Compromising Garmin's Sport Watches

A Deep Dive into GarminOS and its MonkeyC Virtual Machine
Roadmap for Today

• Overview of Garmin’s Sport Watches
• Reconnaissance
• MonkeyC
• Firmware Analysis
• Vulnerabilities
• Demo
• Conclusion
• Future Research Areas
Garmin Revenue Share by Segment

Garmin Forerunner Sport Watches

- 44 different models to date
  - First (model 101) released in 2003
  - Last (model 955) released in 2022
- GPS
- Wrist-based heart rate
- Sensors (running pods, HRM)
- Virtual coach and workouts
- Track activities, cadence, pace
- Built-in apps
2nd in Shipment Revenue Share %

Source: https://www.counterpointresearch.com/global-smartwatch-market-revenue-h1-2020/
Also Issued in the US Military

Why are U-2 jet pilots wearing Garmin satellite navigation smartwatches?

They're useful flight- and pilot-monitoring tools, says the Air Force.

ERIC TEGLER - 3/13/2020, 6:15 PM

The current model U-25 aircraft features an all-glass cockpit, improved sensors, and propulsion systems. But its pilots still wear backup GPS/GLO/NASS-enabled watches, just in case.

...where we looked in photos we shot on the east and west coast among the Hornet and Super Hornet munity we saw Garmin watches being used.
Reconnaissance
Garmin Operating System

- Custom, in-house proprietary OS
- Little to no public information
- Mainly in C (with some C++ for UI layer)
- Supports third-party apps
  - Custom MonkeyC language
  - ConnectIQ Store
Compromising Garmin's Sport Watches
Prior Research

- “A Watch, a Virtual Machine, and Broken Abstractions” (2020)
  - Dionysus Blazakis from Atredis
- Vulnerabilities in MonkeyC opcodes
  - `newa`, `news`, `lgetv`, `lputv`, `dup`
- Piqued my interest
  - How are app files loaded?
  - How are permissions implemented?
  - What are native functions?
Attack Surface

- WiFi
- GPS
- Ant
- Ant+
- BLE
- USB
- Apps
## Attack Surface – Apps

<table>
<thead>
<tr>
<th>Section parsing (entry point, code, data, etc.)</th>
<th>Signature validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource parsing (images, videos, fonts)</td>
<td>Permissions</td>
</tr>
<tr>
<td>App storage</td>
<td>MonkeyC (opcodes, SDK functions)</td>
</tr>
</tbody>
</table>

**Apps**

---

*Compromising Garmin’s Sport Watches*
MonkeyC

Compromising Garmin's Sport Watches

April, 2023
MonkeyC

- “Hello Monkey C!”
- Mix between Java, JS, Python, etc.
- Developed from scratch
- SDK with documentation
- Compiled to bytecode
using Toybox.Graphics;
using Toybox.Lang;
using Toybox.System;
using Toybox.WatchUi;

class SimpleHelloWorldView extends WatchUi.WatchFace {

    function initialize() {
        WatchFace.initialize();
    }

    // Load your resources here
    function onLayout(dc as Dc) as Void {
        setLayout(Rez.Layouts.WatchFace(dc));
    }

    // Update the view
    function onUpdate(dc as Dc) as Void {
        // Get and show the current time
        var clockTime = System.getClockTime();
        var timeString = Lang.format("$1:$2$", [clockTime];
        var view = View.findDrawableById("TimeLabel") as View;
        view.setText(timeString);

        // Call the parent onUpdate function to redraw the view
        View.onUpdate(dc);
    }
}
From Code to PRG File

- MonkeyC code
- Bytecode
- PRG file

monkeyc (implemented in Java)
April, 2023

Compromising Garmin's Sport Watches

Link: https://java-decompiler.github.io/
Compromising Garmin's Sport Watches
Firmware Analysis
Beta Firmware and GCD File Format

• GCD file format
• Unofficial format analysis
  • By Herbert Oppmann

Garmin GCD Firmware Update File Format
Filename extension *.gcd
This documentation is based on own research and the sources listed in the references section.

Basic data types
All values are serialized in little-endian byte order (least significant byte first).

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1</td>
<td>ASCII character (see [6])</td>
</tr>
<tr>
<td>byte</td>
<td>1</td>
<td>8 bit unsigned integer (range 0 .. 255)</td>
</tr>
<tr>
<td>ushort</td>
<td>2</td>
<td>16 bit unsigned integer (range 0 .. 65535)</td>
</tr>
<tr>
<td>uint</td>
<td>4</td>
<td>32 bit unsigned integer (range 0 .. 4294967295)</td>
</tr>
</tbody>
</table>
Binwalk Entropy Analysis

Forerunner 245 Music 8.09 Beta
(Model released in 2019)

Forerunner 945 8.09 Beta
(Model released in 2021)
Reverse Tips – Teardown

- Search the FCC ID online
- https://fccid.io/IPH-03568
  - Unfortunately, the one we’re interested in seems to be the shiny one we can’t read

- I supposed that it ran a Cortex M3
  - Same as Forerunner 235 Music
  - (NXP Kinetis K8x MCU family)*
Compromising Garmin’s Sport Watches
Vulnerabilities
Kaitai Structure for PRG

- Kaitai Structure
- Kaitai Web IDE
- Easy to describe file format
- Compile to C, C#, Go, Java, Python, Ruby, etc.

https://github.com/anvilsecure/garmin-ciq-app-research/blob/main/ciq.ksy
Compromising Garmin’s Sport Watches
Vulnerabilities
How are app files loaded?
Resources

• Possible to embed resources
  • Strings, images, fonts, and others
• Compiled into PRG
• Available at run time

```javascript
function initialize() {
    font = WatchUi.loadResource(Rez.Fonts.myFont);
}
```
# String Resources

<table>
<thead>
<tr>
<th>Index</th>
<th>Size</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>2</td>
<td>Length</td>
</tr>
<tr>
<td>0x04</td>
<td>1 * Length + 1</td>
<td>UTF-8 data</td>
</tr>
</tbody>
</table>

```c
#define __DATA_ENTRIES__

const struct __DATA_ENTRIES__ {
    #define __ENTRY__
    uint8_t __ENTRY__[6];
} __sentry__ = {0x1, 0x1};

const struct __DATA_ENTRIES__ {
    #define __ENTRY__
    uint8_t __ENTRY__[6];
} __dataEntry__ = {0x6, 0x6};
```

```
14 [DataEntries]
- sentinel = 0x1 = 1
- dataEntry [StringDef]
  - length = 0x6 = 6
  - data = onMenu
```

```
01 00 06 6f 6e 4d 65 6e 75 00
```

```
e_tvm_error TVM:vm:tvmsmopcode_news(s_tvm_ctx *ctx)
{
    // [...]
    tvm_value_load_string(ctx, (uint*)ctx->pc_ptr, ctx->stack_ptr);
    // [...]
}
```

```
e_tvm_error tvm_value_load_string(s_tvm_ctx *ctx, uint tvmaddr_str, void *str_value_out)
{
    // [...]
    ret = tvm_tvmaddr_to_ptr(ctx, tvmaddr_str, ptr_str);
    if (ret == SUCCESS) {
        ret = tvm_string_def_to_value(ctx, ptr_str, str_value_out, 1);
    }
    return ret;
}
```
## Virtual to Physical Pointers

<table>
<thead>
<tr>
<th>Virtual Pointer</th>
<th>Physical Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start</strong></td>
<td><strong>End</strong></td>
</tr>
<tr>
<td>0x00000000</td>
<td>0x10000000</td>
</tr>
<tr>
<td>0x10000000</td>
<td>0x20000000</td>
</tr>
<tr>
<td>0x20000000</td>
<td>0x30000000</td>
</tr>
<tr>
<td>0x30000000</td>
<td>0x40000000</td>
</tr>
</tbody>
</table>
Loading Strings

String definition

11 22 33 44 55 66 77 88 99

01|CA FE|AA BB CC DD

?? ?? ??

OOB read
CVE-2023-23301
Font Resources

<table>
<thead>
<tr>
<th>Index</th>
<th>Size</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>4</td>
<td>Sentinel</td>
</tr>
<tr>
<td>0x04</td>
<td>4</td>
<td>Height</td>
</tr>
<tr>
<td>0x08</td>
<td>4</td>
<td>Glyph count</td>
</tr>
<tr>
<td>0x0C</td>
<td>4</td>
<td>Min height</td>
</tr>
<tr>
<td>0x10</td>
<td>2</td>
<td>Data size</td>
</tr>
<tr>
<td>0x12</td>
<td>3 * Glyph count</td>
<td>Glyph table buffer</td>
</tr>
<tr>
<td>n</td>
<td>4</td>
<td>Glyph sentinel</td>
</tr>
<tr>
<td>n + 4</td>
<td>1 * Data size</td>
<td>Extra data buffer</td>
</tr>
</tbody>
</table>

- Glyph count: \(0x4000001A\)
- Font data size: \(0x108\)
- Computed size: \(0x1000001a4 = 0x1a4\)

CVE-2023-23305
Vulnerabilities
How are permissions implemented?
## Permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Applicable Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant</td>
<td>Toybox.Ant</td>
</tr>
<tr>
<td>Background</td>
<td>Toybox.Background</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Communications</td>
<td>Toybox.Communications</td>
</tr>
<tr>
<td></td>
<td>Toybox.Authentication</td>
</tr>
<tr>
<td>PersistedContent</td>
<td>Toybox.PersistedContent</td>
</tr>
<tr>
<td>Positioning</td>
<td>Toybox.Position</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>SensorHistory</td>
<td>Toybox.SensorHistory</td>
</tr>
<tr>
<td>SensorLogging</td>
<td>Toybox.SensorLogging</td>
</tr>
<tr>
<td>UserProfile</td>
<td>Toybox.UserProfile</td>
</tr>
</tbody>
</table>

- XML manifest
- Compiled into entry in permissions section
- Checked at run time
Symbol Resolution

API data section
- Class Def Graphics:
  - ...
- Class Def Position:
  - ...
- Class Def Communications:
  - Module ID
  - Need permission
  - ...
  - Field Def openWebPage:
    - Type
    - Virtual Pointer
    - ...

API code section
- spush Toybox.Communications
getm
- spush openWebPage
getv
- invoke

- encodeURL:
  - 00 11 22 33 ...
- openWebPage:
  - AA BB CC DD ...
- ...

Compromising Garmin's Sport Watches
Class and Field Definitions

- Module ID refers to our object
- Field value is the virtual pointer
  - \(0x10\) → PRG code section
- Symbol value passed to `spush`
  - \(0x800018\) → `<init>`
- Value type
  - \(0x6\) → Method
- Flag for permission required
Checking Permissions

- Iterate through PRG permissions list
  - If there is a match, authorized
  - Otherwise, denied

- Permissions checked:
  - getm
  - getv
  - putv

CVE-2023-23304
Bypassing Permissions

CVE-2023-23299

Compromising Garmin’s Sport Watches
Vulnerabilities

What are native functions?
Native Functions

- SDK functions can be implemented
  - In MonkeyC bytecode
  - In native functions
- 460 native functions identified
  - All implemented in C
  - Graphics, Ant/Ant+, BLE, HTTP, encryption, storage
Compromising Garmin's Sport Watches

Toybox.Cryptography.Cipher.initialize()

```c
// [...] byte static_key_buffer [36];
ushort key_data_length;
// [...] tvm_object_get_attribute(ctx, &options, symbol_key, key)
// [...] tvm_object_get_bytarray_data(ctx, key, &bytearray_data); 
memcpy(static_key_buffer, bytearray_data + 1, (uint)key_data_length);
// [...] if (cipher_options == CIPHER_AES128) {
expected_key_size = 0x10;
} else if (cipher_options == CIPHER_AES256) {
expected_key_size = 0x20;
}
// [...] if (key_data_length != expected_key_size) {
    throw_exception(ctx,
                    object_InvalidOptionsException,
                    "Invalid length of :key for requested cipher.")
    return err;
} // [...] 
```
Toybox.Ant.BurstPayload.add()

```c
1  e_tvm_error native:Toybox.Ant.BurstPayload.add(s_tvm_ctx *ctx, uint nb_args)
2  {
3      // [...]
4      tvm_get_field_size_as_int(ctx, object, &size);
5      if (0x1fff < (int)size) {
6          return OUT_OF_MEMORY_ERROR;
7      }
8      // [...]
9      tvm_message_copy_payload_data(ctx, ctx->frame_ptr + 10, data);
10     // [...]
11     tvm_get_field(ctx, strBurstDataBlob, &burstDataBlob);
12     burstDataBlob[size + 0xc] = data[0:4];
13     burstDataBlob[size + 0x10] = data[4:8];
14     // [...]
15  }
```

CVE-2023-23306
Two for One

```javascript
class MyBurstPayload extends Ant.BurstPayload {
    function initialize() {
        Ant.BurstPayload.initialize();
        self.size = 0xdeadbeef;
    }
}

var burst = new MyBurstPayload();

var data = new[8];
for (var j = 0; j < 8; j++) {
    data[j] = 0x44;
}

burst.add(data);
```
Compromising Garmin’s Sport Watches

April, 2023
Exploiting CVE-2023-23300

Forerunner 55 4.10 Beta
(Model released in 2021)
Exploiting CVE-2023-23300

https://github.com/anvilsecure/garmin-ciq-app-research/tree/main/demo
Conclusion
Results

• Analysis performed on Garmin Forerunner 245 Music
• Focused on its Virtual Machine executing applications
• 14 vulnerabilities reported to Garmin
  • Bypass permissions
  • Hijack execution flow
• Over 100 affected models
  • Including fitness watches, outdoor handhelds, and GPS for bikes
  • Multiple vulnerabilities since CIQ API version 1.0.0 published in 2015
Published Resources

- https://github.com/anvilsecure/garmin-ciq-app-research

README.md

Garmin Forerunner 245 Music

This repository contains information related to Anvil’s research project on Garmin Forerunner 245 Music firmware:

- advisories/: Advisories for the multiple vulnerabilities.
- ciqpy/: Python script to manipulate CIQ apps/PRG files.
- demo/: Demo exploiting CVE-2023-23300
- poc/: Proof-of-concept CIQ apps/PRG files for the multiple vulnerabilities.
- ciq.ksy: The Kaitai Structure for parsing CIQ apps/PRG files.
Coordinated Disclosure

- **2022-07-25**: Anvil submitted the technical report to Garmin via their web form along with our 90-day disclosure policy.
- **2022-09-11**: Garmin acknowledges the vulnerabilities and requests an extension until December 3rd, 2022. We agree.
- **2022-10-14**: Anvil submitted a second technical report regarding the permission bypass.
- **2022-11-09**: Garmin states that they are on track for December 3rd, 2022 for the initial findings. Garmin acknowledges the permission bypass and requests an extension until February 28th, 2023. We agree.
- **2022-12-01**: Garmin states that they identified additional affected products and requests a new extension until March 14th, 2023 for all vulnerabilities.
- **2022-12-06**: Anvil agrees on the new deadline and requests the list of affected products.
- **2022-12-13**: Garmin provides the list of affected devices, identified by Connect IQ API version
- **2023-01-09**: Anvil requests CVE IDs.
- **2023-01-26**: MITRE assigns CVE IDs (CVE-2023-23301, CVE-2023-23298, CVE-2023-23304, CVE-2023-23305, CVE-2023-23302, CVE-2023-23303, CVE-2023-23306, CVE-2023-23300, CVE-2023-23299).
- **2023-01-27**: Anvil shares CVE IDs with Garmin and asks if they are planning on publishing a security advisory.
- **2023-02-01**: Garmin states that they are not planning to publish an advisory listing the CVEs.
- **2023-03-14**: Anvil asks Garmin if they have released the new versions for the affected devices.
- **2023-03-16**: Garmin states that the majority of the updates have been released. They specify that three devices have been delayed and that they are targeting March 22nd, 2023.
Future Research Areas
Scratched the Surface

- Ant and Ant+
- BLE
- WiFi
- GPS
- USB
- Notifications
- Signature
Focused on Static Analysis

- Fuzzing
  - Hardware setup?
  - QEMU patch?
- Debugging
Questions?