Comments and Development Plan for DILI Capstone Project

DILI: A Browser Extension for Detecting Illegal Link Injections in Edited Facebook Links and Malicious Redirects

1. Panel Comment

"Specify the methods for detecting illegal links in FB."

2. Proposed Detection Methods

To address the comment, we identified and specified clear methods that the DILI system will use for detecting illegal or malicious links:

a. Link Integrity Monitoring

- DILI records the original link of a Facebook post when first loaded.
- If the post is edited and the link changes, the extension compares the new link against the original.
- If a mismatch is detected, the post is flagged as suspicious.

b. Blacklist and API Checking

- DILI checks links against known malicious/phishing domains using APIs such as:
    - Google Safe Browsing API
    - PhishTank / OpenPhish databases
- If the link matches a known malicious entry, it is flagged.

c. Redirect Tracking

- Many phishing attacks rely on multiple redirects (e.g., shortened URLs).
- DILI checks if a link leads to multiple or hidden redirects before reaching the destination.

- Excessive or suspicious redirect patterns are flagged as unsafe.

d. Domain and Content Analysis (Optional/Future Enhancement)

- Check domain reputation, age, and SSL certificate validity.
- Detect mismatches between link previews (e.g., Facebook thumbnail) and the actual destination.

3. What Happens After Detection?

Detection alone is not sufficient. DILI also provides **prevention and protection** to users:

a. Real-Time Warning (While Scrolling)

- Suspicious links are marked with a warning badge or overlay (e.g., ⚠️ "This link may have been edited").
- Informs users **before they click**.

b. Click Interception (When User Clicks)

- DILI intercepts the request before the browser loads the target site.
- If flagged, the user is redirected to a **warning page**:

  ⚠️ *"This link may lead to a phishing or malicious site. It is not recommended to proceed."*

- The default option is **"Go Back"**, while "Proceed Anyway" is optional and requires confirmation.

c. Blocking Suspicious Redirects

- If the link triggers excessive or hidden redirects, DILI blocks the chain and alerts the user.
- Prevents hidden malicious payloads.

d. User Education & Logging

- Provide reasons for the warning (e.g., "Domain is newly registered" or "Link was edited after gaining engagements").

- Log unsafe links for user review to build awareness and trust.

4. Scope of Monitoring

- **DILI runs only when the Facebook tab is open.** It is a client-side protection tool, so if the user closes Facebook, DILI stops monitoring. Once the user reopens Facebook, DILI resumes real-time detection.

- **DILI is not limited to one section of Facebook.** Since the content script is injected across the entire facebook.com domain, DILI can scan:

    - Newsfeed (scrolling posts)

    - Shared posts (on timeline, groups, or pages)

    - Group discussions

    - Marketplace posts

- In short, DILI will detect malicious or edited links **wherever they appear on Facebook**.

5. Example Scenario: Facebook Post with Edited Link

- A post gains many likes and comments with a harmless link.

- Later, the link is edited into a fake Shopee promo (e.g., `sh0pee-promo.xyz`).

- **DILI detects the change** (original vs. current link) → marks post as ⚠️.

- If the user clicks, DILI intercepts → runs final safety check → identifies it as phishing → blocks access and warns the user.

6. How DILI Prevents Attacks

- **Proactive defense:** warns while scrolling.

- **Final defense:** intercepts and blocks unsafe clicks.

- **User awareness:** provides an explanation of why the link is unsafe.

7. Tools and Technologies

- **Browser Extension API:** Chrome Extension Manifest V3 (initial focus on Chrome/Edge/Brave).
- **Languages:** HTML, CSS, JavaScript.
- **Detection Tools:** Regex/DOM parsing, Google Safe Browsing API, PhishTank/OpenPhish API.
- **Development Tools:** VS Code, GitHub, Postman, Jest/Mocha for testing.
- **Testing Environment:** Facebook test accounts for controlled scenarios.

8. Development Coverage & Timeline

A 4–6 month development cycle is projected.

- **Month 1**: Requirements gathering, extension setup, environment preparation.
- **Month 2**: DOM scanning, original vs. edited link detection.
- **Month 3**: Integration with Google Safe Browsing / PhishTank APIs.
- **Month 4**: Warning UI, click interception, and logging system.
- **Month 5**: Testing across scenarios (newsfeed, groups, shared posts).
- **Month 6**: Security hardening, documentation, and defense preparation.

9. Structured Phasing

1. **Planning & Setup**: Study Chrome Extension APIs, GitHub repo, and dev environment.
2. **Core Link Detection**: Implement a DOM scanner, link integrity monitoring.
3. **Security Integration**: Connect with APIs, add a redirect checker.
4. **UI Development**: Overlays, extension popup, full warning page.
5. **Click Interception**: Block malicious links, confirm user action.
   Testing & Refinement: Test in all FB sections, optimize performance.
6. **Documentation & Finalization**: Write technical docs, prepare defense demo.