Project 13: DNS Resolver

Ruchit Jagodara (22110102)
Chiragkumar Patel (22110183)

1. Objective

The primary goal of this project was to design, develop, and deploy an interactive and educational tool to demystify the complex, end-to-end Domain Name System (DNS) resolution process. The tool is intended to serve as a visual learning aid for users at various levels of technical expertise, from basic to advanced.

To achieve this, the following specific objectives were established:

- **Visualize DNS Resolution:** To create an interactive visualization that illustrates the entire DNS lookup process with animations, a step-by-step timeline, and clear, step-level insights.
- **Implement Dual Modes:** To build two distinct operational modes:
  1. **Deterministic Simulation Mode** for conceptual clarity, guiding users through the canonical resolution flow (client -> resolver -> root -> TLD -> authoritative) and demonstrating caching behavior.
  2. **Live Mode** for hands-on realism, which parses the output of real DNS traces to show stage-by-stage progress, network timings, and potential errors.
- **Cover Security Concepts:** To provide educational modules and visualizations for advanced DNS topics, including security protocols like DNSSEC, DoH, and DoT.
- **Simulate Attack Scenarios:** To develop visual scenarios for common DNS-related attacks, such as cache poisoning (including the Kaminsky race) and MITM flows, to help users understand vulnerabilities.
- **Ensure Usability:** To facilitate fast onboarding and user understanding by integrating a first-run wizard and a comprehensive DNS glossary.

2. Introduction and Theoretical Background

Introduction

The Domain Name System (DNS) is a foundational, yet frequently misunderstood, service that underpins virtually all internet activity. It functions as the internet's global, distributed "phonebook," responsible for translating human-readable domain names, such as `github.com`, into the numerical IP addresses that computers use to communicate. While seamless to the end-user, this translation involves a complex, hierarchical, and multi-step process.

To demystify this "black box," we developed the **DNS Resolution Simulator**: an interactive, web-based educational tool. The project's primary goal is to provide a clear, visual, and hands-on learning environment for anyone interested in network fundamentals.

The application achieves this by visualizing the entire end-to-end lookup, from the user's client to the final authoritative server. It features two distinct operational modes: a **Deterministic Simulation Mode** for step-by-step conceptual clarity and a **Live Mode** that parses real-world DNS traces for practical, hands-on realism. The simulator also provides educational modules for advanced topics, including modern security protocols like DoH, DoT, and DNSSEC , as well as visualizations of common DNS attack scenarios.

Theoretical Background

A core challenge in understanding DNS is its distributed, hierarchical nature. The process is not handled by a single server but by a chain of specialized servers, each with a distinct role:

1. **DNS Resolver:** This is the server (often provided by an ISP) that receives the initial query from a client.
2. **Root Servers:** These servers sit at the top of the hierarchy and direct queries to the correct Top-Level Domain (TLD) server.
3. **TLD Servers:** These manage specific TLDs, such as `.com`, `.org`, or `.in`, and direct queries to the domain's specific authoritative nameserver.
4. **Authoritative Nameservers:** This is the final server in the chain, which holds the actual IP record for the domain and provides the definitive answer.

The simulator's **Deterministic Mode** was built specifically to illustrate this canonical flow, providing a guided, animated walkthrough of the query's path from the resolver to the root, TLD, and authoritative servers.
This process involves two different types of queries. A client sends a **recursive query** to its resolver, which is a "do-it-all" request for the final answer. The resolver then performs a series of **iterative queries**, talking to the root, TLD, and authoritative servers in sequence, to find that answer. This iterative process is what the simulator's **Live Mode** is designed to capture. By executing and parsing the output of a real `dig +trace` command, it reveals the actual stage-by-stage progress, timings, and server responses that are normally hidden from the user.
Furthermore, traditional DNS was not designed with security in mind, leaving it vulnerable to eavesdropping and manipulation. The simulator's **Security Protocols** modules address this by providing conceptual walkthroughs of modern solutions:

- **DoH (DNS over HTTPS)** and **DoT (DNS over TLS)** solve the privacy problem by encrypting DNS queries within standard web traffic or a secure TLS tunnel.
- **DNSSEC (DNS Security Extensions)** solves the authenticity problem by using digital signatures to create a verifiable "chain of trust," ensuring a DNS response has not been tampered with.

3. High-level implementation detail

Architecture Overview

The DNS Resolution Simulator is designed as a modern client-server application, which separates the user interface (frontend) from the core logic (backend).

- **Frontend:** The frontend is a single-page application (SPA) built using **Vite + React**. It is responsible for rendering the user interface, managing user input, and visualizing the data received from the backend.
- **Backend:** The backend is a **Node.js/Express** service that exposes a set of HTTP API endpoints. It handles all the complex logic, including running deterministic simulations, executing live traces, and generating attack scenarios.

API Design

The backend service exposes a clear API for the frontend to consume. The core endpoints include:

- **`POST /api/resolve`:** This is the main endpoint that handles both deterministic and live resolution requests, routing the logic based on the `mode` specified in the request payload.
- **`POST /api/simulate-attack`:** This endpoint produces a step-by-step timeline for a selected DNS attack scenario.
- **`POST /api/simulate-security`:** This endpoint generates the conceptual sequences used to visualize security protocols like DoH, DoT, and DNSSEC.

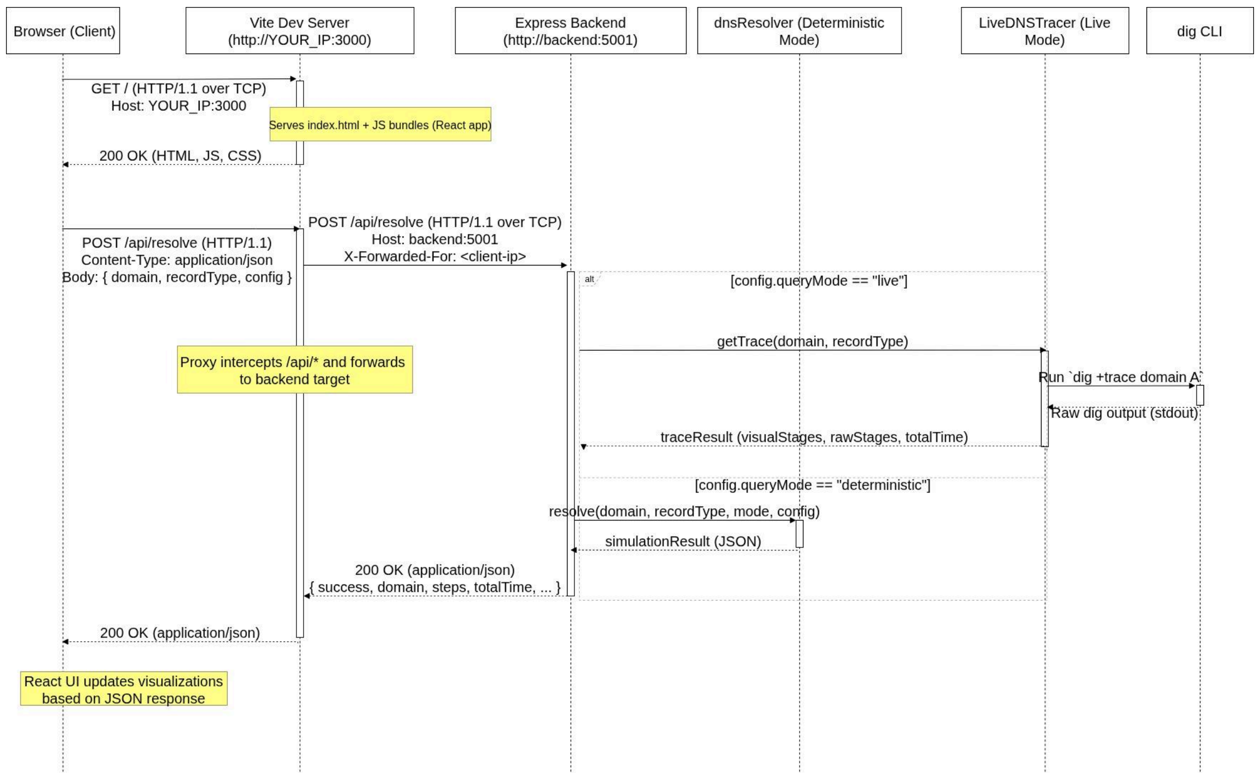
High-Level Data Model

To ensure consistency across the application, all API interactions use a standardized JSON data model.

- **Typical Request:** A request from the frontend typically includes the `domain` (string), `type` (e.g., "A"), `mode` ("deterministic" | "live"), and any other options.
- **Typical Response:** The backend returns a normalized JSON object that contains the `mode` used, an ordered array of `steps`, the `finalAnswer`, and any `errors` or `metrics`. This consistent structure allows the frontend to use the same UI components to render the results from any mode.

Application Request Flow

The complete flow of a user query, from the browser to the backend and back, iss illustrated in the sequence diagram below.



Browser (Client)
Vite Dev Server (http://YOUR_IP:3000)
Express Backend (http://backend:5001)
dnsResolver (Deterministic Mode)
LiveDNSTracer (Live Mode)
dig CLI
GET / (HTTP/1.1 over TCP) Host: YOUR_IP:3000
Serves index.html + JS bundles (React app)
200 OK (HTML, JS, CSS)
POST /api/resolve (HTTP/1.1) Content-Type: application/json Body: { domain, recordType, config }
POST /api/resolve (HTTP/1.1 over TCP) Host: backend:5001 X-Forwarded-For: <client-ip>
alt
[config.queryMode == "live"]
getTrace(domain, recordType)
Proxy intercepts /api/* and forwards to backend target
Run `dig +trace domain A`
Raw dig output (stdout)
traceResult (visualStages, rawStages, totalTime)
[config.queryMode == "deterministic"]
resolve(domain, recordType, mode, config)
simulationResult (JSON)
200 OK (application/json) { success, domain, steps, totalTime, ... }
200 OK (application/json)
React UI updates visualizations based on JSON response

The steps in this flow are as follows:

1. **Browser -> Vite:** The user's browser sends a GET request to the Vite dev server, which returns the React application's HTML, JS, and CSS files.
2. **React App Load:** The frontend application loads in the browser and prepares an API request.
3. **Browser -> Vite (API):** The browser sends a POST /api/resolve request containing the JSON payload with the domain and configuration.
4. **Vite -> Backend:** The Vite server, acting as a proxy, intercepts the /api request and forwards it to the Express backend.
5. **Backend Logic:** The Express server receives the request and checks the config.queryMode to decide which resolver to use.
6. **If Live Mode:** The backend calls the LiveDNSTracer module, which in turn executes a dig +trace command to get real network results.
7. **If Deterministic Mode:** The backend calls the dnsResolver module, which simulates the resolution path step-by-step without making network calls.
8. **Backend -> Vite -> Browser:** The backend returns the JSON response (either from the live trace or the simulation) to the Vite server, which relays it back to the browser.

9. **React UI Update:** The React app receives the JSON data and updates the UI to display the visualizations, timeline, and results.

4. Implementation Details

This section details the internal implementation of the simulator's core features, including the deterministic and live resolution modes, the rationale for the chosen tooling, and the generation of security and attack scenarios.

Deterministic (Simulation) Mode

The deterministic mode is designed to provide a clean, conceptual, and reproducible visualization of the DNS resolution path.

- **Function:** When a user requests a deterministic simulation, the backend does *not* perform any real network lookups. Instead, it synthetically generates the canonical resolution path: Client -> Resolver -> Root -> TLD -> Authoritative Server .
- **Implementation:** The resolver parses the input domain to extract its TLD and labels. It then emits an ordered sequence of "steps," where each step represents a query and a conceptual response from a server in the chain.
- **Educational Focus:** The focus is on pedagogical clarity rather than exact packet details. The simulation generates educational events that illustrate referrals, delegations, and the final answer. It also includes conceptual DNSSEC validation cues. This mode is entirely in-process and sandboxed, ensuring safety and reproducibility.

Live (Real Network) Mode

The live mode provides hands-on realism by executing a real DNS resolution trace against the live internet.

- **Function:** This feature executes a real DNS trace for the given domain and record type using the system's underlying DNS tools. It captures a faithful, stage-by-stage account of the query path, including retries, failures, and timings.
- **Implementation:** The backend executes a `dig +trace` command. The text-based output from this command is then parsed by a state-machine-like parser that segments the log into zones, referrals, answers, and transport attempts. This parsed data is normalized into the same "steps" schema used by the deterministic mode, allowing the UI to render both modes with the same components .
- **Engineering Considerations:** This approach prioritizes authenticity, as it uses real-world DNS behavior without re-implementing the complex recursion logic. The implementation includes strict timeouts and buffer limits to protect the server from long-running or overly verbose traces.

Tooling Choice for Live Mode: `dig +trace`

A significant part of the implementation was selecting the right tool to generate the live traces. After evaluating several alternatives (manual `dig` queries, `dog`, `q`, and `dnspython`), the `dig +trace` command was chosen.

- **Evaluated Alternatives:**
  - **Manual `dig` / `dnspython`:** Tools like `dnspython` or manual `dig +norecurse` queries offer complete programmatic control and can produce structured JSON. However, they require significant implementation effort to manually script the entire iterative chain, including handling referrals, glue records, TCP fallback, and timeouts.
  - **`dog` / `q`:** These modern CLI tools can output JSON directly , which simplifies parsing. However, like the manual methods, they do not perform a full iterative trace automatically and would require a custom script to follow the delegation chain.
- **Reason for Choice:** The `dig +trace` command was selected because it is the only tool that **performs a complete, automatic iterative resolution** starting from the root servers. Despite its text-based output requiring a custom parser , its reliability and ability to accurately mirror real-world resolver behavior (including handling DNSSEC, EDNS, and TCP fallback) made it the most suitable and simplest choice for achieving the project's objective of authentic trace visualization.

Comparative Evaluation Table

Criterion	**dig +trace**	**dig (manual)**	**dog (JSON)**	**q (JSON)**	**Direct Protocol (dnspython)**
Ease of Use	5	3	4	4	2
Automation of Full Trace	5	2	1	1	2

Output Structure (Machine-Readable)	2	2	5	5	5
Protocol Control / Flexibility	2	4	3	3	5
DNSSEC / EDNS Support	5	4	2	2	4
Implementation Complexity	5	3	4	4	1
Reliability / Standard Compliance	5	4	3	3	3
Best for Educational Simulation	4	3	4	4	5
Overall Suitability (for our simulator)	5	3	3	3	4

Attack and Security Visualizations

The simulator's educational modules for attacks and security protocols are implemented as conceptual, timeline-based generators.

- **Attack Scenarios:** The backend produces step-by-step flows for attacks like cache poisoning (including a Kaminsky race visual), NXDOMAIN misuse, and MITM flows. It uses distinct actors (e.g., Client, Attacker) and color-coded paths to visually distinguish legitimate traffic from attack traffic.
- **Security Protocols (DoH/DoT/DNSSEC):** These modules are not full cryptographic implementations. Instead, they generate conceptual sequences that teach how these protocols work. For example, the UI renders animated walkthroughs of encrypted transport (DoT/DoH) and a visual representation of the DNSSEC chain-of-trust validation steps.

5. Improvements

While the project successfully met its primary objectives, several features were planned that could not be implemented due to time constraints. These represent clear opportunities for future development to enhance the simulator's depth and fidelity.

- **Interactive Cache Visualization:** A significant planned feature was an interactive cache simulation for the deterministic mode. This would have allowed users to visualize the cache at different levels (e.g., browser, OS, resolver) and directly modify entries, including adding or deleting records. This would be a powerful tool for demonstrating the mechanics of cache poisoning attacks in a more hands-on way.
- **Deeper Security Protocol Details:** The current visualizations for DoH and DoT are conceptual. A planned improvement was to show the underlying protocol handshakes and the specific packet information being transmitted, offering a more detailed look at how the secure tunnels are established.
- **Full DNSSEC Validation Simulation:** The current DNSSEC module provides educational content rather than a full simulation. The original plan was to build a complete visualization of the certificate validation process, allowing users to step through the entire chain of trust.
- **Enhanced Attack Scenario Fidelity:** The existing attack scenarios could be expanded to include parameterized variants, such as adjustable probabilities for cache poisoning success, randomized TXID race timing, and configurable resolver cache TTLs, all driven by a seeded random number generator for repeatable experiments.
- **Advanced Resolver Artifact Modeling:** The simulation could be made more realistic by modeling advanced resolver behaviors and artifacts. This includes visualizing the effects of EDNS0 on amplification attacks, the impact of QNAME minimization on privacy, and

the mechanisms of resolver defense features like 0x20 bit encoding and source port randomization.

6. References

- Internet Systems Consortium. (n.d.). *dig (domain information groper)*.
- Meta (Facebook). (n.d.). *React – A JavaScript library for building user interfaces*. https://react.dev/
- OpenJS Foundation. (n.d.). *Node.js*. https://nodejs.org/
- OpenJS Foundation. (n.d.). *Express – Node.js web application framework*. https://expressjs.com/
- (n.d.). *A DNS toolkit for Python*. https://www.dnspython.org/
- (n.d.). *dog: A command-line DNS client*. https://github.com/ogham/dog
- (n.d.). *q: A command-line DNS client*. https://github.com/natesales/q
- Mockapetris, P. (1987). *RFC 1034: Domain Names - Concepts and Facilities*.
- Mockapetris, P. (1987). *RFC 1035: Domain Names - Implementation and Specification*.