Entropy-Based Rate Limiting API (EBRL API)

An Advanced, Adaptive Security Mechanism for High-Traffic, Distributed Systems

---

Overview

The **Entropy-Based Rate Limiting API (EBRL API)** is a highly advanced, adaptive rate limiting system that leverages entropy analysis and statistical anomaly detection to dynamically throttle request flows across distributed services. It helps prevent DDoS attacks, abuse patterns, scraping attempts, and bot traffic at the edge—without relying on fixed thresholds.

Unlike traditional token bucket or leaky bucket algorithms, **EBRL** calculates the **entropy of user behavior over time** to make intelligent, real-time decisions about which requests to allow, delay, or block.

Use Cases:

- High-security APIs and microservices
- FinTech, crypto, and fraud-sensitive applications
- Gaming servers and real-time platforms
- Rate-limiting multi-tenant cloud applications
- Preventing scraping, credential stuffing, and L7 DDoS

Why Entropy-Based Rate Limiting?

Traditional rate limiting uses static limits like "1000 req/min." But:

Problem	Traditional Limiting	EBRL Advantage
IP rotation	Fails to detect	Uses behavioral entropy per user-agent fingerprint
Bursty bots	Hard to detect	Detects abnormal entropy drop in intervals

Problem	Traditional Limiting	EBRL Advantage
Adaptive attackers	Learn static limits	EBRL is **non-deterministic** and adapts
Multi-tenant API	Unfair per-IP limits	EBRL uses **identity-aware scoring**

SEO keywords: adaptive rate limiting, entropy detection, DDoS prevention API, behavioral throttling, intelligent request filtering, L7 protection, microservices rate limiting, zero-trust API security

---

API Base URL

http

CopyEdit

https://api.yourdomain.com/v1/ebrl/

All endpoints require an **API key** and must be accessed over HTTPS.

Authentication

EBRL uses **HMAC-based API key authentication**:

Headers required:

http

CopyEdit

X-API-KEY: your-api-key

X-REQUEST-SIGNATURE: HMAC-SHA256(payload, secret)

Endpoint: /score

Description:

Returns an **entropy score** and **recommendation** (allow, delay, block) for a given request context.

Request

POST /v1/ebrl/score

Headers:

http

CopyEdit

Content-Type: application/json

X-API-KEY: your-api-key

Body:

json

CopyEdit

```
{
  "ip": "192.168.1.1",
  "userAgent": "Mozilla/5.0...",
  "endpoint": "/api/payments",
  "method": "POST",
  "identity": "user_923445",
  "timestamp": "2025-06-06T12:04:23Z"
}
```

Response

json

CopyEdit

```
{
  "entropyScore": 0.42,
  "recommendation": "delay",
```

```
  "confidence": 92.4,
  "reason": "Low entropy and bursty access pattern detected"
}
```

Explanation:

- entropyScore: Normalized score between 0 and 1
- recommendation: One of allow, delay, block
- confidence: Confidence % in the recommendation
- reason: Human-readable justification

Endpoint: /observe

Description:

Send a **passive observation** (non-blocking) to help improve the entropy model over time.

POST /v1/ebrl/observe

json

CopyEdit

```
{
  "ip": "192.168.1.1",
  "path": "/api/checkout",
  "identity": "user_555555",
  "result": "allowed",
  "latency": 128,
  "statusCode": 200
}
```

This helps the EBRL engine build a more accurate entropy graph over time.

Endpoint: /stats

Description:

Retrieve usage and entropy statistics.

GET /v1/ebrl/stats?identity=user_923445

json

CopyEdit


```
{
  "identity": "user_923445",
  "avgEntropy": 0.78,
  "last7DaysBlocked": 142,
  "lastRequest": {
    "score": 0.39,
    "recommendation": "block"
  }
}
```

Key Features

Feature	Description
Real-time entropy scoring	Detect behavioral anomalies in <10ms
Machine learning adaptive baseline	Auto-adjusts thresholds per user & endpoint
Identity-aware	Scores based on users, not just IPs
Edge-compatible	Deploy as a sidecar or CDN function
Language-agnostic SDKs	Python, Go, Java, Rust, Node.js

Feature	Description
Defense-in-depth	Use with CAPTCHA, WAF, and geo-fencing

Technical Deep Dive

What is Entropy in this context?

Entropy refers to the **unpredictability** or **randomness** in a user's request pattern. Bots tend to show **low entropy** (repetitive patterns), while humans show higher entropy (diverse timing, navigation paths, etc).

EBRL uses:

- **Shannon Entropy** on request intervals
- **Contextual entropy** across paths, methods, and agents
- **Statistical deviation** from known baselines

Formula:

$H = -\sum p(x) * \log_2 p(x)$ $H=−∑p(x)∗log2p(x)$

Where p(x) is the probability of user behaviors (timing, method, IP range).

Example Use Case: Crypto Wallet API

A crypto wallet service using EBRL noticed 33% drop in fraud API calls after detecting anomalously low entropy from scripted login attempts and blocking them in <30ms. With traditional rate limiting, these bots easily bypassed static IP limits.

Performance Benchmarks

Test	Result
10M requests analyzed	1.2s cold start

Test	Result
Score latency	< 8ms
Model update interval	30s
SDK memory footprint	~1.5MB

Integration SDKs

Python

bash

CopyEdit

pip install ebrl-sdk

python

CopyEdit

```
from ebrl import EBRLClient

client = EBRLClient(api_key="your-api-key")

decision = client.score({
    "ip": "198.51.100.1",
    "userAgent": "curl/7.68.0",
    "endpoint": "/api/login",
    "method": "POST",
    "identity": "anon_232"
})
```

```
print(decision.recommendation)
```

Security Best Practices

- Always use HMAC signature validation
- Enable geo-fencing for known abuse origins
- Pair EBRL with CAPTCHA after low-entropy flags
- Rotate API keys regularly
- Monitor entropyScore trends for early warnings

Roadmap (Q3 2025)

- Redis-backed burst memory for edge scoring
- Federated entropy graph training
- Regex-based anomaly tagging
- Advanced dashboard for identity heatmaps

Support

For enterprise support, integration help, or custom SLAs:

security@yourdomain.com
PGP Fingerprint: 7A84 DCE1 3DAB 1C22 …
Docs version: v1.2.6-beta

Final Thoughts

If you're serious about **zero-trust API security**, **adaptive rate limiting**, and **bot defense**, EBRL API provides a smarter, entropy-driven solution that evolves with your traffic—unlike brittle, static thresholds.