Visualization Dashboards Documentation

Overview

Audityzer provides comprehensive visualization dashboards for monitoring Web3 security, cross-chain bridge analytics, and real-time threat detection. This documentation covers dashboard setup, configuration, and customization.

Dashboard Types

Security Dashboard

Real-time security monitoring and vulnerability tracking
- **Vulnerability Detection**: Live vulnerability scanning results
- **Threat Intelligence**: Real-time threat feeds and indicators
- **Risk Assessment**: Dynamic risk scoring and heat maps
- **Incident Response**: Security incident tracking and response

Bridge Analytics Dashboard

Cross-chain bridge monitoring and analytics
- **Transaction Flow**: Visual representation of cross-chain transactions
- **Liquidity Monitoring**: Real-time liquidity pool analytics
- **Performance Metrics**: Bridge performance and reliability metrics
- **Cost Analysis**: Gas optimization and cost tracking

AI Monitoring Dashboard

AI model performance and detection analytics
- **Model Performance**: AI model accuracy and performance metrics
- **Detection Analytics**: Vulnerability detection patterns and trends
- **Training Metrics**: Model training progress and validation
- **Prediction Confidence**: AI prediction confidence scores

Setup and Installation

Prerequisites

```
# Required dependencies
npm install @audityzer/dashboard
npm install react react-dom
npm install @mui/material @emotion/react @emotion/styled
npm install recharts d3 plotly.js
npm install socket.io-client
```

Basic Setup

```
# Start the dashboard server
npm run dashboard:start

# Or with custom configuration
npm run dashboard:start -- --config ./dashboard.config.js --port 3001
```

Docker Setup

```
# Using Docker Compose
docker-compose -f docker-compose.dashboard.yml up -d

# Or standalone Docker
docker run -p 3001:3001 -v $(pwd)/config:/app/config audityzer/dashboard
```

Configuration

Dashboard Configuration

```
// dashboard.config.js
module.exports = {
  server: {
    port: 3001,
    host: '0.0.0.0'
  },

  dashboards: {
    security: {
      enabled: true,
      refreshInterval: 5000, // 5 seconds
      widgets: [
        'vulnerability-scanner',
        'threat-intelligence',
        'risk-assessment',
        'incident-tracker'
      ]
    },

    bridge: {
      enabled: true,
      refreshInterval: 10000, // 10 seconds
      protocols: ['layerzero', 'stargate', 'radiant'],
      networks: ['ethereum', 'polygon', 'arbitrum'],
      widgets: [
        'transaction-flow',
        'liquidity-monitor',
        'performance-metrics',
        'cost-analysis'
      ]
    },

    ai: {
      enabled: true,
      refreshInterval: 30000, // 30 seconds
      models: ['vulnerability-detection', 'pattern-recognition'],
      widgets: [
        'model-performance',
        'detection-analytics',
        'training-metrics',
        'prediction-confidence'
      ]
    }
  },

  data: {
    sources: {
      prometheus: 'http://localhost:9090',
      elasticsearch: 'http://localhost:9200',
      database: 'postgresql://user:pass@localhost:5432/audityzer'
    },

    realtime: {
      websocket: true,
      socketio: true,
      sse: false
    }
  },
```

```
  authentication: {
    enabled: true,
    provider: 'oauth2',
    config: {
      clientId: 'your-client-id',
      clientSecret: 'your-client-secret',
      redirectUri: 'http://localhost:3001/auth/callback'
    }
  },

  theme: {
    mode: 'dark',
    primaryColor: '#1976d2',
    secondaryColor: '#dc004e'
  }
};
```

Environment Variables

```
# Dashboard Configuration
DASHBOARD_PORT=3001
DASHBOARD_HOST=0.0.0.0
DASHBOARD_BASE_URL=http://localhost:3001

# Data Sources
PROMETHEUS_URL=http://localhost:9090
ELASTICSEARCH_URL=http://localhost:9200
DATABASE_URL=postgresql://user:pass@localhost:5432/audityzer

# Authentication
AUTH_ENABLED=true
AUTH_PROVIDER=oauth2
AUTH_CLIENT_ID=your-client-id
AUTH_CLIENT_SECRET=your-client-secret

# Real-time Updates
WEBSOCKET_ENABLED=true
REFRESH_INTERVAL=5000

# Theme
THEME_MODE=dark
PRIMARY_COLOR=#1976d2
SECONDARY_COLOR=#dc004e
```

Security Dashboard

Vulnerability Scanner Widget

```
import { VulnerabilityScanner } from '@audityzer/dashboard';

function SecurityDashboard() {
  return (
    <VulnerabilityScanner
      config={{
        refreshInterval: 5000,
        autoRefresh: true,
        filters: {
          severity: ['high', 'critical'],
          status: ['active', 'new']
        },
        display: {
          chart: 'donut',
          table: true,
          alerts: true
        }
      }}
      onVulnerabilityDetected={(vulnerability) => {
        console.log('New vulnerability detected:', vulnerability);
        // Handle vulnerability detection
      }}
    />
  );
}
```

Threat Intelligence Widget

```
import { ThreatIntelligence } from '@audityzer/dashboard';

function ThreatIntelligenceWidget() {
  return (
    <ThreatIntelligence
      config={{
        sources: ['mitre', 'cve', 'web3-threats'],
        updateInterval: 60000, // 1 minute
        display: {
          timeline: true,
          heatmap: true,
          indicators: true
        }
      }}
      filters={{
        timeRange: '24h',
        severity: ['medium', 'high', 'critical'],
        categories: ['defi', 'bridge', 'smart-contract']
      }}
    />
  );
}
```

Risk Assessment Widget

```
import { RiskAssessment } from '@audityzer/dashboard';

function RiskAssessmentWidget() {
  return (
    <RiskAssessment
      config={{
        metrics: [
          'vulnerability-score',
          'threat-level',
          'exposure-rating',
          'compliance-score'
        ],
        visualization: 'radar-chart',
        thresholds: {
          low: 30,
          medium: 60,
          high: 80
        }
      }}
      onRiskChange={(riskLevel) => {
        if (riskLevel === 'high') {
          // Trigger alert
        }
      }}
    />
  );
}
```

Bridge Analytics Dashboard

Transaction Flow Widget

```
import { TransactionFlow } from '@audityzer/dashboard';

function BridgeAnalyticsDashboard() {
  return (
    <TransactionFlow
      config={{
        protocols: ['layerzero', 'stargate'],
        networks: ['ethereum', 'polygon', 'arbitrum'],
        visualization: 'sankey-diagram',
        timeRange: '1h',
        filters: {
          minAmount: 1000, // USD
          status: ['completed', 'pending']
        }
      }}
      onTransactionClick={(transaction) => {
        // Show transaction details
      }}
    />
  );
}
```

Liquidity Monitor Widget

```
import { LiquidityMonitor } from '@audityzer/dashboard';

function LiquidityMonitorWidget() {
  return (
    <LiquidityMonitor
      config={{
        protocols: ['stargate', 'radiant'],
        pools: ['USDC', 'USDT', 'ETH'],
        metrics: ['tvl', 'utilization', 'apy'],
        alerts: {
          lowLiquidity: 0.1, // 10%
          highUtilization: 0.9 // 90%
        }
      }}
      onAlert={(alert) => {
        console.log('Liquidity alert:', alert);
      }}
    />
  );
}
```

Performance Metrics Widget

```
import { PerformanceMetrics } from '@audityzer/dashboard';

function PerformanceMetricsWidget() {
  return (
    <PerformanceMetrics
      config={{
        metrics: [
          'success-rate',
          'average-latency',
          'throughput',
          'error-rate'
        ],
        timeRange: '24h',
        aggregation: '5m',
        targets: {
          successRate: 0.99,
          averageLatency: 30000, // 30 seconds
          errorRate: 0.01
        }
      }}
      visualization="time-series"
    />
  );
}
```

AI Monitoring Dashboard

Model Performance Widget

```
import { ModelPerformance } from '@audityzer/dashboard';

function AIMonitoringDashboard() {
  return (
    <ModelPerformance
      config={{
        models: [
          'vulnerability-detection',
          'pattern-recognition',
          'anomaly-detection'
        ],
        metrics: [
          'accuracy',
          'precision',
          'recall',
          'f1-score'
        ],
        timeRange: '7d',
        benchmarks: {
          accuracy: 0.95,
          precision: 0.90,
          recall: 0.85
        }
      }}
    />
  );
}
```

Detection Analytics Widget

```
import { DetectionAnalytics } from '@audityzer/dashboard';

function DetectionAnalyticsWidget() {
  return (
    <DetectionAnalytics
      config={{
        detectionTypes: [
          'reentrancy',
          'overflow',
          'unauthorized-access',
          'price-manipulation'
        ],
        visualization: 'heatmap',
        groupBy: 'hour',
        showTrends: true
      }}
      onPatternDetected={(pattern) => {
        // Handle pattern detection
      }}
    />
  );
}
```

Custom Widgets

Creating Custom Widgets

```
import { Widget, useRealTimeData } from '@audityzer/dashboard';

function CustomSecurityWidget({ config }) {
  const data = useRealTimeData({
    endpoint: '/api/security/custom-metrics',
    refreshInterval: config.refreshInterval || 10000
  });

  return (
    <Widget
      title="Custom Security Metrics"
      loading={data.loading}
      error={data.error}
    >
      <div className="custom-widget">
        {/* Custom widget content */}
        <div className="metric">
          <span className="label">Active Scans:</span>
          <span className="value">{data.activeScans}</span>
        </div>
        <div className="metric">
          <span className="label">Threats Blocked:</span>
          <span className="value">{data.threatsBlocked}</span>
        </div>
      </div>
    </Widget>
  );
}
```

Widget Configuration

```
// Register custom widget
import { registerWidget } from '@audityzer/dashboard';

registerWidget('custom-security', CustomSecurityWidget, {
  category: 'security',
  description: 'Custom security metrics widget',
  configSchema: {
    refreshInterval: {
      type: 'number',
      default: 10000,
      min: 1000,
      max: 300000
    }
  }
});
```

Real-time Updates

WebSocket Integration

```
import { useWebSocket } from '@audityzer/dashboard';

function RealTimeWidget() {
  const { data, connected, error } = useWebSocket({
    url: 'ws://localhost:3001/ws',
    topics: ['security-alerts', 'bridge-transactions'],
    onMessage: (message) => {
      console.log('Real-time update:', message);
    }
  });

  return (
    <div className="realtime-widget">
      <div className="connection-status">
        Status: {connected ? 'Connected' : 'Disconnected'}
      </div>
      {/* Widget content */}
    </div>
  );
}
```

Server-Sent Events

```
import { useSSE } from '@audityzer/dashboard';

function SSEWidget() {
  const { data, connected } = useSSE({
    url: '/api/events/security',
    onEvent: (event) => {
      // Handle server-sent event
    }
  });

  return (
    <div className="sse-widget">
      {/* Widget content */}
    </div>
  );
}
```

Theming and Customization

Custom Themes

```
// themes/custom-theme.js
export const customTheme = {
  palette: {
    mode: 'dark',
    primary: {
      main: '#1976d2',
      light: '#42a5f5',
      dark: '#1565c0'
    },
    secondary: {
      main: '#dc004e',
      light: '#ff5983',
      dark: '#9a0036'
    },
    background: {
      default: '#121212',
      paper: '#1e1e1e'
    }
  },
  typography: {
    fontFamily: '"Roboto", "Helvetica", "Arial", sans-serif',
    h1: {
      fontSize: '2.5rem',
      fontWeight: 300
    }
  },
  components: {
    MuiCard: {
      styleOverrides: {
        root: {
          borderRadius: 8,
          boxShadow: '0 4px 6px rgba(0, 0, 0, 0.1)'
        }
      }
    }
  }
};
```

Applying Custom Themes

```
import { ThemeProvider } from '@mui/material/styles';
import { customTheme } from './themes/custom-theme';

function App() {
  return (
    <ThemeProvider theme={customTheme}>
      <Dashboard />
    </ThemeProvider>
  );
}
```

Performance Optimization

Data Caching

```
import { CacheProvider } from '@audityzer/dashboard';

function OptimizedDashboard() {
  return (
    <CacheProvider
      config={{
        ttl: 30000, // 30 seconds
        maxSize: 100, // 100 entries
        strategy: 'lru'
      }}
    >
      <Dashboard />
    </CacheProvider>
  );
}
```

Lazy Loading

```
import { lazy, Suspense } from 'react';

const SecurityDashboard = lazy(() => import('./SecurityDashboard'));
const BridgeDashboard = lazy(() => import('./BridgeDashboard'));

function App() {
  return (
    <Suspense fallback={<div>Loading...</div>}>
      <SecurityDashboard />
      <BridgeDashboard />
    </Suspense>
  );
}
```

API Integration

REST API

```
import { ApiClient } from '@audityzer/dashboard';

const api = new ApiClient({
  baseURL: 'http://localhost:3000/api',
  timeout: 10000,
  headers: {
    'Authorization': 'Bearer your-token'
  }
});

// Fetch security metrics
const securityMetrics = await api.get('/security/metrics');

// Fetch bridge analytics
const bridgeAnalytics = await api.get('/bridge/analytics', {
  params: {
    timeRange: '24h',
    protocols: ['layerzero', 'stargate']
  }
});
```

GraphQL Integration

```
import { ApolloClient, InMemoryCache, gql } from '@apollo/client';

const client = new ApolloClient({
  uri: 'http://localhost:3000/graphql',
  cache: new InMemoryCache()
});

const GET_SECURITY_DATA = gql`
  query GetSecurityData($timeRange: String!) {
    vulnerabilities(timeRange: $timeRange) {
      id
      severity
      type
      status
      createdAt
    }
    threats(timeRange: $timeRange) {
      id
      category
      confidence
      indicators
    }
  }
`;

const { data, loading, error } = useQuery(GET_SECURITY_DATA, {
  variables: { timeRange: '24h' }
});
```

Deployment

Production Deployment

```
# Build for production
npm run build

# Start production server
npm run start:prod

# Or with PM2
pm2 start ecosystem.config.js --env production
```

Docker Deployment

```
# Dockerfile
FROM node:16-alpine

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY . .
RUN npm run build

EXPOSE 3001

CMD ["npm", "run", "start:prod"]
```

Kubernetes Deployment

```yaml
# k8s/dashboard-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: audityzer-dashboard
spec:
  replicas: 3
  selector:
    matchLabels:
      app: audityzer-dashboard
  template:
    metadata:
      labels:
        app: audityzer-dashboard
    spec:
      containers:
      - name: dashboard
        image: audityzer/dashboard:latest
        ports:
        - containerPort: 3001
        env:
        - name: NODE_ENV
          value: "production"
        - name: DATABASE_URL
          valueFrom:
            secretKeyRef:
              name: audityzer-secrets
              key: database-url
```

Monitoring and Alerting

Health Checks

```javascript
// Health check endpoint
app.get('/health', (req, res) => {
  const health = {
    status: 'healthy',
    timestamp: new Date().toISOString(),
    services: {
      database: checkDatabaseHealth(),
      websocket: checkWebSocketHealth(),
      cache: checkCacheHealth()
    }
  };

  res.json(health);
});
```

Metrics Collection

```
import { register, Counter, Histogram } from 'prom-client';

const dashboardRequests = new Counter({
  name: 'dashboard_requests_total',
  help: 'Total number of dashboard requests',
  labelNames: ['method', 'route', 'status']
});

const dashboardResponseTime = new Histogram({
  name: 'dashboard_response_time_seconds',
  help: 'Dashboard response time in seconds',
  labelNames: ['method', 'route']
});

// Metrics endpoint
app.get('/metrics', (req, res) => {
  res.set('Content-Type', register.contentType);
  res.end(register.metrics());
});
```

Troubleshooting

Common Issues

Dashboard Not Loading

```
# Check server status
curl http://localhost:3001/health

# Check logs
docker logs audityzer-dashboard

# Verify configuration
npm run config:validate
```

Real-time Updates Not Working

```
// Check WebSocket connection
const ws = new WebSocket('ws://localhost:3001/ws');
ws.onopen = () => console.log('Connected');
ws.onerror = (error) => console.error('WebSocket error:', error);
```

Performance Issues

```
# Enable performance monitoring
NODE_ENV=production npm run start -- --profile

# Check memory usage
node --inspect app.js
```

Examples

See the examples directory (../examples/dashboards/) for complete dashboard implementations and configurations.

For more information, visit our documentation site (https://docs.audityzer.com) or join our Discord community (https://discord.gg/audityzer).