Kianxali: Kieler Analyzer for Executables and Libraries

Folke Will, Christian-Albrechts-Universität zu Kiel

fwi@informatik.uni-kiel.de

Es gibt Situationen, in denen das Verhalten von Programmen analysiert werden soll, ohne dass der Quellcode des Programms verfügbar ist. Dies ist beispielsweise bei der Analyse von Schad-Software wie Viren, Root-Kits oder Bot-Netzen der Fall. Zur Analyse von Programmen, die nur in Maschinencode vorliegen, wird häufig ein *Disassembler* verwendet. Ein Disassembler übersetzt den Maschinencode in Assembler-Code und ermöglicht dadurch eine manuelle Analyse des Programmverhaltens. Frei verfügbare Disassembler wie `objdump` analysieren den Maschinencode in einem einzelnen Durchgang, indem die ausführbaren Segmente eines Programms Byte für Byte sequenziell analysiert werden. Der Nachteil dieses Ansatzes ist, dass der Disassembler nicht mehr synchron zum Maschinencode arbeitet, falls in den ausführbaren Segmenten auch Daten vorkommen, die keine Instruktionen darstellen – dies ist beispielsweise bei Tabellen für switch-case-Anweisungen der Fall. Der Disassembler interpretiert diese Daten als Instruktionen, wodurch auch Anweisungen hinter der Tabelle falsch interpretiert werden können, da die Codierungen von Instruktionen unterschiedliche Längen haben. In Schad-Software werden oftmals absichtlich sogenannte *Junk-Bytes* eingefügt, um Fehler in Disassemblern zu provozieren und die Analyse zu erschweren.

Kommerzielle Disassembler wie *IDA Pro* verwenden einen rekursiven Ansatz, bei dem die Instruktionen ab dem Einstiegspunkt analysiert werden. Dem Disassembler ist die Semantik der Instruktionen bekannt, er kann bei Verzweigungen allen Pfaden folgen und bei unbedingten Sprüngen mögliche Daten hinter der Anweisung ignorieren oder als Daten interpretieren. Dieses Vorgehen entspricht einem rekursiven Durchlaufen des Kontrollflussgraphen. Auf diese Weise werden Daten nicht fälschlicherweise als Instruktionen interpretiert. Der Nachteil hierbei ist allerdings, dass Funktionen, die nur über Funktionszeiger angesprochen werden, nicht gefunden werden, da sie nicht mit einer direkten Adresse angesprungen werden und somit im Kontrollflussgraph nicht unmittelbar erreichbar sind.

Mit Kianxali wurde ein freier Disassembler nach dem rekursiven Ansatz entwickelt, der über Heuristiken zum Aufdecken von indirekt aufgerufenen Funktionen verfügt. Auf diese Weise werden die Vorteile beider Disassembler-Ansätze kombiniert. Konstrukte aus Hochsprachen wie Funktionen und switch-case-Anweisungen werden teilweise erkannt, um die Analyse weiter zu vereinfachen. Der Benutzer kann das Assembler-Listing durch Kommentare annotieren und Funktionen mit Namen versehen, um die Lesbarkeit des Assembler-Codes zu verbessern. Zusätzlich verfügt Kianxali über eine Ruby-Schnittstelle zur skript-gesteuerten Analyse des Codes. Es wurden beispielhafte Skripte entwickelt, die Verschleierungen im Maschinencode wieder aufheben können.

In diesem Vortrag wird Kianxali als Werkzeug vorgestellt, mit dem das Verhalten von Programmen ohne das Vorliegen ihres Quellcodes analysiert werden kann. Nach einer Einführung und Darstellung der Motivation des Themas werden typische Situationen einer solchen Analyse mit praxisnahen Beispielen gezeigt.