OWASP ASVS
Key Limitation	Small size and class imbalance	Potential lack of professional realism	Domain-specific; coarse-grained labels	Real-world class imbalance in specific categories

imbalance often restrict its suitability for training complex deep learning architectures.

2.3.2 DOSSPRE

The Dataset of Students' Software Projects Requirements (DOSSPRE) was compiled by Kadebu et al. from documentation within academic curricula (Kadebu et al., 2023), likely from the Harare Institute of Technology. It provides a fine-grained classification through a detailed taxonomy for both security and non-security requirements. However, as the requirements were authored by students, they may not fully reflect the realism and complexity of industry-grade requirements.

2.3.3 Electronic Health Domain Dataset

This corpus focuses on the electronic healthcare industry, a sector defined by stringent regulatory and high-assurance demands (Riaz et al., 2014). The dataset's strength lies in its rigorous multi-stage annotation methodology, which utilized independent researchers and a mediator to ensure high inter-annotator agreement. While it provides a large-scale, authentic resource for healthcare-specific tasks, its narrow focus may limit the generalizability of models to other industries, and its high-level security objectives may be too coarse for detailed technical analysis.

2.3.4 Summary of Datasets

As summarized in Table 1, these datasets vary significantly in their suitability for tasks ranging from general non-functional requirement (NFR) classification to detailed security analysis. Our proposed Agency Security Requirements Dataset (ASRD) addresses gaps in prior work by adopting the OWASP ASVS taxonomy for multi-label annotation. Furthermore, by utilizing real-world projects originally written in Turkish, ASRD introduces both industrial complexity and linguistic diversity to the field.

3 Dataset Curation, Annotation, and Evaluation

3.1 Data Collection and Preprocessing

The dataset compilation began with the collection of six Software Requirements Specification (SRS) documents from an Agency's Security Department. These documents were selected from an initial pool of 19 projects executed between 2019 and 2023, specifically chosen for their domain diversity and amenability to effective anonymization. The original documents were provided in standard document formats (e.g., PDF, DOCX) where requirements were organized in itemized lists. To transform these documents into a machine-readable dataset, we employed a multi-stage preprocessing pipeline:

1. Sentence Segmentation and Extraction: As the source SRS documents utilized standard itemization for requirements, we employed a rule-based extraction method. We utilized pattern matching to identify requirement blocks based on hierarchical numbering schemes and line breaks. Each identified requirement was extracted and treated as a single processing unit, preserving the structural integrity of the original specification.
2. Anonymization: To protect sensitive data, a semi-automated anonymization process was implemented. Custom scripts replaced entities such as project names, specific URLs, and IP addresses with generic placeholders. This was followed by a manual review by domain experts to ensure that the removal of personally identifiable information (PII) did not compromise the semantic meaning or contextual integrity of the requirements.

3. Filtering: The initial extraction yielded 3,264 items. To ensure the quality of the dataset, domain experts conducted a rigorous manual filtering pass to remove 612 items. This exclusion process went beyond simple formatting artifacts (such as headers or tables of contents). We specifically removed non-functional administrative clauses that addressed the vendor rather than the system (e.g., "The contractor must provide training manuals"), project management constraints (e.g., "The project must be delivered by Q3"), and requirements with ambiguous scope that lacked clear verification criteria.
4. Randomization: The remaining requirements were randomized and masked to mitigate potential annotation bias related to the document source or the sequence of requirements.

The resulting ASRD comprises 2,652 well-formed functional requirements across six distinct business areas: Construction (1,202), Legal (177), Education (312), Software (199), Meeting (126), and Accommodation (636). A sample of these requirements is provided in Table 2. The requirements included in this sample were selected based on three specific criteria to ensure a representative overview:

- Samples were chosen to reflect the varied terminology and phrasing styles used across the six different business domains.
- We prioritized requirements that demonstrate how standard functional descriptions conceal underlying security dependencies.
- We ensured that the sample includes representative examples for all 11 OWASP ASVS security classes used in the study to demonstrate the full scope of the classification task.

It is important to note that the ASRD is a Turkish-language corpus; all experiments detailed in this study, including BERT fine-tuning and LLM prompting, were conducted exclusively on the original text. English translations are provided only for readability.

Table 2: Sample Requirements from ASRD

No	**Original Requirements**	**Translated Requirements**
1	Sistem, yetkili kullanıcının hukuk dosyası türü kayıt etmesini sağlayacaktır.	The system shall allow authorized users to record the legal file type.
2	Sistem, kullanıcının kullanılmamış hukuk dosyası türünü güncellemesini sağlayacaktır.	The system shall allow the user to update an unused legal file type.
3	Sistem, kullanıcının kullanılmamış olan hukuk dosyası türünü silmesini sağlayacaktır.	The system shall allow the user to delete an unused legal file type.
4	Sistem, kullanıcının hukuk dosyası türlerini sorgulamasını ve listelemesini sağlayacaktır.	The system shall allow the user to query and list legal file types.

Table 2 – continued from previous page

Nu	Original Requirements	Translated Requirements
5	Sistem, kullanıcının var olan bir hukuk dosyası türüne hukuk dosyası alt türü tanımlamasını sağlayacaktır.	The system shall allow the user to define subtypes for an existing legal file type.
6	Sistem, kullanıcının hukuk dosyası alt türünü aktif/pasife çekmesini sağlayacaktır.	The system shall allow the user to activate or deactivate legal file subtypes.
7	Sistem, kullanıcının dava dosyasına bir veya birden fazla davacı tanımlamasını sağlayacaktır.	The system shall allow the user to define one or more plaintiffs for a legal file.
8	Sistem, davaların konu kodunun zorunlu olmasını sağlayacaktır.	The system shall enforce the mandatory entry of a subject code for cases.
9	Sistem, kullanıcının hukuk dosyasında görüntülendiği evrakın çıktısını almasını sağlayacaktır.	The system shall allow the user to print documents displayed in the legal file.
10	Sistem, sistem soru önerilerine otomatik numara vermesini sağlayacaktır.	The system shall automatically assign numbers to system question proposals.
11	Sistem, kullanıcının dava dosyası evrak listesinden seçtiği bir evrakı UYAP entegrasyonu ile çekmesini sağlayacaktır.	The system shall allow the user to retrieve a document selected from the case file list via UYAP integration.
12	Sistem, toplantı salonunun tahsis edilmiş olduğu tarih saat ile çakışan toplantı salonu tahsis taleplerinin oluşturulmasını engelleyecektir.	The system shall prevent the creation of meeting room allocation requests that overlap with already allocated date and time slots.
13	Sistem, kullanıcının, sayısallaştırılmış genel keşif incelemeleri dokümanını sisteme yüklemesini sağlayacaktır.	The system shall allow users to upload digitized general inspection report documents to the system.
14	Sistem, kullanıcının katılımcı listesinden katılımcı seçerek, o katılımcıya ait detay bilgilerini görmesini sağlayacaktır.	The system shall allow the user to select a participant from the list and view their detailed information.
15	Sistem, kullanıcının eğitici ödeme, iletişim ve ulaşım bilgilerini kaydetmesini sağlayacaktır.	The system shall allow the user to save payment, communication, and contact information for the trainer.
16	Sistem, intranet ve internet ağındaki kullanıcıların eğitim plan duyurusunu indirmesini sağlayacaktır.	The system shall allow users on the intranet and internet network to download training announcements.
17	Sistem, uygulama yazılımında dış ortamlar için geliştirilecek web servislerine HTTP ve/veya HTTPS protokolü üzerinden erişilebilmesi desteğini sağlayacaktır.	The system shall provide access support for web services to be developed for application software in distributed environments via HTTP and/or HTTPS protocols.
18	Sistem, uygulama yazılımının modüler ve dağıtık bir yapıda geliştirilebilmesini sağlayacaktır.	The system shall support the development of the application software in a modular and distributed architecture.
19	Sistem, uygulama yazılımının, Spring framework ile MVC (Model-View-Controller) mimari tasarım şablonuna uygun geliştirilebilmesini sağlayacaktır.	The system shall enable the development of the application software in compliance with the Spring framework and the MVC (Model-View-Controller) architectural design pattern.
20	Sistem, kullanıcının veri girişi sırasında, otomatik olarak maskeleme desteği sağlayacaktır.	The system shall provide automatic masking support during user data entry.

Table 2 – continued from previous page

Nu	Original Requirements	Translated Requirements
21	Sistem, tarayıcı üzerinde gösterilen filigranı, kullanıcı adı ve IP bilgisi gibi alanları şifrelemesi ile oluşturacaktır.	The system shall generate browser-displayed watermarks by encrypting information such as username and IP address.
22	Sistem, yöneticinin, görevli personel için kullanıcı hesabı oluşturabilmesini sağlayacaktır.	The system shall allow the administrator to create user accounts for assigned personnel.
23	Sistem, yöneticinin, kullanıcının aktif dizinde üye olabileceği güvenlik gruplarını seçebilmesini sağlayacaktır.	The system shall allow the administrator to select security groups for users in the Active Directory.
24	Sistem, güçlü şifrelerin kullanılmasını zorunlu kılacaktır.	The system shall enforce the use of strong passwords.
25	Sistem, kullanıcıların şifrelerini yönetebilmesine olanak sağlayacaktır.	The system shall allow users to manage their passwords.
26	Sistem, kullanıcının 10 dk boyunca işlem yapmaması durumunda oturumunu sonlandıracaktır.	The system shall terminate a user session if no transaction is performed within 10 minutes.
27	Sistem, kullanıcıların ilk oturum güvenlik ayarlarının uygulanmasını zorunlu kılacaktır.	The system shall enforce the configuration of security settings during the user's first login.
28	Sistem, yöneticilerin alan ve yetki prensibine göre işlem yapabilmesini sağlayacaktır.	The system shall allow administrators to act according to domain and authorization principles.
29	Sistem, kullanıcıların kayıtlarını tutacaktır.	The system shall maintain logs of user activities.
30	Sistem, kullanıcıların etkileşimleri güvenlik doğrulaması yapılacaktır.	The system shall perform authentication for all users accessing the system.

3.2 Framework and OWASP Classes Selection

Increasing web application complexity and threat sophistication necessitate standardized, requirement-level security verification. Accordingly, this study adopts the OWASP Application Security Verification Standard (ASVS) as its foundational ontology, as it provides a granular, developer-centric taxonomy of verifiable security requirements, in contrast to high-level threat models such as STRIDE or certification-oriented standards like the Common Criteria (ISO 15408), and is supported by industry adoption (ADA[6] , CREST[7]).

An important validation for using ASVS 4.0.3 in academic research is its explicit alignment with the National Institute of Standards and Technology (NIST) Special Publication 800-63b, Digital Identity Guidelines. This synchronization ensures that the requirements derived from ASVS are compatible with federal and international regulatory frameworks. A seminal study by Tan (Tan et al., 2021) serves as a primary example of ASVS adoption in the financial sector. The researchers utilized OWASP ASVS to map security controls directly to the Monetary Authority of Singapore (MAS) Technology Risk Management

[6] https://appdefensealliance.dev/

[7] https://www.crest-approved.org/membership/crest-ovs-programme/

(TRM) Guidelines. In the healthcare domain, Schmeelk and Tao (Schmeelk and Tao, 2022) conducted a case study on mobile health applications.

As of 2024–2025, commercial security platforms and verification tools continue to offer native support for ASVS 4.0.3 compliance templates and mappings (Reqview, 2016; JIT.io, 2024), with requirement managers, compliance tracking systems, and automated security testing platforms (SAST/DAST) explicitly designed around the 4.0.3 control structure. Multiple government and regulatory bodies, including the Moroccan government's Directorate General for Information Systems Security (DGSSI), have formally adopted ASVS 4.0.3 as the basis for their national application security verification frameworks as of October 2024 (DGSSI, 2024).

At the time this study was designed and conducted (2023–2025), ASVS 4.0.3 was the latest stable, officially released version of the standard. Version 4.0.3, released in October 2021, had achieved widespread adoption and had been serving as the industry-standard reference for application security verification across finance, healthcare, technology, and government sectors for multiple years. The ASVS 5.0 major revision was only released on 30 May 2025, after the empirical work for this research had been substantially completed. Therefore, this research's use of ASVS 4.0.3 represents both a temporally appropriate choice at the time of study design and a durable contribution to the literature that will remain relevant and comparable with prior ASVS-based work for years to come.

The standard includes 14 main security classes given in Table 3. For this study, only 11 classes (V2–V10, V12-V13) were used. V1, V11, and V14 were excluded as they are not inferable from functional or non-functional requirements. Each requirement can be mapped to more than one classes.

Table 3: OWASP ASVS 4.0.3 Classes

Class Id	**Class Name**	**Definition**
V1	Architecture, Design and Threat Modeling	This class focuses on the security of the application's architecture, component design, and the process of identifying and mitigating threats throughout the development lifecycle.
V2	Authentication	This class deals with verifying the identity of users, services, or applications, ensuring that only authorized entities can access specific functionalities or data.
V3	Session Management	This class covers the secure management of user sessions, from creation to termination, including protection of session identifiers and prevention of session-related attacks.
V4	Access Control	This class ensures that users can only access the data and functions for which they are explicitly authorized, enforcing the principle of least privilege.

Table 3 – continued from previous page

Class Id	Class Name	Definition
V5	Malicious Input Handling	This class focuses on requirements for validating and sanitizing all input data to prevent injection attacks, such as SQL injection, Cross-Site Scripting (XSS), and others.
V6	Stored Cryptography	This class addresses the secure storage of cryptographic keys and sensitive data at rest, ensuring they are protected from unauthorized disclosure or modification.
V7	Error Handling and Logging	This class ensures that applications handle errors gracefully without leaking sensitive information and that security-relevant events are logged for monitoring and analysis.
V8	Data Protection	This class covers the requirements for protecting sensitive data during transit and in storage, focusing on confidentiality, integrity, and classification.
V9	Communications Security	This class focuses on securing communication channels between system components, such as between the client and server, to prevent eavesdropping, tampering, or spoofing.
V10	Malicious Code	This class deals with ensuring the application code is free from malicious code, backdoors, or other unintended security vulnerabilities introduced intentionally or unintentionally.
V11	Business Logic	This class addresses security concerns within the application's business logic, preventing attackers from abusing workflows or functionalities for unintended purposes.
V12	File and Resources	This class covers the secure handling of files and other resources, including upload, download, and management, to prevent path traversal and other related attacks.
V13	API and Web Service	This class provides security requirements specifically for APIs and web services, covering aspects like authentication, authorization, and protection against common API attacks.
V14	Configuration	This class focuses on securing the configuration of all application components, including the application server, platform, and third-party libraries, to prevent security misconfigurations.

3.3 Annotation Methodology (MATTER Cycle) & Dataset Curation

The annotation process followed the MATTER development cycle, an established iterative framework for corpus annotation (Pustejovsky and Stubbs, 2012). Crucially, the annotation task was defined as a multi-label text classification problem. It consists of the phases Model, Annotate, Train, Test, Evaluate, and Revise. The annotation guidelines were developed and iteratively refined to ensure consistency and clarity.

The Model–Annotate cycle involved writing guidelines, selecting annotators, conducting trial annotations, and refining the process. Once stabilized, this cycle was expanded to the full dataset to generate a Gold Standard corpus.

The annotation workflow included the following processes:

- Annotator Selection: Following Bayerl and Paul's recommendation to use domain-aligned annotators (Bayerl and Paul, 2011), three cybersecurity subject matter experts (SME) from the agency, each with over 15 years of experience in designing security architectures, analyzing threat models, and identifying security requirements, performed the annotations. Their deep practical experience ensured accurate and context-aware interpretation which aligned with the requirement "experts must demonstrate significant skills, knowledge, and experience" emphasized by Hopkins (Hopkins and Unger, 2017). The senior manager among them also provided oversight and led the adjudication process. The senior SME acted as both an annotator and the final adjudicator for resolving disagreements. They also provided feedback to improve the quality and clarity of the annotation guideline.
- Corpus Selection and Preparation: Cohen (Cohen et al., 2005) recommends that during corpus design, balance and representativeness of the corpus is important. In line with his recommendation, of the 19 initial project documents, six were chosen for their domain diversity and potential for effective anonymization by the SMEs. These documents underwent a semi-automated anonymization process to remove all project-specific and personally identifiable information (PII). The process targeted entities such as names of individuals and organizations, project codenames, and specific technical identifiers. This was done using custom scripts with placeholder replacement, followed by manual review to ensure that anonymization did not alter the meaning, structure, or intent of the original requirements. The extracted requirements were then randomized and masked to prevent potential bias related to document origin or sequence during the annotation phase. Total corpus has 2,652 requirements (30 pilot + 2,622 independent annotations).
- Annotation Guideline Development: The Annotation Guidelines' development involved an initial iterative process, refining the guidelines three times to resolve ambiguities before the formal assessment phase. Annotation rules were based on OWASP ASVS categories V2–V13 and refined iteratively using example annotations. Following best practices suggested by Cohen (Cohen et al., 2005), the guidelines and annotation documents ensured:

(1) original text and annotations were recoverable, (2) clear documentation was maintained, (3) annotation quality was consistently high. The annotation template included fields for the software requirement text, selected OWASP labels, and annotator comments or questions. All versions of the guideline and annotation documents were preserved to track revisions based on annotator feedback. For the annotation process, a sample corpus from the requirement set has been selected by inter agreement of SMEs. The first version of the Annotation Guideline based on OWASP ASVS security classes has been developed, where each annotator has individually classified the sample corpus by annotating it with one or more labels following the instructions provided in the first Annotation Guideline. After the initial annotation, the SMEs met under the guidance of the senior SME, who is also the branch manager, to review and discuss the annotation decisions. Each of the 30 software requirements was examined individually, and in cases where there were differing opinions among the annotators, efforts were made to persuade and convince others by providing explanations and reasons for their choices. After three iterations of discussions and deliberations, a full agreement was reached for 30 software requirements.

- Pilot Annotation: An initial batch of 30 software requirements was annotated early in the study, prompting multiple discussion rounds and three updates to the guidelines. Adjudication meetings enabled the resolution of non-consensus requirements through SME-led discussions, during which the rationale for classification was articulated. Discrepancies arising from quality issues, such as guideline violations, were addressed by implementing corrections to ensure conformity with established criteria. Each SME provided justifications for their decisions, which facilitated informed discussions and enhanced understanding during adjudication. This iterative process resolved discrepancies across 98 requirements throughout the entire corpus and was important in establishing the finalized annotation framework.
- Independent Annotation: Each SME classified all remaining 2622 requirements independently, using the finalized guideline. Annotators were required to justify each decision. This practice proved important for facilitating informed discussions and understanding annotator reasoning during the subsequent adjudication phase. The process took 284 days and 267 person-hours of expert time.
- Agreement Metrics: Inter-Annotator Agreement (IAA) was calculated using Fleiss's Kappa and pairwise Cohen's Kappa to assess consistency. IAA scores were instrumental in identifying instances of disagreement among annotators, thereby highlighting areas where the guidelines might require further clarification or where ambiguous requirement phrasings contributed to annotation discrepancies. Fleiss's Kappa coefficient of 0.82 was obtained across the three annotators, indicating strong agreement that significantly exceeded random chance. Pairwise Cohen's Kappa values, ranging from

0.71 to 0.79, further confirmed substantial agreement between individual annotator pairs.
- Adjudication: Disagreements were resolved through structured expert discussions led by the senior SME. When the two annotators disagreed on a requirement, it was resolved through a structured process:
  - Both annotators presented their choice and their reasons.
  - The senior expert reviewed both positions against the guideline.
  - If the guideline clearly supported one choice, that one was used. If not, the senior expert led a discussion to decide.
  - Final decision and the reason was written down.

  The goal was to reach unanimous agreement for each disputed requirement. This process produced final Gold Standard annotations for 135 contested items, ensuring consistency and validation of the dataset.

This multi-stage annotation methodology produced a reliable dataset which comprises 2,652 annotated security-relevant requirements. The category distributions of the requirements can be seen in Figure 1. The class distribution among the classes are largely balanced (90%) except the V6(Stored Cryptography) and V12 (Files & Resources) as 6% each. This imbalance is considered a reflection of the source projects, where requirements for these two categories are inherently less frequent than pervasive controls like access control or input validation. Stored cryptography requirements are typically captured as holistic, architectural decisions rather than numerous, distinct functional requirements. Similarly, explicit security controls for file handling are less ubiquitous, appearing only in projects with specific file processing features, which explains their low frequency in our corpus. Sample annotated requirements can be seen in Table 4.

This imbalance mirrors real-world security requirement frequency - most software functions involve access and data; few involve file uploads or encryption configuration - and introduces a challenge for machine learning. Particularly, the sparsity in V6 and V12 categories may hinder generalization and calls for class imbalance handling techniques. Despite this, the dataset presents a realistic and high-quality foundation for developing and evaluating security-aware NLP and classification models.

Twenty-six requirements (0.98% of the data) labeled as “UNASSIGNED” were requirements unanimously determined by the experts to be purely functional requirements with no discernible implicit security component (e.g., “The system will enable application software to be developed with an object-oriented programming approach”,). These items were intentionally retained in the dataset as true negatives to ensure a realistic distribution and test the models’ ability to correctly reject non-security items.

Fig. 1: Distribution of Classes

Table 4: Sample Requirement Classifications

No	Original Requirements in Turkish	Translated Requirements	V2, Authentication	V3, Session Mngmt.	V4, Access Control	V5, Validation	V6, Cryptography	V7, Error Handling	V8, Data Protection	V9, Communication	V10, Malicious Code	V12, File/Resources	V13, API/Web Service
1	Sistem, otel devir işlemi gerçekleştirecektir.	The system shall carry out the hotel room handover process.	1	0	1	1	0	0	1	1	1	0	1
2	Sistem, demirbaş devir işlemi gerçekleştirecektir.	The system shall carry out the fixed asset handover process.	1	1	1	1	1	0	1	1	1	1	1
3	Sistem, işletme stokları devri gerçekleştirecektir.	The system shall carry out the business stock handover.	1	1	1	1	1	1	1	1	1	1	1
4	Sistem, kullanıcının Demirbaş Devir çizelgesi raporunu almasını sağlayacaktır.	The system shall allow the user to generate the fixed asset handover schedule report.	1	1	1	1	1	1	1	1	1	0	1
5	Sistem, kullanıcının El Senedi Eki raporu almasını sağlayacaktır.	The system shall allow the user to generate the Promissory Note Attachment report.	1	1	1	1	1	0	1	1	1	0	1
6	Sistem, uygulama yazılımına, mevcutta bulunan dil desteğine ilave olarak diğer dillerin property dosyaları olarak eklenebilmesini sağlayacaktır.	The system shall enable additional languages to be added to the application software via property files, in addition to the existing language support.	0	0	1	0	0	0	0	1	1	0	1
7	Sistem, raporun ekranda görüntülenerek ön izlenmesini, yazıcıdan alabilmesini ve dosyaya aktarılabilmesini sağlayacaktır.	The system shall allow reports to be previewed on screen, printed, and exported to files.	1	1	1	0	0	1	1	1	1	0	1
8	Sistem, uygulama yazılımı işlevleri arası dahili iletişim için geliştirilecek web servislerinin REST standartlarını desteklemesini sağlayacaktır.	The system shall ensure that web services developed for internal communication between application functions comply with REST standards.	1	0	1	1	0	0	1	0	0	0	1
9	Sistem, kullanıcının uygulama yazılımı ekranları ve sayfaları üzerinde işlem yapabileceği alanlar Türkçe F ve Q klavye ile uyumlu olacaktır.	The system shall ensure that user input areas on application screens and pages are compatible with Turkish F and Q keyboards.	0	0	0	1	0	0	0	0	0	1	0
10	Sistem, uygulamada oluşan hatalara ilişkin bildirimlerin Türkçe olmasını, hatayı açıklayacak teknik detay ya da hata kodunu içermesini sağlayacaktır.	The system shall ensure that notifications regarding errors occurring in the application are in Turkish and include technical details or error codes explaining the error.	0	0	0	0	0	1	0	0	0	0	0
11	Sistem, uygulama yazılımı ekranlarının, ana sayfaya dönüş sağlayan bir bağlantı içermesini sağlayacaktır.	The system shall ensure that the application software screens contain a link that allows returning to the homepage.	0	0	1	0	0	0	0	1	0	0	0
12	Sistem, kullanıcının, uygulamada birbiriyle ilişkili ekranlar arasında bir üst sayfaya geçiş yapabilmesini sağlayacaktır.	The system shall allow users to navigate to a parent (higher-level) page among related application screens.	1	1	1	1	0	0	0	1	0	0	0
13	Sistem, uygulama ekranları için, en az bir tane yardım ekranı sağlayacaktır.	The system shall provide at least one help screen for application screens.	0	0	0	0	0	0	0	0	0	0	0
14	Sistem, uygulamanın tasarım aşamasında belirlenecek ekranları için yardım videoları sağlayacaktır.	The system shall provide help videos for screens identified during the application design phase.	0	0	0	0	0	0	0	0	0	0	0

4 Implicit Security Requirements Classification Methodology

This section presents the experiment protocol used to assess the two research questions regarding the performance of fine-tuned BERT variant models in classification of requirements statements and LLM models with zero and few-shot prompting.

4.1 Evaluation Protocol and Data Usage

The dataset was split into training (70%, n=1,856), validation (15%, n=398), and test (15%, n=398) sets, with the validation set used for hyperparameter tuning and the test set reserved for final evaluation. Stratified sampling based on security class labels was applied to preserve label distributions across all splits. The test set was sized to include sufficient samples of minority classes (e.g., V6 and V12), while the training set remained large enough to support effective fine-tuning.

Although zero-shot LLMs do not require data partitioning and few-shot settings rely on only a small number of examples, the shared test set was retained to ensure a fair comparison between fine-tuned model variants and prompt-based LLM approaches.

We used a comprehensive suite of metrics for classification performance evaluation. We report Precision, which measures the accuracy of positive predictions, and Recall, which measures the model's ability to identify all relevant instances. The F1-score, the harmonic mean of precision and recall, is used to provide a single measure balancing this trade-off. To assess overall performance across all 11 security classes, we calculated the micro-averaged F1, which reflects aggregate accuracy, the macro-averaged F1, which assesses performance by treating all classes equally regardless of their frequency, and the weighted-averaged F1, which accounts for class support.

4.2 Classification using BERT based Models

We conducted a comparative analysis using four distinct BERT-based models to systematically evaluate the impact of different pre-training strategies. This approach allowed us to test hypotheses related to domain-specificity, language, and multilingualism, with the specific technical details of each model summarized in Table 5.

For the classification task, we adopted a standard transfer learning approach by adding a task-specific classification "head" on top of each pre-trained model. This head consists of a single fully-connected linear layer that maps

[8] `https://huggingface.co/bert-base-uncased`
[9] `https://huggingface.co/ehsanaghaei/SecureBERT`
[10] `https://huggingface.co/bert-base-multilingual-cased`
[11] `https://huggingface.co/dbmdz/bert-base-turkish-cased`

Table 5: Details and Rationale of the BERT-based Models Used in the Study

Model Name (Architecture / Parameters)	**Description, Training Data, and Source**	**Rationale in Study**
BERT Base (English) (12 layers, 768 hidden, 12 heads, 110M params.)	Standard BERT-base architecture. Pre-trained on the BookCorpus and English Wikipedia.[8](Devlin et al., 2019)	To serve as a strong performance baseline.
SecureBERT (12 layers, 768 hidden, 12 heads, 125M params.)	Based on the RoBERTa architecture. Pre-trained on a 12GB corpus of cybersecurity texts (NVD, CVE, CAPEC).[9](Aghaei et al., 2020)	To test the hypothesis that domain-specific vocabulary and context improve classification accuracy.
BERT Base Multilingual (12 layers, 768 hidden, 12 heads, 179M params.)	Standard BERT-base architecture. Pre-trained on the top 104 languages with the largest Wikipedia datasets.[10](Devlin et al., 2019)	To investigate whether training on a broader set of languages provides any generalization benefit.
BERT Base Turkish (12 layers, 768 hidden, 12 heads, 110M params.)	Standard BERT-base architecture. Pre-trained on a 35GB, 4.4B token Turkish corpus from multiple sources.[11](Schweter, 2020)	To explore the performance of a model pre-trained on a language with a different morphological structure than English.

the final hidden state of the [CLS] token to a vector corresponding to our 11 security classes. A sigmoid activation function was applied to this layer to produce independent probabilities, facilitating multi-label classification. The fine-tuning process was performed from end-to-end; all weights were unfrozen and updated to adapt the models to the specific nuances of the security requirements text.

The models were trained using the PyTorch Lightning framework to ensure reproducibility. We employed the AdamW optimizer (Loshchilov and Hutter, 2019), an extension of the Adam optimizer with improved weight decay, utilizing a linear learning rate schedule with a warmup phase covering the first 20% of training steps. This strategy stabilizes training in the initial epochs and improves convergence. Given the multi-label nature of the task, Binary Cross-Entropy (BCE) Loss was used as the primary objective function. Because the security classes are unevenly distributed, the label assignment threshold (Decision Threshold) was tuned rather than fixed at 0.5 to better balance precision and recall across categories.

We performed a systematic grid search using the Weights & Biases [12] platform to identify the optimal configuration for each BERT variant using the validation dataset. The search space included the following parameters:

- Maximum Sequence Length: `[128, 256]` tokens
- Batch Size: `[4, 8, 16]`
- Number of Epochs: `[16, 24, 32]`
- Learning Rate: `[1e-5, 2e-5, 3e-5]`
- Decision Threshold: `[0.3, 0.4, 0.5]`

To prevent overfitting, we monitored validation loss and implemented early stopping with a patience of 2–4 epochs. The checkpoint achieving the lowest validation loss for each model was selected for the final evaluation on the test set.

4.3 Classification using LLM Models

LLMs are transformer-based models trained on massive corpora for general-purpose natural language understanding and generation. This study adopted the "pre-train, prompt, and predict" paradigm, which replaces traditional "pre-train, fine-tune" approaches (Liu et al., 2023a). Rather than adjusting the model's objective or architecture, prompt engineering is used to steer the model toward task-specific outputs through carefully designed textual inputs.

Prompt engineering involves designing prompts that elicit accurate responses for specific tasks. In zero-shot prompting, an LLM is directly used without any task-specific training. Few-shot prompting introduces a small number of labeled examples in the prompt, which allows the model to learn the task pattern. This is particularly beneficial when labeled training data is limited.

Four main prompting strategies were proposed by (Liu et al., 2023a):

- Tuning-free prompting: Requires no model changes, suitable for zero-shot settings, but often demands complex prompt design.
- Fixed-LLM prompt tuning: Involves learning soft prompts while keeping the model fixed; effective for few-shot tasks but typically lacks interpretability.
- Fixed-prompt LLM tuning: Fine-tunes the model on a fixed prompt; improves performance in few-shot settings.
- Prompt + LLM tuning: Jointly tunes both model and prompts; highly expressive but prone to overfitting on small datasets.

In this study, we evaluated the first two of these strategies, as they do not require modification of the base LLM's weights. For our zero-shot experiments, we implemented tuning-free prompting, relying on carefully engineered prompts to guide the models. For our few-shot analysis, our approach aligns

[12] https://wandb.ai/site

with fixed-LLM prompt tuning, where we provided in-context examples within the prompt to guide the model's predictions. The latter two strategies involving model fine-tuning Fixed-prompt LLM tuning and Prompt + LLM tuning) were not considered due to their high computational cost. The zero shot and few shot prompts can be found in Appendix A and Appendix B.

In these experiments, we evaluated nine language models, including open-source models deployed on local infrastructure and commercial models accessed via external application programming interfaces (APIs). The technical specifications of each model and the rationale for their selection are summarized in Table 7. The requirements were processed in batches of 15 because providing the full set in a single prompt exceeded the models' context limits.

Table 6: Details and Rationale of the LLMs Used in the Study

Model Name	**Description**	**Rationale in Study**
Open-Source Models		
gemma-3-27b (Dense Transformer; 27B Params)	A Google-developed language model trained on a large, diverse corpus of web text.[13] (The Google Gemma Team, 2025)	To evaluate a powerful, next-generation open-source model from a major AI research lab, establishing a baseline for locally deployable high-parameter models.
gemma-3-4b (Dense Transformer; 4B Params)	A smaller, efficient model from the Gemma 3 family, designed for resource-constrained environments.13 (The Google Gemma Team, 2025)	To assess the performance of a highly efficient, small-scale model to determine if lower-parameter models can effectively handle complex security classification tasks.
meta.llama3-3-70b-instruct-v1:0 (Dense Transformer; 70B Params)	API version of Meta's Llama 3 70B Instruct. An open-weight model pre-trained on over 15T tokens. [14] (AI at Meta, 2024)	To benchmark against a leading open-weight model known for strong reasoning and instruction-following capabilities, representing the state-of-the-art in non-proprietary models.
DeepSeek-R1 (Mixture-of-Experts (MoE))	An API-accessible model from DeepSeek AI. Assumed to be based on the DeepSeek-V2 architecture (236B total/21B active params). [15] (DeepSeek AI, 2024)	To include a model with a different architecture (MoE) that represents an alternative, highly efficient approach to scaling, specifically testing its reasoning capabilities in security contexts.
Commercial Models (API Access)		

[13] https://blog.google/technology/developers/
[14] https://ai.meta.com/blog/meta-llama-3/
[15] https://deepseek.com/

Table 6 – continued from previous page

Model Name	Description	Rationale in Study
gemini-2.0-flash-thinking-exp-01-21 (MoE Transformer; Reasoning-Optimized)	An experimental API version of Google's Gemini Flash model. This specific version is optimized for "thinking" (Chain-of-Thought) processes while maintaining the speed of the Flash series. [16] (Google DeepMind, 2025a)	To evaluate the impact of built-in reasoning on classification accuracy using an experimental commercial model accessed prior to general availability.
gemini-2.0-pro-exp-02-05 (MoE Transformer; Large Context)	An experimental version of Google's flagship Gemini Pro model, representing the state-of-the-art for complex, multimodal reasoning tasks. 16 (Google DeepMind, 2025a)	To assess frontier-level Gemini capabilities using an experimental model accessed prior to general availability.
gemini-2.5-flash-preview-04-17 (MoE Transformer; Next-Gen Architecture)	A preview version of a next-generation Gemini Flash model, likely offering enhancements in performance, context handling, and efficiency over the 2.0 series. 16 (Google DeepMind, 2025b)	To assess whether preview architectural updates in an experimental, speed- and cost-optimized model improve detection of nuanced security contexts (e.g., Stored Cryptography) using an experimental model accessed prior to general availability.
gpt-4o (Transformer; Multimodal)	OpenAI's flagship "omni" model, natively integrating text, audio, and vision processing for faster and more seamless interaction. [17] (OpenAI, 2025)	To benchmark against the industry standard for high-performance commercial models, serving as a control variable for proprietary model performance.
gpt-4o-mini (Transformer; Cost-Effective)	A smaller, faster, and significantly cheaper version of GPT-4o, designed to make near-GPT-4-level intelligence more accessible. 17 (OpenAI, 2025)	To analyze the performance trade-off between cost and quality, determining if smaller proprietary models are sufficient for large-scale industrial requirements analysis.

5 Results

This section presents both quantative and qualitative results.

5.1 Results and Analysis for BERT Fine-Tuning

Table 7 reports the overall results for the fine-tuned BERT variants. All models achieved high Weighted-F1 scores, which show high performance on well-

[16] https://deepmind.google/technologies/gemini/

[17] https://openai.com/index/gpt-4o-mini-advancing-cost-effective-intelligence/

Table 7: Aggregated F1-Scores for LLM, BERT Models, and Baseline

Model	Prompting Strategy	Macro-F1	Micro-F1	Weighted-F1
	Large Language Model Results			
Llama 3 70b	Zero Shot	0.813	0.941	0.941
Llama 3 70b	Few Shot	0.802	0.925	0.925
DeepSeek R1	Zero Shot	0.812	0.921	0.916
DeepSeek R1	Few Shot	0.820	0.927	0.921
Gemini 2.5	Zero Shot	0.829	**0.957**	**0.953**
Gemini 2.5	Few Shot	0.817	**0.963**	**0.959**
Gemini 2.0	Zero Shot	**0.890**	0.925	0.921
Gemini 2.0	Few Shot	**0.941**	0.938	0.938
Gemini 2.0 Pro	Zero Shot	0.830	0.908	0.904
Gemini 2.0 Pro	Few Shot	0.889	0.959	0.953
Gemma 3 27b	Zero Shot	0.806	0.912	0.907
Gemma 3 27b	Few Shot	0.802	0.917	0.912
Gemma 3 4b	Zero Shot	0.605	0.738	0.742
Gemma 3 4b	Few Shot	0.723	0.871	0.867
Gpt 4o	Zero Shot	0.790	0.912	0.907
Gpt 4o	Few Shot	0.814	0.943	0.939
Gpt 4o mini	Zero Shot	0.703	0.803	0.809
Gpt 4o mini	Few Shot	0.754	0.852	0.854
	BERT Fine-Tuning Results			
bert-base-multilingual-uncased	Fine-Tuning	**0.942**	**0.985**	**0.983**
bert-base-turkish-cased	Fine-Tuning	0.933	0.983	0.981
bert-base-uncased	Fine-Tuning	0.914	0.978	0.976
SecureBERT	Fine-Tuning	0.918	0.979	0.976
	Baseline			
Naive Baseline	Majority Class	0.774	0.940	0.933

represented categories. The bert-base-multilingual-uncased model achieved the highest Weighted-F1 score at 0.983, while the lowest among the group was 0.976 for both bert-base-uncased and SecureBERT. However, these results obscure the impact of class imbalance. Macro-F1 scores, which give equal weight to all categories, were lower, ranging from 0.942 to 0.914, revealing reduced performance on less frequent classes.

This pattern is clear in the category-level results (Appendix C). Performance was high for common categories such as V8 (Data Protection), where the bert-base-multilingual-uncased model achieved an F1-score of 0.992. In contrast, results declined significantly for sparsely represented categories. For V6 (38 samples), F1-scores fell to 0.714 for bert-base-uncased and 0.734 for SecureBERT. The weakest performance was observed for V10 (27 samples), with F1-scores around 0.60, indicating limited generalization due to limited data.

An analysis of precision and recall for these difficult classes reveals specific failure patterns. For class V10, the bert-base-uncased model has a low Recall of 0.481. This indicates that while its predictions were often correct (Precision: 0.813), it failed to identify more than half of the actual V10 requirements, resulting in a high number of false negatives. For class V6, the same model

showed both low Recall (0.658) and low Precision (0.781), indicating it was both missing true cases and incorrectly labeling other requirements as V6.

Models struggle significantly with underrepresented categories, particularly V12 (File and Resources) and V6 (Stored Cryptography). This is evident in the high number of false negatives, where the models fail to identify true instances of these classes. For example, the baseline BERT model incorrectly classified over half of the V12 requirements as negative (14 false negatives vs. 13 true positives). Conversely, the matrices show that models often misclassify requirements by incorrectly assigning them to other categories. The BERT Base model, for example, incorrectly labeled 92 different requirements as V10 (Malicious Code) and 78 requirements as V5 (Malicious Input Handling).

5.2 Experimental Setup for LLM Prompt Engineering

We adapted prompt engineering strategies from Ronanki (Ronanki et al., 2024), originally used for multi-class requirement classification. Two configurations were evaluated: zero-shot prompting (Appendix A), which relied solely on task instructions and the input requirement, and few-shot prompting (Appendix B), which additionally incorporated labeled examples. The few-shot setup used 30 training instances that were chosen to (i) capture multi-label cases where a single requirement spans multiple security categories, (ii) cover all 11 OWASP ASVS categories, and (iii) reflect requirements from different business domains to preserve linguistic variation.

Persona-based prompting was applied in both configurations by instructing the model to act as an experienced cybersecurity specialist. This alignment helped reduce output variance by grounding responses in professional domain expertise rather than general conversational patterns. The final prompt was obtained after five iterations. These iterations focused on three primary areas: (1) instruction clarity to prevent the model from defaulting to binary security labels; (2) contextual grounding, by iteratively refining the OWASP ASVS category definitions to ensure the model adhered to provided standards; (3) constraint formatting, to stabilize the PSV output structure for machine-readability and ensure the inclusion of mandatory explanation fields. All experiments were conducted with fixed parameters (Temperature = 0, Top-p = 0.95) and repeated three times. The reported results are averaged to reduce the impact of any minor, non-deterministic fluctuations in the models' token selection during individual inferences.

Prompt-based methods can be sensitive to phrasing, structure, and example selection. To manage prompt sensitivity and contextual bias, we adopted a unified Persona–Context–Constraint strategy. A single prompt structure was used across all models, without model-specific tuning or iterative optimization aimed at maximizing scores. The few-shot examples were chosen for representativeness with respect to the OWASP ASVS taxonomy rather than performance gains.

Furthermore, to mitigate contextual bias, where a model might rely on its general training data rather than the specific task definitions, we utilized Contextual Anchoring. We injected the explicit definitions of the 11 OWASP ASVS classes and a set of heuristic "Extra Information" rules (e.g., mapping CRUD operations to specific security categories taken from guidelines) directly into the prompt's context window. Including this information in the prompt directed the models to follow the specified standard, resulting in more consistent classifications.

While alternative prompting strategies may yield different quantitative outcomes, this design prioritizes experimental control and cross-model comparability. Hence, the reported results reflect conservative estimates of prompt-based performance rather than optimized upper bounds.

5.3 Results and Analysis for LLM Prompt Engineering

Table 7 shows the overall performance of the LLM-based experiments. The results compare zero-shot and few-shot prompting strategies, with detailed per-category scores reported in Appendix D.

The results show that few-shot prompting often improve performance compared to zero-shot prompting, though the effect is not consistent across models or categories. For instance, while Gemini 2.0 Pro's overall Macro-F1 score rose from 0.830 to 0.889 with a few-shot approach, the score for Meta Llama 3 70b slightly decreased from 0.795 to 0.793. On category wise GPT-4o's F1-score for class V5 rose from 0.924 to 0.948 with few-shot prompting. However, this improvement was not observed across all categories. For the difficult V12 class, Meta Llama 3 70b's score actually dropped from 0.271 to 0.214 when examples were added.

Contrary to expectations, larger models did not consistently gain more from few-shot prompting. In fact, the smaller Gemma 4b model saw its F1-score on class V5 jump from 0.709 to 0.814. By contrast, the larger Gemma 27b model's performance on the same task decreased from 0.946 to 0.903. This shows that for this task, a larger model did not guarantee a greater benefit from few-shot examples.

All models showed weak performance on the underrepresented categories V6 and V12. Few-shot prompting offered limited benefit for these classes. For V6, DeepSeek R1 improved only marginally, from an F1-score of 0.117 to 0.148. For V12, Gemini 2.5 Pro achieved the highest zero-shot score (0.605), but dropped sharply to 0.335 with few-shot prompting, indicating that a small number of examples is insufficient to compensate for the severe class imbalance.

5.4 Comparative Analysis of LLMs and Fine-Tuned BERT Models

As a point of reference, we included a Naive Baseline that always predicts the most frequent class. As shown in Table 7, the model achieves a Weighted-F1

of 0.933 and Micro-F1 of 0.940, which show that the dataset's class imbalance allows a trivial classifier to achieve high accuracy by ignoring minority classes.

The baseline model's Macro-F1 score of 0.774 is significantly lower than the fine-tuned models (≈ 0.94) and its F1 score is 0.00 for the minority classes V6 (Stored Cryptography) and V12 (Files and Resources) as anticipated. Models like SecureBERT (based on RoBERTa) achieve an F1 of 0.734 on V6, demonstrating that the deep learning models are successfully learning distinct features for rare classes rather than merely exploiting the label distribution.

Overall, the fine-tuned BERT models outperformed the LLMs in both zero-shot and few-shot configurations. The best fine-tuned model, bert-base-multilingual-uncased, achieved a Macro-F1 score of 0.942, which is substantially higher than any LLM model's score.

The top-performing LLM was Gemini 2.0, which in its few-shot configuration reached a Macro-F1 score of 0.941. This score is on par with the best fine-tuned BERT model, indicating that few-shot prompting with a powerful LLM can match the performance of a specialized, fine-tuned model. However, most other LLMs did not reach this level. For example, GPT-4o's few-shot Macro-F1 score was only 0.814. In zero-shot tests, where no examples were provided, the best LLM (Gemini 2.0) scored a Macro-F1 of 0.890. This score remains below the weakest fine-tuned BERT model, with fine-tuned models performing best, followed by few-shot LLMs and then zero-shot LLMs.

5.5 Statistical Evaluation of Classifiers Performance

We also examined whether the observed models' performance differences were statistically significant. Following the comparison framework of (Demšar, 2006), we compared the distributions of classifier performance scores. The 11 OWASP ASVS security classes were treated as independent domains (stratified subsamples from the main data set). The performance of the 20 evaluated classifiers was assessed using the Friedman test, followed by pairwise Wilcoxon signed-rank tests.

The null hypothesis states that all classifiers perform equivalently and their rank distributions are identical. In our analysis, the Friedman test yielded a statistic of 188.103 with a p-value of 2.38×10^{-28}. This result rejects the null hypothesis ($p < 0.05$), confirming that the observed performance differences across the models are non-random and statistically significant.

Following the rejection of the null hypothesis, we proceeded to identify the specific pairs of models that differ significantly. For these pairwise comparisons, we employed the Wilcoxon Signed-Rank Test. Demšar recommends the Wilcoxon test as a "safe and robust" non-parametric alternative to the paired t-test for comparing two classifiers. Unlike the t-test, which assumes commensurability of differences and normality, the Wilcoxon test compares the ranks of the differences. This ensures that outliers (such as the extreme performance drops observed in minority classes like V6 and V12) do not disproportionately skew the results.

The analysis (Table 8) highlights the significant pairwise contrasts and reveals important performance distinctions:

- Baseline vs. Advanced Models: The Naive Baseline was consistently outperformed by the advanced models. The difference between the Baseline and models such as SecureBERT ($p < 0.001$) and Gemini 2.0 Few-Shot ($p < 0.001$) was statistically significant, confirming that the high Weighted-F1 scores of these models reflect genuine learning beyond simple majority-class prediction.
- Fine-Tuning vs. Zero-Shot LLMs: Fine-tuned models like SecureBERT demonstrated statistically significant performance distinctions ($p < 0.001$) when compared to general-purpose LLMs operating in a zero-shot setting, such as GPT-4o ($p < 0.001$) and Llama 3 70B ($p < 0.001$). This indicates that without in-context examples, domain-specific fine-tuning retains a measurable statistical advantage.
- Fine-Tuning vs. Few-Shot State-of-the-Art: The pairwise comparison between the best fine-tuned model (SecureBERT) and the best LLM approach (Gemini 2.0 Few-Shot) yielded a p-value of 0.102. This value is not statistically significant at the $\alpha = 0.05$ level, supporting our finding that few-shot prompting with state-of-the-art LLMs can effectively match the performance of specialized, fine-tuned BERT models in this domain.

Table 8: Significant Pairwise Differences (Wilcoxon Signed-Rank Test)

Model A	**Model B**	**Statistic**	**p-value**
Naive Baseline	SecureBERT	0.0	0.00098
Naive Baseline	BERT Turkish	0.0	0.00098
Naive Baseline	Gemini 2.0 (Few-Shot)	0.0	0.00098
Naive Baseline	Gemini 2.0 Pro (Few-Shot)	0.0	0.00098
SecureBERT	GPT-4o Mini (Few-Shot)	0.0	0.00098
SecureBERT	GPT-4o Mini (Zero-Shot)	0.0	0.00098
SecureBERT	GPT-4o (Zero-Shot)	0.0	0.00098
SecureBERT	GPT-4o (Few-Shot)	0.0	0.00098
SecureBERT	Gemini 2.0 (Zero-Shot)	25.0	0.501
SecureBERT	Gemini 2.0 (Few-Shot)	14.5	0.102

To visualize the relative performance and statistical significance groups of all evaluated models, we constructed a Critical Difference (CD) diagram (Figure 2) using the Nemenyi post-hoc test.

In the CD diagram, models are arranged along the horizontal axis according to their average rank. The best performing models, such as Gemini 2.0 Pro (Few-Shot) and BERT-Multilingual, appear on the left (lowest ranks). A horizontal bar connects groups of models that are not statistically significantly different from one another. The wide span of the CD bar (10.46) reflects the high number of models compared (23) relative to the number of datasets (11). Despite this conservative threshold, the diagram clearly separates the top-tier models (Fine-tuned BERTs and Few-Shot Gemini/GPT-4o) from the lower-performing group (Naive Baseline, Zero-Shot Gemma/Llama).

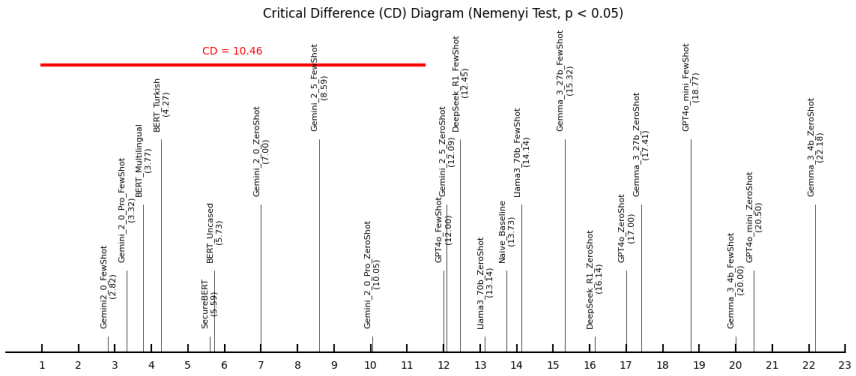

Fig. 2: Critical Difference (CD) diagram visualizing the statistical comparison of classifiers. The horizontal axis represents the average rank of each model across all 11 OWASP ASVS classes (lower ranks indicate better performance). The Critical Difference (CD = 10.46) represents the minimum difference in average rank required for two models to be considered statistically significantly different at $p < 0.05$.

5.6 Comparative Visualization Analysis

To complement the statistical rankings, we visualized the performance landscape across all 20 model configurations.

Heatmap Analysis: Figure 3 shows F1-scores by class. Requirements related to V2 (Authentication), V3 (Session Management), and V4 (Access Control) are handled well by nearly all models, with scores typically above 0.95. These categories are common and appear consistently across training sources. In contrast, V6 (Stored Cryptography) and V12 (Files and Resources) show low scores across the board, including for GPT-4o and Gemini 2.0 Pro. This pattern points to a systematic difficulty with sparse categories rather than a weakness of a particular model.

Gap Analysis: Figure 4 contrasts average performance on frequent and infrequent classes. Fine-tuned models such as SecureBERT and BERT-Multilingual perform better on rare classes, with F1-scores around 0.70–0.80. Zero-shot LLMs perform poorly in these cases, often below 0.30. Few-shot prompting improves results for some models, such as Gemini 2.0, but the gap remains substantial.

6 Threats to Validity

Our dataset consists of 2,652 requirements derived from six Software Requirements Specification (SRS) documents. While the number of source documents

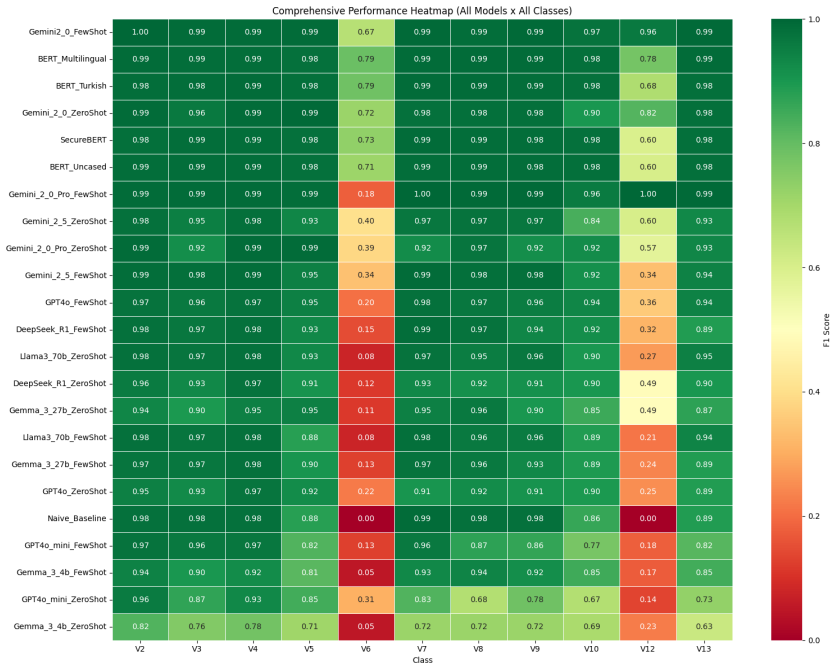

Fig. 3: **Comprehensive Performance Heatmap.** F1-scores for all 20 evaluated models across 11 OWASP ASVS classes. The visualization highlights the "zone of failure" (red) for classes V6 and V12 across models, contrasting with the high performance (green) on ubiquitous classes like V2 and V4.

is limited, potential bias was mitigated by strategically selecting projects from six distinct business domains: Construction, Legal, Education, Software, Meeting Management, and Accommodation. This semantic diversity ensures that the models are evaluated on a broad vocabulary and varied functional contexts rather than a single industry vertical. A specific consideration regarding the external validity of this study is that the ASRD is a Turkish-language corpus which allows for the evaluation of automated tools in linguistically diverse industrial contexts.

Furthermore, because security requirements (e.g., Authentication, Logging) represent cross-cutting concerns that apply universally across software systems, we believe the fundamental patterns captured in the ASRD are transferable to other industries. Our current evaluation used a stratified split across the entire dataset, meaning requirements from all six source projects are present in the training, validation, and test sets. However, we acknowledge that this approach does not fully measure the models' ability to generalize to entirely new projects with different authors or linguistic styles, as the test set is not completely "unseen" in terms of project context.

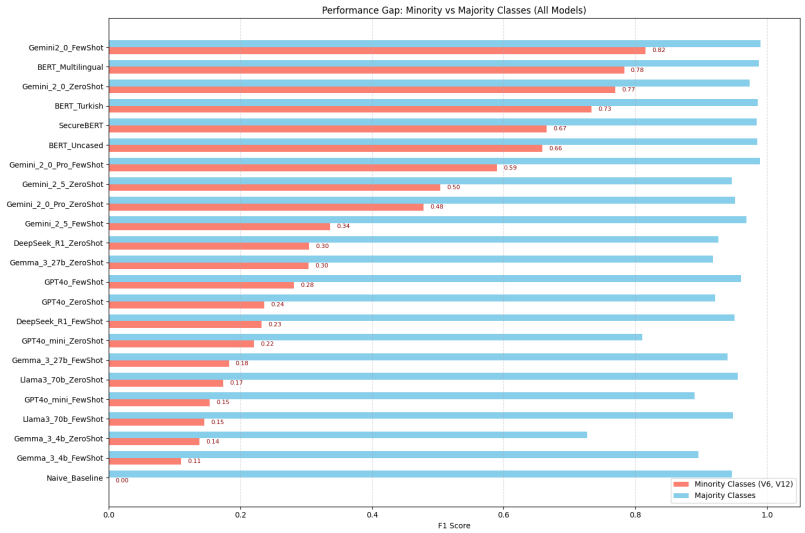

Fig. 4: **Minority vs. Majority Class Performance Gap.** A comparative analysis revealing that while advanced LLMs and fine-tuned models achieve near-parity on Majority Classes (Blue), a significant performance gap remains on Minority Classes (Red), underscoring the "Long Tail" challenge in SRE.

Annotation was performed by three experienced cybersecurity experts. While their domain knowledge ensured quality, the small annotator pool may introduce subjective bias and limit the range of interpretations. This was mitigated through the iterative MATTER cycle which includes training, iterative guideline refinement, and consensus-based adjudication.

A key threat to internal validity is subjectivity as classifications rely heavily on the latent domain knowledge and industry experience, rather than explicit indicators within the requirement text itself. This expert-driven approach provides a "gold standard" for realistic industrial needs but it may introduce inconsistencies if replicated by annotators with different backgrounds or if the broader system context is not explicitly documented.

The test set was manually selected to balance computational efficiency and generalizability assessment. While cross-validation could improve test calibration, it was avoided due to its high computational cost.

To minimize researcher bias and ensure the results reflect real-world industry needs, domain experts used for the selection of both the test set and the LLM suite. Rather than selecting data or models arbitrarily, these experts curated a test set containing the most complex, "implicit" security scenarios to challenge the models beyond simple pattern matching. For the model suite, they prioritized architectures with advanced reasoning capabilities, hypothesizing that these specific technical features are essential for the multi-step deduction required in security requirements engineering. Although the experts

lacked prior experience in Natural Language Processing (NLP) annotation, this was addressed through focused training sessions to align their domain expertise with the technical requirements of the MATTER cycle.

Variations in model versions and dependencies (e.g., library updates) pose challenges to reproducibility.

A significant class imbalance, with categories like V6 and V12 severely underrepresented, restricts model learning and generalization. To address this, extensive training, refined annotation protocols, iterative consensus-building, and detailed error analysis were employed.

A significant challenge in evaluating Large Language Models (LLMs) is the rapid pace of model iteration. In this study, we deliberately selected several "experimental" and "preview" models (specifically gemini-2.0-flash-thinking-exp-01-21, gemini-2.0-pro-exp-02-05, and gemini-2.5-flash-preview-04-17) alongside stable releases. The rationale for including these unversioned, experimental models was to benchmark the absolute state-of-the-art capabilities in "reasoning" (Chain-of-Thought) and architectural efficiency that are not yet present in General Availability (GA) versions at the time of the study. We hypothesized that the improved reasoning capabilities of models like Flash-Thinking would be particularly advantageous for identifying implicit security requirements, which often require multi-step deduction rather than simple pattern matching.

However, we acknowledge that the use of experimental endpoints poses a threat to internal validity regarding reproducibility. These models are subject to deprecation or unannounced updates by the provider, meaning other researchers may not be able to query the exact same model weights in the future. To mitigate this, we have documented the exact model identifier strings used during our data collection window. While this does not guarantee future access to these specific snapshots, it ensures transparency regarding the specific technological generation evaluated. We argue that this trade-off is necessary to provide a forward-looking analysis of how emerging LLM capabilities (such as intrinsic reasoning) impact the field of Security Requirements Engineering.

7 Discussion

The combined experimental results obtained from fine-tuning BERT models and employing various LLM prompting strategies, further refined by the comparison between zero-shot and few-shot methods, yield several key insights.

An important dimension of our comparison is the trade-off between the effort required to curate the dataset and the resulting classification performance. As detailed in Section 3, the construction of the ASRD involved 267 person-hours of expert labor to annotate 2,652 requirements. This extensive effort was a prerequisite for the fine-tuned BERT models, which rely on large-scale supervised data.

As summarized in Table 9, the BERT Fine-Tuning approach achieved the highest performance (Macro-F1 0.942) but at the highest cost. In contrast, the

Table 9: Comparison of Effort vs. Performance across Approaches

Approach	Data Requirement	Est. Setup Effort	Best Macro-F1
BERT Fine-Tuning	High (∼2,600 ex)	High (267 hours)	**0.942**
LLM Few-Shot	Low (∼30 ex)	Low (<10 hours)	**0.941**
LLM Zero-Shot	None	Minimal (<2 hours)	0.890

LLM Few-Shot approach (using Gemini 2.0) achieved a similar performance (Macro-F1 0.941) while requiring only the selection of a small number of in-context examples (approx. 30 examples per prompt).

This comparison points to diminishing returns from large-scale annotation when capable LLMs are available. Although extensive annotation was required here to establish a reliable ground truth and validate the test set, the results indicate that, in industrial settings, a few-shot LLM approach can offer a more favorable balance between effort and accuracy. Comparable performance can be reached with substantially less data preparation, reducing the need for prolonged expert annotation.

The main contribution of this work is the introduction and validation of the ASRD. The consistent performance of both fine-tuned BERT models and LLM-based approaches on ASRD supports its reliability and practical value. The dataset addresses a persistent limitation in SRE research which is the lack of large, well-curated, professionally sourced benchmarks for security requirement classification. Compared with existing datasets, ASRD offers several advantages. Unlike DOSSPRE (Kadebu et al., 2023), which is derived from student projects, ASRD includes 2,652 requirements from six commercial software systems, reflecting real industrial language and complexity. Its OWASP ASVS–based taxonomy spans 11 security classes, providing finer detail than PROMISE exp (Lima et al., 2019), where security is treated as a single category. Although the healthcare dataset by Riaz et al. (Riaz et al., 2014) is larger, it focuses on six broad objectives within a single domain, limiting its scope. By contrast, ASRD supports cross-domain analysis and aligns with a framework widely used in practice.

The dataset also benefits from a rigorous annotation process. Three cybersecurity specialists, each with over 15 years of experience, labeled the data using an iterative MATTER-based procedure. This resulted in strong agreement (Fleiss' Kappa = 0.82) and helps mitigate the subjectivity that often affects security-related requirement datasets.

Beyond the dataset construction, this study highlights the complexity of implicit security requirement elicitation. A significant portion of security vulnerabilities stems not from missing explicit security constraints (e.g., "The system shall encrypt passwords"), but from functional requirements that implicitly demand robust security controls. Our analysis of the ASRD reveals that functional descriptions of business logic often conceal a dense network of security dependencies. A representative example from our results is Requirement No. 2 ("The system will carry out the asset handover process"). While

explicitly stated as a functional transaction, our domain experts classified this requirement across nearly all OWASP ASVS categories. In an industrial context, an "asset handover" implies a legally binding transfer of custody, necessitating Authentication (V2) and Access Control (V4) to verify identity and authority, Data Protection (V8) for non-repudiation, and Secure Logging (V7) for auditability. A purely keyword-based or explicit extraction approach would likely miss these dependencies, leaving the module vulnerable. Our findings demonstrate that LLMs, when prompted with few-shot examples, can effectively mimic this expert reasoning, bridging the gap between functional specifications and security compliance.

While the ASRD dataset accurately reflects the natural distribution of security requirements in industrial projects, the performance drop on minority classes reveals distinct limitations across the two model architectures evaluated. For the LLMs, the limitation lies in the rigidity of Static Few-Shot Prompting. Our results indicate that providing a fixed set of 1–3 examples (Few-Shot) yielded inconsistent gains and, in cases like Llama-3 on V12, actually degraded performance compared to Zero-Shot. This suggests that "implicit" requirements for minority classes are highly context-dependent. A fixed example of V12 related to "file upload" does not help the LLM identify a V12 requirement related to "directory traversal." Consequently, the models fail to generalize the concept of the minority class from static prompts, leading to the low F1-scores (0.077–0.172) observed. This confirms that for rare, high-variance security classes, static prompting is insufficient.

Our study found that while fine-tuned BERT models provide strong results (up to 0.942 Macro-F1), few-shot prompting with a top-tier LLM like Gemini 2.0 can achieve nearly identical performance (0.941 Macro-F1). This supports the growing body of literature suggesting that in-context learning with large models can close the performance gap with task-specific fine-tuning, often with significantly less labeled data. This shift from a traditional "pre-train, fine-tune" to a "pre-train, prompt, and predict" paradigm, as surveyed by Liu et al. (Liu et al., 2023a), offers a more agile and resource-efficient path for deploying NLP solutions in specialized industrial contexts. Our results provide a concrete example of this trend within the SRE domain and align with findings given by Karlsson et.al (Karlsson et al., 2025).

These findings have immediate implications for the software industry, particularly in implementing "Security by Design" and "Shift-Left" strategies. In diverse development environments—ranging from Agile teams to regulated sectors like healthcare or defense—developers often lack deep security expertise. They focus on writing functional requirements (SRS) to meet business needs, often assuming security will be "added on" later. By automating the classification of implicit security requirements using the method proposed in this study, organizations can instantly map functional specs to actionable OWASP ASVS controls before a single line of code is written. Rather than replacing human oversight, this approach functions as a human-in-the-loop productivity multiplier. It serves as a sophisticated pre-screening tool that drastically speeds up the initial analysis phase, allowing scarce cybersecurity experts to

focus their efforts on high-risk adjudication rather than manual discovery. Our results suggest that commercially available LLMs can serve as always-available assistants, democratizing high-level security analysis for development teams. Furthermore, future studies could investigate the impact of this tool on practitioners with varying levels of experience. Specifically, research is needed to determine if such automation allows junior developers to identify security concerns more accurately or if it primarily eases the cognitive load for senior experts by filtering out routine classifications.

This study highlights class imbalance as a persistent challenge in SRE. The issue is not specific to ASRD but reflects the nature of the domain itself. In typical specifications, references to architectural security concerns such as cryptographic storage or low-level file handling, occur far less often than user-facing controls like authentication (V2) or access control (V4). This creates a "long-tail" distribution where the most important security failures often reside in the least represented classes. Standard supervised learning and even few-shot prompting struggle to generalize from such limited examples. This is a well-documented issue in requirements datasets, including benchmarks like PROMISE_exp (Lima et al., 2019), which also suffers from imbalanced classes.

The improvement from zero-shot to few-shot prompting shows the positive impact of adding in-context examples. For example, Gemma 4b's F1 score on class V5 rose from 0.709 to 0.814 with only a small number of examples. This gain shows how limited, well-chosen examples can steer model behavior toward the intended classification task. Similar observations have been reported in prior work across domains, where prompt design and example selection were shown to play a decisive role in task performance (Mann et al., 2020; Liu et al., 2023a).

8 Conclusion and Future Work

This work examined the feasibility of automated security requirements classification under realistic conditions, where requirements are implicit, unevenly distributed across categories, and drawn from industrial settings rather than curated benchmarks. By introducing ASRD and evaluating supervised and prompt-based methods on a shared benchmark, this work enables direct comparison across approaches. The results show that few-shot prompting is suitable when labeled data is limited, while fine-tuned models are better at handling rare security requirements.

As a future work, we plan to explore hybrid SRE approaches that combine LLMs with retrieval and agent-based components. In particular, improving performance on underrepresented classes remains an open problem and may benefit from targeted data augmentation, retrieval-augmented methods (RAG) such as those proposed by Liu et al. (Liu et al., 2025), or agent-based strategies. For instance, instead of relying on learned patterns from non-existent training examples, a RAG-enabled system can dynamically retrieve the specific verification requirements and definitions relevant to the input text at inference

time. This approach effectively substitutes the need for dense in-context examples with explicit external standards, ensuring accurate classification for rare security events where traditional fine-tuning or few-shot prompting fails.

Consequently, this study suggests that future NLP research in SRE cannot rely solely on organic dataset expansion. To overcome this inherent sparsity, we propose as future work three targeted algorithmic advancements:

1. Dynamic Few-Shot Selection with Retrieval-Augmented Generation (RAG): Systems might dynamically retrieve the explicit definitions and verification criteria from standards like OWASP ASVS to ground their classification of rare requirements. Future work should implement RAG for Dynamic Prompting in agentic approach. Instead of fixed examples, a retriever should select the k most semantically similar valid requirements from the training set to serve as in-context examples for the specific query. This ensures the LLM is grounded with relevant architectural patterns (e.g., "encryption" vs. "hashing") rather than generic class examples.
2. Human-AI Collaborative Multi-Agent Framework: A promising direction for future work is a Human-in-the-Loop multi-agent framework that combines automated classification with targeted expert oversight for ambiguous or high-risk cases. In this setup, a retrieval-augmented agent would dynamically select the most relevant annotated requirements using semantic similarity, replacing static few-shot examples. A second agent would assess classification confidence and consistency, flagging unclear cases and requesting focused clarifications from a human expert when needed. A supervisor agent would then consolidate these inputs and ensure that the final labels remain consistent with the OWASP ASVS taxonomy and the broader project context. In parallel, an optimization agent would learn from human interventions to refine retrieval strategies and prompting over time, reducing manual effort while improving accuracy on challenging security requirements.
3. Synthetic Data Injection (for Supervised Models): To address the sparsity hindering BERT-based fine-tuning, we recommend a Teacher-Student Data Generation loop. High-reasoning LLMs (e.g., GPT-4o, Gemini 2.5 Pro) should be used to generate diverse synthetic requirements that implicitly trigger V6/V12 classifications. These synthetic samples can populate the training set, allowing smaller supervised models (BERT) to learn robust decision boundaries without requiring expensive manual annotation.

Additionally, future efforts should include a more stringent evaluation of model generalizability using a leave-one-project-out cross-validation approach. This method involves training a model on requirements from five of the projects and testing it on the single, held-out project. Repeating this process for all six projects would provide a much clearer and more realistic assessment of how well these models perform on entirely new, unseen SRS documents, which is a significant measure for real-world applicability.

Furthermore, to bridge the gap between our current findings and the global research community, we plan to undertake a professional translation of the

ASRD into English. This initiative will involve a rigorous validation process by bilingual cybersecurity experts to ensure that the nuanced "implicit" security dependencies are accurately preserved across languages.

9 Acknowledgement

The authors sincerely thank the subject matter experts for their valuable time and insight during the annotation process.

10 Declarations

Funding: This work has been funded by the Scientific and Technological Research Council of Türkiye (TÜBİTAK), Scientist Support Programs Presidency (BİDEB), within the scope of the 2211-National Graduate Scholarship Program.

Conflict of Interest/Competing Interests: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical Approval: This study did not involve human participants, clinical trials, or personal data, and therefore did not require formal approval from an institutional ethics committee. The subject matter experts contributed solely in their professional capacity to the annotation process.

Informed Consent: Informed consent was not applicable as no human participants were involved. The subject matter experts participated in their professional capacity and provided voluntary contributions.

Author Contributions: Yusuf Gür conceptualized and designed the study, including data collection and analysis. Tuğba Taşkaya Temizel contributed to formulating research questions and supervising methodological and computational aspects. Banu Günel Kılıç provided oversight on the broader scientific narrative and its positioning in the field. All authors contributed to manual verification, manuscript drafting, and approval of the final version.

Data Availability Statement: The dataset used in this study is publicly available on Figshare[18] and can be accessed freely via the provided link.

Clinical Trial Number: Not applicable.

References

Abbasi MA, Ihantola P, Mikkonen T, Mäkitalo N (2025) Towards human-ai synergy in requirements engineering: A framework and preliminary study. In: 2025 Sixth International Conference on Intelligent Data Science Technologies and Applications (IDSTA), IEEE, pp 81–88

[18] https://figshare.com/s/671a40aaa1f23c33ddcb

Aghaei E, Niu X, Shadid W, Al-Shaer E (2022) Securebert: A domain-specific language model for cybersecurity. In: International Conference on Security and Privacy in Communication Systems, Springer, pp 39–56

Aghaei S, Al-Shaer E, Duan Z (2020) SecureBERT: A domain-specific language model for cybersecurity. 2105.04293

AI at Meta (2024) The llama 3 herd of models. Tech. rep., Meta, URL https://ai.meta.com/research/publications/the-llama-3-herd-of-models/

Alam JA, Ayman M, Rehman GA, Sarlan AB, et al. (2025) Security requirements engineering: A review and analysis. Computers 14(10):429

Andrade R, Torres J, Ortiz-Garcés I, Miño J, Almeida L (2023) An exploratory study gathering security requirements for the software development process. Electronics 12(17):3594

Anwar Mohammad MN, Nazir M, Mustafa K (2019) A systematic review and analytical evaluation of security requirements engineering approaches. Arabian Journal for Science and Engineering 44(11):8963–8987

Batool R, Naseer A, Maqbool A, Kayani M (2025) Automated categorization of software security requirements: an nlp and ml based approach. Requirements Engineering pp 1–13

Bayerl PS, Paul KI (2011) What determines inter-coder agreement in manual annotations? a meta-analytic investigation. Computational Linguistics 37(4):699–725

Bolanos F, Salatino A, Osborne F, Motta E (2024) Artificial intelligence for literature reviews: Opportunities and challenges. Artificial Intelligence Review 57(10):259

Christodoulopoulos C, Chakraborty T, Rose C, Peng V (2025) Proceedings of the 2025 conference on empirical methods in natural language processing. In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing

Cohen KB, Fox L, Ogren P, Hunter L (2005) Corpus design for biomedical natural language processing. In: Proceedings of the ACL-ISMB workshop on linking biological literature, ontologies and databases: mining biological semantics, pp 38–45

Dalpiaz F, Dell'Anna D, Aydemir FB, Çevikol S (2019) Requirements classification with interpretable machine learning and dependency parsing. In: 2019 IEEE 27th International Requirements Engineering Conference (RE), IEEE, pp 142–152

DeepSeek AI (2024) DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:240504434 URL https://arxiv.org/abs/2405.04434

Dekhtyar A, Fong V (2017) Re data challenge: Requirements identification with word2vec and tensorflow. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), IEEE, pp 484–489

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research 7(Jan):1–30

Devlin J, Chang MW, Lee K, Toutanova K (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pp 4171–4186

DGSSI (2024) Application security verification framework. https://www.dgssi.gov.ma/en/publications/application-security-verification-framework, accessed 2025-12-08

Goldberg Y, Kozareva Z, Zhang Y (2022) Proceedings of the 2022 conference on empirical methods in natural language processing. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing

Google DeepMind (2025a) Gemini 2.0: Advancements in large-scale multimodal models. Tech. rep., Google, hypothetical source for forthcoming model series.

Google DeepMind (2025b) Technical report for gemini 2.5 series. Tech. rep., Google, hypothetical source for forthcoming model series.

Hey T, Keim J, Koziolek A, Tichy WF (2020) Norbert: Transfer learning for requirements classification. In: 2020 IEEE 28th international requirements engineering conference (RE), IEEE, pp 169–179

Hopkins P, Unger M (2017) What is a subject-matter expert? Journal of Pipeline Engineering 16(4)

Infrastructure PK, Profile TP (2002) Common criteria for information technology security evaluation. National Security Agency

Jeong C (2024) Domain-specialized llm: Financial fine-tuning and utilization method using mistral 7b. Journal of Intelligence and Information Systems 30(1):93–120, DOI 10.13088/jiis.2024.30.1.093, URL http://dx.doi.org/10.13088/jiis.2024.30.1.093

Jindal R, Malhotra R, Jain A (2016) Automated classification of security requirements. In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, pp 2027–2033

JITio (2024) How to use owasp asvs to protect web applications. https://www.jit.io/resources/security-standards/owasp-asvs-to-protect-web-applications, accessed 2025-12-08

Kadebu P, Sikka S, Tyagi RK, Chiurunge P (2023) A classification approach for software requirements towards maintainable security. Scientific African 19:e01496

Karhu K, Kasurinen J, Smolander K (2025) Expectations vs reality–a secondary study on ai adoption in software testing. arXiv preprint arXiv:250404921

Karlsson F, Chatzipetrou P, Gao S, Havstorm TE (2025) How reliable are gpt-4o and llama3. 3-70b in classifying natural language requirements? IEEE Software

Khan R, McLaughlin K, Laverty D, Sezer S (2017) Stride-based threat modeling for cyber-physical systems. In: 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), pp 1–6, DOI 10.1109/

ISGTEurope.2017.8260283
Khan RA, Akbar MA, Rafi S, Almagrabi AO, Alzahrani M (2024) Evaluation of requirement engineering best practices for secure software development in gsd: An ism analysis. Journal of Software: Evolution and Process 36(5):e2594
Knauss E, Houmb S, Schneider K, Islam S, Jürjens J (2011) Supporting requirements engineers in recognising security issues. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, pp 4–18
Labrak Y, Rouvier M, Dufour R (2023) A zero-shot and few-shot study of instruction-finetuned large language models applied to clinical and biomedical tasks. arXiv preprint arXiv:230712114
Lima M, Valle V, Costa E, Lira F, Gadelha B (2019) Software engineering repositories: expanding the promise database. In: Proceedings of the XXXIII Brazilian Symposium on Software Engineering, pp 427–436
Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G (2023a) Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM computing surveys 55(9):1–35
Liu Y, Zhang H, Chen X (2023b) Large language models for requirements classification: Zero-shot and few-shot approaches. In: Proceedings of the 2023 IEEE International Requirements Engineering Conference (RE), IEEE, pp 45–56, DOI 10.1109/RE.2023.00012
Liu Z, Wang H, Xu T, Wang B (2025) Rag-driven multiple assertions generation with large language models. Empirical Software Engineering 30(3):105
Loshchilov I, Hutter F (2019) Decoupled weight decay regularization. arXiv preprint arXiv:171105101 URL https://arxiv.org/abs/1711.05101
Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, et al. (2020) Language models are few-shot learners. arXiv preprint arXiv:200514165 1(3):3
Masoudifard A, Sorond MM, Madadi M, Sabokrou M, Habibi E (2024) Leveraging graph-rag and prompt engineering to enhance llm-based automated requirement traceability and compliance checks. URL https://arxiv.org/abs/2412.08593, 2412.08593
Maturi A, Alshammari R, Alqahtani F, Alqahtani M (2025) Detecting reentrancy vulnerabilities for solidity smart contracts with bidirectional lstm and explainable ai. In: 2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, pp 1–8, DOI 10.1109/ICBC.2025.10926491, URL https://ieeexplore.ieee.org/document/10926491
Muresan S, Nakov P, Villavicencio A (2022) Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers). In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)
Necula SC, Fotache D, Rieder E (2024) Assessing the impact of artificial intelligence tools on employee productivity: insights from a comprehensive survey analysis. Electronics 13(18):3758
OpenAI (2025) GPT-4o mini: Advancing cost-effective intelligence. https://openai.com/index/

gpt-4o-mini-advancing-cost-effective-intelligence/
Pustejovsky J, Stubbs A (2012) Natural Language Annotation for Machine Learning: A guide to corpus-building for applications. " O'Reilly Media, Inc."
Rajbhoj A, Somase A, Kulkarni P, Kulkarni V (2024) Accelerating software development using generative ai: Chatgpt case study. In: Proceedings of the 17th innovations in software engineering conference, pp 1–11
Reqview (2016) Owasp application security verification standard (asvs) template. https://www.reqview.com/doc/asvs-template/, accessed 2025-12-08
Riaz M, Williams L (2012) Security requirements patterns: understanding the science behind the art of pattern writing. In: 2012 Second IEEE International Workshop on Requirements Patterns (RePa), IEEE, pp 29–34
Riaz M, King J, Slankas J, Williams L (2014) Hidden in plain sight: Automatically identifying security requirements from natural language artifacts. In: 2014 IEEE 22nd international requirements engineering conference (RE), IEEE, pp 183–192
Ronanki K, Cabrero-Daniel B, Horkoff J, Berger C (2024) Requirements engineering using generative ai: Prompts and prompting patterns. In: Generative AI for effective software development, Springer, pp 109–127
Schmeelk S, Tao L (2022) A case study of mobile health applications: the owasp risk of insufficient cryptography. Journal of Computer Science Research 4(1):22–31
Schweter S (2020) Türkische BERT modelle. DOI 10.5281/zenodo.4158499, URL https://doi.org/10.5281/zenodo.4158499
Sommerville I, Sawyer P (1997) Requirements engineering: a good practice guide. John Wiley & Sons, Inc.
Souppaya M, Scarfone K, Dodson D (2022) Secure software development framework (ssdf) version 1.1. NIST Special Publication 800(218):800–218
Subahi AF (2023) Bert-based approach for greening software requirements engineering through non-functional requirements. IEEE Access 11:103001–103013
Tan V, Cheh C, Chen B (2021) From application security verification standard (asvs) to regulation compliance: A case study in financial services sector. In: 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), IEEE, pp 69–76, DOI 10.1109/ISSREW53611.2021.00046
The Google Gemma Team (2025) Gemma 3: Next-generation open models for responsible ai. Tech. rep., Google, hypothetical source for forthcoming model series.
Villamizar H, Kalinowski M, Viana M, Fernández DM (2018) A systematic mapping study on security in agile requirements engineering. In: 2018 44th Euromicro conference on software engineering and advanced applications (SEAA), IEEE, pp 454–461
Wang F, Harker A, Edirisinghe M, Parhizkar M (2024) Tackling data scarcity challenge through active learning in materials processing with electrospray.

Advanced Intelligent Systems 6(7):2300798
Wen SF, Katt B (2023) A quantitative security evaluation and analysis model for web applications based on owasp application security verification standard. Computers & Security 135:103532, DOI 10.1016/j.cose.2023.103532
White J, Hays S, Fu Q, Spencer-Smith J, Schmidt DC (2024) Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In: Generative AI for Effective Software Development, Springer, pp 71–108
Ye J, Yao Z, Huang Z, Pan L, Liu J, Bai Y, Xin A, Weichuan L, Che X, Hou L, et al. (2025) How do transformers learn implicit reasoning? In: The Thirty-ninth Annual Conference on Neural Information Processing Systems
Zadenoori MA, Dabrowski J, Alhoshan W, Zhao L, Ferrari A (2025) Large language models (llms) for requirements engineering (re): A systematic literature review. arXiv preprint arXiv:250911446
Zhou X, Li Y, Chen H (2025) The transformer architecture: Foundations and advances. IEEE Transactions on Neural Networks and Learning Systems Forthcoming
Zhu Y, Wang J, Liu K (2025) Emergent abilities of large language models: In-context learning, instruction following, and reasoning. Nature Machine Intelligence In press
Łukasiewicz K, Cygańska S (2019) Security-oriented agile approach with agilesafe and owasp asvs. In: 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, pp 653–662, DOI 10.23919/FedCSIS.2019.00012

A Zero-shot Prompt

ZERO SHOT PROMPT

Persona: You are an experienced cybersecurity specialist with extensive knowledge of the OWASP ASVS 4.0.3-tr and secure software development practices. You are meticulous, detail-oriented, and focused on identifying potential security vulnerabilities and ensuring that applications meet the highest security standards.

****

Task: Classify and analyze project requirements based on their security and functional aspects from a cybersecurity perspective, aligning with the OWASP Application Security Verification Standard (ASVS) 4.0.3-tr. Provide detailed explanations for all classifications and potential security concerns.

****

Instructions:

1. You will receive a list of project requirements in PSV (Pipe Separated Values) format.
2. Each line in the input will represent a requirement in “Requirement Number|Requirement Description” format and will contain:
   * Requirement Number (integer)
   * Requirement Description (string)
3. For each requirement, classify it into the categories defined in the table below, which are derived from the OWASP ASVS 4.0.3-tr.
4. Output the classification in PSV format, including all columns from the table, AND an "Explanation" column within a new line.
5. Use "1" if the requirement aligns with the ASVS category, and "0" if it does not.
6. Provide a concise explanation for each classification, justifying why you assigned a "1" or "0" to each category.
7. Ensure the output has the same number of rows as the input.
8. Use the Extra Information for Choosing Categories to aid classification.

****

Context:

The categories below are based on the OWASP Application Security Verification Standard (ASVS) 4.0.3-tr. Refer to the ASVS documentation for detailed information on each category. The specific ASVS levels (L1, L2, L3) are not explicitly represented in the columns, but the categories themselves align with the ASVS requirements.

****

Categories (Aligned with OWASP ASVS 4.0.3-tr):

V2, Kimlik Doğrulama (Identity Verification) - Covers requirements related to user authentication, password management, and session handling (e.g., ASVS 2.0).
V3, Oturum Yönetimi (Session Management) - Covers requirements related to secure session handling, including session timeouts, protection against session fixation, and secure cookie management (e.g., ASVS

3.0).
V4, Erişim Kontrolleri (Access Controls) - Covers requirements related to authorization, role management, and preventing unauthorized access to resources (e.g., ASVS 4.0).
V5, "Doğrulama, Temizleme ve Şifreleme" (Verification, Cleaning, and Encryption) - Covers requirements related to input validation, output encoding, data sanitization, and cryptographic protection of sensitive data (e.g., ASVS 5.0).
V6, Depolanmış Kriptografi (Stored Cryptography) - Covers requirements for protecting data at rest using cryptography, including key management and secure storage of encryption keys (e.g., ASVS 6.0).
V7, Hata Ayıklama ve Kayıt (Debugging and Logging) - Covers requirements related to secure logging practices, error handling, and preventing information leakage through debug information (e.g., ASVS 7.0).
V8, Veri Koruma (Data Protection) - Covers requirements related to data at rest and in transit protection, including encryption, backups, and data retention policies (e.g., ASVS 8.0).
V9, İletişim (Communication) - Covers requirements related to secure communication protocols, protecting against network attacks, and ensuring data integrity during transmission (e.g., ASVS 9.0).
V10, Zararlı Kod (Malicious Code) - Covers requirements related to preventing and mitigating the risk of malicious code injection, such as cross-site scripting (XSS) and SQL injection (e.g., ASVS 10.0).
V12, Dosya ve Kaynakları (Files and Resources) - Covers requirements related to secure file handling, access control to files and resources, and preventing directory traversal attacks (e.g., ASVS 12.0).
V13, API ve Web Servisleri (API and Web Services) - Covers requirements specific to securing web services, APIs, and microservices, including authentication, authorization, input validation, and error handling (e.g., ASVS 13.0).

****

Extra Information for Choosing Categories (Aligned with OWASP ASVS 4.0.3-tr):

According to given descriptions for kind of requirements, we can decide categories as in PSV:

If Requirement describes a CRUD operation and if it is Create operation|1|1|1|1|0|1|1|1|1|0|1|
If Requirement describes a CRUD operation and if it is Read operation|1|1|1|1|0|1|1|1|1|0|1|
If Requirement describes a CRUD operation and if it is Update operation|1|1|1|1|0|1|1|1|1|0|1|
If Requirement describes a CRUD operation and if it is Delete operation|1|1|1|1|0|0|1|1|1|0|0|0|
If Requirement describes User Types: usage of various user types as role management|1|1|1|1|0|0|0|0|0|0|0|0|
If Requirement describes Store: actions related to storage and backup of the assets at rest, e.g., backing up log files|0|0|0|1|0|1|1|1|0|0|1|0|
If Requirement describes Transfer: actions related to transfer or sharing of the assets, e.g., sending data from one service to another|0|0|0|0|0|1|1|1|1|0|0|1|
If Requirement describes Upload and Download Files etc. operation|1|1|1|1|1|0|1|0|1|1|1|1|
If Requirement describes operation on sensitive data like personal, financial, health etc|1|0|1|1|1|0|1|0|0|0|0|0|

****

Output (PSV- The requirements that are classified):
We need each requirement response in given format below within new line separator:
<Requirement Number>|<Requirement Description>|<V2>|<V3>|<V4>|<V5>|<V6>|<V7>|<V8>|<V9>|<V10>|<V12>|<V13>|<Explanation> format.
and will contain:
- Requirement Number (integer)
- Requirement Description (string)

- <V2> through <V13>: (either 1 (if the requirement aligns with the category) or 0 (if it doesn't))
- Explanation (string)

*****

Input (PSV - The requirements you want to classify):

1|Sistem, kullanıcının konaklama ücret tipi bilgilerini girmesini sağlayacaktır
3|Sistem, kullanıcının konaklama ücret tipi bilgilerini silmesini sağlayacaktır
5|Sistem, kullanıcının oda tiplerini güncellemesini sağlayacaktır
16|Sistem, kullanıcının otele giriş nedeni bilgisini girmesini sağlayacaktır
22|Sistem, kullanıcının çalışma tip bilgisini girmesini sağlayacaktır
28|Sistem, kullanıcının otel rezervasyon para yatırma süre bilgisini girmesini sağlayacaktır
31|Sistem, kullanıcının misafir türü kaydetmesini sağlayacaktır.
37|Sistem, kullanıcının oda tiplerini listelemesini sağlayacaktır.
41|Sistem, kullanıcının otele giriş neden bilgilerini listelemesini sağlayacaktır.
44|Sistem, kullanıcının arananlar bilgisini listelemesini sağlayacaktır.
52|Sistem, kullanıcının yakınlık tanım bilgisini seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
54|Sistem, kullanıcının arananlar bilgisini seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
55|Sistem, kullanıcının otel rezervasyon iptal süre bilgisini seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
65|Sistem, kullanıcının otelin katlarındaki oda doluluk oranlarını seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
70|Sistem, kullanıcının otel raporlarında kullanıcı imza bilgisi girmesini sağlayacaktır.

B Few-shot Prompt

FEW SHOTS PROMPT

Persona: You are an experienced cybersecurity specialist with extensive knowledge of the OWASP ASVS 4.0.3-tr and secure software development practices. You are meticulous, detail-oriented, and focused on identifying potential security vulnerabilities and ensuring that applications meet the highest security standards.

****

Task: Classify and analyze project requirements based on their security and functional aspects from a cybersecurity perspective, aligning with the OWASP Application Security Verification Standard (ASVS) 4.0.3-tr. Provide detailed explanations for all classifications and potential security concerns.

****

Instructions:

1. You will receive a list of project requirements in PSV (Pipe Separated Values) format.
2. Each line in the input will represent a requirement in “Requirement Number|Requirement Description” format and will contain:
   * Requirement Number (integer)
   * Requirement Description (string)
3. For each requirement, classify it into the categories defined in the table below, which are derived from the OWASP ASVS 4.0.3-tr.
4. Output the classification in PSV format, including all columns from the table, AND an "Explanation" column within a new line.
5. Use "1" if the requirement aligns with the ASVS category, and "0" if it does not.
6. Provide a concise explanation for each classification, justifying why you assigned a "1" or "0" to each category.
7. Ensure the output has the same number of rows as the input.
8. Use the Extra Information for Choosing Categories to aid classification.

****

Context:

The categories below are based on the OWASP Application Security Verification Standard (ASVS) 4.0.3-tr. Refer to the ASVS documentation for detailed information on each category. The specific ASVS levels (L1, L2, L3) are not explicitly represented in the columns, but the categories themselves align with the ASVS requirements.

****

Categories (Aligned with OWASP ASVS 4.0.3-tr):

V2, Kimlik Doğrulama (Identity Verification) - Covers requirements related to user authentication, password management, and session handling (e.g., ASVS 2.0).
V3, Oturum Yönetimi (Session Management) - Covers requirements related to secure session handling, including session timeouts, protection against session fixation, and secure cookie management (e.g., ASVS 3.0).

V4, Erişim Kontrolleri (Access Controls) - Covers requirements related to authorization, role management, and preventing unauthorized access to resources (e.g., ASVS 4.0).
V5, "Doğrulama, Temizleme ve Şifreleme" (Verification, Cleaning, and Encryption) - Covers requirements related to input validation, output encoding, data sanitization, and cryptographic protection of sensitive data (e.g., ASVS 5.0).
V6, Depolanmış Kriptografi (Stored Cryptography) - Covers requirements for protecting data at rest using cryptography, including key management and secure storage of encryption keys (e.g., ASVS 6.0).
V7, Hata Ayıklama ve Kayıt (Debugging and Logging) - Covers requirements related to secure logging practices, error handling, and preventing information leakage through debug information (e.g., ASVS 7.0).
V8, Veri Koruma (Data Protection) - Covers requirements related to data at rest and in transit protection, including encryption, backups, and data retention policies (e.g., ASVS 8.0).
V9, İletişim (Communication) - Covers requirements related to secure communication protocols, protecting against network attacks, and ensuring data integrity during transmission (e.g., ASVS 9.0).
V10, Zararlı Kod (Malicious Code) - Covers requirements related to preventing and mitigating the risk of malicious code injection, such as cross-site scripting (XSS) and SQL injection (e.g., ASVS 10.0).
V12, Dosya ve Kaynakları (Files and Resources) - Covers requirements related to secure file handling, access control to files and resources, and preventing directory traversal attacks (e.g., ASVS 12.0).
V13, API ve Web Servisleri (API and Web Services) - Covers requirements specific to securing web services, APIs, and microservices, including authentication, authorization, input validation, and error handling (e.g., ASVS 13.0).

****

Extra Information for Choosing Categories (Aligned with OWASP ASVS 4.0.3-tr):

According to given descriptions for kind of requirements, we can decide categories as in PSV:

If Requirement describes a CRUD operation and if it is Create operation|1|1|1|1|0|1|1|1|1|0|1|
If Requirement describes a CRUD operation and if it is Read operation|1|1|1|1|0|1|1|1|1|0|1|
If Requirement describes a CRUD operation and if it is Update operation|1|1|1|1|0|1|1|1|1|0|1|
If Requirement describes a CRUD operation and if it is Delete operation|1|1|1|0|0|1|1|1|0|0|0|
If Requirement describes User Types: usage of various user types as role management|1|1|1|0|0|0|0|0|0|0|0|
If Requirement describes Store: actions related to storage and backup of the assets at rest, e.g., backing up log files|0|0|0|1|0|1|1|0|0|1|0|
If Requirement describes Transfer: actions related to transfer or sharing of the assets, e.g., sending data from one service to another|0|0|0|0|0|1|1|1|0|0|1|
If Requirement describes Upload and Download Files etc. operation|1|1|1|1|0|1|0|1|1|1|1|
If Requirement describes operation on sensitive data like personal, financial, health etc|1|0|1|1|1|0|1|0|0|0|0|

****

Few-Shot Examples (PSV):
1|Sistem, yetkili kullanıcının hukuk dosyası türü kayıt etmesini sağlayacaktır.|1|1|1|1|0|1|1|1|1|0|1| Identity Verification (V2): The requirement specifies "yetkili kullanıcı," implying a need for authentication to verify the user's identity and ensure they are authorized to perform this action.Session Management (V3): Managing the user session after authentication is essential to maintain the user's authenticated status and prevent unauthorized acces. Access Controls (V4): The requirement explicitly states that only "yetkili kullanıcı" (authorized user) can perform this action, indicating the need for access controls to restrict unauthorized users.

Verification, Cleaning, and Encryption (V5): Recording the "hukuk dosyası türü" (legal file type) likely involves validation and data integrity checks to ensure the information is accurate and complete.
Debugging and Logging (V7): Logging this action is important for auditing and security monitoring purposes, allowing for tracking of who registered what file type and when.
Data Protection (V8): The stored file type data should be protected with appropriate security measures, such as encryption or access controls, to prevent unauthorized access or modification.
Communication (V9): Secure communication is needed if the registration action involves data transfer over a network to protect the confidentiality and integrity of the data.
Malicious Code (V10): Input validation should be implemented to prevent malicious code injection, ensuring that the file type input is properly sanitized and does not contain any malicious scripts or commands.
Files and Resources (V12): Secure file handling and storage procedures should be in place to protect the files and resources associated with the legal file type registration process.
API and Web Services (V13): If an API is used for the registration process, appropriate security measures should be implemented to protect the API endpoints and the data transmitted through the API.
Analysis & Recommendations: The exact method for verifying user authorization needs to be defined to prevent privilege escalation. File type validation is crucial. Input sanitization is needed to avoid potential injection attacks.

2|Sistem, kullanıcının kullanılmamış hukuk dosyası türünü güncellemesini sağlayacaktır.|1|1|1|1|0|1|1|1|1|0|1| Similar to the previous example, this requirement involves identity verification (V2), session management (V3), access control (V4), verification and validation (V5), logging (V7), data protection (V8), secure communication (V9), protection against malicious code (V10), and file and resource management (V12).
Analysis & Recommendations: Version control or backups are necessary in case of accidental data corruption during updates. Consider implementing input validation for the new file data to ensure it meets requirements and protect against injection attacks.

3|Sistem, kullanıcının kullanılmamış olan hukuk dosyası türünü silmesini sağlayacaktır.|1|1|1|0|0|1|1|1|0|0|0| Identity verification (V2), session management (V3), and access control (V4) are needed to ensure only authorized users can delete file types.
Debugging and Logging (V7): Logging deletion actions is essential for auditing and security monitoring.
Data Protection (V8): Appropriate data protection measures, including potential backups or recovery mechanisms, should be in place to prevent accidental or malicious data loss.
Communication (V9): Secure communication is vital if the deletion process involves transmitting data over a network.
Analysis & Recommendations: A soft delete (marking as deleted but not actually removing) is recommended along with a process for permanently deleting files. Access control on who can permanently delete needs careful configuration.

4|Sistem, kullanıcının hukuk dosyası türlerini sorgulamasını ve listelemesini sağlayacaktır.|1|1|1|1|0|1|1|1|1|0|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Verification, Cleaning, and Encryption (V5), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Malicious Code (V10), Files and Resources (V12), API and Web Services (V13): Standard security practices apply to data retrieval as well as creation or modification.
Analysis & Recommendations: Protect against SQL injection and other query-based injection attacks by using parameterized queries or prepared statements. Careful definition of permissions regarding who can query what is essential to prevent information leakage.

5|Sistem, kullanıcının var olan bir hukuk dosyası türüne hukuk dosyası alt türü tanımlamasını sağlayacaktır.|1|1|1|1|0|1|1|1|1|0|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Verification, Cleaning, and Encryption (V5), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Malicious Code (V10), Files and Resources (V12), API and Web Services (V13): Standard security practices apply to data retrieval as well as creation or modification.
Analysis & Recommendations: Ensure proper input validation and data sanitization for the new subtype. Prevent naming collisions to avoid confusion.

6|Sistem, kullanıcının hukuk dosyası alt türünü aktife/pasife çekmesini sağlayacaktır.|1|1|1|0|0|1|1|1|0|0|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Files and Resources (V12): Standard security practices apply.
Analysis & Recommendations: Consider an approval process before deactivating a subtype, and ensure proper documentation and versioning.

7|Sistem, kullanıcının dava dosyasına bir veya birden fazla davacı tanımlamasını sağlayacaktır. |1|1|1|1|0|1|0|1|0|0|1| Identity verification (V2), session management (V3), access control (V4), verification, and validation (V5), logging (V7), secure communication (V9), and file and resource management (V12) are all relevant.
Analysis & Recommendations: Validation is critical for plaintiff information, especially if it involves personal data. Protect against Cross-Site Scripting (XSS) and other injection attacks.

8|Sistem, davaların konu kodunun zorunlu olmasını sağlayacaktır.|0|0|0|1|0|1|0|0|0|0|0| Verification, Cleaning, and Encryption (V5), Debugging and Logging (V7): Focuses on data integrity.
Analysis & Recommendations: Client-side and server-side validation is necessary. Standardize the "konu kodu" format.

9|Sistem, kullanıcının hukuk dosyasında görüntülediği evrakın çıktısını almasını sağlayacaktır.|1|1|1|0|0|1|1|1|0|1|0| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Files and Resources (V12): Security for document access and printing.
Analysis & Recommendations: Watermarks or other identifying marks on printouts can deter unauthorized distribution. Secure the print queue to prevent unauthorized access.

10|Sistem, sistem soru önergelerine otomatik numara vermesini sağlayacaktır.|0|0|0|0|0|1|0|0|0|0|0| This requirement primarily focuses on logging (V7), as the automatic numbering process should be tracked for troubleshooting and potential issues.
Analysis & Recommendations: Implement proper logging and error handling for the numbering process to ensure accountability and traceability.

11|Sistem, kullanıcının dava dosyası evrak listesinden seçtiği bir evrakı UYAP entegrasyonu ile çekmesini sağlayacaktır.|1|1|1|0|0|1|1|1|0|1|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Files and Resources (V12): Standard practices for data retrieval.
API and Web Services (V13): Secure integration with external systems is crucial.
Analysis & Recommendations: Ensure proper authentication and authorization mechanisms are in place for accessing and retrieving documents from UYAP. Encrypt data both in transit and at rest.

12|Sistem, toplantı salonunun tahsis edilmiş olduğu tarih saat ile çakışan toplantı salonu tahsis taleplerinin oluşturulmasını engelleyecektir.|0|0|0|0|0|1|0|0|0|0|1| Debugging and Logging (V7): Logging booking

requests and conflicts is important for debugging and auditing.
API and Web Services (V13): If bookings are made through an API, security best practices should be followed.
Analysis & Recommendations: Implement robust input validation to prevent invalid or malicious booking requests. Consider implementing rate limiting to protect against denial-of-service attacks.

13|Sistem, kullanıcının, sayısallaştırılmış genel keşif incelemeleri dokümanını sisteme yüklemesini sağlayacaktır.|1|1|1|1|0|1|1|1|1|1|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Verification, Cleaning, and Encryption (V5), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Malicious Code (V10), Files and Resources (V12), API and Web Services (V13): Standard security practices for data retrieval as well as creation or modification.
Analysis & Recommendations: Implement robust file upload controls, including file type validation, size limits, and virus scanning. Ensure proper access control to prevent unauthorized uploads or downloads.

14|Sistem, kullanıcının katılımcı listesinden katılımcı seçerek, o katılımcıya ait detay bilgilerini görmesini sağlayacaktır.|1|1|1|0|0|1|1|1|0|0|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Files and Resources (V12): Standard security practices apply for access control and data protection.
Analysis & Recommendations: Ensure proper authorization checks before allowing users to view participant details. Implement data masking or obfuscation techniques to protect sensitive information.

15|Sistem, kullanıcının eğitici ödeme, iletişim ve ulaşım bilgilerini kaydetmesini sağlayacaktır.|1|1|1|1|0|1|1|1|1|0|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Verification, Cleaning, and Encryption (V5), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Malicious Code (V10), Files and Resources (V12), API and Web Services (V13): Standard security practices apply.
Analysis & Recommendations: Encrypt sensitive trainer data at rest and in transit. Implement strict access control to protect this data.

16|Sistem, intranet ve internet ağındaki kullanıcıların eğitim plan duyurusunu indirmesini sağlayacaktır.|1|1|1|0|0|1|1|1|0|1|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7), Data Protection (V8), Communication (V9), Files and Resources (V12): Standard practices for data retrieval.
API and Web Services (V13): Securely provide files for download and ensure integrity.
Analysis & Recommendations: Validate user authorization before allowing downloads. Implement checksums or digital signatures to ensure file integrity.

17|Sistem, uygulama yazılımında dış ortamlar için geliştirilecek web servislerine HTTP ve/veya HTTPS protokolü üzerinden erişilebilmesi desteğini sağlayacaktır.|0|0|0|0|0|1|0|1|0|0|1| Debugging and Logging (V7): Log access to web services for debugging and security monitoring.
Communication (V9): Secure communication is essential for web services.
API and Web Services (V13): Securely expose and consume web services.
Analysis & Recommendations: Use HTTPS to protect data in transit. Implement authentication and authorization for web services.

18|Sistem, uygulama yazılımının modüler ve dağıtık bir yapıda geliştirilebilmesini sağlayacaktır.|0|0|0|0|0|0|0|0|0|0|0| This is an architectural requirement that doesn't directly relate to ASVS categories.

19|Sistem, uygulama yazılımının, Spring framework ile MVC (Model-View-Controller) mimari tasarım şablonuna uygun geliştirilebilmesini sağlayacaktır.|0|0|0|0|0|0|0|0|0|0| This is a technical requirement that doesn't directly relate to ASVS categories.

20|Sistem, kullanıcının veri girişi sırasında, otomatik olarak maskeleme desteği sağlayacaktır.|1|1|1|1|0|1|1|0|0|0|0| Identity Verification (V2), Session Management (V3), Access Controls (V4), Verification, Cleaning, and Encryption (V5), Debugging and Logging (V7), Data Protection (V8), Communication (V9): Standard practices for data input and protection.
Analysis & Recommendations: Ensure proper masking techniques are used to protect sensitive data. Consider using different masking techniques for different data types.

21|Sistem, tarayıcı üzerinde gösterilen filigranı, kullanıcı adı ve IP bilgisi gibi alanların şifrelenmesi ile oluşturacaktır.|1|0|0|1|0|1|0|0|0|0|0| Identity Verification (V2): The watermark is linked to the user's identity.
Verification, Cleaning, and Encryption (V5): Encrypting the watermark data ensures its integrity and confidentiality.
Data Protection (V8): Protecting the watermark information is important.
Analysis & Recommendations: Use strong encryption algorithms for the watermark data. Ensure the watermark is securely generated and displayed to prevent tampering.

22|Sistem, yöneticinin, görevli personel için kullanıcı hesabı oluşturabilmesini sağlayacaktır.|1|1|1|0|0|1|0|1|0|0|1| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7), Data Protection (V8), Communication (V9): Standard practices for data retrieval as well as creation or modification.
Files and Resources (V12): Administering user accounts involves managing files and resources.
API and Web Services (V13): Account creation may involve API calls.
Analysis & Recommendations: Implement strong password policies and account lockout mechanisms. Ensure proper segregation of duties for managing accounts.

23|Sistem, yöneticinin, kullanıcının aktif dizinde üye olabileceği güvenlik gruplarını seçebilmesini sağlayacaktır.|1|1|1|0|0|1|0|1|0|0|0| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7), Data Protection (V8): Standard security practices apply.
Analysis & Recommendations: Regularly review and update security group memberships to ensure users have appropriate access. Use a role-based access control model to manage permissions effectively.

24|Sistem, güçlü şifrelerin kullanılmasını zorunlu kılacaktır.|0|0|0|0|0|0|0|0|1|0|0| Malicious Code (V10): Enforcing strong passwords helps prevent brute-force and dictionary attacks.
Analysis & Recommendations: Define a clear password policy with requirements for length, complexity, and regular updates. Consider using a password manager to help users generate and store strong passwords.

25|Sistem, kullanıcıların şifrelerini yönetibilmesine olanak sağlayacaktır.|1|1|1|1|0|1|1|1|0|0|0| Identity Verification (V2), Session Management (V3), Access Controls (V4), Verification, Cleaning, and Encryption (V5), Debugging and Logging (V7), Data Protection (V8), Communication (V9): Standard security practices apply.
Analysis & Recommendations: Implement secure password reset and recovery mechanisms. Use strong encryption for storing and transmitting passwords. Consider using multi-factor authentication for added security.

26|Sistem, kullanıcının 10 dk boyunca işlem yapmaması durumunda oturumunu

sonlandıracaktır.|0|1|0|0|0|1|0|0|0|0|0| Session Management (V3), Debugging and Logging (V7): Session timeouts enhance security and should be logged.
Analysis & Recommendations: Enforce session timeouts to prevent unauthorized access from unattended sessions. Log session termination events for security monitoring.

27|Sistem, çerez kullanımında gerekli tüm güvenlik ayarlarının uygulanmasını zorunlu kılacaktır.|0|1|0|0|0|1|1|0|0|0|0| Session Management (V3), Debugging and Logging (V7), Data Protection (V8): Secure cookie handling is crucial for protecting user sessions.
Analysis & Recommendations: Use secure cookies with the HttpOnly flag. Encrypt sensitive data stored in cookies. Implement proper cookie expiration policies.

28|Sistem, kullanıcıların en az yetki prensibine göre işlem yapabilmesini sağlayacaktır.|1|1|1|0|0|1|0|0|0|0|0| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7): Enforcing least privilege requires proper authentication, authorization, and logging.
Analysis & Recommendations: Implement role-based access control (RBAC) to manage user permissions effectively. Regularly review and update user permissions.

29|Sistem, kullanıcı hareketlerinin kaydını tutacaktır.|1|1|1|0|0|1|0|0|0|0|0| Identity Verification (V2), Session Management (V3), Access Controls (V4), Debugging and Logging (V7): User activity logging is crucial for security auditing and incident response.
Analysis & Recommendations: Log user actions with sufficient detail for security analysis. Protect log files from unauthorized access and modification.

30|Sistemde kullanılan eklentilerin güvenlik doğrulaması yapılacaktır.|0|0|0|0|0|1|1|0|1|0|1| Debugging and Logging (V7), Data Protection (V8), Malicious Code (V10), Files and Resources (V12), API and Web Services (V13): Plugin security is crucial to prevent vulnerabilities.
Analysis & Recommendations: Validate plugins before installation and use. Regularly update plugins to patch security vulnerabilities. Restrict plugin permissions to limit potential damage.
****
Output (PSV- The requirements that are classified):
We need each requirement response in given format below within new line separator:
<Requirement Number>|<Requirement Description>|<V2>|<V3>|<V4>|<V5>|<V6>|<V7>|<V8>|<V9>|<V10>|<V12>|<V13>|<Explanation> format.
and will contain:
- Requirement Number (integer)
- Requirement Description (string)
- <V2> through <V13>: (either 1 (if the requirement aligns with the category) or 0 (if it doesn't))
- Explanation (string)
*****
Input (PSV - The requirements you want to classify):
1|Sistem, kullanıcının konaklama ücret tipi bilgilerini girmesini sağlayacaktır
3|Sistem, kullanıcının konaklama ücret tipi bilgilerini silmesini sağlayacaktır
5|Sistem, kullanıcının oda tiplerini güncellemesini sağlayacaktır
16|Sistem, kullanıcının otele giriş nedeni bilgisini girmesini sağlayacaktır
22|Sistem, kullanıcının çalışma tip bilgisini girmesini sağlayacaktır
28|Sistem, kullanıcının otel rezervasyon para yatırma süre bilgisini girmesini sağlayacaktır
31|Sistem, kullanıcının misafir türü kaydetmesini sağlayacaktır.
37|Sistem, kullanıcının oda tiplerini listelemesini sağlayacaktır.
41|Sistem, kullanıcının otele giriş neden bilgilerini listelemesini sağlayacaktır.

44|Sistem, kullanıcının arananlar bilgisini listelemesini sağlayacaktır.
52|Sistem, kullanıcının yakınlık tanım bilgisini seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
54|Sistem, kullanıcının arananlar bilgisini seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
55|Sistem, kullanıcının otel rezervasyon iptal süre bilgisini seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
65|Sistem, kullanıcının otelin katlarındaki oda doluluk oranlarını seçilen kriterlere göre sorgulayabilmesini sağlayacaktır.
70|Sistem, kullanıcının otel raporlarında kullanıcı imza bilgisi girmesini sağlayacaktır.

C Test Results of the Bert Models

Test Results of the BERT models and Naïve Baseline used in the study

bert-base-multilingual-uncased	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.984	0.982	0.982	0.975	0.744	0.985	0.980	0.984	0.983	0.864	0.991
RECALL	0.997	0.995	0.995	0.984	0.842	0.995	1.000	1.000	0.983	0.704	0.994
F1	0.990	0.988	0.988	0.979	0.790	0.990	0.990	0.992	0.983	0.776	0.992
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

bert-base-turkish-cased	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.974	0.974	0.980	0.975	0.756	0.985	0.982	0.974	0.980	0.800	0.972
RECALL	0.997	0.997	1.000	0.994	0.816	0.997	1.000	1.000	0.977	0.593	0.994
F1	0.985	0.985	0.990	0.984	0.785	0.991	0.991	0.987	0.978	0.681	0.983
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

bert-base-uncased	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.982	0.979	0.982	0.981	0.781	0.985	0.980	0.974	0.977	0.813	0.978
RECALL	0.995	0.995	0.995	0.975	0.658	0.992	0.997	0.997	0.987	0.481	0.987
F1	0.988	0.987	0.988	0.978	0.714	0.988	0.988	0.985	0.982	0.604	0.982
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

SecureBERT	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.974	0.977	0.980	0.972	0.707	0.985	0.977	0.974	0.964	0.700	0.969
RECALL	0.995	0.995	0.997	0.984	0.763	0.997	1.000	0.997	0.990	0.519	0.997
F1	0.984	0.986	0.988	0.978	0.734	0.991	0.988	0.985	0.977	0.596	0.983
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Naive Baseline	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.960	0.950	0.960	0.790	0.000	0.980	0.970	0.960	0.750	0.000	0.800
RECALL	1.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	1.000	0.000	1.000
F1	0.980	0.980	0.980	0.880	0.000	0.990	0.980	0.980	0.860	0.000	0.890
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

D Test Results of the LLMs

Test Results of the LLMs used in the study

Meta_Llama_3_70b_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.969	0.961	0.974	0.897	0.077	0.967	0.946	0.956	0.849	0.258	0.917
RECALL	0.987	0.982	0.984	0.968	0.077	0.979	0.964	0.974	0.957	0.286	0.978
F1	0.978	0.971	0.979	0.931	0.077	0.973	0.955	0.965	0.900	0.271	0.947
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Meta_Llama_3_70b_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.974	0.964	0.974	0.875	0.077	0.977	0.956	0.956	0.882	0.214	0.919
RECALL	0.984	0.976	0.979	0.889	0.077	0.987	0.969	0.961	0.897	0.214	0.959
F1	0.979	0.970	0.977	0.882	0.077	0.982	0.962	0.958	0.889	0.214	0.938
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

DeepSeek_R1_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.953	0.927	0.972	0.876	0.082	0.926	0.910	0.907	0.875	0.455	0.887
RECALL	0.963	0.937	0.979	0.940	0.205	0.933	0.924	0.921	0.930	0.536	0.912
F1	0.958	0.932	0.975	0.907	0.117	0.930	0.917	0.914	0.901	0.492	0.899
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

DeepSeek_R1_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.972	0.959	0.972	0.922	0.143	0.982	0.964	0.930	0.909	0.200	0.880
RECALL	0.987	0.976	0.982	0.940	0.154	0.992	0.982	0.942	0.930	0.750	0.899
F1	0.979	0.967	0.977	0.931	0.148	0.987	0.973	0.936	0.919	0.316	0.890
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemini_2_5_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.967	0.939	0.972	0.882	0.280	0.960	0.962	0.954	0.783	0.479	0.884
RECALL	0.990	0.963	0.990	0.994	0.718	0.977	0.984	0.982	0.900	0.821	0.981
F1	0.978	0.951	0.981	0.934	0.403	0.968	0.973	0.968	0.837	0.605	0.930
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Test Results of the LLMs used in the study

Gemini_2_5_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.979	0.972	0.980	0.914	0.229	0.985	0.972	0.974	0.868	0.203	0.900
RECALL	0.997	0.992	0.992	0.984	0.641	0.995	0.992	0.995	0.967	0.964	0.994
F1	0.988	0.982	0.986	0.948	0.338	0.990	0.982	0.984	0.915	0.335	0.945
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemini_2_0_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.967	0.940	0.972	0.894	0.819	0.960	0.962	0.955	0.812	0.937	0.897
RECALL	0.967	0.939	0.972	0.882	0.280	0.960	0.962	0.954	0.783	0.479	0.884
F1	0.990	0.963	0.990	0.994	0.718	0.977	0.984	0.982	0.900	0.821	0.981
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemini_2_0_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.982	0.972	0.985	0.922	0.754	0.985	0.967	0.975	0.894	0.764	0.917
RECALL	0.982	0.972	0.985	0.910	0.210	0.985	0.967	0.974	0.874	0.223	0.905
F1	0.997	0.990	0.992	0.990	0.667	0.995	0.987	0.995	0.973	0.964	0.994
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemini_2_0_Pro_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.975	0.902	0.972	0.910	0.837	0.910	0.952	0.905	0.847	0.942	0.899
RECALL	0.974	0.900	0.972	0.897	0.188	0.909	0.952	0.902	0.820	0.410	0.881
F1	0.992	0.919	0.992	0.990	0.385	0.923	0.971	0.919	0.923	0.571	0.934
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemini_2_0_Pro_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.975	0.972	0.975	0.970	0.917	0.985	0.975	0.975	0.930	0.869	0.962
RECALL	0.974	0.972	0.974	0.963	0.175	0.985	0.974	0.974	0.911	0.350	0.955
F1	0.992	0.992	0.990	0.987	0.179	0.997	0.995	0.995	0.960	1.000	0.994
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Test Results of the LLMs used in the study

Gemma_3_27b_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.930	0.891	0.944	0.919	0.102	0.944	0.952	0.894	0.823	0.415	0.862
RECALL	0.942	0.906	0.953	0.975	0.128	0.954	0.971	0.906	0.883	0.607	0.884
F1	0.936	0.898	0.948	0.946	0.114	0.949	0.961	0.900	0.852	0.493	0.873
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemma_3_27b_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.966	0.961	0.974	0.894	0.128	0.969	0.954	0.927	0.883	0.226	0.882
RECALL	0.971	0.971	0.982	0.911	0.128	0.977	0.969	0.935	0.903	0.250	0.896
F1	0.969	0.966	0.978	0.903	0.128	0.973	0.961	0.931	0.893	0.237	0.889
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemma_3_4b_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.812	0.743	0.770	0.663	0.047	0.710	0.708	0.708	0.645	0.206	0.603
RECALL	0.838	0.766	0.788	0.762	0.051	0.722	0.727	0.725	0.743	0.250	0.664
F1	0.825	0.755	0.778	0.709	0.049	0.716	0.717	0.717	0.690	0.226	0.632
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gemma_3_4b_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.929	0.888	0.914	0.760	0.044	0.924	0.926	0.908	0.799	0.167	0.803
RECALL	0.955	0.916	0.933	0.876	0.051	0.941	0.951	0.929	0.903	0.179	0.896
F1	0.942	0.902	0.923	0.814	0.048	0.932	0.938	0.918	0.848	0.172	0.847
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gpt_4o_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.943	0.922	0.967	0.905	0.214	0.903	0.918	0.906	0.876	0.250	0.880
RECALL	0.950	0.932	0.974	0.943	0.231	0.907	0.927	0.914	0.920	0.250	0.899
F1	0.947	0.927	0.970	0.924	0.222	0.905	0.922	0.910	0.898	0.250	0.890
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Test Results of the LLMs used in the study

Gpt_4o_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.969	0.956	0.969	0.946	0.205	0.974	0.961	0.951	0.928	0.357	0.938
RECALL	0.982	0.971	0.977	0.949	0.205	0.982	0.974	0.961	0.943	0.357	0.953
F1	0.975	0.964	0.973	0.948	0.205	0.978	0.968	0.956	0.936	0.357	0.945
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gpt_4o_mini_Zero_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.953	0.865	0.928	0.829	0.254	0.824	0.671	0.779	0.659	0.093	0.720
RECALL	0.963	0.877	0.938	0.863	0.385	0.828	0.680	0.785	0.690	0.250	0.736
F1	0.958	0.871	0.933	0.846	0.306	0.826	0.675	0.782	0.674	0.136	0.728
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398

Gpt_4o_mini_Few_Shot	V2, Authentication Verification Requirements	V3, Session Management Verification Requirements	V4, Access Control Verification Requirements	V5, Validation, Sanitization and Encoding Verification Requirements	V6, Stored Cryptography Verification Requirements	V7, Error Handling and Logging Verification Requirements	V8, Data Protection Verification Requirements	V9, Communications Verification Requirements	V10, Malicious Code Verification Requirements	V12, File and Resources Verification Requirements	V13, API and Web Service Verification Requirements
PRECISION	0.969	0.956	0.964	0.821	0.128	0.959	0.871	0.862	0.768	0.179	0.813
RECALL	0.976	0.966	0.969	0.829	0.128	0.964	0.878	0.866	0.773	0.179	0.821
F1	0.973	0.961	0.966	0.825	0.128	0.962	0.874	0.864	0.771	0.179	0.817
SUPPORT	382	381	385	314	38	389	384	380	299	27	317
TOTAL	398	398	398	398	398	398	398	398	398	398	398