PhishGuard AI - Browser Extension Detailed Documentation

This project was developed by:

- Tejas Santosh Paithankar (24BCY10104)
- Ashwin C (24BCY10218)
- Sudhanshu Singh (24BCY10410)
- Aashish Kumar Singh (24BCY10182)
- Niyati Agarwal (24BCY10293)

Introduction

PhishGuard AI is a security-focused browser extension developed to protect users from phishing websites in real-time. It leverages multiple security mechanisms including domain reputation checks, AI-based phishing detection, and SSL verification. The extension intercepts navigation events, analyzes URLs, and blocks malicious websites before the user can interact with them.

This document provides a detailed explanation of every component in the project, including manifest configuration, background processes, UI elements (popup and blocked page), supporting scripts, and styles. Additionally, it explains how the workflow ties all these components together.

Manifest File (manifest.json)

The manifest.json file defines the configuration of the Chrome extension. It specifies metadata, permissions, background services, content scripts, and the popup interface.

Key Sections:
- manifest_version: 3 (latest Chrome extension standard).
- name: Extension name (PhishGuard AI).
- version: 1.0.0.
- description: Explains that this extension uses AI to block phishing websites.
- permissions:
  * tabs → Required to access and analyze the URL of open tabs.
  * storage → For persisting allowlist, blocklist, and reported sites.
  * notifications → To alert users about blocked sites or confirmations.
  * webRequest → To monitor and intercept web requests.
- host_permissions: <all_urls> to allow URL scanning across all websites.
- background: Uses background.js as a service worker for URL monitoring.
- content_scripts: Injects content.js and style.css into every page to allow interaction with blocked.html.
- action: Defines popup.html as the extension's popup UI.
- icons: Provides extension icons in three resolutions (16, 48, 128).

Background Script (background.js)

This is the core logic of the extension, running as a background service worker. It listens to tab updates and performs security analysis whenever a user navigates to a new URL.

Main Responsibilities:
1. **Tab Monitoring**: Uses chrome.tabs.onUpdated to detect when a user loads a new page.
2. **URL Analysis**: When a new page loads, analyzeUrl() is called to determine if the URL is safe.

Steps inside analyzeUrl:
- Check allowlist first: If a site is allowlisted, it is ignored.
- Perform parallel checks:
    • Domain Reputation: Queries APIVoid's API to see if the domain is blacklisted.
    • AI Analysis: Calls Gemini API (LLM) to analyze the structure of the URL and determine if it is suspicious.
    • SSL Verification: Ensures that HTTPS is being used and that the SSL handshake is valid.
- Aggregates results. If one or more checks fail, mark as phishing.
- If phishing detected: Call blockAndAlert().

3. **Blocking & Notification**:
- Stores information about the blocked URL and reasons in Chrome storage.
- Redirects the tab to blocked.html.
- Sends a desktop notification to warn the user.

4. **User Reports**: The background script listens for messages from the popup (report button). When a user reports a site, it stores the URL locally and displays a thank-you notification.

5. **Utility Functions**:
- addUrlToList() → Adds domains to allowlist/reported lists.
- isUrlInList() → Checks if a given domain is in the allowlist or reported list.

Blocked Page (blocked.html)

This HTML page is displayed when a site is detected as phishing. Its purpose is to interrupt the browsing process and clearly inform the user about the risks.

Contents:
- Header with extension icon and title (Phishing Attempt Blocked).
- Dynamic content area:
    • Shows the blocked URL.
    • Lists the specific reasons (from AI, blacklist, SSL check).
- Warning message: Advises the user not to proceed.
- Actions:
    • Go Back to Safety → Returns user to the previous page.
    • Proceed Anyway (Not Recommended) → Allows user to bypass the block by adding the site to the allowlist.
- Linked with content.js for dynamic updates.

Content Script (content.js)

This script is injected into every page, but its key role is when blocked.html is loaded. It dynamically displays the details of why a site was blocked.

Functionality:
- Reads the blockedSiteInfo object (URL + reasons) from Chrome storage.
- Updates the blocked.html DOM elements with this data.
- Handles button actions:
  • Go Back: Uses history.back() to navigate away from the blocked site.
  • Proceed Anyway: Adds the hostname to the allowlist and reloads the original URL.

Popup Page (popup.html)

The popup is the user-facing dashboard that appears when the extension icon is clicked.

Features:
- Displays extension status (Protected).
- Report Current Site button → Allows user to manually report suspicious websites.
- Add to Allowlist button → Quickly adds the current domain to the allowlist.
- Allowlist Section:
  • Shows a list of all allowlisted domains.
  • Includes input field for manual domain entry.
  • Each entry has a remove button to delete from allowlist.

Popup Script (popup.js)

The logic behind the popup interface, handling user interactions and managing storage.

Responsibilities:
- Reporting: Sends a message to background.js with the current URL to mark it as reported.
- Allowlist Management:
  • Adds current site or manually entered domains to the allowlist.
  • Displays allowlist in real-time by reading from Chrome storage.
  • Allows removal of domains from the allowlist.
- User Feedback: Displays confirmation messages (success/info) when actions are taken.
- Ensures smooth user experience by dynamically updating the allowlist without requiring extension reload.

Styling (style.css)

The CSS file ensures the extension has a clean, professional look while reinforcing the security-focused theme.

Popup Styles:
- Container with padding, shadows, and rounded corners.
- Header with logo and title.
- Status bar with green background for safe browsing.
- Buttons styled with red (report), green (allow), and blue (add domain).

- Allowlist section styled with list items, remove buttons, and scrollable view.

Blocked Page Styles:
- Red-themed warning container with borders and shadows.
- URL highlighted in red box with word wrapping.
- Reasons displayed as a list.
- Buttons for safe navigation clearly separated.

Workflow Summary

The complete extension workflow is as follows:

1. User navigates to a new website.
2. background.js intercepts the URL and runs checks:
   - Is it on allowlist? → Skip.
   - Query APIVoid API for blacklist check.
   - Query Gemini AI for phishing characteristics.
   - Validate SSL/HTTPS.
3. Results aggregated → If suspicious, site is blocked.
4. Blocked.html is loaded, displaying URL and reasons.
5. User choices:
   - Go back → Leaves unsafe site.
   - Proceed anyway → Site is added to allowlist, bypassing future checks.
6. User can interact with popup.html to:
   - Report sites manually.
   - Manage allowlist (add/remove domains).
   - Check protection status.
7. Notifications are shown when sites are blocked or reported.