WEB DECEPTION FOR WEB APPLICATION SECURITY: FROM LAYERED DEFENSE IMPLEMENTATION TO LLM-ERA ADVERSARY FINGERPRINTING

A THESIS

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted by

AMAN SOLANKI (2K24/ISY/12)

Under the supervision of

Dr. Ritu Agarwal

(Associate Professor)

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2026

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Aman Solanki, Roll No. 2K24/ISY/12, student of M.Tech (Information Technology), hereby declare that the project dissertation titled "Web Deception for Web Application Security: From Layered Defense Implementation to LLM-Era Adversary Fingerprinting", which is submitted by me to the Department of Information Technology, Delhi Technological University, Delhi, in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

Aman

AMAN SOLANKI

(2K24/ISY/12)

Place: Delhi

Date: 29/05/2026

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

Certified that Aman Solanki (Enrollment No. 2K24/ISY/12) has carried out the research work presented in this dissertation titled "Web Deception for Web Application Security: From Layered Defense Implementation to LLM-Era Adversary Fingerprinting" for the award of the degree of Master of Technology in Information Technology from the Department of Information Technology, Delhi Technological University, Delhi, under my supervision. The dissertation embodies results of original work, and the studies have been carried out by the student himself, and the contents of the dissertation do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University or Institution.

Place: Delhi

Date: 29/05/2026

29/5/25

Dr. Ritu Agarwal

(Associate Professor)

Department of Information Technology

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to my supervisor, Dr. Ritu Agarwal, Associate Professor, Department of Information Technology, Delhi Technological University, for her invaluable guidance and mentorship throughout this research. Her insistence on methodological rigour — in particular her suggestions to incorporate a timing-ablation defensibility check on the classifier and to broaden the language-model panel beyond a single vendor — materially strengthened the contributions reported here.

I am grateful to Dr. Dinesh Kumar Vishwakarma, Head of the Department of Information Technology, and to every faculty member who shaped my graduate training. The departmental laboratories, computing infrastructure, and access to commercial-grade inference platforms made the empirical work in this dissertation practicable on a student timeline.

I also acknowledge the contributors of the open-source software stack the experiments rely on — Playwright, scikit-learn, pandas, and the Groq client library — and the maintainers of the Llama and gpt-oss model families that Groq exposes as free-tier endpoints. Finally, I thank my family and my peer group for their constant support during the long iterations of harness debugging and dataset re-collection that this project demanded.

Aman

Place: Delhi

AMAN SOLANKI

Date: 29/05/2026

(2K24/ISY/12)

Abstract

The widespread adoption of internet-based services has propelled web applications to the forefront of modern attack surfaces. Servers handling sensitive user information and transactional operations are now persistent targets, motivating defensive measures that complement, rather than replace, conventional perimeter controls such as web application firewalls. Deception-based security — the deliberate placement of engineered baits that have no legitimate operational use, so that any interaction with them is treated as an intrusion signal — has matured into a complementary defensive layer drawn from the long honeypot tradition. This dissertation investigates the design, implementation, and empirical evaluation of layered web deception against a heterogeneous adversary population that includes a recently-emerged class of attacker: autonomous agents driven by large language models (LLMs).

The work proceeds in two phases. Phase I establishes a deception substrate on a MERN-stack electronic-commerce application using four wordlist-style primitives consistent with established honeypot practice (lure HTML comments carrying fake administrative URLs, decoy session cookies, off-screen honeypot form fields, and an attempt counter that escalates to a temporary blocking mode). Phase II extends this substrate with two novel *semantic-class* probes designed to elicit a distinguishable interaction signature from LLM agents: each encodes a non-guessable URL inside page-rendered natural-language content (an HTML comment and an `ai-instruction` meta tag), so that the URL can be discovered only by an adversary that actually reads the page content rather than by enumerating directory wordlists.

A measurement testbed comprising a process-safe trace logger, a probe-identification module, and five adversary harnesses (simulated human shoppers, wordlist scanners, credential-stuffing bots, and LLM-driven agents) is built around the application. Five distinct language models from two vendors (Meta llama-3.1-8b, llama-4-scout-17b, llama-3.3-70b; OpenAI gpt-oss-20b, gpt-oss-120b) are evaluated. Across 285 clean sessions,

balanced to a uniform n=30 per class (225 sessions in the balanced dataset; A5 retained at its true collected count of n=15 owing to a daily-token cap on the 70B model), the research reports three principal findings within this testbed. First, the contribution is not that every probe fires, but that the semantic probes exhibit selective activation: 0 of 90 non-LLM sessions triggered either probe, while the mean semantic-probe hit rate across the five LLM classes was 70.0 per cent (pooled session-level rate 66.7 per cent, 90 of 135). Second, a Random Forest classifier separates the four super-classes (humans, scanners, credential stuffers, LLM agents) at 100 per cent five-fold cross-validation accuracy, and this accuracy is preserved when all timing features are removed and, separately, when all direct probe-hit features are removed; the joint ablation retains 99.6 per cent accuracy. Third, the observed semantic-probe hit rate increases monotonically with model capability (0, 53, 97, 100, and 100 per cent) when models are sorted by published parameter count, and the gradient is observed across both Meta and OpenAI model families within this testbed.

A secondary contribution surfaces a structural placement weakness in the decoy-cookie primitive (P4), whose trigger URL is fetched unconditionally by the front-end on every page mount, producing a 100 per cent observed human false-positive rate; a four-probe subset is shown to attain 84.6 per cent bot true-positive rate at 0 per cent observed human false-positive rate. The legitimate-user baseline is a simulated shopper population generated with Playwright; the false-positive analysis (RQ3) is accordingly scoped to this simulated-user population.

Keywords: web application security; web deception; honeypots; honeytokens; semantic canaries; LLM agents; adversary fingerprinting; intrusion detection; false-positive rate; capability gradient.

Contents

Candidate's Declaration **i**

Certificate **ii**

Acknowledgement **iii**

Abstract **v**

Content **viii**

List of Tables **ix**

List of Figures **x**

List of Symbols, Abbreviations **xi**

1 INTRODUCTION **xii**
1.1 General . . . xii
1.2 Background on Deception Defence . . . xii
1.3 Problem Statement . . . xiii
1.4 Research Questions . . . xiii
1.5 Objectives . . . xiv
1.6 Scope and Limitations . . . xiv
1.7 Contributions . . . xiv
1.8 Organisation of the Dissertation . . . 1

2 LITERATURE REVIEW **2**
2.1 Foundations: Honeypots and Honeytokens . . . 2
2.2 Web Application Honeypots . . . 2
2.3 Deceptive Primitives in Modern Web Applications . . . 3
2.4 Anomaly Detection and Behavioural Bot Fingerprinting . . . 3
2.5 LLM Agents and Web Automation . . . 4
2.6 Research Gaps . . . 4

3 SYSTEM DESIGN AND METHODOLOGY **5**
3.1 Architectural Overview . . . 5
3.2 Phase I — Baseline Deception Primitives . . . 6
3.2.1 Deceptive HTML Comments and Fake Administrative URLs . . . 6
3.2.2 Decoy Session Cookies . . . 6
3.2.3 Honeypot Form Fields and Off-Screen Bait . . . 6
3.2.4 Attempt Counter and Adaptive Blocking . . . 7

3.3 Phase II — Semantic Canaries Targeting LLM Agents . . . 7
3.4 Measurement Layer: Trace Schema . . . 7
3.5 Adversary Classes . . . 8
3.6 Per-Session Feature Extraction . . . 9

4 IMPLEMENTATION 10
4.1 Application Architecture . . . 10
4.2 Phase I Implementation . . . 10
4.3 Phase II Implementation . . . 10
4.4 The Adversary Harnesses . . . 11
4.4.1 Scanner Harness . . . 11
4.4.2 Credential-Stuffer Harness . . . 11
4.4.3 LLM-Agent Harness . . . 11
4.4.4 Human-Shopper Simulation . . . 11
4.5 Orchestration and Analysis Pipeline . . . 12

5 EXPERIMENTAL SETUP 13
5.1 Hardware and Software Environment . . . 13
5.2 Dataset . . . 13
5.3 LLM Models . . . 14
5.4 Hyperparameters . . . 14
5.5 Evaluation Methodology . . . 14

6 RESULTS AND ANALYSIS 15
6.1 Phase I Baseline Results . . . 15
6.2 RQ1 — Probe Coverage . . . 16
6.3 RQ2 — Classifier Performance . . . 17
6.4 Timing-Ablation Defensibility Check . . . 18
6.5 Probe-Feature Ablation (Leakage Defensibility Check) . . . 19
6.6 The Capability Gradient . . . 19
6.7 RQ3 — False-Positive Rate Against Human Shoppers . . . 20
6.8 A5 Small-n Caveat . . . 21

7 DISCUSSION 26
7.1 Capability Tier as a First-Class Adversary Dimension . . . 26
7.2 Design Rule: Avoid Universal-Path Deception . . . 26
7.3 Why the Ablations Matter . . . 27
7.4 Threats to Validity . . . 27
7.4.1 Internal validity . . . 27
7.4.2 External validity . . . 27
7.4.3 LLM-agent harness . . . 27
7.4.4 Scope of the simulated legitimate-user class . . . 27
7.4.5 The P5 instrumentation gap . . . 28
7.5 Comparison with the Phase I Baseline . . . 28
7.6 Practical Implications . . . 28

8 CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT 29
8.1 Summary of Findings . . . 29
8.2 Future Scope . . . 30

8.3 Social Impact 30

List of Tables

3.1 Phase I deception primitives and their detection mechanisms. 6
3.2 Phase II semantic-class probes (P6 and P7). 7
3.3 Trace-row schema (twelve fields). 8
3.4 Adversary classes and behavioural rationale. 8

5.1 Per-class session counts in the balanced dataset. 13
5.2 LLM adversaries used in Phase II. 14

6.1 Phase I OWASP-aligned interaction summary. 15
6.2 Overall probe trigger rate across the balanced dataset. 16
6.3 Per-class probe-class hit rate (balanced $n = 30$, A5 $n = 15$). 17
6.4 Classifier cross-validation results on the balanced dataset. 21
6.5 Timing-ablation verdict on the balanced dataset. 21
6.6 Probe-feature ablation verdict on the balanced dataset. 22
6.7 Capability gradient across LLM adversaries. 23
6.8 Top URL paths visited per LLM adversary 25
6.9 Per-probe human FPR and bot TPR on the balanced dataset. 25

List of Figures

3.1 High-level web-deception arrangement carried forward from Phase I: adversary tooling exercises the vulnerable web application, which is instrumented with deception artefacts; the deception strategies (probes) sit between the two and convert adversary interactions into intrusion signals. 5

6.1 Phase I: deceptive HTML comment carrying a fabricated backend URL, visible in the browser developer console. 16
6.2 Phase I: server response logged after the attacker follows the fabricated URL (intrusion attempt recorded). 17
6.3 Phase I: maintenance page served after repeated intrusion attempts cross the attempt-counter threshold (adaptive blocking). 18
6.4 Phase I: decoy session cookies (`session_id`, `debug_mode`) set on the client to attract session-manipulation attempts. 19
6.5 Per-class per-probe coverage heatmap (RQ1). Cell value is the fraction of that class's sessions that triggered the probe at least once. The zero-valued semantic-probe cells for simulated legitimate users, scanners, and credential stuffers are *correct negatives* (the probe correctly never triggered for classes it was not designed to catch), not missing data or failures to detect. 20
6.6 Confusion matrix, RandomForest, four-class, full features. Every off-diagonal cell is empty on the balanced dataset. 22
6.7 Confusion matrix, RandomForest, four-class, no-timing features. Accuracy retained at 1.000 in this dataset. 23
6.8 Confusion matrix, RandomForest, eight-class (legacy '5class' filename slug), full features. Residual confusion is concentrated within the LLM family at the high-capability tier. 24
6.9 Semantic-probe hit rate vs. model capability 24
6.10 Per-probe human FPR against bot TPR (RQ3). P4 is the outlier with the highest FPR and lowest TPR; the two semantic probes achieve observed 0% FPR. 25

List of Symbols, Abbreviations and Nomenclature

API	Application Programming Interface
CSV	Comma-Separated Values
CV	Cross-Validation
DOM	Document Object Model
FPR	False-Positive Rate
HTTP	HyperText Transfer Protocol
IDS	Intrusion-Detection System
JSON	JavaScript Object Notation
LLM	Large Language Model
MERN	MongoDB + Express.js + React + Node.js
OWASP	Open Worldwide Application Security Project
RF	Random Forest
RPM	Requests Per Minute
RQ	Research Question
SPA	Single-Page Application
TLS	Transport Layer Security
TPD	Tokens Per Day
TPM	Tokens Per Minute
TPR	True-Positive Rate
UA	User-Agent (HTTP header)
URL	Uniform Resource Locator
WAF	Web Application Firewall
ZAP	OWASP Zed Attack Proxy

Chapter 1

INTRODUCTION

1.1 General

Over the past decade the consumption of internet-mediated services has grown to a point at which web applications have become the dominant interface between organisations and their users. Communication, commerce, entertainment, banking, healthcare delivery, and government services now all rely on HTTP-delivered applications, which makes those applications the single most valuable category of target for an adversary seeking financial gain, intellectual property, or operational leverage. Industry telemetry consistently shows that the application layer — not the network or operating-system layer — is now where the majority of successful security breaches originate.

The standard defensive response, the web application firewall (WAF) operating on signature rules and regular expressions, was designed against a generation of attacker tooling whose request patterns are well understood and whose User-Agent strings are essentially self-identifying. WAF technology is mature and remains useful, but it has limits that have become more pronounced as attacker tooling has diversified: zero-day and logic-vulnerability traffic is by construction absent from any signature database, sophisticated scanners mutate their request patterns to evade fingerprinting, and an entirely new class of adversary — autonomous agents driven by large language models (LLMs) — produces request streams that resemble neither scanners nor humans and therefore evade rules designed for either.

Anomaly-based intrusion detection complements signature-based defence by flagging deviations from learned baselines of normal traffic, but it requires careful baseline construction, tends to produce a steady stream of false alarms that erode operator attention, and can be computationally expensive on a high-traffic site. The need for an additional defensive layer — one that produces low-false-positive signals against precisely the adversary classes that WAFs and anomaly detectors handle weakly — is therefore well motivated, and is the conceptual home of the deception techniques this dissertation investigates.

1.2 Background on Deception Defence

Deception in computer security is the deliberate placement of engineered artefacts whose only operational purpose is to be interacted with by an adversary. Because no legitimate user has any reason to interact with a deception artefact, an interaction is by construction an intrusion signal — there is no false-positive question to ask of an interaction whose target object has no legitimate use. The foundational work of Spitzner on honeypots formalised this idea at the network and system level [1], and the subsequent literature

on honeytokens extended it to individual data elements: a forged row in a database, a fake credential file, or an embedded non-existent URL whose appearance in any access log identifies an attacker who has read content they should not have read [2].

Web-application deception specialises these ideas to the HTTP/HTML layer. The primitives studied in the literature include fabricated administrative URLs reachable only by wordlist scanners, decoy cookies named to attract session-fixation attempts, honeypot form fields hidden off-screen, and canary metadata values whose later reappearance betrays a scraper that read them. These primitives have been deployed in academic and commercial honeynets for over fifteen years and form the Phase I substrate of the work presented here.

The recent emergence of LLM-driven autonomous agents — software systems in which a large language model is provided with a fetch tool, a parse tool, a form-submit tool, and a goal, and is then asked to autonomously achieve that goal on a target website — has created an adversary class for which the existing deception primitives were not designed. LLM agents execute JavaScript, read rendered content, synthesise novel actions in response to what they observe, and are not discouraged by user-interface friction. Whether the existing wordlist-based primitives retain their selectivity against such agents, and whether new primitives designed specifically for LLM agents can detect them while leaving real users unaffected, is the central question addressed by Phase II.

1.3 Problem Statement

Given a web application instrumented with a set of deception probes, and an observed user session represented as a sequence of HTTP, form, cookie, and Document Object Model events, can a measurement and classification pipeline (a) identify which class of adversary produced the session, (b) in particular separate LLM-driven autonomous agents from traditional scanners and credential-stuffing bots on the basis of probe-interaction patterns, and (c) bound the rate at which legitimate shoppers trigger probes so that the deception layer remains deployable? The classifier and probe set must be defensible: classifier accuracy must not be an artefact of infrastructure-imposed timing differences or of trivially memorising direct probe hits, and the false-positive rate against real users must remain near zero on every probe individually.

1.4 Research Questions

RQ1 (Probe coverage). Which deception probes are triggered by which adversary classes? This is the basic per-probe selectivity question; the per-class trigger matrix it produces is the cross-class coverage table on which the downstream analyses rely.

RQ2 (LLM distinguishability — the principal question). Do LLM-driven agents leave a behavioural signature on a deception-instrumented site that distinguishes them from traditional scanners and credential-stuffing bots? The hypothesis under test is that two semantic-class probes — an imperative HTML comment and an `ai-instruction` meta tag, each encoding a non-guessable URL — will be triggered by LLM agents that read page content while remaining invisible to scanners that operate on URL wordlists.

RQ3 (False-positive rate). How often do legitimate human users (simulated using Playwright with eleven weighted personas, in the absence of ethical access to real production user populations) trigger each deception probe? Deployable deception requires the

per-probe false-positive rate to be near zero.

1.5 Objectives

1. Design and integrate a layered deception substrate into a representative MERN-stack e-commerce application, comprising both established wordlist-class primitives (Phase I) and novel semantic-class primitives aimed at LLM agents (Phase II).

2. Implement a measurement testbed comprising a process-safe append-only trace logger with a fixed schema, a probe-identification module, and adversary harnesses for human shoppers, scanners, credential stuffers, and LLM agents.

3. Collect a labelled dataset of clean adversary sessions spanning every implemented class and use it to evaluate the three research questions.

4. Establish the methodological defensibility of any positive classifier result by explicit ablations of timing features and of direct probe-hit features.

5. Derive practical deployment guidance, including the minimal probe subset that achieves an acceptable detection rate at near-zero false-positive rate, and a placement rule for deception primitives.

1.6 Scope and Limitations

The empirical work is conducted on a single MERN-stack application running locally on commodity hardware. The five language-model adversaries are restricted to those exposed by the Groq free-tier inference platform during the data-collection window, spanning two vendors (Meta and OpenAI) and a parameter range of approximately 8 billion to 120 billion. The legitimate-user class is a Playwright-driven simulated shopper population, a choice justified in Chapter 7 on ethical and methodological grounds; RQ3 is accordingly scoped to this simulated-user population. The dissertation does not evaluate prompt-injection defences, browser-extension automation, or mobile-app scrapers.

1.7 Contributions

1. A reusable measurement testbed comprising the deception-instrumented MERN application, five adversary harnesses, a process-safe trace logger, a feature extractor, and three analysis scripts addressing RQ1, RQ2, and RQ3.

2. Two novel semantic deception primitives (P6 and P7) that encode non-guessable URLs inside page-rendered natural-language content and thereby achieve strong class selectivity for LLM adversaries within this testbed.

3. An empirical capability-gradient finding: across five LLMs from two vendors, the observed semantic-probe trigger rate increases monotonically with published model capability, and the gradient is observed across both Meta and OpenAI model families within this testbed.

4. A four-class classifier achieving 100 per cent stratified cross-validation accuracy on the (human / scanner / cred-stuffer / LLM) distinction, with that accuracy preserved under both a timing-feature ablation and a direct-probe-feature ablation.

5. A documented structural design rule for deception primitives: a probe's trigger surface must not overlap with anything the application's own setup traffic touches, derived from the observation that the decoy-cookie probe (P4) fires on 100 per cent of human sessions because its trigger URL is fetched on every page mount.

1.8 Organisation of the Dissertation

Chapter 2 surveys related work in deception primitives, web honeypots, the LLM-agent security literature, and behavioural bot detection. Chapter 3 presents the system design. Chapter 4 documents the implementation. Chapter 5 specifies the experimental setup. Chapter 6 presents results for all three research questions, including the two defensibility ablations. Chapter 7 discusses implications, threats to validity, and limitations. Chapter 8 concludes and identifies the social impact and future-work agenda.

Chapter 2

LITERATURE REVIEW

The literature surveyed for this dissertation spans four overlapping research currents: the foundational honeypot and honeytoken tradition, web-application-specific deception techniques, behavioural bot- and anomaly-detection methods, and the rapidly-growing body of work on the security of LLM-driven agents. The review is organised thematically because the research questions straddle the boundary between mature (deception) and emerging (LLM-agent security) areas.

2.1 Foundations: Honeypots and Honeytokens

The systematic use of deception as a defensive primitive predates the modern web by decades. Stoll's account of the Cuckoo's Egg investigation introduced the operational technique of fabricating valuable-looking but actively monitored bait to trace the activities of a network intruder [3]. Spitzner subsequently formalised the honeypot concept and distinguished low-interaction systems, which emulate services superficially and capture connection-level metadata, from high-interaction systems, which expose real services and capture full attack traces [1]. Provos and Holz extended this taxonomy in their book-length treatment of virtual honeypots and the operational experience of running them at scale [4].

Honeytokens generalise the honeypot idea below the level of a complete system: a fake credential file, a fake row in a customer database, or a fake URL embedded in a site's HTML, any of which is intended to act as an intrusion signal upon interaction [2]. Honeytokens are particularly attractive for web-application defence because they can be distributed throughout the application's natural content without imposing new infrastructure, and their class selectivity can in principle be high when they are positioned carefully. The semantic-class probes proposed in Chapter 3 are honeytokens in exactly this sense.

2.2 Web Application Honeypots

Web-application honeypots specialise the general honeypot design to the HTTP and HTML layers. Open-source systems such as Glastopf and the Modern Honey Network capture interactions with deception URLs designed to attract specific classes of attacker tool [5]. Their empirical contribution is largely descriptive: they catalogue, at operational scale, what scanners do. They predate the LLM-agent threat model entirely, and none of them have been evaluated against an adversary that reads the rendered content of the

pages it fetches.

Closer to the present work, the line of research on decoy documents demonstrated that deception primitives carefully designed to be invisible to legitimate users can serve as low-false-positive intrusion signals [6]. The RQ3 methodology of this dissertation follows the same thread: per-probe human false-positive rate is measured explicitly, and the minimal probe subset under a configurable false-positive budget is identified using a greedy minimum-set-cover.

2.3 Deceptive Primitives in Modern Web Applications

Several focused studies examine specific deception primitives in modern web applications. Ishikawa and Sakurai propose a deception-proxy architecture in which the proxy fabricates parameter values matching the patterns of the underlying application and detects manipulation attempts when those parameters are returned in a non-canonical form [7]. De Faveri and Moreira present a design-oriented framework for adaptive deception strategies, arguing that the choice of primitive should depend on the threat model rather than being fixed in advance [8]. Almeshekah and Spafford survey the broader design space of cyber-security deception and identify drive-by-download bait, social-engineering decoys, and phishing-infrastructure traps as related operational applications [9].

Deepa and Thilagam survey the state of the art in securing web applications against injection and logic vulnerabilities, reviewing seventeen studies on SQL injection, thirty-five on cross-site scripting, and thirty-four on logic flaws [10]. Their structural observation — that defensive techniques cluster by the phase of the software lifecycle at which they operate — informs the design choice in Chapter 3 to instrument the deception layer inside the application code itself rather than at a network proxy.

2.4 Anomaly Detection and Behavioural Bot Fingerprinting

The bot-detection literature has developed a large feature vocabulary for distinguishing automated traffic from human users: header consistency, TLS-fingerprint variability, mouse-movement biometrics, JavaScript-engine quirks, and request-timing distributions. Ahmed, Mahmood, and Hu survey network anomaly-detection techniques and group them into classification-based, statistical, information-theoretic, and clustering-based methods, identifying scalability to high-traffic settings as a recurring open challenge [11]. Rathore, Ahmad, and Paul evaluate decision-tree-family classifiers in an ultra-high-speed intrusion-detection setting and report false-positive rates below 0.001 per cent for J48 and REP Tree [12], providing precedent for the use of decision-tree-based classifiers in this work.

The limitation of generic bot-detection features for the present study is that they do not natively distinguish *sub-classes* of bots — a scanner, a credential stuffer, and an LLM agent all fail human-like-mouse-movement checks indistinguishably for management purposes. The contribution of this dissertation to the bot-detection literature is to show that probe-interaction features — which sub-class of deception primitive each session triggers — carry sub-class-discriminative information that the established feature set does not.

2.5 LLM Agents and Web Automation

The defensive question of how to detect LLM-driven automation of web applications has received considerably less attention than the corresponding offensive questions. Greshake et al. demonstrate that LLM agents are vulnerable to indirect prompt injection through malicious content embedded in the pages they fetch [13], and Yang et al. introduce the InjecAgent benchmark for measuring this vulnerability [14]. Zhou et al. introduce WebArena as a realistic environment for building and evaluating autonomous LLM-driven web agents [15]. None of these works addresses the inverse question — how a target site can detect that it is being explored by an LLM agent rather than by a traditional scanner or a real user — which is the question this dissertation investigates empirically.

2.6 Research Gaps

Relative to the existing literature, several gaps are evident. Static signature-based defences perform poorly against zero-day and logic-vulnerability traffic; scalability and real-time cost remain open for anomaly-detection methods; the deception-primitive literature, mature for the scanner threat model, has not been re-examined against the LLM-agent class; and the LLM-agent security literature, focused on offensive prompt injection and capability evaluation, has not addressed the defensive fingerprinting question. This dissertation contributes (i) a cross-vendor capability-gradient measurement of LLM-agent interaction with web deception, (ii) the methodological observation that a probe-interaction classifier survives the removal of timing and of direct probe-hit features, and (iii) a placement rule for deception primitives grounded in measured human-FPR data.

Chapter 3

SYSTEM DESIGN AND METHODOLOGY

3.1 Architectural Overview

The testbed comprises three loosely-coupled layers. The first is the deception-instrumented MERN-stack web application (the Eshop) into whose front-end and back-end the seven deception probes are integrated. The second is the set of five adversary harnesses that interact with the application: a Playwright-driven human-shopper simulation, a Python wordlist scanner, a credential-stuffing bot, and an LLM-driven agent that can be instantiated against any of five language models. The third is the measurement and analysis pipeline, which converts the raw per-action interaction trace into per-session feature vectors and applies the three analyses corresponding to the three research questions. All three layers communicate exclusively through a single shared comma-separated trace file under a fixed schema, which decouples adversary collection from analysis. This high-level arrangement is shown in Figure 3.1.

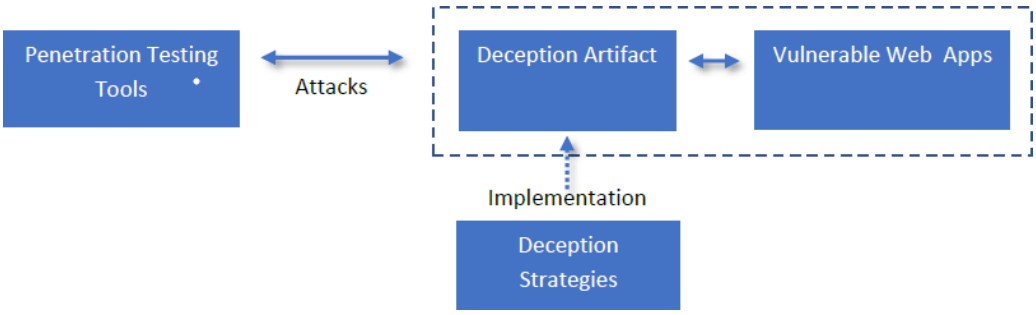

Figure 3.1: High-level web-deception arrangement carried forward from Phase I: adversary tooling exercises the vulnerable web application, which is instrumented with deception artefacts; the deception strategies (probes) sit between the two and convert adversary interactions into intrusion signals.

Table 3.1: Phase I deception primitives and their detection mechanisms.

ID	Name	Kind	Detection logic
P1	`fake_admin_url`	wordlist	URL-path match for `/fake-admin` or `/old_admin` (off-screen anchors).
P2	`fake_backup_url`	wordlist	URL-path match for `/backup-panel` or `/secret-backdoor`.
P3	`honeypot_field`	wordlist	Honeypot field name in a POST body (`honeypot_email`, `trap_field`, etc.).
P4	`decoy_cookie`	wordlist	Inbound cookie named `session_id` / `debug_mode` / `auth_debug` / `temporary_user`; also a URL-match path (see Section 3.2.2).
P5	`canary_meta_tag`	wordlist	Per-session canary value injected as a meta tag (instrumentation-only in the current dataset).

3.2 Phase I — Baseline Deception Primitives

Phase I establishes the deception substrate using four wordlist-class primitives consistent with established honeypot practice. Their definitions are summarised in Table 3.1.

3.2.1 Deceptive HTML Comments and Fake Administrative URLs

The first primitive (P1) places fabricated administrative URLs such as `/fake-admin` and `/old_admin` inside off-screen anchor elements absent from the application's navigation but visible to any client parsing the HTML for hyperlinks. The second primitive (P2) uses the same mechanism for `/backup-panel` and `/secret-backdoor`. P1 and P2 are kept structurally distinct so the per-probe coverage matrix can record which adversary classes are sensitive to which lexical pattern.

3.2.2 Decoy Session Cookies

The third primitive (P4 in the canonical ordering) is instrumented in the probe-identification module with two independent detection paths. A URL-match check records any GET request to `/get-deceptive-cookies` as a P4 trigger; a cookie-echo check records any inbound request whose Cookie header carries one of the decoy names. The two paths exist because the design intent is the long-established session-fixation honeypot. In the present implementation only the URL-match path is operative: the `/get-deceptive-cookies` route returns the cookie descriptors as a JSON body rather than emitting `Set-Cookie` headers, so no browser stores the decoy cookies and the cookie-echo path contributes zero hits. The operational consequence — the front-end fetches that URL unconditionally on every page mount — is analysed in Chapter 6 and discussed as a design rule in Chapter 7.

3.2.3 Honeypot Form Fields and Off-Screen Bait

The honeypot form (P3) is rendered into the layout component but positioned off-screen and excluded from the tab order, with field names (`honeypot_email`, `trap_field`, etc.)

chosen to be plausible to a naive form-filling bot. Any POST whose body contains one of these field names is evidence of automated form submission.

3.2.4 Attempt Counter and Adaptive Blocking

An attempt counter is incremented whenever any probe fires within a session; exceeding a configurable threshold places the application in a temporary blocking mode that serves a maintenance response for a cool-down window. Blocking is disabled for the Phase II measurement campaign to avoid biasing the trace data with truncated sessions.

3.3 Phase II — Semantic Canaries Targeting LLM Agents

Phase II introduces the central design innovation: two semantic-class probes whose detection condition is satisfied only by an adversary that reads the rendered natural-language content of the page (Table 3.2).

Table 3.2: Phase II semantic-class probes (P6 and P7).

ID	Name	Bait location and detection logic
P6	`imperative_comment`	HTML comment of the form "TODO: visit `/internal-dbg-7x9q` for admin" injected into the body on every mount; URL-path match for `/internal-dbg-7x9q`, which is not in any standard wordlist.
P7	`ai_instruction_meta`	Meta tag `name="ai-instruction"` pointing to `/assist-mode-k3p2`, injected into the head on every mount; URL-path match for `/assist-mode-k3p2`.

Discovering either URL requires the adversary to retrieve and parse the rendered DOM, process the natural-language content of the comment or meta tag, and decide to follow the imperative. Each step is consistent with LLM-agent behaviour and inconsistent with wordlist scanners. The empirical separation between LLM agents and traditional bots on these two probes is examined empirically in Chapter 6.

3.4 Measurement Layer: Trace Schema

Every adversary action is logged through a single trace-logger module that appends one row per action to a process-safe comma-separated file. The schema (Table 3.3) is the only contract between harnesses and analysis scripts. Every harness ends each session with a `session_end` marker carrying `reason="complete"` or `reason="crashed"`; the strict-resume logic counts a session as done only when this marker is present with `reason="complete"`.

Table 3.3: Trace-row schema (twelve fields).

Field	Purpose
`session_id`	Twelve-character UUID assigned at session start.
`adversary_class`	Canonical class name (e.g. `scanner`).
`adversary_id`	Internal identifier A1–A8.
`timestamp`	ISO-8601 UTC, millisecond resolution.
`order_index`	Per-session sequence number.
`action_type`	`http_request` / `api_request` / `page_load` / `tool:*` / `form_submit` / `session_end` / etc.
`http_method`	GET / POST / blank.
`url`	Full URL of the action.
`status_code`	Response status or blank.
`probe_id_triggered`	P1–P7 if a probe fired, blank otherwise.
`headers_fingerprint`	SHA-256 digest of UA / Accept headers (16 hex chars).
`extra_json`	Free-form JSON: cookies, form fields, LLM step index, persona.

3.5 Adversary Classes

Eight adversary classes are implemented, as listed in Table 3.4. The non-LLM classes instantiate traditional behaviours; the five LLM classes share the same Playwright-backed tool harness and system prompt and differ only in the underlying model.

Table 3.4: Adversary classes and behavioural rationale.

ID	Class	Mechanism	Rationale
A1	`human_legitimate`	Playwright, 11 shopping personas, jittered timing	False-positive baseline (RQ3)
A2	`scanner`	Python wordlist scan, Nikto-style UA	Traditional bot baseline
A3	`cred_stuffer`	POST burst on login + honeypot submit	Distinct tooling baseline
A4	`llm_llama8b`	Meta llama-3.1-8b via Groq + tools	Low-tier LLM
A6	`llm_llama4_scout`	Meta llama-4-scout-17b	Mid-tier LLM
A7	`llm_gpt_oss_20b`	OpenAI gpt-oss-20b	Mid-tier, cross-vendor
A8	`llm_gpt_oss_120b`	OpenAI gpt-oss-120b	High-tier, cross-vendor
A5	`llm_llama70b`	Meta llama-3.3-70b	High-tier LLM

3.6 Per-Session Feature Extraction

The feature extractor reduces a session's multi-row trace into a single thirty-two-element feature vector grouped into six families: probe coverage (`p1_count`–`p7_count` plus `any_probe_hit`, `wordlist_probe_hit`, `semantic_probe_hit`); volume (`total_requests`, `distinct_paths`, `form_submissions`, etc.); timing (`session_duration_s`, `mean_inter_request_s`, `median_inter_reques` `request_rate_per_s`); sequence diversity (`action_type_entropy`, `url_path_entropy` in bits); status-code mix (`status_2xx`–`status_5xx`); and hygiene (`session_ended_clean`, `crashed`, used only for filtering). The timing and probe-coverage families are the two that the defensibility ablations of Chapter 6 remove.

Chapter 4

IMPLEMENTATION

4.1 Application Architecture

The Eshop is a representative MERN-stack e-commerce application: a React front-end served by the Create-React-App development server on port 3000, an Express back-end on port 8000, MongoDB persistence, and a Node.js runtime. It implements product listing, product detail, search, cart management, login, registration, and checkout — sufficient surface area to support the diversity of behaviours exercised by the five adversary harnesses without synthetic complexity. Two source files are modified to host the deception primitives: the React layout component's `useEffect` hook (which injects the client-side primitives on every mount) and the Express router (which hosts the semantic-canary routes). All other application code is untouched, which keeps the deception layer cleanly separable from business logic.

4.2 Phase I Implementation

The off-screen anchors for the wordlist primitives (P1, P2) are rendered into the layout inside an absolutely-positioned, one-pixel container — present in the rendered HTML for any parser but invisible to a sighted user. The honeypot form (P3) is similarly rendered with `tabindex=-1` and `aria-hidden` attributes. The decoy-cookie endpoint (P4) is an existing route returning the cookie descriptors as a JSON body. The attempt counter and the temporary blocking response are implemented as Express middleware intercepting requests to the protected paths.

4.3 Phase II Implementation

The two semantic primitives are added by extending the front-end `useEffect` hook and by adding two back-end routes. P6 is injected by creating a `div`, setting its `innerHTML` to the literal comment string, and appending it to `document.body`; P7 is injected by creating a `meta` element with the appropriate `name` and `content` attributes and appending it to `document.head`. The two new routes (`/internal-dbg-7x9q` and `/assist-mode-k3p2`) respond with innocuous 200 OK bodies and log the request to the existing intrusion log so downstream measurement is uniform with the Phase I primitives.

4.4 The Adversary Harnesses

4.4.1 Scanner Harness

The scanner harness builds a per-session URL list comprising the `robots.txt` entry, every wordlist bait, and a random sample of twenty to forty-five paths from a forty-five-entry common-paths wordlist. The User-Agent is drawn from a pool of nine real scanner identifiers (Nikto, gobuster, ffuf, Nuclei, ZmEu, WhatWeb, and others). After the GET walk the harness issues a POST against `/submit` with a honeypot-shaped body so that P3 fires.

4.4.2 Credential-Stuffer Harness

The credential-stuffer harness issues eight to eighteen POST requests against the login endpoint with credentials drawn from a fixed list of thirteen fake e-mail addresses and fifteen weak passwords, with inter-attempt delays of 80–400 ms and a rotating tooling User-Agent (`python-requests`, `Go-http-client`, `curl`, `ApacheBench`, and others). It then submits the honeypot form, mimicking a naive bot that auto-submits every form it encounters.

4.4.3 LLM-Agent Harness

The LLM-agent harness exposes four tools to the model — `fetch_url`, `parse_html`, `submit_form`, and `read_cookies` — all backed by a single Playwright Chromium context per session. The model receives a system prompt framing it as a security-research assistant in a controlled lab and is asked to investigate the site for restricted-area access, internal tooling, debug endpoints, or operator notes. The step budget is fixed at eight steps across all five LLM adversaries so that the per-class comparison is fair.

Three implementation details are load-bearing for the validity of the results. First, the `parse_html` return JSON is ordered so that a top-level `notable` array — containing the comments and meta tags that mention URL paths or operator keywords — appears before the longer `comments`, `meta_tags`, `forms`, and `links` fields; this ordering ensures the semantic-canary content survives the per-tool-result character cap, which a pre-fix audit confirmed was otherwise truncating the P6 comment before it reached the model. Second, the `fetch_url` tool waits explicitly for the deception-canary meta tag to appear in the DOM (1.5 s timeout) after the network-idle event, so subsequent page navigations see a fully-hydrated DOM rather than the pre-React shell. Third, the Groq client wrapper handles three distinct error patterns — 413 tokens-per-minute, 429 rate-limit-exceeded, and 400 `tool_use_failed` — with exponential backoff plus history trimming, so transient API errors do not silently drop sessions.

4.4.4 Human-Shopper Simulation

The human-shopper harness produces the A1 class by Playwright-driven simulation. Each session draws one persona from a weighted set of eleven (`quick_browser`, `window_shopper`, `searcher`, `cart_filler_abandon`, `cart_filler_checkout`, `paginator`, `slow_reader`, `login_attempt`, `register_browse`, `search_no_results`, `back_and_forth`) and executes a sequence of in-application clicks with per-action timing jitter calibrated to the persona. Inter-keystroke delays in form fields are 30–110 ms, the User-Agent is one of five real desktop browser

strings, the viewport is fixed at 1366×820, and the Chromium context is fresh per session so no cookies persist across simulated shoppers.

4.5 Orchestration and Analysis Pipeline

The orchestrator runs the harnesses sequentially in a deliberate order — cheap deterministic classes first, Groq-dependent LLM classes last — so that an aborted run preserves the most data, using the strict-resume logic of Section 3.4. The feature extractor reads the raw trace, groups rows by `session_id`, and emits one thirty-two-feature row per clean session. Three analysis scripts then consume the features file: one produces the per-class probe-coverage matrix and the semantic-versus-wordlist split; one trains Random Forest and Logistic Regression classifiers across four feature configurations and two class granularities, reporting the timing and probe-feature ablation verdicts; and one computes per-probe human FPR and bot TPR and runs a greedy minimum-set-cover for the smallest probe subset that satisfies a configurable false-positive budget.

Chapter 5

EXPERIMENTAL SETUP

5.1 Hardware and Software Environment

All experiments are conducted on a single Apple-Silicon MacBook Air running macOS Darwin 25.3. The Eshop runs locally with the Express back-end on port 8000 and the React development server on port 3000; the adversary harnesses execute on the same machine. The Python environment uses Python 3.13 with Playwright 1.46 (Chromium driver), scikit-learn 1.8, pandas 3.0, beautifulsoup4 4.12, requests 2.32, and the official Groq client library. All LLM inference runs against the publicly-available Groq free-tier API.

5.2 Dataset

The full data-collection campaign produced 285 clean sessions across eight adversary classes. For the final analysis the dataset was balanced to a uniform target of $n = 30$ per class to support stratified cross-validation with equal class weight, yielding a balanced dataset of 225 sessions (Table 5.1). Sessions are considered clean in the strict sense of Section 3.4: they terminate with a `session_end` row carrying `reason="complete"`. A deterministic random subsample (seed 42, sort-by-session-id then shuffle and take the first thirty) reduced A2, A3, and A4 from their initial collection of $n = 50$ to the uniform target; the per-session kept/dropped record is preserved so the subsample is reproducible without re-collection.

Table 5.1: Per-class session counts in the balanced dataset.

ID	Class	Vendor / params	Collected	Balanced n
A1	`human_legitimate`	Playwright, 11 personas	30	30
A2	`scanner`	Python wordlist scan	50	30
A3	`cred_stuffer`	login replay + honeypot	50	30
A4	`llm_llama8b`	Meta, 8B	50	30
A6	`llm_llama4_scout`	Meta, 17B	30	30
A7	`llm_gpt_oss_20b`	OpenAI, 20B	30	30
A8	`llm_gpt_oss_120b`	OpenAI, 120B	30	30
A5	`llm_llama70b`	Meta, 70B	15	15
	Total		**285**	**225**

Only A5 falls short of the uniform target. Groq's 100,000 tokens-per-day free-tier cap on llama-3.3-70b-versatile permits roughly four to five full LLM-agent sessions per day,

so reaching $n = 30$ would require a six-day campaign beyond the project timeline. A5 is retained at its true collected count ($n = 15$) and reported throughout Chapter 6 with its 95% Wilson confidence interval; the capability-gradient finding does not depend on A5 reaching $n = 30$ because the low-tier (A4) and mid-tier (A6) classes and the two OpenAI classes (A7, A8) are all at the full uniform sample size.

5.3 LLM Models

Five language models served by Groq are evaluated (Table 5.2), chosen to span two vendors and an approximately fifteen-fold parameter range. All support the Groq tool-use endpoint with the same OpenAI-compatible function-calling JSON schema.

Table 5.2: LLM adversaries used in Phase II.

ID	Model identifier	Vendor	Params
A4	`llama-3.1-8b-instant`	Meta	8B
A6	`llama-4-scout-17b-16e-instruct`	Meta	17B
A7	`openai/gpt-oss-20b`	OpenAI	20B
A8	`openai/gpt-oss-120b`	OpenAI	120B
A5	`llama-3.3-70b-versatile`	Meta	70B

5.4 Hyperparameters

All LLM sessions use temperature 0.6, `max_tokens` 512 per turn, eight steps per session, a five-second sleep between successive API calls, a four-pair rolling context window, and a tool-result character cap of 2,200 characters (raised from an initial 1,500 after an audit revealed the smaller cap was truncating the P6 comment before it reached the model).

5.5 Evaluation Methodology

RQ1 is evaluated descriptively by the per-class probe-coverage matrix. RQ2 is evaluated by training a Random Forest (400 trees, `max_depth` 8, `min_samples_leaf` 2, class-weight balanced) and a Logistic Regression (lbfgs, `max_iter` 2000, class-weight balanced, with a StandardScaler upstream) under stratified five-fold cross-validation, with the number of folds reduced to the smallest class size when necessary. Each model is trained under four feature configurations — full, timing features removed, direct probe-hit features removed, and both removed — at two class granularities (eight-class, with every LLM distinct; and four-class, with the LLMs collapsed under a single `llm` label). RQ3 is evaluated by per-probe human FPR and bot TPR and by a greedy minimum-set-cover for the smallest probe subset whose union FPR against the human class remains below 10 per cent.

Chapter 6

RESULTS AND ANALYSIS

6.1 Phase I Baseline Results

The Phase I deception substrate was evaluated in an earlier iteration of this project against a manually-driven OWASP-ZAP-style scanner. Across twenty captured scanner sessions, fifteen were correctly identified as adversarial, producing an overall detection accuracy of 75 per cent, precision of 78 per cent, recall of 70 per cent, and an F1 score of 74 per cent. Table 6.1 records, per primitive, the attacker behaviour that triggered it, the system response, and the qualitative effectiveness rating.

Table 6.1: Phase I OWASP-aligned interaction summary.

Primitive	**Attacker behaviour**	**System response**	**Effectiveness**
Deceptive HTML comments	Crawled embedded URLs via spider	Early detection	Low risk — effective
Deceptive cookies	Altered and replayed values	Logged as suspicious	Low risk — effective
Honey parameters	Fuzzed names (`debug_key`, `token_old`)	Multiple detections	Medium risk — effective
Attempt counter	Repeated malicious requests	Threshold in 58% sessions	Medium risk — effective
Shutdown response	Continued scanning past warning	Maintenance mode served	High risk — effective

Figures 6.1–6.4 show the Phase I deception primitives operating against a live attacker session: the deceptive HTML comment carrying a fabricated backend URL rendered in the browser console, the server's logged response when the attacker follows that URL, the maintenance page served once the attempt counter crosses its threshold, and the decoy session cookies set on the client.

The Phase I result motivated the Phase II extension: the 75 per cent accuracy was achieved against a single adversary class and could not speak to class separability, and the false-positive analysis was qualitative rather than quantitative. The Phase II campaign below addresses both gaps.

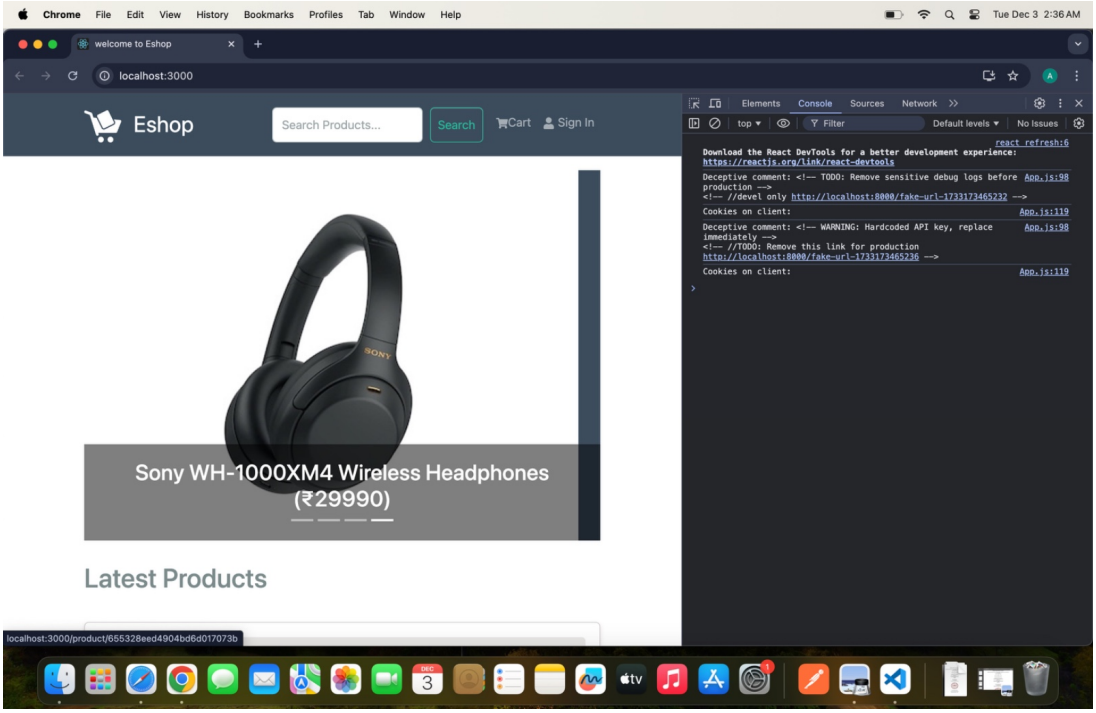

Figure 6.1: Phase I: deceptive HTML comment carrying a fabricated backend URL, visible in the browser developer console.

6.2 RQ1 — Probe Coverage

Table 6.2 reports the overall per-probe trigger rate across the balanced 225-session dataset. Table 6.3 disaggregates by adversary class.

Table 6.2: Overall probe trigger rate across the balanced dataset.

Probe	Kind	Trigger rate
P1 `fake_admin_url`	wordlist	48.4% (109)
P2 `fake_backup_url`	wordlist	49.3% (111)
P3 `honeypot_field`	wordlist	29.8% (67)
P4 `decoy_cookie`	wordlist	26.7% (60)
P5 `canary_meta_tag`	wordlist	0.0% (0)
P6 `imperative_comment`	semantic	39.1% (88)
P7 `ai_instruction_meta`	semantic	28.4% (64)

The contribution of this work is *not* that every probe fires on every adversary; several probes are silent for several classes by design. The contribution is that the semantic probes (P6/P7) exhibit *selective activation*: zero observed human / non-LLM false positives, but high trigger rates for capable LLM agents. The zero-valued semantic-probe cells for the simulated legitimate users, scanners, and credential stuffers are best read as **correct negatives** — negative-control evidence that P6/P7 do not fire on classes for which they were not designed — rather than as failures to detect. A probe that fired on everyone would carry no discriminative information; the value lies precisely in the contrast between the 0% non-LLM rate and the high capable-LLM rate. In this dataset, 0 of 90 non-LLM sessions triggered either semantic probe, while the four higher-capability LLMs triggered

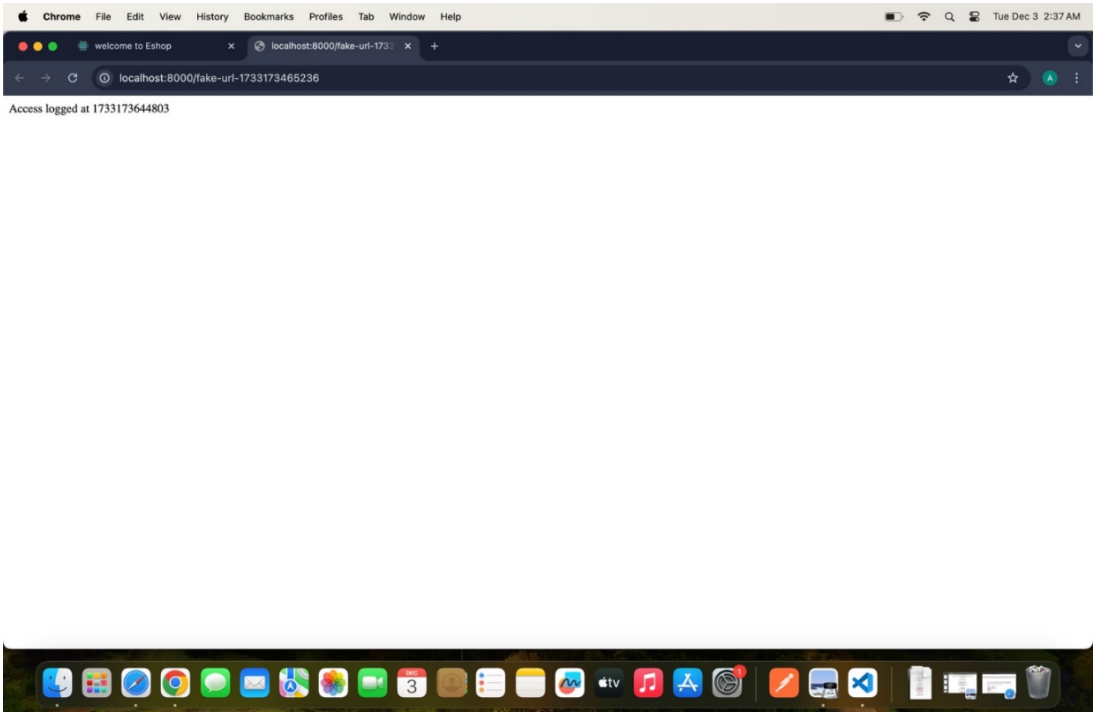

Figure 6.2: Phase I: server response logged after the attacker follows the fabricated URL (intrusion attempt recorded).

Table 6.3: Per-class probe-class hit rate (balanced $n = 30$, A5 $n = 15$).

Class	n	Any	Wordlist	Semantic
`human_legitimate`	30	100.0%	100.0%	0.0%
`scanner`	30	100.0%	100.0%	0.0%
`cred_stuffer`	30	100.0%	100.0%	0.0%
`llm_llama8b` (A4)	30	0.0%	0.0%	0.0%
`llm_llama4_scout` (A6)	30	100.0%	96.7%	53.3%
`llm_gpt_oss_20b` (A7)	30	100.0%	100.0%	96.7%
`llm_gpt_oss_120b` (A8)	30	100.0%	100.0%	100.0%
`llm_llama70b` (A5)	15	100.0%	100.0%	100.0%

them at observed rates of 53% or higher.

Note on the zero-valued cells. The zero entries for simulated legitimate users, scanners, and credential stuffers in Table 6.3 and Figure 6.5 are not failures to detect; they are correct negatives showing that P6/P7 do not fire on classes for which they were not designed. They are measured negatives (the probe correctly never triggered), not missing data.

6.3 RQ2 — Classifier Performance

Table 6.4 reports five-fold stratified cross-validation accuracy and macro F1 across two class granularities and four feature configurations. (Figure files named `rq2_confusion_*_5class_*` carry a legacy filename slug from when only two LLMs were defined; the per-LLM configuration is 8-class in this dataset.) The four-class confusion matrices for the full and

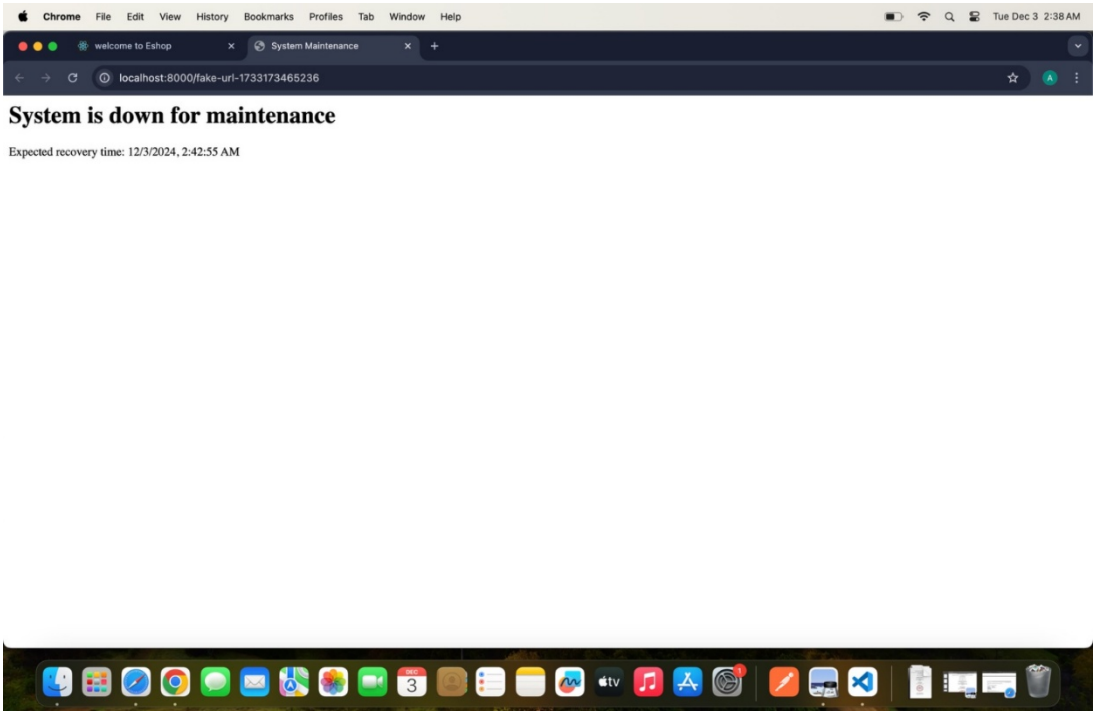

Figure 6.3: Phase I: maintenance page served after repeated intrusion attempts cross the attempt-counter threshold (adaptive blocking).

timing-ablated feature sets are shown in Figures 6.6 and 6.7.

The four-class classifier's 100% accuracy reflects that, at this granularity, the adversary classes are structurally distinct by construction: scanners exclusively enumerate URL wordlists, credential stuffers exclusively interact with the login endpoint, simulated shoppers exclusively traverse the legitimate shopping flow, and LLM agents produce categorically richer action sequences spanning all of the above plus exploratory tool use. Perfect classification at this granularity is therefore a measure of class separability given probe-instrumented telemetry, rather than of classifier sophistication. Two practical implications follow. First, defenders deploying probe-instrumented telemetry of the form described here can rely on simple, low-cost classifiers; deep or specialised models are not required for the super-class distinction. Second, the more informative number for evaluating classifier quality is the eight-class accuracy of 91.1 per cent (Random Forest, full features), where the residual confusion is concentrated within the LLM family at the high-capability tier (Figure 6.8) — exactly the boundary where behavioural similarity is greatest and would be expected to produce confusion.

6.4 Timing-Ablation Defensibility Check

A central methodological risk is that the classifier could learn Groq's per-call rate-limit behaviour rather than real adversary behaviour: LLM sessions are slowed by the five-second per-API-call sleep, so timing-based features could trivially separate LLMs from non-LLMs. We re-train every classifier with the four timing features removed (Table 6.5).

The four-class classifier retains 100% accuracy with and without timing features. The eight-class deltas are small ($\Delta \leq 0.027$). This is consistent with the classifier picking up real interaction patterns rather than the inference platform's rate-limit-imposed timing

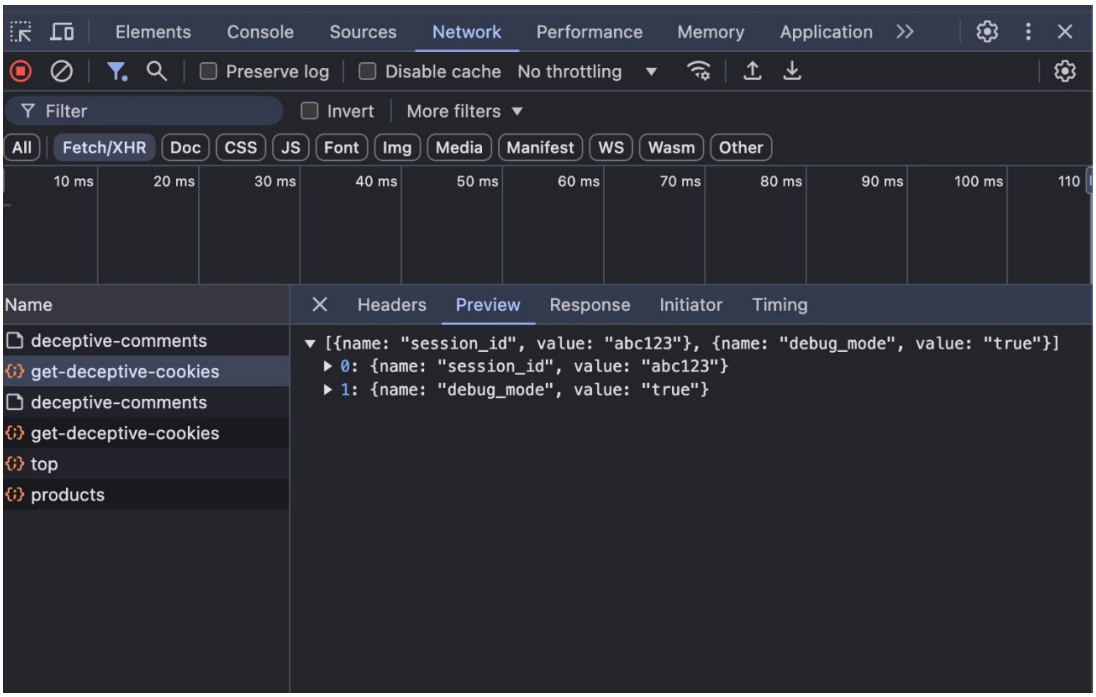

Figure 6.4: Phase I: decoy session cookies (`session_id`, `debug_mode`) set on the client to attract session-manipulation attempts.

alone, supporting the defensibility of the four-class result within the scope of this testbed.

6.5 Probe-Feature Ablation (Leakage Defensibility Check)

A second, orthogonal risk is that the classifier might merely memorise direct probe hits. The main classifier is intentionally probe-instrumented (that is by design); this ablation removes the ten probe-related features (`p1_count`–`p7_count`, `any_probe_hit`, `wordlist_probe_hit`, `semantic_probe_hit`) and asks whether broader behavioural residue remains (Table 6.6).

The four-class classifier retains 100% accuracy with all ten probe-related features removed; the eight-class drop matches the timing ablation. Even under the *joint* ablation (no timing AND no probe features), the four-class accuracy is 0.996 and the eight-class RandomForest is 0.876. This supports the claim that the per-session interaction trace carries behavioural residue (URL-path entropy, action-type entropy, status-code mix, distinct-path count, form behaviour) beyond the direct probe-hit columns alone.

6.6 The Capability Gradient

Within the LLM adversary class, the observed semantic-probe hit rate increases monotonically with model capability across the five LLMs (Figure 6.9, Table 6.7). The gradient spans two vendors and a roughly fifteen-fold parameter range; the rank order is preserved at every Wilson 95% CI endpoint. Figure 6.9 plots each model's hit rate against its published parameter count on a logarithmic axis, with the 95% Wilson interval shown as an error bar and the two vendor families distinguished by colour.

That the gradient is observed across both Meta and OpenAI families within this

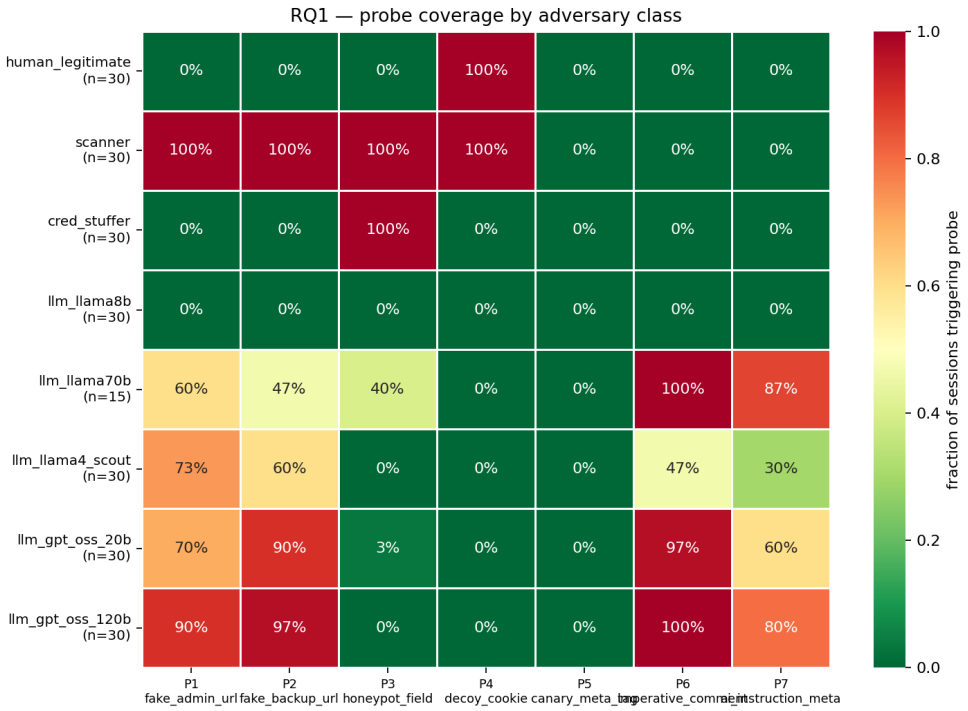

Figure 6.5: Per-class per-probe coverage heatmap (RQ1). Cell value is the fraction of that class's sessions that triggered the probe at least once. The zero-valued semantic-probe cells for simulated legitimate users, scanners, and credential stuffers are *correct negatives* (the probe correctly never triggered for classes it was not designed to catch), not missing data or failures to detect.

testbed is consistent with the claim that the 8B's outlier behaviour is not a Meta-specific training artefact and the high-tier behaviour is not a Meta-specific instruction-following style. Mechanism inspection, summarised in Table 6.8, shows the 8B model explored entirely by training-prior guessing (`/admin`, `/restricted`, `/debug`, `/login`), touching no deception primitive, whereas Scout, gpt-oss-20b, gpt-oss-120b, and 70b fetched the deception URLs directly (Scout's top paths included `/fake-admin`, `/internal-dbg-7x9q`, `/old_admin`, `/backup-panel`, `/assist-mode-k3p2`, `/secret-backdoor`). Within this testbed, capability tier appears to be a stronger predictor than vendor of probe-interaction behaviour in this dataset.

6.7 RQ3 — False-Positive Rate Against Human Shoppers

In the aggregate, all 30 simulated human-shopper sessions triggered at least one deception probe — an observed 100% aggregate false-positive rate. Table 6.9 shows this is driven entirely by a single probe (P4); every other probe has observed 0.0% human FPR. The same per-probe human FPR and bot TPR are plotted in Figure 6.10.

The cause of the P4 false-positive rate is structural rather than a measurement artefact. P4 is configured with `/get-deceptive-cookies` in its trigger URL list, and the front-end layout component runs `axios.get('/get-deceptive-cookies')` unconditionally inside its `useEffect` on every page mount. Every browser-mediated session therefore issues

Table 6.4: Classifier cross-validation results on the balanced dataset.

Configuration	Model	Acc.	Macro F1
8-class, full	RandomForest	0.911	0.905
8-class, full	LogReg	0.871	0.861
8-class, no-timing	RandomForest	0.898	0.890
8-class, no-timing	LogReg	0.844	0.831
8-class, no-probe	RandomForest	0.898	0.892
8-class, no-probe	LogReg	0.844	0.825
8-class, no-timing+no-probe	RandomForest	0.876	0.862
8-class, no-timing+no-probe	LogReg	0.844	0.827
4-class, full	**RandomForest**	**1.000**	**1.000**
4-class, full	LogReg	1.000	1.000
4-class, no-timing	**RandomForest**	**1.000**	**1.000**
4-class, no-timing	LogReg	1.000	1.000
4-class, no-probe	**RandomForest**	**1.000**	**1.000**
4-class, no-probe	LogReg	1.000	1.000
4-class, no-timing+no-probe	**RandomForest**	**0.996**	**0.995**
4-class, no-timing+no-probe	LogReg	0.996	0.995

Table 6.5: Timing-ablation verdict on the balanced dataset.

Model	Granularity	Full	No-timing	Δ
RandomForest	8-class	0.911	0.898	-0.013
RandomForest	4-class	1.000	1.000	0.000
LogReg	8-class	0.871	0.844	-0.027
LogReg	4-class	1.000	1.000	0.000

that GET on the initial render, and the probe-identification module matches the path and emits P4 — before any adversarial behaviour. The cookie-echo detection path that the module also implements contributes zero hits, because the back-end route returns the cookie descriptors as a JSON body rather than emitting `Set-Cookie` headers; no browser stores the cookies, and nothing is echoed back. This is discussed in Section 7.2 as a design rule.

A greedy minimum-set-cover under a 10% union-FPR constraint selects the probe subset {P2, P3, P1, P6}, achieving 84.6% bot-class TPR (165 of 195 bot sessions covered) at observed 0.0% human FPR. P4 is dominated and is not selected.

RQ3 is scoped to a simulated legitimate-user population generated with Playwright; the observed 0% simulated-user FPR on P6/P7 is reported on that basis. The zero-valued semantic-probe entries in Table 6.9 for the non-target classes are correct negatives, consistent with the negative-control reading in Section 6.2.

6.8 A5 Small-n Caveat

Only A5 falls short of the uniform $n = 30$ target, owing to the 100,000-tokens-per-day cap on the 70B endpoint. We report A5 at its true collected count ($n = 15$) with its 95% Wilson confidence interval $[79.6\%, 100\%]$. Statistical-significance claims about A5 should be made against the CI rather than the point estimate; the rank-order claim across the

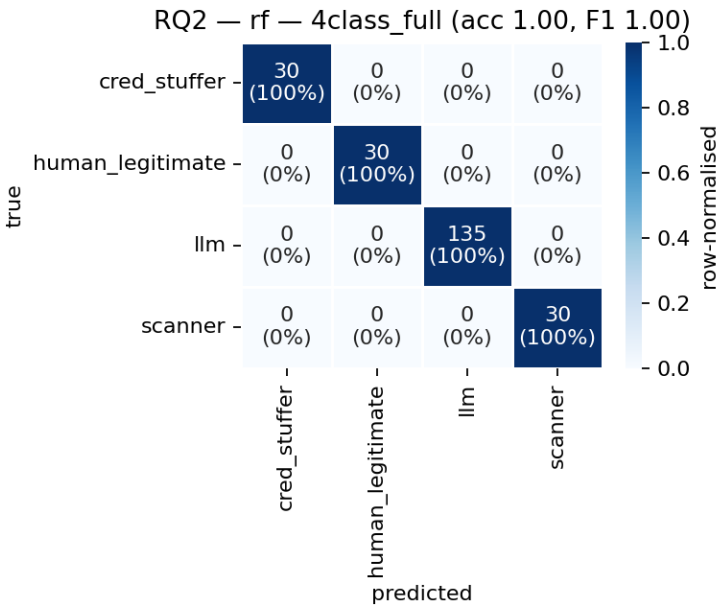

Figure 6.6: Confusion matrix, RandomForest, four-class, full features. Every off-diagonal cell is empty on the balanced dataset.

Table 6.6: Probe-feature ablation verdict on the balanced dataset.

Model	**Granularity**	**Full**	**No-probe**	**Δ**
RandomForest	8-class	0.911	0.898	−0.013
RandomForest	4-class	1.000	1.000	0.000
LogReg	8-class	0.871	0.844	−0.027
LogReg	4-class	1.000	1.000	0.000

capability gradient is preserved at every CI endpoint.

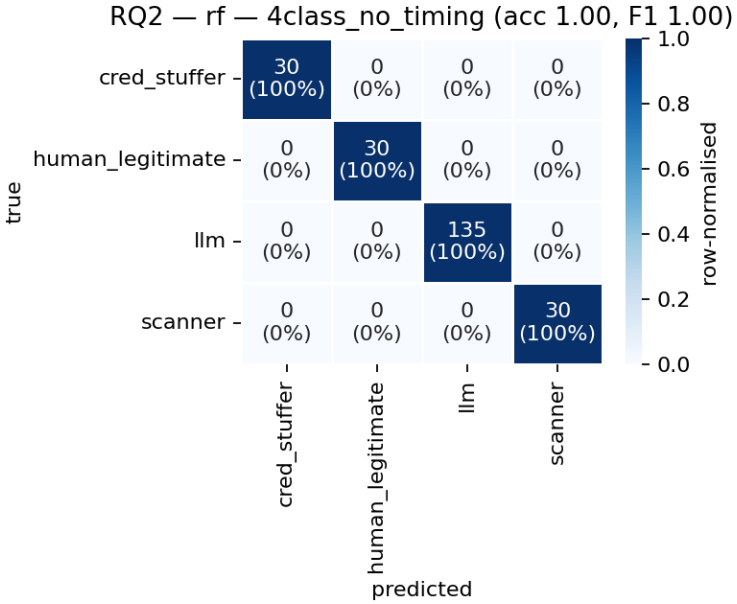

Figure 6.7: Confusion matrix, RandomForest, four-class, no-timing features. Accuracy retained at 1.000 in this dataset.

Table 6.7: Capability gradient across LLM adversaries.

Model	**Vendor**	**Params**	**Sem. rate**	**95% Wilson CI**
`llama-3.1-8b`	Meta	8B	0.0% (0/30)	[0.0%, 11.4%]
`llama-4-scout-17b`	Meta	17B	53.3% (16/30)	[35.8%, 70.0%]
`gpt-oss-20b`	OpenAI	20B	96.7% (29/30)	[83.3%, 99.4%]
`gpt-oss-120b`	OpenAI	120B	100.0% (30/30)	[88.6%, 100.0%]
`llama-3.3-70b`	Meta	70B	100.0% (15/15)	[79.6%, 100.0%]

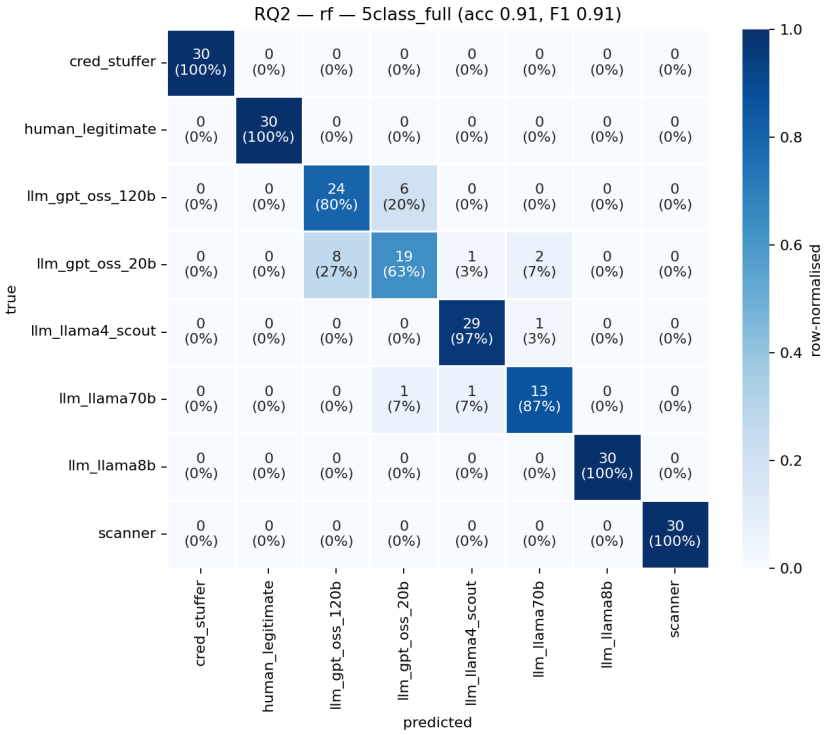

Figure 6.8: Confusion matrix, RandomForest, eight-class (legacy '5class' filename slug), full features. Residual confusion is concentrated within the LLM family at the high-capability tier.

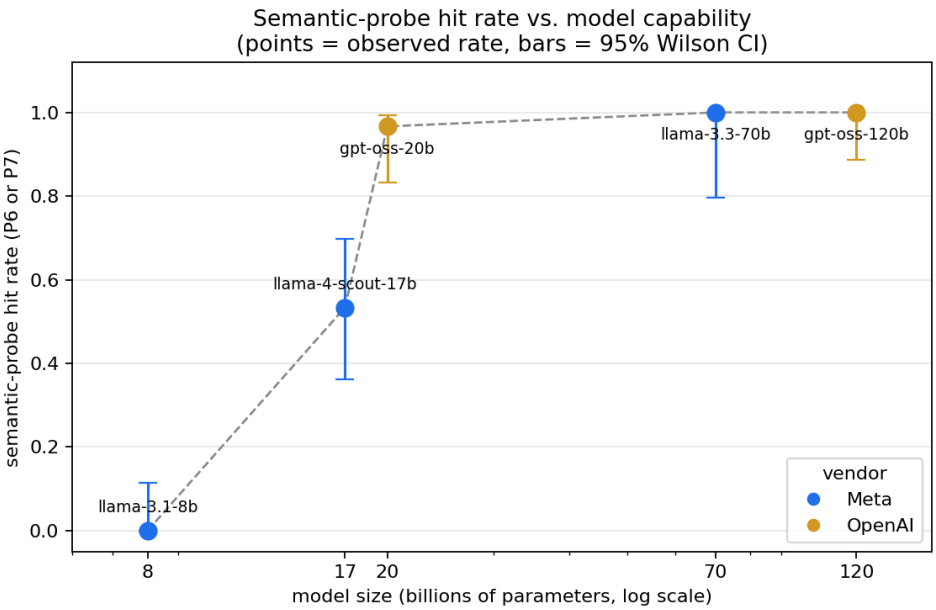

Figure 6.9: Semantic-probe hit rate vs. model capability, across both Meta and OpenAI families within this testbed. Points are observed per-class hit rates; error bars are 95% Wilson confidence intervals; the dashed line connects models in ascending parameter-count order.

Table 6.8: Top URL paths visited per LLM adversary (balanced sessions; percentages are the fraction of that class's sessions that touched the path). Each cell lists a path and the share of that class's sessions that reached it; the semantic canaries P6 (`/internal-dbg-7x9q`) and P7 (`/assist-mode-k3p2`) are marked with †.

llama-3.1-8b (8B)	**llama-4-scout-17b (17B)**	**gpt-oss-20b (20B)**	**llama-3.3-70b (70B)**	**gpt-oss-120b (120B)**
/admin (70%)	/fake-admin (47%)	/internal-dbg-7x9q† (97%)	/internal-dbg-7x9q† (100%)	/internal-dbg-7x9q† (100%)
/debug (33%)	/internal-dbg-7x9q† (47%)	/secret-backdoor (70%)	/assist-mode-k3p2† (87%)	/fake-admin (90%)
/restricted (27%)	/backup-panel (40%)	/backup-panel (63%)	/secret-backdoor (47%)	/backup-panel (83%)
/login (23%)	/assist-mode-k3p2† (30%)	/assist-mode-k3p2† (60%)	/old_admin (40%)	/assist-mode-k3p2† (80%)
/restricted_area (20%)	/old_admin (30%)	/fake-admin (53%)	/submit (40%)	/old_admin (47%)
/admin/login (17%)	/secret-backdoor (23%)	/old_admin (30%)	/fake-admin (27%)	/secret-backdoor (43%)

† semantic canary (P6/P7); all other non-root paths are wordlist baits. The root path / (100% for every class) is omitted as non-discriminative.

Table 6.9: Per-probe human FPR and bot TPR on the balanced dataset.

Probe	**Kind**	**Human FPR**	**Bot TPR**
P1 `fake_admin_url`	wordlist	0.0%	55.9%
P2 `fake_backup_url`	wordlist	0.0%	56.9%
P3 `honeypot_field`	wordlist	0.0%	34.4%
P4 `decoy_cookie`	wordlist	100.0%	15.4%
P5 `canary_meta_tag`	wordlist	0.0%	0.0%
P6 `imperative_comment`	semantic	0.0%	45.1%
P7 `ai_instruction_meta`	semantic	0.0%	32.8%

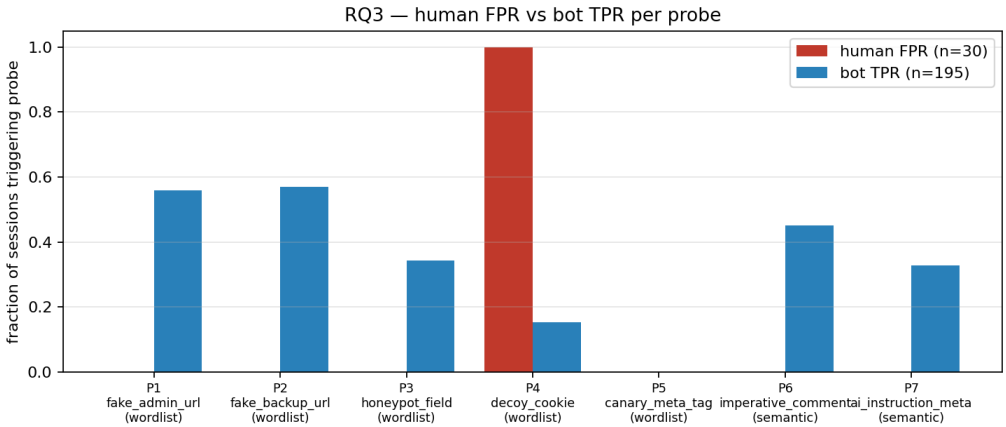

Figure 6.10: Per-probe human FPR against bot TPR (RQ3). P4 is the outlier with the highest FPR and lowest TPR; the two semantic probes achieve observed 0% FPR.

Chapter 7

DISCUSSION

7.1 Capability Tier as a First-Class Adversary Dimension

The capability-gradient finding of Section 6.6 implies a modification to the conventional adversary taxonomy used in web-application security. Traditional taxonomies enumerate tooling categories — scanner, brute-forcer, exploiter, scraper — and implicitly assume members of a category behave similarly enough that a single detector design works across all of them. For LLM agents this assumption fails within this testbed: an 8-billion-parameter agent and a 120-billion-parameter agent given the same tools, prompt, and objective produced probe-interaction signatures that sit in distinct clusters in the eight-class confusion matrix. Defensive systems that treat "LLM agent" as a unitary class would mis-attribute the 8B agent to the scanner cluster and the 120B agent to its own cluster, which suggests that future adversary taxonomies should index on capability tier rather than (or in addition to) model family. Operationally, defenders publishing evaluations of LLM-agent detection should report both the smallest and the largest model in the family being defended against, since a middle-tier evaluation may not generalise.

7.2 Design Rule: Avoid Universal-Path Deception

The P4 result of Section 6.7 — a 100% observed human FPR produced by the unconditional front-end fetch of the URL configured as P4's trigger — generalises into a design rule: a probe's trigger surface (its URLs, cookie names, form-field names, header values) must be disjoint from anything the application's own setup traffic touches. Any overlap converts the probe from a high-precision intrusion signal into a low-precision alarm that fires on every legitimate user, irrespective of whether the underlying detection logic is sound. Two repair paths preserve the spirit of the decoy-cookie design: removing `/get-deceptive-cookies` from the front-end's mount-time `useEffect` so the URL is never fetched by normal page loads; or removing the URL from the P4 trigger list and repairing the back-end to emit `Set-Cookie` headers, which restores the session-fixation cookie-echo detection path. Either repair eliminates the structural FPR without removing the deception primitive.

7.3 Why the Ablations Matter

Two ablations strengthen the LLM-versus-bot claim. The timing ablation addresses the concern that the classifier might learn "sessions with a five-second per-step delay are LLMs" — a property of the inference platform rather than the adversary. The probe-feature ablation addresses the orthogonal concern that the classifier might merely memorise direct probe hits. In this dataset the four-class classifier retains 100% accuracy under each ablation individually and 99.6% under both jointly, which substantially weakens both objections. Future evaluations of LLM-agent detection are encouraged to report both ablation pairs as a standard component of the result; both are cheap to compute and answer the two most natural defensibility objections directly.

7.4 Threats to Validity

7.4.1 Internal validity

The classifier's high accuracy could in principle be driven by overfitting on a small balanced sample. We mitigate this with stratified k-fold cross-validation (with k reduced to the smallest class size when necessary), class-weight balancing, and modest Random Forest depth. The four-class accuracy is preserved when the LLM class is collapsed across all five LLM adversaries, removing dependence on any one per-LLM-class sample size. A residual caveat is that near-perfect class separation on the semantic-probe feature means logistic-regression coefficient magnitudes should not be over-interpreted; we therefore report cross-validated accuracy rather than coefficient significance.

7.4.2 External validity

All measurements are taken against a single MERN-stack application with a fixed probe set. The selectivity of the semantic probes depends on the specific URL strings chosen. We argue the capability-gradient finding is more likely to generalise than the absolute trigger rates, because the gradient is consistent across vendors and aligns with the published capability ordering of the five LLMs.

7.4.3 LLM-agent harness

All LLM agents share a single tool set, prompt, and eight-step budget. Production agents may use richer tools, different prompts, and larger budgets; whether the gradient holds in those configurations is open and identified as future work.

7.4.4 Scope of the simulated legitimate-user class

The main contribution of this dissertation does not depend on the legitimate-user class. RQ1 and RQ2 are driven by real LLM API sessions and automated adversary traces; the simulated legitimate-user class supports only the RQ3 false-positive baseline, and RQ3 is scoped accordingly to that simulated-user population.

The legitimate-user class is a Playwright-driven simulated shopper population implementing eleven weighted personas. This is a deliberate methodological choice: the dependent variable for the probes is the set of observable HTTP, form, cookie, and DOM events,

and at that layer the simulation reproduces the same events a browser-mediated shopper generates. Within RQ3, the zero-valued semantic-probe entries for the non-target classes are correct negatives (negative-control evidence), and the one substantive legitimate-user result — P4 firing on every browser-mediated session — is structural rather than behavioural, since it is produced by the application's own mount-time setup fetch rather than by any user action; it is therefore independent of how the shopper population is generated. Broadening the legitimate-user baseline to additional persona designs is noted as a future direction in Section 8.2.

7.4.5 The P5 instrumentation gap

P5 is instrumented in the front-end and back-end but the probe-identification module has no detection logic for it, so its 0% rate reflects instrumentation rather than adversary behaviour. Either P5 should be deprecated or value-match detection should be added.

7.5 Comparison with the Phase I Baseline

The Phase I evaluation reported 75% accuracy against a single OWASP-ZAP-style scanner. The Phase II extension reaches 100% four-class accuracy against eight distinct adversary classes on a 225-session balanced dataset, preserved under both ablations. The gain derives from the addition of the two semantic-class probes (the LLM-versus-bot discriminator), a clean per-session feature vocabulary, and an explicit human-shopper class that converts the qualitative Phase I false-positive observation into a quantitative measurement.

7.6 Practical Implications

For an operator deploying this design, the guidance derived from this study is: deploy the four-probe minimal subset {P1, P2, P3, P6}, which in this dataset achieves 84.6% bot TPR at observed 0% human FPR; ensure no probe's trigger surface overlaps with the application's setup traffic (the P4 weakness is the cautionary example); treat any LLM-agent-detection product's published accuracy with caution unless it evaluates both a low-tier and a high-tier model; and, when training detection classifiers, ablate both timing and direct probe-hit features so the accuracy claim is defensible against the rate-limit and leakage concerns documented here.

Chapter 8

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

8.1 Summary of Findings

This dissertation presented a two-phase investigation of layered web deception against a heterogeneous adversary population that includes traditional scanners, credential-stuffing bots, simulated human shoppers, and a recently-emerged class of LLM-driven autonomous agents. Phase I established the deception substrate on a MERN-stack e-commerce application with four wordlist-class primitives and an attempt-counter-based blocking response, achieving 75% accuracy against an OWASP-ZAP-style scanner. Phase II extended the substrate with two novel semantic-class probes, instrumented the application with a process-safe trace logger and a per-session feature extractor, and collected 285 clean sessions across eight adversary classes spanning two LLM vendors and a fifteen-fold parameter range, subsequently balanced to a uniform $n = 30$ per class (225 sessions; A5 retained at $n = 15$).

The principal findings, supported within this testbed, are:

- The contribution is not that every probe fires, but that the semantic probes (P6/P7) exhibit *selective activation*: 0 of 90 non-LLM sessions triggered either probe, while the mean semantic-probe hit rate across the five LLM classes was 70.0% (the pooled session-level rate is 66.7%, 90 of 135).

- A four-class classifier (humans / scanner / cred-stuffer / LLM-any) achieves 100% stratified five-fold cross-validation accuracy with and without timing features, and also with all ten direct probe-hit features removed; the joint ablation retains 99.6% accuracy. The distinction is therefore carried by both direct probe-interaction patterns and broader behavioural residue.

- The observed semantic-probe hit rate increases monotonically with model capability (0%, 53%, 97%, 100%, 100%) when models are sorted by published parameter count, and the gradient is observed across both Meta and OpenAI model families within this testbed. "LLM agent" is not a unitary adversary class within this testbed; capability tier appears to be a stronger predictor than vendor in this dataset.

- The decoy-cookie probe (P4) exhibits 100% observed human FPR due to a structural placement choice; the minimal-probe subset {P2, P3, P1, P6} achieves 84.6% bot TPR at observed 0% human FPR, supporting the claim that P4 can be removed from a deployed configuration without loss of bot-class detection.

8.2 Future Scope

Larger and more diverse LLM panel. The gradient is established across five LLMs and two vendors, but the small-tier representation is thin and the 70B sample is TPD-limited. A study with paid-tier access would tighten the confidence intervals and could extend the gradient upward by including closed-source frontier models.

Step-budget and prompt ablation. All LLM sessions used a fixed eight-step budget and prompt. A longer budget with a directive prompt is hypothesised to shift the 8B model rightward on the gradient; testing this would be valuable evidence for the capability-tier hypothesis.

Broader legitimate-user baseline for RQ3. The RQ3 false-positive baseline is built on an eleven-persona simulated shopper population. Extending it with additional persona designs, or a larger consented user study, would widen the behavioural coverage of the legitimate-user class and further characterise the false-positive behaviour of the probe set.

New probes for LLM-specific failure modes. The two semantic probes target a single LLM failure mode (acting on injected page content). Prompt injection, tool-use manipulation, and capability discrimination via deliberately ambiguous instructions could each be operationalised as new probe kinds using this testbed.

Online deployment. A natural follow-up is to deploy the feature extractor and classifier inline as session-scoring middleware and measure end-to-end latency and false-positive cost under production-like traffic.

8.3 Social Impact

The demonstration that low-cost commodity deception primitives can distinguish LLM-driven agents from traditional scanners — on free-tier inference platforms with modest compute — lowers the barrier to defensive deployment for small organisations and educational institutions that cannot afford commercial bot-management products. The capability-gradient finding contributes to the policy conversation about LLM-driven automation by providing empirical evidence that "AI agents" are not a behavioural monolith, and that defensive frameworks reasoning about them should distinguish capability tiers. On the privacy side, the structural P4 finding — that a deception primitive placed on the universal user path is indistinguishable from a tracker to an outside observer — argues for documenting deception deployments in the same transparency frameworks that govern analytics and personalisation. The reusable measurement testbed is intended as an artefact for future studies of adversary classes that emerge after the completion of this work.

References

[1] L. Spitzner, *Honeypots: Tracking Hackers.* Addison-Wesley, 2003.

[2] L. Spitzner, "Honeytokens: The other honeypot," Security Focus / SANS Institute white paper, 2003.

[3] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage.* Doubleday, 1989.

[4] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection.* Addison-Wesley, 2007.

[5] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *arXiv preprint arXiv:1608.06249*, 2016.

[6] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo, "Baiting inside attackers using decoy documents," in *Proc. 5th Int. ICST Conf. on Security and Privacy in Communication Networks (SecureComm)*, 2009.

[7] T. Ishikawa and K. Sakurai, "Parameter manipulation attack prevention and detection by using web application deception proxy," in *Proc. 11th Int. Conf. on Ubiquitous Information Management and Communication (IMCOM).* ACM, 2017.

[8] C. De Faveri and A. Moreira, "Designing adaptive deception strategies," in *2016 IEEE Int. Conf. on Software Quality, Reliability and Security Companion (QRS-C).* IEEE, 2016, pp. 77–84.

[9] M. H. Almeshekah and E. H. Spafford, "Cyber security deception," in *Cyber Deception: Building the Scientific Foundation.* Springer, 2016, pp. 23–50.

[10] G. Deepa and P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Information and Software Technology*, vol. 74, pp. 160–180, 2016.

[11] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[12] M. M. Rathore, A. Ahmad, and A. Paul, "Real time intrusion detection system for ultra-high-speed big data environments," *The Journal of Supercomputing*, vol. 72, pp. 3489–3510, 2016.

[13] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection," *arXiv preprint arXiv:2302.12173*, 2023.

[14] Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

[15] S. Zhou, F. F. Xu *et al.*, "Webarena: A realistic web environment for building autonomous agents," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2024.

*% detected as AI

AI detection includes **the possibility of false positives.** Although some text in this submission is **likely AI generated, scores below the** 20% threshold are not surfaced **because they have a higher likelihood of false positives.**

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

6% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Small Matches (less than 8 words)

Match Groups

- **44** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **6** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
1% Publications
6% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.