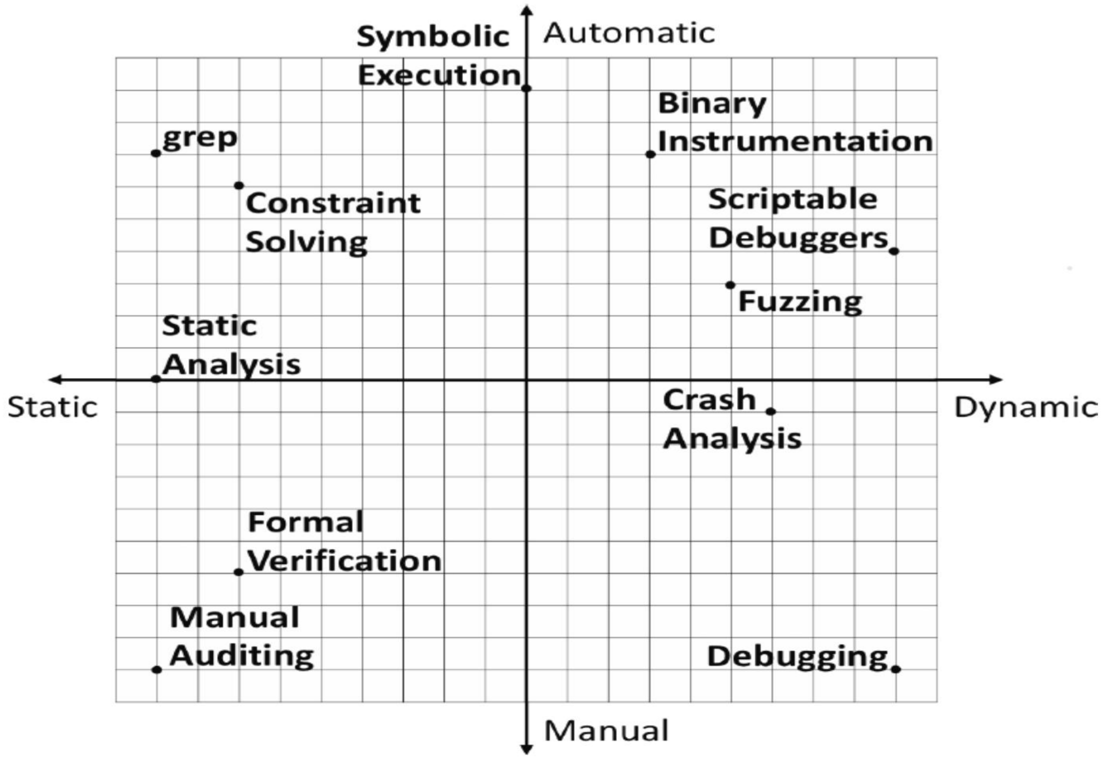
Static Analysis vs. and Dynamic Analysis

Vulnerabilities Detection

Detection of the Vulnerability [Julian Cohen's Contemporary Automatic Program Analysis]



Detection of the Vulnerability

- Detect the presence of vulnerabilities in the code during the **development**, **testing**, and **maintenance**
- Trade-off between **soundness** and **completeness**
- A detection technique is **sound** for a given category if it can correctly conclude that a given program has no vulnerabilities
    - An unsound detection technique may have ***false negatives***, i.e., actual vulnerabilities that the detection technique fails to find
- A detection technique is **complete** for a given category, if any vulnerability it finds is an actual vulnerability
    - An incomplete detection technique may have ***false positives***, i.e., it may detect issues that do not turn out to be actual vulnerabilities

Detection of the Vulnerability

- Achieving **soundness** requires considering **all executions** of a program.
  - This can be done by static checking of the program code and building up an abstraction of the program execution
- Achieving **completeness** can be done by **executioning** the program that shows the vulnerability
  - The analysis technique should provide concrete inputs that triggers a vulnerability
  - Like in software testing: the developer writes test cases with concrete inputs and specific checks for the outputs

Detection of the Vulnerability

Usually Detection tools use an **hybrid approach by using static and dynamic analysis** techniques to achieve a good trade-off between **soundness and completeness**

Static Analysis vs Symbolic Execution

- **Static analysis is any off-line computation that inspects code** and produces results about the code quality. You can apply this to source code or binary code. Static analysis can use multiple techniques (although **classic compiler control and dataflow** often figure prominently as foundation machinery for SA).
- Static analysis may use symbolic execution and inspect the resulting formula. Or it may use some other techniques such as regular expressions, classic compiler flow analyses, etc. or some combination. But **static analysis does not have to use symbolic execution**.
- **Symbolic execution** is a specific kind of off-line computation that computes some approximation of what the program actually does by constructing formulas representing the program state at various points. It is called "symbolic" because the approximation is usually some kind of formula involving program variables and constraints on their values.