1. Description and scope

What the tool helps you explore

V2X Observation Lab is an open-source, browser-local MVP for automotive security education and evidence review. It asks what a receiving vehicle can establish from position messages, and what further evidence is needed before interpreting a result.

Three workspaces

- Observation Playground (/): compare synthetic true position, current position claims and the latest claim received by another vehicle. Adjust constant offsets, reception delay and message omissions.

- Learning Lab (/learn/): inspect two worlds with identical position messages, answer a question, then reveal their different true positions. Create and exchange lessons using a bounded template.

- Evaluation Inspector (/evaluate/): inspect supplied binary predictions and reference labels. Review individual errors and export a scoped evidence report. It is separate from the vehicle simulation.

Reading the scene

A is the sending vehicle. B is the receiving vehicle. A-prime is the position A claims, not another car. The purple "B remembers" pin is the position stored in the latest received message, not B itself. Simulation-wide comparison exposes synthetic ground truth to the analyst; Receiver-only excludes truth and unreceived claims.

Evidence boundaries

A constant translation cancels in position differences within an unchanged-offset segment. Matching changes therefore do not establish correct absolute position. Sampling and delayed information are observation limits, not automatic evidence of an attack. The tool does not run an AI detector, transmit RF, encode CAM/BSM packets, simulate collisions or certify safety. Included scenarios and prediction fixtures are authored synthetic examples.

License and development

The project code is MIT licensed. Retain the bundled Three.js license and third-party notices. The current release is a working MVP. Report reopening, compatible evaluation comparison and explicit case-to-scenario adapters are planned, not current features.

Source and contribution guidance

License

Updated changelog

Third-party notices

2. Installation and requirements

Use the hosted version

Open https://v2x.tcybertrust.id.vn/ in a modern browser. The interface and bundled examples use English. The overview video has US English narration, optional English captions and a text transcript. Nothing plays automatically.

Run a local copy

The site is static HTML, CSS and JavaScript, with Three.js for the 3D scene. Any static HTTP server can serve the web directory. The following example uses Git and Python 3; a downloaded GitHub ZIP can replace the clone step.

```
git clone https://github.com/thaocyber/V2X-Observation-Lab.git
cd V2X-Observation-Lab
python3 -m http.server 8095 --bind 127.0.0.1 --directory web
```

Open http://127.0.0.1:8095/ and keep the terminal running. Routes /learn/, /evaluate/ and /docs/ open the other workspaces and this guide. Stop the server with Ctrl+C. On Windows, use py -m http.server if python3 is unavailable. Do not open index.html using file://; module loading and browser security features require HTTP or HTTPS.

Platforms and browser capabilities

The static bundle is intended for Windows, macOS and Linux with a current browser supporting JavaScript modules, SVG, file input and Blob downloads. WebGL is needed for 3D; coordinate/message information remains available if 3D fails. The Inspector requires Web Crypto: use HTTPS or localhost. Desktop workflows have been checked in the development browser; a full browser, mobile-device and assistive-technology matrix has not been certified.

Offline operation and privacy

After downloading the complete bundle, the workspaces run without a remote API, account, model download or CDN. Imported lesson and evaluation files are processed in the browser, not uploaded. There is no autosave; download your lesson or report before leaving. Exports can contain user-supplied identifiers and notes, so review them before sharing.

Verify the source package

```
python3 scripts/verify_public.py
```

Verification requires Python 3.10+ and Node.js 20+. It checks the file manifest, local references, limited private-content patterns and deterministic implementation tests in an isolated copy. It is not a security certification or model benchmark. Media and PDF rebuilding have optional development dependencies; they are not required to use the downloaded site.

Public demo

Implementation checks

3. Using the Observation Playground

A short guided walkthrough

1. Open the homepage and choose "1 · Observation gap". At 6.5 s, true A is at x = -10.5 m while B retains x = -12 m from the message sent at 6 s. There is no offset in this example: a moving vehicle can outpace its last sample even with zero transmission delay.

2. Choose "2 · Offset, equal differences". At 10 s, A is at x = 0 m and its claim is x = 8 m. The latest two sent positions change by 3 m in both cases. Use the illustrated movement and the optional numerical details to compare these equal changes.

3. Switch Observation mode to "Receiver-only". Ground truth and unreceived claims disappear. Their absence is intentional: this view represents the information available to B.

4. Choose "3 · Late information". At 7 s the retained message was sent at 6 s and has just arrived after a 1 s delay. Its age since transmission is 1 s; its age since arrival is 0 s. These are different measures.

Parameters and update behavior

- Offset X / Offset Y: translation in metres, each from -20 to 20. The presets use a planar X offset; this is not GNSS signal spoofing.

- Start / End: simulation seconds defining the interval start <= time < end. The UI uses 0.5 s increments and requires end after start.

- Reception delay: fixed delay from 0 to 2 s. Omit every Nth message: 0 means none; the UI permits 0 through 10.

- Click "Apply & reset" after editing. Pending fields do not change the scene or exports. Applied settings reset time to zero. The timeline runs from 0 to 20 s, with one message per second; arrivals beyond 20 s remain outside playback.

Reproduce and interpret

Use "Download scenario" to save an importable configuration. Reopen it through advanced settings, then seek to the same time. "Download steps + expected results" records the selected moment. JSON reports include the full synthetic trace and ground truth for analysis; they do not grant the receiver extra knowledge. A trace or report is not a scenario-import file.

Within a constant-offset segment, differences match. At offset onset or removal, they may jump. Compare message identifiers and send times before comparing sender and receiver differences: omissions can make the pairs different. Data age is time since transmission, not a risk score or an IP TTL. Crossing teaching trajectories do not establish a collision.

Scenario schema

Report format

4. Lessons and evaluation evidence

Create a lesson

1. Open /learn/. Read the identical messages shown for B, choose an answer, then select "Reveal true positions". In the default example, world 2 has an 8 m offset while both worlds send the same positions.

2. Choose "Create your lesson". Edit the title, objective, question and explanation. Set the world-2 offset from 2 to 16 m and speed from 1 to 5 m/s. Select "Apply & try lesson".

3. Use "Download lesson JSON" to keep or share the exercise. "Open lesson" imports that lesson file. This template keeps receiver messages identical; it is not a general traffic editor and does not validate the author explanation.

Inspect evaluation results

1. Open /evaluate/. The fictional teaching sample has eight rows: TP = 2, TN = 3, FP = 1 and FN = 2. No model was executed. Read the scope and definition of class 1 before interpreting the results.

2. Download sample JSON as a schema example, or import your own authorized evaluation. Supply declared dataset, model, split, intendedUse, positiveClass, labelSource, thresholdPolicy and dataOrigin metadata. Each record needs a unique id plus numeric expected and predicted values of 0 or 1. The UI permits up to 10,000 rows and a 2 MiB file.

3. Use "Filter outcomes" to inspect errors. The sample FN filter shows two cases. Accuracy is (TP+TN)/N; precision is TP/(TP+FP); recall is TP/(TP+FN); F1 is 2TP/(2TP+FP+FN). A zero denominator is shown as N/A, not zero.

4. Add reviewer notes and select "Export evidence report". The report includes all validated input rows even when the table is filtered, derived metrics, assumptions, unknowns and notes. Export before leaving; no autosave is provided.

What an evidence report does not establish

Labels and provenance are importer declarations. SHA-256 identifies the normalized input, not authenticated origin; it does not cover reviewer notes. Metrics describe these rows, not deployment performance. Prediction rows are not mapped into 3D. Evidence-report reopening is not implemented: preserve the original evaluation input separately. Scenario, lesson, evaluation-input and evidence-report JSON files are distinct formats.

Troubleshooting and support

For a blank 3D view, check browser WebGL support or use numerical details. For Inspector hashing errors, use HTTPS or localhost. For import errors, start with the matching sample/schema and inspect field types and size. Report ordinary bugs through GitHub Issues; use private vulnerability reporting for security issues.

Evaluation schema and sample

Report a bug

Report a vulnerability privately