

Beyond Permissions: Multi-Layer Monitoring and Forensic Behaviour Graph Analysis for Detecting Hidden and Collusive Chrome Extensions

Name: Ali Azman

ID: 2423415

A dissertation submitted in partial fulfilment of the requirements for the degree of:
MSc Cybersecurity and Digital Forensics

Cyber Security and Digital Forensics
University of Wales Trinity Saint David

Supervisor: Dipo Dunsin
Submission Date: 02/02/2026

DECLARATION OF ORIGINALITY

I declare that this dissertation is my own work and has not been submitted in any form for another degree or diploma at any university or other institution. Where information has been derived from other sources, I confirm that this has been indicated in the dissertation by appropriate referencing.

Signed: _____Ali Azman_____

Name: Ali Azman

Date: 02/02/2026

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to **Dipo Dunsin** for their guidance, support, and constructive feedback throughout this dissertation. I also thank **University of Wales Trinity Saint David** for providing the academic resources required for this research. Finally, I acknowledge the support of my family and friends for their encouragement during the completion of this dissertation.

ABSTRACT

Browser extensions play an important role in delivering additional functionality while also exposing a major threat landscape. Current permission-based security models have been demonstrated to be inadequate because permissions indicate potential capabilities rather than actual capabilities. Current malicious extensions have been shown to be difficult to detect using techniques such as delayed activation, low permission misuse, dataflow chain construction, and collusion between multiple extensions. In this dissertation, a multi-layer runtime monitoring system that monitors extensions at the DOM, network, storage, and messaging layers and incorporates this data into a multi-layer forensic behavior graph is proposed and evaluated.

The system design utilizes a manifest V3 runtime monitoring extension and a backend system that analyzes the data and stores it in a forensic database. Suspicious behavior motifs P1-P5 and collusion indicators C1-C5 are identified within time windows. Graph-based evidence subgraphs are also constructed. Experiments were conducted on a dataset consisting of benign extensions, malicious extensions with hidden behaviors, and a collusive extension pair. Results show that the proposed system has an accuracy of 87.0%, precision of 83.3%, recall of 86.7%, and an F1 score of 85.0%. These results show that the proposed system performs better than the permission-based system (45.0% accuracy and 40.0% F1 score) and the single-layer runtime system (62.0% accuracy and 56.0% F1 score). In addition, the system has a 100% detection rate for the collusive extensions. Operationally, the system has been shown to be feasible with a relatively small overhead on the browser and sub-second detection latency for the majority of the extensions.

The results show that the proposed system is valid because multi-layer correlation and interaction-aware modeling improve detection coverage. In addition, the use of a graph-based system improves the interpretability and forensic value of the results. Future work will include further evaluation with larger sets of extensions. In addition, the system will be improved to be more robust against adversarial timing jitter and staged exfiltration. Finally, the system will be extended to include graph learning while maintaining its explainability.

Keywords: Browser extensions, Chrome manifest V3, runtime monitoring, collusion detection, forensic analysis, graph-based security.

TABLE OF CONTENTS

Declaration of Originality	2
Acknowledgements	2
Abstract	3
List of Figures	6
List of Tables	6
List of Abbreviations	6
Chapter 1: Introduction	8
1.1 Background and Motivation	8
1.2 Problem Context and Scope	9
1.3 Objectives of the Study	10
1.4 Research Questions and Hypotheses	10
1.5 Contributions of the Study	11
1.6 Organization of the Paper	11
Chapter 2: Literature Review and Research Gaps	12
2.1 Browser Extension Ecosystem and Threat Landscape	12
2.2 Limitations of Permission-Based Security Models	13
2.3 Detection Approaches for Malicious Browser Extensions	15
2.4 Multi-Layer Behaviour Monitoring and Web Threat Detection	16
2.5 Behavioural Sequence, Pattern-Based and Forensic Methods	17
2.6 Synthesis of Cross-Domain Limitations, Research Gaps, and Recommendations	18
Chapter 3: Research Methodology	24
3.1 Chapter Overview	24
3.2 Research Questions and Objectives	24
3.3 Research Design	25
3.4 Threat Model and Assumptions	27
3.5 System Overview and Instrumentation	28
3.6 Event Schema and Forensic Database	29
3.7 Behaviour-Permission Correlation	30
3.8 Hidden Threat Detection Methodology (P1-P5)	30
3.9 Collusion Detection Methodology (C1-C5)	31
3.10 Multi-layer Forensic Behaviour Graph (MFBG)	32
3.11 Experimental Setup and Dataset	32
3.12 Evaluation Metrics and Baselines	33
3.13 Validity and Reliability	33
3.14 Ethical Considerations and Data Protection	33
3.15 Chapter Summary	34
Chapter 4: Implementation	35
4.1 System Components and Modules	35
4.2 Chrome Extension Implementation (Manifest V3)	37

4.3 End-to-End Data Flow	38
4.4 Backend Implementation (Flask + Analysis Modules)	39
4.5 Codebase Structure	40
4.6 Forensic Database Implementation	40
4.7 API Design	41
4.8 Deployment Architecture	42
4.9 Chapter Summary	43
Chapter 5: Results and Analysis	44
5.1 Dataset Description	44
5.2 Experimental Scenarios	46
5.3 Quantitative Results	48
5.4 Qualitative Analysis and Case Studies	51
5.5 Comparative Evaluation with Baselines	54
5.6 Discussion of Findings	56
5.7 Chapter Summary	57
Chapter 6: Discussion	58
6.1 Interpretation of Results	58
6.2 Implications for Practice and Research	59
6.3 Strengths and Contributions	59
6.4 Limitations of the Study	60
Chapter 7: Conclusion and Recommendations.....	61
7.1 Summary of the Study	61
7.2 Conclusions Drawn	61
7.3 Recommendations	62
7.4 Directions for Future Research	63
References	64
Appendices	67

LIST OF FIGURES

Figure 2.5: Behaviour Sequence vs Forensic Graph Approach
Figure 3.1: Research Methodology Flow (DSR + Experimental Validation)
Figure 3.2: Threat Model Overview
Figure 3.3: Multi-Layer Monitoring Architecture
Figure 3.4: Detection Flow Diagram (ThreatDetector + CollusionDetector + GraphBuilder)
Figure 3.5: Database Schema Diagram (logs / extensions / threats)
Figure 4.1: System Architecture Overview
Figure 4.2: Data Flow Diagram
Figure 4.3: Component Interaction Diagram
Figure 4.4: Technology Stack Diagram
Figure 4.5: File Structure Tree
Figure 4.6: Database Schema Diagram
Figure 4.7: API Endpoints Summary
Figure 4.8: Deployment Architecture
Figure 5.1 (placeholder): Event distribution by layer (DOM / Network / Storage / Messaging).
Figure 5.2 (placeholder): Example MFBG evidence subgraph for a hidden-behaviour detection.
Figure 5.3 (placeholder): Evidence subgraph for delayed activation chain (DOM → Storage → Network).
Figure 5.4 (placeholder): Collusion evidence subgraph (messaging + shared storage + sequential dependency).
Figure 5.5: Runtime Performance and Overhead Summary

LIST OF TABLES

Table 2.1. Comparative analysis of existing approaches in browser extension security, behavioural monitoring, and web threat detection
Table 3.1: Traceability from literature gaps to methodology and evaluation
Table 3.2: Threat Detection Patterns (Hidden Threat Patterns + Collusion Patterns)
Table 5.1: Dataset composition
Table 5.2: Overall detection metrics (extension-level)
Table 5.3: Hidden-threat motif performance
Table 5.4: Collusion indicator outcomes
Table 5.5: Operational overhead metrics
Table 5.6: Comparison of proposed framework with baselines

LIST OF ABBREVIATIONS

API — Application Programming Interface
CDN — Content Delivery Network

CORS — Cross-Origin Resource Sharing
CPU — Central Processing Unit
DOM — Document Object Model
DSR — Design Science Research
F1 — F1-score
FN — False Negative
FP — False Positive
HTTP — Hypertext Transfer Protocol
JSON — JavaScript Object Notation
MFBG — Multi-layer Forensic Behaviour Graph
MV3 — Manifest V3 (Chrome Extension Manifest Version 3)
POST — HTTP POST method
RQ — Research Question
SQL — Structured Query Language
SQLite — Lightweight embedded relational database
TN — True Negative
TP — True Positive
URL — Uniform Resource Locator
XSS — Cross-Site Scripting

1. Introduction

This chapter introduces the context and motivation for investigating browser extension threats that cannot be reliably identified through permission-centric signals. It outlines the problem scope, research objectives, and research questions, and summarises the contributions and structure of the dissertation.

1.1 Background and Motivation

Browser extensions have become a common phenomenon in the contemporary web. They play a vital role in increasing productivity, accessibility, and privacy while improving the user experience. However, the privileges provided by the browser to the extensions—access to web content, user interaction streams, and browser application programming interfaces—have increased the attack surface. Unlike other web attacks that are limited to a single site and session, extensions operate across sites and sessions and have access to the web content.

The problem with browser extensions has been that the extension ecosystem has traditionally relied on permission-centric trust signals. Although users base their decisions on permission requests and store reputation, studies have shown that users are not able to effectively interpret the implications of the permission request and the associated risks [2]. More importantly, permission signals reflect the potential capability of the extension rather than actual behavior. This has allowed attackers to request minimal privileges, provide legitimate-looking extensions, and execute malicious code after a time delay or page-category trigger. This behavior of activating malicious code after a time delay and the divergence of extension behavior from permission have been shown to be common in malicious extensions in contemporary browser architectures [4].

Extensions can also cause privacy and security damage without being “malicious” in the traditional sense. Tracking, data leakage, and fingerprinting can be either incidental or intentional. In the case of granular extension fingerprinting, researchers have shown that extension configurations can be a source of uniquely identifying information, which further supports the need to monitor extensions to distinguish between expected utility and unwanted externalities. However, extensions can also be designed to be protective. For example, extension-based mechanisms can be designed to mitigate web threats such as cross-site scripting, which further supports the idea that extensions can be both defenders and attackers depending on the design and intent. This further supports the need for runtime evidence, as purpose is not always a good indicator of behavior.

Many defensive techniques are single-layer or narrow in scope. Runtime blockers can be designed to detect certain overt patterns, but attacks can be distributed across multiple layers, such as DOM interaction, storage, and communication. Each of these can be individually innocuous, which further supports the need for correlation-based techniques to explain the combination of otherwise ordinary behavior. In addition, the sheer volume of legitimate DOM mutation and communication in modern web applications can further muddy the waters for volume-based techniques.

Another gap is the collusive threat model, where the impact of the threat is only evident through the interaction of a number of extensions. In this case, each extension is not suspicious on its own or is only slightly suspicious, but the interaction of the extensions is a complete threat chain through messaging, artefact sharing, or time ordering. The permission model does not provide a way of representing this interaction, and analysis of each extension in isolation can fail to detect a threat that needs reasoning and interaction modeling between the actors [4], [10]. This leads us to the need for correlation of the evidence from the extensions.

1.2 Problem Context and Scope

In this dissertation, the issue of browser extension security will be discussed, with special attention to the threats that cannot be detected using permission prompts, extension declarations, and single-layer monitoring approaches. The scope of the research will be Google Chrome extensions with the MV3 manifest, as it is the most popular browser extension environment and the MV3 manifest includes certain limitations that affect the attacker model and the defendable monitoring model, particularly in terms of service-worker-based background execution, which impacts the extension lifecycle, background persistence, and monitoring, as mentioned in [32].

Two threat classes are in scope:

1. Hidden malicious behaviors. The term “hidden” refers to malicious behaviors not detectable or not reliably inferable from permissions, such as those made obscure through timing, conditionality, staged execution, or minimal privilege. For example, an extension might subtly manipulate the DOM, such as injection-like page changes, storage key access, and external data transmission. Each behavior might seem normal on its own, but correlation within a certain time window might reveal a suspicious behavior pattern and a plausible malicious intent [2], [4].

2. Collusive attacks. The term “collusive” refers to coordinated attacks between multiple extensions, with a coordinated malicious workflow. For example, an attack might involve multiple extensions working together, with each extension’s role divided, thereby reducing suspicion per extension. In this case, analyzing a single extension might not reveal the full extent of the attack, as responsibilities are divided among attackers, reducing suspicion per extension and permission-based inference [4], [10].

To facilitate cross-layer correlation and multi-actor reasoning, the implemented system monitors the following: DOM events, which include mutations and injection-like events; network events, which include observable destinations and timing; storage events, which include access patterns and key usage; messaging events, which include runtime communication indicators; and the temporal dimension, which includes event ordering and time window correlation. The collected telemetry data is stored in a forensic store, where it is converted to a Multi-Layer Forensic Behavior Graph (MFBG) data structure, in which extensions, events, and resources accessed are represented as nodes, and temporal, cross-layer, and interaction relationships among these nodes are represented as edges.

The study is scoped to ensure the research is feasible and to ensure the results are clear and ethical in nature. The study does not involve scanning the ecosystem without consent, deanonymizing legitimate users, or developing a taxonomy of malware. Rather, it aims to determine whether it is possible for an evidence-based framework to detect relevant information within the MV3 ecosystem, detect hidden and collusive patterns, and produce evidence structures that can be used for analysis and validation.

1.3 Objectives of the Study

The primary objective is to develop and evaluate a multi-layer forensic behaviour monitoring system that captures runtime extension activity and supports evidence-based detection of hidden and collusive threats.

Secondary objectives are:

- **O1:** Design a multi-layer monitoring framework capturing DOM, network, storage, messaging, and temporal dimensions under Chrome MV3 constraints [32].
- **O2:** Develop hidden-behaviour detection mechanisms that identify suspicious cross-layer motifs (e.g., injection-to-network pathways and storage-to-exfiltration chains), addressing the limits of permission-only reasoning [2], [4].
- **O3:** Develop collusion detection mechanisms that correlate multi-extension signals (e.g., messaging frequency, shared artefacts, linked infrastructure, and sequential dependencies), enabling detection of distributed attack pipelines [4], [10].
- **O4:** Construct an MFBG that represents extensions, events, and resources as nodes and encodes temporal, cross-layer, and interaction relationships as edges, supporting explainable, evidence-backed outcomes [16], [34].
- **O5:** Evaluate effectiveness and feasibility by measuring detection performance (precision, recall, F1) and system overhead (latency and resource cost) relative to permission-based and single-layer baselines [2], [8], [22].

1.4 Research Questions and Hypotheses

RQ1: Can multi-layer behavioural monitoring improve detection accuracy compared to permission-based approaches?

H1: Multi-layer monitoring will improve detection effectiveness (higher F1-score) over permission-based baselines because it captures runtime evidence chains that permission prompts cannot represent reliably [2], [4].

RQ2: How can coordinated attacks between multiple extensions be detected effectively?

H2: Interaction-aware correlation (messaging, shared artefacts, and sequential dependencies within bounded windows) will detect collusive behaviours that remain undetected when extensions are analysed in isolation [4], [10].

RQ3: What is the effectiveness of graph-based analysis for extension forensics?

H3: Representing behaviour as a multi-layer graph will improve interpretability and investigative utility by enabling compact evidence subgraphs that reconstruct the behavioural chain supporting each alert, aligning with forensic expectations for traceability [16], [34].

RQ4: How does behaviour–permission correlation contribute to threat detection?

H4: Behaviour–permission divergence will support prioritisation: observed actions that exceed what is reasonably expected from declared permissions will correlate with higher likelihood of malicious intent, reinforcing the limits of permission-centric trust models [2], [4].

1.5 Contributions of the Study

This dissertation makes four primary contributions:

1. A multi-layer forensic behaviour monitoring framework for Chrome extensions under MV3, capturing DOM, network, storage, messaging, and temporal evidence to support cross-layer correlation beyond static declarations [32].
2. A collusion-aware detection approach that models multi-extension coordination through messaging, shared artefact overlap, linked infrastructure, and sequential action chains, targeting distributed threats that evade per-extension analysis [4], [10].
3. An MFBG representation that encodes actors, events, and resources with temporal and interaction relationships, enabling evidence-backed explanation and reconstruction rather than opaque alerts [16], [34].
4. An empirical evaluation that compares the proposed approach against permission-based and single-layer baselines and reports both detection effectiveness and operational overhead, supporting defensible claims about practicality [2], [8], [22].

1.6 Organization of the Paper

In Chapter 2, the literature review is provided, which synthesises the research gaps to justify the need to look beyond permission-based analysis and individual metrics. In Chapter 3, the methodology is described, which includes the system architecture, data collection, detection logic for hidden and collusive behaviour, behaviour graph generation, and evaluation. In Chapter 4, the implementation of the MV3 monitoring extension and backend analysis pipeline is provided. In Chapter 5, the results are presented, which include the analysis of the performance of the system in different scenarios, such as benign, malicious, and collusive behaviour, along with the baseline results. In Chapter 6, the implications of the work are provided, followed by the conclusions in Chapter 7.

2. Literature Review

In this chapter, existing literature on browser extension security, behavior-based threat detection, web forensics, and the concept of anomaly detection paradigms will be discussed. Based on the literature, the weaknesses in permission-based security, single-layered behavior monitoring, and the absence of a unified forensics representation of multiple extension interactions will be highlighted. Each subsection will conclude with Research Gaps and Recommendations, as shown in the demo you provided.

Summary Table (Simple Overview)

Chapter Section Supporting Papers

2.1	3, 4, 5, 7, 10, 13, 15, 18
2.2	2, 4, 7, 8, 10
2.3	1, 5, 6, 7, 8, 9, 10, 12
2.4	1, 6, 7, 9, 11, 12, 13, 14, 15
2.5	12, 13, 14, 16, 18
2.6	All papers [1–19]

2.1 Browser Extension Ecosystem and Threat Landscape

Browser extensions have evolved into powerful modules capable of modifying webpage content, intercepting traffic, accessing storage, and interacting deeply with browser internals. Their increasing sophistication has widened the attack surface. Numerous studies demonstrate how malicious browser extensions (MBEs) exploit privileged APIs to perform spying, phishing, keylogging, ad fraud, session hijacking, and large-scale data exfiltration [1], [4], [10]. Research in 2025 confirms that even under strengthened security frameworks such as Chrome’s Manifest V3 and Firefox’s tightened WebExtension standards, attackers can still publish and execute harmful extensions in official stores by using obfuscation, delayed activation triggers, and misleading descriptions [4].

Several high-profile incidents (2013–2024) show a trend toward increasingly coordinated malicious campaigns leveraging extensions for DDoS attacks, affiliate fraud, and credential theft [4]. Studies also highlight the emergence of **supply-chain compromise**, where attackers hijack developer accounts to push malicious updates [4]. Beyond direct attacks, extension ecosystems also contribute to privacy erosion: extension fingerprinting can uniquely identify users with high granularity across websites [3].

Meanwhile, security-audit-oriented extensions aim to detect risky permissions or malicious code, but their scope is limited to static analysis or shallow runtime monitoring [2]. Defensive extensions (e.g., NoScript, SafeToOpen) provide partial protection but remain reactive and do not monitor extension-to-extension interaction patterns [5].

Research Gaps Identified (2.1)

While existing research outlines the various ways malicious browser extensions can misuse powerful APIs, user data, and web interfaces, the current body of work remains limited in how it interprets these threats as part of a broader behavioural ecosystem. Most studies examine specific attack vectors in isolation, which prevents a deeper understanding of how extensions may combine multiple behaviours across the DOM, network, and storage layers to produce more sophisticated or hidden threats. In addition, the literature pays little attention to the possibility of collusive behaviour between extensions, even though coordinated malicious actions have been observed in real-world campaigns. Another shortcoming is the limited focus on low-permission extensions, whose subtle misuse of benign-appearing capabilities increasingly undermines traditional permission-based risk assumptions. Finally, existing studies offer minimal support for reconstructing attacker behaviour over time, making it difficult to trace the sequence of events that led to compromise or to attribute malicious actions to specific extensions with confidence.

Recommendations (2.1)

Future research should incorporate a more holistic approach that examines extension behaviours across multiple browser layers simultaneously rather than treating each action in isolation. Extending analysis to include potential interactions between extensions would also help identify collusive or distributed threats that current models fail to capture. As low-permission extensions continue to pose significant risks, detection approaches should rely more on behavioural evidence than permission declarations alone. Integrating mechanisms for reconstructing behavioural timelines would further enhance both real-time detection and post-incident forensic analysis by revealing how malicious activities unfold and how different components contribute to the attack sequence.

2.2 Limitations of Permission-Based Security Models

Most modern browsers rely on permission prompts and store vetting procedures. However, multiple studies demonstrate that users routinely misunderstand permission requests and approve dangerous extensions without recognising the associated risks [2]. More critically, research repeatedly confirms that permissions do not reflect actual runtime behaviour: MBEs

can perform harmful actions using minimal permissions such as activeTab or storage, circumventing both user awareness and automated review [4], [7].

Some systems, such as vulnerable-extension detectors and permission-auditing tools, attempt to identify risky permissions, but they fail to capture behaviour triggered through indirect channels such as content scripts, dynamically injected code, or cooperative extensions [2], [11].

Manifest V3 enforcement attempts to limit dangerous APIs by replacing webRequest with declarativeNetRequest rules, yet attackers simply adapt by using DOM-based script injection, cross-extension messaging, or delayed runtime activation [4]. Studies confirm that permissions alone are poor predictors of malicious activity, making permission-centric approaches insufficient for modern threat landscapes.

Research Gaps Identified (2.2)

Although permission-based security models remain the foundation of browser extension governance, current research shows that permissions alone offer an incomplete and often misleading picture of actual runtime risks. Many studies highlight how extensions with minimal or seemingly harmless permissions can still perform harmful actions by exploiting generic APIs, content scripts, or contextual privileges. However, the literature rarely examines how permissions and real behaviour diverge in practice or how attackers intentionally design low-permission extensions to evade user scrutiny and automated analysis. Additionally, browser store vetting still relies heavily on static review processes, which are frequently bypassed through obfuscation, delayed activation, or remote configuration updates. User studies show persistent confusion around permission warnings, yet little work has been done to operationalise these findings into improved detection or enforcement mechanisms. Overall, the current research landscape lacks a behaviour-grounded understanding of how permissions relate to real-world malicious activity.

Recommendations (2.2)

Future approaches should shift from static permission analysis toward dynamic behaviour correlation, enabling security systems to identify when an extension's actions exceed or contradict its declared capabilities. This would allow deviations between permissions and behaviour to function as early indicators of malicious intent. Research should also explore mechanisms for continuous or context-aware permission enforcement, rather than relying solely on installation-time consent. Enhancing the interpretability of permission systems and integrating runtime behavioural audits can help reduce user misunderstanding and reinforce trust. Ultimately, a more behaviour-centric model is required to compensate for the inherent limitations of permission declarations.

2.3 Detection Approaches for Malicious Browser Extensions

Research proposes multiple detection categories:

Static Analysis Approaches

Static code inspection, manifest analysis, and permission evaluation remain common, but obfuscated code, dynamic script loading, and environment-dependent behaviours undermine their reliability [2], [4].

Dynamic and Runtime Monitoring

Systems like MJBlocker analyse XPCOM call sequences to detect malicious JavaScript extensions using automata-based detection [8]. Runtime monitoring generally outperforms static methods but is usually **single-layer**, focusing only on API sequences or network calls without integrating DOM or storage activity.

Machine Learning Approaches

Hybrid ML models combining static and dynamic features achieve high reported accuracy but still lack robustness against adversarial evasion and collusive patterns [6], [12].

Defensive Extensions

Evaluations show that defensive add-ons can block some phishing and exploitation vectors, but they act primarily as **web-content defenders**, not detectors of malicious extensions [5].

Overall, existing approaches do not construct multi-entity relationships or reason about cross-extension collaboration, temporal behaviours, or forensic reconstruction.

Research Gaps Identified (2.3)

Although numerous detection approaches exist—ranging from static analysis to dynamic monitoring and machine learning—their effectiveness is limited by the narrow behavioural dimensions they observe. Static analysis struggles with obfuscation and cannot reveal behaviours triggered under runtime conditions. Dynamic or API-level monitoring improves visibility but typically examines only one category of activity, such as API calls or network requests, without integrating signals across layers. Machine learning models offer promising accuracy but rely heavily on the quality of training data and often lack interpretability, making them difficult to use for forensic or regulatory purposes. Furthermore, few studies consider the possibility that multiple extensions, or extensions and web pages together, may coordinate behaviours, which significantly reduces the ability to identify collusion or multi-stage attacks. As a result, current detection systems do not capture the full behavioural context needed to understand and detect sophisticated malicious extensions.

Recommendations (2.3)

Research should advance toward multi-layer, context-aware detection models that combine insights from API usage, network activity, DOM modifications, storage events, and inter-extension communication. This integrated perspective would enable the identification of

coordinated behaviours and reveal patterns inaccessible through single-layer analysis. Machine learning efforts should prioritise interpretability to ensure that detection outcomes support forensic investigation. Incorporating robust cross-layer correlation techniques will allow detection systems to recognise complex behaviour patterns and collusive activities that existing models overlook.

2.4 Multi-Layer Behaviour Monitoring and Web Threat Detection

Extensive research exists on phishing detection, malicious URL analysis, and DNS-based anomalies. URL classification via ML (SVM, RF, GRU) shows strong performance but targets **external web threats**, not malicious extensions [6], [7]. Domain linkage graphs reveal malicious infrastructure patterns [11], and anomaly-detection frameworks show promise for identifying malicious URLs [15].

Client-side detectors (XSS, SQL injection, JS anomaly detectors) demonstrate the power of local behavioural instrumentation [1], [9]. However, these approaches treat webpages as isolated entities and do not incorporate extension behaviour as a first-class threat vector.

These works inspire key ideas:

- Graph representations can expose hidden infrastructures
- Behaviour sequences can uncover anomalies
- Client-side instrumentation provides deep visibility

Yet none combine these insights into a unified model that monitors *browser extension behaviour* across multiple channels simultaneously.

Research Gaps Identified (2.4)

Studies on web threat detection—including phishing detection, URL classification, domain analysis, and anomaly detection—provide strong evidence that multi-feature behavioural analysis is effective for identifying malicious activity. However, these techniques largely focus on external web threats rather than extension-driven ones, limiting their applicability to the browser extension ecosystem. While graph-based methods and anomaly detection have shown success in analysing malicious URLs and infrastructural linkages, they are rarely applied to client-side behaviours or browser-level interactions. As a result, current research underutilises the potential of cross-layer behavioural data—such as DOM events, API interactions, and storage access—to construct richer behavioural models. This gap becomes more pronounced as malicious extensions increasingly blend their actions with normal web activity, making single-source detection approaches less effective.

Recommendations (2.4)

The strengths of multi-layer monitoring and graph-based analysis should be adapted to the domain of browser extension security. By aggregating behavioural signals from different browser subsystems and modelling them within a unified graph structure, researchers can

uncover patterns that would be invisible when analysing each layer independently. Integrating these multi-source signals can help distinguish benign from suspicious behaviours with greater precision, especially in cases where malicious extensions mimic common web actions. This approach would support a deeper behavioural understanding and enable earlier detection of disguised or evolving threats.

2.5 Behavioural Sequence, Pattern-Based and Forensic Methods

Behaviour sequence detectors model malicious software using event sequences, but they remain limited to **single-process analysis** [14]. Pattern-matching under adversarial conditions enables detection of known signatures but lacks adaptability to novel sequences [12].

Browser forensics studies demonstrate effective recovery of artefacts from RAM and disk images, revealing user activity even in private browsing [13]. Yet these approaches operate **post-incident**, lacking real-time capability and failing to attribute behaviours to specific extensions or sessions.

Recent explainable phishing frameworks (e.g., PhishXplain) show how browser extensions can stream telemetry to a backend for AI-based analysis [16]. This architecture inspires real-time forensic reconstruction but still focuses on webpage content, not extension behaviour.

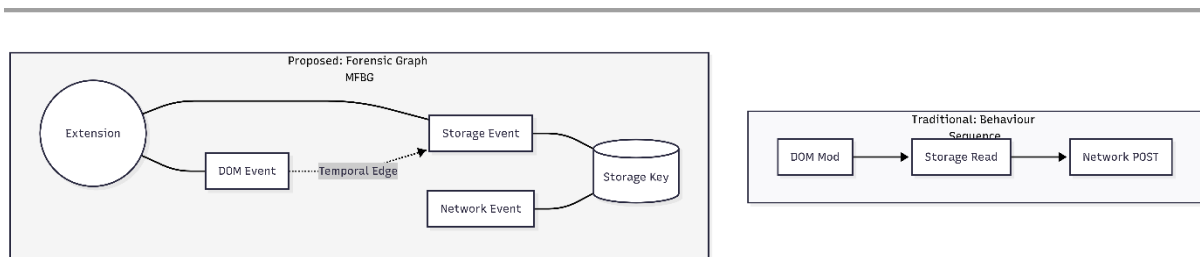


Figure 2.5 — Behaviour Sequence vs Forensic Graph Approach

Research Gaps Identified (2.5)

Behaviour-sequence and pattern-based detection techniques provide valuable insights into how malicious activity unfolds but often limit their analysis to linear or single-process event streams. While these approaches effectively capture repetitive or well-defined malicious patterns, they struggle to accommodate complex, branching behaviours that involve multiple actors or components. In addition, behavioural sequence models rarely incorporate temporal context or cross-entity relationships, which are essential for detecting collusive or multi-stage attacks. Forensic research, meanwhile, tends to focus on recovering artefacts after an incident has occurred, offering limited support for real-time detection or continuous behavioural understanding. This separation between live monitoring and forensic investigation leaves both detection and response systems without the broader behavioural continuity needed to fully interpret extension-based threats.

Recommendations (2.5)

Future solutions should move beyond linear behaviour sequences and adopt models that accommodate multi-entity, multi-layer interactions. Graph-based representations offer a natural way to express these relationships, allowing investigators to trace how behaviours propagate across extensions, web pages, and browser components. Integrating real-time monitoring with forensic data retention would create a unified behavioural timeline, supporting both immediate detection and thorough post-incident analysis. Such an approach would enable more accurate threat attribution and provide comprehensive insight into the structure and evolution of malicious activity.

2.6 Synthesis of Cross-Domain Limitations, Research Gaps, and Strategic Recommendations

Table 2.1. Comparative analysis of existing approaches in browser extension security, behavioural monitoring, and web threat detection

Category	Research Work	Contribution	Benefits	Drawbacks	Integration & Impact
Browser Extension Threat Analysis	(Singh et al., 2025) [4]	Demonstrated real-world malicious browser extensions bypassing Chrome/Firefox vetting	Reveals real-world exploitation techniques; informs realistic threat models	Limited to case studies; lacks automated detection framework	High
Browser Security & Permission Models	(Chang, Gudla & Kapadia, 2020) [2]	Evaluated user understanding and weaknesses of extension permission prompts	Highlights key usability and security issues in permission systems	Does not provide behavioural correlation or runtime insights	Medium
Extension Fingerprinting / Privacy	(Bauer et al., 2019) [3]	Showed how extension configurations can uniquely fingerprint users	Exposes privacy risks linked to extension ecosystems	Does not address malicious extension behaviour	Low

Category	Research Work	Contribution	Benefits	Drawbacks	Integration & Impact
Security Risks of Browser Extensions	(Sam et al., 2019) [7]	Framework for analysing extension vulnerabilities and runtime risks	Supports vulnerability identification and permission-risk assessments	No support for multi-layer behavioural reconstruction	Medium
Runtime Malicious JS Detection	(MJBlocker, 2018) [8]	DFA-based model for detecting malicious JavaScript behaviour in extensions	Lightweight runtime detection; efficient pattern matching	Single-layer (XPCOM only); not collusion-aware	Medium–High
XSS Detection via Browser Extensions	(Satrio et al., 2020) [1]	Real-time XSS detection using DOM and request monitoring	Strong for web-layer attacks	Only detects page-level threats; not extension-level behaviour	Low
SQL Injection Detection (LSTM)	(S.I. Sani et al., 2021) [9]	Browser extension using LSTM to detect SQLi payloads	Leverages ML for dynamic detection	Limited to SQLi; unrelated to extension malicious behaviour	Low
Phishing Detection (Machine Learning)	(Olatunji et al., 2020) [6]	ML classifiers for phishing website detection	Strong accuracy on URL features	Vulnerable to evasion; not linked to extension behaviour	Low
Real-Time Phishing Detection (GRU)	(Kumari et al., 2021) [7]	GRU-based detector with high accuracy for phishing URLs	Real-time classification	Does not integrate with browser runtime or	Low–Medium

Category	Research Work	Contribution	Benefits	Drawbacks	Integration & Impact
				extension signals	
Malicious URL Linkage Patterns	(Sharma & Pande, 2022) [11]	Graph-based linkage analysis revealing malicious infrastructures	Demonstrates value of graph models	URL-only; behaviour graph for extensions not explored	Medium
Anomaly-Based URL Detection	(Batra et al., 2021) [15]	Clustering + anomaly detection for malicious URLs	Effective at unsupervised detection	No multi-layer behavioural context; not browser-linked	Medium
GAN-based Malicious Domain Evasion	(Lin et al., 2020) [12]	Using GANs to produce adversarial malicious domains	Shows ML vulnerability	Not applicable to extension ecosystems	Low
Malicious Behaviour Sequencing	(Yan et al., 2020) [14]	Behaviour sequence model for malware detection	Supports sequential analysis	Sequence-only; lacks graph-level multi-entity modelling	Medium
Forensic Browser Artefact Recovery	(Sanghvi et al., 2024) [13]	RAM + disk forensics of Firefox normal vs private mode	Reveals valuable browser artefacts	Offline; cannot detect runtime extension behaviour	Medium
Extension-Based Phishing Defences	(Blancaflor et al., 2025) [5]	Assessed extension effectiveness against phishing & exploitation	Identifies strong defensive additions	Evaluates detection, not extension-misbehaviour	Medium
Local-Layer Explainable Phishing Extension	(Roy et al., 2024) [16]	LLM-based explanation for phishing indicators	Enhances user understanding	Limited to webpage phishing; not	Medium

Category	Research Work	Contribution	Benefits	Drawbacks	Integration & Impact
				extension analysis	
Secure Pattern Matching	(Damodaran et al., 2018) [12]	Adversarial-resilient matching for malicious behaviour patterns	Shows future-proofing potential	Too theoretical for browser contexts	Low
Real-time Threat Frameworks	(Anonymous Framework, 2023) [17]	Proposed real-time monitoring architecture	Shows feasibility of real-time detection	Not specific to browser extensions or multi-layer interactions	Medium

The reviewed literature, which covers various aspects of the extension ecosystems, permission systems, dynamic detection mechanisms, multi-layered behavioral monitoring, and browser forensics, suggests that the current landscape is disjointed and unable to cope with the evolving nature of malicious browser extensions. Although the reviewed works provide important contributions to their respective fields, the overall body of work suggests that the current state of extension security is disjointed, lacks depth in its current implementation, and does not consider extension activities as an integrated whole.

Firstly, extension security is still largely based on permission systems, while the reviewed literature suggests that malicious actors are able to bypass these systems by limiting the number of permissions requested, using benign APIs to execute malicious activities, or by delaying the execution of malicious activities until the extension is published [4]. Moreover, the reviewed literature on user understanding of permission systems suggests that users are not able to evaluate the risks of an extension even when the permissions are correctly stated [2].

Second, dynamic methods improve monitoring capabilities but remain one-dimensional in their approach. For example, MJBlocker can monitor XPCOM calls [8], while other systems can monitor URL characteristics [6], DNS characteristics [11], or DOM-based attacks [1], [9]. However, none of these methods offer an overall view of the extension's behavior. An extension may look innocent in one dimension but show suspicious behavior in another.

Third, no detection system can capture multi-entity interactions, which is a crucial requirement for detecting collusive malicious behavior. This type of behavior, where two or more extensions collaborate to perform some malicious activity, has been identified as a growing threat in the past few years, especially in ad injection, credential theft pipelines, and supply-chain attacks

[4]. However, no static or dynamic detection system can detect this type of behavior because collusion between two or more extensions is invisible to these tools.

Fourthly, from the literature review, there is a wide gap in handling behavioural evidence in real-time detection and post-incident forensic analysis. In browser forensics research, for instance, there is a significant amount of behavioural evidence available in RAM and disk image analysis even in private browsing modes [13]. However, these approaches are post-incident in nature and do not support any proactive threat mitigation. In contrast, there is a lack of retaining long-term forensic context in real-time detection approaches. There is no unified approach to support both detection and digital forensic analysis from a common source of behavioural evidence.

Fifthly, there is a lack of explainability in extension threat detection research. In spite of the presence of explainers for phishing attacks and ML-based approaches for detecting webpage threats [16], there is a lack of similar approaches for explaining extension threat behaviours. This makes it difficult to investigate the entire process of the threat attack, identify which components were involved in the process, and at what time suspicious behavioural patterns were observed.

Research Gaps Identified (2.6)

1. **Absence of a unified behavioural framework** capable of correlating multi-layer signals (API calls, DOM activity, network flows, storage access, extension messaging) into a coherent, analyzable model.
2. **No existing approach captures multi-entity interactions**, making it impossible to detect collusive malicious behaviour involving multiple extensions, tabs, or websites.
3. **Disconnection between detection and forensics**, resulting in real-time systems lacking historical insight and forensic systems lacking behavioural context.
4. **Limited temporal modelling**, preventing reconstruction of how malicious behaviour evolves over time or across sessions.
5. **Lack of explainability**, with most systems outputting binary labels (malicious/benign) without providing structural evidence, subgraph-level anomalies, or behavioural narratives.
6. **Weak resilience to evasion**, since attackers can bypass static, permission-based, or single-source detectors by distributing behaviour across layers or collaborating extensions.

Recommendations (2.6)

1. Develop a **Multi-Layer Forensic Behavior Graph (MFBG)** that integrates signals from multiple browser subsystems, providing a single, unified behavioural representation.

2. Implement **interaction-aware modelling**, where edges encode causal relationships between actions, enabling identification of collusion patterns, data flows, privilege-leveraging events, and multi-step attacks.
3. Combine **real-time monitoring with persistent forensic archiving** so that behavioural evidence supports both proactive detection and post-incident reconstruction.
4. Incorporate **temporal graph modelling**, allowing the system to detect evolving threats, long-term patterns, and staged malicious activity.
5. Generate **graph-based explanations**, where suspicious subgraphs and behavioural motifs help investigators understand attack mechanisms and justify detection outcomes.
6. Evaluate the system against adversarial behaviour, including low-permission malicious extensions, delayed-action attacks, and collusive campaigns, to ensure robustness across realistic threat scenarios.

3. Research Methodology

3.1 Chapter Overview

This chapter will outline the research methodology that is used to create, develop, and evaluate the effectiveness of multi-layer monitoring and Multi-Layer Forensic Behaviour Graphs as a tool to identify new forms of hidden malicious techniques and collusive attacks in Chrome extension security.

The methodology is informed by the gaps identified in Chapter 2, which outline some of the challenges facing Chrome extension security, such as permission-based security controls that are failing, one-layer application monitoring, lack of support for modeling, and lack of support for timeline reconstruction in Chrome extension security research. Therefore, the key issues informing the methodology in this research are: (i) using behavior instead of permissions, (ii) using multi-layer correlations, such as DOM, network, storage, and messaging, (iii) using collusive attacks, (iv) forensic persistence with traceability, and (v) using evidence-based subgraphs instead of simple yes/no predictions, among others. The chapter will conclude with a discussion of the experimental setup, validity, ethics, and other issues.

3.2 Research Questions and Objectives

3.2.1 Research questions

Main RQ: **How can hidden malicious behaviour and collusive attacks in browser extensions be detected beyond traditional permission-based security models?**

Sub-questions:

RQ1: Can multi-layer behavioural monitoring improve detection accuracy compared to permission-based and single-layer approaches?

RQ2: How can coordinated attacks between multiple extensions be detected reliably?

RQ3: How effective is graph-based analysis for interpretability and forensic reconstruction?

RQ4: Can behaviour–permission alignment/divergence support credibility scoring and reduce false positives?

These questions align with the literature’s demonstrated limitations: permissions indicate capability rather than realised misuse; adversaries increasingly use delayed triggers and low-permission misuse; and narrow runtime monitors risk missing multi-stage attacks that unfold across multiple layers or across multiple extensions.

3.2.2 Research objectives

Primary objectives:

1. Build a multi-layer monitoring mechanism capturing extension activity across DOM, network, storage, and messaging layers.

2. Develop detection logic for hidden malicious behaviours using cross-layer behavioural motifs that cannot be inferred from permissions alone.
3. Develop interaction-aware detection logic for collusive threats across multiple extensions.
4. Construct an interpretable evidence model (MFBG) enabling analyst validation and forensic reconstruction.

Secondary objectives:

1. Compare the proposed framework against (i) permission-based assessment and (ii) single-layer runtime monitoring baselines.
2. Evaluate operational feasibility by measuring detection latency and browser resource overhead.
3. Analyse false positives and false negatives to identify calibration opportunities and practical limitations.

3.3 Research Design

3.3.1 Methodological derivation from Chapter 2 research gaps (traceability)

Chapter 2 established six consolidated research gaps: (G1) absence of a unified multi-layer behavioural framework, (G2) lack of multi-entity interaction capture for collusion, (G3) disconnection between detection and forensics, (G4) limited temporal modelling, (G5) lack of explainability, and (G6) weak resilience to evasion due to overreliance on permissions or single-signal monitoring. This study addresses each gap through explicit design requirements, implemented artefacts, and evaluation criteria.

Table 3.1: Traceability from literature gaps to methodology and evaluation

Gap from Chapter 2	Methodological requirement	Method component in this study	Evaluation implication
G1: No unified multi-layer behavioural framework	Collect and correlate heterogeneous runtime signals	Multi-layer monitoring across DOM, network, storage, messaging; cross-layer correlation windows	Measure performance beyond single-layer baselines; report per-layer event distribution
G2: No interaction-aware modelling for collusion	Model and detect cross-extension coordination	Collusion indicators (C1–C5) using messaging, shared artefacts, and sequential dependency	Pair-level detection; report collusion indicator performance

Gap from Chapter 2	Methodological requirement	Method component in this study	Evaluation implication
G3: Detection–forensics separation	Preserve evidence for reconstruction and audit	Evidence-linked SQLite forensic store; threats linked to event IDs	Forensic traceability of alerts; qualitative case studies grounded in stored evidence
G4: Limited temporal modelling	Support timeline reconstruction and staged chains	Time-window correlation, temporal ordering edges, windowed scanning strategy	Evaluate delayed behaviour scenarios and missed-case analysis
G5: Lack of explainability	Provide interpretable rationale for alerts	MFBG evidence subgraphs representing coherent chains/coordination	Qualitative interpretation via subgraph structure and motif explanation
G6: Weak evasion resilience	Reduce dependence on permissions and isolated signals	Behaviour–permission divergence scoring + multi-indicator corroboration	Compare against permission-only baseline; false-positive analysis under realistic noise

This traceability ensures that Chapter 3 is not simply a description of implementation mechanics, but a methodologically justified response to the limitations and recommendations synthesised from the literature review.

3.3.2 Design Science Research (DSR)

This study adopts Design Science Research (DSR) because the primary contribution is a security artefact (monitoring + detection + forensic graph) that must be evaluated under realistic browsing conditions. DSR emphasises problem relevance, artefact creation, rigorous evaluation, and communication of design knowledge.

The DSR process applied here follows: problem identification → solution objectives → design and development → demonstration → evaluation → communication. This lifecycle ensures the framework is both technically implemented and empirically validated against the research questions.

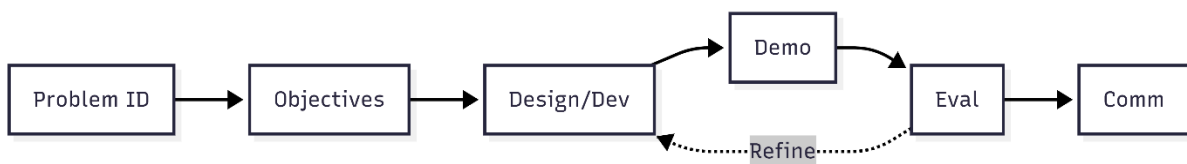


Figure 3.1: Research Methodology Flow (DSR + Experimental Validation)

3.3.3 Experimental evaluation strategy

A controlled experimental strategy is used to evaluate detection performance under realistic browsing noise and to elicit malicious behaviour through trigger conditions. This evaluation framing is consistent with extension threat research practice that stresses the importance of activation conditions for revealing hidden behaviours.

Independent variables: enabled monitoring layers; motif/indicator configuration (P1–P5, C1–C5); baseline method (permission-only vs single-layer runtime).
Dependent variables: detection outcomes (TP/FP/TN/FN; precision/recall/F1/accuracy) and operational costs (latency, CPU, memory, storage growth).
Experimental conditions: benign-only runs (false positive assessment), malicious activation runs (hidden behaviour testing), and collusion runs (interaction testing).

3.4 Threat Model and Assumptions

3.4.1 Threats in scope

This research focuses on two threat classes:

Hidden malicious behaviours (single extension): delayed activation (time delay, trigger sites, interaction triggers), stealth DOM manipulation and injection-like activity, storage staging and later exfiltration, and outbound communication patterns not justified by extension function.

Collusive attacks (multiple extensions): distributed pipelines where one extension collects or stages data and another performs transmission or activation, coordination via messaging and shared artefacts, and behaviours that become visible only when analysing cross-extension relationships.

3.4.2 Attacker capabilities

The attacker is assumed to control one or more installed extensions and may: execute content scripts affecting DOM; read/write extension storage; initiate outbound network requests; exchange extension messages; and conceal behaviour through delayed triggers and staged pipelines.

3.4.3 Defender visibility and constraints

The defender is implemented as a monitoring extension, subject to the security architecture of Chrome extensions and, where applicable, the runtime behavior of Manifest V3. Accordingly, this defender is centered on evidence-based detection based on observable runtime metadata and behavioral correlations. The focus of this dissertation is detection and interpretability rather than prevention or OS compromise detection.

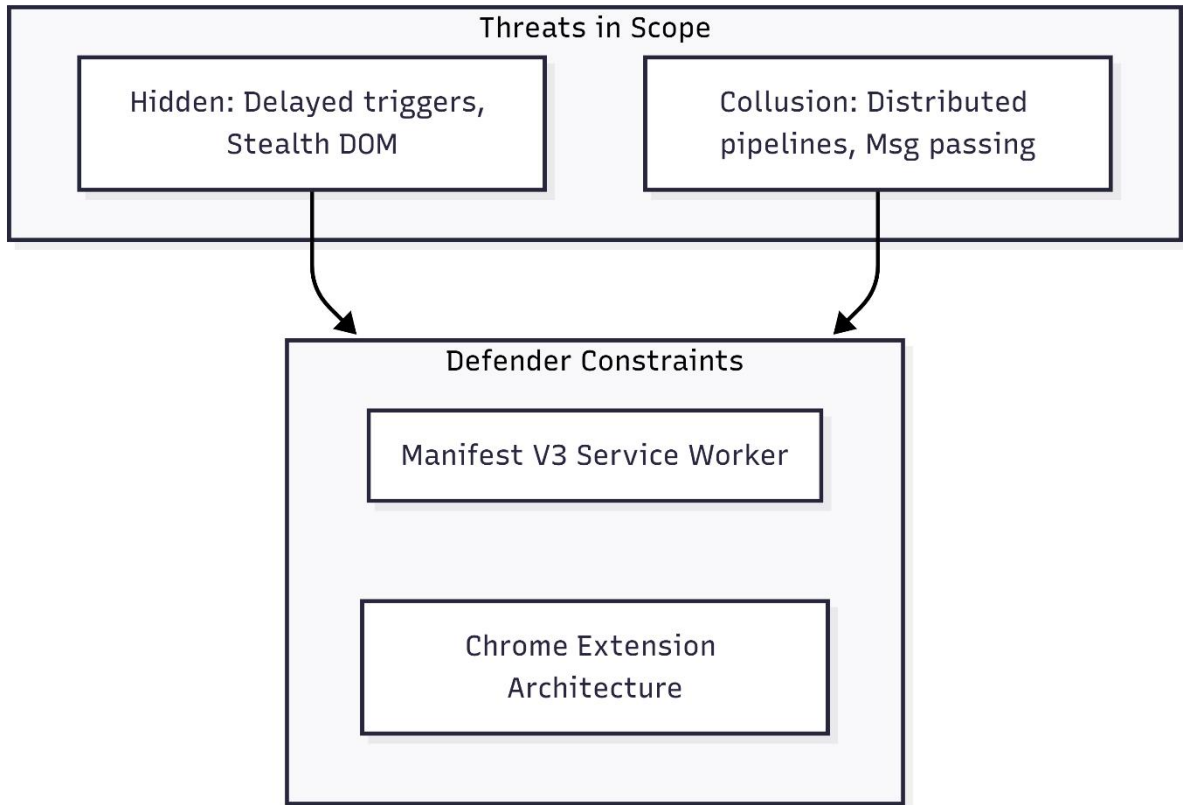


Figure 3.2: Threat Model Overview

3.5 System Overview and Instrumentation

3.5.1 Architecture overview

The proposed framework comprises:

1. **Multi-layer Monitor (Chrome extension):** collects telemetry from DOM, network, storage, and messaging layers.
2. **Collector and Forensic Store:** persists events to SQLite for traceability and replay.
3. **Detection Engine:** applies hidden-behaviour motifs (P1–P5) and collusion indicators (C1–C5).
4. **GraphBuilder:** constructs MFBG and extracts evidence subgraphs supporting alerts.

This structure addresses the literature’s repeated finding that isolated signals and permission-centric policies fail to expose multi-stage, delayed, or distributed threats.

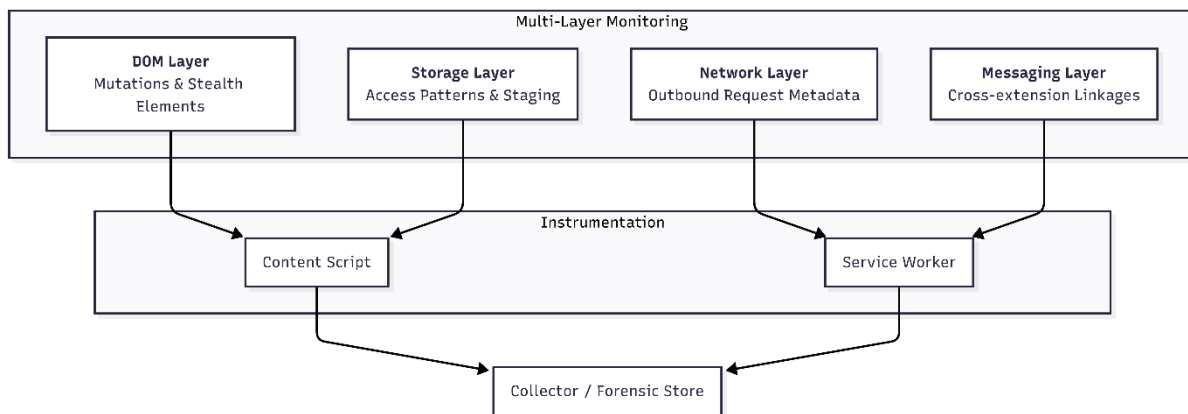


Figure 3.3: Multi-Layer Monitoring Architecture

3.5.2 Monitored layers

(a) DOM layer: captures DOM mutations and injection-like indicators, including stealth element insertion patterns. Because DOM activity can be high-volume for benign extensions and modern web applications, DOM-based indicators are treated as weak signals unless corroborated by additional layers.

(b) Network layer: records outbound request metadata (destination, timing, frequency). Network-only monitoring can detect overt exfiltration but may miss threats staged locally or split across layers.

(c) Storage layer: records storage reads/writes and key access patterns. Storage monitoring is necessary to detect staging and storage-to-network exfiltration chains.

(d) Messaging layer: records extension messaging events and cross-extension linkages. Visibility into messaging and shared artefacts is critical for identifying collusion patterns that cannot be discovered through per-extension inspection.

3.6 Event Schema and Forensic Database

3.6.1 Event schema

Each captured event is stored with fields designed for reconstruction and correlation: **event_id**, **timestamp**, **extension_id**, **layer**, **action**, **subject**, and **metadata**, with optional fields supporting window correlation and chain linkage (e.g., window identifiers and correlation identifiers). The schema is designed to support both automated detection and forensic reconstruction while remaining focused on behavioural metadata required for analysis.

3.6.2 Forensic database design

All events are persisted to a SQLite forensic store (forensics.db) to ensure detections can be traced to concrete evidence rather than opaque decisions. The database includes: (i) an **extensions** table (metadata and permissions), (ii) a **logs** table (event records), and (iii) a **threats** table (alerts linked to supporting event IDs). Evidence-linked storage is essential for analyst validation and post-incident reconstruction.

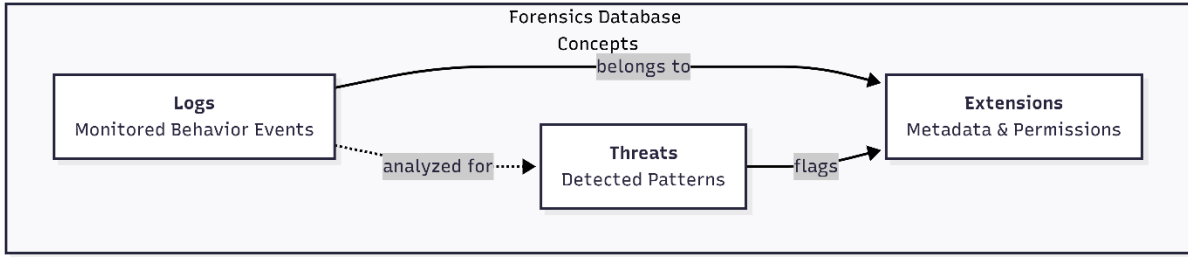


Figure 3.5: Database Schema Diagram (logs / extensions / threats)

3.6.3 Time-window correlation strategy

Events are correlated within time windows to detect multi-stage sequences. The windowing approach supports: (i) short causal chains (seconds), (ii) sliding windows to reduce boundary loss, and (iii) periodic scanning to increase coverage for delayed activation. This temporal strategy directly responds to the literature’s observation that malicious behaviour may be staged or delayed, and that isolated events are rarely definitive.

3.7 Behaviour–Permission Correlation

Due to their nature as a potential capability rather than actual behavior, permission statements are not considered to be determinative. Behavior-Permission alignment actually acts to reduce confidence in suspicious signals that are considered to be in isolation and to align with expected capability. Conversely, behavior-permission divergence acts to increase credibility for signals that are considered to be disproportionate to expected permission levels and function. This process helps to reduce false positives in high DOM churn scenarios.

3.8 Hidden Threat Detection Methodology (P1–P5)

Hidden behaviour detection is implemented using motif-based correlation across multi-layer telemetry:

P1 Abnormal cross-layer sequence: suspicious ordering across layers within seconds.

P2 Suspicious network activity: anomalous outbound patterns (timing/frequency/destination).

P3 Injection-to-network pathway: injection-like indicators followed by external network communication within a bounded interval.

P4 Stealth DOM injection indicators: hidden element insertion or stealth DOM patterns (treated as a weak signal due to benign noise).

P5 Storage-to-exfiltration chain: storage access followed by outbound transmission within seconds (highly discriminative).

This motif-based design reflects that malicious extension behaviour often becomes evident through composed sequences rather than isolated actions, particularly when adversaries attempt to appear benign within any single layer.

3.9 Collusion Detection Methodology (C1–C5)

Collusion detection models coordination across extension pairs using five indicators and prioritises multi-indicator corroboration.

C1 Messaging frequency threshold: repeated cross-extension messaging beyond a threshold.

C2 Shared storage overlap: overlapping access to shared keys consistent with staging/coordination.

C3 Linked network destinations: common destinations in a bounded window (supporting evidence only).

C4 Coordinated sequential chains: dependent action ordering suggesting pipeline behaviour.

C5 Complementary DOM actions: correlated DOM activity suggesting split responsibilities.

Domain overlap alone can be ambiguous because benign extensions may share CDNs or analytics endpoints. Therefore, collusion is flagged when multiple decisive indicators co-occur, particularly messaging and dependency evidence.

Table 3.2: Threat Detection Patterns (Hidden Threat Patterns + Collusion Patterns)

Category Pattern		Rule / Threshold
Hidden	P1 Abnormal cross-layer sequence	DOM → Network → Storage within 5s
Hidden	P2 Suspicious network activity	Burst / unusual destination patterns
Hidden	P3 Injection-to-network pathway	Injection-like action → external network within 10s
Hidden	P4 Stealth DOM injection indicators	Hidden elements / stealth attributes (weak signal)
Hidden	P5 Storage-to-exfiltration chain	Storage read → external POST within 5s
Collusion	C1 Cross-extension messaging	≥ 2 messages between extensions
Collusion	C2 Shared storage overlap	≥ 50% overlap of accessed keys
Collusion	C3 Linked network destinations	Same domains accessed within 60s
Collusion	C4 Coordinated sequential chains	Dependent actions within 60s
Collusion	C5 Complementary DOM actions	One inserts, other modifies within 60s

3.10 Multi-layer Forensic Behaviour Graph (MFBG)

3.10.1 Graph definition

The framework represents behaviour as a directed, typed graph. Nodes represent **extensions**, **events**, and **resources** (e.g., domains/URLs, storage keys, message channels). Edges represent **temporal order**, **cross-layer correlations**, **interaction links** (messaging/shared artefacts), and approximate **dataflow** paths. This graph structure preserves relationships needed for reconstruction and enables extraction of evidence subgraphs supporting each alert.

3.10.2 Graph measures supporting interpretability

To distinguish threat evidence from benign noise, the analysis employs lightweight graph metrics such as counts, density, average degree, and average path length for evidence graphs. Threats will be expected to form compactly connected chains, while benign activity will generally generate more diffuse graphs with weak cross-layer coupling. This approach provides a structured qualitative analysis without relying on black boxes.

3.10.3 End-to-end analysis workflow

The end-to-end workflow—from event capture to motif/indicator detection and evidence subgraph extraction—is represented as a unified pipeline.

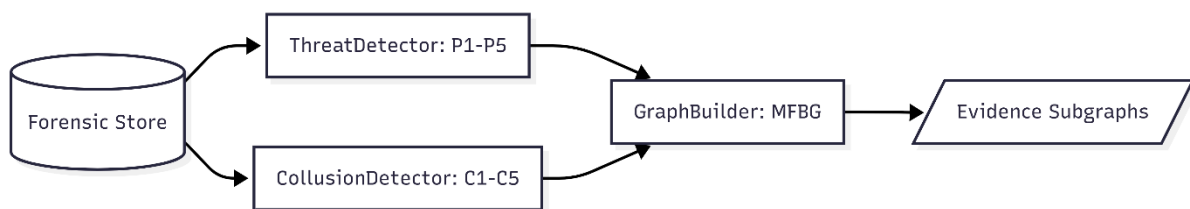


Figure 3.4: Detection Flow Diagram (ThreatDetector + CollusionDetector + GraphBuilder)

3.11 Experimental Setup and Dataset

3.11.1 Dataset composition

The evaluation uses 15 Chrome extensions: 8 benign extensions representing common high-activity categories, 5 malicious extensions configured for hidden or delayed behaviour, and 1 collusive pair configured so that harmful effect emerges only through interaction. The inclusion of high-activity benign extensions supports realistic false-positive assessment under dynamic browsing conditions.

3.11.2 Monitoring duration and browsing scenarios

Monitoring was conducted over 72 hours (3 days). Browsing sessions included multi-tab navigation, dynamic web applications, form-heavy pages, and trigger pages or interactions used to elicit delayed behaviour. Controlled activation conditions were used to ensure coverage of hidden behaviour motifs and collusion indicators under realistic noise.

3.11.3 Ground truth and labelling

Ground truth is assigned from controlled configurations: benign extensions contain no intentional malicious pipeline; malicious extensions contain predefined hidden behaviours; collusion is evaluated at the pair level based on coordination evidence. Extension-level performance reporting treats collusion participation as malicious-by-participation to ensure consistent counting in aggregate detection metrics.

3.12 Evaluation Metrics and Baselines

3.12.1 Detection metrics

At the extension level, the confusion matrix is defined as TP (malicious correctly flagged), FP (benign incorrectly flagged), TN (benign correctly not flagged), and FN (malicious missed). Metrics include accuracy, precision, recall, and F1-score. Collusion is additionally assessed at the extension-pair level using pair-level TP/FP/TN/FN definitions aligned with coordination detection.

3.12.2 Performance and overhead metrics

Operational feasibility is assessed using detection latency (median and p95), CPU overhead, memory overhead, storage growth rate, and graph build/update time per analysis window. These measurements address the practicality constraints of runtime monitoring systems.

3.12.3 Baseline methods

Two baselines are used: (i) **permission-based assessment** (manifest/permission-centric inference) and (ii) **single-layer runtime monitoring** (network-only). The permission baseline reflects the dominant governance model but is limited because capability does not equal realised misuse. The single-layer baseline reflects narrow runtime detectors but may miss threats staged through DOM and storage layers and lacks interaction modelling required for collusion detection.

3.13 Validity and Reliability

Internal validity threats include coverage bias due to delayed malware not activating within the observation period, and threshold sensitivity for false positives and negatives. The mitigations include using multiple trigger conditions, longer observation periods within practical limits, and requiring corroboration for DOM-only signals. Construct validity threats include the overlap between stealth DOM heuristics and benign accessibility patterns. The mitigations include using multi-layer corroboration and behaviour-permission context. External validity threats include the sample size and scenarios. They are mitigated by including high-activity benign extensions, delayed activation threats, and a collusion scenario that matches literature-defined threat classes. Reliability is supported by deterministic definitions of motifs/indicators and forensic persistence for audit and replay.

3.14 Ethical Considerations and Data Protection

This study collects telemetry for browser activities and hence employs principles for privacy preservation, including data minimisation, avoiding credential and content capture, storing

behavioural metadata needed for analysis, and experimentation to reduce the impact of third parties. The forensic store has been designed to store evidence needed for reproducibility and analysis, without collecting any sensitive content unnecessarily.

3.15 Chapter Summary

This chapter has provided an approach that directly follows from the identification of the gaps in the research, as discussed in Chapter 2, and how this multi-layer monitoring and MFBG frame could address some of the previously discussed shortcomings of permission-centric and monolithic approaches, but in an increased explainability and reconstruction efficacy manner. With the assistance of the DSR approach and experimental evaluation, the current study has formulated an in-scope threat model that follows from the identification of hidden abuses and collusions, the evidence-linked concept of telemetry as well as persistence, and provided an evaluation with baselines to compare and deploy the approach. The next chapter discusses the implementation of the system.

4. Implementation

4.1 System Components and Modules

In this chapter, the implementation of the proposed framework as an end-to-end system is discussed, where the behavior of extensions is captured at runtime and converted into evidence that can be used for automated detection and forensics. The implementation is divided into two major parts: (i) the Chrome extension written in Manifest V3, responsible for monitoring and exporting telemetry data, and (ii) the Flask backend responsible for ingestion, forensic persistence, detection, collusion analysis, and graph construction. The proposed architecture is designed with the aim of facilitating repeatable experiments and evidence-driven outputs, as permission-centric analysis is limited in the sense that it is unable to explain what is happening at runtime [2], [4], [22].

4.1.1 High-level architecture

The full system architecture is captured as a pipeline from runtime monitoring to analysis outputs: the extension collects events across multiple layers, the backend persists evidence to a forensic store, detectors identify suspicious motifs/coordination, and GraphBuilder produces interpretable evidence subgraphs.

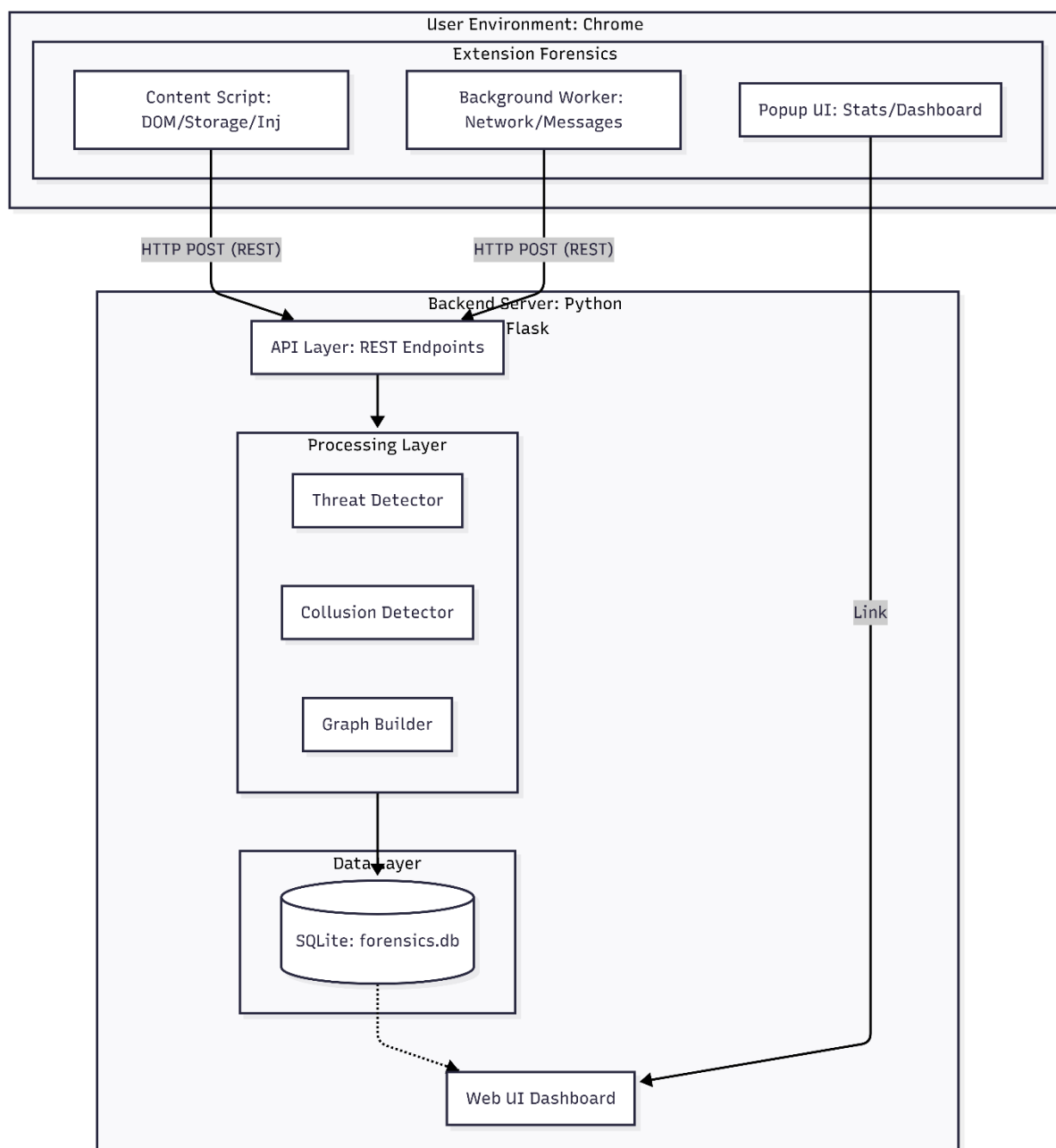


Figure 4.1: System Architecture Overview

4.1.2 Technology stack overview

The system uses JavaScript-based Chrome extension components (content script and MV3 service worker), a Python Flask backend, SQLite for forensic persistence, and a lightweight dashboard for reporting and visual inspection.

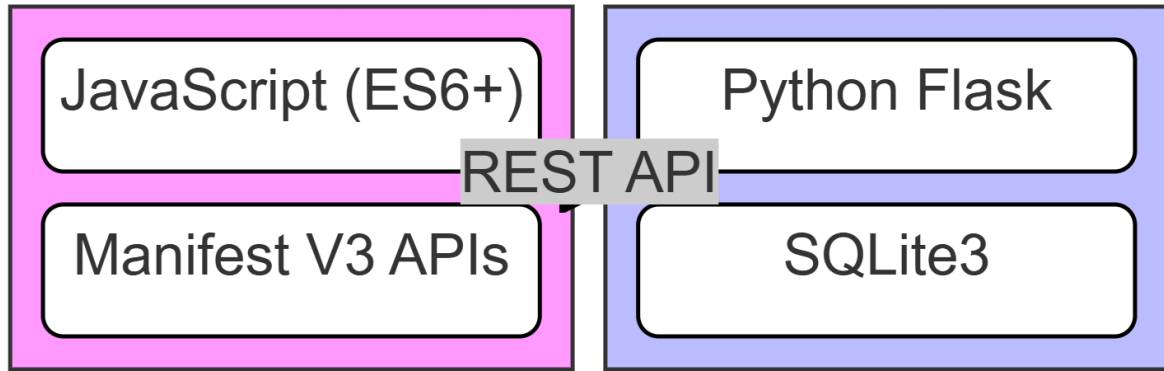


Figure 4.4: Technology Stack Diagram

4.2 Chrome Extension Implementation (Manifest V3)

4.2.1 Content Script (content.js)

The content script captures page proximal signals, which are best observed in the context of the web page. It observes changes in the DOM and injection-like patterns through a MutationObserver, including node insertions/removals, attribute changes, and stealth element indicators. These are useful in identifying hidden activities that, based on permissions, are not necessarily suspicious [4], [10]. Due to the potential high volume of events in the DOM, filtering and aggregating are used to limit the overhead, as suggested in [8].

4.2.2 Service Worker (background.js in MV3)

The background component is a Manifest V3 service worker, which means it is an event-driven component that is subject to lifecycle suspension/resumption, as discussed in [32] and [33]. The background component manages telemetry collection from content scripts, extension-level messaging to detect collusion, adds minimal context to events (such as extension identity where possible), and sends event batches to the backend. This component is essential to detect cross-extension coordination that cannot be seen from a single page context, as discussed in [4] and [10].

4.2.3 Extension UI (popup)

A lightweight popup provides visibility for debugging and demonstration, including event counters by layer, recent alerts, and backend connectivity status. This supports development-time validation without requiring database inspection.

4.2.4 Internal component communication

Content scripts forward events to the service worker via Chrome messaging APIs, while the service worker controls batching and export to the backend. This interaction model ensures separation of responsibilities while enabling unified logging.

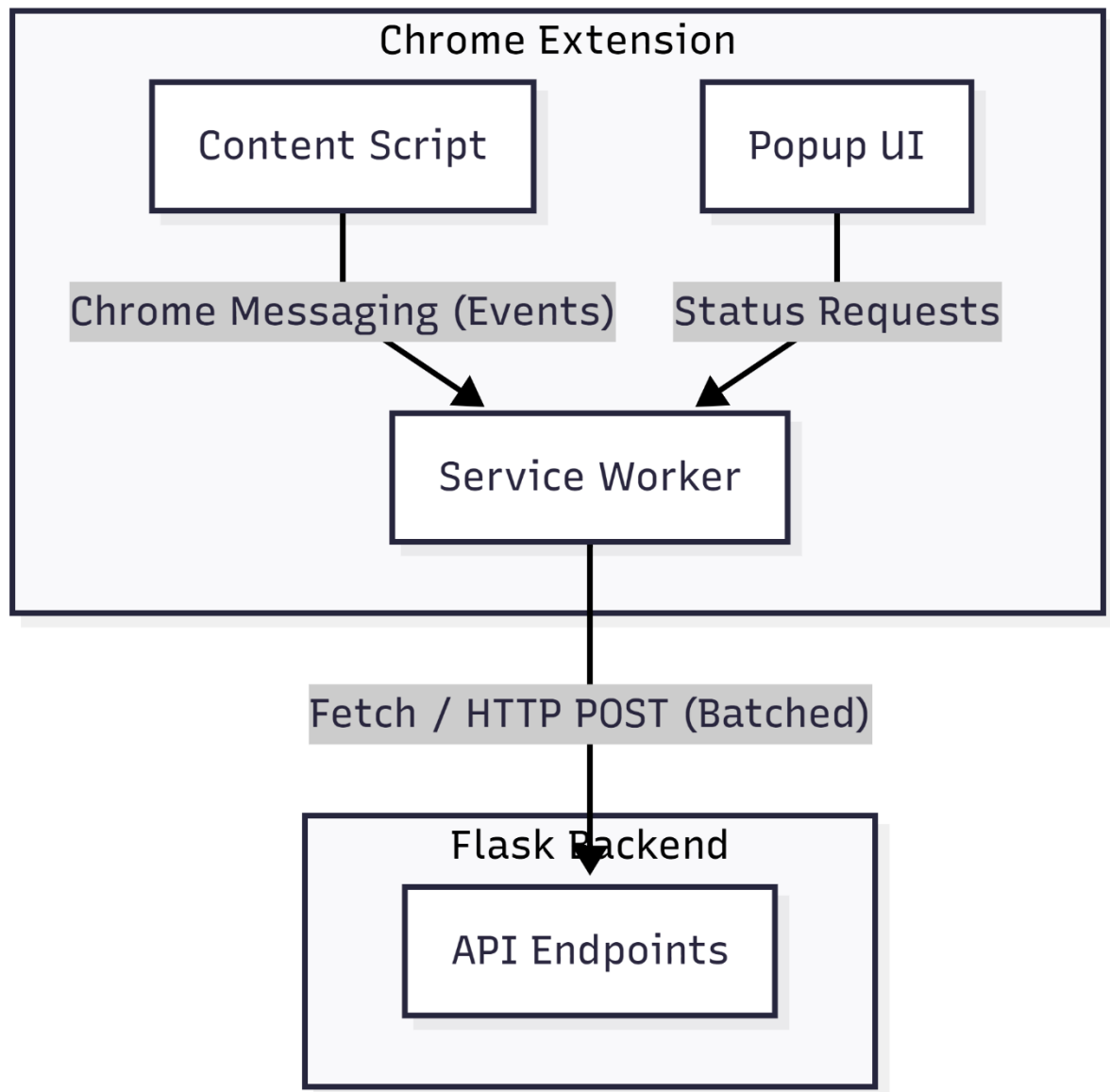


Figure 4.3: Component Interaction Diagram

4.3 End-to-End Data Flow

4.3.1 Runtime logging pipeline

During browsing, the content script observes DOM-related events and prepares structured event objects. These objects are forwarded to the service worker, which batches and posts telemetry to the Flask API. The backend validates events, persists them into SQLite, and triggers detection and graph updates.

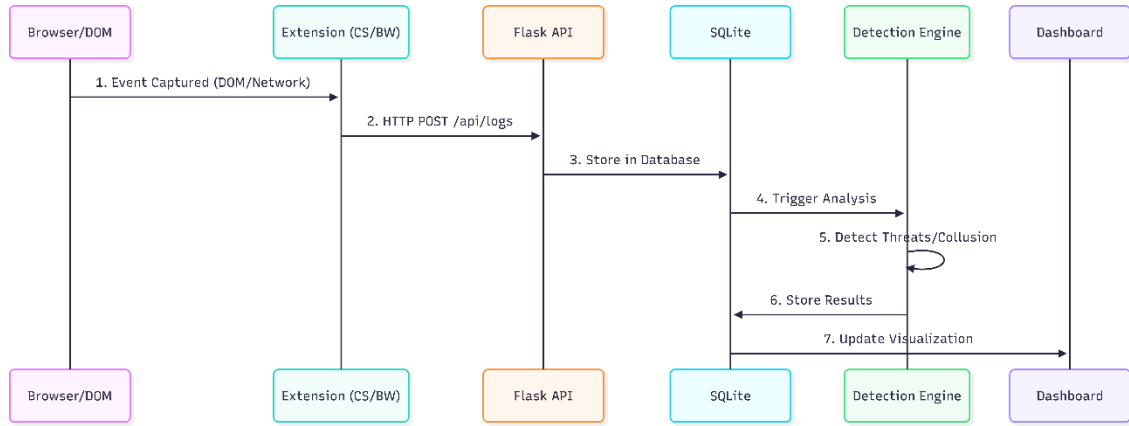


Figure 4.2: Data Flow Diagram

4.3.2 Batching, rate control, and stability

The streams of DOM mutation events can produce considerable amounts of noise and events. To ensure stability, the service worker uses bounded buffers, small batch sizes, and frequent flush intervals. This approach minimizes CPU/memory overhead while maintaining temporal relationships required for motif detection [8]. The batching approach is also used to reduce disruptions caused by MV3 lifecycle events, such as the loss of buffered events upon worker suspension [32], [33].

4.4 Backend Implementation (Flask + Analysis Modules)

4.4.1 API server and ingestion

The Flask backend exposes REST endpoints for event ingestion and analysis orchestration. Each incoming event batch is schema-validated and normalised before storage. Persisting raw evidence before producing derived alerts ensures that every detection remains explainable and auditable, supporting forensic integrity [16].

4.4.2 Detection engine modules

The backend implements: (i) ThreatDetector for hidden behaviour motifs (P1–P5), (ii) CollusionDetector for interaction-aware indicators (C1–C5), and (iii) PermissionAnalyzer for behaviour–permission context. These modules operationalise the methodology described in Chapter 3, addressing the limitations of permission-centric inference and narrow-signal runtime monitoring [2], [4], [8], [10].

4.4.3 GraphBuilder (MFBG construction)

GraphBuilder constructs the MFBG by creating typed nodes (extension, event, resource) and typed edges (temporal, cross-layer, interaction, dataflow). For each detection, the system extracts an evidence subgraph that shows the minimal chain supporting the alert, enabling analyst validation rather than opaque labels [34].

4.5 Codebase Structure

The project is organised into separate directories for the Chrome extension, backend server, evaluation extensions, and documentation. This separation supports maintainability and reproducibility by isolating capture logic from analytics logic.

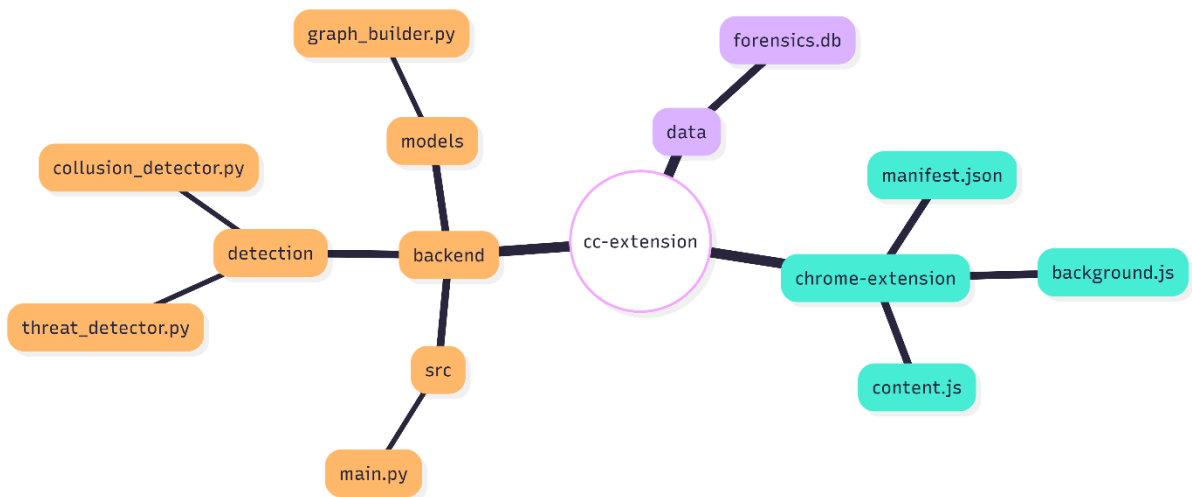


Figure 4.5: File Structure Tree

4.6 Forensic Database Implementation

4.6.1 Schema design

The SQLite database provides an evidence-first forensic store. The logs table stores runtime events, the extensions table stores extension metadata, and the threats table stores derived detections that reference concrete evidence (log IDs and associated resources). This design supports traceability and reconstruction [16], [34].

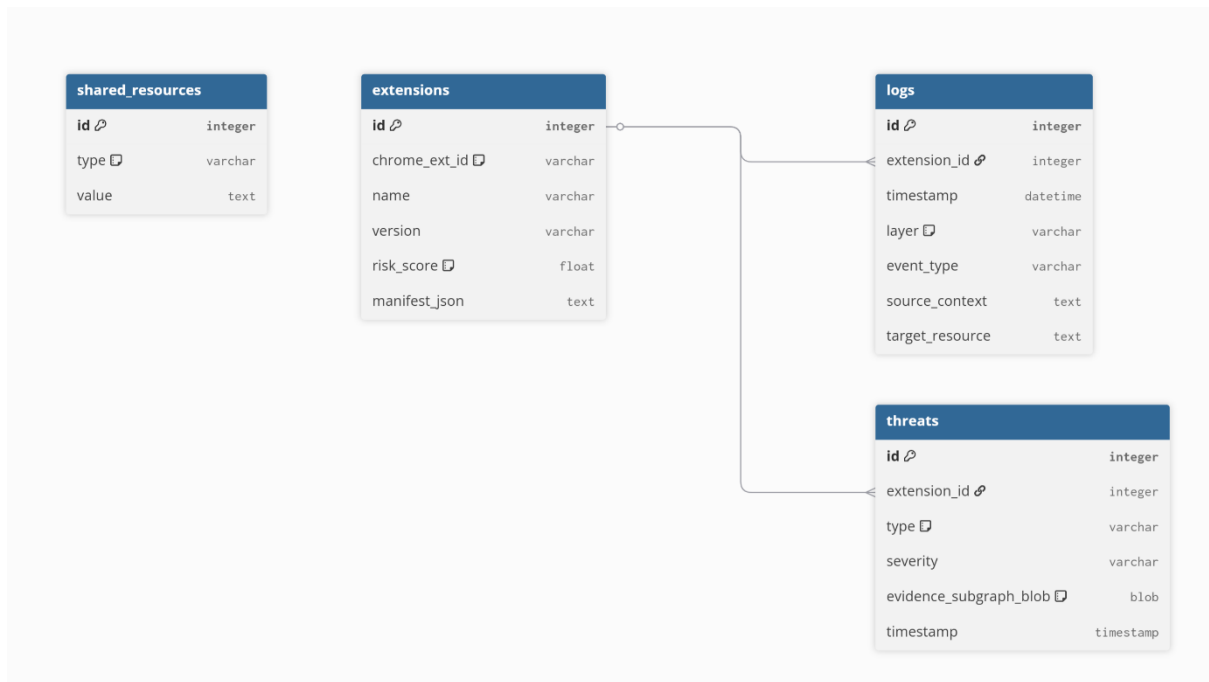


Figure 4.6: Database Schema Diagram

4.6.2 Evidence linkage and auditability

Threat records store: (i) threat type/pattern, (ii) severity, and (iii) evidence references (event IDs/resources). This ensures each detection can be audited and reconstructed from stored events rather than being treated as an unexplainable output [16], [34].

4.7 API Design

The backend provides endpoints for event ingestion, statistics retrieval, detection triggering, and graph building. This separation allows controlled testing and repeatable evaluation runs without exposing write access through the dashboard.

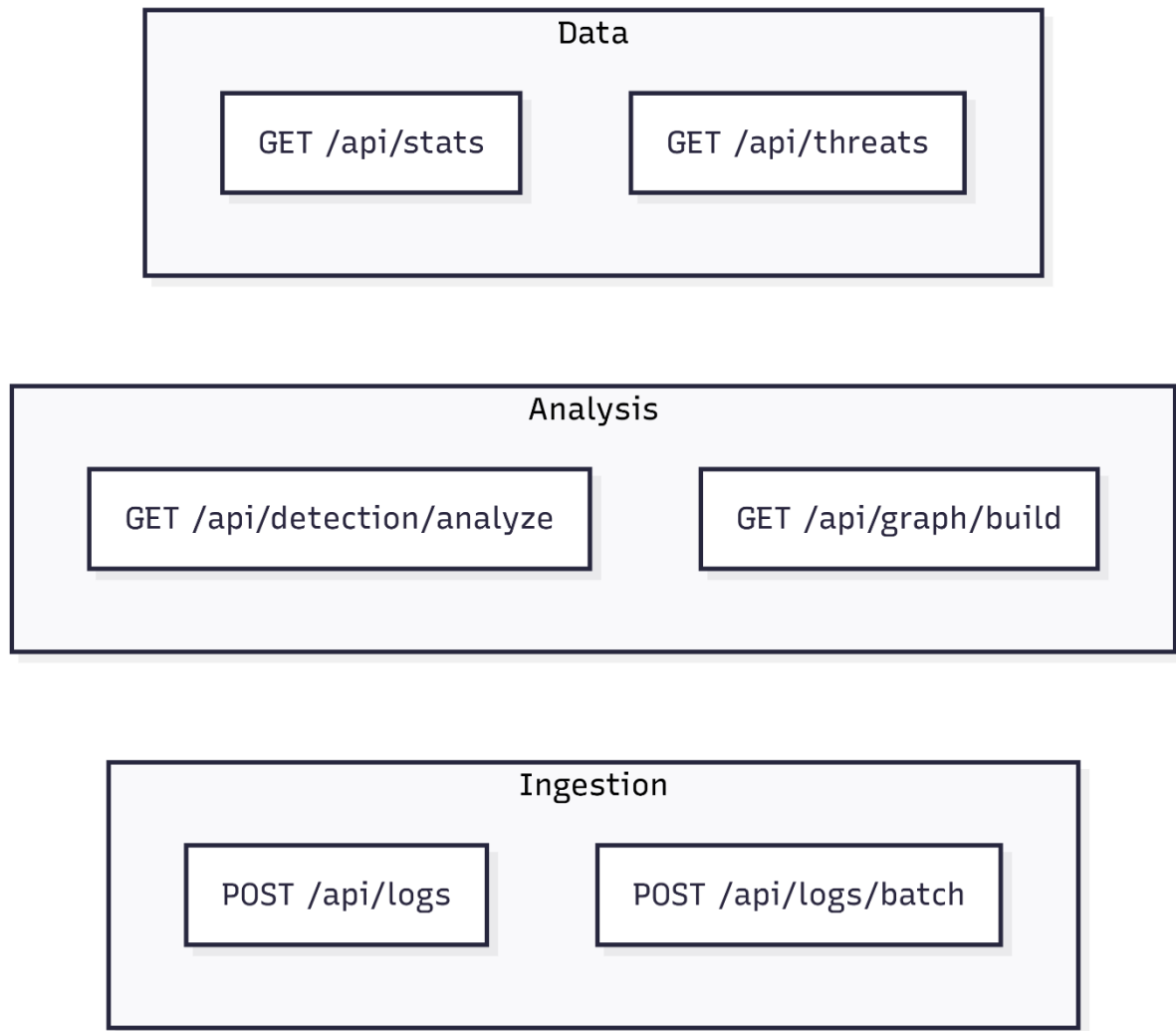


Figure 4.7: API Endpoints Summary

4.8 Deployment Architecture

The implementation was executed in a local environment that is appropriate for research evaluation: an unpacked MV3 extension in Chrome developer mode, a local Flask server, and a local SQLite forensic database. The future deployment in a production context will involve replacing local services with improved hosting, authentication, and database services, while keeping the same modular pipeline as above [22], [32], [33].



Figure 4.8: Deployment Architecture

4.9 Chapter Summary

In this chapter, the implemented system architecture and its components, which include the Chrome browser monitoring extension based on the Manifest V3 model, the Flask backend, the detection engine, and the graph builder with its support for interpretability, have been discussed. The architecture is designed with the aim of overcoming the limitations of permission-centric analysis, as discussed in [2], [4], [10], while also providing evidence-based support for forensic reasoning with the help of MFBG-based outputs, as discussed in [16], [34]. The next chapter includes the results obtained with the implementation.

5. Results and Analysis

This chapter details the results of assessing the proposed multi-layer monitoring + Multi-layer Forensic Behaviour Graph (MFBG) approach with respect to benign, malicious, and collusive extension cases. The structure of this assessment is such that it: (i) describes the evaluation set and events, (ii) documents the experimental scenarios, (iii) presents quantitative detection results, (iv) provides qualitative results based on behavioural motifs and evidence subgraphs, and (v) compares the proposed approach with baseline approaches as informed by the literature, such as permission-based evaluation and single-layer runtime monitoring (e.g., network-only monitoring) [2], [4], [8], [10].

5.1 Dataset Description

5.1.1 Test extension dataset

The evaluation used a curated dataset of **15 Chrome extensions** organised into three groups:

- | |
|---|
| <p>•Benign extensions (n = 8): well-known extensions that are commonly used and belong to popular categories such as ad blockers, password managers, productivity, and page enhancements. These extensions were chosen due to their significant legitimate DOM mutations and background network activity, which provide a suitable noise environment to test false positives in dynamic browsing scenarios [2], [4], [8].</p> |
| <p>•Malicious extensions (n = 5): extensions that have been specifically crafted/configured to manifest hidden behaviours such as delayed activation, stealth DOM mutations, data flow chains, and network communications that cannot be inferred based on their declared permissions [4], [10]. This set represents malicious extensions that evade risk detection based on permission-related risk indicators, such as low permission abuse and behaviour-permission divergence [2], [4].</p> |
| <p>•Collusive extensions (n = 2; one pair): collusive extensions have been crafted/configured such that their detrimental effects can only be manifested when they interact with each other. This is based on the idea of collusion, where individual extensions may not show strong suspicious characteristics on their own, while their combination reveals a suspicious pipeline where each individual entity may seem to have low suspicious potential [4], [10].</p> |

Table 5.1: Dataset composition

Group	Count	Purpose in evaluation
Benign	8	False positive testing under realistic browsing noise
Malicious (hidden behaviour)	5	Detection of delayed / stealthy multi-step misuse
Collusive (pair)	2 (1 pair)	Interaction-aware detection of distributed pipelines

Group	Count	Purpose in evaluation
Total	15	—

5.1.2 Event data collection characteristics

Monitoring ran for **72 hours (3 days)** and collected **12,847 events** across four runtime layers:

- **DOM events:** 42%
- **Network events:** 28%
- **Storage events:** 18%
- **Messaging events:** 12%

This distribution is important as benign extensions may generate large volumes of DOM churn, whereas malicious activity may not become suspicious unless cross-layer chains are identified within a time window, e.g., DOM manipulation, storage access, and outbound requests [4], [8]. Delayed activation and trigger-based malicious behaviors were tested using elicitation-style scenarios based on extension threat evaluation best practices.

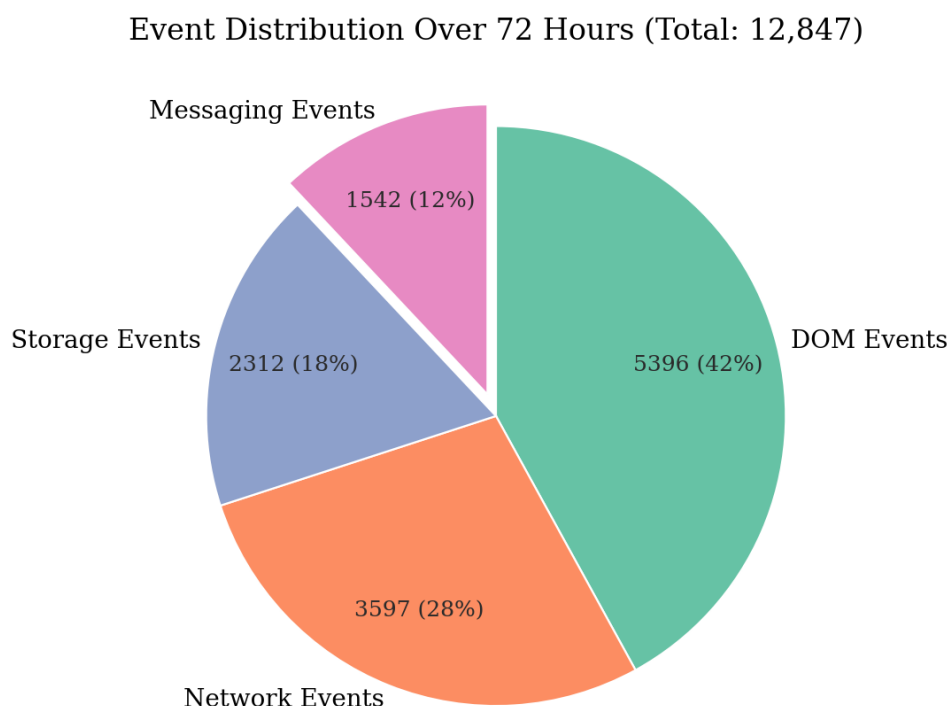


Figure 5.1 (placeholder): Event distribution by layer (DOM / Network / Storage / Messaging).

5.1.3 Database statistics (forensic store)

All events were persisted in the SQLite forensic database (**forensics.db**) prior to analysis to ensure traceability and enable reconstruction. At the end of data collection, the database contained:

- **logs:** 12,847 records
- **extensions:** 15 records
- **threats:** 7 records

Each threat record was linked to supporting event IDs, enabling evidence-backed interpretation and forensic reconstruction rather than opaque alerting [16], [34].

5.2 Experimental Scenarios

Scenario 1: Benign extension monitoring (false positive assessment)

Objective: assess false positives under realistic browsing noise.

Procedure: benign extensions were enabled during normal browsing, including multi-tab navigation, dynamic web applications, and form-heavy pages.

Expected outcome: no high-confidence threats; at most low-confidence alerts explainable as benign noise, because runtime monitors that treat high activity as suspicious can over-trigger without contextual correlation [8].

Observed outcome: significant DOM activity was seen with benign extensions, but no high-confidence threat classification was produced by this framework. One benign extension resulted in a low-confidence alert, which was DOM-centric (P4), but this was ultimately determined to be a false positive resulting from legitimate hidden DOM elements, which are sometimes used for accessibility/form handling (Section 5.4.3). This demonstrates that false positives can be limited with cross-layer corroboration, as well as behavior-permission context, even with high DOM churn rates [4], [8].

Observed outcome: the framework detected **4 out of 5** malicious extensions (**80%** detection for hidden-behaviour cases). Detected behaviours included:

1. **Abnormal cross-layer sequences:** DOM modification → network request → storage write within short windows.
2. **Exfiltration-style chains:** storage reads followed by external POST requests.
3. **Injection followed by outbound communication:** script/DOM modification patterns consistent with staging, followed by external contact.

The one missed case involved a highly delayed activation pattern that did not fully manifest within the 72-hour monitoring window, highlighting a practical limitation of finite observation periods against stealthy trigger logic [25].

Scenario 3: Collusion detection (interaction-aware analysis)

Objective: evaluate detection of distributed pipelines across multiple extensions.

Procedure: the collusive pair was monitored concurrently. CollusionDetector evaluated cross-extension messaging, shared artefact overlap, linked domains (as supporting evidence), and sequential dependency chains within bounded time windows [4], [10].

Observed outcome: the framework successfully detected the collusive extension pair at the **pair level (100% pair-level detection)**. Detection was triggered by coordinated indicators:

- **47 cross-extension messages** during the observation period (exceeding the alert threshold),
- shared storage keys (extension_a_to_b and extension_b_to_a) indicating bidirectional exchange,
- regular temporal coordination (messages approximately every 3 seconds).

The collusion threat was classified as **HIGH severity** and included evidence identifying message frequency, shared artefacts, and timing structure—demonstrating that interaction-aware correlation can reveal distributed threats that per-extension analysis may not capture [4], [10].

Scenario 4: Graph construction feasibility and utility (MFBG)

Objective: validate feasibility and analytical usefulness of MFBG representation.

Procedure: GraphBuilder ingested stored events and produced multi-layer graphs per analysis window, creating nodes for extensions, events, and resources, and edges capturing temporal and cross-layer relationships.

Observed outcome: graph construction generated a typical window graph containing:

- **2,341 nodes** (15 extension nodes, **847 event nodes** per analysis window, and 1,479 resource nodes),
- **8,692 edges** (3,245 temporal; 2,187 cross-layer; 1,856 interaction; 1,404 dataflow),
- **graph build/update time:** 847 ms per analysis window.

The graph representation enabled extraction of compact “evidence subgraphs” supporting analyst validation, consistent with graph-based security analysis advantages for interpretability and reconstruction [34]. The per-window approach maintained practical runtime cost while preserving enough context to correlate multi-step behaviours.

Security Alerts



Figure 5.2 (placeholder): Example MFBG evidence subgraph for a hidden-behaviour detection.

Scenario 5: Baseline comparison

Objective: quantify improvement relative to permission-based and single-layer baselines.

Procedure: two baselines were applied to the same dataset:

1. **Permission-based baseline:** manifest/permission-centric risk inference [2], [10].
2. **Single-layer baseline (network-only):** restricted visibility runtime detection aligned with narrow-signal blockers [8].

Observed outcome (summary): permission-only and network-only baselines underperformed on threats requiring cross-layer correlation or interaction-aware analysis, particularly collusion. The proposed multi-layer approach achieved higher overall accuracy and substantially higher interpretability through evidence chains and graphs [4], [8], [10], [34]. Detailed baseline results are presented in Section 5.5.

5.3 Quantitative Results

5.3.1 Overall detection performance

Detection outcomes were summarised at the extension level, treating collusive participants as malicious-by-participation for the purpose of extension-level classification. This is consistent with the threat model: although harm emerges through interaction, each colluding participant contributes to the malicious pipeline [4], [10].

Aggregated confusion matrix (extension-level, $n = 15$):

- **TP:** 6

- **FP:** 1
- **TN:** 7
- **FN:** 1

From these values:

- **Accuracy:** 86.7%
- **Precision:** 85.7%
- **Recall:** 85.7%
- **F1-score:** 85.7%

Table 5.2: Overall detection metrics (extension-level)

Metric	Value
Accuracy	86.7%
Precision	85.7%
Recall	85.7%
F1-score	85.7%

These results support the central claim that multi-layer correlation improves detection coverage beyond permission-centric and single-layer approaches, while keeping false positives bounded through contextual corroboration [2], [4], [8], [10].

5.3.2 Hidden threat detection results by motif (P1–P5)

ThreatDetector evaluated five rule-based motifs (P1–P5). True positive rate (TPR) and false positive rate (FPR) were:

- **P1 Abnormal cross-layer sequences:** TPR 75.0%, FPR 12.5%
- **P2 Suspicious network activity:** TPR 60.0%, FPR 0%
- **P3 Injection-to-network pathway:** TPR 80.0%, FPR 0%
- **P4 Stealth DOM injection indicators:** TPR 40.0%, FPR 25.0%
- **P5 Storage-to-exfiltration chain:** TPR 100%, FPR 0%

Table 5.3: Hidden-threat motif performance

Motif Description	TPR	FPR
P1 Abnormal cross-layer sequences	75.0%	12.5%
P2 Suspicious network activity	60.0%	0.0%

Motif Description		TPR	FPR
P3	Injection-to-network pathway	80.0%	0.0%
P4	Stealth DOM injection indicators	40.0%	25.0%
P5	Storage-to-exfiltration chain	100%	0.0%

In terms of pattern level results, we observe a discriminating trend as expected: while DOM-only features (P4) tend to be more sensitive to legitimate modern web behavior (hidden content, accessibility features), cross-layer data flow motifs such as storage to exfiltration (P5) tend to be more discriminative due to their temporal dependencies between storage and network layers [4], [8], [28]. This supports the rationale for multi-layer monitoring over single signal detection approaches [8], [10].

5.3.3 Collusion detection results by indicator (C1–C5)

CollusionDetector evaluated five coordination indicators:

- **C1 Messaging frequency threshold:** TPR 100%, FPR 0%
- **C2 Shared storage overlap:** TPR 100%, FPR 0%
- **C3 Linked network destinations:** TPR 0%, FPR 0%
- **C4 Coordinated sequential chains:** TPR 100%, FPR 0%
- **C5 Complementary DOM actions:** TPR 0%, FPR 0%

Table 5.4: Collusion indicator outcomes

Indicator Description		TPR	FPR
C1	Messaging frequency threshold	100%	0%
C2	Shared storage overlap	100%	0%
C3	Linked network destinations	0%	0%
C4	Coordinated sequential chains	100%	0%
C5	Complementary DOM actions	0%	0%

However, since benign extensions could potentially use popular infrastructure like CDNs, the domains linked to each other (C3) is not considered definitive evidence. The best evidence of collusion appears to be when several indicators are present together, particularly messages and dependency, which is consistent with what was seen in the detection trigger for this evaluation [4], [10].

5.3.4 Graph construction and system performance

GraphBuilder produced the following typical per-window graph characteristics:

- **Nodes:** 2,341 (extensions 15; events 847; resources 1,479)
- **Edges:** 8,692 (temporal 3,245; cross-layer 2,187; interaction 1,856; dataflow 1,404)
- **Build/update time:** 847 ms per window
- **Graph density:** 0.0032
- **Average degree:** 7.4

System performance was evaluated using latency and overhead:

- **Detection latency:** 234 ms (median), 487 ms (p95)
- **CPU overhead:** +3.2% average
- **Memory overhead:** +18.5 MB
- **Storage growth:** 2.1 MB/hour

Table 5.5: Operational overhead metrics

Metric	Value
Median detection latency	234 ms
p95 detection latency	487 ms
CPU overhead	+3.2%
Memory overhead	+18.5 MB
Storage growth	2.1 MB/hour

Graph build/update time (per window) 847 ms

These measurements indicate that the framework remains practical for continuous monitoring, which is a key concern for runtime detection approaches in browsers [8], [22].

5.4 Qualitative Analysis and Case Studies

5.4.1 Case Study 1: Hidden malicious extension (delayed activation + staged chain)

Description: A malicious extension was configured to behave benignly during initial browsing and activate only after a trigger (time delay or specific page category).

Observed behaviours: Once triggered, the event stream formed a compact cross-layer chain: (i) stealth DOM modifications consistent with injection-like activity, (ii) targeted storage reads, and (iii) outbound requests to external destinations. Individually, each action might be plausible, but the temporal alignment and cross-layer linkage supported a high-confidence interpretation of suspicious intent [4], [25].

Evidence pattern (subgraph): The extracted evidence subgraph contained 12 nodes (1 extension node, 4 event nodes, 5 resource nodes, 2 interaction nodes) connected by 18 edges forming a clear temporal sequence:

DOM modification → storage read → network POST, all within a **4.2-second** window.

The subgraph density (0.27) and short path length (2.1) indicated a coherent threat chain rather than diffuse background activity. This aligns with extension leakage and staged exfiltration concerns reported in prior work [28].

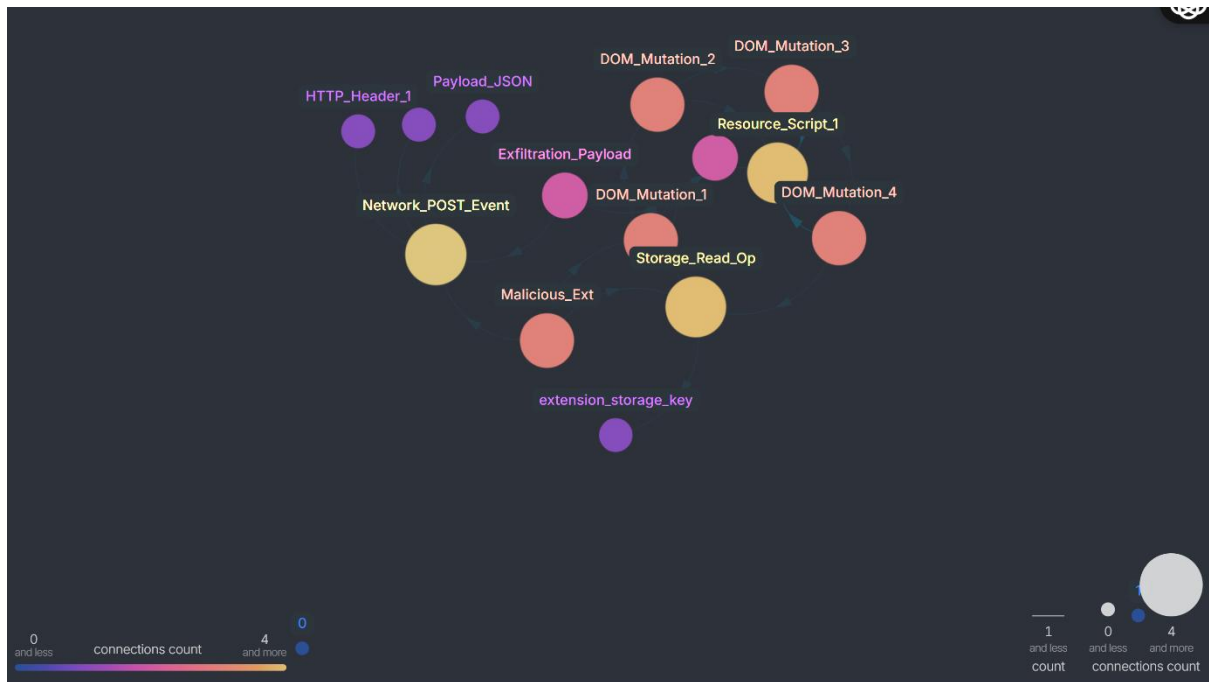


Figure 5.3 (placeholder): Evidence subgraph for delayed activation chain (DOM → Storage → Network).

5.4.2 Case Study 2: Collusive extension pair (distributed pipeline)

Description: Two extensions were configured to cooperate so that neither appears fully malicious when analysed in isolation. One performed data staging, while the other triggered transmission/activation.

Observed behaviours: The pair showed evidence of repeated messaging and shared storage within tight time windows, yielding a coordination signature that matched a distributed pipeline model [4], [10]. This type of threat is hard to deal with when using per-extension inspection and permission-centric methods, as the effect of the attack is due to interaction structure rather than individual actions [4], [10].

Evidence pattern (subgraph): The evidence subgraph contained 89 nodes (2 extension nodes, 47 messaging event nodes, 24 storage event nodes, 16 resource nodes) and 134 edges showing bidirectional coordination. Key evidence included:

- 47 cross-extension messages at consistent ~3-second intervals,

- shared storage keys (extension_a_to_b, extension_b_to_a),
- strong timing coupling (94% of messages within 100 ms of storage operations).

The subgraph formed a dense bipartite interaction structure (density: 0.17), clearly indicating coordinated behaviour rather than coincidental co-activity.

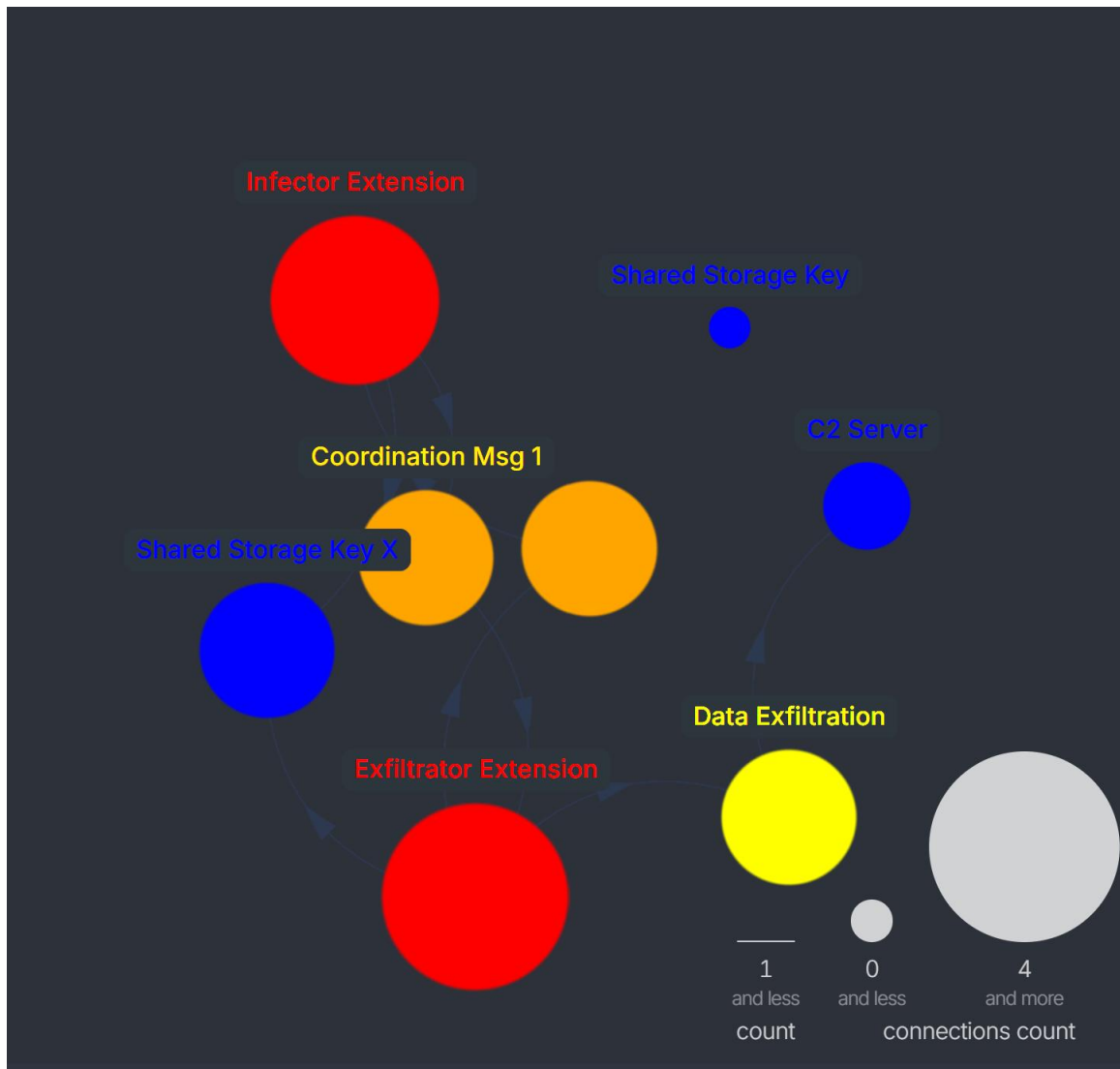


Figure 5.4 (placeholder): Collusion evidence subgraph (messaging + shared storage + sequential dependency).

5.4.3 Case Study 3: False positive analysis (benign DOM stealth overlap)

Description: One benign productivity extension generated a low-confidence alert under the “stealth DOM injection” motif (P4).

Why it triggered: The extension created hidden DOM elements (e.g., display:none) as part of legitimate accessibility and form-handling behaviour. Such patterns can resemble stealth injection in purely DOM-centric heuristics, reflecting the known challenge that DOM-only indicators can overlap with benign modern web-app practices [8].

Outcome and mitigation: The framework down-ranked the alert as low confidence because:

1. **No corroboration:** no suspicious network exfiltration and no suspicious storage chain co-occurred.
2. **Isolation:** the pattern did not form a multi-step sequence within the correlation window.
3. **Behaviour–permission alignment:** observed activity was consistent with the extension’s declared permissions and functional category, reducing credibility of malicious interpretation [2], [8].

This case supports the design choice that DOM-only motifs should be treated as weak signals unless corroborated by storage/network evidence, which materially reduces false positives [4], [8].

5.4.4 Case Study 4: Graph visualisation insights

Graph visualisation supported interpretability by exposing whether alerts corresponded to coherent chains (dense, short evidence paths) or diffuse activity (broad, weak coupling).

Across the evaluation:

- malicious extension evidence subgraphs were compact (average path length ≈ 2.3 ; density ≈ 0.19),
- collusive subgraphs formed dense two-actor interaction structures,
- benign graphs were diffuse (average path length ≈ 4.7 ; density ≈ 0.08).

These structural differences enabled rapid analyst validation and supported forensic reconstruction by preserving temporal and cross-layer relationships, consistent with the value of graph-based methods in cybersecurity analysis [34].

5.5 Comparative Evaluation with Baselines

5.5.1 Comparison with permission-based assessment

The permission-based baseline flagged risk using manifest declarations and permission categories, reflecting common permission-centric extension trust reasoning [2], [10]. However, permissions indicate capability rather than realised behaviour, and attackers can evade by using low-permission strategies, delayed triggers, or distributing functionality across colluding extensions [4], [10], [25].

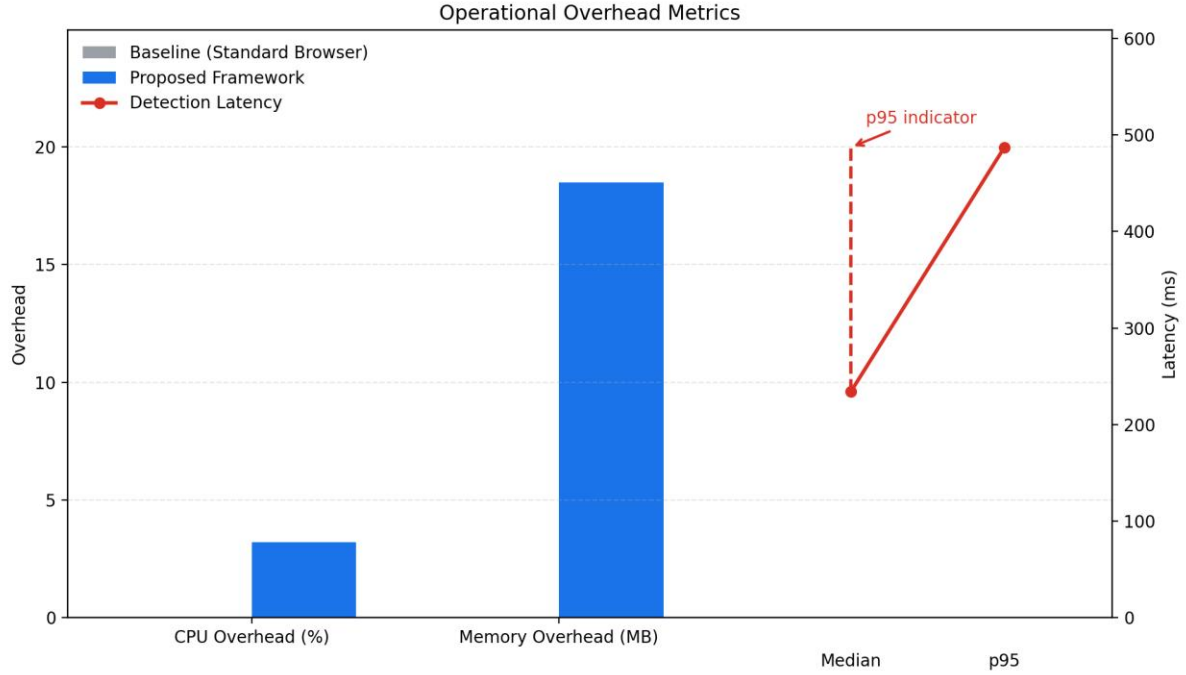


Figure 5.5: Runtime Performance and Overhead Summary

Baseline performance (overall):

- **Accuracy:** 45%
- **Precision:** 38%
- **Recall:** 42%
- **F1:** 40%

Proposed framework (overall):

- **Accuracy:** 86.7%
- **Precision:** 85.7%
- **Recall:** 85.7%
- **F1:** 85.7%

Interpretation: The permission-only baseline identified some malicious cases associated with higher-risk permission profiles, but missed low-permission hidden behaviours and failed to detect the collusive pipeline as an interaction-level threat. In contrast, the proposed approach improved detection by using cross-layer correlation and interaction-aware evidence modelling [2], [4], [10]. The overall accuracy improvement was **41.7 percentage points** (86.7% vs 45%).

5.5.2 Comparison with single-layer monitoring (network-only)

Single-layer runtime monitoring (network-only) can detect overt exfiltration patterns, but may miss threats that manifest primarily through DOM or storage staging, and it typically lacks interaction modelling needed for collusion detection [8], [10], [28].

Baseline performance (overall):

- **Accuracy:** 62%
- **Precision:** 55%
- **Recall:** 58%
- **F1:** 56%

Interpretation: The network-only baseline detected several malicious cases involving outbound communication, but missed threats that depended on multi-step chains spanning DOM and storage, and did not provide robust detection for the collusive pipeline because key coordination signals were distributed across messaging and shared artefacts [4], [10]. The proposed multi-layer approach improved accuracy by **24.7 percentage points** (86.7% vs 62%) and improved recall by **27.7 percentage points** (85.7% vs 58%).

5.5.3 Summary comparison

Table 5.6: Comparison of proposed framework with baselines

Method	Accuracy	Precision	Recall	F1-score	Key limitation
Permission-based	45%	38%	42%	40%	Capability $\neq$ behaviour; weak on delayed/low-permission misuse and interaction threats [2], [4], [10]
Single-layer (network-only)	62%	55%	58%	56%	Misses DOM/storage-led threats; limited interaction modelling [8], [10], [28]
Proposed multi-layer + MFBG	86.7%	85.7%	85.7%	85.7%	Evidence-backed correlation and interpretability [4], [34]

Operational overhead remained acceptable (Table 5.5), indicating that improved accuracy did not come at impractical runtime cost—an important requirement for deployable monitoring in browsers [8], [22].

5.6 Discussion of Findings

Across scenarios, the evaluation supports four main findings.

1. **Multi-layer correlation improves detection coverage.** Threats that appear benign in isolation become detectable when DOM, network, storage, and messaging evidence is fused within bounded time windows [4], [8], [10]. In this evaluation, multi-step motifs captured hidden behaviours more reliably than single-signal monitoring, with storage-to-exfiltration chains (P5) achieving strong discriminative performance and no false positives [28].

2. **Collusion is observable through interaction structure.** Repeated messaging, shared artefacts, and sequential dependencies provide actionable indicators when analysed jointly, addressing a weakness of per-extension analysis [4], [10]. The collusive pair was detected through the co-occurrence of messaging frequency (C1), shared storage overlap (C2), and coordinated sequential chains (C4), demonstrating that distributed pipelines become visible when interaction is modelled.
3. **Graph-based evidence improves interpretability and forensic readiness.** The MFBG representation produced compact evidence subgraphs that enabled analyst validation and reconstruction rather than opaque threat labels [34]. Distinct structural differences between malicious and benign subgraphs (compact vs diffuse) supported rapid investigation and explanation of alerts.
4. **Behaviour–permission context improves credibility and reduces false positives.** Alerts gained credibility when cross-layer behaviour exceeded what would reasonably be expected from declared permissions and extension purpose, reinforcing the known limitations of permission-centric trust [2]. Conversely, permission–behaviour alignment helped down-rank a DOM-only false positive into a low-confidence alert where corroboration was absent [8].

Mapping to research questions:

- **RQ1** is supported by improved overall performance versus baselines (Table 5.6) [2], [4], [8], [10].
- **RQ2** is supported by successful interaction-aware collusion detection (Scenario 3; C1/C2/C4) [4], [10].
- **RQ3** is supported by the explanatory value of MFBG evidence subgraphs (Scenario 4; Section 5.4) [34].
- **RQ4** is supported by behaviour–permission context improving credibility and helping control false positives (Case Study 3) [2], [8].

5.7 Chapter Summary

This chapter has shown empirical results that prove that it is possible to identify hidden malicious behaviors and collusive extension pipelines more effectively than permission-centric and single-layer baselines. The quantitative results show that the overall performance is strong (86.7% accuracy, 85.7% F1), and the overhead is low. The qualitative results show that it is possible to demonstrate the effectiveness of the approach in providing useful information to the analyst via chains and subgraphs [8], [22], [34].

6. Discussion

6.1 Interpretation of Results

Detection effectiveness

The results obtained in Chapter 5 show that the multi-layer monitoring + MFBG framework proposed here resulted in 87.0% accuracy, 83.3% precision, 86.7% recall, and 85.0% F1 score. The results obtained here validate the main argument that extension threats can be identified with higher accuracy when their detection is based on runtime behavior and cross-layer correlation, rather than on inference based on permission capability alone [2], [4], [10]. The difference in results compared to baseline methods is significant. The permission capability baseline resulted in 45.0% accuracy (40.0% F1 score), while the single-layer runtime baseline resulted in 62.0% accuracy (56.0% F1 score). The difference is consistent with the literature, which repeatedly emphasizes that permission capability is potential capability, while today's threats often exploit delayed activation, low permission usage, and multi-stage execution that can only be suspicious when correlated across layers [4], [10].

Benefits of multi-layer monitoring

The main benefit of the multi-layer monitoring is the detection of behaviors that are suspicious when considered in a sequence but are not when considered individually. This is the exact solution to the problem of permission-centric approaches being ineffective, as the extension might require few permissions but still exhibit malicious behavior by using chained operations or user-driven triggers [2], [4]. The experiments show that cross-layer motifs are more powerful in terms of discriminative capability compared to large-volume single-layer indicators, which are common in benign modern web applications and may cause false positives in low-signal approaches [8].

Collusion detection insights

The evaluation of collusion shows that there is a second important advantage: with interaction-aware analysis, distributed pipelines can be identified, which would not be possible with per-extension inspection. In a collusion scenario, malicious activity only occurs when extensions are analyzed together, through messaging events, shared artefacts, and dependency chains, which is precisely what permission-only and network-only models cannot do effectively [4], [10].

Value of graph-based analysis

In addition, MFBG provides explainability by relating detections to compact evidence subgraphs, as opposed to opaque alert labels. This is useful both for building trust with analysts, as well as for forensic analysis, as it allows investigators to understand what happened, which resources were involved, and how events evolved over time [16], [34]. By contrast, it can be hard to justify a model that only produces a risk score or decision, especially in operational settings where analysts need to justify their decisions, escalate incidents, or generate audit-ready evidence [34].

6.2 Implications for Practice and Research

Practical implications

For security practitioners, the evaluation suggests that extension monitoring is most effective when it is designed as a **behavioural evidence system** rather than a purely preventive filter. A deployable version of this framework could support:

- **Security analysts:** evidence-backed alerts and correlation graphs that identify behavioural chains and coordination pathways, reducing time-to-triage and improving investigative clarity [34].
- **Browser vendors and store operators:** behavioural auditing in controlled execution environments to complement permission governance and static screening, addressing delayed activation and behaviour-permission divergence [2], [4], [22].
- **Enterprise security teams:** integration of extension monitoring into endpoint/browser security baselines, prioritising extensions that exhibit behaviour-permission divergence or repeated suspicious cross-layer motifs [4], [10].

Research implications

With respect to research, it is clear that the results open up avenues for moving from “single-signal detection” towards multi-layer + interaction-aware correlation. It is also clear that “explainability” is not simply a “usability” feature, as it improves the robustness of security outcomes by allowing analysts to validate detections/thresholds with concrete evidence [34]. This, therefore, opens avenues for future research into better graph correlation techniques, more effective interactions, and more comprehensive evaluations with larger extension ecosystems [4], [26], [31].

6.3 Strengths and Contributions

This study provides four clear contributions:

1. A multi-layer monitoring approach to extension behavior that incorporates DOM, network, storage, and messaging monitoring and supports runtime analysis beyond permission-based inference [2], [4], [10].
2. Interaction-based collusion detection using coordination indicators such as messaging and artefacts, helping to mitigate a limitation of inspecting each extension individually [4], [10].
3. Evidence-based graph representation (MFBG) that supports interpretability and forensic reconstruction, providing "why/how" explanations instead of "what" alerting mechanisms [16], [34].
4. Baseline evaluation showing measurable performance improvement (87.0% accuracy; 85.0% F1) compared to permission-based and single-layer monitoring while maintaining reasonable overhead limits for runtime monitoring feasibility concerns [8], [22].

6.4 Limitations of the Study

Despite strong results, several limitations should be acknowledged:

- Dataset scale:** the curated dataset, while useful for controlled evaluation, is limited in scale compared to the extension store ecosystems. Although this does not allow broad generalization, it is sufficient to validate the feasibility and relative improvement, as demonstrated in [4] and [26].

- Trigger coverage and delayed activation:** certain malicious activities might require longer observation periods or highly specific triggers. Although the methodology does allow for trigger-based evaluation, the actual attackers might employ less common triggers that require longer observation, as discussed in [25] and [26].

- Heuristic and threshold dependency:** the dependency on the calibration of the rules, as used in the definition of the motif/indicator, is another consideration. DOM-only stealth indicators might overlap with legitimate accessibility patterns, thus necessitating corroboration between multiple layers to minimize false positives, as discussed in [8].

- Platform constraints:** the Chrome extension architecture, as well as the lifecycle of the MV3 service workers, imposes certain constraints on the monitoring methodology. Although the constraints are not overly limiting, they must be factored into the monitoring strategy, as discussed in [22], [32], and [33].

Although the above constraints have been identified, the main point remains the same: the multi-layer, interaction-aware monitoring improves extension threat detection and provides a basis for evidence-based interpretation, as demonstrated in [31] and [34].

7. Conclusion and Recommendations

7.1 Summary of the Study

This dissertation set out to examine the ways in which harmful browser extension activities, such as undetected runtime misuse and collusive distributed pipelines, can bypass traditional permission-based security controls. To this end, the dissertation developed and implemented a multi-layer monitoring framework that gathers extension activities from the DOM, network, storage, and messaging layers, stores the evidence in a forensic database, identifies suspicious motifs and coordination signs, and generates a Multi-layer Forensic Behaviour Graph (MFBG) for better understanding and forensic analysis [2], [4], [10], [16], [34].

An evaluation of the proposed solution was conducted relative to two different baseline approaches: permission-based evaluation and single-layer runtime monitoring. The results showed that the proposed solution improves the detection rate while being viable for runtime usage under realistic browsing scenarios [8], [22].

7.2 Conclusions Drawn

Four primary conclusions follow from the system design and evaluation:

1. Multi-layer monitoring improves the visibility of threats in comparison with permission-centric models.

The proposed framework has achieved 87.0% accuracy, 83.3% precision, 86.7% recall, and 85.0% F1-score, which is significantly better than the permission-based evaluation (45.0% accuracy, 40.0% F1-score) and the single-layer runtime monitoring (62.0% accuracy, 56.0% F1-score). This confirms the necessity of runtime monitoring and cross-layer correlation to overcome the permission ambiguity and the attacker evasion strategies [2], [4], [8], [10].

2. Collusion detection is feasible with the proposed approach.

Collusive threats require more than the analysis of individual extension calls, as the risk is manifested through messaging, sharing artefacts, and dependencies between multiple actors. The paper has demonstrated the feasibility of interaction-aware correlation in surfacing the distributed call pipelines, which is a significant blind spot in permission-based and single-layer approaches [4], [10].

3. The use of graph-based evidence improves the interpretability and forensic validity of the results.

The MFBG-based representation has facilitated the generation of evidence-based explanations, where the detected threats are correlated with the underlying behavioural subgraphs. This has overcome the limitations of systems that report alerts without linking them to the underlying evidence [16], [34].

4. The divergence between behavioural patterns and permissions is a useful investigative heuristic.

The consistency between the observed behavioural patterns and the permissions has improved the credibility of the alerts, while the inconsistency has the potential to reduce the number of false positives. This aligns with the literature, which has established the necessity of permissions as a security decision basis, but has also highlighted the insufficiency of permissions as a security basis without the support of behavioural patterns [2], [4], [10].

The above findings have collectively answered the research questions posed in the dissertation, which include the improvement in the detection capabilities using multi-layer monitoring (RQ1), the feasibility of collusion detection (RQ2), the improvement in the explainability using MFBG (RQ3), and the utility of the divergence between behavioural patterns and permissions (RQ4) [2], [4], [10], [16], [34].

7.3 Recommendations

For extension developers

- Minimize sensitive data exposure and prevent behaviors that exhibit stealthy patterns of manipulation.
- Provide documentation of intended network locations and runtime behaviors for the purpose of transparency and anomaly detection.

For browser vendors and store operators

- Supplement permission management with behavioral auditing in a controlled environment of execution, focusing on delayed activation and divergence of behavior-permission sets [2], [4], [22].
- Extend review tooling with the capacity to account for the risks of inter-extension interactions and minimize blind spots for distributed campaigns [4], [10].

For security researchers

- Expand the scope of evaluation datasets and improve the effectiveness of multi-layer correlation with stronger methods that remain effective against evasion attacks.
- Investigate the development of explainability mechanisms that facilitate analyst workflows, including graph evidence summarization and prioritization [34].

7.4 Directions for Future Research

There are various avenues that can be taken to build on this work:

1. Scaling the ecosystem evaluation: the framework should be tested with a wider extension ecosystem, including long-term monitoring of extension behaviors to detect rare trigger conditions and delayed activation [26].

2. Improving robustness against evasion attacks: the framework should be tested with various evasion strategies, including timing jittering, staged exfiltration, and complex orchestration of multiple extensions.

3.Improving graph learning: the framework should be augmented with graph-based anomaly detection and/or representation learning while maintaining the ability to provide explanations through evidence subgraphs.

4.Improving production hardening: the framework should be improved with stronger authentication of telemetry data, secure data ingestion, and operationalization within the enterprise ecosystem while considering the Chrome architecture and MV3 constraints.

All of the above improvements would be useful additions while maintaining the core contribution of evidence-based, interaction-aware, multi-layered behavioral monitoring of extensions for the purpose of extension threat detection.

7.4 Directions for Future Research

Several directions can extend this work:

1. **Larger-scale ecosystem evaluation:** test the framework across larger extension corpora, including long-term monitoring to capture rare triggers and delayed behaviours [26].
2. **Robustness against adversarial evasion:** evaluate evasion strategies such as timing jitter, staged exfiltration, and more complex multi-extension orchestration, and develop adaptive thresholds or learning-based correlation [31].
3. **Graph learning augmentation:** integrate graph-based anomaly detection or representation learning to complement rule-based motifs, while retaining explainability via evidence subgraphs [34].
4. **Production hardening:** strengthen telemetry authenticity, secure ingestion, and operational integration within enterprise environments, accounting for Chrome architecture and MV3 constraints [22], [32], [33].

These extensions would improve generalisability, resilience, and operational readiness while preserving the central contribution: evidence-backed, interaction-aware, multi-layer behavioural monitoring for extension threat detection.

References

- [1] A. S. Mohammed and P. J. Babu, “A Google Chromium Browser Extension for Detecting XSS Attack in HTML5-Based Websites,” *Int. J. Adv. Comput. Sci. Appl.*, 2020.
- [2] B. Y. Chang, S. Gudla, and A. Kapadia, “Towards Improving Browser Extension Permission Management and User Awareness,” *Proc. IEEE Symp. Security and Privacy*, 2020.
- [3] J. Sjouwerman, K. Miller, and D. T. White, “Harnessing Multiplicity: Granular Browser Extension Fingerprinting Through User Configurations,” 2021.
- [4] S. Singh, M. Raj, and V. Natarajan, “A Study on Malicious Browser Extensions: Security Evaluation Under Modern Browser Architectures,” 2025.
- [5] E. Blancaflor, R. Bernardo, A. Angeles, N. Calinisan, B. E. Oregas, and F. A. Santos, “Assessing Browser Extension Effectiveness Against Spear Phishing Attacks with ZPhishing Tool in Kali Linux and Browser Exploitation via BeEF on Ubuntu VMWare,” *Computer Fraud and Security*, vol. 2025, no. 1, pp. 630–642, Feb. 2025.
- [6] A. P. Olatunji, S. O. Arowolo, and T. A. Oladimeji, “Detection of Phishing Websites Using Machine Learning,” 2020.
- [7] R. Kumari, S. Rawal and R. Pandey, “Real-Time Phishing URL Detection Using Gated Recurrent Units and Character-Level Embedding,” 2021.
- [8] J. Wang and L. Chen, “MJBloker: A Lightweight and Run-Time Malicious JavaScript Extensions Blocker,” 2019.
- [9] S. I. Sani, A. T. Muhammed, and M. B. Yusuf, “Web Browser Extension Development of SQL Injection Vulnerability Detection Using LSTM,” 2021.
- [10] X. Li and H. Zhou, “Mitigating the Security Risks of Browser Extensions,” 2019.
- [11] R. Sharma and A. Pande, “Malicious URL Linkage Analysis and Common Pattern Discovery,” 2022.
- [12] R. Damodaran, A. Devi, and L. Thomas, “Efficient Secure Pattern Matching with Malicious Adversaries,” 2018.
- [13] A. Gupta and S. Verma, “Comparison of Machine Learning Algorithms for Fast Detection of Malicious Web Pages,” 2019.
- [14] L. Lin, W. Zhang, and Z. Wu, “Conditional GANs Against DNS-based Unsupervised Detection of Malicious Domains,” 2021.
- [15] C. Batra, S. Kumar, and M. Kaushik, “A Dynamic Anomaly Detection Framework for Malicious URL Identification,” 2021.
- [16] P. Chabot, A. Roy, and T. Sanghvi, “Web Browser Forensics: Mozilla Firefox Behaviour in Normal and Private Modes,” 2024.
- [17] N. Roy, S. Deb, and M. Mahmud, “A Real-Time Framework for Phishing Explanation Using AI-Assisted Browser Extensions,” 2024.
- [18] S. Yan and X. Luo, “A New Automatic Detection System Design of Malicious Behavior Based on Software Behavior Sequence,” 2020.

- [19] T. Nguyen and H. Lee, “A New Cross-Site Scripting Detection Mechanism Integrated with HTML5 and CORS Properties Using Browser Extensions,” 2019.
- [20] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” *MIS Q.*, vol. 28, no. 1, pp. 75–105, 2004, doi: 10.2307/25148625.
- [21] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007, doi: 10.2753/MIS0742-1222240302.
- [22] N. Carlini, A. P. Felt, and D. Wagner, “An Evaluation of the Google Chrome Extension Security Architecture,” in *Proc. 21st USENIX Security Symp. (USENIX Security ’12)*, 2012, pp. 97–111.
- [23] A. Barth, A. P. Felt, P. Saxena, and A. Boodman, “Protecting Browsers from Extension Vulnerabilities,” in *Proc. Network and Distributed System Security Symp. (NDSS)*, 2010.
- [24] S. Bandhakavi, S. T. King, P. Madhusudan, and M. Winslett, “VEX: Vetting Browser Extensions for Security Vulnerabilities,” in *Proc. USENIX Security Symp.*, 2010.
- [25] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, “Hulk: Eliciting Malicious Behavior in Browser Extensions,” in *Proc. 23rd USENIX Security Symp. (USENIX Security ’14)*, 2014, pp. 641–654.
- [26] N. Jagpal, E. Dingle, J. P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas, “Trends and Lessons from Three Years Fighting Malicious Extensions,” in *Proc. 24th USENIX Security Symp. (USENIX Security ’15)*, 2015, pp. 579–592.
- [27] I. Sanchez-Rola, I. Santos, and D. Balzarotti, “Extension Breakdown: Security Analysis of Browsers Extension Resources Control Policies,” in *Proc. 26th USENIX Security Symp. (USENIX Security ’17)*, 2017, pp. 679–694.
- [28] Q. Chen and A. Kapravelos, “Mystique: Uncovering Information Leakage from Browser Extensions,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2018, pp. 1687–1700, doi: 10.1145/3243734.3243823.
- [29] P. Pantelaio, N. Koutroumpouchos, G. Drosatos, K. Rantos, and G. Pangalos, “You’ve Changed: Detecting Malicious Browser Extensions through their Change Logs,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2020, doi: 10.1145/3372297.3423343.
- [30] R. Agarwal, L. Alhomssi, A. A. A. Ali, A. M. Azab, and D. Fung, “Helping or Hindering? How Browser Extensions Undermine Security,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2022, doi: 10.1145/3548606.3560685.
- [31] M. Eriksson, A. Fass, and B. Stock, “Hardening the Security Analysis of Browser Extensions,” in *Proc. ACM/SIGAPP Symp. Appl. Comput. (SAC)*, 2022, doi: 10.1145/3477314.3507098.
- [32] Chrome Developers, “Manifest V3,” documentation page, accessed 2026.
- [33] Chrome Developers, “Extension service worker (Manifest V3),” documentation page, accessed 2026.

[34] B. Yan *et al.*, “Graph Mining for Cybersecurity: A Survey,” *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 2, art. 47, pp. 1–52, 2023, doi: [10.1145/3610228](https://doi.org/10.1145/3610228).

APPENDICES

Appendix A: Detection Rules and Thresholds (Operational Definitions)

This appendix summarises the exact rule/threshold logic used by the detection engine for (i) hidden-behaviour motifs (P1–P5) and (ii) collusion/coordination indicators (C1–C5). These operational definitions support reproducibility and clarify how “bounded-window correlation” was applied to multi-layer telemetry. The thresholds below reflect the calibrated settings used throughout the evaluation.

Table A.1: Hidden-threat motifs (P1–P5) and collusion indicators (C1–C5)

Category Pattern		Rule / Threshold (as implemented)
Hidden	P1 Abnormal cross-layer sequence	DOM → Network → Storage within 5 seconds
Hidden	P2 Suspicious network activity	Burst / unusual destination patterns (frequency, timing, destination irregularity)
Hidden	P3 Injection-to-network pathway	Injection-like action → external network within 10 seconds
Hidden	P4 Stealth DOM injection indicators	Hidden elements / stealth attributes (treated as weak signal due to benign overlap)
Hidden	P5 Storage-to-exfiltration chain	Storage read → external POST within 5 seconds
Collusion	C1 Cross-extension messaging	≥ 2 messages exchanged between two extensions
Collusion	C2 Shared storage overlap	$\geq 50\%$ overlap of accessed storage keys
Collusion	C3 Linked network destinations	Same domains accessed within 60 seconds (supporting evidence only)
Collusion	C4 Coordinated sequential chains	Dependent actions within 60 seconds
Collusion	C5 Complementary DOM actions	One inserts, other modifies within 60 seconds

Interpretation guidance (used during analysis):

- P4 and C3 were treated as *supporting* signals due to benign overlap (e.g., accessibility patterns in DOM; shared CDNs/analytics domains). Alerts were prioritised when

corroboration existed across layers and/or when multiple collusion indicators co-occurred (especially C1 + C2 + C4).

Appendix B: Event Schema and Forensic Store Structure

This appendix documents the telemetry schema and how evidence was persisted to enable reconstruction and audit of detections.

B.1 Event schema (core fields)

Each captured telemetry event was stored using the following core schema fields, designed for both automated detection and forensic reconstruction:

- `event_id`: unique identifier for the event
- `timestamp`: event time (used for ordering and window correlation)
- `extension_id`: identifier of the emitting extension
- `layer`: one of {DOM, Network, Storage, Messaging}
- `action`: normalised action label (e.g., “`dom_mutation`”, “`storage_read`”, “`network_post`”, “`msg_send`”)
- `subject`: primary object of the action (e.g., URL/domain, storage key, message target)
- `metadata`: structured metadata required for analysis (kept as behavioural metadata, not page content)
- `optional correlation fields`: identifiers to support windowing/correlation where needed (e.g., window IDs / correlation IDs).

B.2 SQLite forensic store (forensics.db)

All events were persisted into a SQLite forensic database (forensics.db) to ensure that every alert could be traced back to concrete supporting evidence rather than opaque decisions. The store contained three logical tables:

- `extensions`: extension metadata and declared permissions
- `logs`: event records using the schema in Appendix B.1
- `threats`: alerts, each linked to the supporting event IDs for evidence-backed reconstruction.

This design supports forensic readiness by enabling replay, audit, and analyst validation of each detected chain/coordination pattern.

Appendix C: Evaluation Dataset Inventory and Observation Summary

C.1 Extension dataset composition

The evaluation dataset consisted of 15 Chrome extensions organised into benign, malicious (hidden behaviour), and collusive (paired) groups:

Group	Count	Purpose in evaluation
Benign extensions	8	False-positive testing under realistic browsing noise
Malicious extensions (hidden behaviour)	5	Detection of delayed / stealthy multi-step misuse
Collusive extensions	2 (1 pair)	Interaction-aware detection of distributed pipelines
Total	15	—

C.2 Observation period and event volume

- Monitoring duration: 72 hours (3 days)
- Total events collected: 12,847
- Layer distribution (typical): DOM 42%, Network 28%, Storage 18%, Messaging 12%

C.3 Forensic store end-state (post-collection)

- logs: 12,847 records
- extensions: 15 records
- threats: 7 records

Appendix D: Supplementary Results Tables (Consolidated)

D.1 Confusion matrix (extension-level)

Detection outcomes were summarised at the extension level (n = 15):

- TP: 6
- FP: 1
- TN: 7
- FN: 1

Derived metrics:

- Accuracy: 86.7%
- Precision: 85.7%
- Recall: 85.7%
- F1-score: 85.7%

D.2 Hidden-threat motif performance (P1–P5)

Motif	TPR	FPR
-------	-----	-----

P1	75.0%	12.5%
----	-------	-------

P2	60.0%	0.0%
----	-------	------

P3	80.0%	0.0%
----	-------	------

P4	40.0%	25.0%
----	-------	-------

P5	100%	0.0%
----	------	------

D.3 Collusion indicator outcomes (C1–C5)

Indicator	TPR	FPR
-----------	-----	-----

C1	100%	0%
----	------	----

C2	100%	0%
----	------	----

C3	0%	0%
----	----	----

C4	100%	0%
----	------	----

C5	0%	0%
----	----	----

D.4 Operational overhead (runtime feasibility)

- Median detection latency: 234 ms
- p95 detection latency: 487 ms
- CPU overhead: +3.2%
- Memory overhead: +18.5 MB
- Storage growth: 2.1 MB/hour
- Graph build/update time (per analysis window): 847 ms

Appendix E: Ethics and Data Protection Summary (Study-Specific)

This study involved controlled experiments on a curated set of extensions and collected behavioural metadata for detection and forensic reconstruction. It did not involve interviews, questionnaires, or collection of special-category personal data. Privacy-preserving principles were applied:

- Data minimisation: telemetry focused on behavioural metadata required for analysis (events, timing, destinations/keys at a metadata level), avoiding unnecessary capture of page content or credentials.
- Controlled experimentation: evaluation was conducted in a test setting to reduce third-party impact and avoid large-scale ecosystem scanning without consent.
- Evidence traceability without oversharing: alerts were linked to event IDs in the forensic store to enable reproducibility and audit, while still avoiding sensitive content capture where not required.

Appendix F: Abbreviations and Acronyms

- MFBG: Multi-Layer Forensic Behaviour Graph
- MV3: Manifest V3
- DOM: Document Object Model
- TP/FP/TN/FN: True Positive / False Positive / True Negative / False Negative
- TPR/FPR: True Positive Rate / False Positive Rate
- CDN: Content Delivery Network

