Outline

• Summary
• Origins
• Core Principles
• Component Details
• GNU Radio Integration
• Roadmap
• Contributing
Summary

- **Open-source** RF and reverse engineering framework
- Contains **hooks** for detection, classification, protocol discovery, attack execution, vulnerability analysis, automation, AI/ML
- Consolidates **all-things RF**: software modules, radios, protocols, signal data, scripts, flow graphs, reference material, and third-party tools
- Speeds up the **characterization of signals** and the **identification of vulnerabilities** in RF protocols, waveforms, and devices
- Mostly **Python** & PyQt with support for legacy systems
- Out-of-the-box, **pain-free** software installer
- Meant for everyone: **experts** and **beginners**, edit pieces on your own
FISSURE – The RF Framework

Summary

• On GitHub since August 10, 2022
• Promoted at DEF CON Demo Labs August 12, 2022

• You may have seen:
  ◦ Griffiss Institute Lecture + Education Series April 28, 2021
  ◦ 2021 C4I and Cyber Technology Conference August 3, 2021

• Learn more at:
  ◦ Twitter (@FissureRF)
  ◦ Discord (see GitHub)
  ◦ chat.gnuradio.org (cpoore1)
  ◦ Reddit (r/FISSURE)
  ◦ ainfosec.com (future)
Origins

Background & Context

• Electrical engineer at a cybersecurity company
  ◦ Assured Information Security, Inc. (HQ: Rome, NY)
    • Advanced Research, Cyber Operations, Intelligence Analysis, Security Testing, Trusted Systems
    • Artificial Intelligence, Machine Learning, Computer Architectures, Behavioral Science, Software and Malware Analysis, Virtualization, Cross Domain Solutions, Reverse Engineering, Embedded Systems, Penetration Testing, and more

• Worked on RF projects entire career
  ◦ Constantly jumping around to different technologies with each project
  ◦ Always something/somewhere new

• I need to know everything about the different RF technologies to:
  ◦ Characterize systems
  ◦ Assess security
  ◦ Exploit and interact with targets
  ◦ Perform research
  ◦ Develop tools
  ◦ Teach others
Origins

- **What does FISSURE mean?**
  - Frequency Independent SDR-based Signal Understanding and Reverse Engineering

- **Where did it come from?**
  - 2014 project for designing a flexible system for automatic RF device assessment
  - Internal research and development over several years

- **What does it look like?**
  - PyQt GUI with many tabs and menu items
  - Dedicated Python components communicating to a central hub over ZMQ
  - YAML schema for input sanitization and error handling

- **Why extend it to the Open-Source community?**
  - Makes all our jobs easier
  - Brings in outside knowledge
  - Benefit future generations
  - Revolutionize engineering and cybersecurity
Principles

The Fundamental Elements of FISSURE

- Speed up signal characterization
- Support rapid integration of existing tools and algorithms
- Provide flexibility to support new features
- Consolidate useful software and information
- Access commonly performed operations
- Allow data to be passed over a network and between components
- Contain transparent, easy-to-understand code

- Help with identification of vulnerabilities in protocols and devices
- Utilize commercial SDRs and other commonplace hardware
- Be a testbed for AI/ML and automation
- Promote RF and Cyber in education
- Easy-to-use, helpful visualizations
- Simple and reliable installation
- Support the latest and legacy
Principles

FISSURE is a Framework

• Not a finished product, it will look different over time
• Filled with examples of how to do things
• Everything can be improved, nothing is complete, and more can be integrated
• Requires feedback from the community
• Will adapt to help as many as possible
• Needs to be built up before automation can have a larger role
• Meant for everyone
  ◦ Experts can expose cutting-edge solutions
  ◦ Professionals can perform their daily tasks
  ◦ Educators can teach
  ◦ Students, hobbyists, developers can learn
Components

- Target Signal Identification
- Protocol Discovery
- Attacks
- IQ Manipulation
- Online Signal Archive
- Packet Crafting
- Third-Party Tools
- Lessons
Components

Target Signal Identification

- Detector
  - Fast-scanning, slow-scanning
  - Power, frequency, time

- Signal Conditioner
  - Isolate, condition signals from a stream of raw I/Q

- Feature Extractor
  - Extract predetermined list of signal characteristics dependent on AI/ML classification method

- Protocol/Emitter Classifier
  - Interpret feature sets and provide confidence levels for protocol and emitter classification

- Future
  - hackrf_sweep, rtl_power fast-scanning detectors
  - Swap and compare AI/ML techniques, automation
Components

Protocol Discovery

- Recursive Demodulation
  - Load flow graphs, extract signal parameters, work towards a bitstream
- Bit Slicing
  - Identify patterns in a circular buffer filled with data to isolate fields and add messages to the library
- Data Viewer
  - Manipulate bits, view hex, compare to known protocol & message formats
- Custom Dissectors
  - Create Lua Wireshark dissectors and view/record messages returned from demodulation flow graphs
- CRC Calculator
  - Apply common CRC algorithms
  - Find the CRC polynomial from two messages with known CRCs
- Future
  - Parameter acquisition, protocol confidence levels, pattern recognition, variable length messages
Components

Attacks

- Single-stage
  - Python2/Python3 scripts
    - Simple header added to the file for default values
  - Flow Graphs with/without GUIs
  - Wireless or wired applications

- Multi-stage
  - String multiple attacks in succession
  - Run each one on a loop for a specified duration
Components

Attacks (Continued)

• Data field fuzzing
  ◦ Check Fields
  ◦ Choose random or sequential fuzzing
  ◦ Specify ranges for fuzzing
  ◦ Enter a transmit interval
  ◦ Automatically recalculates CRCs

• Flow graph variable fuzzing
  ◦ Fuzz individual GNU Radio variables for blocks with callbacks

• Future
  ◦ More attacks
  ◦ Further hardware support
  ◦ Vulnerability analysis
 Components

IQ Manipulation

- Live inspection flow graphs
- Record and playback
- View data
  - Plot, zoom, pan, save, measure
- Modify data
  - Crop, convert, append, apply timeslots, overlap, resample, OFDM analysis, normalize
- Perform analysis
  - Magnitude, instantaneous frequency, spectrogram, FFT, moving average, morse code, polar plot
- Future
  - Time/frequency measurement, obtaining symbol rates
  - Radar data analysis
  - Filtering
  - Better inspection flow graphs
Components

Online Signal Archive

- Download online files
- Create playlists to simulate traffic and test systems
- Future
  - Standardized metadata format: SigMF
  - Create data sets, collections
  - Import from other sources
  - Build playlists of data sets
  - Save/load playlists
Components

Packet Crafting
- Assemble custom packets for protocols in library
- Calculate CRC values
- Construct sequences of messages
- Scapy integration for Wi-Fi
- Future
  - More protocols and packet types
  - Quick links to attacks with file sources
Components

Third-Party Tools

• Standalone flow graphs
  ◦ Favorites that can be edited
  ◦ Separate from the rest of FISSURE

• Third-party tools included with the install
  ◦ Launch directly from the menu
  ◦ Open a terminal with example commands

• Online tools and reference material
  ◦ Maps, calculators, databases, etc.

• Future
  ◦ More protocols, tools
  ◦ Cleaner, more organized installer
  ◦ Docker alternative
Components

Lessons

• Instructions on how the technology, protocols, and tools work
• Tie the lessons and steps into FISSURE
• Topics like:
  ◦ OpenBTS
  ◦ Lua dissectors
  ◦ Sound eXchange
  ◦ ESP boards
  ◦ Radiosonde tracking
  ◦ RFID
  ◦ Data types
  ◦ Custom GNU Radio blocks
  ◦ TPMS
  ◦ Ham Radio Exams
  ◦ Wi-Fi Attacks

• Future
  ◦ More topics, updates to existing topics
  ◦ Colleges, High schools
  ◦ Clubs, Workshops
  ◦ Hacking/Cyber/RF events
GNU Radio Integration

What is the Role of GNU Radio?

- Running several types of flow graphs, passing data in different ways
  - Detection
  - Inspection
  - Protocol discovery
  - Demodulation
  - Sniffing
  - Attacks
  - Fuzzing

- Changing variable values, loading flow graphs
  - Before runtime
  - While running
  - Running flow graphs with and without GUIs from Python

- Support for 3.7, 3.8, and 3.10 (as separate branches)

- Out-of-Tree modules are submodules pulled from repos
  - Will need to be monitored

- Looking for better and additional ways of using GNU Radio
What’s Next?

- The open source governance model has room to evolve
  - Founder-leader > Corporate-backed > Do-ocracy
- Continuing the push for funding avenues
- Establishing more ties with education
- Releasing updated documentation and videos
  - At AIS domain (ainfosec.com)
- Improving existing software
  - Cleaning code, removing bugs, testing more SDRs, expanding lessons
  - Detection, signal conditioning, feature extractor, protocol/emitter classifier
  - Sensor node deployment scheme
- Adding new pieces
  - Not re-inventing the wheel
  - Recursive demodulation, protocol discovery, tracking, vulnerability analysis
Contributing

What can you do?

- **Showing** your interest is vital
  - Makes for an easier sell to internal/external customers
  - Star the project on GitHub, join the Discord server, follow on Twitter
  - Contact the developers/AIS

- **Contributions** strengthen the software and saves development time

- **Suggestions** focus the updates and help others who feel the same way
  - Software tools, hardware suggestions, IQ analysis algorithms, attacks scripts, new operating systems, bugs, improvements
  - New tabs, components, features

- **Collaborate** with AIS
  - Speeds up FISSURE development and can aid your project at the same time

- **Submit a resume** to AIS for full-time employment
Any Questions

Chris Poore
Senior Reverse Engineer
poorec@ainfosec.com
Assured Information Security, Inc.

https://github.com/ainfosec/FISSURE