International Journal of AI, Big Data, Computational and Management Studies
Noble Scholar Research Group | ICAIDSCT26-Conference Proceeding
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.ICAIDSCT26-134

AI Enabled Extraction of Secure Authentication and Data Access Flows from HAR Traces

Surya Ravikumar
Independent Researcher, USA.

Abstract: To safely access protected data modern online and mobile applications rely on sophisticated authentication and authorization techniques like OAuth, OpenID Connect, API keys, multi factor authentication and proprietary token exchanges. Platform integrators, security teams, QA engineers and financial data aggregators must comprehend these methods. Nevertheless, documentation is sometimes sparse, outdated or unavailable. By recording headers, payloads, cookies, redirection and time data, HTTP Archive (HAR) traces offer a comprehensive, low level record of network interactions between clients and servers. In this paper, an AI enabled method for automatically extracting data access procedures and secure authentication from HAR traces is presented. The suggested method reconstructs end-to-end authentication procedures, determines token lifecycles and maps data access patterns by fusing machine learning, sequence modeling and rule based security heuristics. With an emphasis on fintech and regulated contexts, the paper addresses system design, feature engineering, model selection, security issues and real world implementations.

Keywords: HAR Traces, Authentication Workflows, Data Access Flows, Artificial Intelligence, Machine Learning, Oauth, API Security, Fintech Integration, Network Traffic Analysis.

1. Introduction

Workflows for authorization and authentication have become more complex due to the quick expansion of mobile and web based apps. Layered security controls are used to ensure secure access to APIs and user data in industries like cloud platforms, healthcare and financial services. Identity verification, token issuance, consent enforcement and fine grained permission regulations are some of these restrictions.

It might be difficult for developers, integrators and security analysts to comprehend how these workflows function in real world scenarios. Third party access to official API documentation may be restricted, high level or lacking. Using browser developer tools to manually reverse engineer authentication routes is difficult, prone to mistakes and challenging to scale across numerous apps.

Detailed records of HTTP requests and responses are captured by HTTP Archive (HAR) files, which are produced by modern web browsers and proxy software. They offer a genuine ground truth about how data access and authentication take place during runtime. HAR traces, on the other hand, are

Unstructured and verbose, with hundreds or thousands of requests for a single user session. Automation and intelligence are needed to extract useful workflows from this data.

In order to automatically extract safe authentication and data access flows from HAR traces, this article investigates the application of AI algorithms. The objective is to model and comprehend acceptable procedures for integration testing, monitoring, compliance validation and platform interoperability rather than to get around security measures.

2. Background and Motivation

Over the past few years, authentication routines have undergone tremendous change. Modern systems use federated identification protocols based on tokens, whereas earlier systems depended on simple authentication or session cookies. OpenID Connect and OAuth 2.0 are now considered widely accepted standards, especially in the fintech and open banking sectors.

The initial credential submission, redirection to identity providers, consent collection, token exchange, token refresh and allowed API access are all part of these procedures. Multiple HTTP exchanges, frequently across various domains, are produced by each stage. Such flows cannot be scaled by manual analysis. Faster onboarding, lower integration risk and increased security visibility are made possible by AI driven extraction.

3. Overview of HAR Traces

A JSON formatted archive that documents HTTP communication between a client and servers is called a HAR file. HTTP Archive files, which are standardized JSON logs that record everything a browser or mobile client transmits to and receives

from a server throughout a session, are referred to as HAR traces. Consider a HAR file as a web traffic black box flight recorder. It displays you exactly what transpired on the wire, not what the program was supposed to perform.

A HAR trace records every HTTP interaction, including:

- The request URL and HTTP method (GET, POST, PUT, DELETE)
- Request headers such as Authorization, cookies and content types
- Request payloads, including form data or JSON bodies
- Response status codes and headers
- Response payloads
- Redirect chains
- Timing information, such as when requests were sent and how long responses took Because authentication and data access operations are sequences of specific HTTP requests and answers rather than abstract notions, HAR traces are incredibly important.

For example, a simple login flow might appear in a HAR trace as:

- A POST request to a login endpoint with credentials
- A redirect to an identity provider
- A token exchange request
- A response containing an access token
- Subsequent API calls that include that token in the Authorization header

All of these steps are explicitly visible in the HAR file.
The ability of HAR traces to record cross domain behavior is another crucial feature.

Several domains are frequently involved in modern authentication flows, such as:

- The application domain
- A third party identity provider
- An authorization or consent service
- Protected API domains

This cross domain sequence is preserved by HAR traces, which is essential for reconstructing end-to-end authentication flows. HAR traces offer rich, sequential and structured data from an AI standpoint.

They contain both syntactic information (such as URLs and headers) and semantic hints (such as error messages, parameter names and token forms) and they are time stamped and arranged.

HAR traces are noisy, though. Hundreds of unrelated requests, such as those for fonts, photos, analytics calls and background polling, may be present in a single user session. Differentiating signal from noise and determining whether requests are genuinely related to data access or authentication is one of the difficulties.

Security is another key consideration. HAR traces may include:

- Session cookies
- OAuth tokens
- Personally identifiable information

This is why any AI based system must first **sanitize and mask sensitive fields** before analysis.

In summary, HAR traces represent:

- A ground truth view of real authentication behavior
- A protocol level record that is independent of documentation
- A powerful but raw data source that requires intelligence to interpret

This is why they provide such a solid basis for data access operations and safe authentication extraction facilitated by AI.

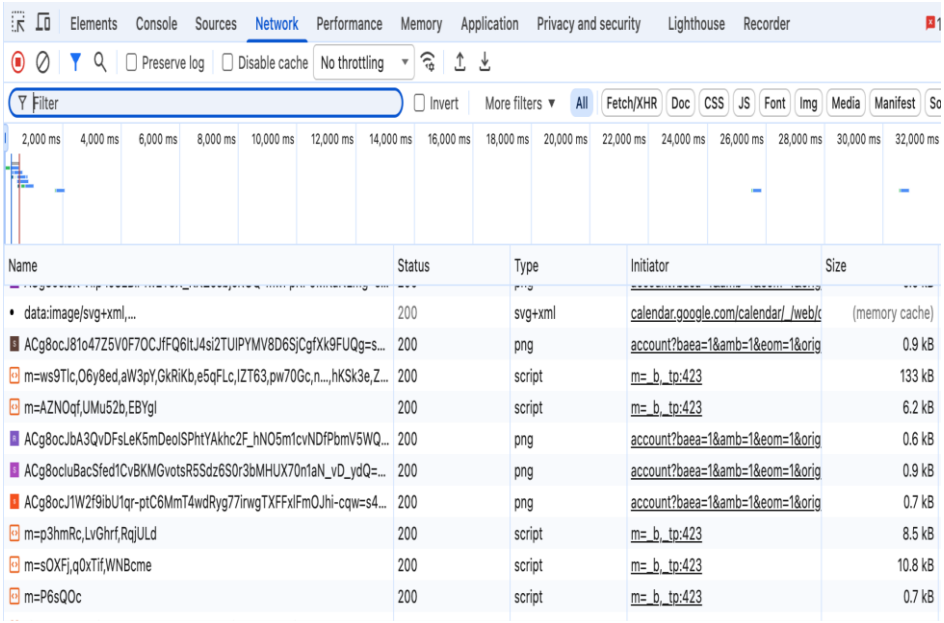

Figure 1: Network Traces Sample


```
{
 "log": {
  "version": "1.2",
  "creator": {
   "name": "WebInspector",
   "version": "537.36"
  },
  "pages": [
   {
    "startedDateTime": "2025-01-01T03:42:32.844Z",
    "id": "page_4",
    "title": "https://connect.secure.xxx.com/auth/login/do",
    "pageTimings": {
     "onContentLoad": 1701.1079993098974,
     "onLoad": 2740.0969993323088
    }
   }
  ],
  "entries": [
   {
    "_connectionId": "33437412",
    "_initiator": {
     "type": "other"
    },
    "_priority": "VeryHigh",
    "_resourceType": "document",
    "cache": {},
    "connection": "443",
    "pageref": "page_4",
    "request": {
     "method": "POST",
     "url": "https://connect.secure.xxxx.com/auth/login/do",
     "httpVersion": "HTTP/1.1",
     "headers": [
      {
       "name": "Accept",
       "value": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
*;q=0.8,application/signed-exchange;v=b3;q=0.7"
```


Figure 2: HAR File Sample

4. Problem Definition

The main issue this study attempts to solve is how to automatically rebuild user interactions with an application given one or more HAR traces.

- Authentication workflows (login, token issuance, refresh, logout)
- Authorization mechanisms (scopes, roles, consent)
- Secure data access flows (API calls gated by authentication)

The solution must minimize false interpretations, support different authentication standards and manage application variability.

Modern apps require complex, multi step authentication procedures, particularly in finance, open banking and SaaS systems. Redirects, token exchanges, consent screens and frequent changes over time are all part of these workflows, which are usually dispersed across several services and domains.

These workflows are invisible from the outside. There are several obstacles for teams attempting to interact with these systems. First, the amount of detail required for implementation of authentication logic is rarely described. Even when there is documentation, it frequently explains the planned flow rather than the runtime behavior.

Secondly, flows of authentication are dynamic. Token lives, refresh methods, scopes, headers and endpoint behavior can change depending on the type of user, location, device or even time. Reverse engineering these flows manually with browser tools is quite error prone and does not scale.

Third, experimentation is risky due to security limitations. Rate limitations, account lockout or security alarms may be triggered, particularly in financial systems, by repeated unsuccessful login attempts, improper token usage or misordered requests.

This leads us to the core problem:
How can we securely and automatically extract the following from one or more HAR traces that record actual user interactions with an application?

- The authentication workflow such as login, redirection, token issuance, refresh and logout
- The authorization model including scopes, roles and consent enforcement
- The secure data access flow which APIs require authentication, what tokens are used and how access is validated And importantly, how can we accomplish this without requiring human intervention, breaching security protocols and presuming system knowledge?

From a technical standpoint, the problem is difficult because HAR traces are:

- Large and noisy, containing many irrelevant requests
- Semi structured, with variability across applications
- Sequential, where meaning depends on request ordering and timing
- Security sensitive, containing confidential information

Bypassing security measures or extracting secrets are not the objectives.

The objective is to recreate valid workflows in a high level, structured format that both systems and people can comprehend.

In other words, we want to transform low level HTTP traffic into a clear answer to questions like:

- Where does authentication start?
- How does identity get established?
- How is access granted and validated?
- Which requests depend on which authentication steps?

Because static rules cannot manage the diversity, scale and growth of contemporary authentication systems, this problem definition lays the groundwork for an AI enabled solution.

5. AI Enabled System Architecture

The proposed system consists of several stages:

- Data Ingestion and Sanitization Sensitive fields like passwords, tokens and personal information are hidden or hashed by ingesting and sanitizing HAR files.
- Feature Extraction HTTP methods, endpoint patterns, header names, parameter keys, response codes and temporal relationships are among the structured aspects that are retrieved from requests and responses.
- Sequence Modeling Data access and authentication are intrinsically sequential processes. Common request response patterns are learned using models like transformer based sequence models, Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs).
- Semantic Classification Requests are categorized by supervised or semi supervised models into groups such as data access, token exchange, refresh, consent and login.
- Workflow Reconstruction A human readable representation of authentication and data access logic is created by assembling classified events into high level workflows.

6. Machine Learning Techniques

Several AI techniques are applicable,

- Natural Language Processing: Used to examine error messages, parameter names and endpoint locations
- Clustering: Identifies recurring trends by grouping similar requests.
- Graph Modeling: Uses graphs to represent interactions with nodes representing endpoints and edges representing transitions.
- Anomaly Detection: Detects variations from the anticipated authentication processes

Synthetic HAR traces, known integrations or semi automated labeling can all be used to create training data.

The main query arises when HAR traces have been consumed and cleaned: How can raw network traffic be used to educate a system to comprehend data access and authentication behavior?

The most important realization is that authentication flows are not single occurrences but rather patterns. They are recurring patterns of requests, answers, redirections and token exchanges that differ between users and sessions. For finding such patterns, machine learning is very well adapted. Feature engineering is the initial method used.

We collect structured characteristics from every HTTP request and response, including:

- HTTP method and status code
- URL path structure and domain
- Header names like Authorization, Cookie or X-API-Key
- Presence and format of tokens
- Request timing and order within a session

These features transform raw JSON logs into a machine readable representation. Next, we apply Natural Language Processing methods to textual fields.

Semantic indications, such as "login," "token," "authorize," "consent," or "refresh," are frequently found in endpoint routes, parameter names and error messages. Even when naming conventions vary between systems, NLP models assist in identifying and classifying these semantically related requests.

Then, comparable requests are grouped together using clustering methods. For instance, all token refresh calls may have somewhat different appearances, but they all have the same essential features. Reducing noise and identifying recurring behavioral patterns between sessions are two benefits of clustering. Sequence modeling is essential because authentication workflows are naturally organized.

Models that learn the likelihood of one request coming after another include transformer based sequence models, Conditional Random Fields and Hidden Markov Models. This enables the system to deduce that specific API requests only take place once authorization is achieved or that a token exchange usually follows a successful login.

In parallel, **classification models** label individual requests or request groups into categories such as:

- Login initiation
- Credential submission
- Token issuance
- Token refresh
- Authorized data access
- Logout or session termination

Both semi supervised learning from unlabeled HAR traces and labeled data from known integrations can be used to train these classifiers. Graph based modeling is used to recreate workflows once requests are categorized and sequences are understood.

Transitions become edges, endpoints become nodes and authentication flows form interconnected subgraphs. Visualizing and reasoning about complex multi domain interactions is made simple by this model. Lastly, anomaly detection methods keep an eye out for departures from workflows that have been learned.

The system can indicate a new HAR trace if it reveals a rearranged sequence, an unexpected token usage or a missing step. This is especially useful for identifying potentially dangerous changes in authentication logic. Crucially, machine learning is not a stand alone process.

The best systems integrate machine learning models with domain specific rules, such as identifying common authorization headers or OAuth token fields. Both accuracy and interpretability are enhanced by this hybrid method.

In conclusion, machine learning makes it possible for the system to go from unprocessed HTTP logs to an intelligent comprehension of authentication and data access behavior by identifying patterns, reassembling workflows and making adjustments when systems change.

7. Security and Compliance Considerations

Handling HAR traces introduces security risks. AI systems must ensure:

- No storage of plaintext credentials
- Strict access controls for trace data
- Compliance with regulations such as GDPR,PSD2 and SOC 2

Importantly, the system is not intended for exploitation or circumventing security measures, but rather for defensive and integrative use cases.

8. Applications and Use Cases

Key applications include:

- Financial Data Aggregation: Rapid onboarding of new financial institutions
- API Integration Testing: Validating authentication flows across environments
- Security Auditing: Verifying adherence to documented auth flows
- Regression Monitoring: Detecting breaking changes in authentication logic

In fintech, this approach significantly reduces manual effort and integration timelines.

9. Challenges and Limitations

Despite its promise, AI based extraction faces challenges:

- Encrypted payloads limit visibility
- Highly dynamic client side logic may obscure flows
- Limited labeled data for training

Hybrid approaches combining AI with expert rules often yield the best results.

10. Conclusion

An effective answer to a persistent issue in contemporary software integration and security analysis is provided by AI-enabled extraction of secure authentication and data access flows from HAR traces. Organizations can increase visibility, save integration costs, and strengthen security posture by converting low-level network data into high-level workflows. Intelligent, automated analysis will become a crucial skill as authentication techniques develop further, especially in regulated industries like fintech.

References

1. RFC 2068: Hypertext Transfer Protocol -- HTTP/1.1. (n.d.). IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc2068
2. Hardt, D., & Jones, M. (2012). The OAUTH 2.0 Authorization Framework: Bearer Token usage. RFC, 6750, 1–18. https://art.tools.ietf.org/html/draft-ietf-oauth-v2-bearer
3. Final: OpenID Connect Core 1.0 incorporating errata set 2. (n.d.). https://openid.net/specs/openid-connect-core-1_0.html
4. Kisller, E. (2025, March 18). HAR files web requests and web traffic. LivePerson Customer Success Center. https://community.liveperson.com/kb/articles/1556-har-files-web-requests-and-web-traffic
5. Malashkevych, O. (2025, June 9). How to download, view, and analyze HAR Files: A Step-by-Step Guide. Webvizio | Website Feedback & Bug Tracking Tool. https://webvizio.com/blog/how-to-download-view-and-analyze-har-files/