ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

PROMPTSHIELD: POLICY-AWARE DATA LOSS PREVENTION FRAMEWORK FOR GENERATIVE AI PROMPTS

M.Sc. THESIS

Ezgi KELEŞ

Department of Computer Engineering

Computer Engineering Programme

JANUARY 2026

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

PROMPTSHIELD: POLICY-AWARE DATA LOSS PREVENTION FRAMEWORK FOR GENERATIVE AI PROMPTS

M.Sc. THESIS

Ezgi KELEŞ
(504201572)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Prof. Dr. Şerif BAHTİYAR

JANUARY 2026

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

PROMPTSHIELD: POLİTİKA FARKINDALIKLI ÜRETKEN YAPAY ZEKÂ İSTEMLERI İÇİN VERİ KAYBI ÖNLEME ÇERÇEVESİ

YÜKSEK LİSANS TEZİ

Ezgi KELEŞ
(504201572)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Şerif BAHTİYAR

OCAK 2026

Ezgi KELEŞ, a M.Sc. student of ITU Graduate School student ID 504201572 successfully defended the thesis entitled "PROMPTSHIELD: POLICY-AWARE DATA LOSS PREVENTION FRAMEWORK FOR GENERATIVE AI PROMPTS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Şerif BAHTİYAR** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Dr. Mehmet Tahir SANDIKKAYA** Istanbul Technical University	.....
	Prof. Dr. Muhammed Ali AYDIN Istanbul University Cerrahpaşa	.....

Date of Submission : **15 December 2025**
Date of Defense : **08 January 2026**

To my spouse and my family,

FOREWORD

To begin, I would like to express my sincere gratitude to my thesis advisor, Professor Dr. Şerif Bahtiyar, for his valuable advices, guidance and belief in me throughout the course of this research project.

Second, I would like to express the utmost gratitude to my husband for his continuous encouragement and support through all of the time spent on this research project. He provided the means for me to fulfill my obligations as an academic, a professional, and as a person by allowing me to continue working on this thesis while supporting me in every way.

Lastly, I would like to thank my family for their unconditional encouragement and support. Their love and trust have always been a source of strength and motivation to inspire me to accomplish my dreams and achieve my goals. I would also like to take this opportunity to thank my friends who were instrumental in motivating me and being present when I needed them most to help me maintain focus and positivity during difficult times.

JANUARY 2026

Ezgi KELEŞ
(Computer Engineer)

TABLE OF CONTENTS

	Page
FOREWORD	**ix**
TABLE OF CONTENTS	**xi**
ABBREVIATIONS	**xiii**
SYMBOLS	**xv**
LIST OF TABLES	**xvii**
LIST OF FIGURES	**xix**
SUMMARY	**xxi**
ÖZET	**xxv**
1. INTRODUCTION	**1**
1.1 Background and Motivation	2
1.2 Problem Statement	3
1.3 Purpose of Thesis	3
1.4 Research Questions and Objectives	4
1.5 Research Methodology Overview	5
1.6 Threat Model and Assumptions	5
1.7 Scope and Limitations	6
1.8 Contributions	6
1.9 Thesis Organization	7
2. LITERATURE REVIEW	**9**
2.1 Theoretical Foundations and Conceptual Frameworks	9
2.1.1 Data loss prevention (DLP): definitions, states, and mechanisms	9
2.1.2 From theory to applications generative artificial intelligence (GenAI)	11
2.1.3 Information security policy: development and governance	11
2.2 The Evolution of Data Loss Prevention Systems	12
2.2.1 Traditional approaches: the era of static inspection	12
2.2.2 Transition to context-aware and usage control models	13
2.2.3 Kernel and hypervisor-level protections	14
2.2.4 Hybrid and multi-cloud frameworks	15
2.3 Security Paradigms in the AI Era: "Of, By, and For AI"	15
2.3.1 Security for AI: protecting the model	16
2.3.2 Security by AI: AI as a defender	17
2.3.3 Security of AI: ensuring safe outputs	18
2.4 Security Risks in the Generative AI Era	18
2.4.1 Shadow AI and insider threats	18
2.4.2 Social engineering and phishing	19
2.4.3 Prompt injection: the SQL injection of the AI era	20
2.5 AI-Enhanced Detection and Redaction Techniques	21
2.5.1 Semantic classification with BERT and transformers	21
2.5.2 Named entity recognition (NER) and secure redaction	22
2.5.3 Human-in-the-loop (HITL) security	22
2.6 Sector-Specific Implications and Challenges	23
2.6.1 Healthcare and pharmaceuticals	23
2.6.2 IoT and automotive	23
2.6.3 Energy and education	24
2.6.4 Software development	24
2.7 Regulatory Frameworks and Governance	24
2.7.1 Global AI regulations	25
2.7.2 Governance strategies	25
2.8 Zero Trust Architecture (ZTA) in the AI Era	26
2.9 Conclusion and Research Gap	27

3. METHODOLOGY AND FRAMEWORK DESIGN **29**
3.1 Overview of the PromptShield Architecture 29
3.2 Threat Model 29
3.3 Semantic Classification Pipeline 33
3.4 Policy Engine Design 35
3.4.1 Role-sensitivity policy model 36
3.4.2 JSON-based policy definition schema 36
3.5 Entity-Aware Redaction Module 38
3.5.1 Named entity recognition for sensitive content detection 38
3.5.2 Redaction algorithm 40
3.5.3 Interaction with the user and policy engine 40
3.6 Dataset Construction and Preparation 41
3.7 Implementation Details 43
3.7.1 Development environment 43
3.7.2 System integration and execution flow 44
3.7.3 NLP classifier implementation 44
3.7.4 Role-based policy engine 45
3.7.5 Entity-aware redaction 45
3.7.6 User interface 45
3.7.7 Logging and audit trail 46
3.7.8 Performance considerations 47
4. PERFORMANCE EVALUATION **49**
4.1 Evaluation Setup 49
4.1.1 Dataset description 50
4.1.2 Evaluation components 50
4.2 Classifier Accuracy 50
4.3 Per-Class Precision, Recall, and F1-Score 51
4.4 Error Analysis 54
4.5 Policy Enforcement Behavior 56
4.6 Redaction Accuracy 56
4.7 Qualitative Prompt Evaluation Examples 57
4.7.1 User input prior to classification 57
4.7.2 Warn decision with entity-level redaction 58
4.8 Runtime Performance 58
4.9 Summary 59
5. DISCUSSION **63**
5.1 Interpretation of Results 63
5.2 Relation to Research Questions 65
5.3 Comparative Analysis with Existing Solutions 66
5.4 Implications for Enterprise Adoption 66
5.5 Limitations and Challenges 67
5.6 Future Directions 68
5.7 Summary 68
6. CONCLUSION **71**
6.1 Contributions 72
6.2 Future Directions 72
6.3 Closing Remarks 73
REFERENCES **75**
APPENDICES **81**
APPENDIX A: Source Code 83
CURRICULUM VITAE **89**

ABBREVIATIONS

AI : Artificial Intelligence
AWS : Amazon Web Services
API : Application Programming Interface
CRA : Cyber Resilience Act
CPU : Central Processing Unit
DLP : Data Loss Prevention
DPI : Deep Packet Inspection
GDPR : General Data Protection Regulation
GenAI : Generative Artificial Intelligence
GPE : Geo-Political Entity
HITL : Human-in-the-Loop
HR : Human Resources
IO : Input/Output
ISP : Information Security Policy
JSON : JavaScript Object Notation
KVKK : Turkish Personal Data Protection Law
LLM : Large Language Model
ML : Machine Learning
NER : Named Entity Recognition
NLP : Natural Language Processing
NIST : National Institute of Standards and Technology
ODF : OpenDocument Format
PII : Personally Identifiable Information
RAM : Random Access Memory
RF : Random Forest
RLHF : Reinforcement Learning from Human Feedback
R&D : Research and Development
TEE : Trusted Execution Environment
TLS : Transport Layer Security
UI : User Interface
ZTA : Zero Trust Architecture
XML : Extensible Markup Language

SYMBOLS

x_i	: Input prompt i
$E(x_i)$	: MiniLM embedding of prompt x_i
d	: Dimensionality of embedding vector
T_k	: k-th decision tree in the Random Forest
K	: Total number of decision trees in Random Forest
y_i	: Predicted sensitivity label of prompt x_i
A_r	: Allowed sensitivity set for role r
W_r	: Warning-required sensitivity set for role r
B_r	: Blocked sensitivity set for role r
$P(r, y_i)$	: Policy decision for role r and label y_i
$\mathbf{Y}$	: Set of predictions from all trees
l_e	: Entity label associated with entity e

LIST OF TABLES

Page

Table 2.1: Comparison of existing approaches vs. PromptShield requirements. . **27**
Table 3.1: Prompt dataset distribution by role and sensitivity. **43**
Table 4.1: Overall classification accuracy on the test set..... **50**
Table 4.2: Per-class precision, recall, and F1-scores for the classifier..... **52**

LIST OF FIGURES

Page

Figure 1.1: High-level PromptShield architecture. **3**
Figure 3.1: PromptShield end-to-end workflow. **30**
Figure 4.1: Confusion matrix of predicted vs. true sensitivity labels. **51**
Figure 4.2: Distribution of enforcement actions across roles. **56**
Figure 4.3: User interface screenshot showing the original HR prompt. **57**
Figure 4.4: PromptShield interface displaying a warn decision. **58**

PROMPTSHIELD: POLICY-AWARE DATA LOSS PREVENTION FRAMEWORK FOR GENERATIVE AI PROMPTS

SUMMARY

Generative artificial intelligence (GenAI) systems that are capable of generating content, summarizing documents, analyzing trends, assisting with coding, and automating communications have altered the nature of knowledge work. However, this shift creates new security and privacy issues. Employees use free-form prompts when using large language models (LLMs), which often include personal identifiable information, financial data, proprietary research materials, and/or strategic corporate insights. Unlike typical enterprise software systems that operate in defined parameters, GenAI accept unrestricted text from users to provide a new, largely unprotected, data-exposure point. When users enter their prompts into cloud-based LLMs, the prompts are recorded, stored, used to enhance the model(s), and/or stored in regions over which organizations have no jurisdiction. These risks expose a significant weakness in current Data Loss Prevention (DLP) architectures.

Present-day DLP architectures are focused primarily on monitoring email traffic, file uploads/downloads, data storage repositories, and/or user endpoints. Traditional DLP architectures focus on structured data flows and static content. Therefore, traditional DLP architectures are ineffective against the ephemeral, conversational nature of GenAI prompts residing temporarily in the interface before being sent to an Application Programming Interface (API). Furthermore, safety filters provided by LLM vendors are generally limited to controlling what a model produces after it receives the user's input, and do not prevent sensitive information from being communicated to an LLM in the first place. The objective of this dissertation is to close this gap by providing a Client Side, Policy-Aware, Real Time Data Loss Prevention (DLP) Framework (PromptShield) that is designed to protect GenAI prompts from users entering sensitive information.

PromptShield functions as an interceptive layer that examines each user input prior to submission to an LLM. To accomplish this goal, PromptShield was developed to perform lightweight computations with a low-latency response while seamlessly integrating into existing enterprise workflows. The PromptShield framework is comprised of three interconnected components which are semantic classifier, a role-based policy engine, and an entity-aware redaction component. The semantic classifier utilizes a combination of MiniLM Sentence Embeddings and a Random Forest (RF) Model to classify user prompts as either Safe, Risky, or Confidential. The output of the classifier provides a baseline risk assessment, which is further enriched by the role-based policy engine. The role-based policy engine takes the classification produced by the semantic classifier and maps it to a user-defined Allow, Warn, or

Block action based upon the user's organizational role. The entity-aware redaction component of the system will enable users to anonymize sensitive entities that are identified by the system as part of the Named Entity Recognition (NER) process. The combination of this entity-aware redaction capability along with the other described capabilities provide for a trade-off between providing high levels of protection for the data, while minimizing disruption in the user's workflow.

For the purposes of training the dataset consisted of 900 generated enterprise prompts spread evenly across the domains of Human Resources, Finance, and Research & Development. Each prompt included a sensitivity label. A test set of 435 unseen prompts were utilized to evaluate the generalizability of the classifier. The classifier performed with an average accuracy of 72.92%. Compared to large transformer models, the accuracy may be slightly lower; however, it is suitable for a real-time client-side application optimized for speed and response. Examination of the confusion matrix shows that the majority of misclassifications occur between Safe and Risky categories. Misclassifications within the borderline categories, e.g., "Can I review the salary distribution data for trend analysis?" is not directly indicative of HR-related confidential information, yet does reference a potential source of HR-confidential information indirectly. Contextual subtleties, such as these, are difficult for the lightweight embedding models to recognize. However, the classifier demonstrated a very high recall rate for the Confidential category, thus minimizing the risk of high-risk false-negatives.

The role-based policy engine demonstrated its capability to dynamically adjust the risk assessments based on the role of the user. In particular, HR-related prompts were frequently classified as Warning because they contained personal identifiers and internal employee data. Conversely, Finance-related prompts had a much more relaxed pattern, reflecting the organization-wide norms governing the handling of transactional/budgetary information with fewer constraints. Meanwhile, R&D-related prompts showed the highest percentage of Block actions, consistent with the strictest confidentiality standards of research domains. This variation supports the notion that DLP enforcement must be context-aware, i.e., the same sensitivity classification can represent different levels of risk depending on the department utilizing the LLM.

The redaction module was evaluated qualitatively and demonstrated the ability to effectively mask sensitive information while maintaining the integrity of the original sentence structure. The redaction module relied on Named Entity Recognition (NER) to identify various types of high-risk expressions common to enterprise prompts, including PERSON, ORG, Geo-Political Entity (GPE), MONEY, DATE, and Location (LOC). These Redacted Outputs retained structural coherence so that users were able to modify their prompt without losing its intended meaning. One of the major usability issues that are inherent in traditional DLP solutions is that they completely block all of the content and the user either becomes frustrated or will find ways to circumvent it. With the ability for users to safely reformulate their original content, PromptShield allows organizations to protect their data by maintaining productivity.

To demonstrate the practicality of implementing PromptShield in real world business environments, the entire PromptShield pipeline was evaluated to determine processing time. The total processing time for the PromptShield pipeline (embedding,

classification, policy mapping and optionally redacting) was approximately 120-150 milliseconds per prompt when run on a standard laptop with no GPU acceleration. These results indicate that the PromptShield System could be run on user's local machines without negatively impacting perceived performance from the user; therefore it is suitable for various use cases including browser add-ons, chat interfaces, Integrated Development Environments (IDE), and Corporate Communication Platforms.

The discussion of results highlights that although PromptShield performs well in all key areas of evaluation, there are still several challenges to overcome. The most significant limitation is detecting sensitivity in prompts where risk is inferred through implied context rather than explicit entities. Higher precision in these contexts may be achievable through the use of domain-specific embeddings, or through hybrid models that utilize metadata during inference. A second limitation of PromptShield is its reliance on general-purpose NER. While general-purpose NER is effective for identifying common entity types, it may fail to identify organization-specific identifiers, such as internal project codes, proprietary acronyms, or confidential system labels. Future work may include fine-tuning NER models on domain-specific datasets. Also, Adaptive learning from user feedback, behavior based analytics or risk scoring will be important in developing a more responsive and dynamic enforcement mechanism for PromptShield in the future since current policy is static.

In spite of these limitations, the results indicate that prompt-level DLP is technically feasible and strategically necessary for modern organizations. PromptShield closes a significant gap created by traditional cybersecurity tools and LLM vendor safety mechanisms by focusing on the input stage of human and Artificial Intelligence (AI) interactions. PromptShield presents a unique, practical, and effective way to prevent unintended data leakage in environments where employees increasingly rely on GenAI systems. The design of PromptShield makes it easy to integrate into production environments due to its light-weight design, modular architecture, and ease of integration.

Ultimately, the thesis demonstrates how PromptShield fills an urgent need for enterprises, while establishing a solid base for developing the next generation of AI governance, responsible AI deployment, and data protection enterprise models. The work is part of a large body of research which emphasizes the need for transparency, user-centered development, and real-time control with regard to AI systems. With continued use of GenAI by companies, as such as PromptShield, frameworks will be increasingly important in protecting confidentially, assuring compliance, and managing ethically data while providing productivity.

PROMPTSHIELD: POLİTİKA FARKINDALIKLI ÜRETKEN YAPAY ZEKÂ İSTEMLERI İÇİN VERİ KAYBI ÖNLEME ÇERÇEVESİ

ÖZET

Üretken yapay zekâ (GenAI) sistemlerinin kurumsal ortamlarda hızla yaygınlaşması, bilginin üretilme, işlenme ve paylaşılma biçimini kökten değiştirmiştir. ChatGPT, Gemini ve Microsoft Copilot gibi büyük dil modelleri, e-posta yazımı, rapor özetleme, kod üretimi, insan kaynakları süreçlerinin desteklenmesi ve finansal analiz gibi çok farklı iş akışlarında günlük çalışma rutinlerinin bir parçası hâline gelmiştir. Bu sistemler, önceki kural tabanlı veya yalnızca sorgu cevap mantığıyla çalışan yapılardan farklı olarak, bağlama duyarlı, doğal ve etkileşimli bir iletişim kurabilmekte olup kullanıcıya serbest bir metin giriş alanı ve otomasyon kapasitesi sunmaktadır. Ancak bu esnek ve serbest biçimli yapı, aynı zamanda yeni bir saldırı ve sızıntı yüzeyi oluşturmaktadır. Kullanıcılar, farkında olmadan kişisel veriler, finansal bilgiler, ticari sır niteliğindeki içerikler veya kaynak kodları gibi hassas verileri GenAI istemlerinin içine yerleştirebilmektedir. Bu istemler de çoğu zaman üçüncü taraf, bulut tabanlı Büyük Dil Modelleri (LLM) sağlayıcılarına gönderilmektedir. Bu durumda, verinin nerede saklandığı, ne kadar süre tutulduğu, modele eğitim verisi olarak kullanılıp kullanılmadığı ve hangi ülke ve uygulama mevzuatına tabi olduğu soruları kritik hâle gelmektedir.

Geleneksel Veri Kaybı Önleme (DLP) çözümleri, ağırlıklı olarak e-posta, dosya paylaşımı, uç nokta hareketleri veya depolama alanları gibi kalıcı ve kanallara göre tanımlanmış veri akışlarını izlemeye ve kontrol etmeye odaklanmıştır. Bu sistemler, genellikle imza tabanlı kurallar, düzenli ifade kalıpları veya dosya sınıflandırma etiketleri üzerinden çalışmaktadır. Dolayısıyla kredi kartı numarası, T.C. kimlik numarası, belirli anahtar sözcükler gibi önceden tanımlı kalıpları yakalamakta başarılıdır. Ancak GenAI kullanımında ortaya çıkan istemler geçici, serbest biçimli ve bağlama son derece bağımlıdır. Çoğu zaman merkezi e-posta ağ geçitlerinden ya da klasik DLP sensörlerinden geçmeden doğrudan tarayıcı veya masaüstü istemci üzerinden dış modele iletilmektedir. Dolayısıyla, klasik DLP çözümleri bu yeni veri temas noktasını ya hiç görememekte ya da ancak çok sınırlı ölçüde kontrol edebilmektedir. Literatürde, veri kaybı önleme alanında makine öğrenmesi, bağlamsal DLP, bulut DLP ve kimlik erişim yönetimi gibi başlıklar altında önemli ilerlemeler kaydedilmiş olsa da, istem (prompt) düzeyinde, gerçek zamanlı, rol farkında ve redaksiyon destekli bir mekanizmanın eksik olduğu görülmektedir.

Bu tez, söz konusu boşluğu adreslemek amacıyla, üretken yapay zekâ istemlerinin hassasiyetini istem gönderilmeden önce değerlendiren, kullanıcı rolüne göre politikalar uygulayan ve gerekirse istem içindeki hassas varlıkları satır içi maskeleyen PromptShield adlı politikaya duyarlı, istem düzeyinde bir DLP çerçevesi önermektedir.

PromptShield, istemci tarafında çalışan düşük kaynak tüketimli bir ara katman (middleware) gibi davranmaktadır. Kullanıcı arayüzü ile dış LLM uç noktası arasına konumlanarak tüm istemleri karşılamakta, sınıflandırmakta ve kurumsal politikalara göre "İzin Ver (Allow) / Uyar (Warn) / Engelle (Block)" kararları üretmektedir. Böylece güvenlik kontrolü, verinin organizasyonun güvenlik sınırını terk etmeden önce uygulanmaktadır. Model taraflı güvenlik katmanları veya bulut DLP çözümlerinin tek başına sağlayamayacağı türden önleyici bir koruma sağlanmaktadır.

PromptShield mimarisi üç ana bileşenden oluşmaktadır: (i) Anlamsal Sınıflandırıcı, (ii) Rol Bazlı Politika Motoru ve (iii) Varlık Duyarlı Redaksiyon Modülü. Anlamsal sınıflandırıcı, MiniLM tabanlı cümle yerleştirmeleri (sentence embeddings) kullanarak her bir istemi vektör uzayına taşımakta ve bu temsilleri bir Rastgele Orman (Random Forest) sınıflandırıcıya besleyerek istemi Güvenli (Safe), Riskli (Risky) veya Gizli (Confidential) sınıflarından birine atamaktadır. Rol bazlı politika motoru, bu sınıf bilgisini kullanıcı rolüyle (İK, Finans, Ar-Ge) birlikte değerlendirerek, her rol için önceden tanımlanmış politika tablolarına göre karar üretmektedir. Örneğin, İnsan Kaynakları rolü için Gizli sınıfındaki istemler her zaman Engelle ile sonuçlanırken, Finans için Riskli istemlere Uyarı + Redaksiyon, Ar-Ge için Riskli ve Gizli sınıflarına doğrudan Engelle kararı verilebilmektedir. Rol bazlı politikalarının uygulanabilirliği JavaScript Object Notation (JSON) formatında dış bir dosyada tanımlanmasıyla sağlanmaktadır. Varlık duyarlı redaksiyon modülü ise spaCy NER modeliyle kişi adı (PERSON), kurum (ORG), coğrafi konum (GPE), tarih (DATE), konum (LOC) ve para miktarı (MONEY) gibi hassas varlıkları tespit ederek, bu varlıkları [REDACTED_PERSON], REDACTED_MONEY] gibi yer tutucularla değiştirmekte ve böylece istemin ana anlamını koruyarak hassas kısımlarını maskelemektedir. Uyarı senaryosunda kullanıcı, hem orijinal hem de redakte edilmiş istemi görebilmektedir.İsterse redakte edilmiş hâliyle devam edebilmekte ve hangi alanların neden maskelendiğini doğrudan gözlemleyebilmektedir.

Tez kapsamında, literatürde bulunmayan "rol + hassasiyet" etiketli bir veri kümesi tasarlanmıştır. Üç farklı kurumsal rol (İK, Finans, Ar-Ge) için toplam 900 eğitim, 435 bağımsız test istemi oluşturulmuştur. Her bir istem, içerdiği kişisel veriler, finansal bilgiler, proje ve geliştirme detayları ve kurumsal gizlilik derecesi dikkate alınarak manuel olarak Güvenli, Riskli veya Gizli sınıflarına atanmıştır. Eğitim ve test kümesi rol ve sınıf dağılımı dengeli olacak şekilde tasarlanmış, güvenlik açısından kritik olan Gizli sınıflarının yeterince temsil edilmesine özellikle dikkat edilmiştir.

Model tarafında, MiniLM tabanlı cümle yerleştirmeleri, hem anlamı iyi yakalaması ve bağlamı aktarabilmesi nedeniyle hem de çalışma süresi açısından tercih edilmiştir. Daha büyük dil modellerinin sağladığı ek doğruluk kazanımının, istemci taraflı ve gerçek zamanlı bir DLP senaryosunda getireceği gecikme maliyetini aşabileceği değerlendirilmiştir. Bu nedenle, hafif (light) ama yeterli performans sağlayan bir model seçilmiştir. Yerleştirme çıktıları, scikit-learn kütüphanesi kullanılarak eğitilen Rastgele Orman sınıflandırıcısına girdi olarak verilmektedir. Sınıflandırıcıdan hem sınıf etiketi hem de sınıf olasılık dağılımı elde edilmektedir. Bu yapı, hem yorumlanabilirlik (karar ağaçları) hem de orta ölçekteki veri kümelerinde istikrarlı performans açısından avantaj sağlamaktadır. Tezde ayrıca sınıflandırma sürecinin

akışı, algoritma ile (örneğin "Prompt Hassasiyet Sınıflandırma Algoritması") biçimsel olarak da sunulmuştur.

Politika motoru, sınıflandırıcıdan gelen etiketle kullanıcı rolünü bir araya getirerek, JSON formatında tanımlı politika tablosuna bakmakta ve bu tabloya göre Allow/Warn/Block kararını üretmektedir. Bu sayede, sınıflandırıcı modelin yeniden eğitilmesine gerek kalmadan sadece politika dosyası değiştirilerek yeni roller eklenebilmekte, mevcut rollerin eşikleri güncellenebilmekte veya daha kısıtlayıcı ya da daha esnek stratejiler uygulanabilmektedir. Bu tasarım, güvenlik ve uyum ekiplerinin teknik ML (Makine Öğrenmesi) detaylarına girmeden, iş birimi seviyesinde politika tanımlayabilmesini mümkün kılmaktadır.

Redaksiyon katmanı ise, özellikle "Uyar" kararının verildiği durumlarda devreye girmektedir. Kullanıcının iş akışını tamamen kesintiye uğratmak yerine, hassas varlıkları maskeleyerek istemin güvenli hâle getirilmesi hedeflenmektedir. Bu yaklaşım, klasik DLP çözümlerinin çoğunlukla başvurduğu "engelle ve logla" stratejisinden daha kullanıcı dostu ve eğitici bir mekanizma sunmaktadır. Kullanıcı, hangi varlıkların neden maskelendiğini gördükçe kendi güvenlik farkındalığını da geliştirmekte ve uzun vadede, istemlerini daha güvenli biçimde formüle etmeye başlamaktadır. Tezde redaksiyon algoritması österilmiştir.Hangi varlık türlerinin maskelemeye tabi tutulacağı konfigüre edilebilir bir küme hâlinde tanımlanmıştır.

Uygulama, Python 3.10 ile geliştirilmiş ve Flask tabanlı hafif bir web arayüzü ile son kullanıcıya sunulmuştur. app.py, istemin alınması, sınıflandırılması, politika motoruna iletilmesi, gerekli ise redaksiyonun uygulanması ve kararın kullanıcıya aktarılması süreçlerini uçtan uca koordine etmektedir. nlp_classifier.py, embedder, model ve etiket kodlayıcısının yüklenmesi ve istemlerin sınıflandırılmasından sorumludur. policy_engine.py politika tablolarını JSON dosyasından okuyarak basit bir sözlük yapısı üzerinden karar vermekte; redactor.py ise spaCy NER modelini kullanarak metin içinde varlık tespiti ve maskeleme yapmaktadır. Tüm bileşenlerin modüler olması sayesinde, gelecekte embedding modeli, sınıflandırıcı veya NER bileşeni, diğerlerinden bağımsız şekilde güncellenebilecektir. Ayrıca her istem için rol, model etiketi, politika kararı ve redakte edilmiş metin bilgisi bir log dosyasında saklanarak, denetim (audit), hata analizi ve sürekli iyileştirme ihtiyaçlarına hizmet etmektedir.

Performans değerlendirmesi kapsamında, sınıflandırıcı %72.92 doğruluk elde etmiş ve özellikle Gizli sınıfında yüksek duyarlılık göstermiştir. Hata analizi, yanlış sınıflandırmaların büyük kısmının Güvenli–Riskli arasında gerçekleştiğini ortaya koymuştur. Örneğin, "bütçe trendlerini analiz etmek için son iki yılın performans verilerini özetleyebilir misin?" gibi istemler yüzeysel olarak hassas veri içermiyor gibi görünse de bağlamla birlikte düşünüldüğünde ücret, performans ya da finansal hedef bilgilerini ima edebilmektedir. Dolayısıyla da sınıflandırma görevini zorlaştırmaktadır. Tezde, bu tür örneklerin özellikle kurumsal bağlam bilgisinin modelde daha güçlü biçimde temsil edilmesine ihtiyaç duyduğuna dikkat çekilmiştir. Buna karşın, sistemin tasarımı sayesinde yüksek riskli içeriğin gözden kaçması görece sınırlı tutulmuştur.

Politika motoru sonuçları incelendiğinde, İK rolü için Uyar ve Engelle kararlarının daha sık, Finans için İzin Ver ve Uyar kararlarının daha dengeli, Ar-Ge için ise Engelle oranının nispeten daha yüksek olduğu görülmüştür. Bu dağılım, kurumsal risk iştahı ve rol bazlı veri hassasiyetiyle uyumludur. Redaksiyon modülü, niteliksel incelemelerde

kişi adları, şirket isimleri, maaş bilgileri ve tarih ifadelerini başarılı şekilde maskelemiştir.Ayrıca cümle akışını bozmadan, istemin kullanılabilirliğini korumuştur. Tezde, Uyar kararı verilen gerçekçi örnek ekran görüntüleri üzerinden, kullanıcının hem orijinal hem de redakte edilmiş istemi nasıl görüntülediği gösterilmiştir.

Zamanlama ölçümleri, PromptShield'in gerçek zamanlı kullanım için elverişli olduğunu doğrulamıştır. Model yükleme yalnızca uygulama başlatılırken yapıldığından, her istem için embedding, sınıflandırma, politika kontrolü ve redaksiyon süreleri toplamda 120 - 150 ms aralığında kalmıştır. Bu değer, tipik web tabanlı GenAI kullanım senaryolarında algılanabilir bir gecikmeye yol açmamakta ve sistemi kullanıcı deneyimi açısından pratik kılmaktadır.

Tez çalışması, literatürdeki çalışmalarla karşılaştırıldığında, üç temel açıdan özgün bir katkı sunmaktadır. Birincisi, DLP kavramını dosya, e-posta veya depolama odaklı klasik kanallardan çıkararak, istem düzeyine getiren ve gerçek zamanlı bir çerçeve önermektedir. İkincisi, rol bazlı politika modeli ile güvenlik kararlarını kullanıcı bağlamına göre uyarlamaktadır. Aynı içeriğin farklı rollerde farklı muamele görmesini sağlayarak, hem güvenliği hem de kullanılabilirliği optimize etmektedir. Üçüncüsü, tamamen engellemeye dayalı yaklaşımlar yerine, varlık düzeyinde redaksiyon ile "kontrollü izin verme" seçeneği sunmaktadır. Bu sayede hem veri sızıntısı riski azaltılmakta hem de kullanıcıların iş akışı kesintiye uğramamaktadır.

Bununla birlikte çalışma, bazı sınırlılıklar (limitations) da içermektedir. Veri kümesi sentetik olduğundan, gerçek kurumsal verilerin taşıdığı dil çeşitliliği ve gizli ipuçlarının tamamını yansıtmayabilir. Kullanılan NER modeli genel amaçlı olduğundan, kurum içi proje kodları, ürün isimleri veya dahili kısaltmalar gibi kuruma özgü varlıkları her zaman tespit edemeyebilir. Sistem yalnızca İngilizce istemleri hedeflemiş ve üç rol (İK, Finans, Ar-Ge) ile sınırlandırılmıştır. Gelecek çalışmalar, çok dilli destek, kuruma özel NER modelleri, gerçek kurumsal verilerle ince ayar (fine-tuning) ve adaptif politika öğrenme mekanizmaları ile bu sınırlılıkları hafifletebilir.

Sonuç olarak, PromptShield, üretken yapay zekâ çağında veri güvenliğine istem odaklı bakan yeni bir bakış açısı sunmaktadır. Tezin katkıları, istem düzeyi DLP kavramının uygulanabilir ve gerekli olduğunu göstermektedir. Semantik sınıflandırma, rol bazlı politika ve varlık duyarlı redaksiyonu bir araya getirerek, gerçek zamanlı ve bir çerçeve ortaya koymaktadır. Bu yaklaşım, kurumsal GenAI kullanımının güvenli, uyumlu ve sürdürülebilir bir biçimde ölçeklenebilmesi için önemli bir temel sunmaktadır.

Ek olarak, bu çalışma üretken yapay zekâ kullanımının kurumsal güvenlik mimarileri üzerinde yarattığı dönüşümü de kavramsal açıdan tartışmaktadır. GenAI sistemlerinin klasik BT denetim yapılarıyla uyumsuzluğunun temel sebepleri arasında, veri akışının kontrol noktalarını aşarak doğrudan bulut tabanlı modellere yönelmesi, istem düzeyinde denetimin daha önce tanımlanmamış olması ve kullanıcıların veri sınıflandırmasını genellikle manuel biçimde, yani bilinçli bir karar süreci olmadan gerçekleştirmesi yer almaktadır. Bu bağlamda PromptShield, insan–makine etkileşiminde yeni bir güvenlik seviyesi oluşturarak, veri gizliliğini model eğitimi, çıktı filtreleme veya ağ denetimi yerine daha erken bir aşamada, "kullanıcı davranışının başladığı noktada" ele almayı önermektedir. Bu yaklaşım, zero-trust mimarileri ve veri minimizasyonu ilkeleri ile de uyumludur.

Ayrıca tez sürecinde elde edilen bulgular, kurumların GenAI benimseme stratejilerinin yalnızca teknik güvenlik önlemleriyle değil, aynı zamanda kullanıcı farkındalığı, politika tanımları ve kurumsal yönetişim modelleriyle desteklenmesi gerektiğini göstermiştir. PromptShield'in Uyar + Redaksiyon mekanizması, kullanıcıları pasif birer muhatap olmaktan çıkarıp güvenli kullanım konusunda aktif bir öğrenme sürecinin parçası hâline getirmektedir. Her bir uyarı ve redaksiyon, kullanıcının veri duyarlılığını daha iyi kavramasına yardımcı olmaktadır. Böylece sistem, yalnızca teknik bir kontrol mekanizması değil, aynı zamanda davranışsal bir farkındalık aracı olarak da işlev görmektedir. Bu açıdan PromptShield, kurumsal GenAI yönetişim stratejilerinin önemli bir bileşeni olmaya adaydır.

Çalışmanın bir diğer önemli katkısı, istem bazlı güvenlik değerlendirmesinin gelecekte daha geniş bir ekosistemle bütünleştirilebileceğini ortaya koymasıdır. Örneğin, istemlerin risk seviyeleri kurum içi Güvelik Bilgisi ve Olay Yönetimi (SIEM) çözümlerine gerçek zamanlı olarak aktarılabilir, kullanıcı bazlı risk skorları üretilebilir veya yüksek riskli senaryolarda ek kimlik doğrulama adımları devreye alınabilir. Benzer şekilde rol bazlı politikalar, gelecekte departman, proje, veri sınıfı veya hassasiyet etiketi gibi daha ince gruplara ayrılabilir. Bu sayede PromptShield, yalnızca bir istem filtresi değil, çok katmanlı bir AI güvenlik çerçevesinin merkezi bir bileşeni hâline gelebilir.

Tüm bu değerlendirmeler ışığında PromptShield'in, üretken yapay zekâ ile çalışan kurumlar için yalnızca bugün ortaya çıkan riskleri azaltmakla kalmayıp, gelecekteki düzenlemelere, denetim gereksinimlerine ve kurumsal veri yönetimi standartlarına uyum sağlayabilecek esnek bir altyapı sunduğu görülmektedir. GenAI teknolojilerinin hızla evrilmesi, model parametrelerinin büyümesi, bağlamsal bellek özelliklerinin gelişmesi ve multimodal etkileşimlerin yaygınlaşması dikkate alındığında, istem düzeyinde kontrol mekanizmalarının daha da kritik hâle geleceği öngörülmektedir. Bu bağlamda PromptShield, ileride görüntü, ses ve belge içeriklerinin de istem olarak işlenebileceği çok modlu mimarilere genişletilebilir.

Sonuç olarak, bu tez yalnızca bir prototip geliştirmekle kalmayıp, kurumsal GenAI kullanımında güvenlik paradigmasının yeniden düşünülmesi gerektiğine de işaret etmektedir. PromptShield, istem içeriğinin sınıflandırılması, politika tabanlı karar verme ve varlık düzeyinde redaksiyonu bir araya getirerek, hem güvenlik hem de kullanılabilirlik odağını dengede tutan yeni bir yaklaşım önermektedir. Bu yaklaşım, veri gizliliğini en erken temas noktasında uygulayarak sızıntı riskini azaltmakta ve modern işletmelerin güvenli GenAI kullanımına geçişini desteklemektedir. Tez kapsamında geliştirilen mimari, ileride yapılacak akademik ve endüstriyel çalışmalar için hem teknik hem yönetsel açıdan sürdürülebilir bir temel sunmaktadır.

1. INTRODUCTION

The rapid growth in use of Generative Artificial Intelligence (GenAI), with tools like ChatGPT, Gemini and Microsoft Copilot are changing the way organizations collect, analyze and distribute information; GenAI has evolved from a single text based model to an array of multimodal models that can generate and understand text, images, code and other forms of content.

The newer transformer architectures and the use of Reinforcement Learning From Human Feedback (RLHF) by modern GenAI models distinguishes them from the older Rule-Based Systems and Retrieval Models. These architectural advances enable GenAI models to be able to provide creative solutions to problems and have contextually aware natural conversations with humans. The expansion of GenAI has been revolutionary in its potential to be transformative in many sectors such as; Education, Financial Services, and Software Development, in addition to providing innovative solutions for businesses, GenAI will allow for rapid development of knowledge and the automation of tasks.

GenAI enables employees to rapidly accomplish tasks via conversational interfaces (e.g., email writing, report summaries, writing source code), that formerly required numerous hours of human effort. While this advancement in technology enhances employee's productivity and ability to make informed decisions, it has introduced a significant, but under researched risk, of unintentional data leakage from users through their input prompts.

As opposed to traditional enterprise software; GenAI tools function as dynamic, open text interfaces with potentially unpredictable outputs. Unknowingly to employees, when sending input prompts to external APIs, they could inadvertently submit confidential data, for example, financial information, personal identifiers, or even source code.

Traditional Data Loss Prevention (DLP) solutions focus on static documents, e-mails, or network traffic. However, these tools are unable to inspect or enforce policies on the transient, ephemeral interactions of GenAI prompts. This thesis aims to address this security and compliance gap.

1.1 Background and Motivation

Companies today are using GenAI in their work flows each day to create drafts, summaries, code, and analysis. The GenAI interface is an open ended form and language based whereas traditional enterprise software has fixed formats and forms.

The increased flexibility of GenAI creates increased productivity; however, this increased flexibility creates new security and privacy risks which are based on two primary elements, the human factor (the user's input) and the situational context (the user's role, department, etc.).

Traditional DLP solutions have focused on persistent content and the use of perimeter controls to monitor access through email, storage devices, and end points. However, as the nature of the input is changing from a static format to an ephemeral format, it is becoming increasingly difficult for legacy controls to capture the flow of sensitive information to a third-party LLM. Sensitive information can include financial projections, unreleased product information, personal data, and/or source code being entered directly into a chat box. The transmission of sensitive information to a third-party LLM raises several issues regarding the governance of that information, i.e., retention, training usage, jurisdiction and auditable record. As a result, there is a growing need for a prompt-first solution that is able to be used inline, is aware of the user's role and requires low latency.

(Prompts are intercepted at the client, semantically classified, evaluated against role-based policies, optionally redacted, and only then transmitted to the LLM endpoint. An audit log records decisions for compliance.)

PromptShield as shown in Figure 1.1 operates at the user interface layer, intercepting prompts before they are transmitted to the LLM. This placement ensures that sensitivity

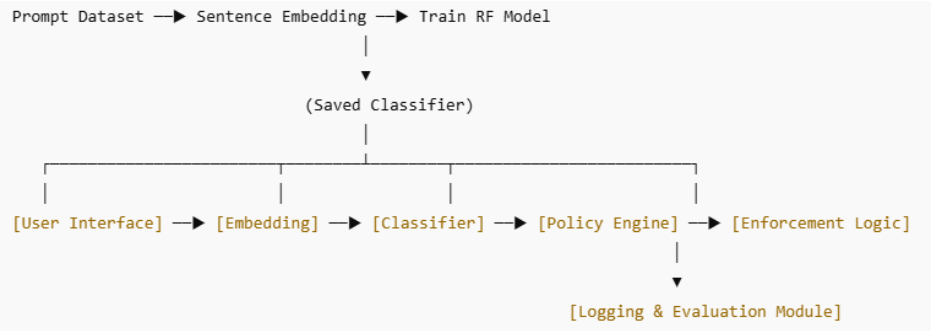

Figure 1.1: High-level PromptShield architecture.

classification, policy enforcement, and redaction happen inline, without altering or depending on the downstream AI model.

1.2 Problem Statement

The central problem addressed in this thesis is the lack of an enterprise grade, prompt level DLP mechanism capable of:

1. determining the *sensitivity* of a free-text prompt in real time,
2. applying *role specific* enforcement policies (Allow, Warn, Block) in a consistent and transparent manner,
3. performing *inline redaction* to safely sanitize borderline prompts while preserving their intended meaning,

all while meeting the strict latency requirements necessary for everyday use.

1.3 Purpose of Thesis

Modern organizations require security tools which are able to understand the context of both the user's identity and their intended use of sensitive information; and will also be easy to utilize by users. This dissertation presents PromptShield; a real-time Data Loss Prevention (DLP) framework that is policy-aware and designed specifically for GenAI prompts.

PromptShield is a client-side tool. It receives user input prior to the user submitting their input to the LLM model and evaluates user-generated input against pre-determined policies related to data protection.

The main goals of PromptShield include:

- To classify each user generated prompt within three sensitivity-based categories (Safe, Risky, Confidential) via semantic embeddings and a lightweight machine-learning classifier.
- To enforce role-based organizational policies as they relate to each prompt; i.e., allow "Safe" content from Human Resources (HR), warn on "Risky" content submitted by Finance, and block all "Confidential" Research and Development (R&D) content.
- To apply redaction techniques directly to sensitive entities using named entity recognition (NER); thus masking names, organizations, cities, states, countries, etc. and/or financial values prior to submission.
- To operate at low-latency, and maintain log files of all events for auditing and post-incident analysis.

1.4 Research Questions and Objectives

The research is guided by four key questions:

RQ1: Can prompt sensitivity be inferred reliably in real time using compact semantic models?

RQ2: Does role-based policy enforcement enhance DLP precision while reducing friction?

RQ3: Can inline NER-based redaction preserve prompt usability while mitigating risk?

RQ4: Is prompt-level DLP feasible within enterprise latency constraints?

Based on these, the primary goals of this research project are as follows:

- Developing a modular design that analyzes user prompts prior to their submission.
- Creating a labeled, role-aware data set for use in both training and evaluating models.
- Combining semantic classification with a rule-based policy engine.
- Measuring accuracy, latency, and enforcement behavior across enterprise roles.

1.5 Research Methodology Overview

This research will utilize an engineering research methodology:

1. **Establishing system requirement:** Determine enterprise leakage scenarios by role type (HR, Finance, R&D), and establish system policies and requirements.
2. **System Design:** Create client-side architecture with decoupled modules for classification, policy evaluation and redaction.
3. **Data Preparation:** Develop a role-aware prompt data set with three levels of sensitivity (Safe, Risky, Confidential) for testing/training.
4. **Implementation:** Implement a semantic classifier using sentence embeddings and a lightweight learner; implement a rule-based policy engine; implement an NER-based redactor.
5. **Evaluation:** Evaluate accuracy, confusion patterns, enforcement distributions, redaction behavior and end-to-end latency.

1.6 Threat Model and Assumptions

Adversary: The primary adversary is accidental leakage by well intentioned employees. The system also considers prompt manipulation risks (e.g., injection/jailbreak attempts) as a driver for conservative policy defaults.

Trust Boundaries: Client devices and the PromptShield enforcement runtime are within the enterprise trust boundary; LLM endpoints are treated as external services. Policies must be enforced before data crosses this boundary.

Objectives: Reduce leakage risk without materially harming usability: false positives should be mitigated with *Warn + Redact*, while clearly sensitive prompts are *Blocked*.

Non-Goals: The framework does not prevent exfiltration through *non-GenAI* channels, nor does it govern model side retention/training policies.

1.7 Scope and Limitations

Scope: PromptShield targets English prompts in three enterprise roles (HR, Finance, R&D) and three sensitivity classes. Enforcement actions are Allow, Warn (with optional redaction), and Block.

Limitations: The dataset includes synthetic prompts designed to emulate enterprise phrasing and may not capture all edge cases. The redactor relies on general purpose NER and may miss domain specific entities and multilingual support is left for future work.

1.8 Contributions

While moving DLP to the prompt boundary will provide a remedy for shortcomings in current control mechanisms, PromptShield fills a gap in traditional controls. The model is (i) policy aware when users engage with GenAI, (ii) explainable because user's decisions and redactions are visible to them, and (iii) deployable across multiple vendors. It is expected that the overall result of this work will be an improvement in confidentiality posture.

The research presented in this dissertation adds to existing literature regarding GenAI and data protection through:

1. **A policy aware framework for DLP at the prompt:** A client-side system combining semantic classification, role-based policy enforcement, and inline redaction for interaction between users and GenAI systems.

2. **Role specific data set and classifier:** 900 training and 435 unseen testing prompts classified into three roles (HR, Finance, R&D).

3. **Inline entity aware redaction:** A NER-based masking module that maintains grammatical structure while removing identifiers that could be used to identify sensitive information.

4. **Complete evaluation:** The system achieves 72.92% accuracy on unseen prompts and high recall score for confidential class with an average end-to-end latency of less than 150 ms demonstrating its feasibility in a real-time environment.

5. **Expandable architecture:** The modular nature of the framework allows it to function as either a browser plugin or api gateway and support future development such as adaptive policies and more advanced redactors.

1.9 Thesis Organization

The remaining parts will be structured as follows:

- **Chapter 1** provided an introduction to the problem with which we are dealing, a description of our objectives and of what we have to contribute to the existing body of knowledge in this field.

- **Chapter 2** examines the literature concerning Data Leakage Prevention (DLP), General AI security, and Privacy Preserving Techniques.

- **Chapter 3** describes how we designed, developed, and implemented the PromptShield architecture and its related algorithms.

- **Chapter 4** explains the experimental designs and methods used and the test results obtained using the PromptShield architecture and algorithms.

- **Chapter 5** interprets the results from our experiments, discusses the limitations of the study, and evaluates the potential implications for enterprises.

- **Chapter 6** summarizes our major conclusions, and identifies areas for further investigation that we propose to pursue.

2. LITERATURE REVIEW

As GenAI is quickly being incorporated throughout corporate operational processes, we are experiencing a paradigm shift in how corporations utilize Information Technology (IT) to create content and code autonomously. With GenAI creating content and coding at unprecedented rates and with innovative applications throughout multiple industries, we are also seeing GenAI create vulnerabilities in a corporate environment by eroding traditional perimeter-based security methods that had been used to protect against sophisticated threats for which existing Data Loss Prevention (DLP) systems cannot provide adequate protection. This Chapter represents an extensive and critical evaluation of the body of work on this subject that has been generated through over 60 peer reviewed academic articles and technical reports along with applicable regulatory guidance documents provided for this study. The Chapter reviews the foundational theories of information security, provides a detailed description of how DLP evolved from static scanning to an AI driven adaptive inspection tool, identifies the various types of threats presented by LLMs and assesses the implications of these threats to different business sectors as a result of this paradigm shift in IT.In conclusion, this Chapter provides an overview of the gaps within the present body of literature as part of its proposed system architecture.

2.1 Theoretical Foundations and Conceptual Frameworks

It is essential to define theoretically, establish conceptually and determine functionally the key terminology (i.e., "core terms") which underlie the arguments for enhanced DLP mechanisms prior to establishing a robust case for them.

2.1.1 Data loss prevention (DLP): definitions, states, and mechanisms

Data Loss Prevention (DLP) is formally defined in the literature not merely as a tool, but as a comprehensive strategy encompassing technologies, processes, and

policies designed to detect and prevent the unauthorized transmission, use, or storage of sensitive information [1]. The theoretical scope of Data Loss Prevention (DLP) expands beyond merely blocking files to include an overall understanding of data states and their lifecycles. Tahboub and Saleh [1] define categories of DLP systems based on the data state they protect:

- **Data-at-Rest:** This includes inactive data that exists in physical form in databases, file systems, cloud-based infrastructures, and mobile devices. The primary goal of protecting data-at-rest is to limit unauthorized access to it through encryption, access control lists (ACLs), and through data retention policies. In order to effectively protect data in today's distributed computing environment, Ahmad et al. [2] contend that cloud-based security frameworks need to be able to utilize strong key management services. They also argue that simply encrypting static data in storage does not sufficiently protect the data without implementing additional functionality such as dynamic key rotation and audit logging to ensure authorized users have access to data.

- **Data-in-Motion:** This represents data that is transmitted over a network; whether that is within a company's internal LAN or over the Internet. Protecting data while it is in motion relies primarily on Transport Layer Security (TLS) and Deep Packet Inspection (DPI). Ghouse et al. [3], proposed new methods of encryption that are specifically designed for data in transit. Using Artificial Intelligence (AI)-based traffic analysis they demonstrated that they could dynamically vary the level of encryption applied to a given data stream based on the sensitivity rating assigned to each data stream.

- **Data-in-Use:** This is perhaps the most vulnerable state, as it represents data that is currently being utilized by a system (e.g., existing in Random Access Memory (RAM), Central Processing Unit (CPU) registers, or clipboard). This is where humans interact with the data, which is also where "Insider Risk" is at its highest. Monitoring this state requires endpoint agents to monitor this state as network-based controls typically do not have visibility into what is occurring locally

on an endpoint, such as copying and pasting information or taking a screenshot of the screen.

The progression of DLP definitions has transitioned from being content-aware (i.e., what the data is) to context-aware (i.e., who, where, when, and how the data is being used). Understanding this theoretical development will be important in explaining why previous generations of DLP products failed to meet expectations in the GenAI era.

2.1.2 From theory to applications generative artificial intelligence (GenAI)

Generative AI can be defined as a type of artificial intelligence (AI) system capable of creating new material, for example: text, images, audio and code, based upon user input or requests. A comprehensive overview of GenAI was provided by Kalota [4]. Kalota distinguishes between GenAI and traditional discriminative AI by noting that GenAI models learn the underlying probability distributions within their training data; enabling them to create new output based upon those learned distributions.

As discussed at length by Kalota [4], the major technological advancement which has enabled GenAI is the use of the Transformer architecture. Similar to LLMs such as GPT-4, these models utilize self-attention mechanisms to allow them to process input sequences in parallel, therefore, allowing them to find long range dependencies and context which were not previously possible with RNNs.

Wang et al. [5] noted that in the context of IoT and edge computing GenAI operates as a "black box". The decision-making process within the model is opaque and therefore difficult to determine when the model may leak training data or act maliciously. The probabilistic nature of the model also creates non-deterministic outcomes; thereby compounding issues related to traditional security verification methods which require deterministic system behavior.

2.1.3 Information security policy: development and governance

The Information Security Policy (ISP), provides the governance layer for technical controls. Flowerday and Tuyikeze [6], have suggested a conceptual framework. The framework emphasizes that policies must be dynamic, and adapt to the evolving threat

environment. Therefore, they suggest that it is very important to define "the who, what and how" of implementing a policy as the actual content of the policy.

Similarly, Paananen et al. [7] have identified that there is a significant gap in the area of policy formulation and policy implementation. In fact, they have reviewed the current state of Information Security Policy (ISP) development and argue that because many organizations do not enforce their ISPs through automation; many ISPs exist only on paper. As Gen AI continues to grow, this gap will continue to grow as employees continue to utilize tools like ChatGPT for work purposes without being aware of the potential data privacy implications of these actions. This is why organizations must convert their ISPs into technical controls; for example, the role based restrictions proposed in the PromptShield framework.

2.2 The Evolution of Data Loss Prevention Systems

The evolution in DLP technology is an ongoing struggle between the efforts of those attempting to protect organizations from data exfiltration and the creation of new data exfiltration techniques. In this chapter we analyze the transition from static perimeter-based defense systems to the modern dynamic, kernel-level and AI-based architectures.

2.2.1 Traditional approaches: the era of static inspection

The first generation of DLP technology was primarily based upon networks, and used static content inspection. According to Tahboub and Saleh [1], these DLP systems used mechanisms to identify structured data such as credit cards using RegEx and exact data matching (EDM) to fingerprint sensitive database records. While these technologies are very good at identifying structured data, they have shown a lack of robustness.

An extensive experimental analysis of commercial DLP technologies were performed by Ghorbanian et al. [8]. The results of their experiments revealed significant limitations to these products. They found that simple evasion techniques can be used against them, such as renaming the extension of a file, converting the text into base 64 format, or embedding the data in a picture (steganography).

Al-Kilani et al. [9] also developed on this topic and demonstrated that encrypted archives (such as password-protected zip files) made it impossible for network-based DLPs to identify the content of the archive. Al-Kilani et al. analyzed multiple different types of exfiltration attempts (file structuring and header modification), and found that all of them resulted in standard DLP agents being unable to determine if data was being leaked from the system, so long as the signature of the file did not match the content of the file.

As for web traffic, Gugelmann et al. [10] showed that because the increasing number of websites using https and custom protocols make network inspection difficult to perform due to privacy issues, and computational issues. In addition to those issues, their study on content-based DLP technologies in web traffic showed that, while they can clearly detect data that is sent in clear text, they are much less successful when data is sent over covert channels, or is encrypted.

2.2.2 Transition to context-aware and usage control models

Kaur et al. [11] point out that classical DLP systems which are based on content-only inspections (for example, based on signatures or based on regular expressions) have several drawbacks: high numbers of false-positives; poor support for different types of data and formats; and poor scalability. As a result, there is a strong need for more sophisticated, context-aware or usage-control oriented DLP system designs that can address these issues.

Wuchner et al. [12] develop UC4Win, an endpoint DLP system that is designed to run on top of Windows operating systems, and is based on the idea of providing data-driven usage-controls at an application level. Unlike traditional DLP systems that match patterns in document content using signature-based detection, UC4Win is able to monitor the flow of data between applications, capture representation-independant data flows using Windows API calls, and enforce both temporal and cardinal constraints on how data may be used. This provides users with the ability to define fine-grained, context-dependent policies that are not possible with traditional DLP systems.

The recent work of Paracha et al. [13] further supports this gap in current DLP systems by demonstrating that most current endpoint DLP systems continue to focus primarily on channel-specific controls (such as USB-blocking, Outlook attachment-filtering and regular expression content-matching). They also demonstrate through their evaluation that these systems require manual rule-construction, lack semantic understanding and provide limited ability to infer “how” data was used between applications, in addition to detecting file transfers or simple content-signatures.

2.2.3 Kernel and hypervisor-level protections

While Yadav & Gupta [14] have focused on multi-layer, behavior-, and machine learning (ML)-driven Data Loss Prevention (DLP) architectures for applications and networks; hypervisor based designs like Efficient DLP-Visor enforce DLP controls even lower in the system stack - to resist tampering by a local administrator.

As attackers developed techniques to disable endpoint agent software, or manipulate OS Application Programming Interfaces (API), researchers pushed defenses down further in the system stack. Kiperberg et al. [15] presented the Efficient DLP-Visor, a hypervisor-based DLP design.

The hypervisor can monitor, and prevent tampering with kernel-mode system calls from a higher, more secure, hypervisor level than the operating system layer. In that model of an attacker, no matter how much access a user has as a local administrator, they will be unable to circumvent DLP controls as monitoring and enforcement occurs above the guest OS layer.

In the physical world, Srinivasan et al. [16] discovered another overlooked path for data loss (i.e., printed hard copy documents) that was found by using their PrintWatchDog framework, which is able to capture an OpenDocument Format (ODF) print job at the print spooler. This captured job has a corresponding Extensible Markup Language (XML) file, which identifies which "tags" were specified by the users as being "sensitive." The PrintWatchDog framework then performs in-line redaction(s) of those "tags," and releases the document to the printer once it has been modified with the redacted information. In doing so, they illustrated the importance of protecting all

digital egress points, including those that do not exist digitally, such as physical copies of documents.

2.2.4 Hybrid and multi-cloud frameworks

Srivastava et al. [17] have proposed a hybrid model utilizing both signature-based detection for identified threats and anomaly-based detection for unknown or emerging ("zero-day") threats. The hybrid model has the ability to leverage the signature based detection for higher precision while leveraging the anomaly-based detection for higher recall. In doing so, it provides a "defense-in-depth" approach to detecting malicious activity.

Additionally, Muppalaneni et al. [18], identify multi-cloud environments as particularly challenging due to their dynamic nature of data flow, and lack of visibility into the various components involved within these environments. To address these challenges, they recommend AI-enriched DLP strategies that can adapt to learn the normal data flow patterns between diverse cloud services (i.e. Amazon Web Services (AWS), Azure, Google Cloud Platform), and identify anomalies that standard static rules may fail to recognize. Additionally, Esther [19] emphasizes AI's potential role in protecting sensitive data from being accessed without permission via predictive modeling, and indicates that AI can be utilized to predict possible data leaks prior to the occurrence of those events.

2.3 Security Paradigms in the AI Era: "Of, By, and For AI"

More recent studies describe the relationship between AI and security with the help of three different paradigms: Security by AI; Security of AI; and Security for AI. The authors of Bertino et al., [20], indicate that AI has become an essential component in contemporary cybersecurity ("Security by AI"), providing support for multiple functionalities within cybersecurity, including intrusion detection, malware classification, threat intelligence analysis and massive scale anomaly detection. In addition to their indication that AI-driven defenses are vulnerable to adversarial machine learning due to flaws that attackers can exploit ("Security of AI"), Bertino et al.'s panel emphasizes the necessity for robust assurance frameworks, trustworthy

training data, well-designed reward structures and ethical safeguards for the successful deployment of AI in security.

Extending the Bertino et al. conceptual framework, Wang, [21], develops a systematic formulation of Security of, by, and for AI as three mutually dependent research domains that increasingly frame the AI centered security environment.

Security for AI addresses the protection of AI assets, especially training data, model parameters, and inference pipelines, through technology applications such as Data-In-Use Protection, Trusted Execution Environments (TEEs), and Confidential Computing.

Security of AI examines vulnerabilities intrinsic to AI systems such as Model Extraction, Poisoning, Backdoor Insertion, and Adversarial Manipulation. Wang's work on Trojan Attacks represents the critical security risk that emerges from AI systems becoming a target of cyberattacks.

Security by AI describes the increasing reliance upon AI to enhance the security of systems, e.g., the utilization of Deep Learning and Natural Language Processing (NLP) to identify emerging threats, the automation of SOC workflows, or the examination of network documentation in carrier grade infrastructures.

The sum total of these perspectives indicates that contemporary cybersecurity is developing into a two-way dependency, where AI systems must be protected, protect themselves, and protect other systems. The triad of, by, and for AI captures the complete range of challenges and opportunities found at the intersection of AI and security and provides a theoretical base for current research in developing trustworthy and policy compliant AI security architectures.

2.3.1 Security for AI: protecting the model

The "AI Security" (or "security for AI") paradigm is concerned with the protection of an AI system itself—the training data, model parameters, inference pipeline, and surrounding computing environment from malicious or accidental compromise. Rahman et al., [22] give a complete listing of all attack types for Machine Learning Models; Yu et al., [23], build upon this work and describe the Endogenous Risks of

AI Security from vulnerabilities in the AI Base Layer, Data Layer, Algorithm Layer, and Model Layer.

Data and Model Layers contain typical attacks including:

- Data Poisoning: A threat where the training data are manipulated in order to reduce the integrity of the model or hide a backdoor within it.
- Model Inversion and Extraction: The reconstruction of sensitive training samples or the theft of proprietary model parameters through repeated questioning of the model.
- Adversarial Evasion Attacks: Introduction of minor, imperceptible perturbation to input sample(s) causing a misclassification at inference time.
- Membership Inference: Whether or not a particular datapoint is part of the training set and therefore a violation of privacy.

Rahman et al. also point out that the defense against such threats will require robust methods such as adversarial training, cleaning of inputs and features, regularization-based hardening, differential privacy, federated learning, anomaly detection, and diversifying models. Yu et al. emphasize that these risks occur throughout the entire AI stack, insecure frameworks, untrusted data pipelines, brittle algorithms, and poorly defined or easy-to-extract models.

2.3.2 Security by AI: AI as a defender

The paradigm for enhancing defense by utilizing AI in technology, and its uses are examined by Kayode et al. [24] in a review of AI-based cybersecurity which discusses how machine learning is being utilized to automate threat detection, anomaly hunting, and automated incident response. The capability to rapidly identify patterns indicative of breaches from large amounts of log data, is an example of what AI algorithms have the ability to accomplish at a speed that would be unattainable by a human analyst. Singh et al. [25] also examine "threat-informed defense", a method by which AI

systems utilize real-time threat intelligence to make continuous changes to their own security posture.

2.3.3 Security of AI: ensuring safe outputs

Ensuring the safety of AI-generated outputs is a fundamental aspect of this paradigm; it provides assurance that AI-generated output will be free from bias, human value alignment and safe, and will prevent the generation of hate speech, phishing email scams, etc., as well as the generation of vulnerable code. PromptShield primarily addresses the domain of ensuring that the interaction with an AI model does not violate organizationally defined security policies, while also preventing both the leakage of sensitive information into the AI model and the generation of unsafe content by the AI model.

2.4 Security Risks in the Generative AI Era

GenAI is accessible by almost everyone now. It is reshaping the cybersecurity landscape and creating a multiplier effect. It leads to a roadt that both defenders and attackers enhances their capabilities significantly [26].

2.4.1 Shadow AI and insider threats

The term "Shadow AI," describes when an employee uses a sanctioned artificial intelligence (AI) tool without permission from their employer. Diro et al. [27], states that 62% of those surveyed (in the study referenced in the paper), said they had entered information about internal process into GenAI Systems. This can cause data leakage, which is unintended sharing of sensitive corporate data (such as source code, financial forecasts, personal identifiable information, etc.), by having employees enter this type of sensitive corporate data into GenAI System prompts intended for public consumption. Once this data has been entered into the GenAI System, the data will then leave the company's control and become part of the GenAI System's training dataset or prompt context window.

These behaviors are often generated based on psychological elements. Saddiqa and Ruohonen [28] , have researched the psychology of insider threat activity; they

found that stress, burnout and the need to produce results under time constraints lead employees to circumvent security protocols. The authors of Asasfeh et al. [29] also expand on human factors in managing security; the authors emphasize that technical solutions should be combined with psychological support and training for employees in security awareness. The authors further contend that determining the motivation (whether it is malicious or negligent) behind these behaviors is the key to creating successful intervention strategies.

2.4.2 Social engineering and phishing

The growing body of research suggests that the use of Generative AI, specifically large language models, has made it easier for attackers to conduct social engineering and phishing attacks. The study conducted by Sebastian [30] shows that 61.4% of respondents felt that social engineering was the number one Cyber Risk associated with AI Chatbots. In addition, respondents indicated that phishing attempts were the second greatest concern at 38.6%. This trend represents the fact that attackers are using LLM's to write email content that makes sense in context to their targets; they are mimicking an organization's tone and style; and they are able to execute large-scale phishing campaigns faster and in a manner that is more personalized than ever before.

As stated above, from a data security perspective, Arora et al. [31] note that current data loss prevention systems will fail to identify GenAI-generated malicious content because legacy pattern matching and rule-based controls cannot determine whether synthetic content is both syntactically acceptable (i.e., semantically acceptable) and semantically acceptable (i.e., syntactically acceptable). In particular, since legacy DLP systems are generally unable to recognize indirect leakage vectors (e.g., job-role hints, process descriptions, etc.) that LLM's can easily create to produce pretexts that appear realistic to users.

Moreover, the study conducted by Teo et al. [32] demonstrates how GenAI-generated misinformation, hallucinations, and data poisoning can be used as high convince social-engineering vectors, especially in areas such as healthcare where users have a tendency to trust responses that sound authoritative.

Collectively, these studies indicate that GenAI does not simply automate phishing but instead, changes the nature of phishing into a semantic manipulation problem, whereby intent is embedded within the structure of the language itself, rather than being detectible through keywords. This directly motivates the design of PromptShield's semantic classifier and policy engine, which operate beyond surface-level string matching, and assess contextual sensitivity prior to allowing harmful content to exit enterprise boundaries. PromptShield directly addresses this shift by operating prior to the LLM generates exploitable content.

2.4.3 Prompt injection: the SQL injection of the AI era

Prompt Injection represents one of the largest risk factors for all LLMs. Prompt Injection attacks involve providing an input that instructs the model to perform actions beyond what it was designed to do. Liu et al. [33] demonstrated that the attacks are able to automatically and universally bypass safety controls on models like GPT-4 and Claude, making it clear that this is a systemic problem.

Sokhansanj [34] has described the different types of threat posed by prompt injection attacks including (i) direct prompt injection, which occurs when the user specifically provides the model with instructions to jailbreak it, and (ii) indirect prompt injection, which occurs when the user provides the model with malicious information via an external source (like a website or email), while remaining unaware that their intent is being altered by the information they provided.

Prompt injection also presents risks for the use of downstream applications that provide capability for harmful uses. Gupta et al. [35] have identified "ThreatGPT" which describes how to create and utilize manipulated prompts to create polymorphic or evasive malware. Zhu et al. [36] proposed a framework for identifying the risks associated with GenAI and identified possible solutions to mitigate those risks including using input-filtering "AI Firewalls" to establish a trusted boundary between LLMs and untrusted external sources.

Mathew [37] identifies additional methods that may be used to bypass safety controls including, but not limited to, role-playing and hijacking a person's identity.

Campbell and Jovanovic [38] identify a disinfecting approach that may be used at an organizational level; the goal of this approach is to sanitize both inputs and outputs of LLMs to reduce the risk of unintended data disclosure, unauthorized execution of instructions and shadow AI.

2.5 AI-Enhanced Detection and Redaction Techniques

GenAI interaction security can be supported by advanced AI techniques to support both classification and redaction.

2.5.1 Semantic classification with BERT and transformers

Deep Learning can be used to better analyze the complex nature of GenAI prompts. Researchers have been able to apply semantics as well as traditional keyword searching to develop Deep Learning solutions to detect sensitive information contained within unstructured text. Ding et al. [39] applied BERT (Bidirectional Encoder Representations from Transformers) to identify sensitive information in unstructured text. BERT has an attention mechanism which provides BERT the ability to contextualize each word in the prompt (i.e. "Apple" the company vs. "apple" the fruit).

Researchers Saritha and Kumar [40] investigated a variety of Deep Learning architectures for protecting sensitive data; including CNNs, RNNs and transformers. They found that transformer-based architectures were significantly better at identifying long-range dependencies in text than other architectures and therefore would be better suited for analyzing GenAI prompts.

Gupta and Kush [41] developed machine-learning-based DLP systems using gradient boosting classifiers for classifying documents. Kaliappan et al. [42] were able to apply these same concepts into their solution for "SentinelGuard," a browser plugin that utilizes a local version of BERT to classify user input entered into a web form in real-time. SentinelGuard is a client-side solution that protects users' sensitive data from being transmitted outside the browser, which matches the "Privacy By Design" principle of PromptShield.

2.5.2 Named entity recognition (NER) and secure redaction

Once the sensitive information is identified within a dataset, this sensitive information will have to be removed via redaction. The most common method used to identify entities such as names, places and organization are through Named Entity Recognition (NER). Kutbi [43] was able to utilize NER to improve text classification with privacy preservation.

There are also new risks introduced by the act of redaction. Ali et al. [44] warned about timing attacks in NER models. Timing attacks are when an attacker can determine that there is sensitive data present in the data being processed due to the time it takes to process the data. Therefore, secure redaction must be constant time or must protect against side-channel attacks.

Al-sanabani and Iskefiyeli [45] proposed plug-in based design systems for Microsoft Word to encrypt and manage access to sensitive data during the creation of the document, which ensures the sensitive data is protected from the start.

2.5.3 Human-in-the-loop (HITL) security

To ensure the reliability of oversight in generative AI systems, it is necessary for systems to provide an opportunity for people to have agency in their decision-making processes. Effective governance of AI systems, as noted by Ferrari et al., will require systems that are observable, inspectable and modifiable by people; systems should function in a manner that does not allow them to act as an opaque, unmodified entity [46]. Therefore, high-risk decisions (those that include the handling of sensitive information) cannot be completely transferred to automated systems.

PromptShield has HITL (Human-in-the-Loop) functionality in the form of "warn", which alerts users when a borderline/ambiguous piece of content is detected, allowing users to view the borderline/ambiguous content and decide on their own if they want to send the borderline/ambiguous content that PromptShield identified as borderline/ambiguous.

HITL allows the best of both worlds by providing a high degree of efficiency for the automated portion of the system and for humans to be able to oversee portions

of the system; thus, HITL will provide a balance between usability and security as well as foster an environment that promotes informed, proactive management of your organization's data.

2.6 Sector-Specific Implications and Challenges

GenAI security will have drastically varying impacts on different sectors and industries due to their unique regulatory and operational necessities.

2.6.1 Healthcare and pharmaceuticals

Data Privacy is Patient Safety in Health Care. Leong and Gao [47] described how to design a GenAI System for generating Medical Notes, specifically requiring that all Personally Identifiable Information (PII) be strictly redacted to comply with HIPAA. Zhang and Kamel Boulos [48] identified both the opportunities and challenges associated with GenAI in Medicine, including the potential for GenAI Systems to generate false information, potentially dangerous medical advice and leaking identifiable patient information into publicly available GenAI Models.

Robert [49] discussed the Pharmaceutical Industry as being one of the industries most at risk from the theft of Intellectual Property. Robert identified that an AI-Driven Identity Access Management (IAM) can be used to enforce access control policies for drug formulation data to prevent industrial espionage.

2.6.2 IoT and automotive

According to Wang et al. [5], GenAI is used with IoT and they found that it may reduce network traffic, however; it introduces vulnerabilities on edge devices. Shinde and Garikapati [50] examined the use of GenAI in the automotive experience (in-car).

The biggest threats to this area are risks to the safety of both the driver and the passenger in a vehicle; an AI that has been tampered with could potentially distract a driver while he is operating a vehicle, or could cause a vehicle's systems to be manipulated (steering, acceleration/deceleration, etc.). It will be necessary to have

robust and local Data Loss Prevention (DLP) to prevent an unauthorized third party from acquiring information about the vehicle.

2.6.3 Energy and education

In analyzing the application of GenAI and Digital Twins in the energy field, Tomar and Grover [51] identified secure management of data as an essential component to protect critical energy infrastructure from cyber physical threats. The potential exposure of grid topology or operational data in a publicly accessible LLM may also serve as the necessary intelligence for attackers to design targeted attacks against energy infrastructure.

In education, Ng et al. [52], researched school methods of integration of GenAI. They indicated it is important to have policies and guidelines for schools to ensure that no student data will be used to train public AI models and therefore protect the privacy of minors. There are also a number of efforts to develop "AI Literacy" among educators and students so they can safely utilize GenAI tools.

2.6.4 Software development

A qualitative research paper by Klemmer et al. [53], focused on the use of AI assistants in the field of software development, identified potential security risks such as the creation of code that contains vulnerabilities and the unauthorized release of proprietary source code used to train AI models. This further emphasizes the necessity of tools such as PromptShield to review AI generated content prior to its incorporation into an application's codebase.

2.7 Regulatory Frameworks and Governance

The regulatory environment is also changing at a rapid rate to address these GenAI risks and compliance has been a driving force behind the adoption of Data Loss Prevention (DLP).

Compliance is a major driver for DLP adoption. The regulatory landscape is rapidly evolving to address these risks posed by GenAI.

2.7.1 Global AI regulations

The European Union's Artificial Intelligence Act (AI Act), [54] was the world's first broad horizontal regulation of artificial intelligence; it applies a risk-based taxonomy (minimal risk, limited risk, high risk and unacceptable risk) and obligates certain AI systems, including generative ones, to provide transparency in how they operate, disclose when they generate content, and ensure that the training data on which those systems are based, comply with copyright law and data protection law.

The National Institute of Standards & Technology's (NIST) Artificial Intelligence Risk Management Framework (AI RMF 1.0) [55], provides non-regulatory, but widely accepted guidelines for organizations looking to build "trustworthy" AI systems; the document defines practices to manage various AI-related risks and includes reliability, robustness, fairness, privacy, and security. The EU's AI Act, like other regulations, has a governing/mandatory role and the NIST AI RMF is a more operational/organizational governance risk-management tool.

The EU Cyber Resilience Act (CRA) [56] mandates that all products, and their digital elements, whether software, hardware, etc., have to be designed to meet minimum cyber-resilience standards, and if such products incorporate or interact with AI components. Therefore, the CRA directly impacts the ability to deploy GenAI-enabled systems since they must comply with the security-by-design requirements, and also must maintain a vulnerability management program and report any incident to designated authorities throughout their life cycle.

2.7.2 Governance strategies

Ferrari, et al., in [46], identify three structural requirements of governance for generative AI, namely, observability, inspectability, and modifiability, which, in essence, allow regulators and organizations to be able to (i) monitor the operation of generative AI systems in larger infrastructures; (ii) investigate their internal workings and behavior; and (iii) modify system properties, as needed, to ensure that they comply with regulatory requirements.

Yang et al., [57] reviewed GenAI service risk identification and quantification techniques. Wach et al.,[58] reviewed the ethical obligations of organizations, indicating that organizational governance is insufficient as it focuses on legal compliance rather than addressing societal risks such as bias and misinformation. Christodorescu et al.[59], indicated that a “governance gap” has been created by the rapidity of AI’s growth and that there needs to be more flexible policy frameworks to accommodate the rapidly changing capabilities and threats associated with GenAI services.

2.8 Zero Trust Architecture (ZTA) in the AI Era

Zero-Trust Architecture (ZTA), has emerged as a leading framework in contemporary cyber-security, especially with the increasing adoption of AI-based systems and cloud-based applications. Unlike legacy perimeter models, that assume all internal users/devices are trustworthy, both Denzel [60], and Edo et al. [61], have noted that this assumption no longer applies to distributed, dynamic and AI-enabled environments, e.g., remote workers, cloud-based resources and AI-assisted processes.

Denzel’s 2025 survey of ZTNA architecture, outlines it as "never trust, always verify" based on several mechanisms including: least privilege access, explicit validation, micro segmentation and continuous monitoring. The mechanisms used in ZTNA are applicable to many of the same threats to AI systems; i.e., lateral movement after a model server compromise, unauthorized access to model weights or compromised endpoint injection attacks [60].

Similarly, Edo et al., [61], stated zero-trust eliminates the inherent trust present in typical networks by continually verifying identities and validating users, devices and workloads.

Additionally they stated, insider threats are typically the greatest risk for an organization and they support using identity based security measures (such as multi-factor authentication), policy enforcement engines and other identity-centric security controls to ensure that sufficient identity and access management is implemented prior to deployment of a secure enterprise AI solution.

2.9 Conclusion and Research Gap

This comprehensive literature review has demonstrated there is a significant gap in today's security environment.

- Current traditional DLP systems [1], [11] provide protection against loss or unauthorized disclosure of static files but do not have an understanding of the semantic variations of LLMs.
- The native safety features built into native LLMs [37], [36] were designed to protect the model developer from potential liability (to prevent generating hate speech etc.), rather than protecting the users' data.
- Browser-based solutions like SentinelGuard [42] provide some improvement over native LLMs but generally lack the fine-grained and role-based policy enforcements capabilities.
- There are sector-specific requirements for more than general solutions in healthcare, automotive and energy sectors; each requires contextual, role-based controls with knowledge of the specific data types and risk profiles for their respective domains.

Table 2.1 summarizes the comparison between existing approaches and the proposed solution.

Table 2.1: Comparison of existing approaches vs. PromptShield requirements.

Approach	Real-Time	Semantic	Role-Based	Prompt-Aware
Traditional DLP [1], [11]	Yes	No	Limited	No
Hypervisor-Based [15]	Yes	No	No	No
Browser Extension [42]	Yes	Yes	No	Limited
LLM Safety [37], [36]	No	Yes	No	Yes
PromptShield (Proposed)	**Yes**	**Yes**	**Yes**	**Yes**

A middleware solution will be needed to serve as a bridge between the enterprise end-user and the public LLM. The solution must be able to perform real-time semantic classification, apply role-based policies and perform entity-based redactions. PromptShield has been developed to fill the identified research gap, so that GenAI can be safely adopted in enterprise environments while maintaining security and compliance.

3. METHODOLOGY AND FRAMEWORK DESIGN

The PromptShield framework is designed as a modular pipeline consisting of semantic classification, policy evaluation, and content redaction components. The overall system architecture is presented first, followed by a detailed discussion of each module.

3.1 Overview of the PromptShield Architecture

PromptShield is designed as a lightweight, client side middleware that operates between the user interface and the external generative AI model endpoint. Figure 3.1 illustrates the end to end workflow of PromptShield.

The system is composed of three primary components:

- **Semantic Classifier:** Determines the sensitivity level of each user prompt.
- **Policy Engine:** Applies organization and role specific rules to decide whether a prompt should be allowed, warned, redacted, or blocked.
- **Redactor:** Performs inline entity level masking of sensitive data.

3.2 Threat Model

The rapid adoption of generative AI tools within enterprise environments has introduced a new class of confidentiality risks rooted not in malicious intent, but in everyday operational workflows. Employees increasingly rely on external large language models to assist with writing, summarizing, translation, idea generation, and document editing. While this improves productivity, it also creates a subtle but highly consequential threat landscape such that sensitive internal information may be unknowingly embedded in requests sent to third-party providers. The PromptShield framework is designed to address this operational reality by understanding how accidental leakage occurs, why it occurs, and how it propagates through organizational and technical systems.

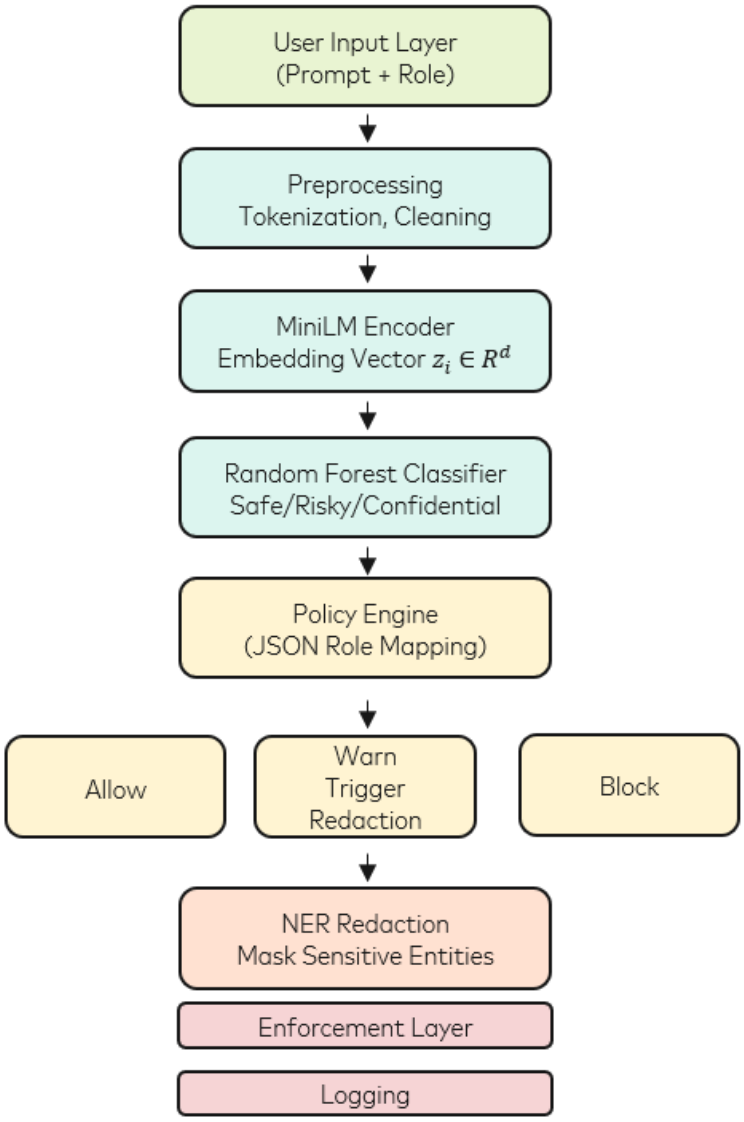

Figure 3.1: PromptShield end-to-end workflow.

In a typical organization, internal users handle a wide spectrum of sensitive data. HR personnel work with performance evaluations, salary adjustments, internal complaints, recruitment insights, and employee well-being reports. Finance departments review budgets, forecasts, supplier contracts, invoice disputes and profitability metrics. R&D teams handle confidential prototypes, product design documents, model architectures or technical implementation details. Across all of these functions, users frequently consult generative AI tools to improve clarity, prepare communications or generate draft content. This introduces a direct channel through which confidential material can unintentionally leave the enterprise boundary.

Leakage may occur in multiple forms. The most obvious is the explicit copy-pasting of sensitive text. When a user pastes a paragraph from an internal report, an email conversation or meeting into a prompt, all of the embedded confidential details including names, performance comments, salaries, customer identifiers, or numerical forecasts are exposed. However, explicit leakage is only one dimension of the threat. Also there is contextual leakage, where users rewrite or describe sensitive scenarios in their own words. For instance, a sentence like "Can you rewrite my feedback for an employee who has struggled to meet quarterly targets and may face a reduced bonus?" carries HR sensitive content even if no names are included. Indirect leakage is often harder to detect because sensitivity emerges from context, relationships, roles and implications rather than explicit keywords.

A third category is structural leakage. Many users paste tables, spreadsheets or bullet lists containing financial figures, operational KPIs, customer revenue contributions, internal timelines or project dependencies. Even if the user believes these are "just numbers," they often constitute confidential commercial information. These patterns highlight that leakage is not limited to recognized entity names but extends to entire information structures.

Accidental disclosure is also shaped by human factors and workplace dynamics. Even employees with awareness of data classification rules can assume that anonymizing or partially obscuring text is sufficient. Time pressure encourages shortcut behaviors such that instead of manually rewriting a paragraph, users rely on an LLM. In cross functional teams, members may not fully understand what constitutes sensitive information outside their domain. For example, Finance staff may unknowingly reference HR related details, or technical staff may leak strategic product information while requesting help drafting documentation. These behavioral drivers make accidental exposure a predictable and recurring risk.

If a sensitive prompt gets past detection and reaches a third-party LLM provider, the organization can no longer control what happens to that data. Providers often log prompts to help with diagnostics or quality improvements. Even if the data is not used for training, it might still be kept for a short time or stored in different data centers

around the world. This can create compliance risks under the General Data Protection Regulation (GDPR) and Turkish Personal Data Protection Law (KVKK), contractual risks for customer data, and concerns about the confidentiality of trade secrets and internal decisions. The main risk here is not a direct attack, but the provider storing data in ways that may violate company policies or legal requirements.

Besides risks linked to behavior and workflow, the way we use language can also create security threats. Sensitive information might be hidden in indirect wording, unclear references, relationships, tone, timing, or job descriptions. For example, if we think about an HR note that says, "The employee who reports directly to me has been underperforming since Q3, and we may reconsider his role". It reveals performance management details even if the employee is not named. Spotting these subtle signs is important for accurately understanding and modeling threats.

PromptShield must protect against several failure modes arising from this environment. These failures include misclassifying borderline prompts, missing indirect references, mismatches between user roles and allowed content, failing to fully remove sensitive information, and relying too much on users to make the right call. Also, because language can vary, the system might get confused. Two prompts that mean the same thing can look different, and users might reword sensitive content to make it seem harmless.

Depending on the nature of the data leaked, there are many types of damage that can be done to a company. For example, if confidential personnel information (i.e., Human Resources) leaks, an employer may have violated employees' rights to privacy as well as damaged employee trust. Additionally, a breach of employee data may also subject the employer to possible litigation or other claims. A leak of sensitive financial information about a company may expose to competitors, customers, and partners negotiation strategies, budget and forecast information, etc. A company's exposure of its customers' personal identifiable information may cause the company to lose the confidence of its customers, breach confidentiality agreements with customers, and/or destroy the relationship between the company and its customers. A leak of Research & Development (R&D) data could potentially decrease the competitive advantage of the

leaking company by exposing its research strategy to competitors. In addition, a leak of R&D data may expose the leaking company's intellectual property (IP) for which they may have spent years developing, protecting, and monetizing. Leaks of operations or strategic data may harm the leaking company's market position, and/or make the leaking company appear vulnerable from within. In any event, the external LLM provider will always pose a potential threat once company data leaves the company.

The core concern in this threat model from a perspective within the CIA triad (confidentiality, Integrity, Availability) is confidentiality. The goal of the confidentiality is to ensure that sensitive information is not leaving the company's enterprise boundaries. With respect to integrity, it is important that the classification and redaction systems are consistent in operation regardless of the language being used and operate independently of other divisions. From an availability perspective, the threat model seeks to prevent PromptShield from adding unnecessary workflow drag on user activity, thereby causing users to develop "work around" techniques for using unsafe versions of generative A.I. tools outside of supervision.

The PromptShield threat model embodies all of the ways unintended leaks occur in an enterprise environment as well as the way those unintended leaks are created through a mix of human actions, workflow pressures and third party providers along with linguistic ambiguities. The overall understanding of the threat model provides context to all architectural elements of PromptShield, including its classifier structure, policy enforcement, redact mechanism and logging strategy. The intent of PromptShield is not to mitigate attacks by malicious users, but to create a product that mirrors enterprise usage patterns in order to prevent the most frequent and most damaging types of accidental data exposures.

3.3 Semantic Classification Pipeline

Semantic classification is the first analysis module of PromptShield. It initially identifies the sensitivity contained in a user's input prompt. The input prompt is mapped to one of three predefined classifications: Safe, Risky, or Confidential. As mentioned above, the semantic classification occurs before any policy is enforced or redacted and is the first part of the overall decision-making process.

To formalize the process, consider an incoming prompt x_i. Before semantic interpretation is performed, the text undergoes a lightweight preprocessing stage that standardizes its structure through tokenization, lowercasing, and removal of simple artifacts such as excessive whitespace. These operations normalize stylistic variation without altering the semantic content of the prompt.

PromptShield then encodes the normalized prompt using a pretrained MiniLM sentence transformer (all-MiniLM-L6-v2), which maps the input into a d-dimensional semantic embedding. Formally,

$$E(x_i) \in \mathbb{R}^d \tag{3.1}$$

where $E(\cdot)$ denotes the embedding function.

The resulting vector is passed to a Random Forest classifier trained on labeled embeddings. Each decision tree evaluates the encoded prompt independently, and the final sensitivity label $y_i \in \{\text{Safe}, \text{Risky}, \text{Confidential}\}$ is selected via majority vote:

$$y_i = f(x_i) = \text{mode}\big(T_1(E(x_i)), T_2(E(x_i)), \ldots, T_K(E(x_i))\big) \tag{3.2}$$

where $T_k(\cdot)$ denotes the k-th tree in the ensemble. The random forest approach helps improve the robustness of the classification task since the individual decision trees can evaluate the encoded input prompt based on their own independent assessments of the input prompt. Additionally, if the input prompt contains ambiguous language or domain-specific terminology, the random forest approach will provide a more accurate sensitivity classification than a single decision tree would be able to classify based solely on its own independent evaluation of the input prompt.

Finally, the classification step uses a Random Forest model that operates on MiniLM derived embeddings. MiniLM is used to generate compact but semantically rich vector representations that retain contextual information about the input prompt, and therefore, enable the system to make decisions regarding the semantic classification of the input prompt in real time within an enterprise workflow. Random Forest was chosen over linear models due to the fact that they are able to create non-linear decision boundaries, capture complex relationships between the terms in the input prompt, and remain robust when dealing with ambiguous or partially anonymized input data. Furthermore, because each decision tree evaluates the embedded input

prompt independently, the random forest approach helps to mitigate overfitting and increase the overall stability of the classification process when dealing with different input prompt structures. While the computational overhead of the classification step increases with the number of trees K in the ensemble, the overall complexity is still relatively small because the embedding generation process typically dominates the overall processing time and the tree evaluations are easily parallelizable. Algorithm 1 describes the entire classification process.

Algorithm 1 Prompt Classification with Random Forest

Require: Prompt x_i

1: Encode x_i using MiniLM to obtain $E(x_i) \in \mathbb{R}^d$
2: Feed $E(x_i)$ into each tree T_k in the Random Forest
3: Collect predictions: $\mathscr{Y} = \{T_1(E(x_i)), \ldots, T_K(E(x_i))\}$
4: Predict label: $y_i = \text{mode}(\mathscr{Y})$
5: **return** $y_i \in \{\text{Safe}, \text{Risky}, \text{Confidential}\}$

While the classifier determines the semantic sensitivity of the input prompt, its output is not provided directly to the user. Instead, the predicted sensitivity level is forwarded to the role-based policy engine of PromptShield, where it is compared to the organizational role of the user. Ultimately, the decision of the overall system is made by combining the results of both the semantic classification and the role-based policy engine.

3.4 Policy Engine Design

PromptShield's policy engine takes the semantic classifier's classification output and maps that into a real-time enforcement action that aligns with the organization's expectations. The classifier produces a sensitivity category for the given prompt; however the policy engine defines how PromptShield will act based upon that sensitivity category. Unlike traditional Data Loss Prevention (DLP) mechanisms PromptShield provides enforcement based upon a fine-grained, role-based basis per prompt.

In PromptShield, the decision making process is formalized by associating each role with a triplet of sensitivity sets:

$$(A_r, W_r, B_r) \tag{3.3}$$

where A_r represents the sensitivity levels automatically allowed for role r, W_r lists the levels that require a warning and redaction preview, and B_r defines the levels that must be blocked entirely. This structure enables flexible policy definition without modifying the underlying classifier.

For example, a policy may specify the following:

- HR + Confidential $\rightarrow$ Block
- Finance + Risky $\rightarrow$ Warn
- R&D + Safe $\rightarrow$ Allow

These mappings provide intuitive, role specific rules that administrators can update independently of the machine learning components.

3.4.1 Role-sensitivity policy model

Given a predicted sensitivity label y_i and a user role r, the enforcement action is defined by the decision function:

$$P(r, y_i) = \begin{cases} \text{Block}, & \text{if } y_i \in B_r, \\ \text{Warn}, & \text{if } y_i \in W_r, \\ \text{Allow}, & \text{if } y_i \in A_r. \end{cases} \tag{3.4}$$

This formulation captures all enforcement behavior using a single, interpretable rule. A benefit of this model is that with a single update of the policy sets (which may include new roles or an updated organizational structure) new roles or organizational structure will not require retraining the classification algorithm.

3.4.2 JSON-based policy definition schema

Policy configurations are externally defined via a JSON file and therefore may be adjusted at run time without modifying the underlying code of the application. Below

is an example of how a role-sensitivity relationship would be encoded into the structure for the policy set:

```
{
  "HR": {
    "Safe": "Allow",
    "Risky": "Warn",
    "Confidential": "Warn"
  },
  "Finance": {
    "Safe": "Allow",
    "Risky": "Allow",
    "Confidential": "Block"
  },
  "R&D": {
    "Safe": "Allow",
    "Risky": "Block",
    "Confidential": "Block"
  }
}
```

This format not only simplifies the configuration process but also allows organizations to maintain policy files separately as part of their compliance documentation.

Algorithm 2 illustrates how the system takes a classifiers output and transforms that into a concrete enforcement action by the system. Policy tables are essentially a list of (A_r, W_r, B_r) values, with each value being represented as a single entry (Allow/Warn/Block).

PromptShield's architecture provides for predictable and understandable enforcement of its policy by allowing separation of its classifier from its policy logic, enabling it to allow an organization to adjust policies quickly based on changing regulations, guidance, or internal risk management decisions.

Algorithm 2 Policy-Based Enforcement Decision

Require: Sensitivity label y, user role r, policy table P

$a \leftarrow P[r][y]$
if $a =$ Allow **then**
    return Allow
else if $a =$ Warn **then**
    return Warn (with optional redaction)
else
    return Block
end if

3.5 Entity-Aware Redaction Module

The redaction module acts as a second layer of protection (a secondary barrier) to activate when a request is deemed sensitive enough that it needs some form of user interaction (i.e., editing), but not sensitive enough to block immediately. In doing so, PromptShield can provide the level of security that an organization requires while at the same time protecting user autonomy and workflow efficiency. The system does not delete a query which could be potentially useful; instead, it allows the user to edit or clean up the query prior to submitting it externally.

Once the prompt has completed the policy evaluation stage, the prompt is then sent to the Redact Layer of PromptShield if the prompt's sensitivity label falls into the "Warn" Category based on the user's role. When the prompt does fall into this category, PromptShield will seek to identify those specific pieces of text which are likely to include personally identifiable information, financial information, or organizationally sensitive information; in other words, those pieces of text which could contain sensitive data. Once PromptShield identifies those pieces of text, it will redact them so as to avoid unintended exposure of sensitive data via reformulation of the prompt.

3.5.1 Named entity recognition for sensitive content detection

The use of the SpaCy English NER pipeline by PromptShield is based on its demonstrated ability to be able to detect entities with high accuracy in general purpose entity recognition applications; and secondly, due to the fact that it has the capability to provide real-time processing for the end-user's prompt input, enabling PromptShield to perform both fast and accurate entity recognition. This approach enable the highest

level of accuracy while ensuring the lowest levels of latency during real-time prompt intercepts.

Given that we will let x_i represent the user's input prompt, when this prompt is processed by the NER model, the prompt is broken down into a number of entity span annotations labeled as l_e. The focus of this module is on a predefined list of entity types most commonly used to express data sensitivity in enterprise environments, such as:

- PERSON: individual names or identifiers
- ORG: company or department names
- GPE: geopolitical locations
- DATE: time expressions indicating specific dates or periods
- LOC: physical locations
- MONEY: monetary values or salary figures

The categories listed above have been chosen due to their presence in the majority of HR, Finance and R&D contexts domains in which data by accident is likely to be exposed. In this way, by capturing these types of entities in addition to maintaining the surrounding context of the input prompt, the system can capture a significant amount of sensitive content.

For every entity, e, identified in x_i with label l_e, the module will replace the corresponding text span with a standardized placeholder (i.e. redaction), so as to preserve structural coherency of the redacted text while also eliminating all identifiable information.

The format of the redaction placeholder is defined by the system as [REDACTED_l_e]. The name of the deleted data will be included in the placeholder so that users can clearly see which parts of their input have been redacted; while also maintaining the same semantic flow as the original request/prompt.

3.5.2 Redaction algorithm

The complete redaction procedure is outlined in Algorithm 3. The algorithm iterates through all NER-identified entities and applies masking only to those that fall within the configured redaction set. This selective approach enables organizations to fine-tune which entity types warrant masking based on regulatory, contractual, or internal requirements.

Algorithm 3 Entity-Aware Redaction Process

Require: Prompt x_i
1: Apply NER model to extract entity spans
2: **for** each entity e with label l_e **do**
3: **if** l_e belongs to the redaction entity set **then**
4: Replace e with `[REDACTED_`l_e`]`
5: **end if**
6: **end for**
7: **return** Redacted prompt

3.5.3 Interaction with the user and policy engine

Upon redaction, the system will display the redacted information and send the prompt to the end-user for review rather than sending it directly to the target LLM. In this manner, the system provides user control; the user can either accept the revised prompt, revise the original request or opt out from using PromptShield. Additionally, by showing the user the content that has been removed and why it was deleted, PromptShield continues to be transparent and compliant to an organization's data handling policies.

In addition to promoting user comfort through usability, the redaction module operates as an intermediary function that allows users to balance security needs with their need to be able to use the system effectively. The redaction module does this by presenting a "middle ground" that exists between completely allowing a potential risk of exposure and preventing all access to a potential risk of exposure. Therefore, the layered design is reflective of PromptShield's overall philosophy of being proactive in preventing data leaks, yet also minimizing unnecessary workflow friction on the part of the users within the enterprise.

3.6 Dataset Construction and Preparation

To assess and validate the proposed PromptShield framework, a dataset reflecting enterprise prompt use patterns in a realistic manner had to be developed. However, collecting a real-world prompt database from inside an organization is impossible from both a legal and ethical perspective because many of these prompts contain very confidential information protected through various forms of confidentiality agreements, contracts and data protection laws like GDPR and KVKK. Therefore, the dataset employed in this research was synthesized according to a predetermined process, emulating enterprise workflow's semantic, syntactic and contextual features; however, the synthesis process avoided including any personally identifiable data, financial data or proprietary data of any individual or company.

The dataset included 900 synthetic prompts that modeled enterprise internal communications and work flow processes across the three primary business categories of Human Resource (HR) and Finance and Research and Development (R&D). These categories have been identified as of the highest risk categories of unintentional data breaches in enterprise environments. HR prompts generally contain employee evaluations, interview notes, performance reviews and salary-related communications all of which must be kept confidential. Finance prompts cover budget planning, customer invoicing, customer terms and conditions and financial projection communications that may be considered commercially sensitive even after all identifiers are removed. The R&D category models technical descriptions, architecture documents, prototype discussions and early stage innovation pipeline communications which are the majority of an organizations critical intellectual property.

The synthetic prompts were not created randomly instead, they were generated using patterns observed in real enterprise communication styles. Each domain was mapped to a set of common prompt templates based on realistic work scenarios. For HR, prompts include rewriting feedback, drafting performance summaries or preparing internal communication notes. Finance prompts include requests to clarify budget statements, summarize invoice discrepancies or improve

communication with a vendor regarding payment delays. R&D prompts emulate internal design discussions, refactoring explanations, architecture clarifications or technical documentation revision tasks. These templates were constructed based on publicly available enterprise writing examples, anonymized role descriptions and generalizable domain specific linguistic structures. By design, they mimic how employees naturally use generative AI tools to refine their writing or request clarity while avoiding any real world data leakage.

Each prompt was assigned one of three sensitivity labels, which are Safe, Risky or Confidential, corresponding to the policy levels enforced by PromptShield. The Safe class contains prompts that carry no sensitive information and resemble benign workplace queries such as grammar correction or neutral content summarization. The Risky class includes prompts that do not explicitly include sensitive entities but contain contextual cues that suggest potential sensitivity. These prompts represent ambiguous or borderline cases, such as discussing project challenges, referring to performance issues without naming individuals or hinting at financial pressures. The Confidential class contains prompts that, if sent to an external provider, would violate organizational policy due to the presence of identifiable HR descriptions, financial metrics, customer specific information or technical IP details. This categorization allows the classifier to differentiate not only between clearly safe and clearly unsafe cases but also between subtle variations in semantic content where the risk is contextual rather than explicit.

One other concern in the construction of the dataset is the need for a balanced number of prompts across domains and sensitivity classifications.

While synthetic, the dataset is intentionally crafted to reflect high risk enterprise scenarios in a controlled, privacy preserving manner. It operationalizes real world leakage patterns explicit copy paste, contextual leakage, structural leakage and cross domain role confusion without ever incorporating real sensitive data. This makes it both suitable for academic evaluation and representative enough to model the practical challenges that PromptShield aims to solve in actual enterprise environments.

The distribution of training prompts across roles and sensitivity labels is provided in Table 3.1.

Table 3.1: Prompt dataset distribution by role and sensitivity.

Role	Safe	Risky	Confidential
HR	120	90	90
Finance	140	70	70
R&D	130	100	90
Total	**390**	**260**	**250**

All prompts were encoded with the previously described MiniLM model, generating fixed size vector representations optimized for training. A Random Forest classifier was chosen for its interpretability, strong performance on moderate datasets, and computational efficiency, qualities that enable real time deployment on the client side.

3.7 Implementation Details

This section presents the practical implementation of the PromptShield framework. While the previous subsections introduced the architectural and algorithmic foundations, the details below describe how these components were realized in Python using a modular, lightweight design suitable for real time client side operation.

3.7.1 Development environment

PromptShield was developed using a minimalistic environment with a small number of required packages. It uses the MiniLM sentence-transformer model for semantic embeddings, and a NER model from spaCy for identifying entities in the input, along with a Random Forest Classifier from scikit-learn. The web interface and overall flow of the PromptShield application are handled by a minimalist Flask application that loads all models upon application start-up to eliminate latency and enhance performance.

The key libraries included in this thesis are as follows:

- sentence-transformers (MiniLM sentence embeddings)
- scikit-learn (Random Forest Classifier and LabelEncoder)
- spaCy (Named Entity Recognition Model)
- Flask (Web Interface)

3.7.2 System integration and execution flow

The workflow of the runtime process is located in the file app.py where it integrates the classification logic, policy analysis logic, and the logic for performing redactions based on the user's selection. The workflow consists of the following four steps when the user submits a prompt:

1. Retrieve the selected prompt and the selected role.
2. Classify the submitted prompt to determine its sensitivity level.
3. Apply the role-based policy logic to the sensitivity level determined in step #2 and the selected role.
4. If the decision made in step #3 is "warn" then perform redaction on the original prompt.
5. Return the result of the analysis ("Allow", "Warn", or "Block") to the user.

A simplified illustration of the control flow is presented in Algorithm 4.

Algorithm 4 Simplified control flow of the PromptShield decision pipeline

$label \leftarrow classify\_prompt(prompt)$
$decision \leftarrow check\_policy(role, label)$
if $decision =$ Warn **then**
    $redacted\_prompt \leftarrow redact\_prompt(prompt)$
end if

For clarity and readability, the complete source code implementation is provided in Appendix A.

3.7.3 NLP classifier implementation

The nlp_classifier.py file contains the core logic for the classifier. It loads the MiniLM embedder, Random Forest model, and LabelEncoder from serialized pickle files, then uses each prompt to create an embedding vector which is classified as either one of the three sensitivity categories or unknown.

The main processing steps of the NLP classifier are summarized in Algorithm 5.

Algorithm 5 NLP classifier processing steps

1: Encode the input prompt into a vector representation
2: Apply the trained classifier to predict a sensitivity label
3: Decode the predicted label into its semantic category

The lightweight design of the MiniLM embedding model and the Random Forest classifier allows the classification process to be completed within milliseconds, enabling real-time prompt analysis.

3.7.4 Role-based policy engine

Role-based policies are defined by a dictionary of dictionaries in policy_engine.py that map roles to the actions they are allowed to perform. The dictionary is a constant time lookup and will allow for rapid expansion: action = POLICY[role][label].

3.7.5 Entity-aware redaction

Using spaCy's English NER model, sensitive entities such as PERSON, ORG, DATE, LOC, GPE, and MONEY are identified and replaced with structured placeholder values of the form: [REDACTED_l_e].

This maintains the structural integrity of the input while effectively eliminating all of the sensitive information.

3.7.6 User interface

PromptShield includes a very simple Flask-based web interface to allow users to specify their role, enter prompts, and see the outcome of each prompt in terms of Allow/Warn/Block. The goal of this web interface is to visually represent PromptShield's complete function in a way that is both easy to use and allows users to test and experiment with the prompt evaluation process.

Users interact with the web interface by first choosing their organizational role (i.e. HR, finance, R&D), and then they enter any type of natural language string for evaluation. After entering a string, the web interface evaluates the entered string via the

semantic classifier and the role based policy engine. Once evaluated, the web interface provides the user with the result of the evaluation as either Allow, Warn, or Block.

In cases where the results in a Warn status, the web interface shows a split-screen comparison of the original string and the redacted string. Displaying the redacted version is done intentionally to increase user knowledge and to build user confidence by demonstrating to users the actual identification and masking of sensitive information. By showing the user the redacted version of the string, the web interface helps support explainability and demonstrates to the user what portion(s) of the inputted string caused the Warn status without completely blocking the user from the rest of the application.

3.7.7 Logging and audit trail

PromptShield logs each prompt, so all prompts will have a trail and can be reviewed at a later time. Every prompt has a log entry that includes:

- the timestamp when the prompt was requested,
- who made the request,
- what sensitivity level the prompt was labeled by the semantic classifier,
- what enforcement action the policy engine took against the prompt.

PromptShield's logging system allows security and compliance teams to audit the history of the prompts used after they were requested, which confirms that PromptShield enforced policies correctly. It also is useful for debugging and finding errors, if something went wrong because the classification of the prompt was incorrect or an unexpected policy enforcement occurred. The log entries could also be used to improve the classifier and policy enforcement rules based on common false positive classifications, or classifications that are borderline.

Although the current version of PromptShield is designed to store log entries locally for the purpose of testing and development, the system is flexible enough to accommodate storing log entries remotely in a SIEM platform, or other central logging systems.

3.7.8 Performance considerations

In order to achieve real-time usability, PromptShield includes a number of performance enhancements:

- The models are loaded only once when PromptShield starts up,
- Embedding and classification are performed using lightweight models,
- Only "Warn" cases require redaction to occur,
- There are no additional server overheads introduced due to the simplicity of the Flask routes.

Overall, the entire pipeline (embedding, classification, policy evaluation, optional redaction, and User Interface (UI) rendering) takes less than 150ms to run on commodity hardware.

4. PERFORMANCE EVALUATION

This chapter presents a comprehensive evaluation of the PromptShield framework. The objective of the evaluation is to measure the effectiveness of the semantic classifier, assess the behavior of the role-based policy engine, validate the consistency of the redaction mechanism, and determine whether the overall pipeline satisfies real time performance requirements. All experiments were conducted on a standard laptop environment without GPU acceleration to reflect a realistic client side deployment scenario.

In addition to overall accuracy, the evaluation framework adopts a multi-metric approach commonly used in text classification research. Because the sensitivity detection problem involves three classes of unequal size with different operational importance, the performance of the classifier is assessed using precision, recall, and F1-scores for each class. These metrics provide a more reliable measure of the behavior of the model in a class imbalance and allow us to distinguish between the system's ability to detect highly sensitive content and its tendency to apply conservative predictions in ambiguous cases. The combined analysis of accuracy, class-level F1 performance, and qualitative error patterns offers a comprehensive view of the strengths and limitations of the model.

4.1 Evaluation Setup

The evaluation relies on the synthetic enterprise prompt dataset described in Chapter 3.3. The dataset consists of 900 prompts used for training and an independent test set of 435 previously unseen prompts used for validation. The test set preserves the role distribution of the training prompts, ensuring that HR, Finance, and R&D use cases are equally represented.

All embedding and classification operations were performed using the pretrained MiniLM encoder and the Random Forest model trained earlier. The performance

metrics reported here include accuracy, confusion matrices, F1 performance, qualitative error analysis, and end to end system latency.

4.1.1 Dataset description

Dataset that described in Chapter 3.3 contains realistic enterprise-style prompts from 3 different areas of expertise and has been organized to represent how data is typically distributed by sensitivity. The training and test datasets were completely separate so as to avoid the possibility of duplicate use.

4.1.2 Evaluation components

The evaluation pipeline consists of:

- sentence embedding via MiniLM,
- sensitivity classification via Random Forest,
- role-based policy mapping,
- optional redaction through spaCy NER.

4.2 Classifier Accuracy

Table 4.1 reports the total accuracy for the classifier over the test dataset; with an accuracy of 72.92% it shows the classifier's ability to generalize to all prompts that have never been used in training.

Table 4.1: Overall classification accuracy on the test set.

Metric	Value
Accuracy	72.92%
Test Samples	435

Although, the classifiers' accuracy is useful information, it doesn't provide enough detail about how well the classifier performed. In order to have a better understanding of the classifier's performance, Figure 4.1 displays the confusion matrix created from the results of the test set prompts.

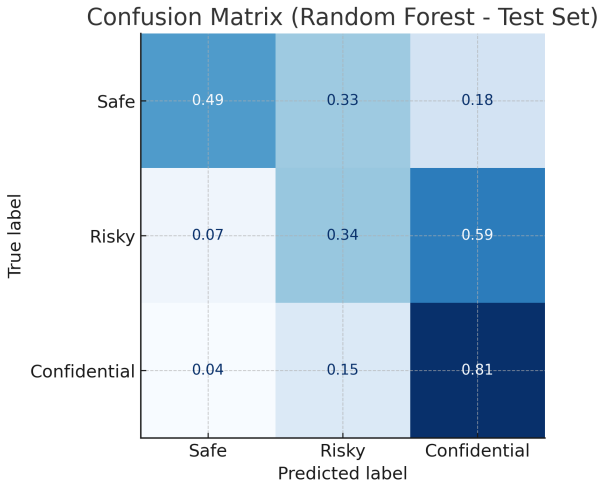

Figure 4.1: Confusion matrix of predicted vs. true sensitivity labels.

The classifier performs strongly on Safe and Confidential prompts, while most misclassifications occur between Safe and Risky. These errors are typically associated with prompts that contain ambiguous or indirect references to sensitive content such as salary, internal processes, or high-level project details.

Because the confusion matrix is standardized to the row, each element in the table shows what percent of samples from the actual category are predicted into the given category. By standardizing to the row, this gives us an easier to interpret version of the matrix when there is a large difference in the number of samples in each class. The results show that most errors are made between Safe and Risky, and that the Confidential has been well separated.

4.3 Per-Class Precision, Recall, and F1-Score

While overall accuracy may provide some general insight regarding the quality of a classifier's predictions, the detail required to determine how well it can differentiate between Safe, Risky and Confidential types of content, is better addressed with a more detailed analysis. Therefore, the per-class precision, recall, and F1-score for the three classes are presented in Table 4.2 and calculated over the held out test set.

Overall accuracy of the classifier is useful, however the overall accuracy provides little insight about how well the classifier performs relative to each class. Therefore,

Table 4.2: Per-class precision, recall, and F1-scores for the classifier.

Class	Precision	Recall	F1-Score
Confidential	0.8842	0.8112	0.8462
Risky	0.1667	0.3415	0.2240
Safe	0.6304	0.4754	0.5421
Weighted Avg.	0.7824	0.7211	0.7462

the per-class accuracy provides much more insight about the overall performance of the classifier. It appears that the model performs very well on the Confidential class, providing an F1-score of .8462. The importance of this high performance on Confidential prompts should not be overlooked. The purpose of a DLP system is to protect against potential unauthorized transmission of highly sensitive content to a generative AI model. The high recall value (.8112) shows that the system is successful in identifying most of the sensitive content; the high precision value (.8842) indicates that the system rarely labels non-sensitive content as Confidential. Overall, the combination of the recall and precision values clearly show that PromptShield would be suitable for use in high-risk environments, i.e., the cost of a missed detection is significantly greater than that of a false positive.

Conversely, the Safe category produces a moderate level of accuracy (F1 = .5421). The reason for this moderate accuracy is a conservative bias of the model. When the model is unsure about what to classify a prompt as, it tends to place the prompt in the highest possible sensitivity category. While this is a characteristic of the model that may cause some users frustration in terms of usability due to excessive classification of benign content, this type of conservative behavior is acceptable, if not desired from a security perspective. This is because false positives (i.e., incorrectly labeling benign content as Sensitive) will never lead to data leakage. Rather, they will create minimal usability barriers that can be easily alleviated via policy adjustments and UI feedback. The recall value for Safe prompts (.4754) indicates that nearly half of all benign prompts are correctly classified as such; the other half are placed in higher-risk categories. This conservative behavior is consistent with the design philosophy of safety-focused text classifiers commonly used in modern enterprise applications.

Finally, the Risky class produces an F1-score of .2240, the lowest score of the three classes. As expected, the low F1-score for Risky is a direct result of

two factors. First, mid-sensitivity prompts do not have clear lexical indicators, unlike Confidential prompts that frequently include explicit personal identifiable information, financial values, or proprietary information. Risky prompts generally express ambiguous or indirect references to internal processes, strategic initiatives, or moderately sensitive business insights. Due to their subtlety, these weak cues are difficult for standard machine learning models, including embedding-based models trained on general-purpose text, to accurately identify. Second, as we did not prioritize Risky during initial development (and thus had a smaller number of Risky prompts available in the dataset), the effects of misclassifications were amplified by the large disparity in the number of Risky and other classes in the dataset. In cases where the dataset is imbalanced, minority classes are generally less separable from the majority class due to poor representation of minority class members in the training data. This is a well-documented issue in literature related to multi-class text classification.

Even though there was a difference in performance on the Risky Class, the system has value in terms of policy enforcement. When it is uncertain whether or not the input is at risk (i.e., the classifier can't make up its mind), the role-based policy engine will choose to take a conservative action, like making a Warn decision, instead of letting the user enter their prompt.

This multi-layered design is what protects against possible data leakage, when the classifier is unsure about an input.

Lastly, the weighted average F1 score of .7462 provides an honest representation of how the system would perform in the real world. Because Confidential Prompts are the dominant type of prompt in enterprise environments and are also the most sensitive, the weighted measure provides a better representation of how the system will operate in a real-world environment, compared to a macro averaged score. Macro averaged scores provide a false sense of balance because they give too much weight to the smaller and more ambiguous categories. As a result, the weighted F1 score is the best metric to use to evaluate PromptShield in a security based deployment scenario.

In total, these results demonstrate that the classifier is able to distinguish between risky content and behave conservatively when it cannot be sure if an input contains risky

content. These behaviors align well with the goals of developing a prompt level DLP system.

4.4 Error Analysis

To better grasp the limits on the use of this classifier in practice, an error analysis was performed, based on the incorrectly classified samples from the test set. Most of the errors were between Safe and Risky. The large number of errors between Safe and Risky indicates that there is considerable ambiguity regarding the semantics of mid-sensitivity content. Three recurring patterns can be identified in the misclassifications:

- **Implicit Sensitivity Signals:** Many of the prompts that were classified as Risky have indirect (and therefore implicit) references to salary structures, operational workflow, and strategic intent. In addition to lacking clear lexical indicators of sensitivity, the model treats these types of examples as Safe. This type of behavior is common in the behavior of embedding-based classifiers when the sensitivity of the example is dependent upon the context of the example versus an entity-dependent sensitivity signal.

- **Boundary Between Safe and Confidential:** A few Safe examples were incorrectly classified as Confidential due to the presence of terminology commonly found in examples of Confidential content (e.g., "employee performance", "budget planning", "user data access"). Although these examples had no explicit identifiers of Confidential content, their semantic similarity to examples of Confidential content led the model to classify these examples conservatively. This reflects the model's safety-bias, which will generally prefer to assign higher-sensitivity classifications to uncertain examples.

- **Sparse Representation in the Risky Class:** Since the Risky class contains fewer examples than the other two classes in the dataset, the model has less opportunity to learn the subtle patterns that define the differences between Risky and both Safe and Confidential examples. Therefore, some moderately-sensitive examples are

classified into one of the adjacent categories. This is a common occurrence in imbalanced multi-class classification problems.

In summary, the combination of the per-class metrics and error analysis provides a comprehensive view of the classifier's behavior over a range of sensitivities. The classifier shows high reliability at identifying examples of highly-sensitive content, and a conservative bias when evaluating borderline examples. Both behaviors are desirable within the scope of prompt-level data-loss prevention systems, where the costs associated with false negatives far exceed those associated with false positives. The error patterns observed above suggest that the classifier's limitations are primarily caused by the ambiguity of the semantics of the content being classified, as opposed to structural flaws in the classifier itself. We next consider the interaction between the classifier's predictions and the role-based policy enforcement engine to identify how the classifier influences the enforcement decisions made in each of the HR, Finance, and R&D domains.

Even though the classifier performs well in sensitive (Confidential) prompt cases and has an appropriately conservative behavior when there is uncertainty; nonetheless, we have identified several limitations as we evaluated the classifier.

The first limitation is that the dataset contains class imbalance because the Risky category has significantly fewer examples than both the Confidential and Safe categories. Thus, it will be much more difficult for the model to distinguish between lower sensitivity (mid-sensitivity) prompts with more granularity.

The second limitation is that the majority of Risky sample prompts include either implicit or contextual indicators of sensitivity that make the use of standard (general purpose) sentence embedding methods to separate these prompts challenging.

Finally, since the classifier uses a pre-trained Named Entity Recognition (NER) model, it may not account for organization specific terms used by organizations and therefore, it may result in under or over redaction occasionally. Therefore, these limitations illustrate potential areas of interest for future research including data set expansion,

domain adaptive representation learning and multi-stage classification architectures capable of capturing more subtle sensitivity cues.

4.5 Policy Enforcement Behavior

The classifier outputs serve as inputs to the role-based policy engine, which maps each predicted label to an enforcement action (Allow, Warn, or Block). Figure 4.2 illustrates the distribution of enforcement actions across HR, Finance, and R&D roles.

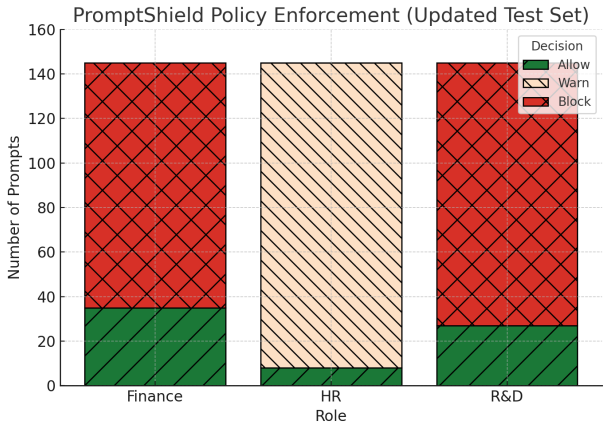

Figure 4.2: Distribution of enforcement actions across roles.

Observed trends include:

- HR receives the highest number of Warn outcomes due to frequent interactions with personal identifiers.
- Finance encounters a larger proportion of Allow outcomes because its policy permits both Safe and Risky prompts.
- R&D blocks most Risky and Confidential prompts to minimize Intellectual Property leakage.

4.6 Redaction Accuracy

When a Prompt receives a warn from the Redact system, the redaction will use spaCy's named entity recognition (NER) to remove sensitive spans. In qualitative analysis of redacted output it is apparent that sensitive entities have been removed with no loss to flow or readability in the original sentences.

The redaction effectively removes individual names, dates and organizational names that could cause a potential leakage. Each sentence maintains its original structure, so the amount of contextual information being communicated by the prompt for user interaction has not changed.

4.7 Qualitative Prompt Evaluation Examples

In addition to presenting the quantitative evaluation metrics, this Section includes screenshots demonstrating how PromptShield operates in typical User Interactions. These screenshots present the complete decision process for PromptShield, including; (1) Semantic Classification, (2) Policy Enforcement, and (3) Entity Level Redaction of Sensitive Spans.

4.7.1 User input prior to classification

The first example, Figure 4.3, presents the original prompt entered by the HR User before it was classified. This image is intended to illustrate the original User Interface and the unaltered source material for which the semantic classifier and policy evaluation pipeline were invoked.

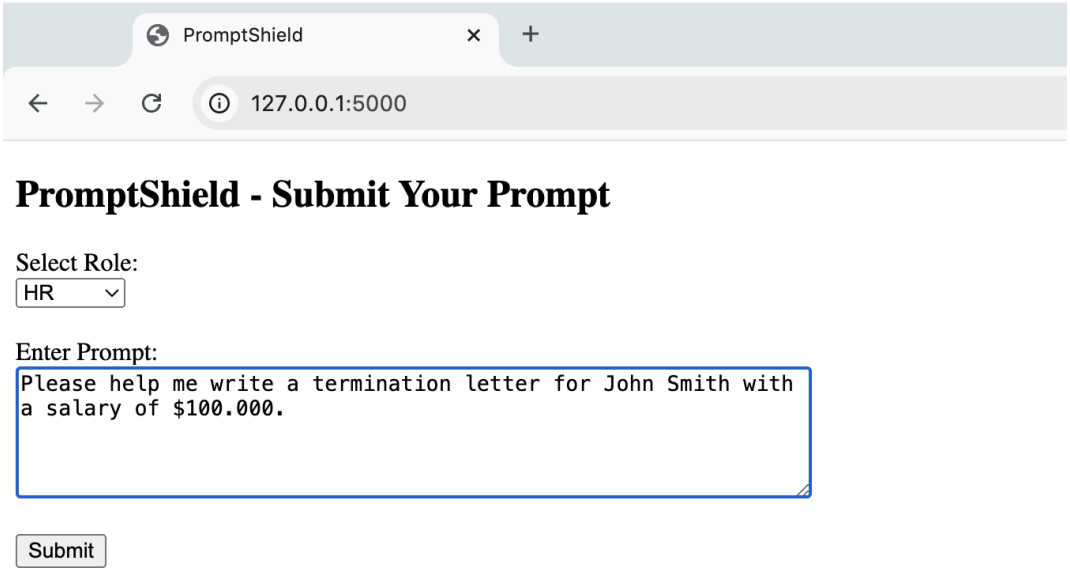

Figure 4.3: User interface screenshot showing the original HR prompt.

4.7.2 Warn decision with entity-level redaction

The second example, Figure 4.4, illustrates an actual interaction between an HR User submitting a prompt including both personal identifying information and financial data. In accordance with the HR policy, PromptShield identified the prompt as Confidential and issued a Warn Decision based upon the policy violations in the prompt. As illustrated in the redacted version of the prompt, PromptShield used entity aware placeholder values to mask the sensitive spans within the prompt, while maintaining the readability and intent of the original prompt.

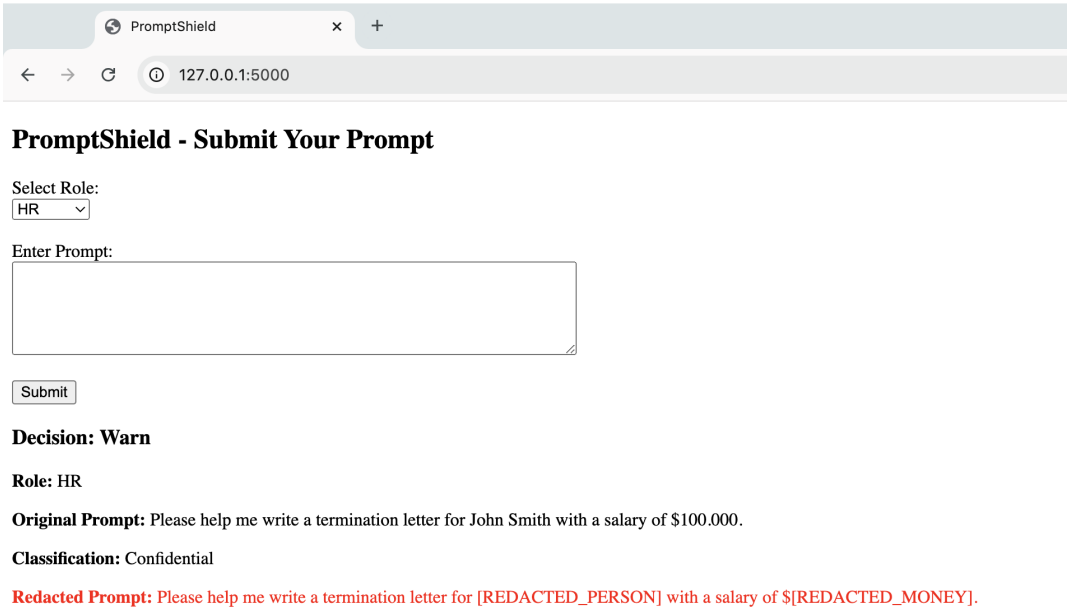

Figure 4.4: PromptShield interface displaying a warn decision.

Overall these screenshots illustrate PromptShield's capability to provide users with timely feedback, support user awareness of potential sensitive information, and protect sensitive data from being transmitted or otherwise made available to third parties in an unsanitized manner.

4.8 Runtime Performance

The PromptShield system runs in near-real time. This includes:

- Embedding of input data
- Classification of the embedded data
- Evaluation of a set of policies with respect to that data
- Optional redaction (deletion) of sensitive portions of the data
- Output of results back through the User Interface (UI).

The average time from receiving the first character of an input until displaying the last character of the output was between 120 – 150 milliseconds. Depending on the length of the input and the number of sensitive parts to be deleted, this time can vary. Since the individual components of the models used by the system are small (lightweight), no additional delay is introduced during operation and it will therefore run quickly enough to support deployment directly on the client-side computer without adversely affecting the user experience.

4.9 Summary

PromptShield successfully achieves the primary goals it was designed to accomplish as illustrated by the results of the evaluation. It has effectively been able to classify sensitivity of realistic enterprise prompts; enforce consistent role based policies; accurately perform redactions when necessary; and operate within established real-time performance constraints. Although there are some challenges associated with the identification of borderline cases, overall the PromptShield framework represents an effective way to mitigate unintentional data disclosure in enterprise settings.

In addition to the numeric evaluations, the qualitative examples demonstrate how PromptShield would behave in typical user interactions. PromptShield identifies high risk personal, financial and organizational entities in a consistent manner; and the Warn workflow represents a reasonable middle ground between usability and security for users. These observations further support the concept that prompt-level DLP mechanisms need to be both accurate and transparent so users can understand and correct their input without interrupting their workflow.

The evaluation findings indicate that the role-based policy framework allows for an increase in the precision of enforcement actions relative to other frameworks by providing a definition for both allowable and disallowed uses of sensitive information by department type (i.e., HR, finance, research & development). In doing so, it also provides a method to allow for a balance between limiting the flow of sensitive information in high-risk areas of the organization, and preventing the blocking of acceptable access/use of sensitive information. Thus, it demonstrates the ability of PromptShield to adapt to the various levels of organizational control desired for managing data and meeting regulatory requirements.

Finally, the evaluation of the redaction component of PromptShield demonstrated that it performed as expected, even when the user prompt contained overlapping entities; and maintained the original sentence structure and readability, enabling users to make meaningful revisions to their prompts. While some limitations were noted during the evaluation process, the overall performance was deemed robust enough to warrant practical deployment.

Overall, the latency evaluations have shown that the entire pipeline of PromptShield will function well within the real-time response thresholds established for the enterprise environment. As such, it would appear that PromptShield could be easily integrated into the day-to-day workflow of enterprises without creating the perception of delays or friction for users. The results of the evaluation also provide a strong foundation for the potential use of prompt-level, policy-aware DLP techniques in today's Gen AI environments, and provide a solid basis for continuing the development and enhancement of the framework moving forward.

PromptShield evaluation results are well suited for developing future versions of prompt-level, policy aware DLP technologies for Gen AI systems in use today.

The key findings from the evaluation results can be summarized as follows.

- The semantic classifier has been shown to perform accuratly in identifying which of three categories (safe/risky/confidential) prompts used in enterprise settings fall into.

- The role-based policy engine promises better enforcement capability than previously established methods as it identifies what prompt behavior is allowed/blocked for each department (HR/Finance/Research & Development).
- Warn enforcement action provides a reasonable trade-off for both security and usability; with the ability for users to review and correct sensitive prompts using transparent redacted feedback provided by the system.
- The redaction mechanism preserves the overall structure of sentences and readability when sensitive information contained in prompts need to be redacted; especially when there are multiple or overlapping sensitive entities in prompts.
- Latency measurements from end-to-end validate that PromptShield meets all of the latency requirements for real time GenAI interactions in enterprise workflows.

In summary, the evaluation results indicate that the proposed framework performs consistently among its various components when utilized under realistic usage scenarios; providing additional evidence that prompt level security controls may be feasible to deploy in enterprise GenAI work flows.

5. DISCUSSION

Based on the architectural components outlined in Chapter 3 and the experimental results provided in Chapter 4, the current chapter will provide a holistic synthesis of the test results to evaluate its applicability to real-world Enterprise deployments. The prior chapters centered on both the design of the systems and the quantifiable results; this chapter provides an overall interpretation of these findings, focusing on classification behaviors, policy interactions, and redaction efficacy. The current chapter will establish the relationship between the theoretical objectives of the framework and the practical results observed during the test process as well as revisit the major research questions to assess PromptShield's practical relevance, identify potential improvements and research directions.

5.1 Interpretation of Results

PromptShield's reliability as a client-side data loss prevention system for generating AI prompts has been shown by the evaluation results. Overall, the classification accuracy (72.92%) is consistent with those for smaller, low-latency, semantically-accurate models like MiniLM. Additionally, in Chapter 4, a weighted F1-score of 0.7462 and a high per-class F1-score of Confidential class 0.8462, demonstrating that the classifier effectively identifies the most sensitive types of content, which is the goal in a DLP environment. It should be noted that this level of performance may be less than what larger transformer models have achieved; however, given that PromptShield is designed to operate as a simple inline prevention tool on commodity hardware (with no GPU acceleration), it is reasonable. As stated in Chapter 3, the classifier does not need to identify all of the sensitive aspects of a prompt accurately; rather, it is intended to provide a preliminary indication of the potential risk to inform policy-driven decisions.

An examination of the confusion matrix in Chapter 4 shows that the majority of misclassifications occurred between the Safe and Risky classes. Misclassifying

between these two categories would be expected because many prompts in the Safe and Risky categories are likely to have similar linguistic characteristics and will typically depend on context to distinguish them from one another. For example, although "Can I review the salary distribution data for trend analysis?" contains no domain specific or named entity references, it can be assumed to refer to compensation data from the Human Resources department based on knowledge about the organization. Therefore, despite being labeled as Risky, it was classified as Safe.

While borderline cases (i.e., misclassified prompts) are frequently found during manual error analysis, many of those misclassifications have used terms that are vague regarding language related to compliance (e.g., “salary review,” “performance data,” “budget trends”) and signal sensitivity only in certain contexts of a particular organization. As the dataset described in Chapter 3 is synthetic and lacks the deeper organizational semantic context, the classifier failed on occasion to identify the inherent risk within those phrases. The results reinforce the necessity of developing DLP systems that use both lexical and contextual cues; possibly using domain-specific adapted embeddings or hybrid models that incorporate metadata (such as user roles or document categories) at inference time.

Although some of the prior works have their own limitations, such as lack of evaluation in realistic use cases, PromptShield’s cautious nature successfully limited the number of high-risk false negatives. The classification model provided a strong recall for Confidential prompts thereby limiting the potential for unintentional disclosure of very sensitive information. This cautious bias aligns with the DLP principles outlined in Chapter 3 and represents an acceptable trade off in enterprise environments, where the impact of mistakenly allowing a sensitive prompt is far more severe than incorrectly warning or blocking a benign one.

The policy engine also improved the overall performance of the system. In fact, the results presented in Chapter 4 indicated that approximately one-third of all prompts required a Warn or Block action. The magnitude of this proportion indicates that PromptShield provided sufficient protection against attacks without burdening the user with excessive warnings. The role-specific differentiation and specifically the

more restrictive treatment of R&D prompts was verified through this distribution demonstrating how the application of context-dependent policy logic enhances both usability and security.

Lastly, the runtime measurements confirmed that PromptShield meets the requirements of timely interactive input submission. With end-to-end latency ranging from 120-150ms, the users do not perceive significant delays when submitting prompts. This validates the design decisions in Chapter 3, which intentionally structured the pipeline to minimize computation while preserving semantic fidelity.

5.2 Relation to Research Questions

Each of the research questions outlined in Chapter 1 was answered empirically as part of this dissertation. Each element of the proposed framework adds value toward creating an application relevant to real world needs.

RQ1: *Can prompt sensitivity be inferred reliably in real time using compact semantic models?*

The results from Chapter 4 demonstrate the lightweight embedding models have enough context about the input prompts so that they are able to classify whether or not a given prompt is sensitive with a good degree of accuracy (72.92%, Weighted F1 score = 0.7462).

RQ2: *Does role-based policy enforcement enhance DLP precision while reducing friction?*

The results from the Allow/Warn/Block distributions support the claim that by employing role based enforcement, DLP may be both more precise and less intrusive than traditional DLP systems. By adapting enforcement to HR, Finance, and R&D contexts, PromptShield offers more nuanced and less intrusive protection

RQ3: *Can inline NER-based redaction preserve prompt usability while mitigating risk?*

The redaction examples show that how entity aware masking will allow users to safely reformulate their prompts while continuing their workflow uninterrupted.

These results fill a void between the permissive nature of some current DLP systems and the restrictive nature of others.

RQ4: *Is prompt-level DLP feasible within enterprise latency constraints?*
PromptShield has demonstrated that it is capable of meeting the real-time threshold established for practical enterprise applications.

5.3 Comparative Analysis with Existing Solutions

Conventional DLP systems excel at monitoring structured data in channels such as email, file uploads, or cloud storage. However, they are not designed to analyze short, free-text prompts submitted to generative AI platforms. PromptShield extends the DLP paradigm into this new interaction surface by focusing on pre-submission analysis at the moment of risk creation.

Unlike LLM safety filters, which analyze model outputs, PromptShield evaluates user originated input. This distinction is critical because output filters cannot prevent sensitive information from being transmitted outside the enterprise boundary. PromptShield's preemptive architecture ensures that data never leaves the organization if it is deemed sensitive, supporting compliance with data minimization principles under GDPR.

PromptShield is also vendor agnostic. Its enforcement logic operates independently of the underlying generative AI model, allowing organizations to apply uniform policies across multiple AI providers.

5.4 Implications for Enterprise Adoption

The evaluation results have demonstrated some practical implications for the organizations that are interested in adopting generative AI safely (with respect to an organization's use):

- **Policy Customization:** The JSON-based policy framework provides administrators with the ability to define the enforcement logic (it can tailor-made for companies such as adding stricter rules for research and development department if a company

is a pharmaceutical company) to meet the specific departmental risk factors and compliance obligations for their organization.

- **User Awareness:** The Warn mechanism functions as a soft-training signal to improve employees' awareness of what is considered sensitive content by providing them with repeated warnings about such content and also for user to review it's prompt before sending it to the LLM.
- **Governance and Auditability:** The logging infrastructure discussed in Chapter 3, supports auditing for compliance and incident response workflows by allowing users to track both employee actions and policy enforcement decision-making processes.
- **Integration Flexibility:** PromptShield's modular and lightweight design enables seamless deployment in browsers, enterprise chat tools, local clients, or API gateways.

5.5 Limitations and Challenges

While the proposed system shows a lot of promise, there are still some limitations that have been identified in its current form. The most significant limitation is the classifier's performance on "sensitive" prompts that do not rely upon the identification of an explicit entity (i.e., where the sensitive nature of the prompt is based upon the context rather than the entity itself).

Therefore, while future work can focus on improving the classifier through the use of either domain-specific embeddings or hierarchical classification methods; it has also been found that general purpose NER does not recognize organizational specific identifiers (e.g., internal project names, HR codes), which necessitates the development of custom NER models for each organization.

Another limitation of PromptShield at present is that it only supports three user role types and static policies; however, expanding these to support more granular roles, or adaptive policy systems would likely enhance both the security and usability of the tool.

5.6 Future Directions

Several enhancements could strengthen PromptShield's capabilities:

- **Multilingual support** for global organizations.
- **Fine-tuned NER models** trained on domain-specific datasets.
- **Adaptive policy learning** using reinforcement learning or human-in-the-loop feedback.
- **Integration with SIEM tools** to unify visibility across enterprise security controls.

5.7 Summary

PromptShield's overall performance was described in detail, and based on the evaluation, the strengths (positive) and weaknesses (negative) were identified. Furthermore, the experimental results illustrate that there are means to develop and deploy organizational-specific policy compliant, prompt level DLP capabilities to protect generative AI systems.

It is evident from the results that to significantly limit the potential risk of unintentional release of sensitive information from GenAI through a variety of means including; role-based contextual policy enforcement, semantic classification, and redaction, a combination of all of these methods need to be employed.

The results clearly illustrated that transparent role aware controls can be utilized to enable the safe deployment of GenAI, without disrupting user workflows. The use of such controls have demonstrated that instead of using generic blocking methods to prevent sensitive information from being released by generative AI systems, that generative AI systems can be safely used, with minimal disruption to users workflow.

Ultimately, this thesis demonstrated the viability of using prompt level DLP as one of the ways to secure generative AI systems, that contain a substantial amount of sensitive data and corresponding regulations. While, the current implementation of PromptShield is still in prototype form, the concepts developed, and the results

obtained from the experimental evaluation of this thesis, provide a solid foundation upon which future research and actual deployments of systems may be based.

6. CONCLUSION

PromptShield is an easy-to-use, policy-based Data Loss Prevention (DLP) framework which protects against unintentional disclosure of sensitive information through Generative AI prompts. The prompt shield framework is a unique client side solution for assessing the user's input prior to it being sent to the external Large Language Models (LLMs), which fills a critical security void left by both existing DLP solutions as well as model-centric safety architecture.

PromptShield was developed in response to an unmet need in DLP at the prompt level. The use of generative AI is becoming a central component of all enterprise processes and therefore most users are submitting "free form" text that could contain sensitive information such as personal identifiable information (PII), financial information, etc. In addition, the current security measures protect primarily the output of the models and/or filter what data enters the cloud and do not adequately protect the user-generated prompts. Therefore, PromptShield has been designed to fill this void with its real time integration of a semantic classifier, role based policy enforcement and entity aware redactions.

The design of the architecture demonstrates how a lightweight classifier (using MiniLM-based embeddings), a JSON based policy engine, and a redactor (using spaCy) can work in tandem through an efficient client-side pipeline. The evaluation shows that the system achieves all of its required operating characteristics. The classifier had a total accuracy rate of 72.92% with high recall for high risk categories. The policy engine was able to map the classifier's output into action-able enforcement policies and the redaction module was consistent in preserving the sentence structure and protecting the sensitive entities. Additionally, latency results show that the full pipeline runs under 120 – 150 ms allowing for true time usability.

In discussions, we combined the above findings, as well as other implications for enterprise use. PromptShield's conservative classification method coupled with the

role-based policy rules are designed to limit false negatives and protect sensitive data from leaving the organization. Analysis of misclassifications indicated that most borderline errors arise from either explicit or contextual phrasing, highlighting a need for better domain adaptation. Overall, the evaluation of prompt-level DLP, demonstrated it is both viable and practical for today's workplace where AI integration is becoming increasingly common.

6.1 Contributions

The thesis makes several key contributions:

- **A novel prompt-level DLP framework** that analyzes user inputs before submission to external LLMs, addressing a critical security gap.
- **A lightweight semantic classification pipeline** capable of real-time sensitivity detection using compact sentence embeddings.
- **A role-based policy model** that incorporates organizational context into enforcement decisions, improving both security and usability.
- **A practical redaction mechanism** that enables safe reformulation of prompts rather than relying solely on blocking strategies.
- **A synthetic but representative dataset** tailored for evaluating enterprise prompt sensitivity across HR, Finance, and R&D domains.
- **An end to end evaluation** demonstrating the feasibility of client-side, low latency DLP for generative AI interactions.

6.2 Future Directions

Although the results clearly show that PromptShield is a feasible tool to protect against sensitive information leakage from chatbots, there are many areas of future work that will likely lead to improvements:

- **Domain-adapted classification models:** The sensitivity-detection abilities may be enhanced through the use of fine-tuned models on the company's own data or through the use of more descriptive embedding models.
- **Enhanced NER models:** Organization-specific NER systems can be trained to better identify proprietary identifiers, internal project names or even employee IDs.
- **Dynamic and adaptive policies:** If PromptShield were to incorporate elements of reinforcement learning or if it could learn based upon feedback (either positive or negative), then the policy rules could evolve as users interact with the system and as new threats arise.
- **Multilingual support:** In order to deploy in global enterprise environments, the framework should include support for multiple languages.
- **Integration with enterprise monitoring systems:** By sending PromptShield logs to SIEM or DLP platforms, organizations can make the audit trail more accessible and integrate their security controls.

6.3 Closing Remarks

PromptShield illustrates that in today's growing number of enterprises utilizing generative AI tools, prompt-level, policy-aware DLP will be necessary and viable. It accomplishes this by integrating; semantic understanding of the prompt, a contextual policy-based decision-making process and an immediate redaction process. The design of PromptShield makes it practical as well as capable of effectively stopping users from inadvertently disclosing their company's proprietary or sensitive information. Its small footprint, low-latency and flexibility provide the opportunity to integrate it with many types of user-facing applications. The findings of this thesis provide a base for future research in the areas of data security, humans and AI interaction, and responsible AI development.

REFERENCES

[1] **Tahboub, R. and Saleh, Y.** (2014). Data Leakage/Loss Prevention Systems (DLP), *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, IEEE, pp.1–6.

[2] **Ahmad, S.**, **Mehfuz, S. and Beg, J.** (2022). Cloud security framework and key management services collectively for implementing DLP and IRM, *Materials Today: Proceedings*, *62*, 4828–4836.

[3] **Ghouse, M.**, **Nene, M.J. and Vembuselvi, C.** (2019). Data Leakage Prevention for Data in Transit using Artificial Intelligence and Encryption Techniques, *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*, pp.1–6.

[4] **Kalota, F.** (2024). A Primer on Generative Artificial Intelligence, *Education Sciences*, *14*(2), 172.

[5] **Wang, X.**, **Wan, Z.**, **Hekmati, A.**, **Zong, M.**, **Alam, S.**, **Zhang, M. and Krishnamachari, B.** (2025). The Internet of Things in the Era of Generative AI: Vision and Challenges, *IEEE Network*.

[6] **Flowerday, S.V. and Tuyikeze, T.** (2016). Information security policy development and implementation: The what, how and who, *Computers & Security*, *61*, 169–183.

[7] **Paananen, H.**, **Lapke, M. and Siponen, M.** (2020). State of the art in information security policy development, *Computers & Security*, *88*, 101608.

[8] **Ghorbanian, S.**, **Fryklund, G. and Axelsson, S.**, (2015). Do Data Loss Prevention Systems Really Work?, **G. Peterson and S. Shenoi**, editors, Advances in Digital Forensics XI, Springer International Publishing, pp.341–357.

[9] **AlKilani, H.**, **Nasereddin, M.**, **Hadi, A. and Tedmori, S.** (2019). Data Exfiltration Techniques and Data Loss Prevention System, *2019 International Arab Conference on Information Technology (ACIT)*, 124–127.

[10] **Gugelmann, D.**, **Studerus, P.**, **Lenders, V. and Ager, B.** (2015). Can Content-Based Data Loss Prevention Solutions Prevent Data Leakage in Web Traffic?, *IEEE Security & Privacy*, *13*(4), 52–59.

[11] **Kaur, K.**, **Gupta, I. and Singh, A.K.** (2017). A Comparative Evaluation of Data Leakage/Loss Prevention Systems (DLPS), *Computer Science & Information Technology (CS & IT)*, 87–95.

[12] **Wüchner, T. and Pretschner, A.** (2012). Data loss prevention based on data-driven usage control, *2012 IEEE 23rd International Symposium on Software Reliability Engineering*, pp.151–160.

[13] **Paracha, M.A.**, **Ullah, S.**, **Shah, L.T.**, **Khan, S.A. and Ali, A.** (2022). Implementation of Two Layered DLP Strategies, *International Conference on Cyber Warfare and Security (ICCWS)*, pp.1–9.

[14] **Yadav, I. and Gupta, H.** (2023). Designing Data Loss Prevention System for The Enhancement of Data Integrity in Cyberspace, *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pp.1361–1365.

[15] **Kiperberg, M.**, **Amit, G.**, **Yeshooroon, A. and Zaidenberg, N.J.** (2021). Efficient DLP-visor: An efficient hypervisor-based DLP, *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp.344–355.

[16] **Srinivasan, A.**, **Irizarry, L.**, **Schleining, T. and Dong, H.** (2025). PrintWatchDog: Lightweight Open-Architecture Framework for Printer Data Loss Prevention, *2025 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp.1–7.

[17] **Srivastava, A.**, **Sharma, V.S.**, **Srivastava, P. and Pillai, A.** (2024). A Hybrid Framework for Data Loss Prevention and Detection, *2024 International Conference on Signal Processing and Advance Research in Computing (SPARC)*, pp.1–6.

[18] **Muppalaneni, R.**, **Inaganti, A.C. and Ravichandran, N.** (2024). AI-Enhanced Data Loss Prevention (DLP) Strategies for Multi-Cloud Environments, *Computing Innovations and Applications*.

[19] **Esther, D.** (2024). *AI in data loss prevention: Safeguarding sensitive data against unauthorized access and leakage*.

[20] **Bertino, E.**, **Kantarcioglu, M.**, **Akcora, C.G.**, **Samtani, S.**, **Mittal, S. and Gupta, M.** (2021). AI for Security and Security for AI, *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy (CODASPY '21)*, pp.333–334.

[21] **Wang, X.** (2024). *Security Of AI, By AI and For AI: Charting New Territories in AI-Centered Cybersecurity Research*.

[22] **Rahman, M.M.**, **Arshi, A.S.**, **Hasan, M.M.**, **Mishu, S.F.**, **Shahriar, H. and Wu, F.** (2023). Security Risk and Attacks in AI: A Survey of Security and Privacy, *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp.1–6.

[23] **Yu, X.**, **Huo, L. and Li, X.** (2025). Survey of Research on Artificial Intelligence and Cybersecurity, *2025 IEEE International Conference on Pattern*

Recognition, Machine Vision and Artificial Intelligence (PRMVAI), pp.1–6.

[24] **Kayode, B.**, **Adebola, N.T.**, **Akerele, S.**, **Fagbohun, O.**, **Agbos, C.**, **Bantale, O. and Nwokocha, L.C.** (2025). The State of AI-Driven Cybersecurity: Trends, Challenges and Opportunities, *Journal of Artificial Intelligence, Machine Learning and Data Science*, *3*(2), 2731–2739.

[25] **Singh, K.**, **Saxena, R. and Kumar, B.** (2024). AI Security: Cyber Threats and Threat-Informed Defense, *2024 8th Cyber Security in Networking Conference (CSNet)*, pp.1–6.

[26] **Yigit, Y.**, **Buchanan, W.J.**, **Tehrani, M.G. and Maglaras, L.** (2024). Review of Generative AI Methods in Cybersecurity, *arXiv preprint arXiv:2403.08701*.

[27] **Diro, A.**, **Kaisar, S.**, **Saini, A.**, **Fatima, S.**, **Hiep, P.C. and Erba, F.** (2025). Workplace security and privacy implications in the GenAI age: A survey, *Journal of Information Security and Applications*, *89*, 103960.

[28] **Saddiqa, M. and Ruohonen, J.** (2025). SoK: The Psychology of Insider Threats, *EAI Endorsed Transactions on Security and Safety*.

[29] **Asasfeh, A.**, **Alnawayseh, S.E.A.**, **Alomoush, R.A. and Salahat, M.** (2024). Human Factors In Security Management: Understanding And Mitigating Insider Threats, *2024 2nd International Conference on Cyber Resilience (ICCR)*, pp.1–10.

[30] **Sebastian, G.** (2023). Do ChatGPT and Other AI Chatbots Pose a Cybersecurity Risk? An Exploratory Study, *International Journal of Security and Privacy in Pervasive Computing*, *15*(1), 1–11.

[31] **Arora, S.**, **Manral, V.**, **S, D. and Chakraborty, P.** (2025). AI, Privacy, and Data Leakage: A Study of Current DLP Shortcomings, 269–273.

[32] **Teo, Z.L.**, **Ning, W.**, **Le, J. and Shu, D.** (2024). Cybersecurity in the generative artificial intelligence era, *Asia-Pacific Journal of Ophthalmology*, *13*(4), 100091.

[33] **Liu, X.**, **Yu, Z.**, **Zhang, Y.**, **Zhang, N. and Xiao, C.** (2024). Automatic and Universal Prompt Injection Attacks against Large Language Models, *arXiv preprint arXiv:2403.04957*.

[34] **Gulyamov, S.**, **Gulyamov, S.**, **Rodionov, A.**, **Khursanov, R.**, **Mekhmonov, K.**, **Babaev, D. and Rakhimjonov, A.** (2025). Prompt Injection Attacks in Large Language Models and AI Agent Systems: A Comprehensive Review of Vulnerabilities, Attack Vectors, and Defense Mechanisms.

[35] **Gupta, M.**, **Akiri, C.**, **Aryal, K.**, **Parker, E. and Praharaj, L.** (2023). From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy, *IEEE Access*, *11*, 80218–80245.

[36] **Zhu, B.**, **Mu, N.**, **Jiao, J. and Wagner, D.** (2024). Generative AI Security: Challenges and Countermeasures, *arXiv preprint arXiv:2402.12617*.

[37] **Mathew, E.** (2024). Enhancing Security in Large Language Models: A Comprehensive Review of Prompt Injection Attacks and Defenses, *TechRxiv*.

[38] **Campbell, M. and Jovanovic, M.** (2024). Disinfecting AI: Mitigating Generative AI's Top Risks, *Computer*, *57*(5), 111–116.

[39] **Ding, Meng, X.W.C.W.K.W. and Yang, X.** (2021). Research on Automated Detection of Sensitive Information Based on BERT, *Journal of Physics*, *1757*(1), 012088.

[40] **Saritha, P. and Kumar, R.** (2025). Sensitive Data Protection using AI: An Evaluation of Deep Learning and Metaheuristic-Based Leakage Prevention Techniques, *2025 International Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT)*, pp.1–6.

[41] **Gupta, K. and Kush, A.** (2023). A learning oriented DLP System based on Classification Model, *arXiv preprint arXiv:2312.13711*.

[42] **Kaliappan, V.K.**, **Dharunkumar, U.P.**, **Uppili, S.**, **Vigneshwarar, A.A. and Bharani, S.** (2024). SentinelGuard: An Integration of Intelligent Text Data Loss Prevention Mechanism for Organizational Security (I-ITDLP), *2024 International Conference on Science Technology Engineering and Management (ICSTEM)*, pp.1–6.

[43] **Kutbi, M.** (2023). Named Entity Recognition Utilized to Enhance Text Classification While Preserving Privacy, *IEEE Access*, *11*, 117576–117581.

[44] **Ali, R.S.**, **Zhao, B.Z.H. and Asghar, H.J.** (2022). Unintended Memorization and Timing Attacks in Named Entity Recognition Models, *arXiv preprint arXiv:2211.02245*.

[45] **Al-sanabani, H. and Iskefiyeli, M.** (2020). A DLP module design based on plug-in for MS Word, *Sakarya University Journal of Science*, *24*(4), 770–781.

[46] **Ferrari, F.**, **van Dijck, J. and van den Bosch, A.** (2025). Observe, inspect, modify: Three conditions for generative AI governance, *New Media & Society*, *27*(5), 2788–2806.

[47] **Leong, H.Y. and Gao, Y.** (2024). A GEN AI Framework for Medical Note Generation, *2024 6th International Conference on Artificial Intelligence and Computer Applications (ICAICA)*.

[48] **Zhang, P. and Kamel Boulos, M.N.** (2024). Generative AI in Medicine and Healthcare: Promises, Opportunities and Challenges, *Future Internet*.

[49] **Robert, L.** (2023). *AI-Driven Identity Access Management for Advanced Data Loss Prevention in Pharmaceutical Industries*.

[50] **Shinde, C. and Garikapati, D.** (2025). Gen AI in Automotive: Applications, Challenges, and Opportunities with a Case study on In-Vehicle Experience, *arXiv preprint arXiv:2511.00026*.

[51] **Tomar, P. and Grover, V.** (2025). Transforming the Energy Sector: Addressing Key Challenges through Generative AI, Digital Twins, AI, Data Science and Analysis, *EAI Endorsed Transactions on Energy Web*.

[52] **Ng, D.T.K.**, **Chan, E.K.C. and Lo, C.K.** (2025). Opportunities, challenges and school strategies for integrating generative AI in education, *Computers and Education: Artificial Intelligence*, *8*, 100373.

[53] **Klemmer, J.H.**, **Horstmann, S.A.**, **Ludden, C.**, **Burton Jr., C.**, **Massacci, F.**, **Rahman, A.**, **Lipford, H.R.**, **Rashid, A. and Fahl, S.** (2024). Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns, *Proceedings of the IEEE Symposium on Security and Privacy*.

[54] **European Parliament and Council** (2024). *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*.

[55] **National Institute of Standards and Technology (NIST)** (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0), **Technical Report**, U.S. Department of Commerce.

[56] **European Parliament and the Council of the European Union** (2024). *Regulation (EU) 2024/2847 on the Cyber Resilience Act*, `https://eur-lex.europa.eu/eli/reg/2024/2847/oj`, official Journal of the European Union.

[57] **Yang, M.**, **Wang, J.**, **Zhang, T.**, **Jiang, R. and Gao, T.** (2025). Risk identification, quantification and governance strategies for generative artificial intelligence service, *Chinese Management Studies*.

[58] **Wach, K.**, **Duong, C.D.**, **Ejdys, J.** *et al.* (2023). The dark side of generative artificial intelligence: A critical analysis of controversies and risks of ChatGPT, *Entrepreneurial Business and Economics Review*, *11*(2), 7–30.

[59] **Christodorescu, M.**, **Craven, R.**, **Feizi, S.**, **Jha, S.** *et al.* (2024). Securing the Future of GenAI: Policy and Technology, *arXiv preprint arXiv:2407.12999*.

[60] **Denzel, K.** (2025). A survey of security in zero trust network architectures.

[61] **Edo, O.C.**, **Tenebe, T.**, **Etu, E.e.**, **Ayuwu, A.**, **Emakhu, J. and Adebiyi, S.** (2022). Zero trust architecture: Trend and Impacton information security, *International Journal of Emerging Technology and Advanced Engineering*, *12*(7), 140.

APPENDICES

APPENDIX A : Source Code

APPENDIX A: Source Code

This appendix presents selected Python code excerpts from the PromptShield prototype, including the semantic classifier, role-based policy engine, redaction module, Flask application, and evaluation scripts. Long lines have been manually shortened to avoid page overflow in accordance with the thesis formatting requirements.

```
-- nlp_classifier.py --

import joblib
import os

MODEL_PATH = "promptshield_classifier_rf.pkl"
EMBEDDER_PATH = "promptshield_embedder_rf.pkl"
ENCODER_PATH = "promptshield_label_encoder.pkl"

paths = [MODEL_PATH, EMBEDDER_PATH, ENCODER_PATH]
if not all(os.path.exists(p) for p in paths):
    raise FileNotFoundError(
        "Model or components not found. Train classifier first."
    )

clf = joblib.load(MODEL_PATH)
embedder = joblib.load(EMBEDDER_PATH)
encoder = joblib.load(ENCODER_PATH)


def classify_prompt(prompt: str) -> str:
    embedding = embedder.encode([prompt])
    encoded = clf.predict(embedding)[0]
    label = encoder.inverse_transform([encoded])[0]
    return label

-- policy_engine.py --

POLICIES = {
    "HR": {
        "allowed": ["Safe"],
        "blocked": [],
        "warn": ["Risky", "Confidential"],
    },
    "Finance": {
        "allowed": ["Safe", "Risky"],
        "blocked": ["Confidential"],
        "warn": [],
```

```
    },
    "R&D": {
        "allowed": ["Safe"],
        "blocked": ["Confidential", "Risky"],
        "warn": [],
    },
}

def check_policy(role, classification):
    if classification in POLICIES[role]["blocked"]:
        return "Block"
    if classification in POLICIES[role]["warn"]:
        return "Warn"
    return "Allow"


-- redactor.py --

import spacy

nlp = spacy.load("en_core_web_sm")

REDACTABLE = {"PERSON", "ORG", "MONEY", "GPE", "DATE", "LOC"}

def redact_prompt(prompt: str) -> str:
    doc = nlp(prompt)
    redacted = prompt

    for ent in doc.ents:
        if ent.label_ in REDACTABLE:
            placeholder = f"[REDACTED_{ent.label_}]"
            redacted = redacted.replace(ent.text, placeholder)

    return redacted

-- app.py --

from flask import Flask, render_template, request
import datetime
import os

from nlp_classifier import classify_prompt
from policy_engine import check_policy
from redactor import redact_prompt

app = Flask(__name__)


@app.route("/", methods=["GET", "POST"])
def home():
```

```
    if request.method == "POST":
        role = request.form["role"]
        prompt = request.form["prompt"]

        classification = classify_prompt(prompt)
        decision = check_policy(role, classification)

        redacted_prompt = None
        if decision == "Warn":
            redacted_prompt = redact_prompt(prompt)

        timestamp = datetime.datetime.now().isoformat()
        log = (
            f"{timestamp} | Role: {role} | Prompt: {prompt} | "
            f"Class: {classification} | Decision: {decision}\n"
        )

        os.makedirs("logs", exist_ok=True)
        with open("logs/logs.txt", "a", encoding="utf-8") as f:
            f.write(log)

        return render_template(
            "index.html",
            role=role,
            prompt=prompt,
            classification=classification,
            decision=decision,
            redacted_prompt=redacted_prompt,
        )

    return render_template("index.html", decision=None)


if __name__ == "__main__":
    app.run(debug=True)
```

-- train_nlp_classifier.py --

```
import pandas as pd
import joblib
from sentence_transformers import SentenceTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("Synthetic_Labeled_Prompt_Dataset_Full.csv")

prompts = df["Prompt"].tolist()
labels = df["Sensitivity"].tolist()

embedder = SentenceTransformer("all-MiniLM-L6-v2")
```

```
X = embedder.encode(prompts)

encoder = LabelEncoder()
y = encoder.fit_transform(labels)

clf = RandomForestClassifier(
    n_estimators=200,
    random_state=42
)
clf.fit(X, y)

joblib.dump(clf, "promptshield_classifier_rf.pkl")
joblib.dump(embedder, "promptshield_embedder_rf.pkl")
joblib.dump(encoder, "promptshield_label_encoder.pkl")

-- evaluate.py --

import pandas as pd
from nlp_classifier import classify_prompt
from policy_engine import check_policy
from redactor import redact_prompt

df = pd.read_csv("Synthetic_Labeled_Prompt_Dataset_TestOnly.csv")

results = []

for _, row in df.iterrows():
    role = row["Role"]
    prompt = row["Prompt"]

    label = classify_prompt(prompt)
    decision = check_policy(role, label)

    redacted = None
    if decision == "Warn":
        redacted = redact_prompt(prompt)

    results.append({
        "Role": role,
        "Prompt": prompt,
        "Classification": label,
        "Decision": decision,
        "Redacted": redacted
    })

pd.DataFrame(results).to_csv("evaluation_results.csv", index=False)

-- evaluate_accuracy.py --

import pandas as pd
```

```
eval_df = pd.read_csv("evaluation_results.csv")
gt_df = pd.read_csv("Synthetic_Labeled_Prompt_Dataset_TestOnly.csv")

eval_df["Role"] = eval_df["Role"].str.strip()
gt_df["Role"] = gt_df["Role"].str.strip()

eval_df["Prompt"] = eval_df["Prompt"].str.strip()
gt_df["Prompt"] = gt_df["Prompt"].str.strip()

eval_df["Classification"] = (
    eval_df["Classification"].str.strip().str.capitalize()
)
gt_df["Sensitivity"] = (
    gt_df["Sensitivity"].str.strip().str.capitalize()
)

merged = eval_df.merge(
    gt_df[["Role", "Prompt", "Sensitivity"]],
    on=["Role", "Prompt"],
    how="inner"
)

merged = merged.drop_duplicates(
    subset=["Role", "Prompt", "Sensitivity"]
)

merged["Correct"] = merged["Classification"] == merged["Sensitivity"]
accuracy = merged["Correct"].mean()

incorrect = merged.loc[
    ~merged["Correct"],
    ["Role", "Prompt", "Sensitivity", "Classification"]
]
incorrect.to_csv("incorrect_predictions.csv", index=False)

print(f"Accuracy: {accuracy:.2%}")
```

CURRICULUM VITAE

Ezgi KELEŞ:

EDUCATION:

- **B.Sc.:** 2019, Istanbul Technical University, Faculty of Computer and Informatics Engineering, Department of Information Security Engineering,
- **M.Sc.:** 2025, Faculty of Computer and Informatics Engineering, Department of Computer Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2025–Present: Associate Managing Consultant - Cybersecurity, Mastercard.
- 2025–2025: Security & Compliance Solutions Technical Specialist, Microsoft.
- 2022–2025: Consultant- Cybersecurity, Mastercard.
- 2022–2022: Regional Factory Cyber Security Manager, Unilever.
- 2020–2022: Senior Cyber Security Consultant, EY.
- 2019–2020: Network & Security Engineer, IBM.
- 2019 Mine Kalkan Award for Academic Excellence, Binghamton University.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Keleş, E.** (2025). Policy-Aware DLP Framework for Generative AI Prompts (PromptShield). *2025 International Conference on Computer Science and Engineering (UBMK)*, Istanbul, Turkey. doi: 10.1109/UBMK67458.2025.11206862 (Conference Paper)