Seminar: Secure Systems Engineering

Software Engineering

Introduction – October 10, 2017
Linghui Luo
Requirements

- Attendance at all meetings
- Writing a seminar essay in English
  - 15 – 20 pages in LaTeX
  - LaTeX-template provided by us
- Delivering a presentation in English
  - 20 mins presentation
  - 10 mins discussion
- Active participation in the peer review process
  - Peer reviewed by students
  - Reviewed by advisors
## Schedule

### First Meetings

<table>
<thead>
<tr>
<th>Date</th>
<th>Time &amp; Place</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, 10 October 2017</td>
<td>11:00 – 13:00 E5.333</td>
<td>Introduction to topics &amp; Assignment</td>
</tr>
<tr>
<td>Tuesday, 17 October 2017</td>
<td>11:00 – 13:00 E5.333</td>
<td>Seminar guidelines &amp; Introduction to scientific work</td>
</tr>
</tbody>
</table>

### Deadlines till 23:59 CET

<table>
<thead>
<tr>
<th>Date</th>
<th>Task</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday, 20 November 2017</td>
<td>Outline &amp; Literature references</td>
<td>Students</td>
</tr>
<tr>
<td>Monday, 11 December 2017</td>
<td>Essay draft for review</td>
<td>Students</td>
</tr>
<tr>
<td>Tuesday, 12 December 2017</td>
<td>Assignment of peer reviews</td>
<td>Advisors</td>
</tr>
<tr>
<td>Monday, 18 December 2017</td>
<td>Completed peer review</td>
<td>Students</td>
</tr>
<tr>
<td>Monday, 15 January 2018</td>
<td>Presentation slides for feedback</td>
<td>Students</td>
</tr>
<tr>
<td>Monday, 22 January 2018</td>
<td>Advisor feedback</td>
<td>Advisors</td>
</tr>
<tr>
<td>Monday, 5 February 2018</td>
<td>Essay for feedback</td>
<td>Students</td>
</tr>
<tr>
<td>Friday, 16 February 2018</td>
<td>Advisor feedback</td>
<td>Advisors</td>
</tr>
<tr>
<td>Wednesday, 28 February 2018</td>
<td>Final Essay Submission</td>
<td>Students</td>
</tr>
</tbody>
</table>

### Block Seminar

<table>
<thead>
<tr>
<th>Date</th>
<th>Time &amp; Place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thursday, 25 January 2018</td>
<td>9:00 - 12:00 TBA</td>
</tr>
<tr>
<td>Friday, 26 January 2018</td>
<td>9:00 - 18:00 E5.333</td>
</tr>
</tbody>
</table>
Allocation of Topics

- Doodle poll
  - You will receive an invitation via e-mail today
  - Choose exactly 3 preferences until 18:00, Thursday, 12th October
- Collisions will be solved by applying the following criterion:
  - Students in higher semesters (at least 2nd semester) are preferred.
  - Students with main field “Software Technology and Information Systems” or “Software Engineering” are preferred.
  - With equivalent qualifications: randomness
- You will be informed via e-mail which topic you are assigned
  - Please confirm this mail until 18:00, Sunday, 15th October
  - You can also reject the topic, but inform us as early as possible!
- If you confirm a topic, we expect you **not to drop** the seminar in the middle of the semester
Relaxing Security Policies through Declassification
Advisor: Christopher Gerking (christopher.gerking@hni.upb.de)

Task for student:
- Survey different approaches for declassification
- Demonstrate advantages and shortcomings
- Assess suitability of the approaches in the context of model-driven security

Literature:
Task for student:
- Compare different security models with respect to compositionality
- Illustrate differences on a “common ground“
- Point out assumptions and limitations for real-world applications

Literature:
Introduction:

- In practice, requirements specifications are still written in natural language.
- This also includes different kinds of security requirements. Examples:
  
  "The system shall enforce access privileges that enable authorized users to edit discharge instructions for a particular patient." (confidentiality)

  "The system shall log every time discharge instructions for a particular patient are edited." (accountability)

- Natural language processing techniques can improve automated processing of such requirements.

Task for student:

- Inspect and critically reflect the given literature wrt. security requirements in natural language artifacts text.
- Try to find related work and compare it to the given literature.

Literature:

Problem

- Implementations of software evolve and are affected by constant change
- ...but the quality of the implementation, including the derived security, has to be ensured
- Minor modifications to the code easily break the complete security model

Approach

- The developer specifies constraints in the code
- From the constraints a static model of the program’s behavior is build
- The model is solved and potentially invalidated constraints are reported to the developer

Task for student:

- Describe the proposed approach and highlight its advantages and disadvantages

Literature

- [Christakis, M.; Müller, P.; Wüstholz, V.: “An Experimental Evaluation of Deliberate Unsoundness in a Static Program Analyzer”. Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015]
Scalable and Precise String Analysis
Advisor: Linghui Luo (linghui.luo@upb.de)

Introduction:

- Client side software may process string-based user input
  - Vulnerabilities: SQL injection, Cross-Site Scripting
  - Some program paths only execute under certain string inputs
  - Imprecise string analysis could miss possible vulnerabilities

Task for student:

- Understand the importance of string analysis and review existing frameworks for it
- Select and compare frameworks which are precise and scalable for large programs and can be used for java and android applications

Literature:


Inferring Specifications for Taint-Style Vulnerabilities
Advisor: Goran Piskachev (goran.piskachev@iem.fraunhofer.de)

- **Introduction:**
  - Many vulnerabilities (e.g. SQL injection) can be detected via taint analysis
  - To configure these analyses, the users need to specify different configuration methods (e.g. source, sink)
  - This is a manual task that requires security-related domain knowledge
  - The users often face many false positives
    - One of the main reasons is incomplete or wrong specifications
  - Few approaches (e.g. Merlin) propose an automatic creation of these specifications

- **Task for student:**
  - Give an overview of the Merlin approach
  - Compare the Merlin approach to other similar approaches (e.g. SuSi)

- **Literature:**
Model-driven Security for Embedded Systems
Advisor: Johannes Geismann (johannes.geismann@upb.de)

- **Introduction:**
  - When designing *safe and secure* embedded systems not only software but also hardware has to be considered
  - Model-driven approaches are used to assist designers and developers in early development steps
  - **SysML-Sec** is a method for this task

- **Task for student:**
  - Give a comprehensive overview
  - Which threats / attacks are considered?
  - Which viewpoints are covered?
  - What are the limitations of this approach?
  - Compare to related approaches

- **Literature:**
  - Ludovic Aprville, Yves Roudier, "*SysML-Sec: A Model-Driven Environment for Developing Secure Embedded Systems*", Proceedings of the 8th conference on the security of network architecture and information systems (SARSSI'2013), Mont de Marsan, France, 16-18 sept. 2013
  - Ludovic Aprville, Yves Roudier, "*SysML-Sec: A Model Driven Approach for Designing Safe and Secure Systems*", Special session on Security and Privacy in Model Based Engineering, 3rd International Conference on Model-Driven Engineering and Software Development (Modelsward), Angers, France, Feb. 2015
Compositional Verification of Security Properties
Advisor: Johannes Geismann (johannes.geismann@upb.de)

- **Introduction:**
  - On architectural level, formal methods can be used to prove security requirements, e.g., model checking
  - For large systems, model checking is infeasible since the state space of the system is too large
  - Compositional verification can help to tackle this problem

- **Task for student:**
  - Explain the approach presented in [1]
  - Explain how compositional model checking can help to analyze security properties of large-scale systems
  - Survey other compositional verification approaches for analyzing security properties and relate them to [1]

- **Literature:**
A Comparison of Hooking Approaches for ART
Advisor: Andreas Peter Dann (andreas.peter.dann@upb.de)

Introduction:

What happens with the personal information?
Can we enforce that our address book is not leaked to any suspicious network?

→ To give an answer, we need to extract runtime information from an app.
   Thus, we need to hook into its methods and check which data it reads or
   which server it connects to.
   For hooking exist several approaches, with different drawbacks and advantages.

Task for student:

- Compare the approaches
- Point out for each approach: Advantages, Drawbacks, What is covered, What is missing?
- Draw your own conclusion based on your results and literature

Literature:

  stock Android
Object-Capabilities as Means of Permission and Authority in Software Systems

Advisor: Dr. Ben Hermann (ben.hermann@upb.de)

Introduction:

- What do we mean when a system is secure?
- How do we characterize its security?
- Recent research in the field of Object-Capability security helps to find answers.

Task for student:

- Go beyond the given literature – systematically find CLOSELY related work
- Give a comprehensive overview and comparison of the different approaches
- What are the open challenges in the field

Literature:


Type Systems as Means to Secure Software Systems
Advisor: Dr. Ben Hermann (ben.hermann@upb.de)

Introduction:
- One class of security problems are confidentiality issues.
- They occur whenever information is leaked to parties that are not permitted to receive the information.
- But, how can we prove that a system does not have such issues?
- Type systems may be the answer to that.

Task for student:
- Go beyond the given literature – systematically find CLOSELY related work
- Give a comprehensive overview and comparison of the different approaches
- What are the open challenges in the field

Literature:
Probabilistic Program Modeling for High-precision Anomaly Classification

Advisor: David Schubert (david.schubert@iem.fraunhofer.de)

- Introduction:
  - Introduce fundamentals: Intrusion Detection Systems, Hidden Markov Models
  - Construct a running example for your thesis
  - Recap the approach by Xu et al. using the running example

- Task for student:
  - Introduce fundamentals: Intrusion Detection Systems, Hidden Markov Models
  - Construct a running example for your thesis
  - Recap the approach by Xu et al. using the running example

- Literature:
Executable Misuse Cases for Modeling Security Concerns
Advisor: Thorsten Koch (thorsten.koch@iem.fraunhofer.de)

- **Problem:**
  - Security is a crucial issues for information systems. However, in Software Engineering security is mainly considered as non-function requirements after the definition of the systems. This approach often leads to problems, which translate to security vulnerabilities.

- **Approach:**
  - The approach of Whittle et al. is supposed to model and analyze security requirements alongside functional requirements by means of misuse cases. It provides a formalization of UML interaction overview diagrams to model scenarios precisely. This formalization is transformed into a set of communicating FSMs to perform the analysis.

- **Task for student:**
  - Describe the approach
  - Especially focus on security attacks that can be detected by the approach

- **Literature:**
Analyzing IEC 61131 Languages using Soot
Advisor: Sven Merschjohann (sven.merschjohann@iem.fraunhofer.de)

- **Introduction:**
  - PLCs are written in IEC 61131 languages
  - Despite being widely used sophisticated static analysis is rare
  - Writing static analysis is complex and time-consuming

- **Approaches:**
  - Reuse existing analysis framework Soot (Java) by transforming Structured Text to Jimple

- **Task for student:**
  - Describe the approach and highlight its advantages and limitations using an own example
  - Evaluate whether this approach can be applied to the other IEC languages as well

- **Literature:**
The State of the Art of Disassembly of x86/x64 code
Advisor: Martin Mory (martin.mory@upb.de)

- **Introduction:**
  - Compilation from high-level programming language to machine code brings information loss
  - Disassembly of machine code is required for understanding, but challenging

- **Task for student:** (can be done by 2 students)
  - Point out challenges for disassembly of x86/x64 code
  - Summarise, compare and assess recent approaches towards disassembly of x86/x64 code

- **Literature:**
  - Dennis Andriesse, Xi Chen, Victor van der Veen, Asia Slowinska, Herbert Bos: *An In-Depth Analysis of Disassembly on Full-Scale x86/x64 Binaries*
  - … more papers to be identified
How to Track Configuration Options
Advisor: Linghui Luo (linghui.luo@upb.de)

- **Introduction:**
  - Many modern software systems are highly configurable
    - Increases flexibility
    - Hard to be understood, maintained and analyzed
  - It is necessary to determine how configuration options affect program behavior

- **Task for student:** (can be done by 2 students)
  - Review and compare existing approaches to track configuration options

- **Literature:**
  - Z. Dong, A. Andrzejak and K. Shao, "Practical and accurate pinpointing of configuration errors using static analysis," 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, 2015, pp. 171-180. ([http://dx.doi.org/10.1109/ICSM.2015.7332463](http://dx.doi.org/10.1109/ICSM.2015.7332463))
A Survey on Common Android Malware Schemes
Advisor: Manuel Benz (manuel.benz@upb.de)

- Introduction:
  - With more than 2 billion monthly active Android devices and lots of sensitive data stored on them, the platform constitutes a perfect target for malware developers. Common schemes of Android malware typically include silently sending premium SMS to raise a fee on the user or encrypting the user's data, to blackmail her for money afterward. Many more exist…

- Task for student: (can be done by 2 students)
  - Dig into literature starting from the given publications.
  - Collect a comprehensive list of common Android malware schemes.
  - The given literature is “old” (2011/12). Try to find current references and elaborate the evolution of malware schemes since then.

- Literature:
  - https://doi.org/10.1007/978-3-319-24018-3_12
  - https://doi.org/10.1109/SP.2012.16
  - https://doi.org/10.1109/BADGERS.2014.7
  - https://doi.org/10.1145/2046614.2046618
Language Theoretic Security for Input Handling
Advisor: Martin Mory (martin.mory@upb.de)

■ Introduction:
  ■ Parsing expected tokens/inputs for an application may be an easy task. But how to react to the invalid/malicious ones?
  ■ Hand-crafted parsers for complex input languages are difficult to implement and prone to vulnerabilities.
  ■ Language Theoretic Security (LangSec) is an upcoming field that intends to handle input using grammars and thereby prevent input-specific threats.
  ■ Hammer is a toolkit for parser construction following the LangSec philosophy.

■ Task for student:
  ■ Examine, where and how flaws in traditional input handling caused severe vulnerabilities and provide prominent examples of recent years.
  ■ What are the LangSec concepts behind the Hammer toolkit? Show why and how these help preventing vulnerabilities. Give a non-trivial example where the utilization of hammer provides a notable security-benefit.
  ■ How does Hammer compare to the state of the art of LangSec parser generators?

■ Literature:
  ■ The Seven Turrets of Babel: A Taxonomy of LangSec Errors and How to Expunge Them
    Falcon Momot, Sergey Bratus, Sven M. Hallberg, Meredith L. Patterson
  ■ Building Hardened Internet-of-Things Clients with Language-theoretic Security
    Prashant Anantharaman, Michael Locasto, Gabriela F. Ciocarlie, Ulf Lindqvist
  ■ https://github.com/UpstandingHackers/hammer
Thank you for your attention