Automated Threat Modeling using Artificial Intelligence on User Stories within the SDLC to Generate Security Tasks

Shantu Asif Hossain
SAP America Inc., Newtown Square, USA

Shantu_33_cse@yahoo.com

Abstract: This research presents an AI-driven system that integrates automated threat modeling directly into the Software Development Lifecycle (SDLC) during the early user story creation phase. Traditional threat modeling is often manual, delayed, and disconnected from developer workflows, resulting in missed vulnerabilities and reactive security measures. The proposed system employs a Large Language Model (LLM)-based Threat Modeling Engine to analyze user stories-textual descriptions of software features from an end-user perspective-and identify potential security threats. Leveraging advanced LLM algorithms, the system correlates detected risks with known threat patterns (e.g., STRIDE) and dynamically maps them to multiple pluggable security and compliance standards such as NIST CSF, ISO 27001, PCI DSS, HIPAA, SOC 2, OWASP, and GDPR. The engine automatically generates prioritized, technical security tasks aligned with these standards, which are seamlessly integrated into popular development tools like Jira, GitHub Issues, or Azure DevOps. This process enables proactive, traceable, and consistent enforcement of security controls throughout the development workflow, reducing human error and enhancing compliance with relevant regulations. A human-in-the-loop approval mechanism ensures full oversight and iterative refinement of threat modeling outputs. Furthermore, the system parses security standard documents in native formats (e.g., PDFs) to maintain up-to-date mappings without manual intervention. By embedding intelligent threat mitigation early in the SDLC, this research improves software security posture, development efficiency, and compliance adherence. It addresses a critical gap in current DevSecOps practices by automating and contextualizing security task generation from user stories, enabling development teams to build secure, compliant software aligned with national and international cybersecurity frameworks.

Keywords: Automated threat modeling, STRIDE, SDLC, Compliance automation, DevSecOps

1. Introduction

Software systems increasingly face threats that exploit gaps between design and implementation. Threat modeling is designed to systematically identify, classify, and mitigate potential security risks before they materialize, and it is typically applied during the early phases of the SDLC, such as requirements engineering and architectural design. Threat modeling is critical for anticipating vulnerabilities early, but practice often lags due to manual effort, timing, and weak integration with developer workflows. Recent work shows both the promise of LLMs for security tasks and the risks they introduce, motivating careful engineering of AI-assisted pipelines that are accurate, traceable, and compliant (Ferrag et al., 2024; Tete, 2024).

Large Language Models (LLMs) are increasingly applied to software engineering tasks such as code completion, vulnerability detection, and natural language requirements analysis. Their ability to parse user stories and generate structured outputs suggests an opportunity to embed automated threat modeling directly into developer workflows.

However, prior approaches have struggled with three interrelated challenges: (RQ1) effectively identifying security threats from user stories despite their informal and often ambiguous nature, (RQ2) aligning those threats with overlapping and frequently updated compliance standards, and (RQ3) ensuring that generated security tasks can be seamlessly integrated into development toolchains while preserving traceability and auditability. This research is structured around three questions, which are presented in Section 3.

Research contributions:

- **Threat modeling (RQ1):** An AI-driven engine derives STRIDE-based threats directly from user stories, producing actionable, traceable subtasks early in the SDLC.
- **Compliance alignment (RQ2):** Generated subtasks map consistently to major security and compliance frameworks (NIST CSF, ISO/IEC 27001, PCI DSS, HIPAA, SOC 2, OWASP ASVS, GDPR), ensuring control-level accuracy and adaptability as standards evolve.
- **Workflow integration (RQ3):** A system architecture processes user stories, maps them to standards, and outputs candidate tasks. Human reviewers validate these tasks before backlog entry, ensuring traceability, auditability, and readiness for agile tools like Jira.

2. Background and Related Work

2.1 Threat Modeling and STRIDE

Threat modeling provides a structured methodology for anticipating vulnerabilities. The STRIDE framework-Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege - remains widely used. However, STRIDE is under-utilized in early SDLC phases due to the high manual effort involved (Jedrzejewski et al., 2024).

2.2 Compliance Frameworks

Software systems must address overlapping frameworks including NIST CSF, ISO/IEC 27001, PCI DSS, HIPAA, SOC 2, OWASP ASVS, and GDPR. Mapping requirements across these frameworks is complex and error-prone (Kosenkov et al., 2025). Although official crosswalks exist (e.g., NIST SP 800-53 to ISO/IEC 27001), they remain largely static (Mussmann, 2021; NIST, 2023). Security Compass (2023) and others call for more dynamic mapping.

2.3 AI for DevSecOps

Recent surveys (Ferrag et al., 2024; Prates et al., 2025) highlight how LLMs are being integrated into vulnerability detection, policy extraction, and security automation. Koball et al. (2024) emphasize both their utility and new risks (e.g., prompt injection, data leakage). Work by Parvathinathan et al. (2025) and Yigit et al. (2025) demonstrates AI-powered continuous security, while UK DSIT (2024) highlights risks of LLM deployment.

2.4 User Stories and Automated Analysis

User stories are central to agile development. Dalpiaz (2025) and Casillo et al. (2022) showed that NLP can extract requirements quality and privacy needs. Robeer et al. (2016) extracted conceptual models from user stories, while Aboukadri et al. (2025) applied RAG-augmented LLMs to derive access-control policies. These works validate user stories as viable inputs for automated reasoning.

2.5 LLMs for Threat Modeling

Elsharef et al. (2024) investigated LLMs for facilitating threat identification. Jedrzejewski et al. (2025) studied STRIDE-based classification for network systems, and Nagaraja et al. (2025) modelled threats in LLM-based healthcare. Yang (2024) fine-tuned LLMs for banking, while Jedrzejewski, Fucci and Adamov (2025) introduced ThreMoLIA, a framework that produced structured threat outputs for large language model–integrated applications. These works demonstrate feasibility, but integration into developer workflows and compliance-aware task generation remains limited.

2.6 Research Gap

There is no unified approach that: (1) ingests user stories, (2) performs STRIDE analysis, (3) maps findings across multiple frameworks, and (4) emits developer-ready tasks with traceability inside issue trackers. This gap motivates our work.

3. Research Questions

3.1 RQ1 – Threat Modeling Capability

Can a large-language-model–powered engine derive meaningful STRIDE-based threats directly from user stories and express them as actionable, traceable security subtasks?

3.2 RQ2 – Compliance Alignment and Sustainability

Can the system automatically map detected threats to established security and compliance standards (e.g., NIST CSF, ISO 27001, PCI DSS, HIPAA, SOC 2, OWASP, GDPR) with measurable consistency*?*

3.3 RQ3 – Workflow Integration and Adoption

Can AI-generated security tasks from user stories be integrated into issue-tracking systems (e.g., Jira, GitHub Issues, Azure DevOps) while maintaining traceability and compliance alignment?

4. Methodology

4.1 Dataset Construction

To ground the study in realistic development practices, user stories were exported directly from Jira in CSV format (see Appendix A). This export preserved Jira's native schema - including fields such as Issue key, Issue id, Status, Summary, Description, Issue Type, Priority etc. Maintaining this structure ensured that the dataset accurately reflected real developer workflows, avoided artificial formatting bias, and allowed generated outputs to be validated in the same environment used by practitioners.

4.2 Motivation and Rationale

User stories represent the earliest and most consistent artifacts of agile workflows, making them the most practical entry point for automated threat modeling. By maintaining Jira-like structure, the evaluation reflects how such a system would operate in practice without introducing artificial complexity.

- For RQ1, the dataset tested whether the engine could generate STRIDE-based threats directly from user stories.
- For RQ2, it allowed mapping the generated subtasks to compliance frameworks, validating regulatory relevance.
- For RQ3: Workflow integration was simulated by formatting system outputs in Jira-compatible JSON. Each generated subtask included its STRIDE category, threat description, risk assessment (E1–E3, D1–D4, N1–N2), security task title with acceptance criteria, and compliance references. This approach would allow integration into agile workflows.

4.2.1 Reference annotations

Table 1 presents the baseline system prompt used for STRIDE threat analysis. The prompt instructs the model to output a risk-ordered (High to Low) table of unique mitigation findings (security subtasks), including assets, short threat scenarios, qualitative risk factors (E1–E3, D1–D4, N1–N2), acceptance criteria, and mappings to NIST CSF, ISO 27001, PCI DSS, HIPAA, SOC 2, OWASP, and GDPR.

The design of this baseline was informed by the OWASP Threat Modeling Process (OWASP, (n.d.-b)) to ensure methodological rigor and reproducibility. This expert baseline was then used to benchmark all evaluated LLMs (see Section 5).

Table 1: System Prompt for Automated Threat Modeling using STRIDE

System Prompt
As an experienced security architect and certified threat modeling expert, do the STRIDE threat modeling to generate security sub tasks, each sub task is treated one finding from the threat modeling exercise. Generate as much security sub tasks as required uniquely as mitigation. Your output should have following table format and Order by RISK = High to Low:
- ID (e.g., 1,2,3 etc.)
- Issue_key (e.g., {US/Bug/RS}-{id})
- Security_Task_ID (e.g. 1-100)
- STRIDE Category (e.g., Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege)
- Assets (e.g., a descriptive name that clearly identifies the asset)
- Threat Description (explain shortly what the threat with a meaningful short scenario is description)
- Risk (e.g., High, Medium, or Low risk calculate based on E1-E3, D1-D4, N1-N2 mentioned below. LIST like E1-D4=Yes/No/NA, N1-N2=5)
- Security Task Title (e.g., Implement, Protect etc.)
- User Story Acceptance Criteria as Mitigation (bullet points)
- NIST CFS Reference (e.g., each reference should have title with its equivalent id e.g. PR.AA-01: Identities and credentials for authorized users, services, and hardware are managed by the organization).
- ISO 27001 (provide full title of the security reference as is in the reference documentation).
- PCI DSS Reference (provide full title of the security reference as is in the reference documentation).
- HIPAA (provide full title of the security reference as is in the reference documentation).
- SOC 2 (provide full title of the security reference as is in the reference documentation).
- OWASP (provide full title of the security reference as is in the reference documentation).
- GDPR (provide full title of the security reference as is in the reference documentation).

Calculate the RISK based on Qualitative Risk Model such as "Ease of exploitation" and "The impact". e.g., "Ease of exploitation""

- E1. Can an attacker exploit this remotely?

- E2. Does the attacker need to be authenticated?

- E3. Can the exploit be automated?

"The impact" calculated from "damage potential" and "number of components that are affected by a threat" e.g., "damage potential":

- D1. Can an attacker completely take over and manipulate the system?

- D2. Can an attacker gain administration access to the system?

- D3. Can an attacker crash the system?

- D4. Can the attacker obtain access to sensitive information such as secrets or PII?

"Number of components that are affected by a threat" e.g.,

- N1. How many connected data sources and systems can be impacted?

- N2. How many layers into infrastructure components can the threat agent traverse?

Now, perform the STRIDE Threat Modeling exercise for following user story:

{User_Story}

4.3 Demonstration Case: OWASP Juice Shop

To illustrate the methodology, we applied it to the OWASP Juice Shop, a deliberately insecure web application widely used for security training and research (OWASP (n.d.-a)). Its architecture and vulnerability landscape make it an ideal candidate for STRIDE-based analysis.

In threat modeling practice, a diagram is often drawn by a security expert to capture assets, actors, components, and trust boundaries. From a SDLC perspective, however, reference architectures are also frequently attached to user stories as part of design discussions. These diagrams are produced either by solution architects or as outputs of exploratory research tasks, serving as guidance for developers during implementation.

For this study, we used the official reference architecture of OWASP Juice Shop as a representative diagram (Figure 2). It illustrates the frontend (Angular + Material), backend (Node.js/Express with Sequelize and NoSQL), and supporting file system components (SQLite, content folders), including integration with external identity providers such as Google via OAuth 2.0.

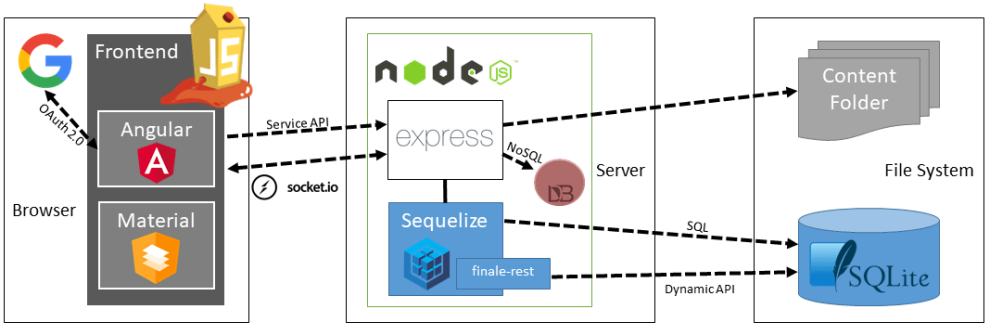

Figure 1: Reference architecture of the OWASP Juice Shop application OWASP (n.d.-c)

Following user story was generated using ChatGPT-5 model from https://chatgpt.com feature called “Add photos & files” by uploading Figure 1, and is used in this research:

> *“As a user, I want to log in with my Google account (OAuth 2.0) via the Angular frontend so that I can securely access application features served by the Node.js/Express backend, which stores data in SQLite/NoSQL and serves files from the content folder.”*

The AI-driven engine was tasked with producing STRIDE-based threats, mitigation subtasks, and compliance mappings. Results are presented in Section 5.

4.4 Models Evaluated

For this conceptual study, we evaluated outputs generated by two advanced large language models: ChatGPT-5 (OpenAI, via ChatGPT plus subscription: https://chatgpt.com) and Gemini-Pro (Google DeepMind, via AI Studio: https://aistudio.google.com/). Both represent cutting-edge generative AI systems optimized for reasoning, structured task generation, and security-relevant content creation.

4.4.1 GPT-5 (OpenAI)

GPT-5 was selected as the primary model for its reliable structured outputs (tabular, JSON, compliance-aligned), extended context window for processing detailed user stories with traceability, and practical availability through OpenAI's API and enterprise tools.

4.4.2 Gemini-Pro (Google DeepMind)

Gemini-Pro was used to test robustness and cross-model consistency. It generated STRIDE-based subtasks mapped to major frameworks (NIST CSF, ISO 27001, PCI DSS, HIPAA, SOC 2, OWASP, GDPR), delivered outputs in Markdown and JSON for reproducibility, and showed strength in cross-mapping ambiguous cases. Accessible via Google AI Studio, it supports research and academic validation.

4.5 System Architecture and Workflow Integration

To evaluate workflow integration (RQ3), we designed a system architecture that processes user stories and generates candidate security tasks (Figure 2).

The workflow begins with a user story submitted by a developer. The AI-driven threat modeling engine analyzes the story, references established security standards and generates candidate security tasks. These tasks flow into a human-in-the-loop review stage:

- If approved, tasks are added to the backlog with preserved traceability and compliance mappings.
- If rejected, they are iteratively refined by the engine until they meet acceptance criteria.

This architecture ensures outputs are technically sound, traceable, and standards-aligned, while remaining directly compatible with agile project management environments such as Jira, GitHub Issues, and Azure DevOps.

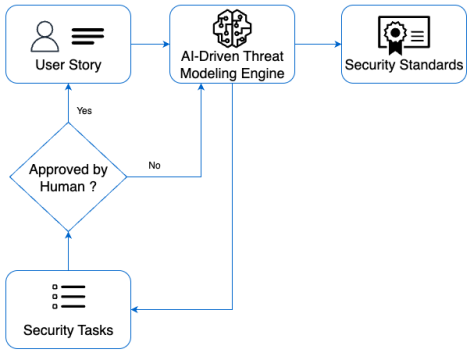

Figure 2: System architecture of the AI-driven threat modeling engine

The architecture consists of four main components:

- **User Story Input:** Represents developer-authored requirements serving as the entry point.
- **AI-Driven Threat Modeling Engine:** Automatically analyzes user stories, classifies threats via STRIDE, and maps results to compliance frameworks.
- **Security Standards:** Provide security standard references (e.g., NIST CSF, ISO 27001, PCI DSS etc.) to align subtasks with regulatory expectations when required, while also supporting ingestion of organization-specific policies in native formats (e.g., PDF) for LLM-based parsing and application.
- **Human-in-the-Loop Review:** Ensures governance by allowing experts to approve, refine, or reject tasks before integration into development backlogs.

4.6 Security Considerations needed in AI Integration

Integrating AI with development artifacts like user stories risks information disclosure, as project systems may contain sensitive logic, design details, or credentials. To mitigate this, the design incorporates safeguards:

- Deploying the LLM on-premise or in a private cloud, or integrating with a company-managed AI service under established governance controls, to prevent uncontrolled data exposure.
- Applying personal and sensitive data anonymization to mask identifiers such as names, email addresses, access tokens, or system details before processing.
- Using encryption and audit logging within the Jira plugin to secure records of all interactions.
- Conducting compliance checks aligned with GDPR and SOC 2 principles to ensure lawful data processing.

Prior research (Ferrag et al., 2024; Tete, 2024) highlights privacy leakage as a key vulnerability. Embedding these safeguards ensures automation improves security without creating new exposure risks.

5. Results

5.1 RQ1 - Threat Modeling Capability

The OWASP Juice Shop login story was evaluated using ChatGPT-5 (14 subtasks, IDs 1–14, see Figure 3) and Gemini-Pro (10 subtasks, IDs 101–110). Both models generated structured threats across all six STRIDE categories, demonstrating that LLMs can approximate expert-level threat modeling.

ID	Issue Key	Security Task ID	STRIDE Category	Assets	Threat Description	Risk (E1–E3, D1–D4, N1–N2)	Security Task Title	User Story Acceptance Criteria (Mitigation)
1	US-001	1	Spoofing	OAuth 2.0 Authorization Code Flow	Attacker intercepts/replays OAuth codes/tokens to impersonate the user during Google login.	High (E1=Yes, E2=No, E3=Yes; D1=Yes, D2=Yes, D3=No, D4=Yes; N1=5, N2=4)	Enforce PKCE, State & TLS	Enforce TLS 1.2+ with HSTS; Require PKCE; Validate state/nonce; Short-lived & rotating tokens
2	US-001	2	Tampering	Service API (Node.js/Express)	Client manipulates request parameters/IDs to bypass authorization (BOLA/BFLA).	High (E1=Yes, E2=No, E3=Yes; D1=Yes, D2=Yes, D3=Yes, D4=Yes; N1=5, N2=4)	Object-Level Authorization (BOLA)	Server-side ownership checks; Default deny RBAC/ABAC; Validate JWT scope; Strict schema validation
3	US-001	3	Elevation of Privilege	Sequelize ORM + SQL/NoSQL Query Layer	Injection via unsafe queries grants admin-level operations.	High (E1=Yes, E2=No, E3=Yes; D1=Yes, D2=Yes, D3=No, D4=Yes; N1=4, N2=3)	Eliminate Injection & Enforce Least Privilege	Parameterized queries only; Disable/review raw queries; Least-privilege DB roles; Automated SAST/DAST
4	US-001	4	Information Disclosure	Token Storage (Browser/App)	ID/Access tokens stored in local/session storage leaked via XSS or logs.	High (E1=Yes, E2=No, E3=Yes; D1=No, D2=No, D3=No, D4=Yes; N1=4, N2=3)	Harden Token Storage	HttpOnly/Secure/SameSite cookies; Don't log tokens; Rotate & revoke tokens
5	US-001	5	Spoofing	OAuth Redirect URI & Client Registration	Manipulated redirect_uri or rogue client_id causes token leakage to attacker.	High (E1=Yes, E2=No, E3=Yes; D1=Yes, D2=No, D3=No, D4=Yes; N1=4, N2=3)	Strict OAuth Client & Redirect URI Policy	Register exact URIs; Enforce HTTPS; Validate iss/aud/azp claims
6	US-001	6	Tampering	JWT Verification Pipeline	Attacker tampers with alg/claims or uses unsigned tokens.	High (E1=Yes, E2=No, E3=Yes; D1=Yes, D2=Yes, D3=No, D4=Yes; N1=4, N2=3)	Robust JWT Verification	Verify signature; Enforce alg allow-list; Check exp/nbf/aud/iss; Reject "none"
7	US-001	7	Information Disclosure	Content Folder (Static Files)	Path traversal or direct URL access exposes sensitive files.	High (E1=Yes, E2=No, E3=Yes; D1=No, D2=No, D3=No, D4=Yes; N1=4,	Secure File Serving	Serve via API only; Allow-list paths; Enforce RBAC

Figure 3: STRIDE-based subtasks generated for OWASP Juice Shop user story by ChatGPT-5 (partial view)

5.1.1 Overlaps

Overlapping findings captured the essential STRIDE threats. Each ID (e.g., ChatGPT 1) refers to the corresponding Security_Task_ID field in the model outputs, full output files can be found in Appendix.

- **Spoofing:** OAuth replay and state validation (Gemini 101; ChatGPT 1, 5).
- **Tampering:** Token theft and unsafe API modification (Gemini 104, 108; ChatGPT 2, 6).
- **Elevation of Privilege:** Backend authorization weaknesses (Gemini 102; ChatGPT 3).
- **Information Disclosure:** Token storage and database leakage (Gemini 103, 109; ChatGPT 4, 7, 10, 12).
- **Denial of Service:** Flooding and database exhaustion (Gemini 106, 110; ChatGPT 8, 14).
- **Repudiation:** Missing audit logs (Gemini 105; ChatGPT 9).

5.1.2 Unique findings

Distinct outputs from each model highlight complementary strengths, expanding overall coverage beyond the overlaps:

- **ChatGPT-5:** CSRF protection (11), secrets management (12), session fixation (13), CORS misconfiguration (10).
- **Gemini-Pro:** CSP/HSTS (107), malicious file uploads (108), environment/config security (109), database query overload (110).

Six subtasks overlapped directly, while each model contributed four to five unique findings. This yields about 20 distinct subtasks rather than 24. ChatGPT-5 provided granular developer-focused mitigations, whereas Gemini-Pro emphasized compliance and operational safeguards. Combined, the two models produced a balanced and comprehensive set of security tasks aligned with the STRIDE methodology.

5.2 RQ2 - Compliance Alignment and Sustainability

The OWASP Juice Shop login story evaluation generated 14 subtasks from ChatGPT-5 (IDs 1–14, see Figure 4) and 10 subtasks from Gemini-Pro (IDs 101–110). Each subtask was mapped to STRIDE categories, OWASP Top 10 categories, and major compliance frameworks (NIST CSF, ISO 27001, PCI DSS, HIPAA, SOC 2, OWASP ASVS, GDPR). Together, the two models demonstrated strong alignment with security and compliance standards, though some coverage gaps remain.

ID	NIST CSF Reference	ISO 27001 Reference	PCI DSS Reference	HIPAA Reference	SOC 2 Reference	OWASP Reference	GDPR Reference
1	PR.AA-01: Identities and credentials for authorized users, services, and hardware are managed	A.9.2.4 Management of secret authentication information of users	Req. 8: Identify users and authenticate access to system components	§164.312(d) Person or entity authentication	CC6.2 Logical access security measures	ASVS V2.1 Authentication design; OWASP API Security: API2 Broken Authentication	Art. 32 Security of processing
2	PR.AA-04: Access to data, assets, and devices is limited to authorized users, processes, and devices	A.9.1.2 Access to networks and services controlled	Req. 7: Restrict access to cardholder data by business need to know	§164.312(a)(1) Access control	CC6.4 Logical access restricted	OWASP API Security: API1 Broken Object Level Authorization	Art. 25 Data protection by design and by default
3	PR.DS-06: Mechanisms for data integrity are implemented and tested	A.12.6.1 Technical vulnerability management; A.9.2.3 Privileged access rights	Req. 6.2/6.3 Secure software and applications; Req. 6.5 Common coding vulnerabilities	§164.312(c)(1) Integrity	CC6.6 Controls protect against injection	OWASP Top 10 A03:2021 Injection	Art. 32 Security of processing
4	PR.DS-05: Access permissions and authorizations are managed, incorporating least privilege and separation of duties	A.10.1.1 Cryptographic controls; A.9.4.1 Information access restriction	Req. 3.4 Protect stored account data; Req. 8.3 Strong authentication	§164.312(e)(1) Transmission security; §164.312(a)(2)(iv) Encryption	CC6.1 Policies and procedures constrain logical access	ASVS V3.4 Sensitive data storage; Token Binding Cheat Sheet	Art. 32 Integrity and confidentiality
5	PR.AA-01: Identities and credentials are managed	A.9.2.1 User registration and de-registration	Req. 8.2 Secure protocols for authentication	§164.312(d) Authentication	CC6.2 Logical access security measures	ASVS V2.2 OAuth/OIDC verification	Art. 32 Security of processing
6	PR.DS-06: Integrity checking mechanisms	A.14.1.2 Securing application services on public networks	Req. 6.5.9 Verify authenticity of sessions/tokens	§164.312(c)(1) Integrity	CC7.2 Monitoring and anomaly detection	ASVS V3.2 Token integrity; JWT Cheat Sheet	Art. 32 Integrity
7	PR.DS-05: Access permissions managed	A.9.4.1 Information access restriction	Req. 7.2 Protect public access to sensitive resources	§164.312(a)(1) Access control	CC6.4 Restrict logical access	OWASP Top 10 A01:2021 Broken Access Control; A05:2021 Security Misconfig	Art. 5(1)(c) Data minimization; Art. 32 Confidentiality

Figure 4: Compliance Alignment Mappings by ChatGPT-5 (partial view)

5.2.1 OWASP category coverage

Both models consistently addressed high-priority OWASP risks, cross-checked against the known vulnerability categories of the OWASP Juice Shop application (OWASP, n.d.-d) for better traceability, while each contributed unique coverage areas:

- Broken Authentication: OAuth/token replay, redirect-URI validation, and session management (ChatGPT 1, 5, 13; Gemini 101, 102).
- Broken Access Control (BOLA): Object-level authorization and file/path controls (ChatGPT 2, 7; Gemini 104).
- Injection: Unsafe ORM and query handling (ChatGPT 3; Gemini 103).
- Sensitive Data Exposure / XSS: Token leakage, client storage, CSP/HSTS, and environment protections (ChatGPT 4; Gemini 107, 109).
- Security Misconfiguration: Static file serving, JWT/CORS defaults, and config protection (ChatGPT 7, 10; Gemini 109).
- Denial of Service / Resource Controls: Flooding, upload quotas, and DB query overload (ChatGPT 8, 14; Gemini 106, 110).
- Logging & Monitoring / Repudiation: Audit logging and log retention (ChatGPT 9; Gemini 105).
- CSRF / Session Management: Explicit mitigations surfaced by ChatGPT (10, 11).
- File Handling Security: Malicious upload checks introduced by Gemini (108).

5.2.2 Gaps

Neither model generated explicit coverage for Insecure Deserialization, Vulnerable/Outdated Components (dependency or SBOM management), or XML External Entities (XXE).

5.2.3 Compliance framework alignment

- **NIST CSF (USA):** The generated subtasks mapped directly to key NIST Cybersecurity Framework categories—PR.AC (Access Control), PR.DS (Data Security), DE.CM (Detection and Monitoring), PR.IP (Protective Technology), and PR.PT (Protective Measures). By translating abstract framework principles into actionable user-story tasks, this research addresses the long-standing gap between high-level policy frameworks and day-to-day developer adoption. It effectively operationalizes "secure-by-design" practices at the earliest stage of the SDLC, ensuring security is built in rather than bolted on. Such integration is directly aligned with U.S. national cybersecurity priorities, including supply-chain protection and small-to-medium business adoption, thereby reinforcing the broader national interest in embedding NIST CSF as a baseline across industries.
- **PCI DSS:** Tasks addressing token protection, session handling, and audit logging map to PCI DSS requirements for data protection, authentication, and monitoring.
- **GDPR (EU):** Encryption, pseudonymization, logging, and retention subtasks provide safeguards consistent with GDPR Article 32.
- **ISO 27001 / HIPAA / SOC 2:** Gemini-Pro in particular emphasized alignment with ISO control domains (A.9, A.12, A.14, A.17) and HIPAA/SOC 2 obligations, strengthening governance aspects.

Table 2: Summary Table

OWASP Category	ChatGPT-5 Security Task IDs	Gemini-Pro Security Task IDs	Compliance Frameworks
Broken Authentication	1, 5, 13	101, 102	NIST CSF PR.AC; PCI DSS 8; ISO 27001 A.9
Broken Access Control (BOLA)	2, 7	104	NIST CSF PR.AC; SOC 2 CC6
Injection	3	103	PCI DSS 6.5; OWASP ASVS 5; ISO 27001 A.14
Sensitive Data Exposure / XSS	4	107, 109	GDPR Art.32; HIPAA §164.312; NIST CSF PR.DS
Security Misconfiguration	7, 10	109	NIST CSF PR.IP; ISO 27001 A.12; SOC 2 CC7
Denial of Service / Resource Limits	8, 14	106, 110	NIST CSF DE.CM; ISO 27001 A.17; SOC 2 CC7
Logging & Monitoring / Repudiation	9	105	NIST CSF DE.AE; PCI DSS 10; ISO 27001 A.12
CSRF / Session Management / CORS	10, 11	-	OWASP ASVS 4.2; NIST CSF PR.AC
File Handling Security	-	108	ISO 27001 A.12; NIST CSF PR.IP

The combined results show that ChatGPT-5 emphasized developer-focused mitigations (CSRF, session fixation, secrets management, CORS) while Gemini-Pro emphasized compliance-linked and operational tasks (CSP/HSTS, file upload risks, environment configs, DB query overload). Together, they produced stronger compliance alignment than either model individually. While some OWASP categories remain uncovered, the ability of both LLMs to map user-story–derived subtasks to STRIDE, OWASP, and compliance frameworks demonstrates a meaningful step toward automated compliance-aware threat modeling.

5.3 RQ3- Workflow Integration and Adoption

The system is designed for direct adoption into DevSecOps workflows by producing machine-readable outputs in a Jira-compatible JSON format. Each generated security task is represented as a discrete JSON object that links to the originating user story and includes identifiers, STRIDE category, asset, threat description, qualitative risk dimensions, mitigation tasks, acceptance criteria, and compliance mappings. This enables seamless import into

platforms such as Jira, GitHub Issues, or Azure DevOps, where security tasks can be managed alongside functional requirements.

Both models, ChatGPT-5 and Gemini-Pro, generate JSON outputs with minor format differences. ChatGPT-5 represents risk as a structured object with subfields (E1–E3, D1–D4, N1–N2) and uses human-readable keys (e.g., "Threat Description"), while Gemini-Pro encodes risk as a compact string and employs snake_case keys (e.g., "Threat_Description", "STRIDE_Category"). Despite these schema variations, both preserve the same essential content: traceability from threat to mitigation and alignment with compliance frameworks (NIST CSF, ISO 27001, PCI DSS, HIPAA, SOC 2, OWASP, GDPR)

Figure 5 illustrates an example export (Security_Task_ID 3 from ChatGPT-5), where an injection threat is captured as a JSON object with structured acceptance criteria and mapped standards. Equivalent artifacts from Gemini-Pro confirm that the format is reproducible across models, reinforcing interoperability.


```
{
  "ID": 3,
  "Issue_key": "US-001",
  "Security_Task_ID": 3,
  "STRIDE Category": "Elevation of Privilege",
  "Assets": "Sequelize ORM + SQL/NoSQL Query Layer",
  "Threat Description": "Injection via unsafe queries grants admin-level operations.",
  "Risk": {
    "Overall": "High",
    "E1": "Yes",
    "E2": "No",
    "E3": "Yes",
    "D1": "Yes",
    "D2": "Yes",
    "D3": "No",
    "D4": "Yes",
    "N1": 4,
    "N2": 3
  },
  "Security Task Title": "Eliminate Injection & Enforce Least Privilege",
  "User Story Acceptance Criteria as Mitigation": [
    "Use parameterized queries/ORM bind parameters only",
    "Disable raw queries or review/approve tightly",
    "DB accounts least-privilege with separated roles",
    "Automated SAST/DAST for injection patterns"
  ],
  "NIST CSF Reference": "PR.DS-06: Mechanisms for data integrity are implemented and tested.",
  "ISO 27001": "A.12.6.1 Technical vulnerability management; A.9.2.3 Management of privileged access rights",
  "PCI DSS Reference": "Req. 6.2/6.3: Secure software and applications; Req. 6.5: Address common coding vulnerabilities",
  "HIPAA": "\u00a7164.312(c)(1) Integrity",
  "SOC 2": "CC6.6 Controls protect against injection and other attacks",
  "OWASP": "OWASP Top 10 A03:2021 Injection",
  "GDPR": "Art. 32 Security of processing"
},
```


Figure 5: AI output in JSON format for security Task ID 3 generated by ChatGPT-5.

Key fields enabling workflow integration include:

- Issue Key / User Story Linkage – ensures context with development tasks.
- Security Task ID – unique identifier for tracking.
- STRIDE Category & Threat Description – structured threat modeling reference.
- Risk (E1–E3, D1–D4, N1–N2) – explicit qualitative assessment.
- Security Task Title & Acceptance Criteria – actionable, testable controls.
- Compliance References – mapping to standards for audit readiness.

This export format provides a standards-annotated, automation-ready schema that ensures traceability and reduces manual transcription. Human review remains the final governance step, but the JSON representation enables scalable integration of AI-generated security tasks into enterprise pipelines.

6. Discussion

The demonstration shows that an LLM-driven threat modeling engine can be used to extract STRIDE-relevant threats directly from plain-language user stories and translate them into actionable, standards-mapped security subtasks. In the OWASP Juice Shop login example the LLM Models produced several distinct subtasks covering all STRIDE categories; each subtask included an affected asset, a concise threat description (written so non-experts can understand why it matters), a qualitative risk profile, acceptance criteria, and mapped controls. The JSON export format preserved traceability to the parent story and supplied the fields required for programmatic ingestion into issue trackers.

Strengths of the approach include: (1) shifting threat analysis left into the requirements phase, enabling earlier remediation decisions; (2) producing developer-friendly, actionable subtasks (titles + acceptance criteria) that reduce ambiguity; and (3) coupling threats with framework references (NIST CSF, ISO 27001, PCI DSS, HIPAA, SOC 2, OWASP, GDPR) to support compliance-aware engineering. The human-in-the-loop review remains essential for governance - to refine mappings, reject false positives, and adapt outputs to project context.

Important challenges surfaced during evaluation: certain STRIDE classes (notably repudiation and denial-of-service) are context-dependent and harder for a text-only model to detect reliably; exact control-ID fidelity sometimes requires manual refinement (parser/mapping improvements); and the demonstration was a

simulated integration (JSON export) rather than a live plugin, so operational behaviors such as CI/CD integration, roles, permissions, and end-to-end latency remain to be validated at scale. Overall, the results support feasibility and practical value while highlighting concrete engineering improvements needed for production use.

7. Limitations and Future Work

This work was constrained by its reliance on Jira-formatted user stories, a single demonstration case, and general-purpose LLMs that were not security-tuned. SME input was limited, and integration was shown only through JSON exports rather than production-grade CI/CD testing.

Future efforts should broaden datasets across industries and user-story types (e.g., complex stories with attachments, bug reports, research tasks), fine-tune LLM models, and expand SME validation. Additional directions include handling non-text artefacts such as UML diagrams and embedding the system into live CI/CD pipelines to assess practical scalability.

8. Conclusion

This paper presents a lightweight, LLM-powered approach to automated threat modeling that operates at the user-story stage of the SDLC. The engine converts plain-language user stories into STRIDE-classified threats, actionable security subtasks with acceptance criteria, and mapped compliance references; outputs are emitted as machine-readable, Jira-compatible JSON to preserve traceability and enable programmatic ingestion. The OWASP Juice Shop demonstration produced a comprehensive set of subtasks that aligned with known vulnerability categories and illustrated practical workflow compatibility. While further work is needed to improve coverage (some threat types and third-party/component risks) and to validate live operational integration, the approach shows promise for embedding standards-aware security earlier in development, reducing rework, and improving the auditability of security activities. All artifacts for reproducibility (datasets, annotations, JSON exports) are available in the project repository (see Appendix A).

Ethics Declaration: This research did not involve human participants, personal data collection, or sensitive experiments; therefore, formal ethical clearance was not required.

AI Declaration: AI tools were used extensively in this work to improve readability, grammar, and overall structure. Their role was limited to editorial support, and they did not contribute to the design of the study, data analysis, or interpretation of results. All substantive content, findings, and conclusions are entirely the author's own, and all AI-generated content was carefully reviewed and validated by the author.

References

Aboukadri, S. et al. (2025) 'Leveraging RAG and LLMs for access-control policy extraction from user stories in Agile software development', *IEEE Access*. Available at: https://www.researchgate.net/publication/393424984 (Accessed: 20 September 2025).

Casillo, F. et al. (2022) 'Detecting privacy requirements from user stories with NLP', *Information and Software Technology*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0950584922000246 (Accessed: 20 September 2025).

Dalpiaz, F. (2025) 'Automated analysis of user story requirements', in *Natural Language Processing for Requirements Engineering*. Springer. Available at: https://link.springer.com/chapter/10.1007/978-3-031-73143-3_12 (Accessed: 20 September 2025).

Elsharef, I. et al. (2024) 'Facilitating threat modeling by leveraging large language models', *AISCC@NDSS 2024*. Available at: https://www.ndss-symposium.org/wp-content/uploads/aiscc2024-16-paper.pdf (Accessed: 20 September 2025).

Ferrag, M.A., Maglaras, L., Shu, L. and Derhab, A. (2024) 'Generative AI in cybersecurity: A comprehensive review of LLM applications and vulnerabilities', *arXiv preprint* arXiv:2405.12750. Available at: https://arxiv.org/abs/2405.12750 (Accessed: 20 September 2025).

Jedrzejewski, F.V. et al. (2024) 'Threat modeling of ML-intensive systems', *ACM Proceedings*. Available at: https://dl.acm.org/doi/10.1145/3644815.3644975 (Accessed: 20 September 2025).

Jedrzejewski, F.V. et al. (2025) 'LLMs' suitability for network security: A STRIDE-based case study', *arXiv preprint*arXiv:2505.04101. Available at: https://arxiv.org/abs/2505.04101 (Accessed: 20 September 2025).

Jedrzejewski, F.V., Fucci, D. and Adamov, O. (2025) 'ThreMoLIA: Threat Modeling of Large Language Model-Integrated Applications', arXiv preprint arXiv:2504.18369. Available at: https://arxiv.org/abs/2504.18369 (Accessed: 21 September 2025).

Koball, C. et al. (2024) 'Machine learning security: Threat model, attacks, and mitigations', *IEEE Computer*. Available at: https://dl.acm.org/doi/abs/10.1109/MC.2024.3396357 (Accessed: 20 September 2025).

Kosenkov, O. et al. (2025) 'Requirements engineering for regulatory compliance: A systematic mapping study', *Information and Software Technology*. Available at: https://www.sciencedirect.com/science/article/pii/S0950584924002271 (Accessed: 20 September 2025).

Mussmann, A. (2021) 'Mapping the state of security standards mappings', *GITO Verlag White Paper*. Available at: https://library.gito.de/wp-content/uploads/2021/08/L4_Mussmann-Mapping_the_State_of_Security_Standards_Mappings-305_c.pdf (Accessed: 20 September 2025).

Nagaraja, N. et al. (2025) 'Cyber threat modeling of an LLM-based healthcare system', *ICISSP/SCITEPRESS*. Available at: https://www.scitepress.org/Papers/2025/132897/132897.pdf (Accessed: 20 September 2025).

NIST (2023) 'Mapping SP 800-53 Rev. 5 to ISO/IEC 27001:2022', *NIST CSRC*. Available at: https://csrc.nist.gov/projects/olir/informative-reference-catalog/details?referenceId=155#/ (Accessed: 20 September 2025).

OWASP (n.d.-a) OWASP Juice Shop Project. Available at: https://owasp.org/www-project-juice-shop/ (Accessed: 21 September 2025).

OWASP (n.d.-b) Threat Modeling Process. Available at: https://owasp.org/www-community/Threat_Modeling_Process (Accessed: 21 September 2025).

OWASP (n.d.-c) OWASP Juice Shop Application Architecture. Available at: https://owasp.org/www-project-juice-shop/ (Accessed: 21 September 2025).

OWASP (n.d.-d) Vulnerability categories. Available at: https://help.owasp-juice.shop/part1/categories.html (Accessed: 21 September 2025).

Parvathinathan, K. et al. (2025) 'AI-powered DevSecOps for continuous security and compliance', *ResearchGate Preprint*. Available at: https://www.researchgate.net/publication/395004862 (Accessed: 20 September 2025).

Prates, L. et al. (2025) 'DevSecOps practices and tools: A multivocal literature review', *International Journal of Information Security*. Available at: https://link.springer.com/article/10.1007/s10207-024-00914-z (Accessed: 20 September 2025).

Robeer, M. et al. (2016) 'Automated extraction of conceptual models from user stories', *IEEE Requirements Engineering Conference*. Available at: https://webspace.science.uu.nl/~dalpi001/papers/robe-luca-werf-dalp-brin-16-re.pdf(Accessed: 20 September 2025).

Security Compass (2023) 'Mapping OWASP ASVS to ISO 27001', *Whitepaper*. Available at: https://www.securitycompass.com/whitepapers/mapping-security-requirements-to-standards-owasp-asvs-to-iso-27001/(Accessed: 20 September 2025).

Tete, S.B. (2024) 'Threat modelling and risk analysis for large language model (LLM)-powered applications', *arXiv preprint* arXiv:2406.11007. Available at: https://arxiv.org/abs/2406.11007 (Accessed: 20 September 2025).

UK DSIT (2024) 'Cyber security risks to artificial intelligence', *UK Government*. Available at: https://www.gov.uk/government/publications/research-on-the-cyber-security-of-ai/cyber-security-risks-to-artificial-intelligence (Accessed: 20 September 2025).

Yang, S., Wu, T., Liu, S., Nguyen, D., Jang, S. and Abuadbba, A. (2024) 'ThreatModeling-LLM: Automating threat modeling using large language models for banking system', arXiv preprint arXiv:2411.17058. Available at: https://arxiv.org/abs/2411.17058 (Accessed: 21 September 2025).

Yigit, Y. et al. (2025) 'Generative AI and LLMs for critical infrastructure protection', *Sensors*. Available at: https://www.mdpi.com/1424-8220/25/6/1666 (Accessed: 20 September 2025).

9. Appendix A. Dataset and Repository

All datasets, , figures, and outputs used in this study are available in the companion GitHub repository: 👉 https://github.com/ShantuApps/Reseach-Automated-Threat-Modeling-using-AI