Unified RF Fuzzing Under a Common API: Introducing TumbleRF

Matt Knight, Ryan Speers

March 15, 2018

River Loop Security

Matt Knight

- Independent software, hardware, and RF engineer
- Security Researcher at River Loop Security
- BE in EE from Dartmouth College
- RF, SDR, PHYs, and embedded systems

Ryan Speers

- Director of Research at Ionic Security
- Co-founder at River Loop Security
- Computer Science from Dartmouth College
- Cryptography, embedded systems, IEEE 802.15.4

River Loop Security

Background

"Making and Breaking a Wireless IDS", Troopers14

"Speaking the Local Dialect", ACM WiSec

- Ryan Speers, Sergey Bratus, Javier Vazquez, Ray Jenkins, bx, Travis Goodspeed, and David Dowd
- Idiosyncrasies in PHY implementations

Mechanisms for automating:

- RF fuzzing
- Bug discovery
- PHY FSM fingerprint generation

Agenda

1. Overview of traditional fuzzing techniques (software and networks)
    1. How these do and don't easily map to RF
2. RF fuzzing overview and state of the art
3. Ideal fuzzer design
4. TumbleRF introduction and overview
5. TumbleRF usage example

Traditional Fuzzing Techniques

What is fuzzing?

Measured application of pseudorandom input to a system

Why fuzz?

- Automates discovery of crashes, corner cases, bugs, etc.
- Unexpected input → unexpected state

What can one fuzz?

Interfaces

- I/O
- File format parsers
- Network interfaces

Software Fuzzing State of the Art

Abundant fully-featured software fuzzers

- AFL / AFL-Unicorn
- Peach
- Scapy

Software is easy to instrument and hook at every level

What else can one fuzz?

Fuzzing Hardware

Challenges:

- H/W is often unique, less "standard interfaces" to measure on
- May not be able to simulate well in a test harness

Some Existing Techniques:

- AFL-Unicorn: simulate firmware in Unicorn to fuzz
- Bus Pirate: permutes pinouts and data rates to discover digital buses
- JTAGulator: permutes pinouts that could match unlocked JTAG
- …

Fuzzing RF

WiFuzz

- MAC-focused 802.11 protocol fuzzer

Marc Newlin's Mousejack research

- Injected fuzzed RF packets at nRF24 HID dongles while looking for USB output

isotope:

- IEEE 802.15.4 PHY fuzzer

Existing RF Fuzzing Limitations

Fuzzers are siloed / protocol-specific

Generally limited to MAC layer and up

RF is hard to instrument – what constitutes a crash / bug / etc?

Implicit trust in chipset – one can only see what one's radio tells you is happening

Trust and Physical Layer Vulnerabilities

Not all PHY state machines are created equal!

Radio chipsets implement RF state machines *differently*

- Differences can be fingerprinted and exploited
- Initial results on 802.15.4 were profound
- Specially-crafted PHYs can target certain chipsets while avoiding others

RF PHYs: A Primer

How Radios Work

Transmitter: digital data (bits) → analog RF energy

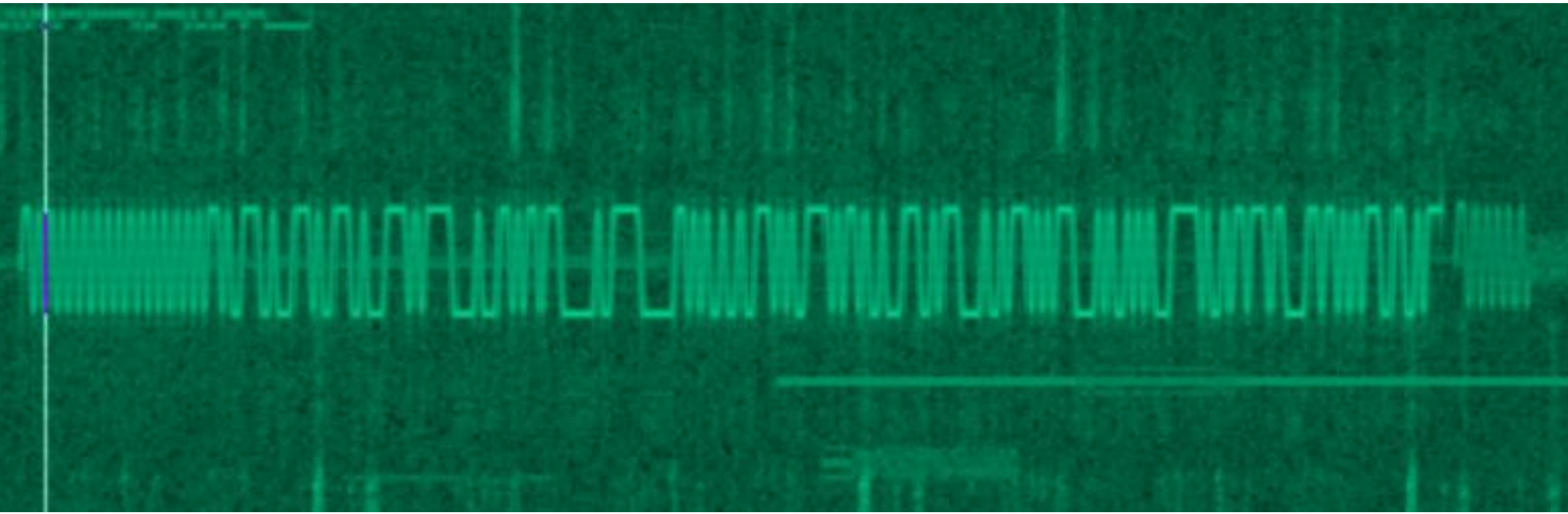
discrete → continuous

Receiver: analog RF energy → digital data (bits)

continuous → discrete

Receiving comes down to sampling and synchronization!

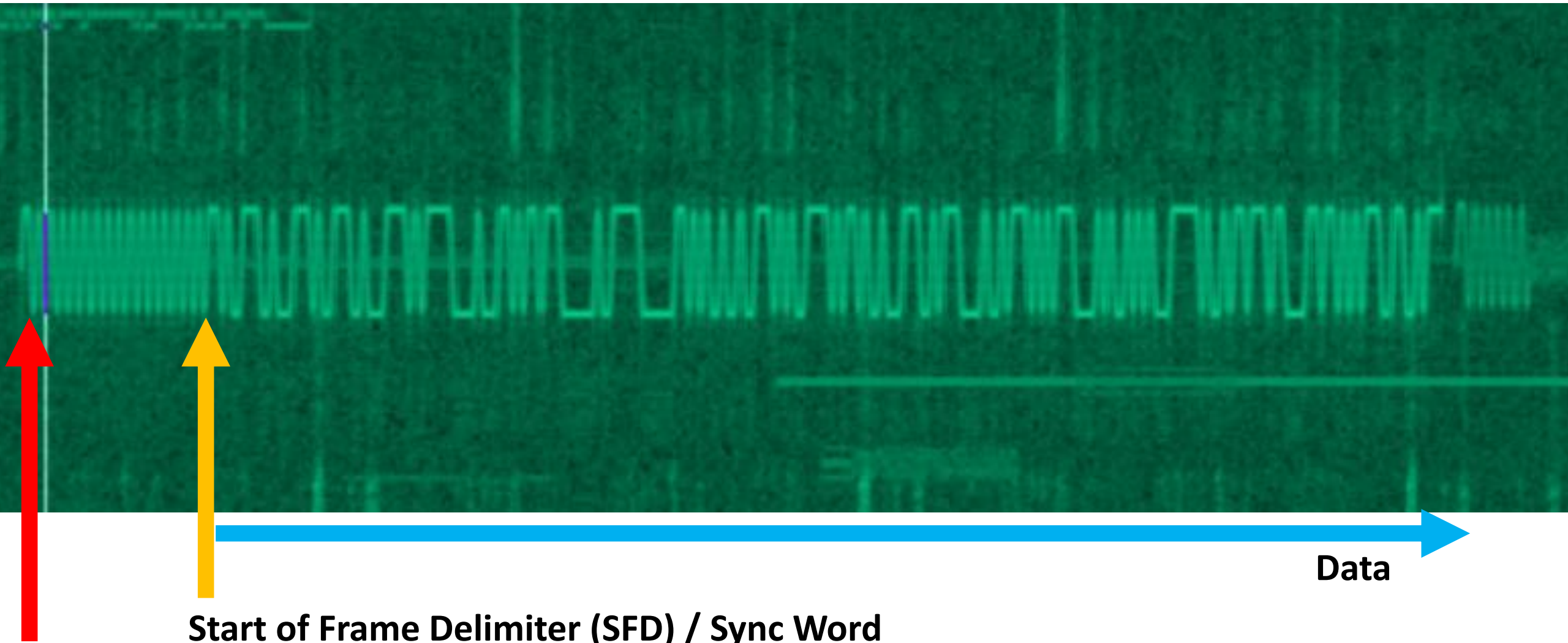
Digitally Modulated Waveforms



River Loop Security

https://hackaday.com/2016/11/18/building-a-lora-phy-with-sdr/

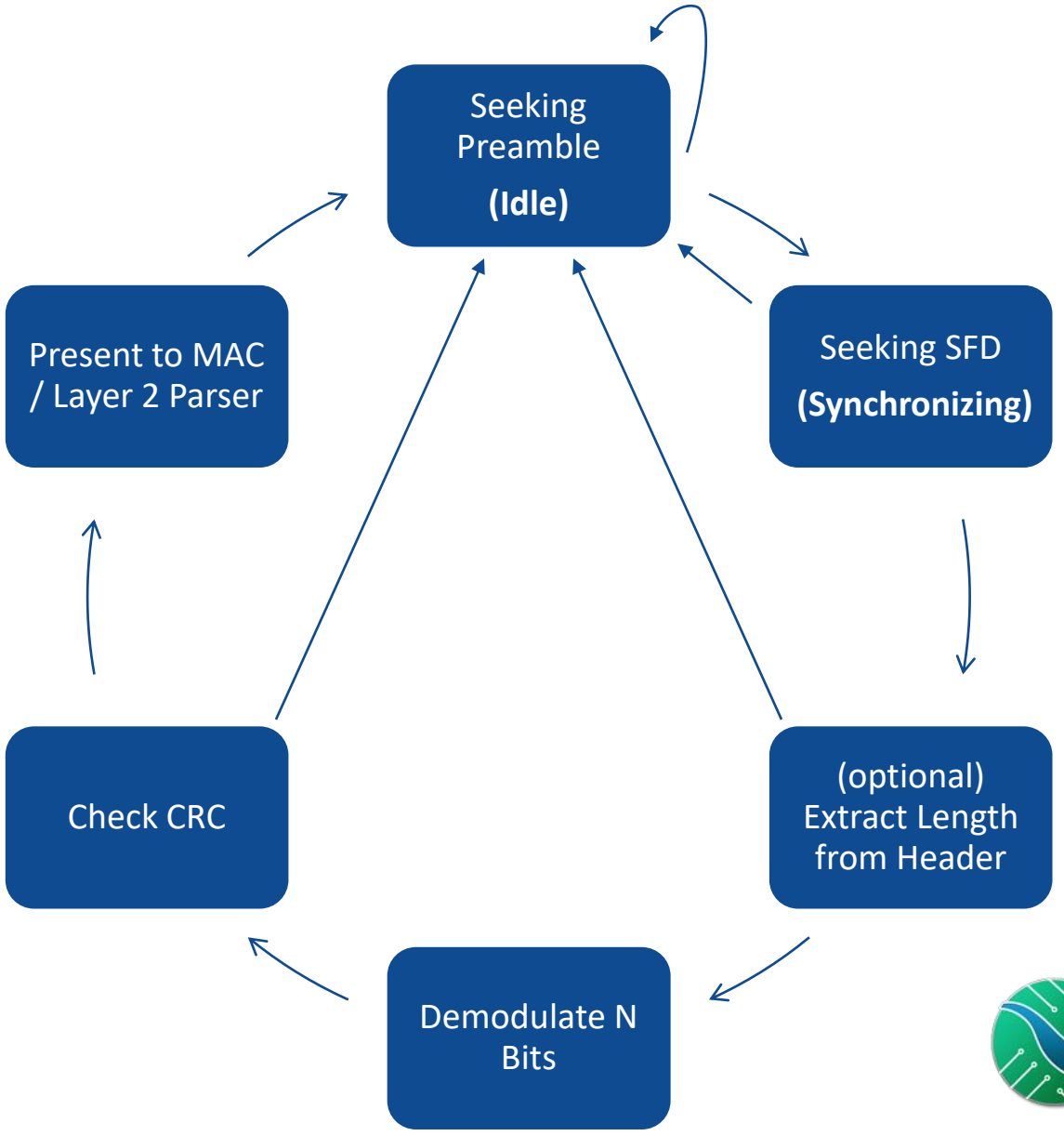
Digitally Modulated Waveforms



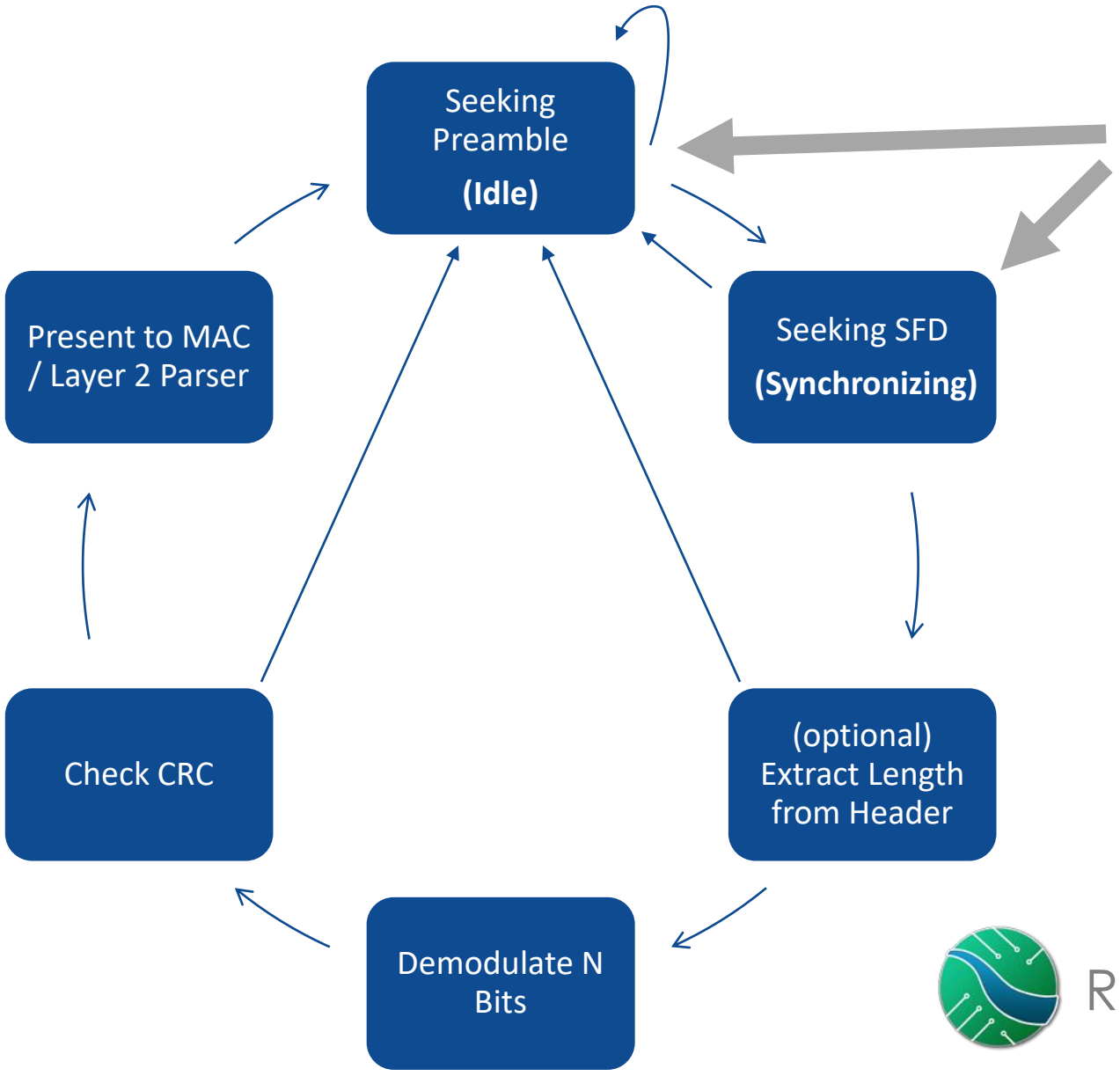
River Loop Security

https://hackaday.com/2016/11/18/building-a-lora-phy-with-sdr/

RF PHY State Machines



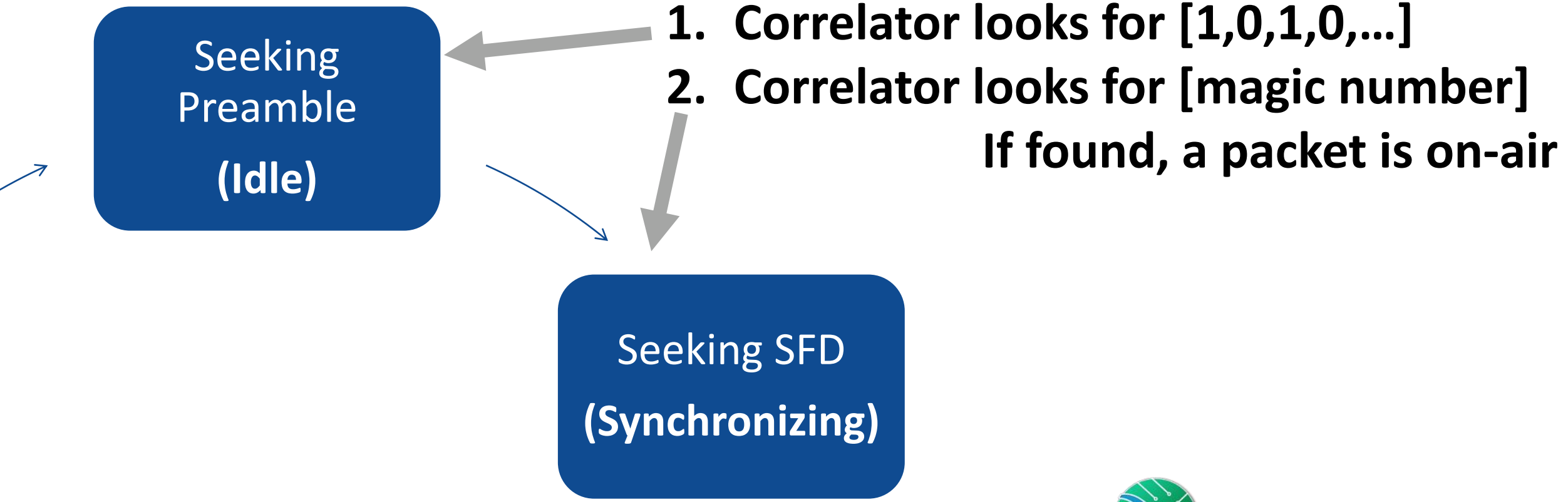
RF PHY State Machines



Let's dig in

RF PHY State Machines

Correlation = shift register clocking bits through at symbol rate looking for a pattern



1. **Correlator looks for [1,0,1,0,…]**
2. **Correlator looks for [magic number]**

If found, a packet is on-air

Sync Words and Magic Numbers

Turns out not all sync words are created equally

- 0x00000000 == 802.15.4 Preamble
- 0xA7 == 802.15.4 Sync Word

The isotope research showed some chipsets correlated on “different” preambles / sync words than others

Sync Words and Magic Numbers

Turns out not all sync words are created equally

- 0x00000000 == 802.15.4 Preamble
- 0xA7 == 802.15.4 Sync Word

strategically malformed

The isotope research showed some chipsets correlated on ~~"different"~~ preambles / sync words than others

Sync Words and Magic Numbers

Turns out not all sync words are created equally

- 0x**XXXX**0000 == 802.15.4 Preamble
- 0xA7 == 802.15.4 Sync Word

strategically malformed

The isotope research showed some chipsets correlated on ~~"different"~~ preambles / sync words than others

Short preamble?

Sync Words and Magic Numbers

Turns out not all sync words are created equally

- 0x**XXXX**0000 == 802.15.4 Preamble
- 0xA**F** == 802.15.4 Sync Word

strategically malformed

The isotope research showed some chipsets correlated on ~~"different"~~ preambles / sync words than others

Short preamble? Flipped bits in SFD?

River Loop Security

Fuzzing Shows the Way

Ideal RF Fuzzer Design

Ideal Features

Extensible: easy to hook up new radios

Flexible: modular to enable plugging and playing different engines / interfaces / test cases

Reusable: re-use designs from one protocol on another

Comprehensive: exposes PHY in addition to MAC

TumbleRF

TumbleRF

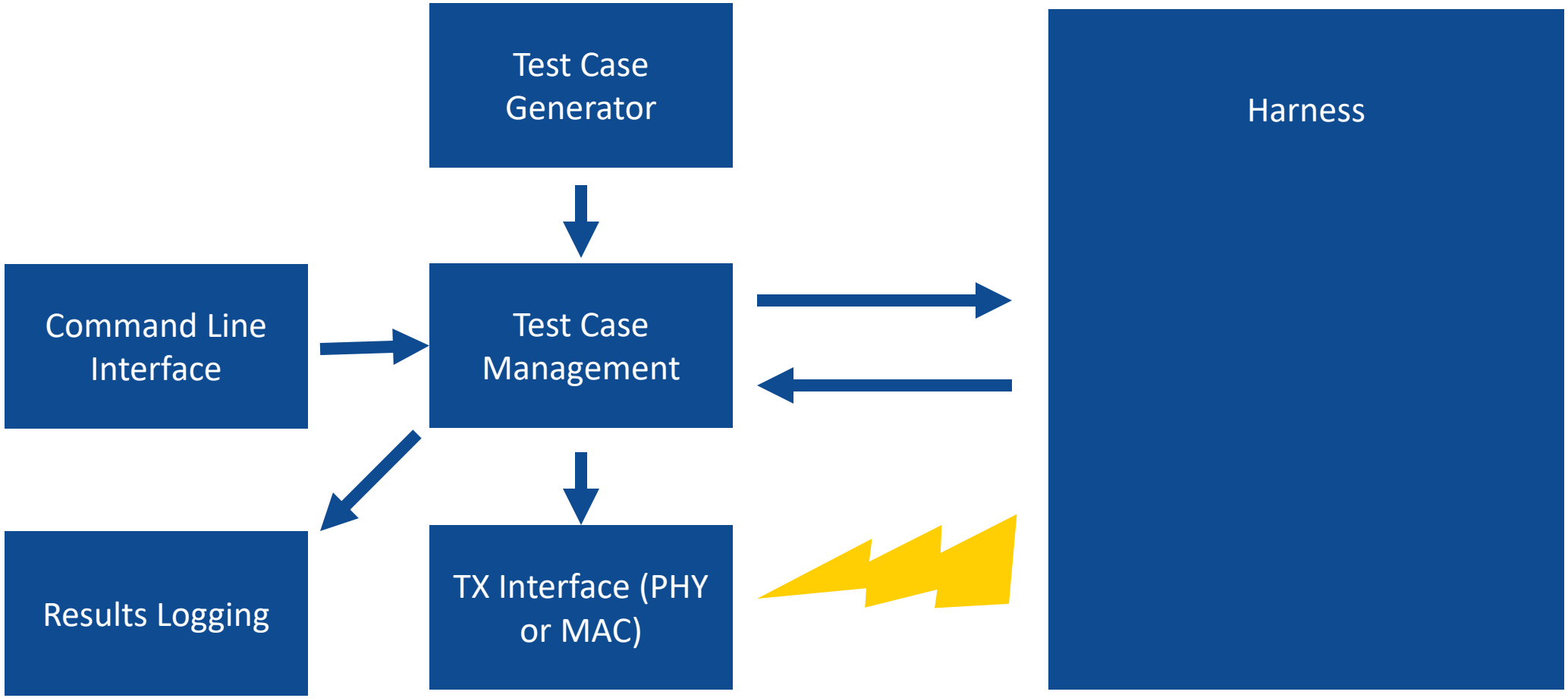
Previously known as unfAPI (Un-Named Fuzzing API)

TumbleRF

Software framework enabling fuzzing arbitrary RF protocols

Abstracts key components for easy extension

TumbleRF Architecture



Interfaces

RF injection/sniffing functions abstracted to generic template

To add a new radio, inherit base class and redefine its functions to map into any driver:

```
[set/get]_channel()
[set/get]_sfd()
[set/get]_preamble()
tx()
rx_start()
rx_stop()
rx_poll()
```

Generators

Rulesets for generating fuzzed input (pythonically)

Extend to interface with software fuzzers of your choice

Implement 2 functions:

```
yield_control_case()
yield_test_case()
```

Three generators currently:

- Preamble length (isotope)
- Non-standard symbols in preamble (isotope)
- Random payloads in message

Harnesses

Monitor the device under test to evaluate test case results

Manage device state in between tests

Three handlers currently:

- Received Frame Check: listen for given frames via an RF interface
- SSH Process Check: check whether processes on target crashed (beta)
- Serial Check: watch for specific ouptut via Arduino (beta)

Test Cases

Coordinate the generator, interface, and harness. Typically very lightweight.

Extend BaseCase to implement `run_test()`

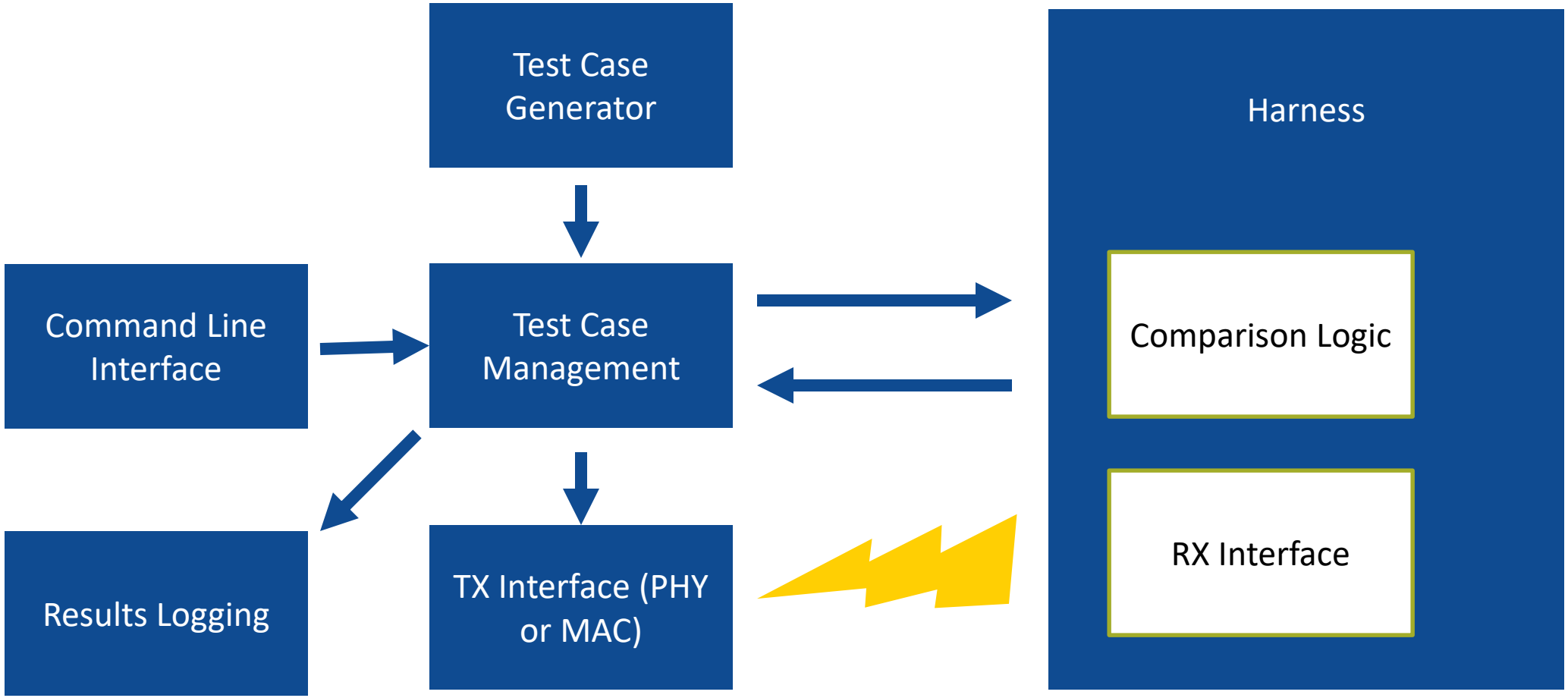
***or** build upon others, e.g.:*

Extend AlternatorCase to implement:

```
does_control_case_pass()
throw_test_case()
```

Alternates test cases with known-good control case to ensure interface is still up

TumbleRF Architecture: Demo Setup

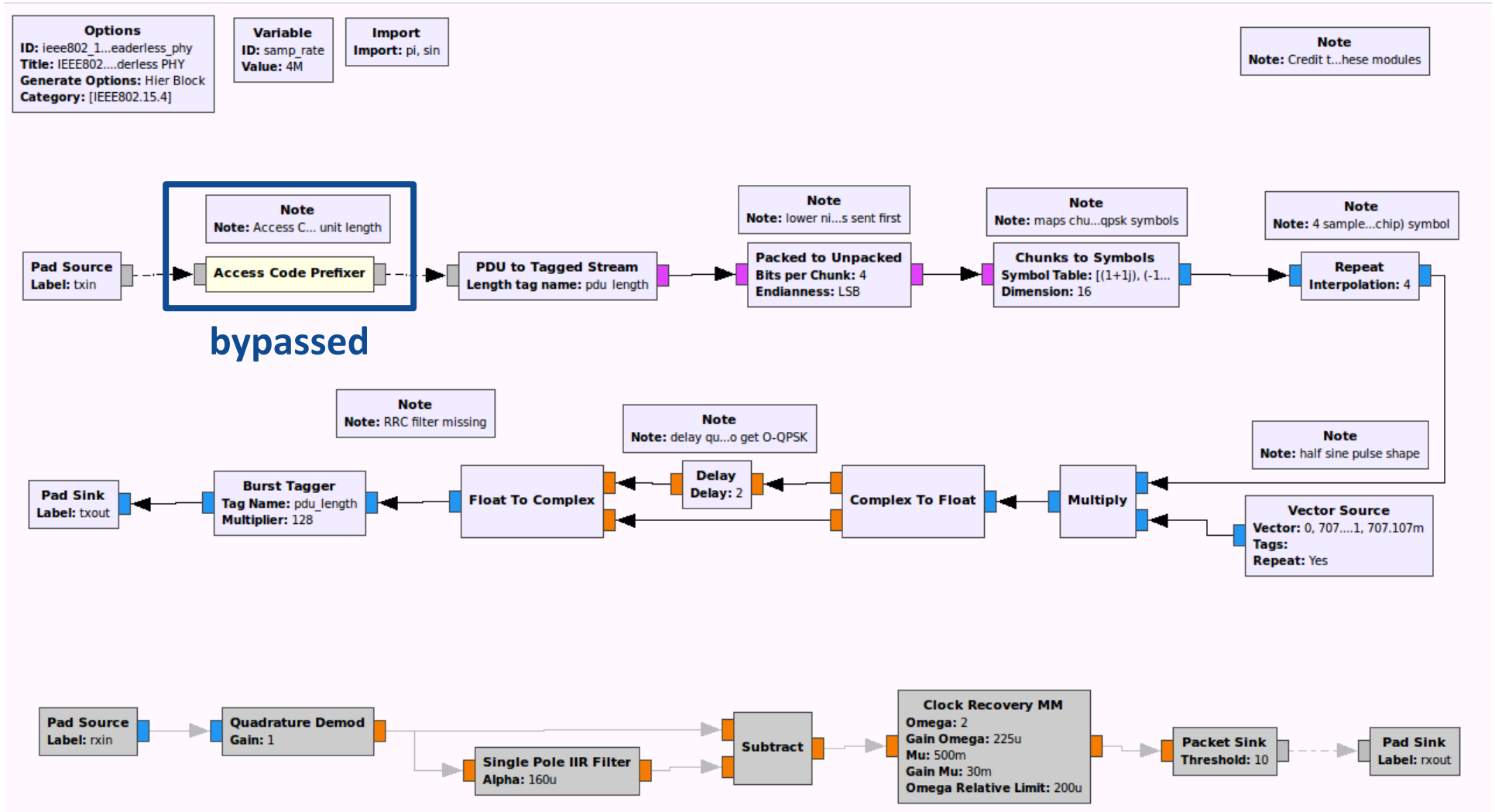


Example Generated Data: Preamble Length

Standard IEEE802.15.4 preamble: 0x00000000

Preamble	SFD	Length
[redacted]	0xA7	0xLL

Example Generated Data: Preamble Length



Arbitrary PHY injection via modified gr-ieee802-15-4

River Loop Security

Demo

Results Dump


```
Test: preamble_length_apimote.json (using Dot15d4PreambleLengthGenerator)
        Case 0: 0 valid, 50 invalid     example case: a70a230800ffff000007fba6
        Case 1: 0 valid, 50 invalid     example case: 70aa308220f0ff0f0070d0eafa
        Case 2: 45 valid, 5 invalid     example case: 00a70a230804ffff00000757b6
        Case 3: 0 valid, 50 invalid     example case: 0070aa308260f0ff0f007010e0fb
        Case 4: 50 valid, 0 invalid     example case: 0000a70a230808ffff000007a387
        Case 5: 0 valid, 50 invalid     example case: 000070aa3082a0f0ff0f007050fff8
        Case 6: 50 valid, 0 invalid     example case: 000000a70a23080cffff0000070f97
        Case 7: 0 valid, 50 invalid     example case: 00000070aa3082e0f0ff0f007090f5f9
        Case 8: 48 valid, 2 invalid     example case: 00000000a70a230810ffff0000074be4
        Case 9: 0 valid, 50 invalid     example case: 0000000070aa308220f1ff0f0070d0c1fe
Test: preamble_length_cc2531.json (using Dot15d4PreambleLengthGenerator)
        Case 0: 0 valid, 50 invalid     example case: a70a230800ffff000007fba6
        Case 1: 0 valid, 50 invalid     example case: 70aa308220f0ff0f0070d0eafa
        Case 2: 13 valid, 37 invalid    example case: 00a70a230804ffff00000757b6
        Case 3: 0 valid, 50 invalid     example case: 0070aa308260f0ff0f007010e0fb
        Case 4: 48 valid, 2 invalid     example case: 0000a70a230808ffff000007a387
        Case 5: 0 valid, 50 invalid     example case: 000070aa3082a0f0ff0f007050fff8
        Case 6: 50 valid, 0 invalid     example case: 000000a70a23080cffff0000070f97
        Case 7: 0 valid, 50 invalid     example case: 00000070aa3082e0f0ff0f007090f5f9
        Case 8: 49 valid, 1 invalid     example case: 00000000a70a230810ffff0000074be4
        Case 9: 0 valid, 50 invalid     example case: 0000000070aa308220f1ff0f0070d0c1fe
Test: preamble_length_rzusbstick.json (using Dot15d4PreambleLengthGenerator)
        Case 0: 0 valid, 50 invalid     example case: a70a230800ffff000007fba6
        Case 1: 0 valid, 50 invalid     example case: 70aa308230f0ff0f007060a8fa
        Case 2: 0 valid, 50 invalid     example case: 00a70a230805ffff0000077cb2
        Case 3: 0 valid, 50 invalid     example case: 0070aa308270f0ff0f0070a0a2fb
        Case 4: 0 valid, 50 invalid     example case: 0000a70a230809ffff0000078883
        Case 5: 0 valid, 50 invalid     example case: 000070aa3082b0f0ff0f0070e0bdf8
        Case 6: 37 valid, 13 invalid    example case: 000000a70a23080efffff000007599f
        Case 7: 0 valid, 50 invalid     example case: 00000070aa308200f1ff0f0070b044fe
        Case 8: 41 valid, 9 invalid     example case: 00000000a70a230813ffff00000736e8
        Case 9: 0 valid, 50 invalid     example case: 0000000070aa308250f1ff0f0070c00cff
```

River Loop Security

Why Care?

Those results can allow for WIDS evasion and selective targeting.

Get Involved

Contribute something:

- Generator for some cool new fuzzing idea you have
- Harness to check the state of a device you care about testing
- Interface to transmit with your favorite radio

Improve the code:

- Written carefully to be extensible, but… things can use improvement
  - More dynamic plugin loading
  - Improve plugin CLI parameter registration
  - ...

Thank You

Troopers and ERNW Crew: Niki, Enno, Rachelle, et al.

River Loop Security
Ionic Security