Hacking Older Garmins

Modifying firmware on the Garmin Forerunner 35

As an avid runner and cyclist, I recently upgraded from my 7-year-old Garmin Forerunner 35 to a Forerunner 255. Like any curious hacker, I decided to tamper with my old watch to see what I could modify. Finding limited up-to-date resources about this older device, I figured I'd document my research to make Garmin hacking more accessible.

This guide will document my journey with the Garmin Forerunner 35 specifically, but most of the **Firmware Modification** section is relevant to *any* Garmin using the RGN update file format, which is most older Garmins.

Methods

There are two approaches to "hacking" the Garmin Forerunner 35: modifying the actual firmware and pushing the modified firmware in an update, or directly modifying system files.

Direct Filesystem Modification

When you plug the Forerunner 35 into Garmin Express, the device's filesystem appears under USB devices. Key folders include `ACTIVITY` (containing workout FIT files), various other folders containing configuration FIT files, and a `TEXT` folder with `.LNG` language files.

FIT Files

Unlike FIT files for workouts which contain waypoints and other workout data, configuration FIT files for the Garmin contain a "header" section with some device information such as the Unit ID, then their various config options. These files are in a binary format but can be converted to CSV using Garmin's FIT CSV Tool. From my testing, tampering with the header section showed no actual changes on the device. Beyond that, most of the config options are just device settings you can change from the watch itself.

LNG Files

LNG files contain program strings in various languages. There's no validation on the strings, so you can change them freely—just keep the file header intact and don't change the overall file length, or the string offsets will be wrong and your device becomes unreadable (if you mess this up, you can always factory reset the watch to go back).

I attempted a buffer overflow by removing strings entirely, but it just produced NULL bytes for all strings instead.

So the LNG file approach lets you modify strings, but what about actual functionality? Or changing strings for the default language, which has no LNG file?

Firmware Modification

Based on Herbert Oppmann's analysis (see Acknowledgements), RGN files serve as Garmin "update files" containing various sections called records. Each record has a header specifying its type and data size. The process of patching firmware on the Garmin looks like this:

1. Make modifications to the RGN file (how to modify it is what this section deals with)
2. Plug the watch into your computer, and drag the modified RGN file into the main folder. Rename it to GUPDATE.rgn
3. Unplug the watch, and you should get prompted to Install an update.

I built a tool to parse RGN files (based on an old C RGN parsing tool I found, see Acknowledgements) that returns headers, offsets, and other relevant information for each record. Running it on a Forerunner 35 firmware file from GMapTools revealed that the actual firmware is in the region record with ID 14:

```
Builder: SQA
Date: Oct 25 1999
Time: 14:16:13

----------------------------

Total Size: 32778 bytes, Type: R (REGION_TYPE)
Record Data Size: 32768 bytes
Record Data Start: 0x0000003C
Record Data End: 0x0000803B
-- RECORD DATA --
Region ID: 5
    * Purpose: Identified but purpose unknown
Delay: 0 ms
Contents (first 16 bytes): 01 48 85 46 01 48 00 47 C0 83 FF 1F D1 1A FF 1F

----------------------------

Total Size: 520202 bytes, Type: R (REGION_TYPE)
Record Data Size: 520192 bytes
Record Data Start: 0x0000804B
Record Data End: 0x0008704A
-- RECORD DATA --
Region ID: 14
    * Purpose: fw_all.bin (In DeltaSmart_350.rgn, the content is again a RGN file which contains the actual firmware in
its region 14.)
Delay: 0 ms
Contents (first 16 bytes): 00 20 00 21 00 22 00 23 00 24 00 25 00 26 00 27
```

Output from RGNTool on a sample RGN

Exploring the firmware record, I found XML-like content, strings, and lots of source file paths. Feeling a bit lost, I simply identified a string showing the device's IC and M/N numbers, that I knew was visible on the watch settings. I changed a digit and uploaded the patched RGN. Surprisingly, it worked! When I viewed the System information page on the watch, these new numbers were shown.

changeSN.rgn

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00074D50	67	61	72	6D	69	6E	6F	73	5C	64	72	69	76	65	72	5C	garminos\driver\
00074D60	61	63	63	65	6C	5C	68	77	6D	5F	61	63	63	65	6C	5F	accel\hwm_accel_
00074D70	6D	67	72	5F	63	6C	69	65	6E	74	5F	77	68	72	2E	63	mgr_client_whr.c
00074D80	3A	20	33	39	33	00	18	FE	19	07	09	C2	0B	60	0C	FC	: 393..þ...Â.`.ü
00074D90	00	55	00	50	00	A9	01	AA	01	AB	01	AC	01	AD	01	AE	.U.P.©.ª.«.¬...®
00074DA0	01	AF	01	06	00	06	00	24	00	10	00	03	00	02	00	00	.¯.....$........
00074DB0	26	61	6E	69	6D	61	74	69	6F	6E	5F	64	72	61	77	5F	&animation_draw_
00074DC0	77	6F	72	6B	71	5F	69	74	65	6D	00	25	30	36	69	00	workq_item.%06i.
00074DD0	76	2E	20	25	75	2E	25	75	25	75	00	49	43	3A	20	31	v. %u.%u%u.IC: 1
00074DE0	37	39	32	41	2D	30	32	39	39	30	0A	4D	2F	4E	3A	20	792A-02990.M/N:
00074DF0	41	30	32	39	39	30	00	47	50	53	3A	20	25	75	2E	25	A02990.GPS: %u.%
00074E00	30	32	75	00	0A	42	4C	45	2F	41	4E	54	3A	20	25	75	02u..BLE/ANT: %u
00074E10	2E	25	30	32	75	00	0A	57	48	52	3A	20	25	75	2E	25	.%02u..WHR: %u.%
00074E20	30	32	75	00	48	65	61	72	74	20	72	61	74	65	2D	62	02u.Heart rate-b
00074E30	61	73	65	64	20	63	61	6C	6F	72	69	65	73	20	63	61	ased calories ca
00074E40	6C	63	75	6C	61	74	65	64	20	62	79	20	46	69	72	73	lculated by Firs
00074E50	74	62	65	61	74	2E	00	26	73	63	72	65	65	6E	5F	74	tbeat..&screen_t
00074E60	72	61	6E	73	5F	77	6F	72	6B	71	5F	69	74	65	6D	00	rans_workq_item.
00074E70	2E	2E	5C	2E	2E	5C	2E	2E	5C	42	54	46	5C	62	74	66	..\..\..\BTF\btf
00074E80	5F	6E	61	6E	6F	70	62	2E	63	3A	20	36	32	00	00	29	_nanopb.c: 62..)
00074E90	00	2A	00	2B	00	2C	00	2D	00	01	02	44	00	48	00	48	.*.+.,.-...D.H.H

changeSN.rgn

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00074D50	67	61	72	6D	69	6E	6F	73	5C	64	72	69	76	65	72	5C	garminos\driver\
00074D60	61	63	63	65	6C	5C	68	77	6D	5F	61	63	63	65	6C	5F	accel\hwm_accel_
00074D70	6D	67	72	5F	63	6C	69	65	6E	74	5F	77	68	72	2E	63	mgr_client_whr.c
00074D80	3A	20	33	39	33	00	18	FE	19	07	09	C2	0B	60	0C	FC	: 393..þ...Â.`.ü
00074D90	00	55	00	50	00	A9	01	AA	01	AB	01	AC	01	AD	01	AE	.U.P.©.ª.«.¬...®
00074DA0	01	AF	01	06	00	06	00	24	00	10	00	03	00	02	00	00	.¯.....$........
00074DB0	26	61	6E	69	6D	61	74	69	6F	6E	5F	64	72	61	77	5F	&animation_draw_
00074DC0	77	6F	72	6B	71	5F	69	74	65	6D	00	25	30	36	69	00	workq_item.%06i.
00074DD0	76	2E	20	25	75	2E	25	75	25	75	00	49	43	3A	20	31	v. %u.%u%u.IC: 1
00074DE0	37	39	32	41	2D	30	32	39	39	30	0A	4D	2F	4E	3A	20	792A-02990.M/N:
00074DF0	41	30	32	39	39	30	00	47	50	53	3A	20	25	75	2E	25	A02990.GPS: %u.%
00074E00	30	32	75	00	0A	42	4C	45	2F	41	4E	54	3A	20	25	75	02u..BLE/ANT: %u
00074E10	2E	25	30	32	75	00	0A	57	48	52	3A	20	25	75	2E	25	.%02u..WHR: %u.%
00074E20	30	32	75	00	49	20	77	65	6E	74	20	69	6E	74	6F	20	02u.I went into
00074E30	74	68	65	20	66	69	72	6D	77	61	72	65	20	61	6E	64	the firmware and
00074E40	20	72	65	70	6C	61	63	65	20	74	68	69	73	20	73	74	replace this st
00074E50	72	69	6E	67	21	2E	00	26	73	63	72	65	65	6E	5F	74	ring!..&screen_t
00074E60	72	61	6E	73	5F	77	6F	72	6B	71	5F	69	74	65	6D	00	rans_workq_item.
00074E70	2E	2E	5C	2E	2E	5C	2E	2E	5C	42	54	46	5C	62	74	66	..\..\..\BTF\btf
00074E80	5F	6E	61	6E	6F	70	62	2E	63	3A	20	36	32	00	00	29	_nanopb.c: 62..)
00074E90	00	2A	00	2B	00	2C	00	2D	00	01	02	44	00	48	00	48	.*.+.,.-...D.H.H

The process of modifying strings in the RGN file, using HxD

I then tried to change some strings, such as the above example. However, attempting to change text strings failed—the updates seemed to install, but the strings did not appear modified on the device. It seemed like an integrity check was allowing those single-digit changes, but not whole strings.

That's when found an old GitHub repo with what looked like actual internal Garmin C code, and it contained a `signed_updates` folder, suggesting some kind of private key signature being used for updates. I was beginning to lose hope, but then I discovered

Herbert Oppmann's BIN format documentation (see Acknowledgements). Oppmann to the rescue again!

I reviewed his documentation, and found that my firmware section had FF padding to `0x0200`, indicating load address `0x1000`. I disassembled using ARM objdump:

```
arm-none-eabi-objdump -marm -Mforce-thumb -b binary -D --adjust-vma=0x1000
    fw_all.bin > disasm_0x1000.txt
```

And this disassembly showed the expected "Clear register pattern":

```
00001000 <.data>:
    1000:       2000            movs    r0, #0
    1002:       2100            movs    r1, #0
    1004:       2200            movs    r2, #0
    1006:       2300            movs    r3, #0
    1008:       2400            movs    r4, #0
```

So I knew my BIN file actually matched this documentation. That's when I read the checksum calculation: "Start with byte value 0. Add each firmware byte sequentially from offset 0, except the last. If file length isn't a multiple of 256, add byte value 255 for missing bytes. Add the checksum byte. Result should be 0."

Much simpler than private key signatures! I added checksum patching to my RGN tool and successfully modified various format specifiers and strings:

Version string displaying huge version numbers

Modified interface text strings

Format specifiers changed from `%u` to `%p` for hex display

Getting overconfident, I tried changing some of the XML values like Unit ID and model number... and bricked the watch.

Conclusion

Modifying Garmin Forerunner 35 firmware is surprisingly easy due to its simple checksum mechanism. This applies to most Garmin devices using RGN format (versus newer GCD format)—many older devices and some modern maritime ones use RGN.

Warning: The firmware binary is complex and contains data in various different formats and levels of compression. Unless you understand exactly what a section

does, **only modify strings whose uses you can isolate**. You can easily brick your device, rendering it unusable and unrestorable.

If you do modify strings, be sure to keep the overall file length the same. You can pad with NULL bytes as necessary. To avoid bricking the device, if you are just hoping to modify strings, the LNG file method is probably best.

RGNTool Usage

If you read the disclaimer and still want to try modifying your Garmin device's firmware, here's how you can use RGNTool to help:

1. Make a copy of the RGN file (These can be found by Googling your device RGN files, or see the link to GMapTools in Acknowledgements)
2. Run the parse command to find the start/end offsets for the firmware record
3. Modify the desired portion of that record with a hex editor. You should also change an easy-to-find string so that you can confirm the firmware was actually patched (if the checksum is somehow incorrect, it may appear to install the update but not actually do so)
4. Run the checksum command with the same start/end offsets. It will prompt you to change the checksum, type Y to do so
5. Plug in your Garmin device and drop the modified RGN file into the main directory. Rename it GUPDATE.rgn
6. Unplug the Garmin device and Install the update when prompted. Your device will restart and the changes should be applied!

This was an excellent learning experience about proprietary file formats. Special thanks to Herbert Oppmann for his invaluable documentation—check out his website for more file format analysis.

If you have an older Garmin device to experiment with, my RGN Tool can help!

Acknowledgements

- **Garmin RGN Firmware Update File Format** & **Garmin BIN Firmware File Format**: Herbert Oppmann's file format analyses
- **Pimp my Garmin**: Blog post that led me to the BIN format analysis, and gave me some interesting ideas.
- **GMapTools**: Source for sample RGN firmware files
- **RGN GitHub repo**: Old C code base with an RGN parsing tool and what seems to be actual (leaked?) Garmin source code.

Disclaimer: This guide is provided for educational and research purposes only. The author is completely unaffiliated with Garmin Ltd. or any of its subsidiaries. Modifying device firmware may void your warranty, brick your device, or cause other unintended consequences. The author assumes no responsibility for any damage, loss of functionality, or other issues that may result from following this guide. Proceed at your own risk.