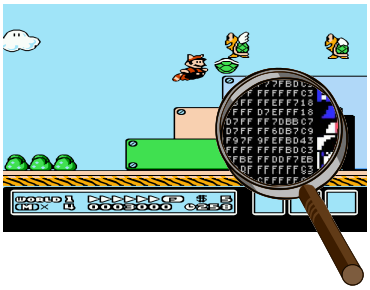
Rolf Drechsler, Cornelia Große,
Jannis Stoppe und Kenneth Schmitz

Optimierte, zertifizierte bis verifizierte Software: Schön und Gut! Aber was passiert, wenn sie niemand (mehr) versteht? Wir stellen uns kompilierter, obfuskierter oder einfach gemeiner Firmware im Binärformat in Form von Retro-Spielen und retten Link, Mario und Luigi anstelle ihrer Prinzessinnen aus den Fängen gemeiner Softwarefehler und Bugs.

Beschreibung



Viele Systeme des modernen täglichen Lebens verwenden Software, die sie steuern. Oftmals beginnt das bereits morgens mit dem Drehen des Zündschlüssels im Auto auf dem Weg zur Arbeit oder bei dem Druck auf einen Knopf des Kaffeeautomaten zum Wachwerden. Eingebettete Systeme (mit oder ohne Betriebssystem) steuern den Alltag mit zunehmender Verbreitung. Selbst die Apollo-Missionen zum Mond benötigten (nach heutigen Maßstäben minimale) Computerunterstützung, um das Ziel zu erreichen.

Je kleiner (und damit stärker integrierter) Systeme werden, desto effizienter, kleiner und sparsamer muss die ausgeführte Software sein. Moderne Compiler helfen dabei in der Regel sehr: `$ g++ myLittleProgram.C -O3 -o myOptimizedLittleBinary.out`. Leider erzeugen Optimierungen (manchmal zum Schutz von intellektuellem Eigentum sogar Obfuskation) unlesbaren Code, der eine weitere Anpassung und Wartung schwierig bis unmöglich macht. Gleiches gilt für Java ByteCode.

Die Analyse und Erstellung dieser Firmware ist besonders interessant, weil sie an der Schnittstelle zwischen Hard- und Software liegt. Ein detailliertes Wissen über das Wirken und die Möglichkeiten zur Erstellung der Firmware geht zwangsweise einher mit einem Verständnis über die zugrunde liegende Hardware sowie das damit zu realisierende System. Die Abwesenheit komplexer Zwischenschichten wie einem Betriebssystem führt dazu, dass die Software Hand in Hand mit den vorhandenen Chips (ganz unten) sowie dem Nutzer (ganz oben) arbeiten.

Die Relevanz des Themas ergibt sich aus den aktuellen Trends des „Internet of Things“ (IoT), Smart-Home Geräten (z.B. Philips Hue oder Waschmaschinen mit WLAN) aber

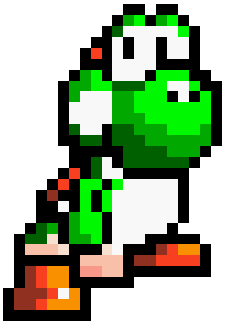
auch im sicherheitskritischen Automobilbereich, wo eine Vielzahl eingebetteter Systeme miteinander in Echtzeit kommunizieren. Jedes dieser Systeme soll günstig sein, jedoch stets zuverlässig und fehlerfrei funktionieren. Um sich dieser Art von Systemen zu nähern, bietet sich ein nahezu antikes Feld an: bei genauer Betrachtung sind die Cartridges der ersten Heimkonsolen lediglich eine Firmware, die von dem eingebetteten System in der Konsole ausgeführt wird. Mögliche Tracks, die von den Studierenden bearbeitet werden sollen, sind:

- **Firmware schreiben**: Firmware wird häufig auf niedrigen Abstraktionsebenen geschrieben. Insbesondere für alte 8- und 16-Bit-Generation-Systeme wurden Programme meist direkt in Assembler entwickelt. Die Erstellung von Back-Ends für modernere Hochsprachen würde die Neuentwicklung von Retro-Firmware für die Originalhardware deutlich vereinfachen.
- **Firmware analysieren**: Sowohl bestehende als auch eventuell im Projekt exemplarisch neu entwickelte Firmware muss fehlerfrei sein - das nachträgliche Patchen ist häufig nicht ohne weiteres möglich oder aber sehr komplex. Die Anwendung moderner Verifikationsverfahren soll Fehler sowohl während als auch nach der Entwicklung unterstützen.
- **Systementwicklung**: Bei Bedarf können Hardwarebestandteile oder -systeme entwickelt werden, welche die alten Systeme erweitern oder neue Hardwareplattformen für die alte Firmware bieten.

Die tiefgehende Analyse (bis hin zur Verifikation) der ausgeführten Software (und den Zwischenprodukten) ist ein integraler Bestandteil für den Entwurf fehlerfreier Systeme. Die Studierenden des SystemXrunner-Projektes sollen ein umfassendes System- und Systemdesign-Verständnis erlangen. Das Projekt orientiert sich somit sowohl am Profil für Sicherheit und Qualität (SQ) von Soft- und Hardwaresystemen als auch an Künstliche Intelligenz, Kognition und Robotik (KIKR) durch die verwendeten Lösungsalgorithmen.

Weitere Informationen

Wer ein tiefes Verständnis für eingebettete Software, sowie den modernen Systementwurf in hardwarenahem Kontext erhalten möchte und / oder mehr über neuartige und effiziente Verfahren für automatische Analyse von Computerspielen bzw. Programmanalyse erfahren möchte, ist bei SystemXrunner der AG Rechnerachitektur richtig aufgehoben.



Grundsätzlich bewegt sich das Projekt in ähnlichen Bereichen wie Praktische Informatik 1/2 und Technische Informatik 1 mit einem Schwerpunkt in systemnaher Software und Reverse-Engineering. Wer Spaß an kniffeligen Fragestellungen, KI-, sowie Learning-Algorithmen und formalen Lösungstechniken hat, wird auch im SystemXrunner-Projekt ähnliche Problemstellungen wiederfinden und mit entwickeln. Das Projekt ist für Studierende aus den Bachelorstudiengängen der Informatik, Digitalen Medien und Systems Engineering offen. Als vorbereitende und begleitende Lehrveranstaltung werden die Veranstaltungen der Arbeitsgruppe Rechnerarchitektur, insbesondere „Qualitätsorientierter Systementwurf“ (SoSe), „Rechnerarchitektur und Eingebettete Systeme“ (WS) sowie „Praktische Einführung in den modernen Systementwurf mit C++“ (WS) empfohlen, aber nicht verpflichtend.