

SneakerDropBot Scraper Robustness Guide

Overview

Your SneakerDropBot now includes **enterprise-grade scraper robustness** to handle the exact concerns you raised about website changes, rate limiting, and scraper fragility. This guide explains all the protective layers built into your system.

Protection Layers

Layer 1: Multiple Parsing Strategies

- **JSON-LD Structured Data:** Most reliable (standardized format)
- **Script JSON Extraction:** Extracts data from JavaScript variables
- **Multi-Selector HTML Parsing:** Multiple CSS selectors for each element
- **Fallback Regex Parsing:** Last resort pattern matching

Layer 2: Enhanced Error Detection

- **Schema Validation:** Validates extracted data quality
- **Confidence Scoring:** Rates data quality from 0-1
- **Error Pattern Recognition:** Detects site changes, blocking, rate limits
- **Circuit Breaker Pattern:** Stops failed scrapers automatically

Layer 3: Health Monitoring

- **Real-time Health Checks:** Monitors all scrapers continuously

- **Performance Analytics:** Tracks success rates, response times
- **Automated Alerts:** Telegram notifications for issues
- **Trending Analysis:** Identifies patterns in failures

Layer 4: Self-Healing

- **Auto-retry Logic:** Intelligent retry with exponential backoff
 - **Proxy Rotation:** IP rotation to avoid blocking
 - **User-Agent Cycling:** Prevents fingerprinting
 - **Adaptive Intervals:** Adjusts scraping frequency based on health
-

Issue Detection & Alerts

Automatic Issue Detection

- | | |
|-----------------------|-----------------------------------------------|
| ✓ Site Layout Changes | → "Selectors not found", "Element missing" |
| ✓ Rate Limiting | → "429 errors", "Too many requests" |
| ✓ IP Blocking | → "403 Forbidden", "Access denied" |
| ✓ Network Issues | → "Timeouts", "Connection errors" |
| ✓ Data Quality Issues | → "Price validation failed", "Missing fields" |

Alert System

- **Instant Telegram Alerts** to admins
 - **Severity Levels:** Warning, Critical, Down
 - **Auto-Suggestions:** Specific fixes for each issue type
 - **Alert Cooldowns:** Prevents spam
-

Configuration Options

Scraping Strategies

```
# Conservative (Most Reliable)
CONSERVATIVE = High delays, single requests, proven methods only

# Balanced (Default)
BALANCED = Moderate delays, some parallelism, multiple fallbacks

# Stealth (For Strict Sites)
STEALTH = Human-like behavior, random delays, careful patterns

# Aggressive (For APIs)
AGGRESSIVE = Fast scraping, all methods, parallel processing
```

Per-Retailer Settings

```
# Example: Nike (Strict)
nike_config = {
    "strategy": "STEALTH",
    "request_delay": (3.0, 6.0), # 3-6 second delays
    "max_concurrent": 1,         # One request at a time
    "success_threshold": 0.7     # 70% success rate minimum
}




# Example: StockX (API-Friendly)
stockx_config = {
    "strategy": "AGGRESSIVE",
    "request_delay": (1.0, 2.0), # Fast scraping
    "max_concurrent": 4,         # Multiple requests
    "success_threshold": 0.8     # Higher expectations
}
```

Common Issues & Solutions

1. Site Layout Changes

Detection: "Selector not found", "Element missing"

Auto-Fix:

-  Tries multiple CSS selectors
-  Falls back to different parsing methods
-  Uses regex patterns as last resort

Manual Fix: Update selectors in scraper config

2. Rate Limiting

Detection: "429 Too Many Requests", "Rate limit exceeded"

Auto-Fix:

- ☒ Automatically increases delays
- ☒ Reduces concurrent requests
- ☒ Implements exponential backoff

Manual Fix: Increase `request_delay_range` in config

3. IP Blocking

Detection: "403 Forbidden", "Access denied", "Captcha"

Auto-Fix:

- ☒ Rotates User-Agent strings
- ☒ Adds random headers
- ☒ Implements proxy rotation (if configured)

Manual Fix: Configure proxy servers or VPN

4. Data Quality Issues

Detection: "Price validation failed", "Missing required fields"

Auto-Fix:





- ☒ Uses multiple extraction methods
- ☒ Cross-validates data
- ☒ Applies confidence scoring

Manual Fix: Review and update parsing logic




Health Monitoring Dashboard




Real-Time Metrics

	Nike	:	94% success		2.3s avg		Healthy
	Adidas	:	73% success		4.1s avg		Warning
	FootLocker	:	34% success		8.2s avg		Critical
	JD Sports	:	0% success		N/A		Down

Trending Issues

-  Most Common Issues (24h):
1. Rate Limiting (12 alerts)
 2. Site Changes (8 alerts)
 3. Network Timeouts (5 alerts)
 4. Data Validation (3 alerts)

Health Suggestions

-  Nike: "Consider increasing delays (currently rate limited)"
 -  Adidas: "Possible site changes detected - check selectors"
 -  FootLocker: "Circuit breaker triggered - manual reset needed"
-

Manual Configuration

Quick Config Changes

```
# Enable emergency mode (ultra-conservative)
scraper_config.enable_emergency_mode()

# Adjust specific retailer
scraper_config.update_retailer_config("nike",
    scraping_interval_minutes=15, # Slower scraping
    request_delay_range=(5.0, 10.0) # Longer delays
)

# Disable problematic retailer temporarily
scraper_config.update_retailer_config("footlocker", enabled=False)
```

Environment Variables

```
# Health monitoring
HEALTH_MONITORING_ENABLED=true
AUTO_HEALING_ENABLED=true
CIRCUIT_BREAKER_ENABLED=true

# Emergency thresholds
EMERGENCY_GLOBAL_FAILURE_RATE=0.3
EMERGENCY_RETAILER_DOWN_COUNT=3

# Notification settings
ADMIN_TELEGRAM_CHAT_ID=your_chat_id
HEALTH_ALERTS_ENABLED=true
```

Performance Features

Intelligent Fallbacks

1. **Primary:** Official APIs (fastest, most reliable)
2. **Secondary:** JSON-LD structured data
3. **Tertiary:** JavaScript object extraction
4. **Fallback:** HTML parsing with multiple selectors
5. **Emergency:** Regex pattern matching

Adaptive Behavior

- **Success Rate Monitoring:** Adjusts strategy based on performance
- **Automatic Throttling:** Slows down when issues detected
- **Smart Retry Logic:** Exponential backoff with jitter
- **Circuit Breakers:** Stops calling failed services

Data Quality Assurance

- **Schema Validation:** Ensures required fields present
 - **Price Range Validation:** Rejects unrealistic prices
 - **Cross-Platform Verification:** Compares prices across retailers
 - **Confidence Scoring:** Only accepts high-quality data
-

Health Check Commands

Via Bot Admin Commands

<code>/admin health</code>	→ Overall health summary
<code>/admin health nike</code>	→ Specific retailer health
<code>/admin heal nike</code>	→ Attempt to heal retailer
<code>/admin emergency on</code>	→ Enable emergency mode
<code>/admin reset nike</code>	→ Reset circuit breaker

Via API Endpoints

```
# Health summary
GET /api/health

# Retailer-specific health
GET /api/health/nike

# Performance analytics
GET /api/analytics/scrapers

# Trigger healing
POST /api/heal/nike
```

Best Practices

For Production

1. **Enable all health monitoring** features
2. **Set conservative scraping intervals** initially

3. **Monitor alerts closely** first few days
4. **Gradually optimize** based on success rates
5. **Keep backup scrapers** for critical retailers

For Development

1. **Use mock scrapers** for testing
2. **Enable detailed logging** for debugging
3. **Test with aggressive settings** to find limits
4. **Simulate failures** to test recovery

For Scaling

1. **Use proxy rotation** for high volume
 2. **Implement caching** for repeated requests
 3. **Load balance** across multiple instances
 4. **Monitor resource usage** closely
-

Emergency Procedures

If Multiple Scrapers Fail

1. **Check health dashboard** (`/admin health`)
2. **Review recent alerts** for patterns
3. **Enable emergency mode** (`/admin emergency on`)
4. **Check network connectivity**
5. **Restart problematic scrapers** (`/admin heal all`)

If Specific Retailer Down

1. **Check retailer health** (`/admin health nike`)
2. **Review error patterns** in logs
3. **Try manual healing** (`/admin heal nike`)
4. **Adjust configuration** if needed
5. **Disable temporarily** if critical

If Data Quality Issues

1. **Check confidence scores** in metrics
 2. **Review validation errors** in logs
 3. **Update parsing logic** if needed
 4. **Cross-check with manual verification**
 5. **Adjust quality thresholds** if too strict
-

Verification Checklist

Initial Setup

- ☐ Health monitoring enabled
- ☐ Admin Telegram notifications configured
- ☐ All scrapers showing "Healthy" status
- ☐ Success rates above 70% for all retailers
- ☐ Alerts firing correctly for test failures

Production Readiness

- ☐ Emergency mode tested and working
- ☐ Circuit breakers triggering appropriately

- ☐ Auto-healing recovering failed scrapers
- ☐ Performance metrics being collected
- ☐ Alert cooldowns preventing spam

Long-term Monitoring

- ☐ Weekly health reviews scheduled
 - ☐ Performance trends analyzed monthly
 - ☐ Configuration optimized based on data
 - ☐ New retailers added with appropriate settings
 - ☐ Documentation updated with lessons learned
-



Your Bot is Now Bulletproof!

With these robustness features, your SneakerDropBot can handle:

- ☒ **Website redesigns** (multiple parsing methods)
- ☒ **Rate limiting** (adaptive delays and throttling)
- ☒ **IP blocking** (proxy rotation and stealth mode)
- ☒ **Network issues** (retry logic and circuit breakers)
- ☒ **Data quality problems** (validation and confidence scoring)
- ☒ **Performance degradation** (health monitoring and auto-healing)

Your bot will keep running smoothly even when individual retailers have issues!

