Software verstehen, zerstören, schützen mit automatischen Software-Modellen

Andreas Zeller[1]

Zusammenfassung

Modelle und Spezifikationen sind die Grundlage jeder vernünftigen Software-Entwicklung. Was aber, wenn man Modelle und Spezifikationen aus bestehenden Programmen ableiten könnte? Unsere aktuellen Arbeiten nehmen ein Programm und einige Beispieleingaben und leiten automatisch *Grammatiken* ab, die *Eingabe- und Ausgabeformat* beschreiben. So kann unser AUTOGRAM-System etwa für die Java-URL-Klasse aus den folgenden Beispiel-URLs

```
http://user:password@www.google.com:80/command?
    foo=bar&lorem=ipsum#fragment
http://www.guardian.co.uk/sports/worldcup#results
ftp://bob:12345@ftp.example.com/oss/debian7.iso
```

diese *Eingabegrammatik* ableiten, die recht anschaulich die Struktur der URL erfasst:

```
URL ::= PROTOCOL '://' AUTHORITY PATH ['?' QUERY] ['#' REF]
AUTHORITY ::= [USERINFO '@'] HOST [':' PORT]
PROTOCOL ::= 'http' | 'ftp'
USERINFO ::= /[a-z]+/ ':' /[a-z]+/
HOST ::= /[a-z.]+/
PORT ::= '80'•
PATH ::= /\/[a-z0-9.]*/•
QUERY ::= 'foo=bar&lorem=ipsum'
REF ::= /[a-z]+/
```

Wir erzeugen diese Grammatiken, indem wir den dynamischen Datenfluss eines jeden Zeichens verfolgen, und Zeichen, die in die gleiche Funktion oder Variable fließen, zu syntaktischen Einheiten zusammenfassen – die wir auch nach dem Funktions- oder Variablennamen benennen können. Da etwa „`http`“ und „`ftp`“ in eine Variable namens „`protocol`“ fließen, können wir direkt die entsprechende Regel ableiten. Dies sorgt für sehr lesbare Grammatiken, wie die obige unbearbeitete Rohausgabe unseres AUTOGRAM-Werkzeugs demonstriert.

Grammatiken wie diese helfen unmittelbar, Datenformate zu verstehen. Sie ermöglichen vollautomatisches massives Sicherheitstesten auf einer Vielzahl von Systemen, indem sie die Produktion syntaktisch korrekter Eingaben unterstützen. Diese können so tief ins Programm vordringen, um dort Fehler zu finden – insbesondere Sicherheitslücken. Da

[1] Universität des Saarlandes, Saarbrücken, Lehrstuhl für Softwaretechnik, zeller@cs.uni-saarland.de

sowohl das Lernen der Grammatik wie auch ihre Anwendung in der Testgenerierung vollautomatisch sind, ergeben sich hier viele neue Möglichkeiten – für Entwickler wie für Angreifer. Auf der anderen Seite ermöglichen erlernte Grammatiken aber auch Schutz vor illegalen Eingaben, indem nicht konforme Ein- oder Ausgaben abgeblockt werden – Techniken, die ich anhand konkreter Beispiele demonstriere.

Besuchen Sie unsere Webseite, um mehr über unsere aktuellen Projekte zu erfahren:

`http://www.st.cs.uni-saarland.de/`

Andreas Zeller *ist seit 2001 Professor für Softwaretechnik an der Universität des Saarlandes in Saarbrücken. Seine* ***Forschung*** *beschäftigt sich mit der Analyse großer Software-Systeme und ihre Entwicklungsgeschichte. In 2010 wurde Zeller zum* ***Fellow der ACM*** *ernannt für seine Beiträge zur automatischen Fehlersuche und der Analyse von Software-Archiven, für die er auch jeweils mit einem 10-Jahres-Impact Award der ACM SIGSOFT und der ICSE ausgezeichnet wurde.*

Bereits 2012 hatten die Saarbrücker Holler, Herzig und Zeller das LANGFUZZ-System entwickelt, das Grammatik-basiert syntaktisch gültige Javascript-Eingaben für Webbrowser erzeugt. Bis heute hat LANGFUZZ als Teil der Firefox-Entwicklung mehr als 4000 Sicherheitslücken in Firefox aufgedeckt. Die aktuellen Arbeiten sind mit Matthias Höschele entstanden und werden aus einem ERC Advanced Grant für Arbeiten über Specification Mining und Testerzeugung gefördert.