UNSW Course Outline

COMP6131 Software Security Analysis - 2024

Published on the 29 May 2024

General Course Information

Course Code : COMP6131
Year : 2024
Term : Term 2
Teaching Period : T2
Is a multi-term course? : No
Faculty : Faculty of Engineering
Academic Unit : School of Computer Science and Engineering
Delivery Mode : In Person
Delivery Format : Standard
Delivery Location : Kensington
Campus : Sydney
Study Level : Postgraduate, Undergraduate
Units of Credit : 6

Useful Links
Handbook Class Timetable

Course Details & Outcomes

Course Description

This course is designed to provide a systematic exploration of automated source code analysis and verification techniques, with the aim of gaining hands-on experience in implementing code analysis tools to identify common yet important software vulnerabilities in software systems. By taking this course, students can put static analysis and verification theories and advanced techniques into practice. They will be able to build source code analysis tools (e.g., written in C++) based on modern compilers and popular open-source frameworks to scan, comprehend and detect programming mistakes and vulnerabilities with the purpose of enhancing code quality and security.

Course Aims

The primary goal of this course is to familiarize students with a variety of source code analysis techniques and algorithms, ranging from fundamental to state-of-the-art. Students will be encouraged to develop their own tools based on an open-source framework that uses a compiler, and they will learn how these techniques and tools can be applied to detect real-world code vulnerabilities. Upon completing the course, students will have a comprehensive understanding of the history of source code analysis, the current techniques used, and the future challenges that must be addressed in this field.

Course Learning Outcomes

Course Learning Outcomes
CLO1 : Explain source code vulnerabilities in system software and motivations for software analysis and verification
CLO2 : Explain basic compiler intermediate representation and its importance for precise software vulnerability detection
CLO3 : Implement source code analysis techniques including control- and data-flow analysis
CLO4 : Develop source code verification techniques including constraint solving and assertion-based verification using automated theorem provers.
CLO5 : Design and implement well-considered, high performance, static analysis algorithm to solve problems
CLO6 : Create effective and minimum unit tests and documentation to validate the correctness of the code analysis tools

Course Learning Outcomes	Assessment Item
CLO1 : Explain source code vulnerabilities in system software and motivations for software analysis and verification	• Fortnightly coding exercises and quizzes
CLO2 : Explain basic compiler intermediate representation and its importance for precise software vulnerability detection	• Assertion-based code verification • Static taint tracking • Fortnightly coding exercises and quizzes
CLO3 : Implement source code analysis techniques including control- and data-flow analysis	• Static taint tracking
CLO4 : Develop source code verification techniques including constraint solving and assertion-based verification using automated theorem provers.	• Abstract interpretation • Assertion-based code verification
CLO5 : Design and implement well-considered, high performance, static analysis algorithm to solve problems	• Abstract interpretation • Fortnightly coding exercises and quizzes
CLO6 : Create effective and minimum unit tests and documentation to validate the correctness of the code analysis tools	• Abstract interpretation • Fortnightly coding exercises and quizzes

Learning and Teaching Technologies

Moodle - Learning Management System

Additional Course Information

Online Resources:

- Course webpage on WebCMS
- Software Security Analysis - Labs and Assignments Specs and Coding Templates
- SVF: Static Value-Flow Analysis Framework for Source Code and its doxygen
- Compilers: Principles, Techniques, and Tools Hardcover
- LLVM Compiler
- Anders Møller and Michael I. Schwartzbach, Static Program Analysis

Assessments

Assessment Structure

Assessment Item	Weight	Relevant Dates
Fortnightly coding exercises and quizzes Assessment Format: Individual	30%	Start Date: Not Applicable Due Date: Weeks 3, 7, 10
Assertion-based code verification Assessment Format: Individual	25%	Start Date: Not Applicable Due Date: Week 8
Abstract interpretation Assessment Format: Individual	25%	Start Date: Not Applicable Due Date: Week 11
Static taint tracking Assessment Format: Individual	20%	Start Date: Not Applicable Due Date: Week 4

Assessment Details

Fortnightly coding exercises and quizzes

Assessment Overview

During labs, students will be given quizzes and/or small-scale coding exercises to complete as part of their assessment. These tasks will cover topics such as understanding compiler intermediate representation, graph representation of code, implementing basic graph traversal algorithms, using Z3 theorem prover for assertion verification, and abstract execution.

These tasks will provide students with the opportunity to revisit and solidify individual concepts from the lectures. The quizzes and coding exercises will be marked based on the correctness of the student's implementation, and feedback will be provided to help students improve their coding skills, and understanding of the key knowledge in lectures.

Course Learning Outcomes

- CLO1 : Explain source code vulnerabilities in system software and motivations for software analysis and verification

- CLO2 : Explain basic compiler intermediate representation and its importance for precise software vulnerability detection
- CLO5 : Design and implement well-considered, high performance, static analysis algorithm to solve problems
- CLO6 : Create effective and minimum unit tests and documentation to validate the correctness of the code analysis tools

Assertion-based code verification

Assessment Overview

The objective of this assignment is to utilize symbolic execution and automated theorem prover to develop assertion-based verification, which automates the verification of code properties through the use of assertions. The assertion verification builds upon the fortnightly lab exercises as well as the control- and data-flow analysis implemented in Assignments 1.

In this assignment, students are tasked with practicing the utilization of the satisfiability solver and applying it to automated assertion verification. By submitting accurate implementations that yield the desired outputs, students will successfully complete the final component of the automated code analyzer for the course. This component enables the analysis and verification of real-world programs.

Course Learning Outcomes

- CLO2 : Explain basic compiler intermediate representation and its importance for precise software vulnerability detection
- CLO4 : Develop source code verification techniques including constraint solving and assertion-based verification using automated theorem provers.

Abstract interpretation

Assessment Overview

The objective of this assignment is to practice and implement abstract interpretation. Students will have opportunities to build their abstract execution solvers and understand sound approximations of code semantics based on monotonic functions over lattices. They will work on fortnightly lab exercises as well as previous assignments to implement abstract interpretation/execution, which can be used to detect code defects.

The objective is for students to grasp these fundamental concepts and apply them effectively. Successful completion of this assignment requires students to not only understand these concepts but also implement them accurately, ensuring that their programs compile without errors and generate the desired outputs.

Course Learning Outcomes

- CLO4 : Develop source code verification techniques including constraint solving and assertion-based verification using automated theorem provers.
- CLO5 : Design and implement well-considered, high performance, static analysis algorithm to solve problems
- CLO6 : Create effective and minimum unit tests and documentation to validate the correctness of the code analysis tools

Static taint tracking

Assessment Overview

The objective of this assignment is to build a a tainted information flow tracker that integrates both the control-flow reachability analysis and data-dependence information. This enables the tracking of tainted input and the identification of tainted information flows on the code graph. The implementation of this assignment is a component of a course-generated automated source code analysis tool.

The assignment requires students to enhance their coding and debugging skills by utilizing online slides and resources. These resources will also provide opportunities to learn how to write efficient and high-quality code checkers. As part of practicing the fundamental programming elements, the student is required to submit their implementation, which should compile correctly and produce the desired outputs. This assignment's implementation enhances the student's comprehension of source code analysis and code representation, serving as preparation for the development of a code analysis tool in future assessments.

Course Learning Outcomes

- CLO2 : Explain basic compiler intermediate representation and its importance for precise software vulnerability detection
- CLO3 : Implement source code analysis techniques including control- and data-flow analysis

General Assessment Information

Grading Basis

Standard

Course Schedule

Attendance Requirements

Students are strongly encouraged to attend all classes and review lecture recordings.

General Schedule Information

Please find the course schedule here:

https://webcms3.cse.unsw.edu.au/COMP6131/24T2/outline

Staff Details

Position	Name	Email	Location	Phone	Availability	Equitable Learning Services Contact	Primary Contact
Convenor	Yulei Sui		501H K17			Yes	Yes

Other Useful Information

Academic Information

I. Special consideration and supplementary assessment

If you have experienced an illness or misadventure beyond your control that will interfere with your assessment performance, you are eligible to apply for Special Consideration prior to, or within 3 working days of, submitting an assessment or sitting an exam.

Please note that UNSW has a Fit to Sit rule, which means that if you sit an exam, you are declaring yourself fit enough to do so and cannot later apply for Special Consideration.

For details of applying for Special Consideration and conditions for the award of supplementary assessment, please see the information on UNSW's Special Consideration page.

II. Administrative matters and links

All students are expected to read and be familiar with UNSW guidelines and polices. In particular, students should be familiar with the following:

- Attendance
- UNSW Email Address
- Special Consideration
- Exams
- Approved Calculators
- Academic Honesty and Plagiarism
- Equitable Learning Services

III. Equity and diversity

Those students who have a disability that requires some adjustment in their teaching or learning environment are encouraged to discuss their study needs with the course convener prior to, or at the commencement of, their course, or with the Equity Officer (Disability) in the Equitable Learning Services. Issues to be discussed may include access to materials, signers or note-takers, the provision of services and additional exam and assessment arrangements. Early notification is essential to enable any necessary adjustments to be made.

IV. Professional Outcomes and Program Design

Students are able to review the relevant professional outcomes and program designs for their streams by going to the following link: https://www.unsw.edu.au/engineering/student-life/student-resources/program-design.

Note: This course outline sets out the description of classes at the date the Course Outline is published. The nature of classes may change during the Term after the Course Outline is published. Moodle or your primary learning management system (LMS) should be consulted for the up-to-date class descriptions. If there is any inconsistency in the description of activities between the University timetable and the Course Outline/Moodle/LMS, the description in the Course Outline/Moodle/LMS applies.

Academic Honesty and Plagarism

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW. *Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.*

Plagiarism is a type of intellectual theft. It can take many forms, from deliberate cheating to accidentally copying from a source without acknowledgement. UNSW has produced a website with a wealth of resources to support students to understand and avoid plagiarism, visit: student.unsw.edu.au/plagiarism. The Learning Centre assists students with understanding academic integrity and how not to plagiarise. They also hold workshops and can help students one-on-one.

You are also reminded that careful time management is an important part of study and one of the identified causes of plagiarism is poor time management. Students should allow sufficient time for research, drafting and the proper referencing of sources in preparing all assessment tasks.

Repeated plagiarism (even in first year), plagiarism after first year, or serious instances, may also be investigated under the Student Misconduct Procedures. The penalties under the procedures can include a reduction in marks, failing a course or for the most serious matters (like plagiarism in an honours thesis or contract cheating) even suspension from the university. The Student Misconduct Procedures are available here:

www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf

Submission of Assessment Tasks

Work submitted late without an approved extension by the course coordinator or delegated authority is subject to a late penalty of five percent (5%) of the maximum mark possible for that assessment item, per calendar day.

The late penalty is applied per calendar day (including weekends and public holidays) that the assessment is overdue. There is no pro-rata of the late penalty for submissions made part way through a day. This is for all assessments where a penalty applies.

Work submitted after five days (120 hours) will not be accepted and a mark of zero will be awarded for that assessment item.

For some assessment items, a late penalty may not be appropriate. These will be clearly indicated in the course outline, and such assessments will receive a mark of zero if not completed by the specified date. Examples include:

- Weekly online tests or laboratory work worth a small proportion of the subject mark;
- Exams, peer feedback and team evaluation surveys;
- Online quizzes where answers are released to students on completion;
- Professional assessment tasks, where the intention is to create an authentic assessment that has an absolute submission date; and,
- Pass/Fail assessment tasks.

Faculty-specific Information

Engineering Student Support Services – The Nucleus - enrolment, progression checks, clash requests, course issues or program-related queries

Engineering Industrial Training – Industrial training questions

UNSW Study Abroad – study abroad student enquiries (for inbound students)

UNSW Exchange – student exchange enquiries (for inbound students)

UNSW Future Students – potential student enquiries e.g. admissions, fees, programs, credit transfer

Phone

(+61 2) 9385 8500 – Nucleus Student Hub

(+61 2) 9385 7661 – Engineering Industrial Training

(+61 2) 9385 3179 – UNSW Study Abroad and UNSW Exchange (for inbound students)

School Contact Information

CSE Help! - on the Ground Floor of K17

- For assistance with coursework assessments.

The Nucleus Student Hub - https://nucleus.unsw.edu.au/en/contact-us

- Course enrolment queries.

Grievance Officer - grievance-officer@cse.unsw.edu.au

- If the course convenor gives an inadequate response to a query or when the courses convenor does not respond to a query about assessment.

Student Reps - stureps@cse.unsw.edu.au

- If some aspect of a course needs urgent improvement. (e.g. Nobody responding to forum queries, cannot understand the lecturer)

You should **never** contact any of the following people directly:

- Vice Chancellor

- Pro-vice Chancellor Education (PVCE)

- Head of School

- CSE administrative staff

- CSE teaching support staff

They will simply bounce the email to one of the above, thereby creating an unnecessary level of indirection and a delay in the response.