Rust: "C ist eine feindselige Sprache"

In modernen Desktop-Betriebssystemen sind Parser allgegenwärtig. Sie verarbeiten ungeprüfte und eventuell nicht vertrauenswürdige Daten meist aus dem Internet und sind damit ein nachvollziehbares Ziel für Angreifer. Das Schreiben von Parsern in der Programmiersprache C sei allerdings berüchtigt dafür, Fehler zu machen, sagt der Mitbegründer des Gnome-Projekts Federico Mena Quintero auf der Guadec-Konferenz. Um diese Fehler zu vermeiden, empfiehlt er stattdessen die Verwendung der Sprache Rust.

Diese Empfehlung von Mena basiert auf den Erfahrungen, die er im vergangenem Jahr als Betreuer von Librsvg gesammelt hat, welche zum Darstellen (Rendern) von SVG-Dateien genutzt wird. Verwendet wird die Bibliothek für eine Vielzahl von Desktop-Anwendungen, so ist Librsvg etwa eine Abhängigkeit des Toolkits GTK. Zudem wird die Bibliothek in Wikipedia zur Darstellung genutzt, falls sehr alte Browser zum Einsatz kommen, die SVG noch nicht selbst darstellen können.

Mena hat die Pflege von Librsvg nach ein paar Bug-Fixes im Jahr 2015 übernommen und dann im Herbst 2016 damit begonnen, kleine Code-Bestandteile in Rust neu zu erstellen, zunächst, um sich die Sprache beizubringen. Der Aufbau von Rust und die große Interoperabilität zu C habe es dabei ermöglicht, leicht stückweise Code auszutauschen. Schließlich sei Mena an dem Punkt angelangt, an dem der Entwickler den gesamten Code nach Rust portieren wollte.

Rust hilft beim Fehlervermeiden

Ein große Hilfe für Mena sei dabei gewesen, wie in Rust mit Fehlern umgegangen werde: So könnten Fehlerausgaben nicht nur sehr einfach umgesetzt werden, das Schreiben von Fehlerbehandlungen sei zudem viel kürzer, als Fehler schlicht zu ignorieren. *"Das ist einfach magisch"*, sagte Mena. Ebenso sei das Erstellen von Unit-Tests sowie von Debug-Nachrichten mit Rust einfacher als mit C.

Bei den Portierungsarbeiten seien Mena eine Vielzahl von eigentlich üblichen Fehlern in dem C-Code aufgefallen wie etwa Überläufe, die jedoch leider nicht immer offensichtlich sind. Der Code für einige dieser Fehler stamme zudem von den laut Mena besten C-Programmierern, die das Gnome-Projekt je hatte. Das bedeute aber keinesfalls, dass das Projekt über keine guten Programmierer verfüge. Der Grund für diese Fehler seien eben nicht Menschen, sondern vielmehr die Sprache C selbst, die schlicht *"feindselig"* sei.

Besonders treffe das auf das Schreiben von Parsern zu, die in vielen grundlegenden Bibliotheken wie eben Librsvg vorkommen. Eines der größten Probleme von C sei in diesem Einsatzbereich das Verarbeiten von Strings, die kein nativer Datentyp

der Sprache sind. Anders ist dies in Rust, was Mena wieder mit einem Codebeispiel beschreibt, in dem wiederum auch die einfache Umsetzung einer Fehlerbehandlung deutlich wird. Der ebenfalls anwesende Google-Angestellte und Sicherheitsforscher Matthew Garrett beschreibt auf Twitter jenes Beispiel einfach nur mit dem Wort *"überzeugend"*.

Zusätzlich zu den bereits genannten Eigenschaften von Rust, kommen noch jene neuen Funktionen hinzu, die explizit darauf ausgelegt sind, die Sicherheit des geschriebenen Codes zu erhöhen. Dazu gehört etwa das Ownership-Prinzip, der Borrowing-Checker oder andere Bestandteile der Sprache, die zur Kompilierzeit erzwungen werden, um Fehler in laufendem Code zu vermeiden.

Mena sagt, er habe selbst etwa einen Monat lang gegen den Borrow-Checker gekämpft, bis seine Rust Kenntnisse gut genug waren, um Code zu schreiben, der problemlos kompiliere. Den Anwesenden gibt Mena dennoch den Rat: *"Kämpft nicht gegen den Compiler, er ist euer Freund"* - immerhin schützt der Rust-Compiler vor vielen Fehlern, die in C auftreten können.