

Seminar Moderne Programmanalysen zur Fehlererkennung

Prof. Erhard Plödereder
Steffen Keul, Stefan Staiger, Gunther Vogel, Eduard Wiebe

1 Organisatorisches

Aktuelle Informationen finden Sie im Web:

http://www.informatik.uni-stuttgart.de/iste/ps/Lehre/WS0708/S_Programmanalysen/

Vortragsdauer: 45-60 Minuten mit anschließender Diskussion

Ausarbeitung: Umfang: 10 – 12 Seiten einschließlich Zusammenfassung und Literaturverzeichnis im zweiseitigen IEEE-Format.

Stilvorlagen für Latex und Word stehen auf der Web-Seite zur Verfügung.

Anwesenheit: Anwesenheit ist Pflicht; bei unentschuldigtem Fehlen droht der Ausschluss.

Hilfsmittel: Beamer und Laptop stehen für den Vortrag zur Verfügung.

2 Themen

Einführung in die Welt der Programmanalysen

Standardtechniken und Formalismen der statischen Programmanalysen:

- Datenflussanalysen und monotone Frameworks
- Abstrakte Interpretation
- Constraint Based Analysis
- Typ-Systeme, Inferenz-Algorithmen

Literatur: [NHH99], [HJ94], [Sch98]

Bearbeiter: Wiltrud Kessler

Betreuer: Eduard Wiebe

Termin: 49. KW

Datenflussanalysen, Prüfen von Variablenverwendungen

Klassische Techniken, die auch im Compilerbau eingesetzt werden, um Zugriffe auf nicht initialisierte Variablen oder verbotene Werte (wie NULL für Zeiger oder eine Division durch 0) zu erkennen. Hierzu zählen: Live-Variable-Analysis, Zeigeranalysen, SSA-Form, Konstantenpropagierung, Out-Of-Bounds-Analyse.

Literatur: [ASU86, Muc97, CFR⁺91, Hin01, PS07]

Bearbeiter: Roland Jacobson
Betreuer: Stefan Staiger
Termin: 50. KW

Fehlererkennung durch Mustersuche

Die meisten Programmierfehler sind trivial. Aus Erfahrungswerten lassen sich Muster für die häufigsten Probleme ableiten. Durch statische Analyse kann nach Instanzen dieser Muster in Quelltexten gesucht werden und somit bereits eine deutliche Verbesserung der Qualität und Korrektheit erzielt werden.

Literatur: [LRY⁺04, HP, HP04, CJ03]

Bearbeiter: Frank Schuh
Betreuer: Steffen Keul
Termin: 51. KW

Sicherheitslücken durch statische Analyse aufdecken

Durch eine Sicherheitslücke gelingt es einem Angreifer ein nicht-beabsichtigtes oder schädliches Programmverhalten auszulösen und so beispielsweise geheime Daten auszulesen oder zu löschen. Mit Hilfe von statischen Analysen können bestimmte Sicherheitslücken aufgedeckt werden. Wie funktionieren solche Werkzeuge und welche Fehler werden erkannt?

Literatur: [CM04, LL05, STFW01, CW02]

Bearbeiter: Steffen Reimann
Betreuer: Gunther Vogel
Termin: 3. KW

Prüfung von Speicherzugriffen, Buffer Overflows

Zugriffe auf null-Zeiger können von modernen Betriebssystemen erkannt und unterbunden werden, sie führen jedoch üblicherweise sofort zu einem erzwungenen Programmende. Als weitaus schlimmer erweisen sich Zugriffe auf beliebigen Speicher durch den die Konsistenz des Programmzustands zerstört wird. In Buffer Overflow-Attacken versucht ein Angreifer gezielt Programmierfehler auszunutzen und den Programmzustand nach seinen Wünschen zu manipulieren.

Literatur: [HSP06, WFBA00, GJC⁺03]

Bearbeiter: Hao Yan
Betreuer: Steffen Keul
Termin: 4. KW

Verifikation von Sequenzbeschränkungen

Verwendungen von Software-Komponenten und APIs unterliegen im allgemeinen bestimmten Sequenzbeschränkungen, so genannten Protokollen. Wie können solche Anforderungen statisch überprüft und Abweichungen aufgedeckt werden?

Literatur: [DLS02, DCCN04, OO92, ECH⁺01]

Bearbeiter: Shaojun Zhang

Betreuer: Gunther Vogel

Termin: 5. KW

Literatur

- [ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullmann. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.
- [CFR⁺91] Ron Cytron, Jeanne Ferrante, Barry K. Rosen, Mark N. Wegman, and F. Kenneth Zadeck. Efficiently computing static single assignment form and the control dependence graph. *ACM Transactions on Programming Languages and Systems*, 13(4):451–490, October 1991.
- [CJ03] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns, 2003.
- [CM04] Brian Chess and Gary McGraw. Static analysis for security. *IEEE Security and Privacy*, 2(6):76–79, 2004.
- [CW02] Hao Chen and David Wagner. Mops: an infrastructure for examining security properties of software. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 235–244, New York, NY, USA, 2002. ACM Press.
- [DCCN04] Matthew B. Dwyer, Lori A. Clarke, Jamieson M. Cobleigh, and Gleb Naumovich. Flow analysis for verifying properties of concurrent software systems. *ACM Trans. Softw. Eng. Methodol.*, 13(4):359–430, 2004.
- [DLS02] Manuvir Das, Sorin Lerner, and Mark Seigle. Esp: path-sensitive program verification in polynomial time. In *PLDI '02: Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation*, pages 57–68, New York, NY, USA, 2002. ACM Press.
- [ECH⁺01] Dawson Engler, David Yu Chen, Seth Hallem, Andy Chou, and Benjamin Chelf. Bugs as deviant behavior: a general approach to inferring errors in systems code. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 57–72, New York, NY, USA, 2001. ACM Press.

- [GJC⁺03] V. Ganapathy, S. Jha, D. Chandler, D. Melski, and D. Vitek. Buffer overrun detection using linear programming and static analysis, 2003.
- [Hin01] Michael Hind. Pointer analysis: Haven't we solved this problem yet? In *PASTE '01: 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, pages 54 – 61, Snowbird, UT, USA, 2001. ACM Press.
- [HJ94] Nevin Heintze and Joxan Jaffar. Set constraints and set-based analysis. In *PPCP '94: Proceedings of the Second International Workshop on Principles and Practice of Constraint Programming*, pages 281–298, London, UK, 1994. Springer-Verlag.
- [HP] David Hovemeyer and William Pugh. Finding concurrency bugs in java.
- [HP04] David Hovemeyer and William Pugh. Finding bugs is easy. *SIGPLAN Not.*, 39(12):92–106, 2004.
- [HSP06] David Hovemeyer, Jaime Spacco, and William Pugh. Evaluating and tuning a static analysis to find null pointer bugs. *SIGSOFT Softw. Eng. Notes*, 31(1):13–19, 2006.
- [LL05] V. Livshits and M. Lam. Finding security vulnerabilities in java applications with static analysis, 2005.
- [LRY⁺04] Yanhong A. Liu, Tom Rothamel, Fuxiang Yu, Scott Stoller, and Nanjun Hu. Parametric regular path queries. In *Proceedings of the ACM SIGPLAN 2004 Conference on Programming Language Design and Implementation*, pages 219–230, Washington, DC, jun 2004. ACM.
- [Muc97] Steven S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann Publishers, San Francisco, 1997.
- [NHH99] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer Verlag, 1999.
- [OO92] Kurt M. Olender and Leon J. Osterweil. Interprocedural static analysis of sequencing constraints. *ACM Trans. Softw. Eng. Methodol.*, 1(1):21–52, 1992.
- [PS07] Erhard Plödereder and Stefan Staiger. *Skript zur Vorlesung Compilerbau und Programmanalysen*. Stuttgart, 2007.
- [Sch98] David A. Schmidt. Data flow analysis is model checking of abstract interpretations. In *POPL '98: Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 38–48, New York, NY, USA, 1998. ACM Press.

- [STFW01] Umesh Shankar, Kunal Talwar, Jeffrey S. Foster, and David Wagner. Detecting format string vulnerabilities with type qualifiers. In *Proceedings of the 10th USENIX Security Symposium*, August 2001.
- [WFBA00] David Wagner, Jeffrey S. Foster, Eric A. Brewer, and Alexander Aiken. A first step towards automated detection of buffer overrun vulnerabilities. In *Network and Distributed System Security Symposium*, pages 3–17, San Diego, CA, February 2000.