Assessing Common Software Vulnerabilities in Undergraduate Computer Science Assignments

Presented by: Andrew Sanders, Augusta University
Dr. Gursimran Singh Walia, Augusta University
Dr. Andrew Allen, Georgia Southern University

Introduction

- Secure coding education is still needed
  - U.S. Dept. of Homeland Security has cited 90% of reported security incidents result from exploits against defects in design or code of software.
  - Verizon’s 2023 Data Breach Investigation Report stated that software code vulnerability exploitation is one of the primary methods by which attackers access an organization.
  - ACM’s Curriculum Guidelines and recent ABET standards have evolved over time to include principles of secure computing.
  - ACM and IEEE Joint Task Force on Computing Curricula developed the Information Assurance and Security Knowledge Area.

Introduction (cont.)

- Student security education is lacking
  - Security course is either not required or offered as a senior-level class.
  - Veracode’s survey of developers and IT professionals found most felt their university-provided software security skills were inadequate for their industry job requirements
- A more targeted approach might be needed
  - The first step is to collect information on the types of vulnerabilities produced by students during code development.
  - For this work, we will be using the Common Weakness Enumerated (CWE) framework which categorizes vulnerabilities and refers to them using their CWE-ID

Introduction (cont.)

- The most common types of vulnerabilities studied by software vulnerability researchers are as follows:
  - (CWE-78) OS command injection
  - (CWE-79) Cross-site scripting
  - (CWE-89) SQL injection
  - (CWE-119) Buffer errors
  - (CWE-120) Buffer overflow
  - (CWE-190) Integer overflow
  - (CWE-306) Missing authentication for critical function
- Each of these is contained in the 2022 CWE Top 25 Most Dangerous Software Weaknesses

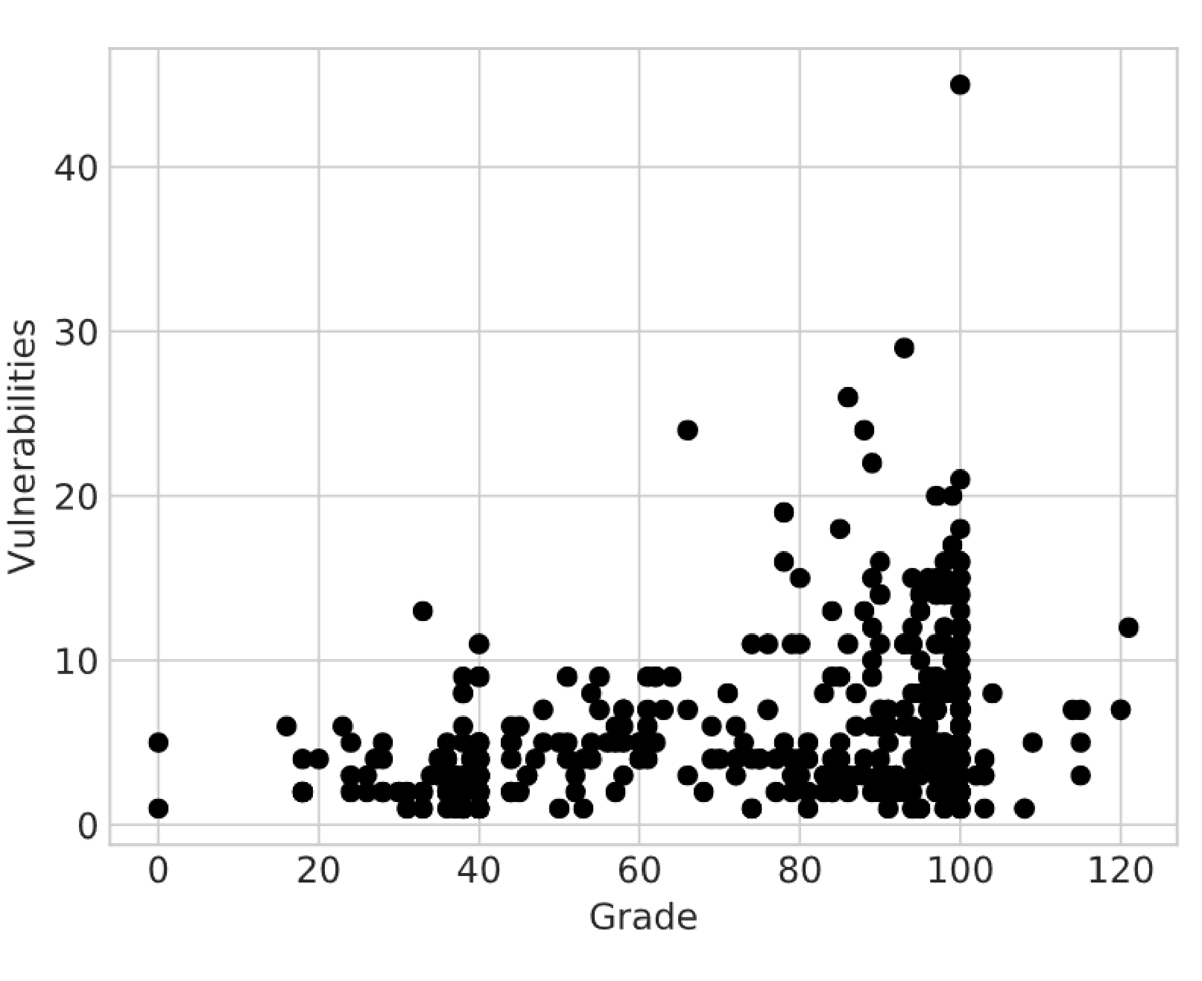
Introduced (cont.)

- The limited existing research (Yilmaz et al.) reports these CWE-IDs. None of these are represented in commonly researched vulnerabilities. Students should be prepared against commonly experienced vulnerabilities, so they are better prepared to make more secure software.

Type	Definition	#
CWE-259	Use of Hard-coded Password	829
CWE-20	Improper Input Validation	761
CWE-564	SQL Injection: Hibernate	751
CWE-943	Improper Neutralization of Special Elements in Data Query Logic	751
CWE-489	Active Debug Code	714
CWE-315	Cleartext Storage of Sensitive Information in a Cookie	23
CWE-117	Improper Output Neutralization for Logs	17
CWE-532	Insertion of Sensitive Information into Log File	17
CWE-778	Insufficient Logging	17
CWE-521	Weak Password Requirements	15
CWE-311	Missing Encryption of Sensitive Data	14
CWE-614	Sensitive Cookie in HTTPS Session Without 'Secure' Attribute	14

Related Work

- Limited research on the types of vulnerabilities produced by students.
- Yilmaz et al. used source code vulnerability analysis to study vulnerabilities introduced by students in a third-year Database Management Systems course. The table on the previous slide shows the most common types of vulnerabilities in their dataset.
- This figure plots the grades awarded to each student along with the number of vulnerabilities.



Related Work

- Hanif et al. studied software vulnerability detection methods and created a taxonomy of research interests.
- Related to this work is that the authors found that most existing works targeted specific types of vulnerabilities for detection.
  – These types are common because they are frequently targeted by vulnerability detection systems.
- They note there is a lack of a large gold-standard dataset for software vulnerability detection. The currently available real-world vulnerability dataset is the National Vulnerability Database (NVD) by the National Institute of Science and Technology (NIST).
- In general, data can be classified as coming from one of three places: NIST, open-source software, and private datasets.

Related Work

- Hu et al. studied vulnerabilities in Java programming textbooks for an undergraduate Java programming course.
- The authors used the open-source vulnerability tool called FindBugs to analyze the byte code of the sample source codes.
- They found many common bugs, raising security concerns. Students could potentially adopt the coding styles of these bugged code samples.
- The table shows the bugs as classified by the authors' vulnerability criteria.

Our Measurement	Book 1	Book 2	Book 3	Book 4
Security Vulnerability	17	13	17	26
Quality Vulnerability	82	36	121	106

Related Work

- Much work has been put into vulnerability detection and secure coding
- Lack of focus on the types of vulnerabilities produced by Computer Science students and graduates.
- This lack of analysis also pairs with a lack of directed pedagogy towards curbing the kinds of software vulnerabilities produced during the education process.

Methodology

- Using existing static analysis tools, we attempted to answer the following research questions:
  - RQ1: What are the most common software vulnerabilities produced by CS2 students in their assignment submission code?
  - RQ2: How do these software vulnerabilities compare and contrast with the types of commonly researched vulnerabilities?

Methodology: Dataset

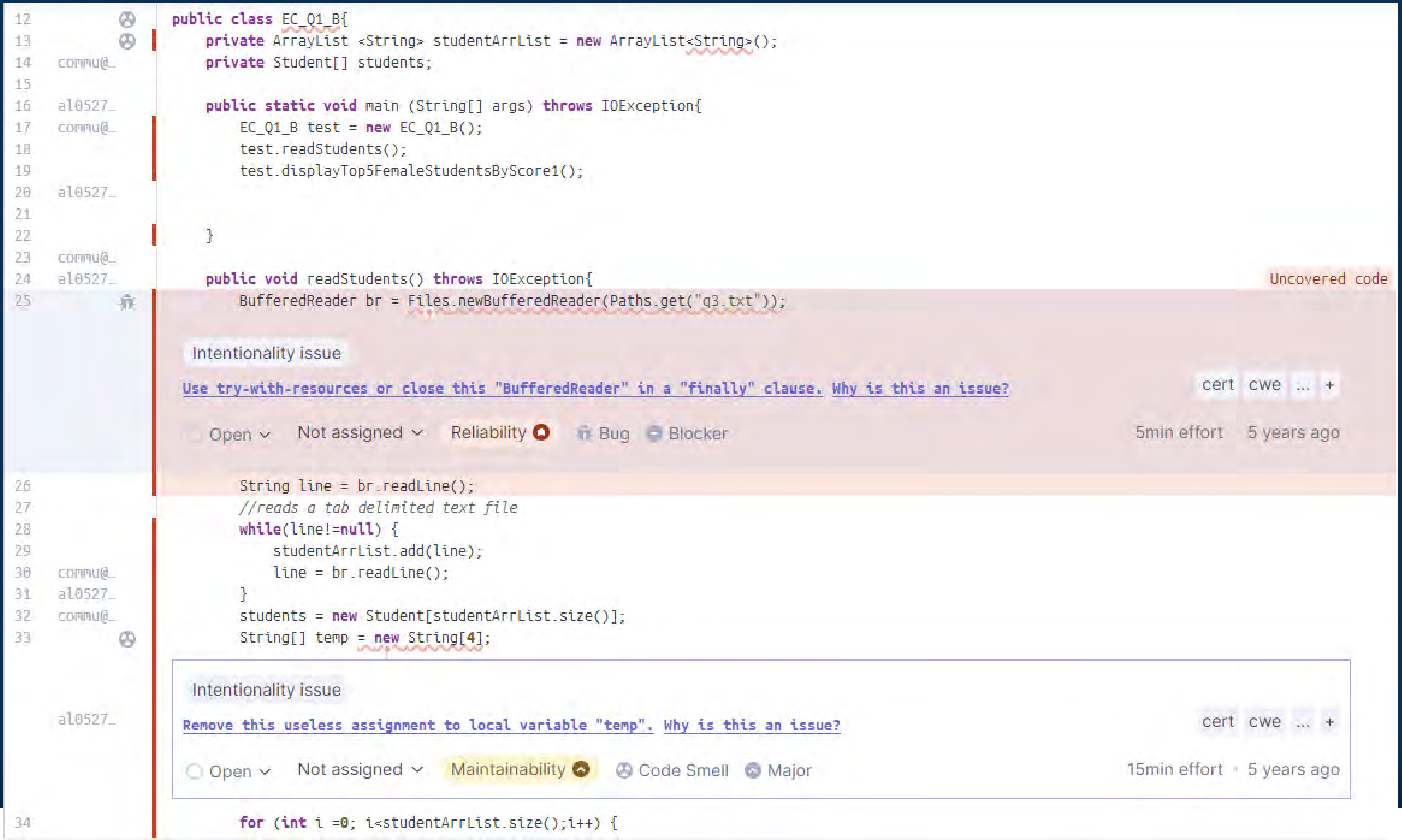
- We generated our dataset by analyzing the Github assignment submissions for a Georgia Southern University Programming Principles II course over the 2017-2023 school years.
- 3537 total assignment submissions (excluding empty projects).
- Assignments consisted of object-oriented assignment using the Java programming language.
- Each assignment was compiled before analysis, and each was grouped by year and semester.
- Our vulnerability tool reported all potential vulnerabilities grouped by CWE-ID. These CWE-ID classifications are grouped per student and per semester to discover the most common software vulnerabilities produced in assignment code.

Methodology: Static Analysis Tool

- We used Sonarqube Community Edition to analyze student assignment submissions for vulnerabilities and weaknesses.
- Sonarqube is a self-managed static analysis tool for continuous codebase inspection.
- Its quality model has different types of rules: reliability (bug), maintainability (code smell), and security (vulnerability and hotspot) rules.
- For this research, we are only considering issues that have a direct CWE-ID mapping, which is indicated by an issue having “cwe” as a tag. These are extracted for later analysis.

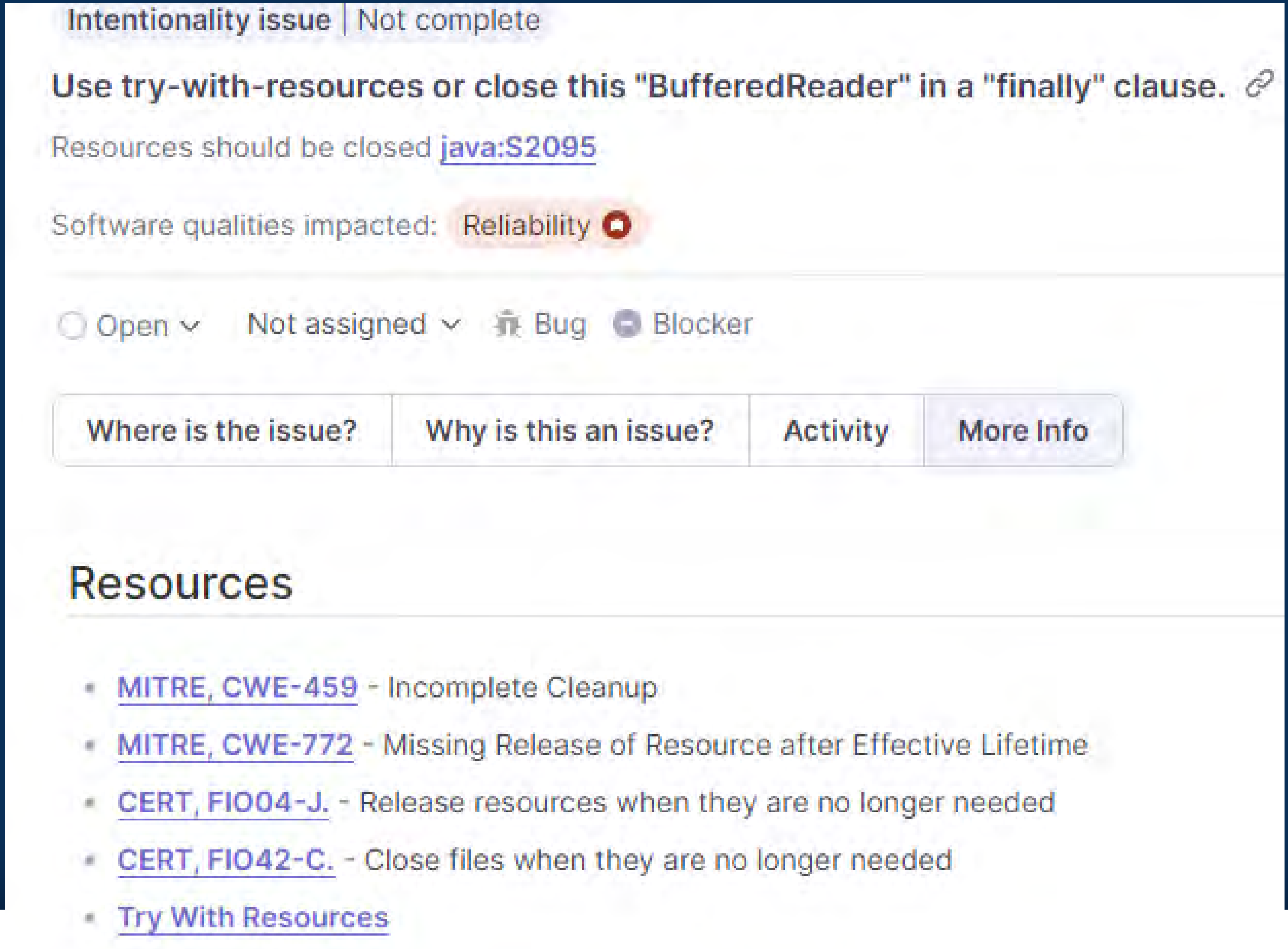
Example Assignment Analysis

- Sonarqube indicates the bugs, code smells, and vulnerabilities by line, as shown in the figure.



Example Assignment Analysis

- This figure shows a description of a bug issue and the related CWE-IDs.

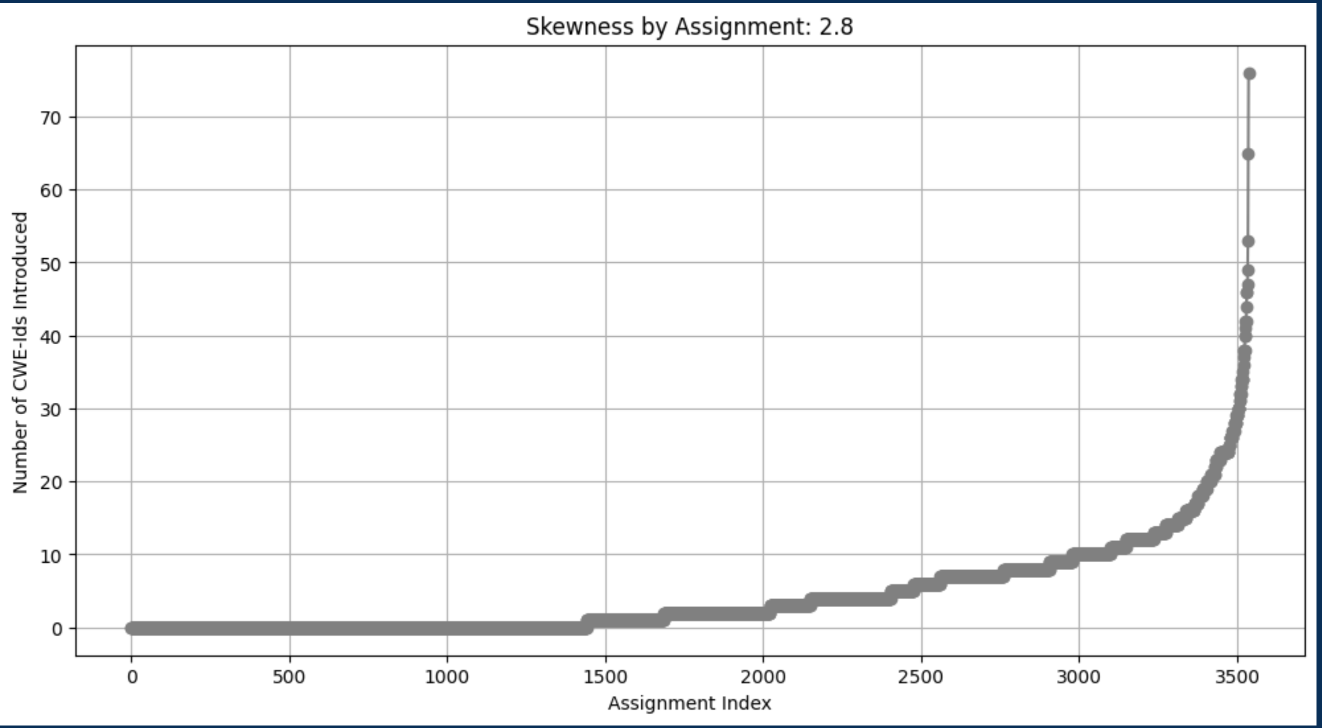


Methodology

- To answer RQ1, we used the analysis produced by Sonarqube and extracted the related CWE-IDs for each issue.
- To answer RQ2, we used the findings from RQ1 and compared the results with the commonly researched vulnerabilities as established by Hanif et al. and reported by Yilmaz et al.
  - Hanif et al. established commonly researched CWE-IDs: 78, 89, 119, 120, 190
  - Yilmaz et al. reported CWE-IDs produced by students: 259, 20, 564, 943, 480, 315, 117, 532, 778, 521, 311, 614.

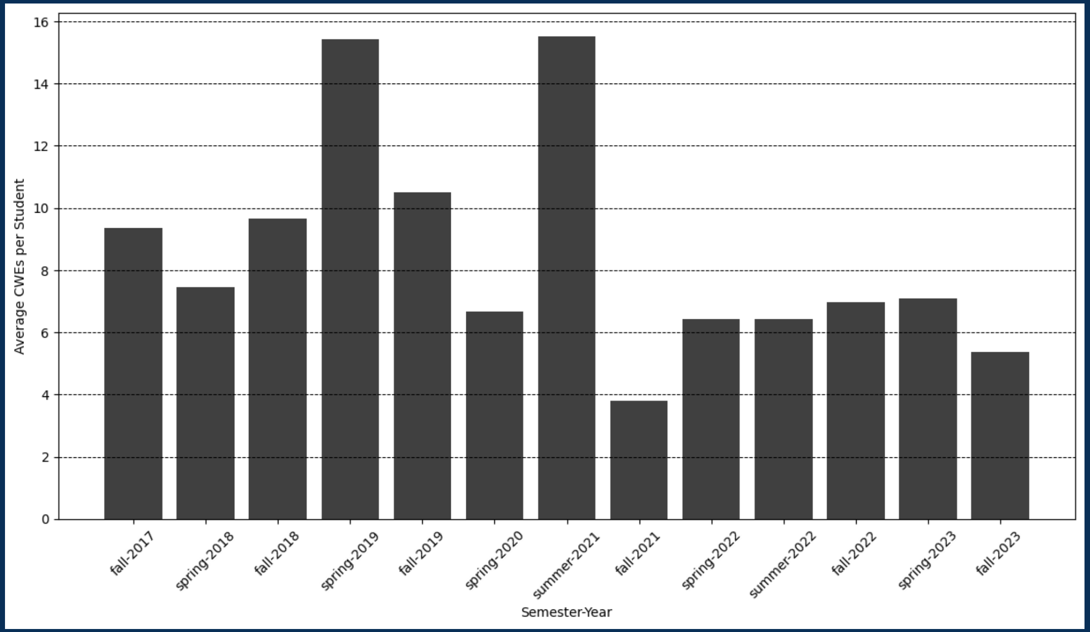
Results: RQ1

- Statistics measures of vulnerabilities found:
  - Mean: 4.37
  - Median: 2.0
  - Mode: 0
  - Min: 0
  - Max: 76
  - St. Dev.: 6.55
  - Variance: 42.92
  - Skewness: 2.83



Results: RQ1

- Of the 3537 assignments, 1442 assignment submissions did not have a mapped CWE-ID.
  - Potentially attributed to simplistic assignment submissions or blind spots in analysis software
- Heavily right-skewed, indicating only a small portion have a large number of mapped CWE-IDs.
  - Only 134 assignments had 20 or more CWE-IDs.
  - Only 558 assignments had 10 or more CWE-IDs.
- Figure shows average assignment vulnerabilities by semester and year.



Results: RQ1

- Most frequent CWE-IDs are shown in figure.

CWE-ID	Description	Frequency
546	Suspicious Comment	1502
581	Object Model Violation: Just One of Equals and Hashcode Defined	1222
476	NULL Pointer Dereference	1089
563	Assignment to Variable without Use	1049
489	Active Debug Code	870
215	Insertion of Sensitive Information Into Debugging Code	870
459	Incomplete Cleanup	833
772	Missing Release of Resource after Effective Lifetime	833
1241	Use of Predictable Algorithm in Random Number Generator	681
326	Inadequate Encryption Strength	681
330	Use of Insufficiently Random Values	681
338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	681
595	Comparison of Object References Instead of Object Contents	606
597	Use of Wrong Operator in String Comparison	606
478	Missing Default Case in Multiple Condition Expression	588
190	Integer Overflow or Wraparound	538
493	Critical Public Variable Without Final Modifier	517

Results: RQ2

- Of the Hanif et al. established commonly research vulnerabilities (CWE-IDs 78, 89, 119, 120, 190, and 306), only CWE-190 Integer Overflow or Wraparound was found to be represented in student assignment submissions. This CWE-ID had 538 occurrences in the 3537 assignment submissions.
- Of the Yilmaz et al. reported vulnerabilities produced by students (CWE-IDs 259, 20, 564, 943, 480, 315, 117, 532, 778, 521, 311, and 614), only CWE-259 Use of Hard-coded Password was found to be commonly represented in both datasets. This CWE-ID ha 41 occurrences in the submissions.
- The contrast in found CWE-IDs could be attributed to course level, programming language, or assignment requirements.
- Overall, there is little overlap, indicating a lack of consensus on what vulnerabilities are produced by students.

Discussion and Conclusion

- We studied the types of vulnerabilities produced by students in a CS2 course.
- We compared these vulnerabilities with the existing limited research on what types of vulnerabilities are commonly produced by students, and by what is commonly research is vulnerability detection studies.
- We found there is little consensus on the types of vulnerabilities produced by students and what is commonly researched.

Future Work

- More work needs to be done to establish the context in which vulnerabilities are produced
  - Programming Level
  - Programming Language
  - Developer Experience
  - Software Requirements
- Work will need to be done in adjusting existing pedagogy to reduce the introduction of the most common vulnerabilities.
- This work could potentially inform the Computer Science curriculum design in terms of software security and secure coding.