

September 23, 2026



FISSURE:

Tactical RF Operations and Situational Awareness with GNU Radio

GNU Radio Conference 2026

Christopher Poore
Lead Developer
Assured Information Security, Inc.
poorec@ainfosec.com

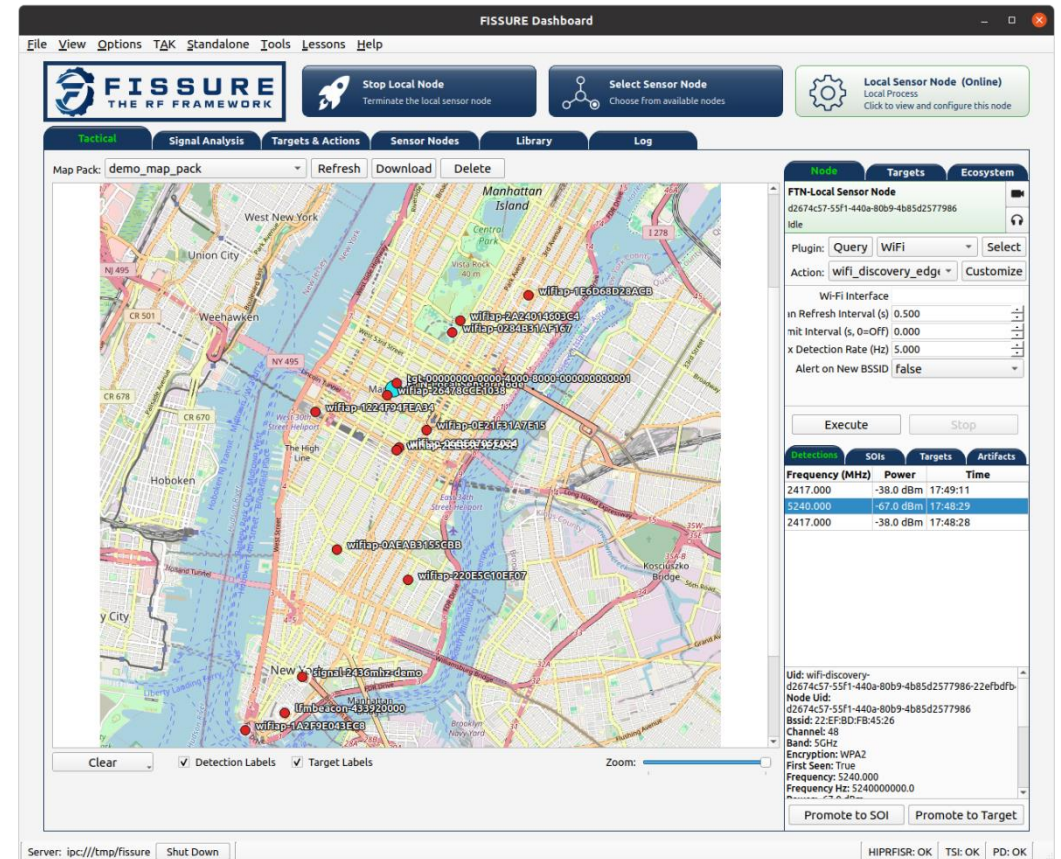
github.com/ainfosec/FISSURE



What is FISSURE?

An open-source framework for RF analysis, automation, and distributed operations

- Brings signal processing, targeting, geolocation, and situational awareness into one environment
- Coordinates SDRs, Sensor Nodes, software, and data across local and remote systems
- Uses GNU Radio extensively for RF processing while FISSURE handles the surrounding workflow
- Extensible through plugins, actions, and reusable operations



The Problem I am Trying to Solve

Open, flexible, affordable, and ready for what comes next

- Create an open platform for learning, experimentation, development, and integration
- Offer developers maximum flexibility while keeping operator workflows simple
- Span RF and cyber without locking FISSURE to one protocol, sensor, platform, or use case
- Build around automation, distributed systems, AI/ML, and affordable deployment



How We Got Here

From a collection of RF tools to connected operational workflows

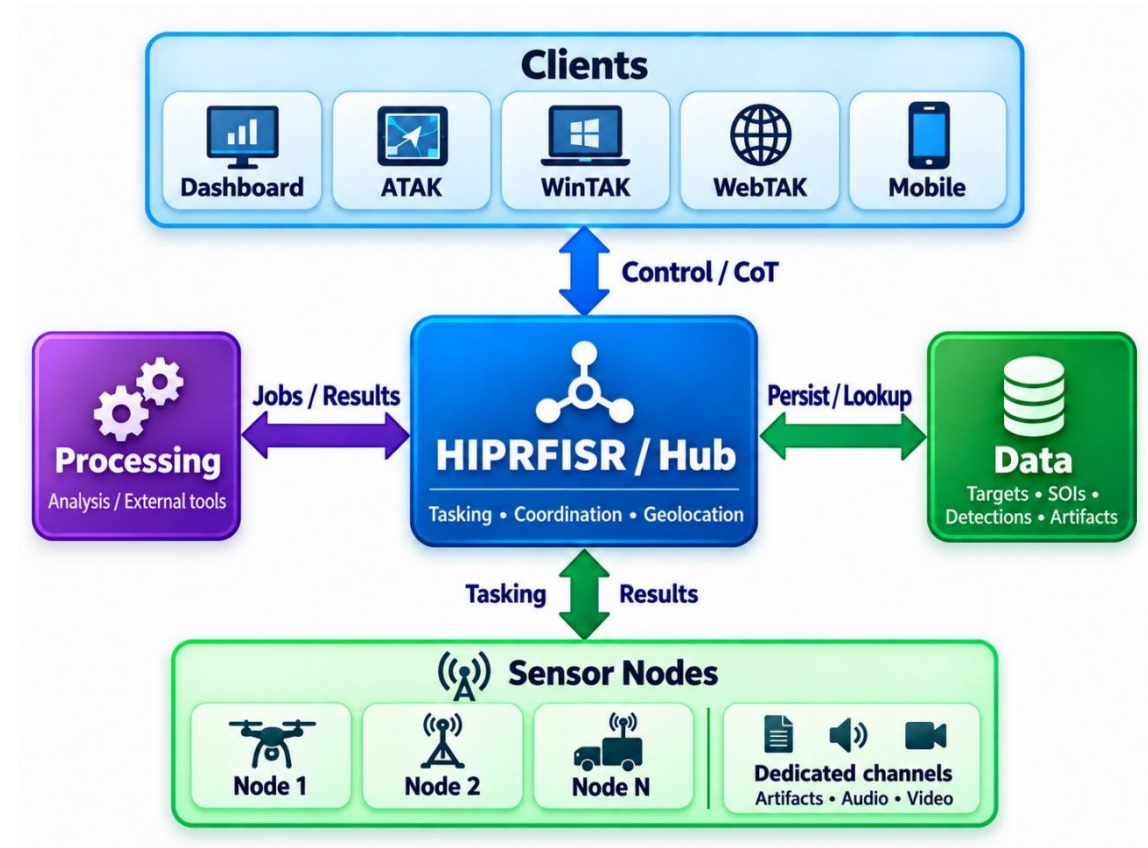
- Started as a broad RF/SDR toolbox with many independent capabilities
- Early development added remote Sensor Nodes, distributed execution, and tactical features
- Recent work has focused on connecting those pieces through clearer workflows, shared records, provenance, and automation
- The goal now is to make FISSURE a stronger bridge between RF hardware, analysis, operators, and emerging AI/ML capabilities



Distributed Architecture

A hub-and-node architecture connecting users, processing, sensors, and data

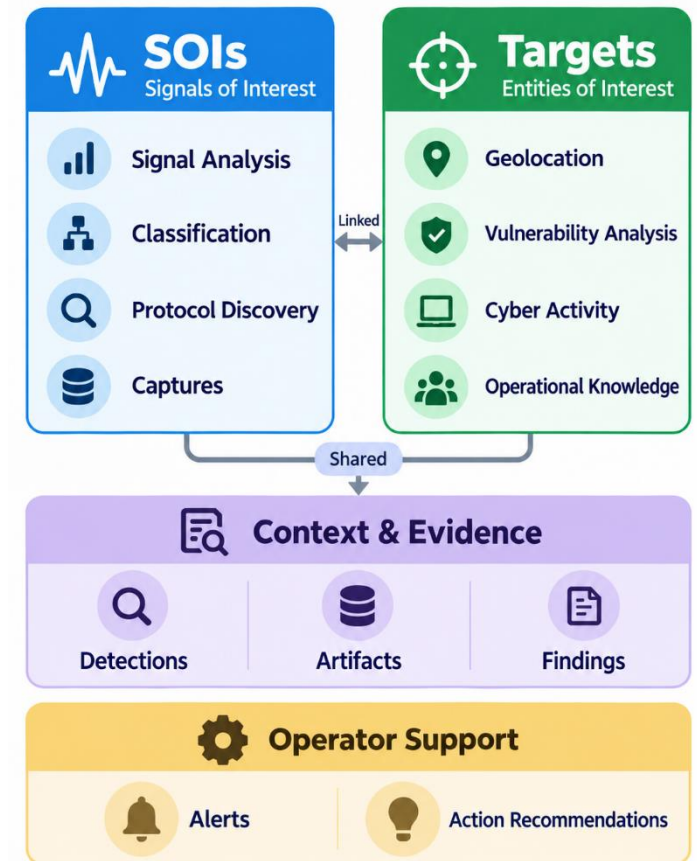
- Clients access FISSURE through the Dashboard or through external systems such as TAK
- HIPRFISR / Hub coordinates tasking, state, multi-node operations, geolocation, and persistent data
- Processing Engines support heavier analysis, network-enabled capabilities, and integration with existing solutions
- Sensor Nodes host SDRs, sensors, plugins, and other tools on distributed or platform-mounted systems



Core Information Model

Organizing signals, targets, and the context built around them

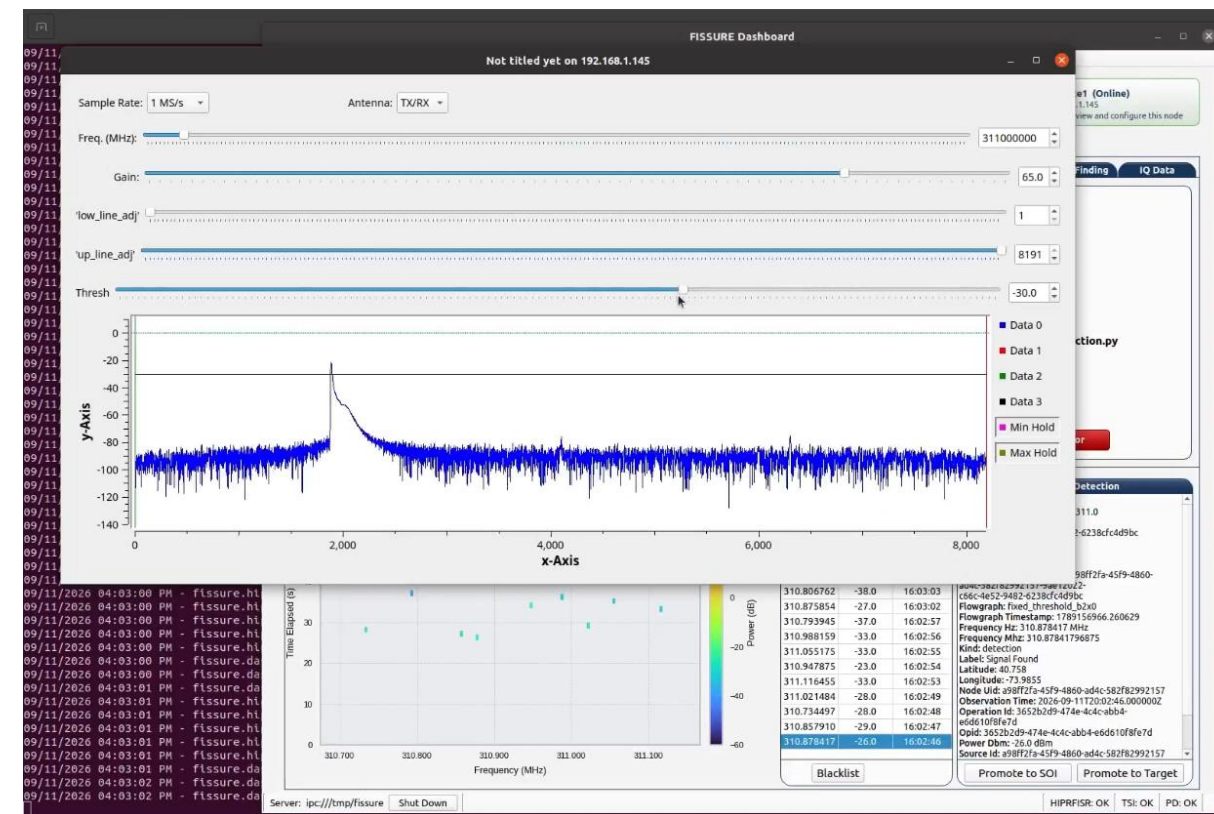
- **SOIs** anchor signal analysis, classification, protocol discovery, and capture
- **Targets** anchor geolocation, vulnerability analysis, cyber activity, and operational knowledge
- Detections provide sensor observations and can drive awareness, geolocation, alerts, and triggers
- Artifacts and Findings preserve evidence and analysis results, while Alerts and Action Recommendations surface what matters and what to do next



GNU Radio in the Workflow

Turning GNU Radio capabilities into local, remote, and automated operations

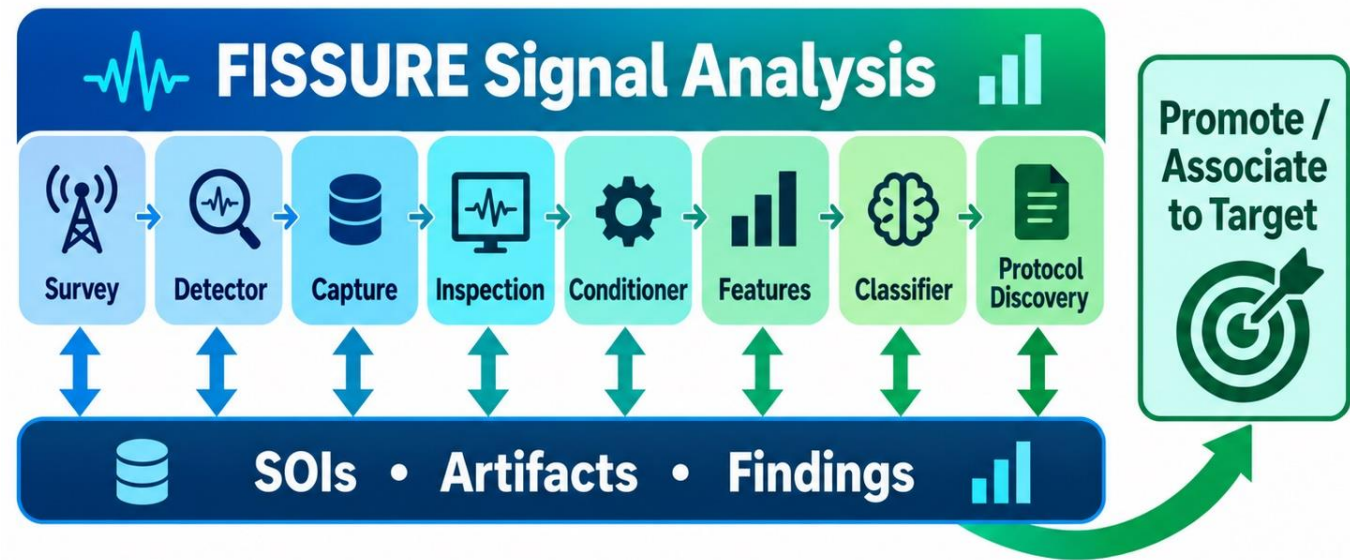
- GNU Radio supports detection, capture, analysis, transmission, and other RF tasks throughout FISSURE
- FISSURE supplies hardware, parameters, context, execution, and status, with values populated from SOIs, Targets, and other records when available
- Capabilities can run locally or on remote Sensor Nodes, with Actions supporting parameterized flow graphs, custom launch logic, and remote GUI streaming through Xpra



Signal Analysis as a Workflow

From discovery to understanding

- Survey and detection help identify signals worth investigating
- SOIs carry context through capture, inspection, conditioning, features, classification, and protocol discovery
- Artifacts and findings preserve the data and results generated along the way
- SOIs can be promoted to Targets when a signal becomes operationally relevant



Plugins and Deployable Capabilities

Add and manage capabilities without changing the FISSURE core

- Plugins can add GNU Radio, hardware, analysis, protocol, and other custom capabilities
- Plugins can manage setup, dependencies, repair, updates, and cleanup
- Capabilities can be deployed locally or to remote Sensor Nodes, while integrating third-party tools
- Mission-specific plugins can reuse Actions and keep proprietary capabilities outside the public FISSURE core

The image displays the FISSURE Dashboard interface and a code editor showing the setup.py file for the WiFi plugin.

FISSURE Dashboard:

- Top navigation: File, View, Options, TAK, Standalone, Tools, Lessons, Demo, Help.
- Buttons: Launch Local Node, Select Sensor Node, Remote1 (Online).
- Tabs: Tactical, Signal Analysis, Targets & Actions, Sensor Nodes, Library, Log.
- Sub-tabs: Activity, Autotun, File Navigation, Plugins.
- Plugin Inventory table:

Plugin	Hub Version	Node Version	Setup Status	State	Location
Base	1.0.0	1.0.0	Ready	Up to Date	Both (Hub & Node)
Dummy	1.0.0	1.0.0	Ready	Up to Date	Both (Hub & Node)
Mission-01	1.0.1	—	Not Deployed	Missing on Node	Hub Only
WiFi	1.0.0	1.0.0	Ready	Up to Date	Both (Hub & Node)
X10	1.0.5	—	Not Deployed	Missing on Node	Hub Only

Buttons: Deploy / Update, Repair Setup, Remove.

Operation Log:

```
11:18:07 Starting plugin removal for Mission-01...
11:18:07 Mission-01: Missing 0/1 was removed. No cleanup was declared.
11:18:07 Mission-01: Remove completed.
11:18:31 Starting plugin removal for WiFi...
11:18:31 WiFi: WiFi was removed. No cleanup was declared.
11:18:31 WiFi: Remove completed.
11:18:43 Starting plugin deployment for WiFi (1.0.0)...
11:18:45 WiFi: Full (1.0.0) deployed successfully. Existing setup is already
11:18:45 WiFi: Node reports SETUP_SETUP.
11:18:45 WiFi: WiFi requires no external setup.
11:18:46 WiFi: Deploy / Update completed.
```

Operation Status:

Current Operation: —
Status: Idle
Progress: 0%
Last Update: 11:18:46

WiFi:

Location: Both (Hub & Node)
Hub Version: 1.0.0
Node Version: 1.0.0
Setup Status: Ready
State: Up to Date
Hub Manifest: Present
Node Manifest: Present
Setup Hook: Present
Cleanup: Not Declared
Setup Check: Setup check passed.

Code Editor (setup.py):

```
#!/usr/bin/env python3
"""
FISSURE plugin setup hook template.

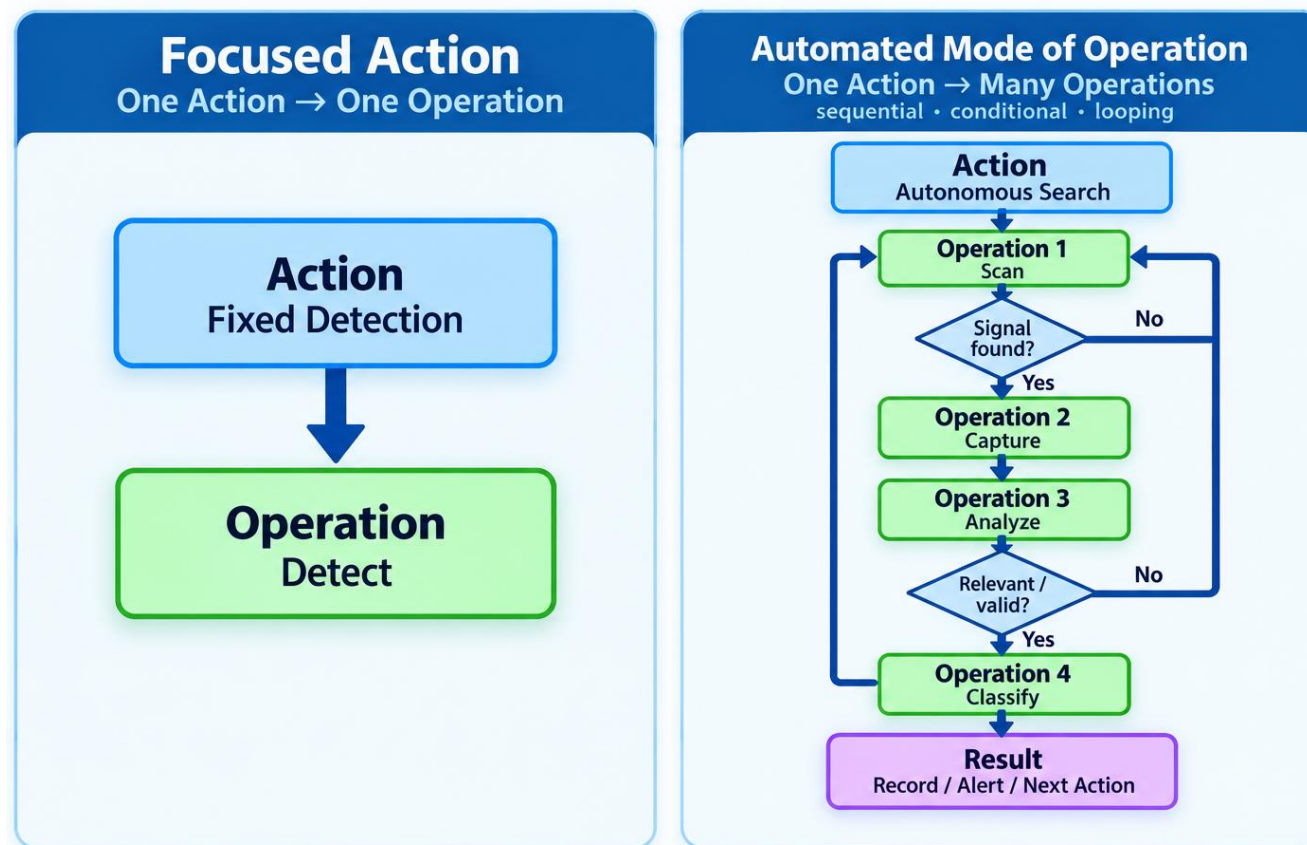
This file may be called by the FISSURE framework with
'check', 'install', or 'cleanup' to perform any plugin
specific setup or teardown. The WiFi plugin does not
require any external setup.
"""
import sys
PLUGIN_NAME = 'WiFi'
def check() -> bool:
    """Check if the plugin is ready to use."""
    print("WiFi plugin requires no external setup")
    return True
def install() -> bool:
    """Perform plugin installation steps."""
    print("WiFi plugin has no external setup to install")
    return True
def cleanup() -> bool:
    """Clean up plugin resources."""
    print("WiFi plugin has no plugin-owned setup resources")
    return True
```



Actions and Operations

From focused one-off tasks to automated modes of operation

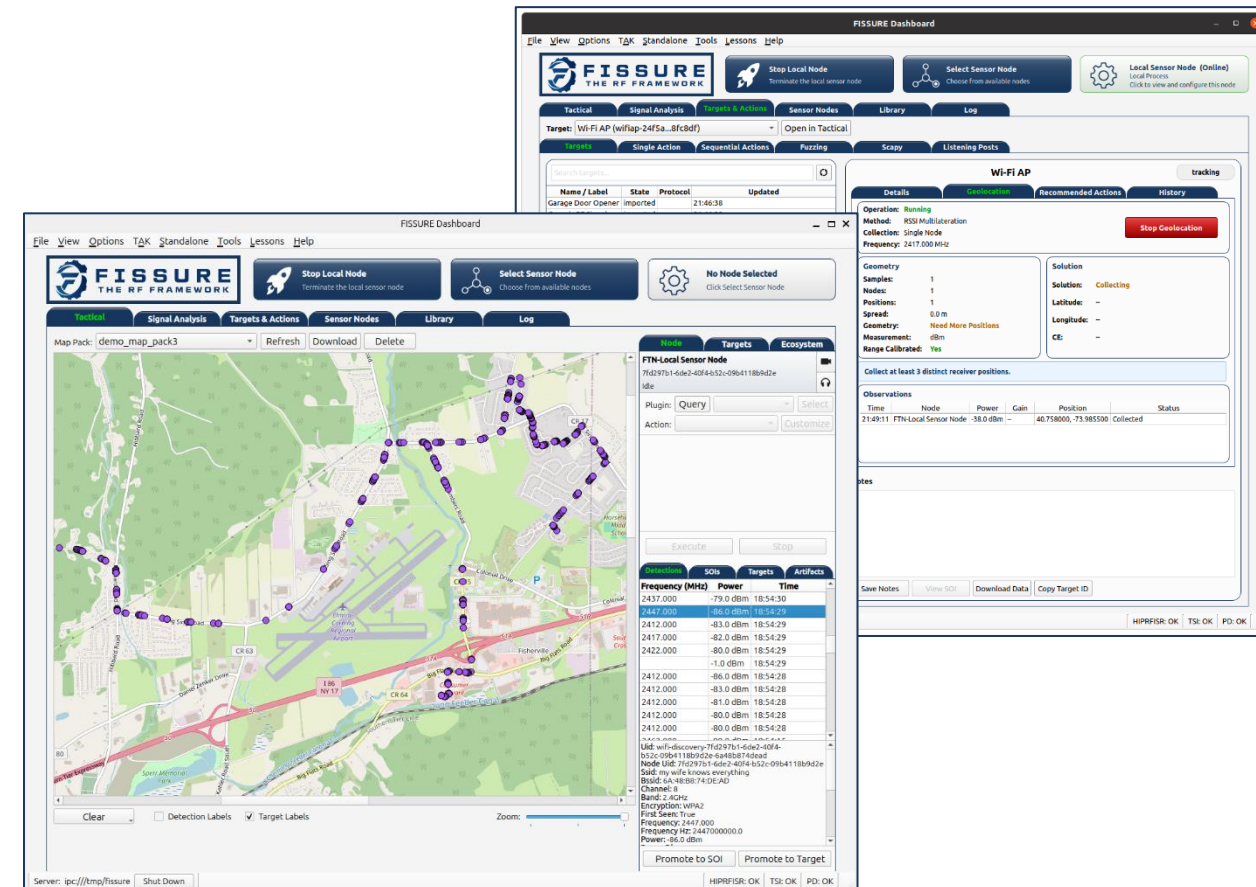
- Operations are the programs that do the work, from GNU Radio and Python to external applications
- Actions expose and configure those Operations for FISSURE, including inputs, parameters, context, and execution
- An Action can call one Operation or orchestrate many sequentially, in parallel, or conditionally, supporting both focused tasks and automated modes of operation



Geolocation and Target Awareness

Turning distributed observations into persistent operational context

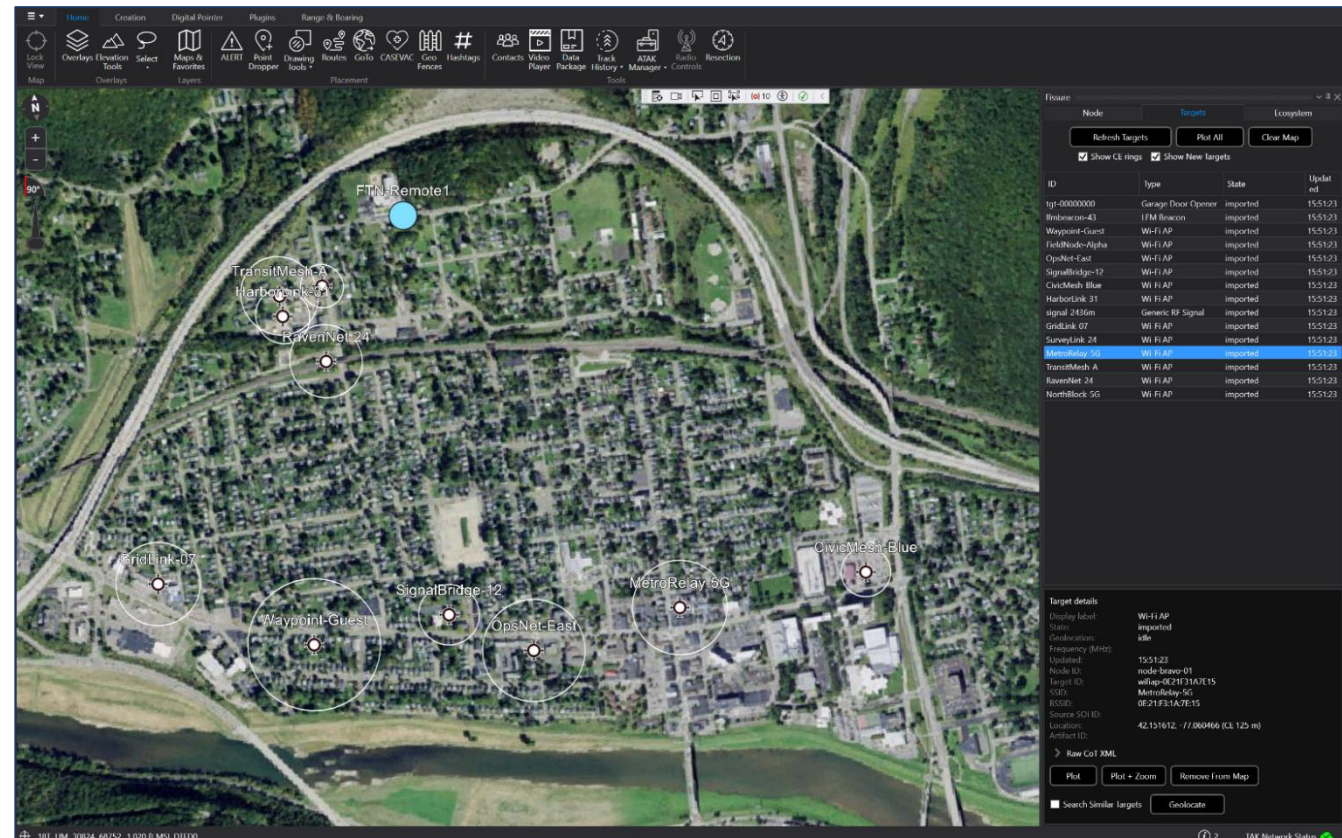
- Targets accumulate signals, locations, observations, and other knowledge about an entity
- Sensor Nodes contribute detections and measurements for geolocation and tracking
- HIPRFISR coordinates multi-node geolocation and feeds results back into Target and mapping workflows
- Target context can recommend or pre-populate relevant Actions, while new observations continue building the record



FISSURE + TAK

Sharing RF awareness through CoT while keeping FISSURE independent

- FISSURE sends and receives CoT for Sensor Nodes, Targets, detections, geolocation, alerts, tracks, and events
- ATAK and WinTAK extend RF-derived awareness to users outside the FISSURE Dashboard, including resources such as remote video
- FISSURE can also provide its own operational map when a TAK client is not needed or available



A Partial FISSURE Workflow (Video)

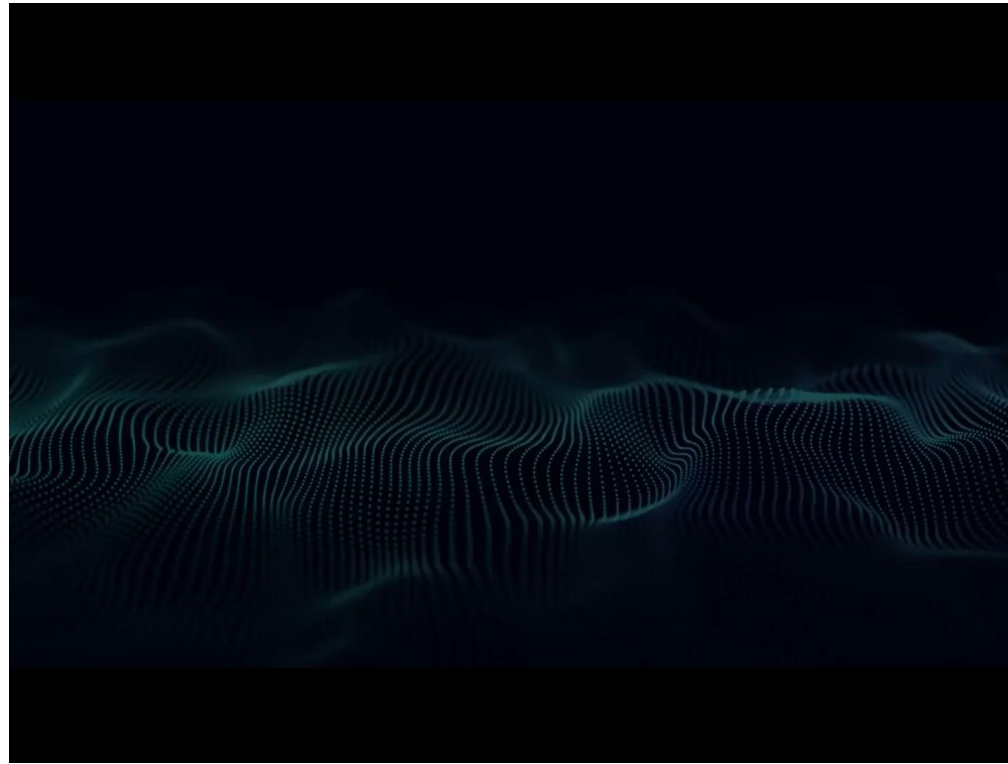
From tactical context to initial signal analysis

- Tactical View
 - Provides an operator-focused view of Sensor Nodes, detections, SOIs, and targets
 - Connects distributed RF activity with persistent operational context
- Signal Analysis Workflow
 - Moves from RF discovery and detection into capture and analysis
 - Associates collected data, artifacts, and findings back to SOIs
- Distributed Operations
 - Capabilities execute across local and remote Sensor Nodes
 - Collected information can move between operational and analysis workflows



A Partial FISSURE Workflow (Video)

From tactical context to initial signal analysis



Fracture

Turning FISSURE capabilities into deployable systems

- Fracture packages FISSURE into purpose-built hardware and software configurations
- SDRs, compute, networking, plugins, and configuration are combined around a mission or use case
- The goal is easier deployment, integration, support, and sustainment
- Fracture can include deployment-specific integrations, including ATAK and WinTAK plugins not available in the public FISSURE release



Where FISSURE Goes Next

Expanding capability, automation, AI integration, and operational maturity

- Complete end-to-end workflows from discovery through sharing
- Expand the plugin ecosystem across protocols, hardware, sensors, analysis, and integrations
- Strengthen automation and provenance for repeatable, traceable workflows
- Connect RF hardware and structured data more directly to AI/ML while hardening distributed deployment and adoption



How You Can Help

There are plenty of ways to contribute

- **Use it:** Try FISSURE with your own hardware, signals, workflows, and problems
- **Improve it:** Report bugs, rough edges, missing capabilities, and workflow issues
- **Build with it:** Contribute code, plugins, documentation, testing, hardware support, or existing RF/Cyber capabilities
- **Grow it:** Share FISSURE with coworkers, students, professors, labs, and teams, or get involved more deeply in shaping the project



Questions?

Chris Poore

Lead Developer

`poorec@ainfosec.com`

Assured Information Security, Inc.

