info@apriorit.com
www.apriorit.com

Case study

USB WiFi Driver Development

Our client is a US-based company that provides cybersecurity solutions for IT companies. While working on a network security analysis solution, they approached us with the task of Windows USB WiFi driver development.

The client already had a Linux driver and wanted to develop a similar one for Windows. After thorough research, our team decided to port the functionality of the existing Linux driver instead of building a Windows driver from scratch. We delivered a solution that met all the client's requirements and fit within the deadline.

The client

Our client is a cybersecurity company that provides solutions and technologies for protecting networks against industrial espionage and cyber attacks. They also offer security audits for websites, mobile applications, and WiFi networks.

The challenge

Before contacting Apriorit, our client already had a Linux driver for one of their solutions for analyzing traffic and detecting cybersecurity threats in WiFi networks. They wanted to build a similar driver for Windows and delegated this challenging task to our team because of our vast experience in kernel and driver development.

The client wanted both drivers to have a similar set of functionality, including the ability to:

- listen to channels and intercept packets with Radiotap headers
- inject random packets into traffic
- measure noise levels in a channel.

Our solution

After analyzing the resources for the existing Linux driver, we offered to port its functionality to a custom Windows driver instead of building a Windows driver from scratch.

We implemented part of the Linux kernel as a user mode library for Windows to get the most functionality out of the existing driver. Since standard Windows drivers don't have all the functionality our client's driver required and their code is closed, we also created a high-level API in C# to allow our client's solution to communicate with the Windows driver.

In addition to providing professional developers and quality assurance (QA) specialists, Apriorit provided our client with a business analyst and project manager.

The result

Thanks to our choice of development strategy, we delivered a ready solution in four months instead of the eight months that it usually takes to develop a new Windows driver from scratch. Our team prepared a WiFi USB driver for Windows, ported the existing Linux driver functionality to it, and built a high-level API within the estimated time and budget.

Project requirements

The major requirement for this project was to build a custom Windows driver that offered the functionality of the client's existing Linux driver.

The Apriorit team needed to develop a Windows driver for inspecting packets and injecting packets into WiFi networks. The driver's main functionality was supposed to be used only for analysis and, thus, the driver didn't need to be integrated into the Windows Network Driver Interface Specification (NDIS) driver or modify it in any way. Also, our team created an API written in C# that enables our client's solution to communicate with the created driver.

After becoming familiar with all the project requirements, the Apriorit team proceeded to choosing the most suitable development approach.

Our approach

To develop a USB WiFi driver, we considered two options: build a new Windows driver with all the requested functionality from scratch or port the functionality of the existing Linux driver.

Our team started with analyzing the Linux driver sources and specifications for the WiFi chipset used by the client. Since the sources were large (several megabytes of code) and the chipset specifications were closed, we decided not to build the Windows driver from scratch. We roughly estimated that building the driver from scratch would take about eight months, which was too long for both our team and the client.

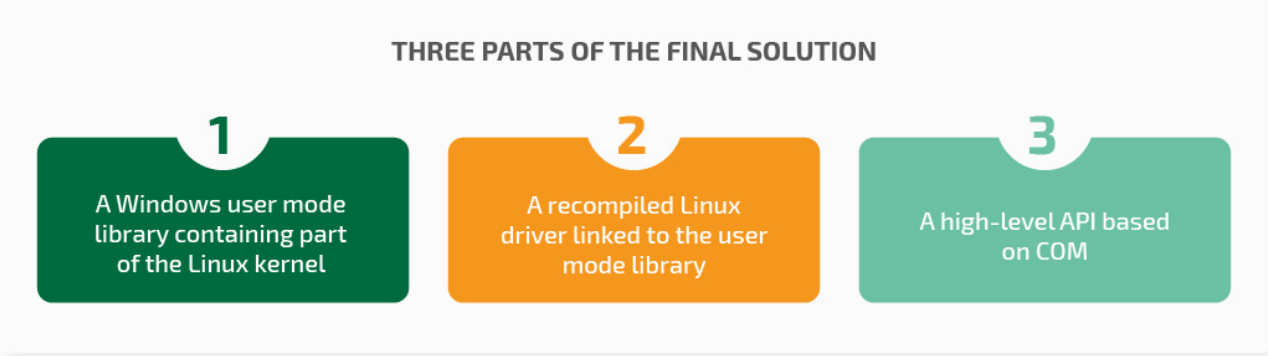
Therefore, we settled on the second option — porting the existing Linux driver to Windows and reusing its functionality to the extent possible. Our team estimated this process at about four months, which was half the estimate for the first option.

When developing the requested driver, our team concentrated on the following three tasks:

Implementing part of the Linux kernel as a user mode library for Windows. This library allowed us to use the client's Linux driver as is.

Recompiling the existing Linux driver and linking it to the created user mode library.

Developing a high-level API based on the Component Object Model (COM) that would allow clients of this API to be written in different languages and operate in different processes.



We combined the resulting components in a single Windows service. This service is responsible for detecting WiFi USB devices and switching their drivers between the standard drivers and our WinUSB driver.

Scope of work

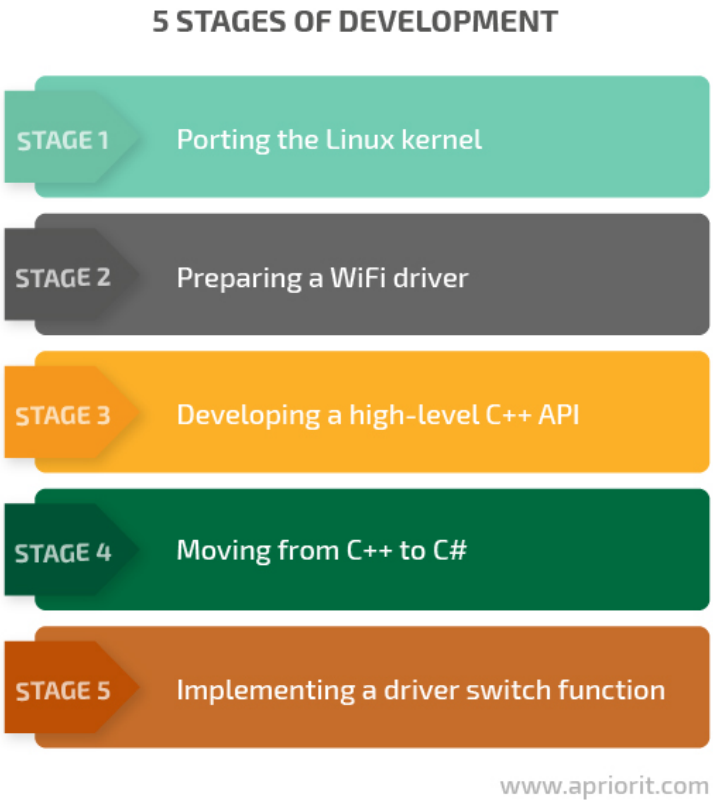
Once we agreed on this approach with the client, we established transparent communication with the client and got down to work.

During the USB WiFi driver development process, our team worked with different technologies and tools:

- C++
- C#

- WinUSB
- COM
- ATL
- CMake
- GoogleTest
- Wireshark
- And more

We divided the project scope into five stages:



Stage 1. Porting the Linux kernel

This stage was a crucial part of the custom USB WiFi driver development process because it allowed us to use the existing Linux WiFi driver on Windows.

First, we created a list of Linux kernel functions required for the WiFi driver and ported them to a Windows library. This list included about 150 functions from 20 categories.

We also created tests using Google Test to check the operability of these functions.

Stage 2. Preparing a WiFi driver

Once we created the library with all required Linux kernel functions, it was time to compose our Windows driver. This process included recompiling the driver for Windows and linking it to the library we had just created.

To do this, we had to resolve the differences between the GNU Collection Compiler (GCC) and Microsoft Visual C++ Compiler (MSVC). We also had to move from the Makefile build system to CMake.

At this point, the main part of the project was ready: we had the driver with all the required functionality that could be used on Windows 10 systems.

Stage 3. Developing a high-level API in C++

Since our client's solution for network traffic analysis needed to interact with our WiFi driver, the driver required an API. We created a high-level API and used it to implement several test applications in C++.

The two largest of these applications were:

- **an application for intercepting packets on the test channel.** This application demonstrated one of the driver's main functions: listening to channels and intercepting and analyzing packets. We also added the pcap library, which allowed us to store intercepted packets in the pcap format and analyze them with Wireshark.
- **an application for packet injection.** This application allowed us to inject beacon packets with information on a fake WiFi access point.

At this stage, our QA engineers performed full testing of the developed API. They used

the provided test applications and compared the results to results from the original Linux driver.

During these tests, we checked the following aspects:

- Correspondence of the developed Windows driver functionality to the client's requirements
- The API's capability to perform all actions required by the client
- The speed, performance, and resource consumption of the solution

Stage 4. Moving from C++ to C#

Our client's network analysis solution was written in C#. Therefore, we needed to change our API so it could be managed using C# instead of C++.

To do this, we described the COM interface in the IDL file and added it to our Windows service. After that, just like in stage 3, we created test applications in C# and tested the driver functionality.

Stage 5. Implementing a driver switch function

During the final stage, we implemented an API function for automatically switching the WiFi

device from the original driver to our custom WinUSB driver and back.

During previous stages, this switch could only be made manually. We decided not to add the switch function at earlier stages because we wanted to implement and verify the driver's main functionality before moving further.

Additionally, we created an INF file for the client's USB WiFi device using the defined Vendor ID and Product ID. We also passed the Windows Hardware Lab Kit tests to receive the Windows Hardware Quality Lab (WHQL) signature from Microsoft. This provided the client with an opportunity to automatically manage USB WiFi drivers on all endpoints without any additional interactions with end users.

Handling project challenges

The biggest challenges we faced while working on this project appeared at the stage of porting the Linux driver to Windows:

1. Differences between the GCC and MSVC compilers. The main issue here was the non-standard C++ syntax that worked with one compiler and didn't work with another one. Fortunately, starting from C++ 11, all non-standard extensions are automatically changed to the standard extensions or standard analogs are created, which significantly simplifies code porting. Another issue here was that the two compilers had different sizes for the long type of data. The only possible solution was to check the driver's code and change the long type to other suitable types.

2. Absence of functionality in CMake. When moving from C++ to C#, we needed to develop a special CMake module for IDL compiling. Our team also faced a few difficulties with C# support, as there were not enough examples in the CMake documentation such as for creating imported targets. However, after conducting additional research and experiments, our developers found a solution and successfully accomplished this task.

The impact

Our client has incorporated the developed driver into their solution that provides end users with a user interface and tools for WiFi network analysis. The driver provided by our Apriorit team has significantly enhanced the ability of our client's product to perform WiFi site surveys, check network security and efficiency, and analyze and redesign WiFi infrastructures in hundreds of communities.

If your project requires a custom driver for Windows or Linux, Apriorit dedicated specialists are ready to assist. Contact us to start working on your dream project right away.