RequestSRC

Overview

RequestSRC is a lightweight, server-side **local middleware** designed for real-time monitoring of HTTP requests. Once integrated into your server, it automatically logs request metadata at every endpoint header and performs the following steps:

1. **Captures Client Request Metadata**:
    - **IP Address** (with an optional feature to anonymize the last octet for privacy compliance).
    - **User Agent** (browser, device, or application details).
2. **Enriches Metadata**:
    - If geolocation data isn't present in the packet, it matches the IP address against the **local MaxMind GeoIP dataset** to determine:
        - **Region** (e.g., state or province).
        - **City** (or nearest general location).
3. **Adds Timestamp**:
    - Each request is logged with an accurate timestamp.
4. **Logs Data to a SQL Database**:
    - Stores all traffic data in a SQL database, ready for analysis.
5. **Pre-Built Dashboard**:
    - Provides a built-in, admin-friendly dashboard at /requestSRC, where you can:
        - View traffic logs.
        - Filter data by date, IP, region, or user agent.
6. **Automatic Data Retention**:
    - Logs older than **60 days** are deleted by default, ensuring efficient storage management and compliance.

This tool is ideal for:

- **API Monitoring**: Track the origin, frequency, and usage patterns of incoming requests.
- **Traffic Tracking**: Analyze geographic traffic patterns and peak usage times.
- **Security Auditing**: Detect and respond to abnormal or suspicious activity (e.g., requests from unexpected regions or IPs).

Features

Automatic Request Metadata Logging:

- Logs IP address, geolocation, user agent, and timestamp of each incoming request.

Built-in Dashboard:

- Includes a pre-configured admin route (/requestSRC by default) to view traffic statistics in real-time.

Database Integration:

- Supports PostgreSQL and MySQL out of the box for traffic log storage.

Privacy-Friendly Design:

- Built-in anonymization options for IPs to ensure compliance with data privacy laws (e.g., GDPR, CCPA).

Customizable Server-Level Configuration:

Configure options like:

- **IP Anonymization:** Mask the last octet for privacy.
- **Dashboard Route:** Change the URL for the admin dashboard.
- **Retention Period:** Automatically delete logs older than a specified number of days.
- **Log Filters:** Enable or disable filtering options in the dashboard (e.g., by date or region).
- **Log Format:** Choose between detailed or basic logging.

Lightweight and Flexible:

- Minimal setup; install, integrate, and start tracking immediately.

Getting Started

Installation

Install the package via npm:

```
npm install request-src
```

Basic Setup

Integrate `RequestSRC` into your Express app in minutes:

```
const express = require('express');
const RequestSRC = require('request-src');

const app = express();

// Initialize RequestSRC middleware
const requestSRC = new RequestSRC({
  databaseConfig: {
    user: 'your_user',
    host: 'localhost',
    database: 'your_database',
    password: 'your_password',
    port: 5432,
  },
 anonymize: true,                // Enable IP anonymization

 dashboardRoute: '/requestSRC',  // Set the dashboard route

 retentionPeriod: 60,            // Retain logs for 60 days

 enableFilters: true,            // Enable filters (ex. date, region)

 logFormat: 'detailed',          // Options: 'detailed' (default), 'basic'
});
// Use the middleware to capture traffic
app.use((req, res, next) => {
  requestSRC.add(req,reqType);//func() to capture req in endpoint body
  next();
});

// Mount the built-in dashboard route
app.use(requestSRC.router);

app.get('/', (req, res) => res.send('Hello World'));

app.listen(3000, () => console.log('Server running on port 3000'));
```

Functionality and Definitions

To start using this middleware, you only need to call one of two functions within an endpoint. Simply pass the req object as an argument and define the request category. The middleware will then automatically log each request.
Parameters:

Parameter	Type	Description
req	Object	The incoming request object received via TCP/IP. Extracts metadata such as IP, user-agent, and geolocation.
reqType	String	A user label to categorize the request (e.g., `"user_creation"`, `"account_deletion"`, `"login_attempt"`).

1. `requestSRC.add(req, reqType)`

📌 **Purpose**: Captures request metadata and logs it into the database.

```
requestSRC.add(req, reqType);
```

What Does `requestSRC.add(req, reqType)` Do?

In short: it captures the request, checks the received IP against the local IP dataset, and logs the data in the database.

In detail:

1. **Extracts Request Metadata**:
    - `req.ip`: Extracts client IP address.
    - `req.headers['user-agent']`: Captures user agent details.
    - `req.timestamp`: Logs the timestamp.
    - **Geolocation**: If IP data lacks location, it **queries** the local MaxMind GeoIP dataset.
2. **Stores in the Database**:
    - Saves all extracted data to the SQL database.
    - Includes an extra column (`reqType`) for **categorizing requests** (useful for filtering logs by request type).
3. **Handles Anonymization** (if enabled):
    - Anonymizes the last octet of the IP before storing (`192.168.1.100` → `192.168.1.0`).

Endpoint Examples

```
app.post('/register', (req, res) => {
  requestSRC.add(req, 'user_creation'); // Logs request under
'user_creation'
  res.send('User registration processed');
});

app.delete('/delete-account', (req, res) => {
  requestSRC.add(req, 'account_deletion'); // Logs request under
'account_deletion'
  res.send('Account deletion request logged');
});
```

2. `requestLOG(req, reqType)`

📌 **Purpose**: Captures request metadata **but does NOT store it in the database**—useful for debugging.

```
const logDetails = requestLOG(req, reqType);
```

***Returns**: An object with extracted request metadata.*

What Does `requestLOG(req, reqType)` Do?

1. Extracts the same metadata as `requestSRC.add(req, reqType)`.
2. Does **NOT** store it in the database.
3. **Returns a JSON object** instead, allowing developers to inspect request details.

Example Usage

```
app.get('/debug', (req, res) => {
  const logDetails = requestLOG(req, 'debug_mode');
  console.log(logDetails);
  res.json(logDetails);
});
```

Example Output

```
{
  "ip": "192.168.1.100",
  "anonymized_ip": "192.168.1.0",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)",
  "timestamp": "2025-01-27T12:34:56Z",
  "geo": {
    "country": "US",
    "city": "San Francisco",
    "region": "California"
  },
  "reqType": "debug_mode"
}
```

Customization on Initialization

You can configure `RequestSRC` during initialization to tailor logging behavior, data retention, and the dashboard interface. Below are the available options:

Option	Description	Description
`databaseConfig`	Database connection configuration	`null`
`anonymize`	Anonymize IP addresses (e.g., mask last octet)	`false`
`dashboardRoute`	Custom route for the dashboard	Customizes the admin dashboard route (default: `/requestSRC`).
`retentionPeriod`	Automatically delete logs older than this period	`Number of days before old logs are automatically deleted.`
`enableFilters`	Boolean	`Enables filtering by date, region, or request type in the dashboard.`
`logFormat`	String	`Determines log detail level. Options: 'detailed' (default), 'basic'.`
`logUserAgent`	Include the user agent in logged metadata	`true`

Example: Custom Initialization

```
const requestSRC = new RequestSRC({
  databaseConfig: {
    user: 'your_user',
    host: 'localhost',
    database: 'your_database',
    password: 'your_password',
    port: 5432,
  },
anonymize: true,                // Enable IP anonymization
dashboardRoute: '/adminLogs',   // Customize dashboard route
retentionPeriod: 60,            // Retain logs for 60 days
enableFilters: true,            // Allow filtering in the dashboard
logFormat: 'detailed',          // Log metadata in detailed format
logUserAgent: false             // Disable user-agent logging
});
```

Option Definitions

- `databaseConfig:`
  - Stores traffic logs in a SQL database (PostgreSQL/MySQL).
  - Example structure:

```
databaseConfig: {
  user: 'db_user',
  host: 'localhost',
  database: 'request_logs',
  password: 'securepassword',
  port: 5432
}
```

- `anonymize:`
  - If enabled, it masks the last octet of logged IPs to protect user privacy.
- `dashboardRoute:`
  - Defines the URL path for the built-in traffic log dashboard.
  - Example: Setting `/adminTraffic` makes the dashboard available at:

```
http://your-domain/adminTraffic
```

- **`retentionPeriod`:**
  - **Logs older than the defined number of days will be automatically deleted.**
  - **Setting `retentionPeriod: 0` disables auto-deletion.**
- **`enableFilters`:**
  - **If `true`, the dashboard allows filtering logs by:**
  - **Date range**
  - **Region (Country/City)**
  - **Request type (e.g., "user_creation", "login_attempt")**
- **`logFormat`:**
  - **Determines how much metadata is stored.**
  - **Options:**
    - **`'detailed'`: Includes IP, user agent, timestamp, geolocation.**
    - **`'basic'`: Only logs IP and timestamp.**
- **`logUserAgent`:**
  - **If enabled, stores the `User-Agent` string from requests.**

Accessing the Dashboard

Once installed, access the **traffic monitoring dashboard** at:

```
http://your-domain/requestSRCDash
```