# Assignment E2a: Finding Vulnerabilities

Security Experiments and Measurements

Vrije Universiteit 2023

## Summary

During experiment E2 you filled-in a survey.

For your actual assignment, you will get the data submitted by the entire class (incl. your own) as well as data from past experiments. Your objective is to analyze the data to answer the research questions below and submit a final report for which you will follow the provided template.

## Intervention

A program file might span several hundreds lines of code and finding vulnerabilities in it during code quality controls and code review might not be compatible with the limited time available to the developers.

To allow developers/code reviewers to focus one might use a linter or static analyzer to suggest that some particular lines are responsible for the vulnerability. Still one may have to abstract away from the irrelevant instructions around the key instructions.

To help developers, a company considers to show developers only a ***slice***: a fragment of a program that is recursively extracted from a seed of some initial instructions.

* One finds the variables that are modified by those initial instructions and then collects the subsequent instructions where the values of these variables is used for the computation (forward slicing).
* Dually, one collects the variables whose value is used in the initial instructions and go back to the preceding instructions to see where these variables have been assigned (backward slicing).
* The process is iterated considering the added instructions as the new initial instructions until no more instructions can be added.

The instructions thus selected make up the slice. The remaining instructions can be deleted. We have therefore a potentially working fragment that exhibits the behaviour of interest.

An example is given below:

| Initial program | Initial Instruction | First Back/Forward | Final Slice |
| --- | --- | --- | --- |
| void call(a,b,c) {  y=c;  x=b;  y=x+2\*b;  z=a;  If (y=1)  z=b;  else  w=a;  w=c;  } | void call(a,b,c) {  y=c;  x=b;  y=x+2\*b;  z=a;  If (y=1)  z=b;  else  w=a;  w=c;  } | void call(a,b,c) {  y=c;  x=b;  y=x+2\*b;  z=a;  If (y=1)  z=b;  else  w=a;  w=c;  } | void call(a,b,c) {  y=c;  x=b;  y=x+2\*b;  z=a;  If (y=1)  z=b;  else  w=a;  w=c;  } |

The idea behind the experiment E2a is that *giving developers a slice of the file instead of the full file helps to find more vulnerabilities*. So from a company’s perspective it makes sense to invest resources (training of developers, expensive consultants, integration of toolchains, etc.) to develop and deploy a plug-in on a developer’s IDE that not only marks the line identified by the analyzers but also to show to the developer the slices (e.g. color coding the not-sliced instruction in the same color of the background).

**Controlled Factors:** The process may not work equally well for different types of vulnerabilities and therefore we may want to distinguish between those types.

## 

## Intended Measures of Success

**Measure 1:** The *actual effectiveness in terms of correct answers* of the slicing intervention measured by correctly identified vulnerable lines rather than wrong not relevant lines identified by the participant (could be measured accuracy, precision, recall, binary overall metrics found/not found per file or other appropriate metrics).

**Measure 2:** The *perceived usefulness* of the slicing intervention as collected by participants’ questionnaires with Likert scale-like questions.

These results should be controlled based on the factor describing the vulnerability.

## Process - Intervention and Control Groups

All participants have received the same training on vulnerabilities.

The artefacts reviewed by the participants included 4 files each containing a single vulnerability for which there are two versions:

* a full version from Github
* a sliced version version of the same file

|  | Full version from Github | Sliced version |
| --- | --- | --- |
| Path Traversal | path\_f.java | path\_s.java |
| User Enumeration | user\_f.java | user\_s.java |
| XSS | xss\_f.java | xss\_s.java |
| DoS | dos\_f.java | dos\_s.java |

The data at your disposal includes survey responses from six groups where various combinations of the above files have been considered.

| Group ID | Shown files |
| --- | --- |
| Group 1 | Group1\_Code |
| Group 2 | Group2\_Code |
| Group 3 | Group3\_Code |
| Group 4 | Group4\_Code |
| Group 5 | Group5\_Code |
| Group 6 | Group6\_Code |

In this simplified experiment we do not measure what would happen if the proposed file is **not** actually vulnerable. Therefore we cannot fully measure the false alert rate at the level of files but only at the level of lines.

## Ground Truth

The survey that you filled-in as part of E2a provides the answers proposed by the students acting in the role of developers. An indication of the correct answers about which lines are vulnerable in each file can be found in the after experiment directory.

You also have available a ***number of previous tests from a different university*** that you want to use to increase the power of your analysis. The results can be found at the after experiment directory.

Unfortunately there is no agreement in the literature on what makes a “correct answer” so that you can calculate it as a true positive. Below are ***possible*** answers

* *If all and only exact lines are identified and everything else is considered an incorrect answer.* 
  + While surely correct, this might be overly restrictive. If you spotted most key lines out of the handful of relevant lines it might still be sufficient to for you to fix the vulnerability, so the overall measure of success of the intervention (helping developers to find vulnerabilities) might be underestimated.
* *If the 2/3 of the ines are identified…*
* Nearby lines…
* *You build a slice with the instructions suggested by the student participant and then check whether all vulnerable instructions are inside the slice*.
  + This has the advantage of automation but might be too optimistic a result. Also, if you already have a slice (so a much smaller file) it might be that no matter what lines the participant selects, such lines will anyhow expand to the entire slice.

**You should identify your own criteria.** Once you define a criteria you should decide how do you identify the correctness of all the data points you have collected. For example you might decide to use a software, and then you should decide how do you test the correctness of the “ground truth generator”.

## Report

You are tasked with performing the data analysis using the data set of the entire class, the ground truth, and the measures above and above all discuss whether it is appropriate. You are also tasked with *presenting and* *interpreting* the results (incl. answering the RQ above) of your analysis and reporting on your final conclusion for the study.

Eventually, you will have to exercise your judgement in determining whether the effect is actually practically significant or just statistically significant or insignificant from all perspectives.

The deliverable for this assignment is a report written and submitted on Canvas according to the provided template.