Syllabus for SE 421 – Software Analysis & Verification for Safety and Security
Spring 2022

Lectures: Room: 1312 Hoover Time: M W F 3:20-4:10 PM

You should attend the lectures regularly. Participate in discussions and Q&A. Lectures will include Q&A for the assignments, live demonstrations of program analysis, and announcements. You are encouraged to ask questions in class. Lecture slides will be posted and the recording of the lecture will be posted on CANVAS after each lecture.

Office Hours: We will hold office hours through WebEx. If necessary, we will arrange an in-person meeting.

Announcements: Pay attention to announcements made during the lecture and/or posted on CANVAS.

Piazza: Questions requiring in-depth explanation will be answered in class by the instructor. TA will answer the routine homework questions on Piazza. *Use Piazza strictly for posting questions about assignments.* ***Do not post*** *on Piazza questions/comments about submission deadline, extension of deadline, grading policy, or any other logistics/policy topics. Use Q&A time during lectures for such questions or comments.*

Please find sign up for the course Piazza with the following link:
https://www.piazza.com/iastate/spring2022/s2022se4211

Email Help: Use email for getting personal help or communicating any questions or comments you may have about the course/grading policy. While sending email to the instructor or TA include in the subject line "SE 421 Student."

Instructor: Suresh Kothari, kothari@iastate.edu, https://www.linkedin.com/in/surajkothari
Office Hours: Monday (2:30 – 3:00), Wednesday (4:30 – 5:00)
WebEx link: https://iastate.webex.com/meet/kothari

TA: Sharwan - sharwan@iastate.edu
Office Hours: Tuesday (4:00 – 5:00), Thursday (4:00 – 5:00)
An occasional office hour would be scheduled to accommodate student requests
WebEx link: https://iastate.webex.com/meet/sharwan

Catalog Description

Significance of software safety and security, Various facets of security in cyber-physical and computer systems, Threat modeling for software safety and security, Categorization of software vulnerabilities, Software analysis and verification: mathematical foundations, data structures and algorithms, program comprehension, analysis, and verification tools, Automated vs. human-on-the-loop approach to analysis and verification, Practical considerations of efficiency, accuracy, robustness, and scalability of analysis and verification, Case studies with application and systems software, Evolving landscape of software security threats and mitigation techniques.
Pre-requisites: COM S 309; CPR E 310 OR COM S 230

Laptop Requirement

All students are required to own or obtain a laptop computer. You can find computer recommendations at:
https://it.engineering.iastate.edu/new-to-engineering/#ComputerRec

Course Content

This course is about analyzing and verifying software with applications to safety and security. It is *not* about network security or computer system administration. There is a different course that covers network security and computer system administration.

Topics

1. Context for Safety and Security of Software Systems
2. Safety and Security Vulnerabilities in Software
3. Mathematical Foundation for Threat Modeling, and for Analysis and Verification of Software
4. Fundamental Challenges of Analyzing and Verifying Software
5. Software Analysis and Verification Algorithms and Tools
6. Efficiency, Accuracy, Robustness, and Scalability of Software Analysis and Verification
7. Automated vs. Human-on-the-loop Software Analysis and Verification
8. Experimental/Empirical/Application Studies of Analysis and Verification

Course outcomes/objectives

The course is designed so that the students can learn to do the following:

1. Apply the knowledge of mathematics, program semantics, and software engineering to model software safety and security vulnerabilities and design algorithms and tools to analyze and verify/fix the vulnerabilities.
2. Design and conduct experiments and interpret the results to understand the fundamental challenges and performance issues of software analysis and verification.
3. Formulate software verification problems and implement practical solutions to achieve efficiency, accuracy, robustness, and scalability goals.
4. Recognize the broad impact of software security engineering solutions in a global, economic, environmental, and societal context.
5. Apply modern program comprehension, software analysis and verification techniques and tools.
6. Recognize the need for continuous learning to keep abreast of the evolving landscape of software security threats and approaches to address those threats.
7. Describe succinctly and clearly the efforts, key aspects, and important findings from software analysis and verification studies.

Correspondence with a to k ABET Outcomes/Objectives: 1-a, 2-b, 3-e, 4-h, 5-k, 6-i, and 7-g

Grading Scheme

The grading policy and the assessment plan are described below. If you have any doubts, please contact the instructor right away.

Homework and Project (65% of the grade): There will be 8 to 10 homework assignments, and one project. The 65% of the grade will be distributed across assignments and project(s). For example, if a particular homework has 20 points and a project has 50 points and the total of homework and project scores is 250 points, then the particular homework will count for (20/250)*65% and the project will count for (50/250)*65%.

HOMEWORK GRADE WILL HAVE SIGNIFICANT IMPACT ON YOUR OVERALL GRADE. YOU SOULD DO ALL ASSIGNMENTS AND SUBMIT THEM ON TIME. MIDTERM GRADES WILL BE BASED ON THE HOMEWORK.

Class Exam (30% of the grade) The Exam will be towards the end, before the dead week. It will be comprehensive. We will announce additional details before the exam.

Optional Comprehensive Final Exam (30% of the grade): The final is optional. If you are satisfied by the end-of-semester grade, you do not need to take final. It will be comprehensive. If you do take the final, *the average*

of the class exam and the final will be used for calculating the course grade. For example, with midterm score 30/50 and the final score 40/50, the average 35/50 will be used, which amounts to 21% out of 30% for the exam grade.

Other (5% of the grade) It will be at instructor's discretion. It would depend on several factors including exemplary work, good participation in discussions, on-time submissions, timely and high-quality questions.

Grading Policy (If your score is S): A: S >= 90, B: 80 <= S < 90, C: 70<= S < 80, D: 60<= S < 70, D-: 55 <= S < 60, F: S < 55.

Checking Your Grades: Using the given weights and the grading scale you can compute your letter grade. As per the ISU policy, we will give the midterm grades for students who have C- or below. The tentative overall percentage will be posted before the final exam. We will post the letter grades after the final exam. DO NOT USE *grades computed by Canvas; they are not accurate*.

Homework and Project Submission Policy

Submit electronically before the class period on the due date. *Work individually unless announced otherwise*. There will be 5% late penalty if submitted before the class period after the due date, and 15% thereafter. Late assignments will not be accepted beyond the 2nd class period after the due date.

The submission must be clearly written, typed (11-point font), spell-checked, properly formatted and organized. It must follow the requirements stated in each assignment. Pay attention to announcements.
For assignments and the exam, we will discuss answers for common mistakes during lectures. We will not post solutions.

Suggestions to Study for the Course

1. Attend the lectures regularly. Ask questions immediately if anything is not clear during the lecture.
2. The course is mostly about conceptual understanding and critical thinking. Trying to memorize the material will not work. It will be a much easier course if you learn the concepts. Not attending or not paying attention during the lectures, will make this course much more difficult.
3. Read the lecture note after the class. If you have any doubt about the lecture, ask questions during the next class.
4. Read the problems the day the homework is assigned. If you have any doubt about the homework, post or ask questions during the class to get clarity. Start working on the homework as soon as possible.

Academic Dishonesty Policy and Guidelines

The policy http://www.dso.iastate.edu/ja/academic/misconduct will be enforced. Here are some guidelines that you should follow.

1. Do your own work. Don't copy solutions/programs obtained from any person or a website. You may be liable for academic misconduct even if it is not an exact copy. When in doubt, ask for instructor's permission to use any other sources beyond the course material and references given to you.
2. Don't collaborate with another student beyond the extent specifically approved by the instructor.
3. Don't fabricate results.
4. Don't allow another student to copy your answers on assignments or exams.
5. Don't take an exam or complete an assignment for another student.
6. Do not collaborate with others on exams.
7. Do not use of any other material except the material that you will be told you can use.

Assessment Plan

Homework problems will be designed to assess knowledge about concepts and algorithms. Some homework assignments will involve hands-on practice to assess skills in formulating and implementing practical software analysis and verification solutions. The homework assignments will be used to assess course outcomes 1 through 6. **There will be weekly homework and it will usually be due on Wednesdays.**

Exam will expand on homework problems. It will require students to show that they can address variations of homework problems that require a deeper understanding. The exam will be used to assess course outcomes: 1, 4, 5, and 6.

Project will involve implementation, experiments, and a well-written technical report. The project will be used to assess course outcomes: 1, 3, 5, and 7.

Accessibility

Iowa State University is committed to assuring that all educational activities are free from discrimination and harassment based on disability status. Information or assistance is available online at www.sas.dso.iastate.edu, by contacting SAS staff by email at accessibility@iastate.edu, or by calling 515-294-7220.

Calendar

Spring 2022 Calendar – Lectures 1-44		
Monday	Wednesday	Friday
January 17 – Holliday	19 - Lecture 1	21 - Lecture 2
24 - Lecture 3	26- Lecture 4	28 - Lecture 5
31 - 6	February 2 – 7	4 - 8
7 - 9	9 – 10	11 - 11
14 - 12	16 – 13	18 - 14
21 – 15	23 – 16	25 – 17
28 - 18	March 2 – 19	4 – 20
7 - 21	9 - 22	11 - 23
March 14 – 18 Spring Break	16	18
21 - 24	23 – 25	25 – 26
28 - 27	30 – 28	April 1 - 29
4 - 30	6 – 31	8 - 32
11 - 33	13 – 34	15 – 35
18 - 36	20 – 37	22 – 38
25 - 39	27 – 40	29 - 41
May 2 - 42	4 - 43	6 - 44
May 9-12 Final Exams		
The optional final exam will be on *the date and time set by the university.* https://www.registrar.iastate.edu/students/exams/springexams		

Lecture Notes and Recorded Lecture

Lecture slides and the recorded lecture will be posted on CANVAS after the lecture. Read the lecture notes and listen to the lecture recording (in case you missed the lecture) before coming to the next lecture. Review the lecture notes while working on homework/project to look for hints. You are expected to attend the lecture during the scheduled time.

Use of Atlas for Analyzing Software

You will need to use a tool to analyze software. We will use Atlas for analyzing and visualizing software. If you find a better tool, let the instructor know. Without a tool, it will be quite laborious and error-prone to analyze software given for assignments. A separate handout will be given with instructions to download and setup Atlas.

Tool Demos and Repositories:

a. ICSE 2014 Atlas Platform Demo Video: https://www.youtube.com/watch?v=cZOWlJ-IO0k&feature=youtu.be
b. ICSE 2015 Android Security Toolbox Demo Video: http://youtu.be/WhcoAX3HiNU
c. Time Complexity Analyzer (TCA) Tool Demo: Video: https://youtu.be/SI1t4G2faDo
d. Linux Verification Tool Demo: https://youtu.be/uIfcuOGvcxw
e. Program Analysis for Cybersecurity: https://github.com/benjholla/PAC
f. Knowledge-Centric Software (KCS) Research Lab URL: http://www.ece.iastate.edu/kcsl/

References

1. F. P. Brooks Jr. The computer scientist as toolsmith II, Communications of the ACM, 39(3): 61{68, 1996, http://www.cs.unc.edu/~brooks/Toolsmith-CACM.pdf

2. F. P. Brooks Jr. No Silver Bullet — Essence and Accidents of Software Engineering, http://worrydream.com/refs/Brooks-NoSilverBullet.pdf

3. Brooks, Jr., F.P. 2003: "Three Great Challenges for Half-Century-Old Computer Science." Journal of the ACM**,** 50, 1: 25-26.

4. David Parnas, On the Criteria To Be Used in Decomposing Systems into Modules, https://www.cs.umd.edu/class/spring2003/cmsc838p/Design/criteria.pdf

5. David Parnas, Software Engineering Programs Are Not Computer Science Programs," *IEEE Software*, November/December 1999.

6. Steve McConnell, Software Engineering, Not Computer Science, http://www.stevemcconnell.com/psd/04-senotcs.htm

7. Kramer, J.: Is abstraction the key to computing? Communications of the ACM 50(4) (2007), pp. 36-42, http://www.ics.uci.edu/~andre/informatics223s2007/kramer.pdf

8. Donald Knuth, Fun with Binary Decision Diagrams (BDDs) – an online lecture, http://myvideos.stanford.edu/player/slplayer.aspx?coll=ea60314a-53b3-4be2-8552-dcf190ca0c0b&co=18bcd3a8-965a-4a63-a516-a1ad74af1119&o=true

9. Donald Knuth, Fun with Zero-suppressed Binary Decision Diagrams (ZDDs) – an online lecture, http://myvideos.stanford.edu/player/slplayer.aspx?coll=ea60314a-53b3-4be2-8552-dcf190ca0c0b&co=af52aca1-9c60-4a7b-a10b-0e543f4f3451&o=true

10. Mark Weiser, Program Slicing, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-10, NO. 4, JULY 1984, https://cseweb.ucsd.edu/classes/fa03/cse231/weiser.pdf

11. Agrawal, De Millo, and Spafford, "Efficient Debugging with Slicing and Backtracking," http://spaf.cerias.purdue.edu/Students/spyder/TR80P.pdf

12. Joseph Silva, "A Vocabulary of Program Slicing-Based Techniques," http://users.dsic.upv.es/~jsilva/papers/Vocabulary.pdf

13. Yunbo Deng, Suraj Kothari, Yogy Namara, "Program Slice Browser," http://www.eecs.yorku.ca/course_archive/2004-05/F/6431/ResearchPapers/Deng.pdf

14. Scott Grant, James R. Cordy, David Skillicorn, "Automated Concept Location Using Independent Component Analysis," http://research.cs.queensu.ca/home/cordy/Papers/GCS_WCRE08_ICA.pdf

15. James Cordy, “There is no problem in software maintenance that cannot be made worse by another level of indirection,” Keynote at International Conference on Software Maintenance and Evolution (ICSME) 2015, http://www.icsme.uni-bremen.de/Abstract_Keynote_Cordy.pdf

16. James Cordy, "Comprehending Reality: Practical Challenges to Software Maintenance Automation", Keynote Address, *IWPC 2003, IEEE 11th International Workshop on Program Comprehension,* Portland, Oregon, May 2003, http://research.cs.queensu.ca/~cordy/Papers/IWPC03_Keynote.pdf

17. Susan Horwitz and Thomas Reps “The Use of Program Dependence Graphs in Software Engineering,” http://research.cs.wisc.edu/wpis/papers/icse92.pdf

18. Andreas Zeller, Why programs fail, http://www.whyprogramsfail.com/

19. De Millo, Lipton, and Perlis, “Social Processes and Proofs of Theorems and Programs,” https://www.cs.columbia.edu/~angelos/Misc/p271-de_millo.pdf

20. A counterpoint to De Millo, Lipton, and Perlis, “Social Processes and Proofs of Theorems and Programs,” http://jxyzabc.blogspot.com/2009/10/nice-counterpoint-to-de-millo-lipton.html

21. Formal Verification: https://blog.inf.ed.ac.uk/sapm/2014/02/20/formal-verification-in-large-scaled-software-worth-to-ponder/

22. D. Beyer and A. K. Petrenko, “Linux driver verification,” in Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies. Springer, 2012, pp. 1–6, http://www.sosy-lab.org/~dbeyer/Publications/2012-ISOLA.Linux_Driver_Verification.pdf

23. J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, “Formal methods: Practice and experience,” ACM Computing Surveys (CSUR), vol. 41, no. 4, p. 19, 2009, http://vsr.sourceforge.net/fmsurvey.pdf

24. B. Gates, “Bill Gates Keynote: Microsoft Tech-Ed 2008,” 2008. http://news.microsoft.com/speeches/bill-gates-keynote-microsoft-tech•ed-2008-developers/

25. T. Deering, S. Kothari, J. Sauceda, and J. Mathews, “Atlas: a new way to explore software, build analysis tools,” in Companion Proceedings of the 36th International Conference on Software Engineering. ACM, 2014, pp. 588–591, https://www.researchgate.net/publication/266656750_Atlas_A_new_way_to_explore_software_build_analysis_tools

26. Benjamin Holland, Tom Deering, Suresh Kothari, Jon Mathews, Nikhil Ranade. Security Toolbox for Detecting Novel and Sophisticated Android Malware. In Proceedings of the 37th International Conference on Software Engineering *(ICSE 2015)*, Firenze, Italy, May 2015, https://benholland.com/papers/Security_Toolbox_for_Detecting_Novel_and_Sophisticated_Android_Malware.pdf

27. D. Beyer, “Competition on software verification,” in “Status report on software verification,” in Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2014, pp. 373–388.

28. DARPA Automated Program Analysis for Cybersecurity (APAC) Program: https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-11-63/listing.html

29. Common Vulnerability Enumeration (CVE), a dictionary of publicly known information security vulnerabilities and exposures, https://cve.mitre.org/

30. Common Weakness Enumeration (CWE), a community-developed dictionary of software weakness types, https://cwe.mitre.org/index.html

31. Tim Boland and Paul E. Black, National Institute of Standards, The Juliet 1.1 C/C++ and Java Test Suite, https://samate.nist.gov/docs/Juliet%201.1%20Oct%202012.pdf

32. DARPA Space/Time Analysis for Cybersecurity (STAC) Program: https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-14-60/listing.html