FIE COMPS:
An analysis on Malicious Browser Extension Detection & Classification Methods

Francisco Arenas, Isha Patel, Brandon Ekow Anderson
Carleton College
Winter Term 2026

FIE Comps Description:

Browser extensions enhance web functionality but often require powerful permissions that, if misused, can compromise user privacy and security. Malicious or poorly designed extensions may steal sensitive information, inject unauthorized scripts, track user behavior, or communicate with suspicious external servers.

This project aims to study how malicious browser extensions operate and to develop a security scanner that analyzes browser extensions and produces an interpretable security report. The scanner will perform static analysis on an extension's manifest (JSON), HTML, CSS, and JavaScript files within a secure Docker environment, searching for patterns and features commonly associated with malicious behavior.

The scanner will serve as the backend for a web-based interface that allows users to submit a browser extension (via URL) and receive a report detailing potential security risks, suspicious behaviors, and permission usage.

Learning Goals:

Through this project, we aim to learn:

1. Malicious Browser Extension Behavior
   a. How malicious browser extensions operate in practice, including data exfiltration, script injection, tracking, and abuse of privileged APIs
   b. Common attack techniques used by malicious or compromised browser extensions and how they manifest in extension source code
2. Malware Analysis Techniques
   a. The differences between static and dynamic malware analysis, including their strength, weaknesses, and appropriate use cases
   b. The limitations of static analysis when applied to obfuscated or dynamically loaded malicious behavior
3. Existing Tools and Research
   a. How existing tools and academic approaches detect malicious extensions and their limitations
4. Dataset Collection and Evaluation of Malware
   a. The challenges involved in collecting and running real-world malicious browser extensions/samples
   b. How sandboxing, containerization, and controlled environments are used to safely analyze untrusted code

Development Goals

By the end of this project, the software will provide the following features:

- A Docker-based environment that securely downloads and extracts browser extension files from a provided URL for analysis
- A scanner/parser that analyzes browser extension architecture files, including the manifest (JSON), JavaScript, HTML, and CSS
    - The scanner will identify and flag features commonly associated with malicious behavior
- Generation of a structured, human-readable security report that explains:
    - Detected suspicious features
    - Permission usage and risk levels
    - Why certain behaviors are flagged as potentially malicious
- Clear documentation describing the criteria and reasoning used to classify features as malicious or suspicious, including references to known attack patterns and prior research
- A functional website that allows users to submit a browser extension via a URL and receive a detailed malicious behavior report and permission analysis
- **[Stretch Goal]** Integration of a sandboxed or simulated environment to observe runtime behaviors, such as network requests or script execution, that may not be detectable through static analysis alone
- **[Stretch Goal]** Create and train a machine learning model (e.g., KNN or similar ML algorithm) to predict the likelihood that a browser extension is malicious based on extracted features

Benchmark Testing

Docker Environment/ Virtual Machine:

- Functionality test:
    - Can we download an extension package?
    - Can we send the installed files to our parser to parse?

Scanner Testing:

- We plan to compare the accuracy of our malicious flagging with other existing tools, like ExtAnalysis, as well as against results reported in prior academic studies.
    - From these existing tools, we will feed our tool the same malware and verify the results with known malicious extensions

Webpage Testing:

- To test the effectiveness and simplicity of our website, we could have user-testing phases with friends who know nothing about our comps project
    - If they can successfully feed in an extension, understand the security report, and make an educated next step, then we would have succeeded

Schedule of Development

Week 3

- ☐ Meet with Mike Tie to discuss the most effective setup for a Docker environment for our use case - Tuesday @3
  - ☐ Pivot to a Virtual Machine if needed, Mike suggests we use a VM
- ☐ Research and curate a malicious portfolio of common malicious tags and features
- ☐ Start creating a parser
  - ☐ Must be able to unpack and read different files in the extension architecture

Week 4 (Demo Test Week)

- ☐ Use Figma to create a web UI design
- ☐ Implement a surface-level website with the ability to download an extension on the Docker environment for testing/scanning
- ☐ Continue to work on the parser
  - ☐ Scan and search for specific tags and features from the malicious characteristics portfolio

Week 5

- ☐ Starting coding/creating a full website with all envisioned functionality
- ☐ Create a security report risk scoring system that lets users know the risk level of extensions based on flagged malicious behaviors by the parser.
- ☐ Have a finished/ fully working parser

Week 6

- ☐ Evaluate the scanner in terms of its effectiveness in analysing patterns and giving a precise and interpretable security report
- ☐ Have a fully functional website with parser, Docker, and security report integrated
- ☐ Have friends test/ and use our website. Integrate constructive feedback

Week 7

- ☐ Shift down previous unfinished work/improvements into this if we need

Week 8 (Thursday Code Freeze)

- Finalize documentation and compose a comps presentation