Burger Pager Hacking - Self Start

Hello! In this short project, you'll get a ground-level view on what it means to hack a radio system. Like many InfoSec professionals, you may already have a wealth of experience reverse engineering protocols in a non-radio context. Often you can leverage those skills for radio hacking if you can get the radio signals transformed into bits.

Although this project is very basic, it does show you the process of getting radio signals into bits. It also shows you how to operate transmitters that send your chosen bits to take over control of the original target - or other targets.

You won't have to grapple with a blank slate, however, because you'll be using Universal Radio Hacker (URH) a GUI-based tool, to complete each step. If you get stuck at any point, raise your hand for a helpful nudge.

This project consists of 7 steps, each step with its own page. Each of these pages has printing on two sides. On the front side, you'll see the instructions for the current step of the project. On the back side, you'll see the solution. Feel free to follow along with the presentation or focus on this guide - whatever works for you! You will, however, get the most out of this lab if you try to work through each step before turning to the solution.

Let's get started!

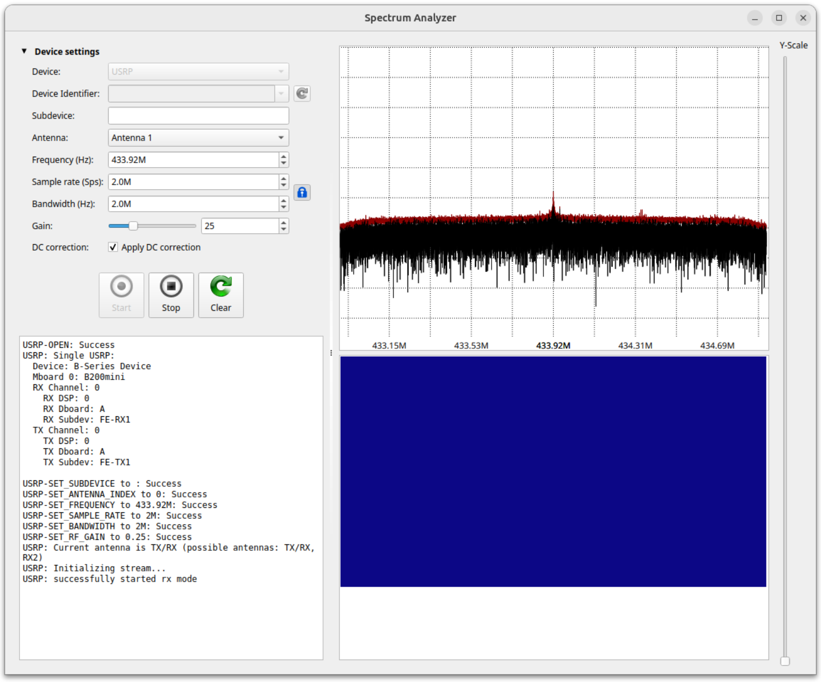
Step 1 - Find the Signal

Before you can start reverse engineering a radio signal, you have to find it. By “find” we’re not talking about a physical location, but rather a radio frequency. At any given moment, the place you’re currently sitting had dozens of radios signals going through it: WiFi, cellular voice/data, Bluetooth, broadcast FM radio, and the list goes on.

The reason all these radio signals can coexist is because they’re occupying different ***frequencies***. When we try to “find” the target signal, then, we’re really talking about the frequency it’s using.

Your computer should already have an instance of Universal Radio Hacker running. If not, please raise your hand and call someone over to get your laptop reset to Step 1 (it only takes a second).

Click the ***File*** menu in the upper left portion of URH and select ***Spectrum Analyzer***. Change the Sample Rate to “2M” and hit the Enter key. Then click the ***Start*** button. After a few seconds, you’ll see something like the display below. This is a live view of the RF spectrum.



Device: USRP
Device Identifier:
Subdevice:
Antenna: Antenna 1
Frequency (Hz): 433.92M
Sample rate (Sps): 2.0M
Bandwidth (Hz): 2.0M
Gain: 25
DC correction: ✓ Apply DC correction

```
USRP-OPEN: Success
USRP: Single USRP:
  Device: B-Series Device
  Mboard 0: B200mini
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: FE-RX1
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: FE-TX1

USRP-SET_SUBDEVICE to : Success
USRP-SET_ANTENNA_INDEX to 0: Success
USRP-SET_FREQUENCY to 433.92M: Success
USRP-SET_SAMPLE_RATE to 2M: Success
USRP-SET_BANDWIDTH to 2M: Success
USRP-SET_RF_GAIN to 0.25: Success
USRP: Current antenna is TX/RX (possible antennas: TX/RX,
RX2)
USRP: Initializing stream...
USRP: successfully started rx mode
```

You can retune to a new frequency by clicking anywhere on the upper right part of the display. Clicking near the left edge of the screen shifts the Spectrum Analyzer range to lower frequencies, near the right edge to higher frequencies. If the display freezes, restart it by pressing the ***Stop*** and then the ***Start*** button.

When you have the tuning set to the correct pager frequency, you will see a momentary spike in the upper right window, and a bright line in the waterfall plot on the lower right.

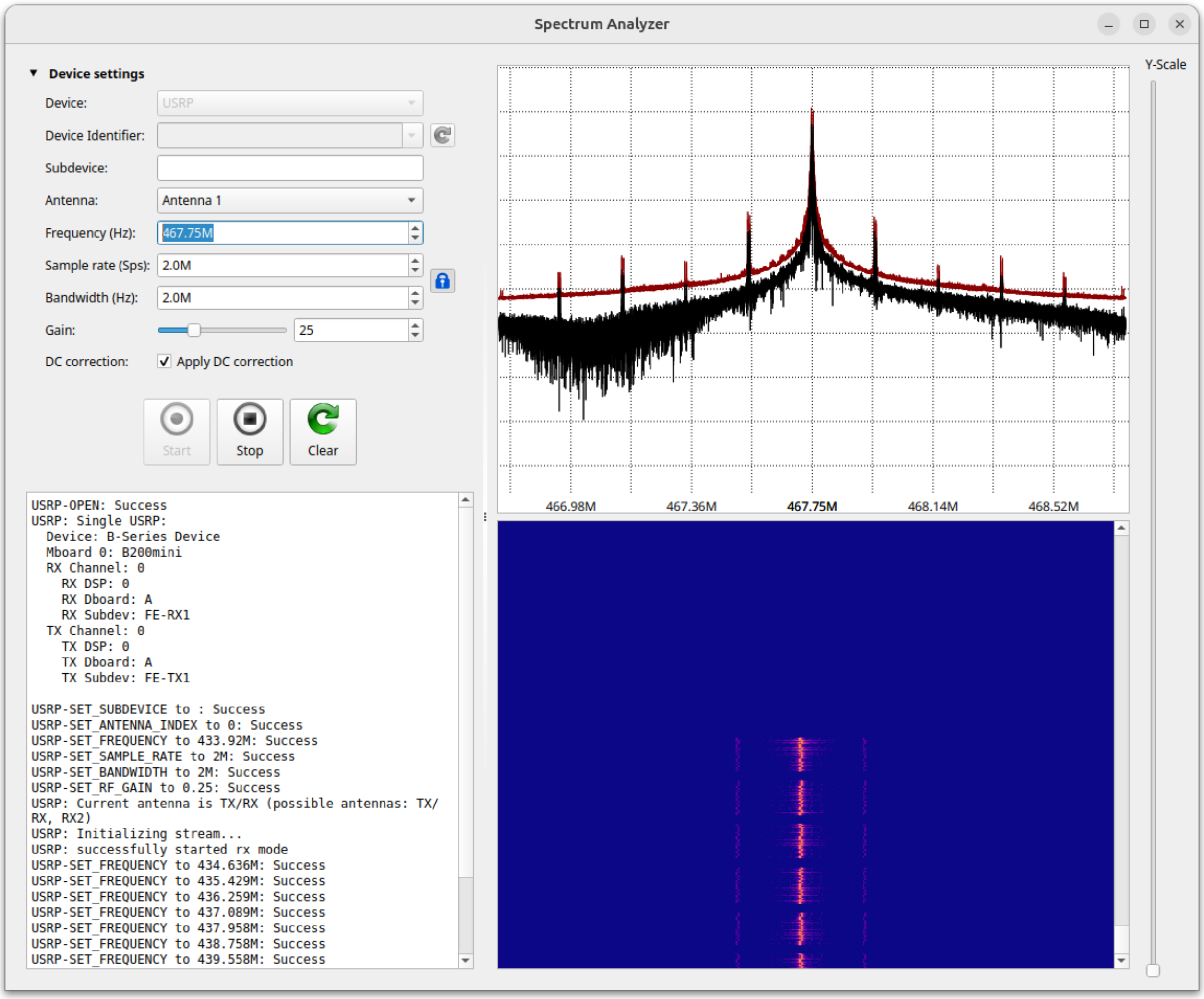
Note the frequency at which you see the activity (it will be in Mega Hertz, or MHz).

Congratulations! You’re done with Step 1!

Solution 1

Searching through the entire RF spectrum is not feasible. One shortcut is to start with some open source research: searching on “LRS star pager frequency” in Google reveals a range of 420-470 MHz. With only this information, the search space is small enough to manually search through with URH’s Spectrum Analyzer.

Further searching will reveal that in the US, these pagers operate at 467.75 MHz. When you’ve tuned to the correct frequency, the display will look like that below:



Device settings

Device: USRP
Device Identifier:
Subdevice:
Antenna: Antenna 1
Frequency (Hz): 467.75M
Sample rate (Sps): 2.0M
Bandwidth (Hz): 2.0M
Gain: 25
DC correction: ✓ Apply DC correction

```
USRP-OPEN: Success
USRP: Single USRP:
  Device: B-Series Device
  Mboard 0: B200mini
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: FE-RX1
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: FE-TX1

USRP-SET_SUBDEVICE to : Success
USRP-SET_ANTENNA_INDEX to 0: Success
USRP-SET_FREQUENCY to 433.92M: Success
USRP-SET_SAMPLE_RATE to 2M: Success
USRP-SET_BANDWIDTH to 2M: Success
USRP-SET_RF_GAIN to 0.25: Success
USRP: Current antenna is TX/RX (possible antennas: TX/
RX, RX2)
USRP: Initializing stream...
USRP: successfully started rx mode
USRP-SET_FREQUENCY to 434.636M: Success
USRP-SET_FREQUENCY to 435.429M: Success
USRP-SET_FREQUENCY to 436.259M: Success
USRP-SET_FREQUENCY to 437.089M: Success
USRP-SET_FREQUENCY to 437.958M: Success
USRP-SET_FREQUENCY to 438.758M: Success
USRP-SET_FREQUENCY to 439.558M: Success
```

Step 2 - Capture the Signal (OPTIONAL)

NOTE: This is the least interesting step to perform, and we've already got a captured file for you on disk, so if you want to skip this part, that's completely fine.

When reversing an RF signal, it's always best to create a copy of the radio signal on your hard drive. This will allow you to analyze the signal by processing the stored copy, instead of having to work on a real-time signal that's only available when someone is pressing that button.

Select ***File->Record signal…*** to bring up the capture interface. Change the ***Sample rate*** to "1M" and hit the Enter key. Then change the Frequency (Hz) to "467.75M" - the frequency we found in the previous step.

When you're ready to capture, hit the ***Start*** button. Then call out to a presenter and we'll activate the transmitter. When a transmission is sent, you'll see a vertical black pattern scroll to the left. After capturing several transmissions, hit the ***Stop*** button. Then hit the ***Save*** button and accept the default file name.

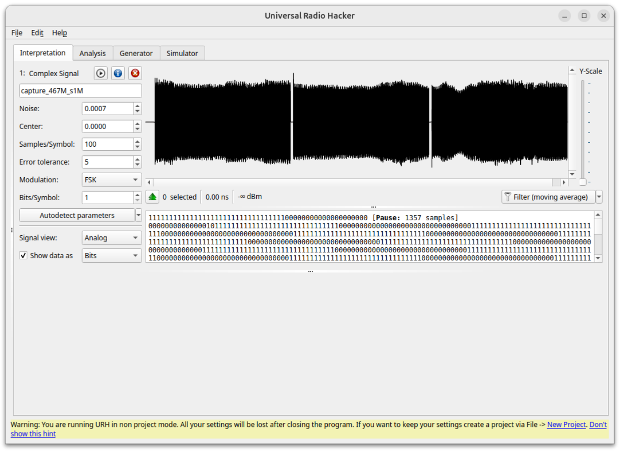
You now have a file on your hard drive containing a numerical description of the radio activity in the room at the time of capture. You can analyze this data and it will produce the same results as if you were analyzing live signals. You can think about this functionality like an audio recorder, except for radio waves.

Solution 2

If you have trouble capturing transmissions, no worries! You can always use the capture file already on your machine for the next steps.

Step 3 - Tune and Demodulate

If you created a capture file yourself, then closing the capture window will automatically take you to the ***Interpretation*** tab with your file loaded. If not, then click on the ***Interpretation*** tab and use ***File->Open*** to open the file on disk called:
`burger-pager-working/capture_c467M_s1M.complex`



capture_467M_s1M
Noise: 0.0007
Center: 0.0000
Samples/Symbol: 100
Error tolerance: 5
Modulation: FSK
Bits/Symbol: 1
Signal view: Analog
✓ Show data as Bits

```
11111111111111111111111111111111100000000000000000000 [Pause: 1357 samples]
000000000000010111111111111111111111111111111000000000000000000000000000000000011111111111111111111111111111
111000000000000000000000000000000000111111111111111111111111111111100000000000000000000000000000000011111111
111111111111111111111111100000000000000000000000000000000111111111111111111111111111111110000000000000000000
000000000000011111111111111111111111111111111000000000000000000000000000000000111111111111111111111111111111
110000000000000000000000000000000001111111111111111111111111111111000000000000000000000000000000000111111111
```

Warning: You are running URH in non project mode. All your settings will be lost after closing the program. If you want to keep your settings create a project via File -> New Project. Don't show this hint

In either case, it's time to analyze the capture RF transmissions. The goal in this step is to fine-tune the signal and demodulate it to produce the data bits.

Start by changing the ***Signal view*** to "Demodulated" and see what results. The goal is to find a demodulated waveform that looks somewhat digital. You could attempt to do this manually, by selecting values for each of the following:
- Modulation
- Samples/Symbol
- Center
- Noise

Instead, click the "Autodetect parameters" button, which will attempt to determine these parameters automatically.

Next, on the left side of the window, change the ***Show data as*** selection to "Hex". There should be one line of hex data for each transmission.

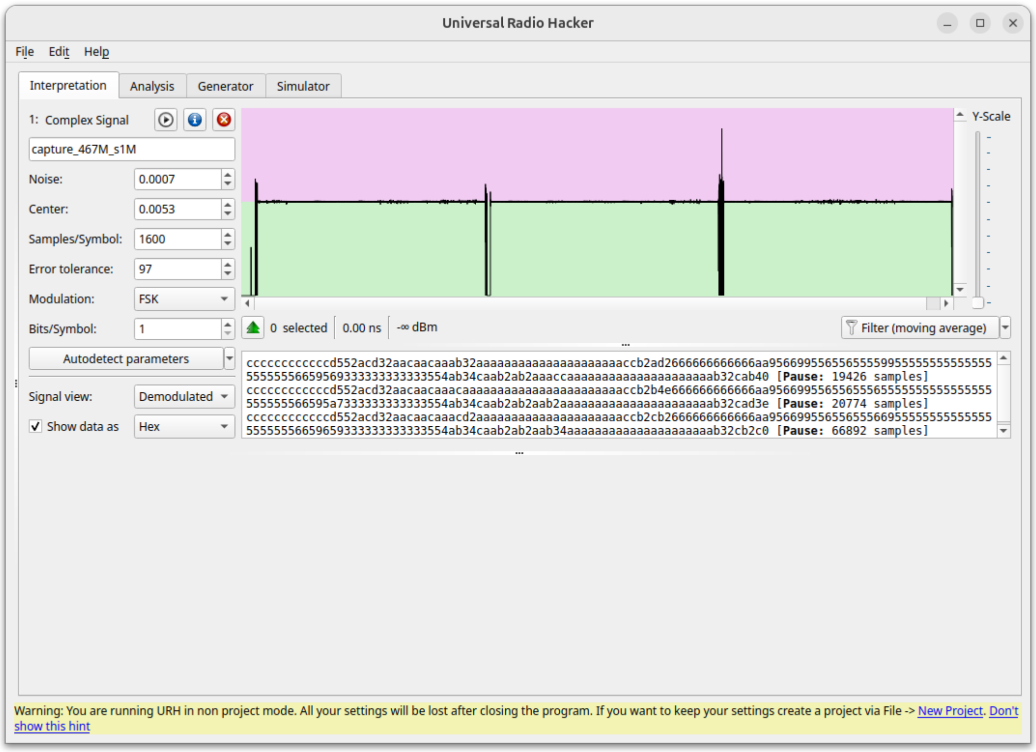
With some adjustment, you should see the contents of each line become more similar, though not exactly the same. This makes sense, as we'd expect signals from the same transmitter to be somewhat similar.

If the hex data isn't uniform, you can also experiment by clicking the "Filter (moving average)" button or by increasing the Noise value.

Solution 3

If you’re using the capture file we provided and clicked “Autodetect parameters”, you shouldn’t need to do anything more.

If you attempted to capture your own file, or manually demodulate the signal, then the smallest pulse in the waveform should equal 1.6 ms - yielding a ***Samples/Symbol*** value of 1600. As you adjust the pink-green threshold and the noise level, you should see each of your raw payloads start with several “c” digits.



capture_467M_s1M

Noise: 0.0007
Center: 0.0053
Samples/Symbol: 1600
Error tolerance: 97
Modulation: FSK
Bits/Symbol: 1
Signal view: Demodulated
✓ Show data as Hex

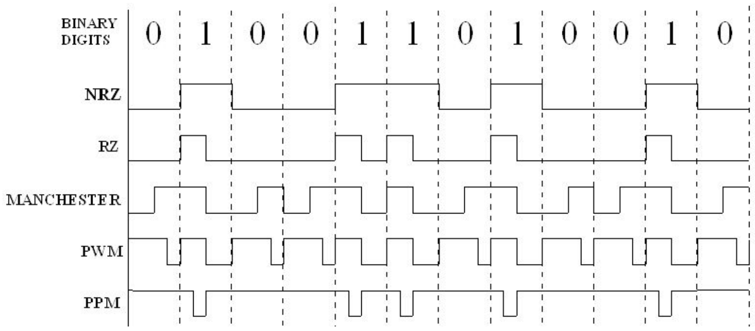
```
ccccccccccccd552acd32aacaacaaab32aaaaaaaaaaaaaaaaaaaaaaccb2ad2666666666666aa95669955656555599555555555555555
5555555665956933333333333554ab34caab2ab2aaaccaaaaaaaaaaaaaaaaaaaaab32cab40 [Pause: 19426 samples]
ccccccccccccd552acd32aacaacaaacaaaaaaaaaaaaaaaaaaaaaaaaccb2b4e666666666666aa95669955656555655555555555555555
555555566595a7333333333333554ab34caab2ab2aab2aaaaaaaaaaaaaaaaaaaaaab32cad3e [Pause: 20774 samples]
ccccccccccccd552acd32aacaacaaacd2aaaaaaaaaaaaaaaaaaaaaaccb2cb2666666666666aa95669955656555669555555555555555
5555555665965933333333333554ab34caab2ab2aab34aaaaaaaaaaaaaaaaaaaaab32cb2c0 [Pause: 66892 samples]
```

Warning: You are running URH in non project mode. All your settings will be lost after closing the program. If you want to keep your settings create a project via File -> New Project. Don't show this hint

Step 4 - Decode Raw Payload

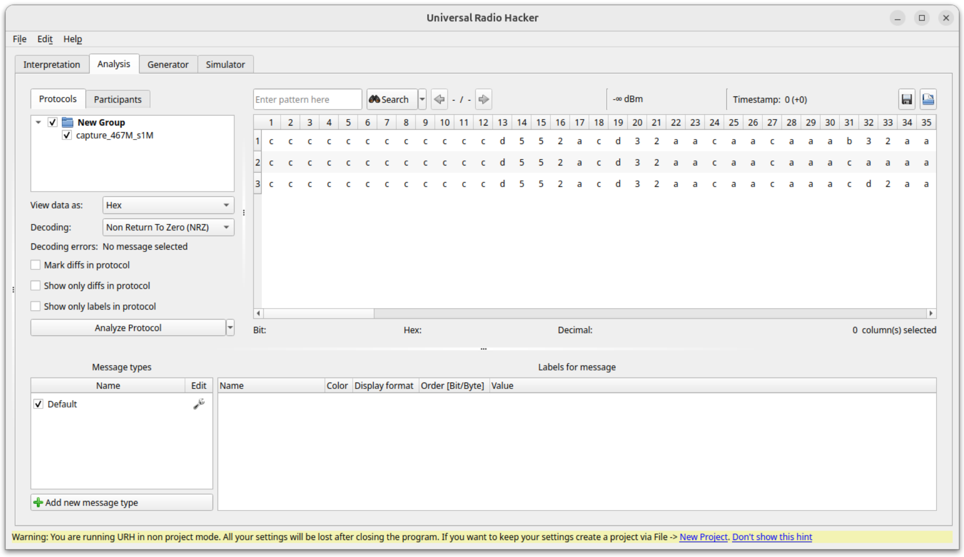
The data contained in a digital radio transmission may sometimes be equivalent to the raw bits you just recovered (technically this is called Non-Return to Zero or NRZ encoding). More often, however, the data is encoded before being sent.

The most common form of encoding is called Manchester encoding, where each data bit is encoded as a pair of bits having opposite values. A data 0 can be encoded as 10, while a data 1 can be encoded as 01 (this can also be reversed). Other forms of encoding such as PWM are also commonly seen in the wild.



First, take a look at the data you have by clicking in the ***Analysis*** tab.

What do you think? Does this look like it could be arbitrary data? On the left side, there is a pulldown control entitled ***View data as:*** - set this to "Hex". Look at the data again. Does it look reasonable to you?



View data as: Hex

Decoding: Non Return To Zero (NRZ)

Decoding errors: No message selected

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	c	c	c	c	c	c	c	c	c	c	c	c	d	5	5	2	a	c	d	3	2	a	a	c	a	a	c	a	a	a	b	3	2	a	a
2	c	c	c	c	c	c	c	c	c	c	c	c	d	5	5	2	a	c	d	3	2	a	a	c	a	a	c	a	a	a	c	a	a	a	a
3	c	c	c	c	c	c	c	c	c	c	c	c	d	5	5	2	a	c	d	3	2	a	a	c	a	a	c	a	a	a	c	d	2	a	a

Name	Edit
Default	

Name	Color	Display format	Order [Bit/Byte]	Value

Warning: You are running URH in non project mode. All your settings will be lost after closing the program. If you want to keep your settings create a project via File -> New Project. Don't show this hint

Immediately below ***View data as:*** you'll find a ***Decoding:*** control. Try the various options and see what encoding seems most likely to be used here. This could include the NRZ, or not-really-encoded option.

Solution 4

The initial, NRZ encoded data is weird. Maybe it looks off just by looking at it, maybe not. The strange thing about the data is what you don't see - several hex digits are entirely absent from data. There are a lot of the following hex value: C, A, 5, and many others.

But where are the following characters: 0, 1, 7, 8, E, F? It seems odd that they are entirely absent from the nearly 200 hex digits in each transmission. What do those characters all have in common?

No worries if it doesn't jump out at you - each of them has 3 or more consecutive 0s or 1s! In fact, they are the only six hex digits with that characteristic. Recall that Manchester consists of pairs of opposite bits strung together - in such an encoding, you will never have more than two of the same bit values consecutively. In other words, this data looks very Manchester-like!

However, there are still two Manchester options in URH, labeled Manchester I and Manchester II. Which one is it? Like a lot of questions in reverse engineering, the simplest answer is "try both." In the next step, we'll see which of the Manchester options produces the most credible data.

Step 5 - Identify the Data

The next challenge is to figure out the purpose of each segment of the decoded data. This can be a bit like a puzzle. Before we tell you anything further, stop reading and just take a look at the data in the ***Analysis*** tab. What kinds of things do you notice?

Questions to consider:

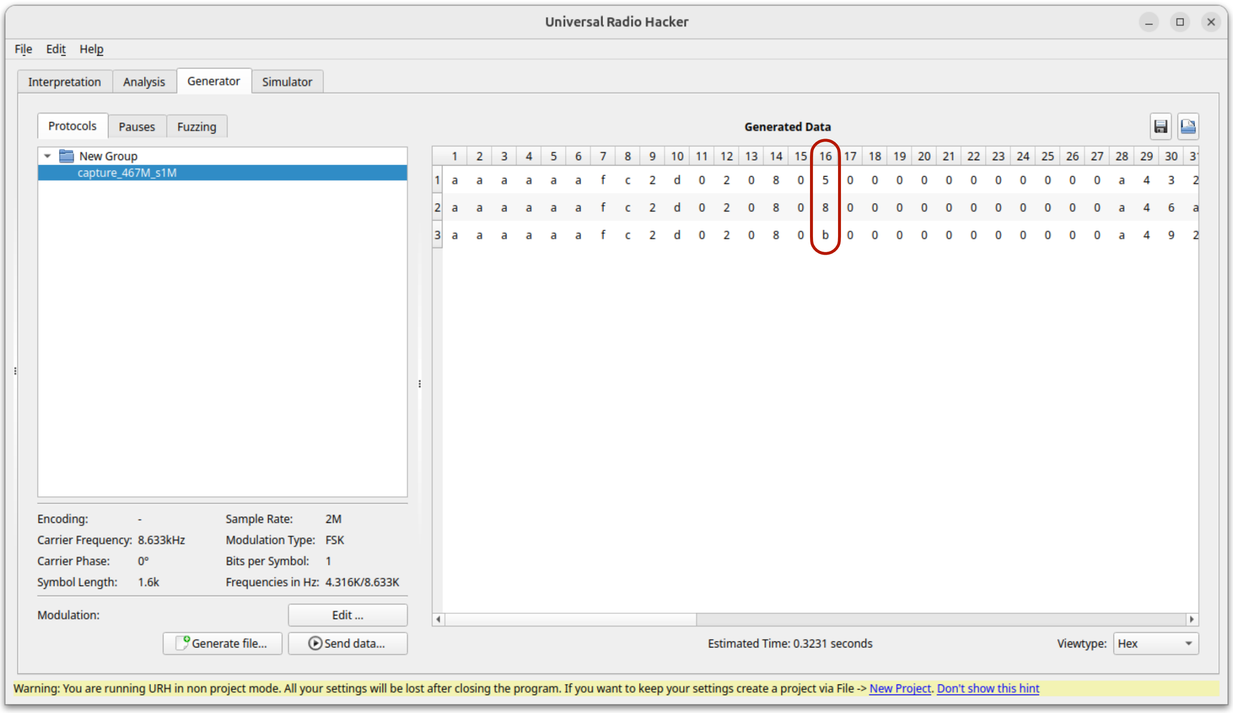
- Which parts of the data are the same between transmissions?
- Which parts are different?
- Do you see portions of the data repeat?
- What functions might the various parts of the waveform perform?
- Where in the packet does a particular sequence of data appear?
  - Preambles and sync words are near the beginning
  - Checksums tend to be at the end

You don't have quite enough data do figure out what all of the data does, but that's OK! If a particular part of the packet doesn't change in any of your captured data, then maybe it doesn't need to change for you to gain control over some systems.

And if you can capture more data in the future, you may be able to learn more. Start by combining through the data looking for the pager ID. If you're using the pre-recorded capture file, then three pager IDs will be present: 5, 8 and 11.

Solution 5

With Manchester II decoding, you can see the pager ID in Hex Digit #16, which is very good evidence for that decoding being correct.



Encoding: -
Carrier Frequency: 8.633kHz
Carrier Phase: 0°
Symbol Length: 1.6k
Sample Rate: 2M
Modulation Type: FSK
Bits per Symbol: 1
Frequencies in Hz: 4.316K/8.633K
Modulation:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	3
1	a	a	a	a	a	a	f	c	2	d	0	2	0	8	0	5	0	0	0	0	0	0	0	0	0	0	0	a	4	3	2
2	a	a	a	a	a	a	f	c	2	d	0	2	0	8	0	8	0	0	0	0	0	0	0	0	0	0	0	a	4	6	a
3	a	a	a	a	a	a	f	c	2	d	0	2	0	8	0	b	0	0	0	0	0	0	0	0	0	0	0	a	4	9	2

Estimated Time: 0.3231 seconds

Viewtype: Hex

Warning: You are running URH in non project mode. All your settings will be lost after closing the program. If you want to keep your settings create a project via File -> New Project. Don't show this hint

Each transmission is actually a burst of 3 identical packets.

Packet 1: Hex digits 1-30
Packet 2: Hex digits 31-60
Packet 3: Hex digits 61-90

Within each packet, you have the following fields:

Function	Hex Digits	Value	Notes
Preamble	1-6	0xAAAAAA	Initial part of packet; for synchronization
Sync Word	7-10	0xFC2D	Unique pattern to identify type of transmission
Site ID	11-12	0x02	Identifier for a physical location; prevents communications with adjacent sites
System ID	13-14	0x08	Identifier for an individual pager system; used if a single location operates multiple pager networks
Pager ID	15-16	Varies	Identifier for an individual pager device; in our case, these are printed on the pager label
Reserved	17-26	0x0000000000	Unused in this implementation of the protocol
Action	27-28	0x0A	Directs the pager to respond in different ways to being paged: one short pulse, longer pulses, beeps, etc
Checksum	29-30	Varies	Provides basic error checking; pager will not respond if this value is incorrect; but how is it computed?

Step 6 - Computing the Checksum

A checksum is a deterministic arithmetic operation performed on a payload, the result of which is appended to the payload upon transmission. The receiver then performs the same calculation on the payload data and compares it to the transmitted checksum. If the two values do not match, the transmission is considered to have an error and is ignored by the receiver.

To successfully transmit your own control signal, you will need to be able to replicate this checksum. The algorithm could be anything from simple arithmetic to a more complex CRC computation.

Look at the data in your packets and see if you can determine the algorithm for the checksum.

Solution 6

If you use the payload values from the pre-recorded file, you'll see only two hex digits change:

- the pager ID value (16)
- the checksum (29 & 30)

When the pager ID changes from:

- 0x08 to 0x0b

… the checksum changes from

- 0x46 to 0x49

Doing some hex arithmetic, the pager ID increased by:
0x0b - 0x08 = 0x03
And the checksum increased by:
0x49 - 0x46 = 0x03

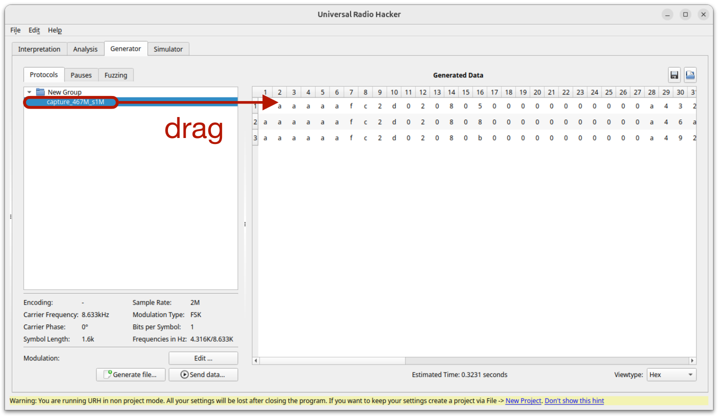
At first glance, the algorithm appears to be a simple sum of some (or all) of the payload bytes. Although it would be good to have more data to validate this hypothesis, it's often faster to just try things. Hence the next step…

Step 7 - Take Over Your Pager

Now you have a strong idea of the type of signal you'll need to send to activate your pager:

- a burst of three packets
- fixed values for all payload bytes except
  - Pager ID (corresponds to the label on your pager <u>in hex</u>)
  - Checksum (appears to increase and decrease in step with the Pager ID)

Now switch to the Generator tab of URH so you can send out your takeover signal. First, click and drag the name of the capture file into the empty generated data window, then select a "Viewtype" as "Hex". You will then see something like the display shown below:



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	[illegible]	a	a	a	a	a	f	c	2	d	0	2	0	8	0	5	0	0	0	0	0	0	0	0	0	0	0	a	4	3	2
2	a	a	a	a	a	a	f	c	2	d	0	2	0	8	0	8	0	0	0	0	0	0	0	0	0	0	0	a	4	6	a
3	a	a	a	a	a	a	f	c	2	d	0	2	0	8	0	b	0	0	0	0	0	0	0	0	0	0	0	a	4	9	2

Encoding: -
Sample Rate: 2M
Carrier Frequency: 8.633kHz
Modulation Type: FSK
Carrier Phase: 0°
Bits per Symbol: 1
Symbol Length: 1.6k
Frequencies in Hz: 4.316K/8.633K
Modulation:
Estimated Time: 0.3231 seconds
Viewtype: Hex

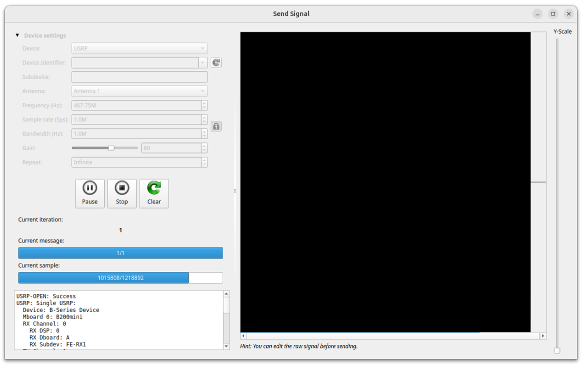
Warning: You are running URH in non project mode. All your settings will be lost after closing the program. If you want to keep your settings create a project via File -> New Project. Don't show this hint

Delete all but one of the lines of data by selecting the row and hitting the DELETE (not Backspace!) key - you only need to send one burst. Modify the single, remaining line of data to match the Pager ID and Checksum you want to send. Make the same change to each of the three packets in the burst:

- Pager ID
  - 16
  - 46
  - 76
- Checksum
  - 29 & 30
  - 59 & 60
  - 89 & 90

Once you've made these edits, you can send this signal by clicking on the ***Send Data*** button in the bottom left. This will bring up a transmit control window. URH will preserve the settings used to capture the original file, so everything should be set correctly except the ***Gain*** value - set this to "60". If your ***Frequency (Hz)*** value isn't "467.75M", change it to that.

Then click the ***Start*** button to transmit. You should see your pager react.



Device settings
Device: USRP
Device Identifier:
Subdevice:
Antenna: Antenna 1
Frequency (Hz): 467.75M
Sample rate (Sps): 1.0M
Bandwidth (Hz): 1.0M
Gain: 60
Repeat: Infinite

Current iteration: 1
Current message: 1/1
Current sample: 1015808/1218892

```
USRP-OPEN: Success
USRP: Single USRP:
  Device: B-Series Device
  Mboard 0: B200mini
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: FE-RX1
```

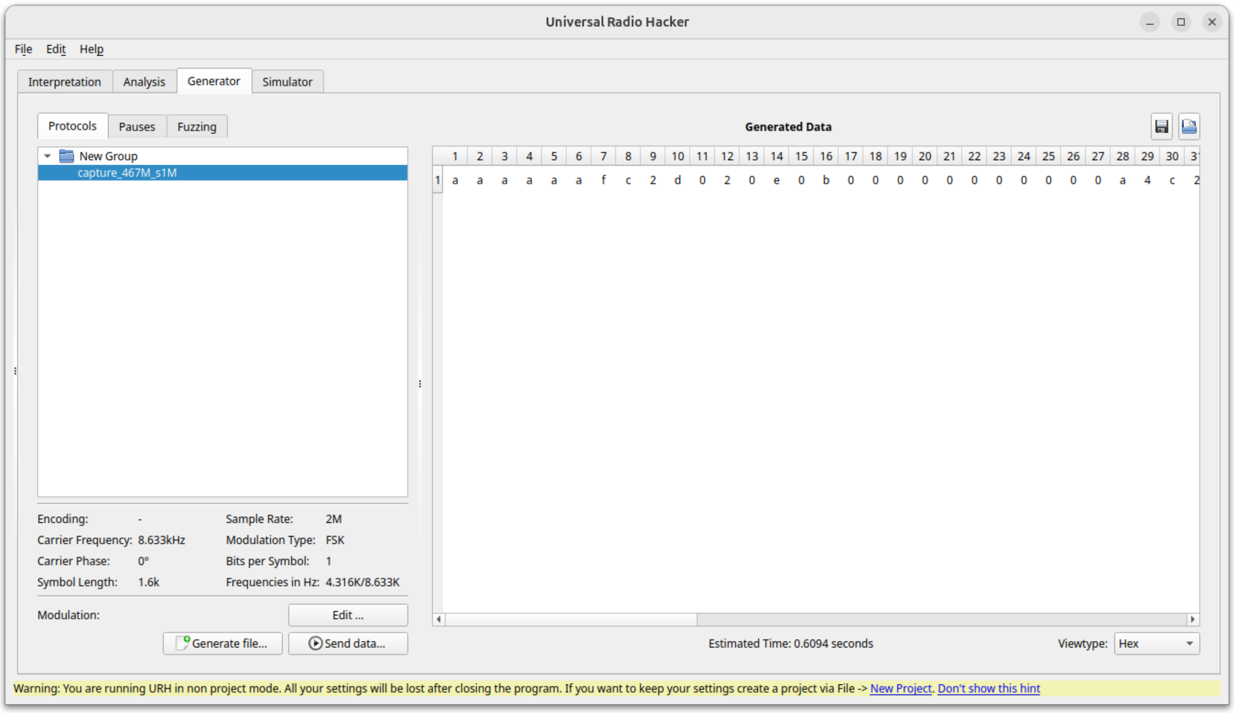
Hint: You can edit the raw signal before sending.

Solution 7 - Hint

You'll first need to set your Pager ID to the hex value on your pager's label. For this example, I'll assume your pager label reads "14" and that you're replacing a transmission with a targeting pager #11. You will then:

- change the three pager ID locations to 0x0e (the hex equivalent of decimal 14)
- increase the existing checksum by the same amount
  - Pager ID went up by
    - 0x0e - 0x0b = 0x03
  - the new checksum is then the old one increased by the same amount
    - 0x49 + 0x03 = 0x4c

Thus for this example, you would change all three checksum locations to 0x4c.



Universal Radio Hacker

Generated Data

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	a	a	a	a	a	a	f	c	2	d	0	2	0	e	0	b	0	0	0	0	0	0	0	0	0	0	0	a	4	c	2

New Group
capture_467M_s1M

Encoding: -
Sample Rate: 2M
Carrier Frequency: 8.633kHz
Modulation Type: FSK
Carrier Phase: 0°
Bits per Symbol: 1
Symbol Length: 1.6k
Frequencies in Hz: 4.316K/8.633K

Estimated Time: 0.6094 seconds
Viewtype: Hex

Warning: You are running URH in non project mode. All your settings will be lost after closing the program. If you want to keep your settings create a project via File -> New Project. Don't show this hint

If you're stuck figuring out the checksum, turn the page to see the values for each pager ID. If the transmission still doesn't work, make sure you have your gain set to 60.

Solution 7 - Full

The following lists the values necessary to trigger each pager. You will need to edit the locations shown below to include the new values:

Pager ID	16	29&30	46	59&60	76	89&90
1	1	3F	1	3F	1	3F
2	2	40	2	40	2	40
3	3	41	3	41	3	41
4	4	42	4	42	4	42
5	5	43	5	43	5	43
6	6	44	6	44	6	44
7	7	45	7	45	7	45
8	8	46	8	46	8	46
9	9	47	9	47	9	47
10	A	48	A	48	A	48
11	B	49	B	49	B	49
12	C	4A	C	4A	C	4A
13	D	4B	D	4B	D	4B
14	E	4C	E	4C	E	4C
15	F	4D	F	4D	F	4D

Conclusion

We hope you had an informative and entertaining time with us! Please let us know if you have any questions or thoughts about this project or any other radio-hacking topics.

Thanks for joining us!