
Consent-Aware Privacy Firewall: Preventing Accidental Sensitive Data Disclosure in Generative AI Interfaces

Mahesh Kandagatla
Independent

Bhargava Krishna Chiliveru
Independent

With
Apart Research

Abstract

Consent-Aware Privacy Firewall (CAF) is a browser-based privacy protection system designed to reduce accidental disclosure of sensitive information when interacting with Large Language Models (LLMs) such as ChatGPT. Users frequently paste personal, medical, financial, or organizational information into AI systems without fully understanding the privacy implications. Existing solutions focus primarily on detecting personally identifiable information (PII) but rarely intervene before information is submitted.

CAF operates locally within the browser and analyzes prompts and uploaded documents before they are sent to an AI service. The system detects sensitive entities including names, emails, phone numbers, identification numbers, medical information, financial information, credentials, and API keys. Users receive a risk assessment and can choose to mask detected information or proceed with the original content.

Our prototype was implemented as a Chrome Manifest V3 extension and includes local document scanning, consent-aware risk evaluation, privacy-focused audit logging, and an interactive dashboard. All processing occurs locally in the browser, ensuring that raw prompt content is never transmitted to external servers. The project demonstrates a practical approach to reducing privacy risks in everyday AI usage and highlights the importance of proactive privacy controls for generative AI systems.

1. Introduction

Large Language Models (LLMs) have become a common tool for work, education, healthcare, and personal productivity. However, users often share sensitive information with AI systems without realizing the potential privacy risks. Examples include medical records, identification numbers, financial details, passwords, API keys, and confidential business information.

Current AI interfaces generally rely on users to manually review their content before submission. This creates a risk of accidental disclosure, particularly when users are focused on obtaining assistance rather than evaluating privacy implications.

To address this problem, we developed the **Consent-Aware Privacy Firewall (CAF)**, a browser-based system that analyzes content before it is submitted to an AI model. CAF provides real-time privacy feedback, detects sensitive information, and enables users to mask information before sharing.

Our main contributions are:

1. A browser-native privacy firewall that analyzes prompts before submission to AI systems.
2. A local-first architecture that performs all privacy processing within the browser without transmitting raw prompt data externally.
3. A consent-aware workflow that gives users control over masking or sharing detected sensitive information.

2. Related Work

Several tools have been developed to identify sensitive information, including Microsoft Presidio, enterprise Data Loss Prevention (DLP) systems, and browser security extensions. These tools focus primarily on detecting personal information or enforcing organizational policies.

However, most existing solutions are not specifically designed for interactions with generative AI systems. They typically lack user-facing consent workflows and do not provide contextual privacy interventions at the point of prompt submission.

Our work differs by focusing on AI-specific privacy risks and providing a lightweight browser-based mechanism that operates entirely on the client side. Rather than simply detecting sensitive information, CAF actively assists users in deciding what information should be shared.

3. Methods

CAF was implemented as a Chrome Manifest V3 extension.

The system consists of five major components:

1. Detection Engine

The detection engine identifies:

- Names
- Email addresses
- Phone numbers
- Identification numbers
- Financial information
- Medical information
- Passwords
- API keys and access tokens

2. Attachment Scanner

Uploaded PDF and DOCX files are processed locally. Extracted text is analyzed using the same detection pipeline used for prompts.

3. Risk Assessment

Detected entities are combined into a privacy risk score:

- Low Risk
- Medium Risk
- High Risk

4. Masking Engine

Users may replace sensitive information with placeholders before submission.

Example:

Original	Masked
John Smith	[PERSON_NAME]
john.smith@email.com	[EMAIL_ADDRESS]

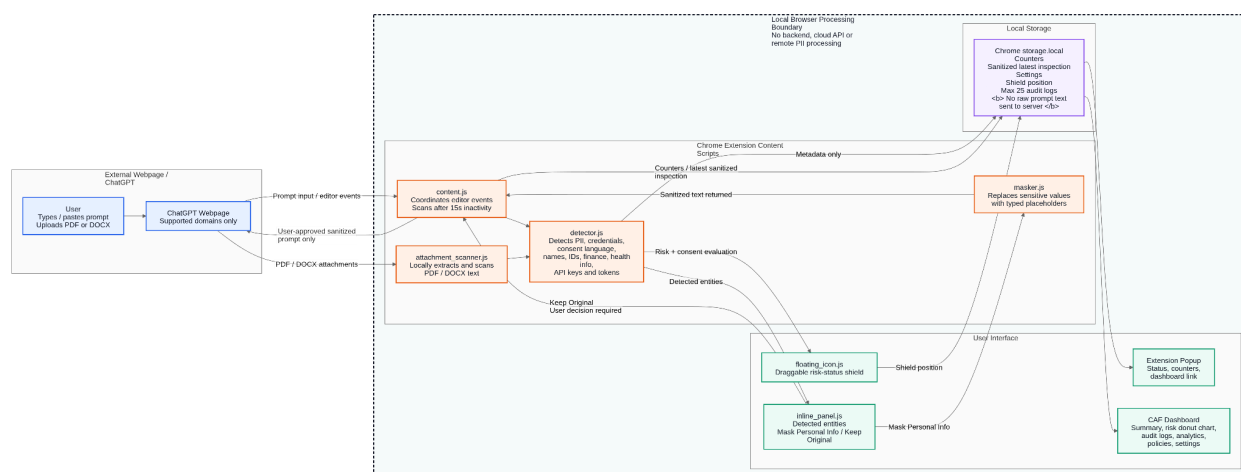
5. Local Storage and Dashboard

The system stores:

- Privacy scan counters
- User settings
- Dashboard analytics
- Limited audit history

Importantly, raw prompt text is never sent to a backend service.

Figure 1: Consent-Aware Privacy Firewall System Architecture.



6. Technology Stack

Layer	Technology	Purpose
Platform	Chrome Extension, Manifest V3	Packages and runs the browser extension
Frontend	HTML5, CSS3	Popup, dashboard, floating shield, and inline panel
Programming language	JavaScript	Detection, masking, storage, attachment scanning, and UI logic
Browser APIs	Chrome Extensions API	Storage, runtime navigation, active-tab access, and content scripts

Local storage	chrome.storage.local	Stores settings, metrics, shield position, and sanitized audit data
Detection engine	JavaScript regular expressions and rule-based scoring	Identifies PII, credentials, consent language, and sensitive information
Document processing	Blob, TextDecoder, DecompressionStream	Locally extracts text from PDF and DOCX attachments

4. Results

The prototype successfully demonstrated real-time privacy analysis within a browser environment.

Capability	Status
Prompt scanning	Implemented
PDF scanning	Implemented
DOCX scanning	Implemented
PII detection	Implemented
Risk scoring	Implemented
Content masking	Implemented
Dashboard analytics	Implemented

Example Privacy Workflow

1. User enters prompt.
2. Sensitive entities are detected.
3. Risk score is displayed.
4. User selects "Mask Personal Info".
5. Sanitized prompt is returned to the editor.
6. User submits approved content to ChatGPT.

The prototype demonstrated that privacy interventions can occur before data leaves the user's browser, reducing the likelihood of accidental disclosure.

Overall Detection Coverage: **76.67%**

5. Discussion and Limitations

Our results demonstrate that browser-based privacy interventions are feasible and can provide an additional safety layer for users interacting with AI systems.

Unlike traditional DLP systems, CAF focuses on user awareness and consent. The system encourages users to actively consider whether sensitive information should be shared.

Limitations

Several limitations remain:

- Detection is primarily rule-based.
- Contextual understanding is limited.
- Some sensitive information may not match predefined patterns.
- The prototype currently supports Chrome only.
- The evaluation dataset was limited due to hackathon time constraints.

Future Work

Future improvements include:

- Extension Store Deployment
- Context-Aware Privacy Analysis using LLMs
- Enterprise Privacy Firewall
- Multi-Platform AI Protection
- Regulatory Compliance Support

6. Conclusion

We developed a browser-based Consent-Aware Privacy Firewall that helps users identify and manage sensitive information before interacting with AI systems. The system combines privacy detection, risk assessment, document scanning, and user-controlled masking within a local-first architecture.

By preventing accidental disclosure before information is transmitted, CAF demonstrates a practical approach to improving privacy and safety in generative AI workflows.

Code and Data

Include links if applicable. If your project doesn't involve code (e.g., policy analysis) or if there are info-hazard considerations, note that here.

- **Code repository:**
<https://github.com/reachmaheshkandagatla/consent-aware-privacy-firewall>
- **Demo Video:** <https://youtu.be/rqrc3ZKoMwc>
- **Architecture Diagram:** uploaded
- **Data/Datasets:** uploaded
- **Other artifacts:** uploaded

Author Contributions

Mahesh Kandagatla led project design, system architecture, privacy workflow design, implementation, testing, and documentation.

Bhargava Krishna Chiliveru contributed to implementation, validation, testing, and project review.

References

1. Microsoft Presidio. Data Protection and PII Detection Framework.
<https://github.com/microsoft/presidio>

LLM Usage Statement

We used ChatGPT to assist with brainstorming, architecture design discussions, documentation drafting, and refinement of project descriptions. All implementation details, results, and claims were reviewed and verified by the project authors.