**Università degli Studi di Padova**

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica


# Analisi di sicurezza e compliance OWASP per AI tramite Gandalf Test

*Tesi di laurea*

*Relatore*
Prof. Marco Zanella

*Laureando*
Carmelo Russello
*Matricola* 2076421

Anno Accademico 2025-2026

"Nami tu sei la mia speranza. In questo mondo dove il potere è tutto, anche se vengo ferita e umiliata, voglio ancora lottare, correre e vivere il più liberamente possibile! Se la penso così è solo grazie a te"

— Ragazzina dai capelli arancioni

Dedicato a ...

# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecentoventi ore, dal laureando Carmelo Russello presso l'azienda Var Group S.p.A. Gli obbiettivi da raggiungere erano molteplici.

- In primo luogo era richiesto lo sviluppo di un sistema di testing automatizzato per l'analisi di sicurezza e compliance OWASP dei modelli di intelligenza artificiale, utilizzando lo strumento Gandalf Test.
- In secondo luogo era richiesta l'implementazione di una dashboard web per la visualizzazione dei risultati dei test eseguiti.

*"Life is really simple, but we insist on making it complicated"*

— Confucius

# Ringraziamenti

*Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Marco Zanella, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.*

*Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.*

*Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.*

*Padova, Aprile 2026* Carmelo Russello

# Indice

# Elenco delle figure



# Elenco delle tabelle

# Capitolo 1

# Introduzione

Il presente capitolo introduce il contesto applicativo oggetto dello stage: automazione di testing su applicazioni che sfruttano l'intelligenza artificiale generativa, come chatbot o assistenti virtuali. Queste applicazioni trattano input testuali e informazioni sensibili, esponendo a rischi generali quali manipolazioni degli input, fuoriuscite di dati e comportamenti imprevisti dei modelli.
Il progetto mira a definire e implementare un approccio integrato per valutare e migliorare la sicurezza, l'affidabilità e la conformità delle applicazioni, attraverso analisi del codice, test e pratiche operative dedicate.

## 1.1 L'azienda

L'azienda con cui ho deciso di intraprendere il mio percorso di Stage è Var Group S.p.A., una realtà italiana specializzata in soluzioni IT e servizi di consulenza per le imprese.
La scelta è stata influenzata dalla mia precedente esperienza con il progetto di Software Engineering svolto presso Var Group, durante il quale ho avuto modo di apprezzare la professionalità e la competenza del team.

## 1.2 L'idea

Il progetto di stage è incentrato sulla sicurezza delle applicazioni basate su AI generativa, con particolare attenzione all'impiego di analisi statica e dinamica del codice per verificarne la conformità alle linee guida OWASP.
L'obiettivo è sviluppare un sistema di testing e verifica capace di rilevare e prevenire vulnerabilità specifiche (come le tecniche di prompt injection) evitando la divulgazione di informazioni riservate a utenti non autorizzati e impedendo l'introduzione di falle nell'applicativo.
Il lavoro comprende inoltre la progettazione e l'integrazione di workflow e strumenti automatizzati per l'analisi del flusso dei dati, il fuzzing e i penetration test, nonché la definizione di procedure di mitigazione e di reporting delle vulnerabilità individuate.

## 1.3 Organizzazione del testo

**Il secondo capitolo** descrive ...

**Il terzo capitolo** descrive lo stage e le attività svolte.

**Il quarto capitolo** ...

**Il quinto capitolo** ...

**Il sesto capitolo** ...

**Nel settimo capitolo** ...

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

# Capitolo 2

# Processi e metodologie

*Brevissima introduzione al capitolo*

## 2.1 Processo sviluppo prodotto

# Capitolo 3

# Descrizione dello stage

*Breve introduzione al capitolo*

## 3.1 Introduzione al progetto

## 3.2 Analisi preventiva dei rischi

Durante la fase iniziale di analisi sono stati identificati i principali rischi potenziali connessi al progetto, classificati per ambito (tecnico, di progetto e infrastrutturale) e priorizzati in base all'impatto e alla probabilità. Per ciascun rischio è stato predisposto un piano di mitigazione che definisce azioni concrete, tempistiche e responsabilità precise.

Le contromisure prevedono attività di sperimentazione controllata degli strumenti, revisioni manuali dei risultati, integrazione e test in ambienti rappresentativi, oltre a piani di escalation per le criticità più gravi. È inoltre prevista una procedura di monitoraggio continuo e revisione periodica delle valutazioni e delle soluzioni adottate, in modo da aggiornare rapidamente le contromisure alla luce di nuovi dati o evoluzioni tecnologiche.

### 3.2.1 Rischi tecnici

**1. Complessità nell'applicare strumenti di security testing ad AI generativa (tool immaturi o non sempre affidabili).**
**Descrizione:** Le difficoltà nell'adattare i tool di security testing all'AI generativa possono derivare dalla loro immaturità o dalla mancanza di affidabilità.
**Soluzione:** Una lunga fase di sperimentazione e testing dei tool mitigherà i rischi, garantendo risultati affidabili.

**2. Possibili falsi positivi o negativi nei test di vulnerabilità.**
**Descrizione:** I test di vulnerabilità potrebbero generare risultati inaccurati, con falsi positivi (segnalazioni errate di vulnerabilità) o falsi negativi (mancata rilevazione di vulnerabilità reali).
**Soluzione:** Implementare una fase di revisione manuale dei risultati dei test per convalidare le segnalazioni e ridurre il rischio di falsi positivi e negativi.

**3. Difficoltà di integrazione dei tool con codice reale e pipeline di sviluppo.**
**Descrizione:** Le difficoltà di integrazione possono derivare da incompatibilità tra i tool di testing e l'infrastruttura esistente, nonché dalla complessità del codice reale su cui si stanno eseguendo i test.
**Soluzione:** Collaborare con gli sviluppatori del codice reale per garantire che i tool di testing siano compatibili con l'infrastruttura esistente e fornire supporto durante l'integrazione.

### 3.2.2 Rischi di progetto

**4. Mancanza di esperienza pregressa su OWASP o sicurezza AI.**
**Descrizione:** La poca familiarità con le best practices di OWASP o con le specificità della sicurezza nell'AI generativa potrebbe rallentare l'avanzamento del progetto.
**Soluzione:** Studio e formazione con risorse adeguate per aumentare la familiarità con OWASP e la sicurezza dell'AI generativa.

**5. Possibile difficoltà a rispettare la pianificazione a causa della curva di apprendimento iniziale.**
**Descrizione:** La curva di apprendimento iniziale per l'utilizzo di nuovi strumenti e tecnologie potrebbe richiedere più tempo del previsto, influenzando la pianificazione del progetto.
**Soluzione:** Pianificare margini di tempo aggiuntivi per la formazione e l'adattamento agli strumenti, nonché monitorare attentamente i progressi.

### 3.2.3 Rischi infrastrutturali

**6. Limitazioni di risorse computazionali nei test di AI.**
**Descrizione:** Le risorse computazionali disponibili per l'esecuzione dei test di AI potrebbero non essere sufficienti, causando rallentamenti o interruzioni nei test.
**Soluzione:** Ottimizzare l'uso delle risorse disponibili e, se necessario, richiedere l'accesso a risorse computazionali aggiuntive.

**7. Problemi di compatibilità con ambienti cloud o di deployment.**
**Descrizione:** Le differenze tra gli ambienti di sviluppo e produzione potrebbero causare problemi di compatibilità, rendendo difficile l'esecuzione dei test in modo uniforme.
**Soluzione:** Testare i tool di testing in ambienti simili a quelli di produzione e documentare eventuali problemi di compatibilità.

## 3.3 Requisiti e obiettivi

### 3.3.1 Obiettivi obbligatori

- Valutazione comparativa degli strumenti di analisi.
- Applicazione pratica dei test su codice reale.
- Prototipo in grado di generare report sulle vulnerabilità AI rispetto a OWASP.
- Documentazione tecnica e presentazione finale.

### 3.3.2 Obiettivi desiderabili

- Dashboard interattiva con visualizzazioni avanzate
- Integrazione del prototipo in pipeline CI/CD esistente.
- Estensione dei test ad altri framework oltre Gandalf Test.
- Raccomandazioni per un framework interno di AI Security by Design.

## 3.4 Pianificazione

La pianificazione del lavoro di progetto è stata suddivisa in fasi settimanali, con obiettivi specifici per ciascuna fase. Di seguito è riportata una panoramica della pianificazione prevista:

| **Settimana** | **Attività** |
|---|---|
| Settimana 1 | Studio preliminare su OWASP e rischi AI, overview di Gandalf Test, setup ambiente di lavoro. |
| Settimana 2 | Analisi comparativa di tool di analisi statica e dinamica (open-source e commerciali). Creazione matrice di valutazione. |
| Settimana 3 | Applicazione degli strumenti a piccoli progetti demo, valutazione dei risultati e raccolta criticità. |
| Settimana 4 | Esecuzione dei primi test su componenti reali del team, documentazione dei risultati, identificazione vulnerabilità. |
| Settimana 5 | Realizzazione di script/report per aggregare risultati, definizione dei KPI di compliance OWASP. |
| Settimana 6 | Sviluppo di dashboard interattiva per monitorare vulnerabilità e andamento dei test. |
| Settimana 7 | Test end-to-end sul prototipo, miglioramento dei tool e dei report. |
| Settimana 8 | Redazione di documentazione tecnica, manuale utente e materiale per la presentazione della tesi. |

**Tabella 3.1:** Pianificazione delle attività di progetto

# Capitolo 4

# Analisi dei requisiti

*Breve introduzione al capitolo*

## 4.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

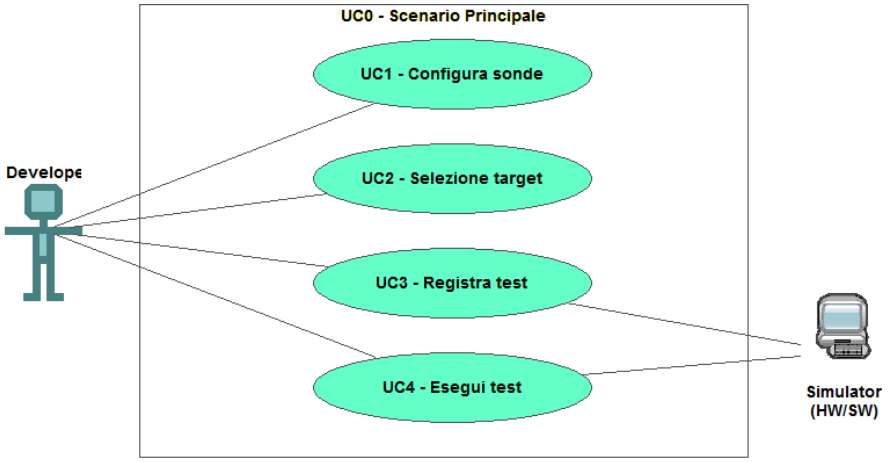


**Figura 4.1:** Use Case - UC0: Scenario principale

### UC0: Scenario principale

**Attori Principali:** Sviluppatore applicativi.

**Precondizioni:** Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

**Descrizione:** La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

**Postcondizioni:** Il sistema è pronto per permettere una nuova interazione.

## 4.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti è così strutturato R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle 4.1, 4.2 e 4.3 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

**Tabella 4.1:** Tabella del tracciamento dei requisti funzionali

| Requisito | Descrizione | Use Case |
|---|---|---|
| RFN-1 | L'interfaccia permette di configurare il tipo di sonde del test | UC1 |

**Tabella 4.2:** Tabella del tracciamento dei requisiti qualitativi

| Requisito | Descrizione | Use Case |
|---|---|---|
| RQD-1 | Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi | - |

**Tabella 4.3:** Tabella del tracciamento dei requisiti di vincolo

| Requisito | Descrizione | Use Case |
|---|---|---|
| RVO-1 | La libreria per l'esecuzione dei test automatici deve essere riutilizzabile | - |

# Capitolo 5

# Progettazione e codifica

*In questo capitolo*

## 5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti presi in considerazione per lo sviluppo del prototipo. L'analisi ha valutato compatibilità, licenza, facilità d'integrazione, costi e capacità di security testing specifiche per modelli generativi. Gli strumenti e le tecnologie esaminate includono:

- **PromptFoo** – piattaforma di testing per LLM con supporto a test automatizzati e integrazione CI/CD.
- **PyRIT** – tool open-source per l'identificazione e mitigazione dei rischi sui modelli generativi.
- **LangFuse** – piattaforma di osservabilità e valutazione di LLM.
- **DeepEval / DeepTeam** – framework di valutazione e tool di red teaming per benchmark e safety testing.
- **Garak** – scanner di vulnerabilità per modelli generativi focalizzato su attacchi pratici (prompt injection, jailbreak, leakage).
- **Giskard** – piattaforma di red teaming automatizzato con opzioni SaaS e libreria Python.
- **Galileo** – piattaforma di osservabilità con SDK, utile per monitoraggio ma meno orientata al security testing.
- **LakeraGuard** – suite di sicurezza commerciale per modelli, include test come il Gandalf test.
- **Tecnologie di base** – Python, Docker, Git/GitHub, CI/CD (GitHub Actions), librerie ML (PyTorch, Hugging Face), e strumenti di logging/monitoring.

Questa panoramica ha guidato la selezione degli strumenti adottati per il prototipo, privilegiando soluzioni open-source facilmente integrabili nell'ambiente di sviluppo e con funzionalità mirate al security testing dei modelli generativi.

### 5.1.1 Strumenti analizzati

Durante le prime due settimane di Stage è stato condotto uno studio preliminare su diversi strumenti di security testing per AI generativa. Di seguito viene fornita la matrice di valutazione dei tools e una breve descrizione di ciascuno strumento analizzato.

| Criterio di valutazione | Categoria | Peso (%) | PromptFoo | PyRIT | LangFuse | DeepEval | Garak | Giskard | Galileo | LakeraGuard |
|---|---|---|---|---|---|---|---|---|---|---|
| Copertura OWASP AI Top 10 | Efficacia Tecnica | 20 | 5 | 4 | 2 | 2 | 3 | 4 | 3 | 4 |
| Velocità di Scansione | Performance | 10 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| Costo Totale (TCO) | Costi | 10 | 4 | 5 | 5 | 5 | 5 | 3 | 1 | 1 |
| Supporto Tecnico & Community | Supporto | 10 | 4 | 3 | 4 | 3 | 3 | 3 | 2 | 4 |
| Adversarial Attack Detection | AI-Specific | 10 | 5 | 5 | 2 | 3 | 3 | 4 | 4 | 5 |
| Efficacia Gandalf Test | Efficacia Tecnica | 8 | 3 | 5 | 1 | 1 | 3 | 3 | 3 | 4 |
| Profondità di Analisi | Efficacia Tecnica | 8 | 4 | 5 | 2 | 2 | 4 | 4 | 3 | 5 |
| Integrazione CI/CD | Usabilità | 8 | 4 | 4 | 3 | 3 | 2 | 4 | 2 | 5 |
| Scalabilità | Performance | 8 | 4 | 4 | 3 | 3 | 4 | 4 | 3 | 4 |
| Model Explainability | AI-Specific | 8 | 4 | 4 | 2 | 2 | 3 | 3 | 4 | 4 |
| TOTALE | | | 4,22 | 4,26 | 2,78 | 2,78 | 3,38 | 3,64 | 2,9 | 3,96 |

**Figura 5.1:** Matrice di Valutazione degli Strumenti Analizzati

#### PromptFoo

PromptFoo è una piattaforma di testing per modelli di linguaggio che consente agli sviluppatori di creare, eseguire e gestire test automatizzati per valutare le prestazioni, l'affidabilità e la sicurezza dei loro modelli di linguaggio naturale. Offre funzionalità come la creazione di casi di test personalizzati, l'integrazione con pipeline CI/CD e report dettagliati sui risultati dei test. Questo tool è stato tenuto in molta considerazione in quanto offre funzionalità specifiche per il testing delle vulnerabilità OWASP top 10 per AI generativa, obbiettivo principale del progetto di stage.

#### PyRIT

PyRIT è uno tool open-source progettato da Azure Microsoft per l'identificazione e la mitigazione dei rischi associati all'uso di modelli di intelligenza artificiale generativa. PyRIT è stato creato per valutare modelli di AI per potenziali vulnerabilità di sicurezza, bias e problemi di conformità, fornendo raccomandazioni su come migliorare la sicurezza e l'affidabilità dei modelli.

#### LangFuse

LangFuse è una piattaforma open-source per la valutazione e l'osservazione delle applicazioni che utilizzano intelligenza artificiale generativa. La piattaforma è integrata con diversi modelli di linguaggio naturale e permette l'utilizzo tramite cloud o in self host. Langfuse offre inoltre un LLM playground per testatre i modelli che aveva catturato la mia attenzione all'inizio del progetto. Tuttavia il tool non offre una vera e propria funzionalità di security testing, focus del progetto di stage.

#### DeepEval / DeepTeam

DeepEval è un framework open-source per la valutazione e il benchmarking dei modelli di intelligenza artificiale generativa. DeepEval non fornisce funzionalità specifiche per

il testing dei modelli. Per ovviare a questa mancanza è stato creato un tool di testing basato su DeepEval chiamato DeepTeam, un tool di red teaming. Tuttavia, DeepTeam non ha un focus sulle vulnerabilità OWASP in quanto si concentra sul testing delle safety guidelines.

#### Garak

Garak è uno scanner di vulnerabilità per modelli di intelligenza artificiale generativa open-source creato da NVIDIA per facilitare il testing dei modelli. Il focus di Garak sta nei metodi di attacco specifici per far fallire in modo imprevisto una LLM o un sistema di dialogo. Garak testa diverse vulnerabilità tra cui allucinazioni, data leakage, prompt injection, jailbreaks ecc.

#### Giskard

Giskard è una piattaforma di red teaming automatizzato per testare, valutare ed analizzare modelli di intelligenza artificiale generativa. Esistono due metodi di utilizzo di Giskard: Come servizio HUB a pagamento o come libreria Python open-source da installare localmente. La libreria fornita è però fortemente limitata nelle funzionalità rispetto al servizio HUB in quanto incentrata sulla ricerca.

#### Galileo

Galileo è una piattaforma che permette la valutazione e osservazione di applicazioni basate su AI generativa. Galileo offre SDK in Python e TypeScript per integrarlo direttamente nei propri progetti. Galileo è molto flessibile per quanto riguarda il deploy e viene utilizzato da molte aziende di rilievo. Nonostante ciò la piattaforma non offre funzionalità specifiche per il security testing, focus di questo progetto, ed è a pagamento.

#### LakeraGuard

LakeraGuard è una piattaforma di sicurezza per modelli di intelligenza artificiale che aiuta gli sviluppatori a proteggere i loro modelli da minacce e vulnerabilità. Offre funzionalità come la scansione delle vulnerabilità, la gestione delle patch e il monitoraggio delle minacce in tempo reale.
Essendo l'azienda svizzera Lakera la creatrice del gandalf test, focus principale delle prime settimane del progetto di stage, il tool da loro creato è stato uno dei primi ad essere analizzato. Tuttavia il tool out of the box fa già quello che lo stage chiede di implementare quindi non ha suscitato uno studio approfondito in quanto avrebbe reso superfluo lo sviluppo del prototipo.
Inoltre è un tool a pagamento che offre un tier gratuito il quale è però limitato a 10000 richieste Application Program Interface (API) al mese, limite che avrebbe reso difficile l'utilizzo futuro del tool.

### 5.1.2 Strumenti scelti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati per lo sviluppo del prototipo.

## 5.2 Ciclo di vita del software

## 5.3 Progettazione

**Namespace 1**

Descrizione namespace 1.

**Classe 1:** Descrizione classe 1

**Classe 2:** Descrizione classe 2

## 5.4 Design Pattern utilizzati

## 5.5 Codifica

Capitolo 6

# Verifica e validazione

# Capitolo 7

# Conclusioni

## 7.1 Consuntivo finale

## 7.2 Raggiungimento degli obiettivi

## 7.3 Conoscenze acquisite

## 7.4 Valutazione personale

# Appendice A

# Appendice A

Citazione

Autore della citazione

# Acronimi e abbreviazioni

**API** Application Program Interface. 15, 25

**UML** Unified Modeling Language. 9, 25

# Glossario

**API** in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 23

**UML** in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 23

# Bibliografia

## Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

## Siti web consultati

*Manifesto Agile*. URL: `http://agilemanifesto.org/iso/it/`.