OrbitWatch: Advanced Space Domain Awareness (SDA) Platform

By: Ritvik Indupuri

Date: January 23, 2026

Executive Summary

The Geostationary Earth Orbit (GEO) belt is a critical area in space housing vital communications and warning satellites. However, it is difficult to tell the difference between normal satellite drifting and deliberate maneuvers in this region. Traditional systems often struggle to filter out the noise in satellite movement data in real-time due to slow internet connections and reliance on distant servers.

OrbitWatch addresses this issue by running analysis directly on the user's computer ("client-side") instead of a central cloud server. By using modern browser technologies, OrbitWatch moves the heavy math and detection work to the local machine. This allows the system to load satellite data, run accurate physics models, and detect threats instantly without waiting for a server response. The result is a fast and reliable tool that provides immediate awareness of satellite movements and signal issues, functioning securely even when disconnected from the internet.

1. System Architecture

OrbitWatch uses a "Thick Client" design, which means the web browser does the heavy lifting instead of just displaying pictures. This ensures the system works even with slow internet connections because the physics and detection models run in the computer's local memory.

1.1 Data Flow & Processing Pipeline

The architecture is divided into three distinct layers:

1. **Ingest Layer:** Fetches raw satellite data from external providers (like Space-Track.org) and saves it to a local database for quick access.
2. **Computation Layer:** A background processing core where the physics engine and the detection models run. This ensures that heavy calculations never freeze the buttons or maps on the screen.
3. **Presentation Layer:** A high-performance interface that manages the dashboard and a 3D view for showing satellite positions.

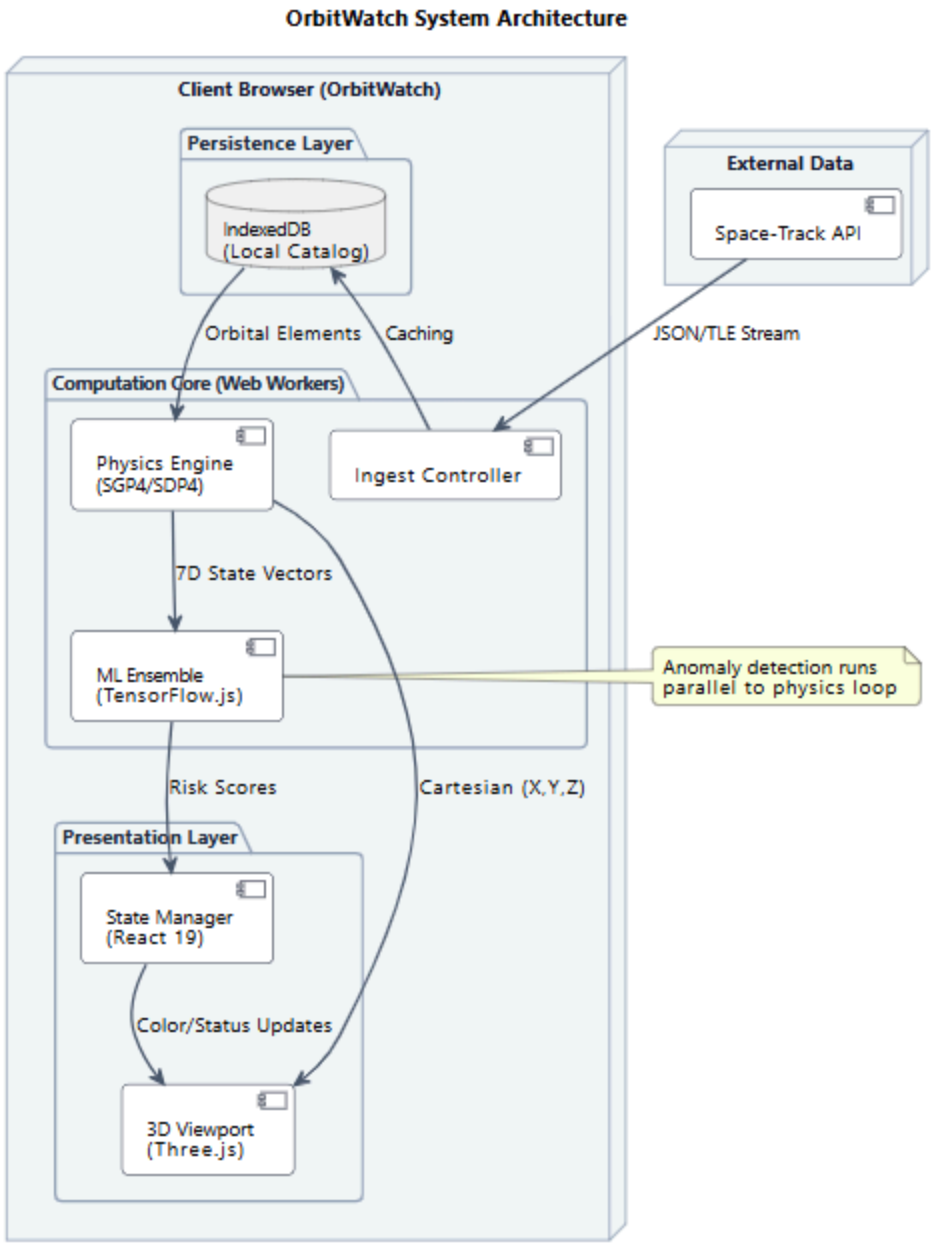

Figure 1: The OrbitWatch Client-Side Architecture

2. Data Lake & Storage Architecture

The architecture described above relies entirely on having quick access to data. Instead of relying on a company server to hold this information, the application builds its own database directly inside the user's browser.

2.1 Ingest Logic

- **Source:** The system gets data from Space-Track.org.
- **Filter:** It only downloads data for satellites in the Geostationary belt. This prevents data from other orbits from confusing the detection models.
- **Update Rate:** The system checks for new data every 60 seconds.

2.2 Storage

Data is saved in the browser's local database (IndexedDB). The system keeps the last five updates, creating a history of about 2,000 to 5,000 records. This history allows the system to analyze trends over time rather than just looking at a single moment.

3. Orbital Dynamics: Movement Prediction

Once the data is securely stored in the Data Lake, the system's next task is to calculate exactly where each satellite is located. OrbitWatch does not use pre-made animations; every dot on the screen is drawn using live physics calculations derived from standard satellite math (SGP4).

3.1 Current Location

- **Apogee & Perigee:** Calculated using the shape and size of the orbit relative to the Earth's radius.
- **Period:** The time it takes for one full orbit, calculated from the satellite's speed.

3.2 Historical Path

To verify the accuracy of the current location, the system calculates the satellite's past movement using **SGP4 Back-Propagation**. The charts for Altitude and Velocity are generated by iterating backwards in time:

- The system starts at the current time and calculates 100 steps into the past using the SGP4 physics model.
- At each step, it determines the exact geodetic height and velocity magnitude.
- A line labeled **PRESENT** connects the historical data with the live stream to visually confirm accuracy.

Figure 2: Orbital Telemetry Graphs

4. Feature Engineering: The 7D Vector

While the physics engine tells us *where* a satellite is, it does not tell us *why* it is moving that way. To understand intent, the system extracts a list of 7 key features from the physics data. These features act as a "Fingerprint" for the satellite's movement.

Feature	Symbol	Purpose
Inclination	i	Detects changes in the orbit's tilt.
Eccentricity	e	Identifies changes in orbit shape (circle vs oval).
Mean Motion	n	Shows changes in altitude or speed.
RAAN	Ω	Monitors the rotation of the orbit's plane.
Arg. of Perigee	ω	Detects rotation of the orbital path.
Mean Anomaly	M	Tracks timing deviations (arriving early/late).
Orbital Age	A	Helps judge drift based on how old the data is.

4.1 Why These Features Matter

Instead of analyzing all data points equally, the system groups these features to detect specific types of maneuvers:

- **Energy & Altitude (Mean Motion + Eccentricity):** These are the loudest indicators of movement. If a satellite fires its main thrusters to climb to a higher orbit, these two numbers will change instantly. This is often the first sign of a repositioning maneuver.
- **Plane Changes (Inclination + RAAN):** These variables describe the tilt of the orbital ring. Changing the angle of an orbit requires a massive amount of fuel. Therefore, any sudden shift in these values is a major red flag, often indicating a priority maneuver that is rarely seen during normal operations.
- **Phasing & Timing (Mean Anomaly):** This feature tracks exactly *when* a satellite arrives at a

specific point. If a satellite uses its thrusters to speed up or slow down—perhaps to synchronize its orbit with a target satellite—this value will show a drift that defies standard prediction.

5. Machine Learning: Detection & Scoring

These 7 features are then fed into the system's intelligence core. To accurately distinguish between benign drifting and suspicious movement, OrbitWatch uses three different detection models running in parallel.

5.1 Model A: Deep Autoencoder (Structure)

This model tries to compress the satellite data and then recreate it. If it fails to recreate the data accurately (high error), it means the satellite is moving in a way that violates the laws of physics the model learned during training.

```
// Model A Architecture
const model = tf.sequential();
model.add(tf.layers.dense({ units: 14, activation: 'tanh', inputShape:
[7] }));
model.add(tf.layers.dense({ units: 8, activation: 'relu' }));
model.add(tf.layers.dense({ units: 3, activation: 'relu' })); //
Compressed State
model.add(tf.layers.dense({ units: 8, activation: 'relu' }));
model.add(tf.layers.dense({ units: 14, activation: 'tanh' }));
model.add(tf.layers.dense({ units: 7, activation: 'linear' }));
```

5.2 Model B: Isolation Forest (Statistics)

This model sorts data points into groups. If a satellite is easy to separate from the rest of the group (it takes very few steps to isolate it), it is statistically rare and potentially suspicious.

```
// Recursive Path Logic for Isolation
private pathLength(instance: number[], node: IsolationTree,
currentPathLength: number): number {
    if (node.isExternal) return currentPathLength +
this.cFactor(node.size);
    if (instance[node.splitFeature] < node.splitValue) {
        return this.pathLength(instance, node.left!, currentPathLength
+ 1);
    } else {
        return this.pathLength(instance, node.right!,
```

```
currentPathLength + 1);
    }
}
```

5.3 Model C: k-Nearest Neighbors (Proximity)

This measures the distance between a satellite and its 5 closest neighbors in the database. If a satellite is "far away" from similar satellites in terms of behavior, it is considered an outlier.

```
// Euclidean Distance Calculation
const diff = this.referenceData!.sub(xNorm);
const squaredDiff = diff.square();
const sumSquaredDiff = squaredDiff.sum(1);
const distances = sumSquaredDiff.sqrt();
```

5.4 The Risk Score: Weighted Consensus

The final **Threat Score** is calculated by combining the outputs of all three models. This is not a simple average; it is a weighted calculation designed to prioritize *physical* behavior over *statistical* likelihood.

- **40% Weight - Deep Autoencoder (Physics First):** This model is given the highest authority because it learns the fundamental laws of motion for the constellation. A high error here means the satellite is physically defying its expected trajectory (e.g., a sudden thrust). This is the strongest indicator of a maneuver.
- **30% Weight - Isolation Forest (Statistical Check):** This acts as a filter. It identifies data that is just "rare" or "weird" compared to the rest of the catalog. While useful, statistical rarity can sometimes happen naturally, so it is weighted lower than the physics engine to prevent false alarms.
- **30% Weight - k-Nearest Neighbors (Context):** This checks for isolation in space. If a satellite leaves its "cluster" of neighbors, it triggers this model. However, since some satellites operate alone by design, this is treated as a supporting indicator rather than a primary alert.

Risk Levels: The system maps the final weighted score (0-100) to actionable status levels:

- **Critical (>90):** All three models agree that the satellite is anomalous.
- **High (70-90):** The Physics model (Autoencoder) and one other model have triggered.
    - **Moderate (45-70):** Indicates a split decision—often occurring when the physics model sees a deviation, but the satellite is still within a valid statistical cluster.

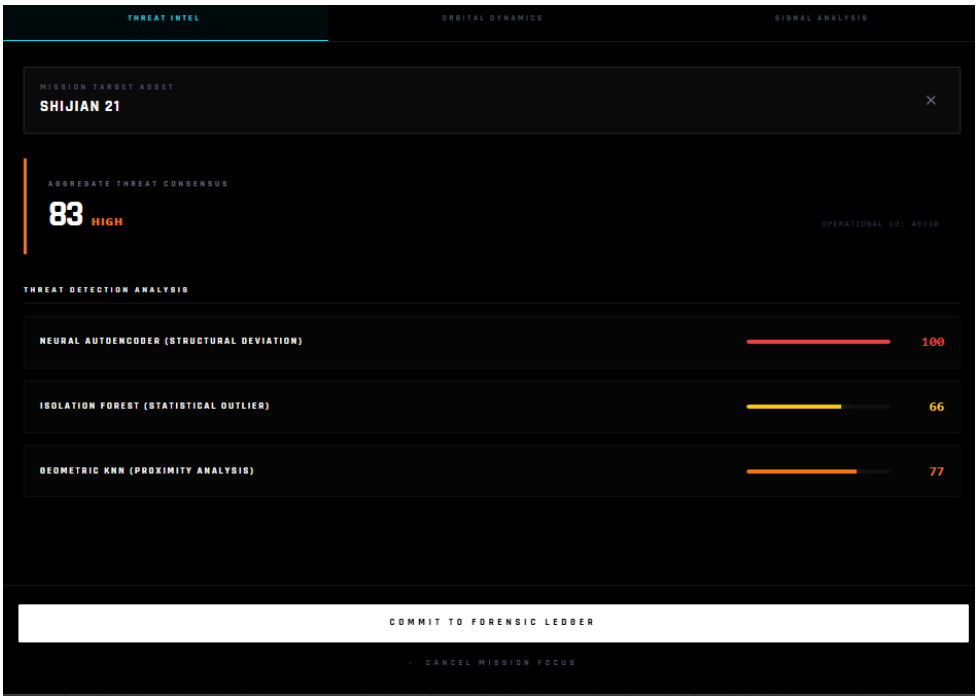

Figure 3: The Risk Score View

6. Command Interface: Dashboard Overview

While the Machine Learning engines run in the background, the Dashboard serves as the primary visual output for the operator. It aggregates all calculations into a single, real-time status view.

- **GLOBAL ASSETS:** The total count of active satellites currently loaded.
- **NOMINAL TRACKING:** The number of satellites behaving normally.
- **ANOMALY FEED:** A count of satellites with a risk score above 25%. This helps the analyst focus on objects showing unusual behavior.
- **THREAT MAX:** A text indicator showing the highest risk level currently detected in the system.

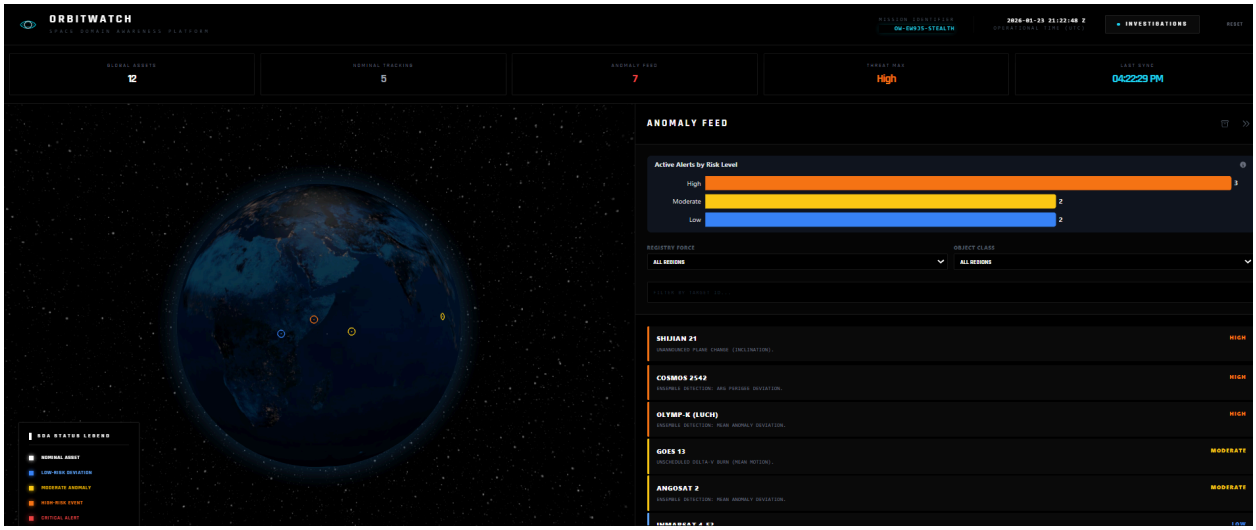

Figure 4: The Command Interface

7. Signal Analysis

In addition to tracking physical movement, OrbitWatch provides a dedicated module to analyze the radio frequency characteristics of a target.

7.1 Signal Data

- **Center Frequency:** Calculated using the Doppler Shift, which changes the frequency based on how fast the satellite is moving towards or away from the observer.
- **Noise Floor:** A baseline noise level is set. If the Risk Score is high, this floor is raised to simulate jamming (interference).
- **Peak Power:** Calculated using the Inverse Square Law, which states that signal strength drops as distance increases.
- **Lock Status:** Shows **LOCKED** when the system successfully tracks the signal.

7.2 The Signal Graph

The graph shows a bell curve centered on the frequency. In a normal state, the curve is sharp. If a threat is detected, the system adds noise to the graph to simulate electronic interference.

Figure 5: The Signal Analysis View

8. Tactical Case Management

Finally, once an anomaly has been detected and analyzed, it must be recorded. The **Investigations Board** serves as the system's "Black Box," allowing operators to create a permanent, tamper-evident log of suspicious events.

- **Trigger:** An investigation starts automatically when an operator clicks "ARCHIVE ASSESSMENT" on a specific satellite. This instantly snapshots the current physics and ML state.
- **Metadata Capture:** Each case is assigned a unique ID (UUID) and a precise timestamp. This ensures that every investigation can be individually tracked and referenced later.
- **User Annotation:** Operators can append detailed text notes to the case file. These notes are stored alongside the automated data, providing context on *why* the operator felt the event was significant (e.g., "Correlated with known launch event").
- **Lifecycle Management:** Cases can be toggled between **OPEN** (active monitoring) and **CLOSED** (resolved). This workflow allows teams to manage a queue of ongoing threats without losing track of past resolutions.
- **Persistence:** All case data is serialized and stored in the browser's LocalStorage. This ensures that the mission log survives browser restarts and page refreshes, providing long-term continuity.

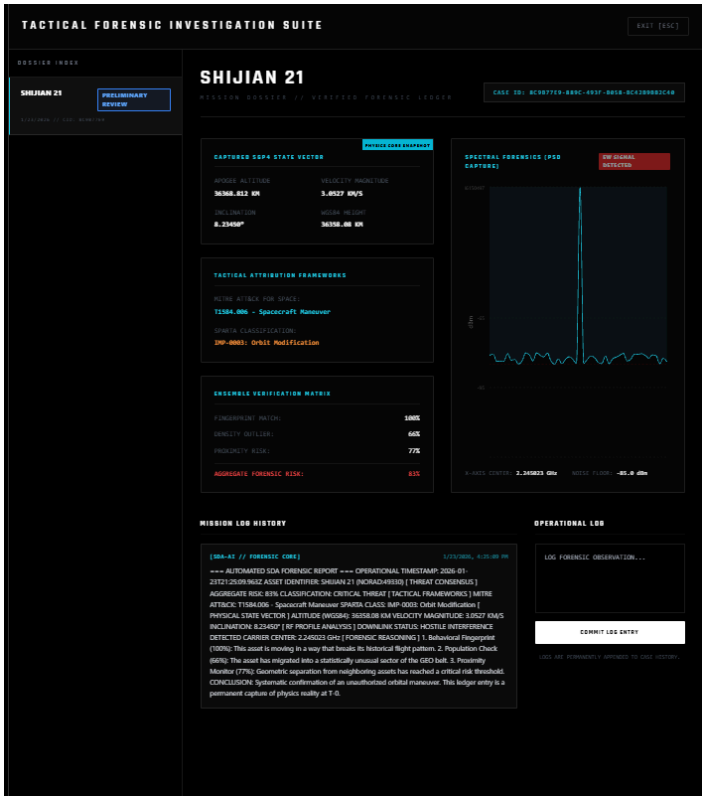

Figure 6: The Investigations Board

Conclusion

OrbitWatch proves that complex orbital analysis can be performed directly on a user's computer without relying on distant servers. This approach ensures that data processing is fast and independent of internet speed. By combining accurate physics models with smart detection systems, the platform gives operators a clear picture of satellite behavior. It moves beyond simple tracking to understand if a satellite is acting normally or suspiciously.

Furthermore, the system's ability to build a local history creates a unique strategic advantage. Unlike traditional tools that depend on external updates, OrbitWatch continues to learn and adapt even in isolation. This capability shifts the power of analysis from central data centers to the edge, ensuring that critical space monitoring can continue without interruption in disconnected or secure environments.