**Symbolic Analysis**

**General Information**

* Author: Micah Flack
* Date: 12/09/2022
* Description: Brief introduction to symbolic and concolic analysis using the angr python library. It is a technique used in computer science and mathematics to manipulate and reason about expressions and equations symbolically, rather than by computing their values statically. It can be used by developers or researchers to do things like: identify code similarities, perform code audits/fuzzing, or identifying code vulnerabilities.

**Why You Should Care**

Symbolic analysis is a technique used in computer science and mathematics to manipulate and reason about expressions and equations symbolically, rather than by computing their values directly. This means that instead of working with specific numbers or values, symbolic analysis operates on the underlying mathematical or logical structure of an expression or equation.

One common use of symbolic analysis is in binary analysis, which is the study of the structure and behavior of binary code, such as computer programs or data files. In this context, symbolic analysis can be used to reason about the possible inputs and outputs of a binary program, as well as the possible execution paths it can take. This can be useful for tasks such as verifying the correctness of a program, or identifying potential vulnerabilities or security flaws.

For instance, the angr library can be used to solve a simple password authenticator problem through symbolic analysis. The binary being used is a dynamically linked x86 binary with a known source code that contains a hard-coded password and uses strcmp to check the input password's validity. The first step is to load the target binary into angr using the project loader. The initial program state is then set, and a path and pathgroup are created for angr to step through. Angr can then explore all possible paths in the pathgroup, and the results can be accessed by pulling information from the state of the found path. Once those paths are all read and their results are determined, angr will be able to output the password used to access the authenticator. This example shows how angr can be used to analyze and understand the behavior of a binary program.

**Main Ideas**

1. Symbolic analysis is a method of constraint solving
2. Tools like Angr and Binary Ninja have intermediary languages that can be used with other symbolic analysis tools
3. Symbolic analysis has uses beyond binary similarity detection or diffing. It even has uses in the music and writing industries for plagiarism detection.

**Future Direction**

In the context of detecting code similarities, symbolic analysis can be used to compare the underlying structure of different code fragments or programs. This can be useful for tasks such as identifying plagiarism or detecting copied or modified code. By analyzing the symbolic structure of the code, rather than just the specific values or sequences of instructions, it is possible to identify similarities that might not be obvious by looking at the code directly.

For example, two code fragments might have different values or instructions, but they might still have the same overall structure or perform the same operations. By using symbolic analysis, it is possible to compare the structure of the code and identify these similarities. Additionally, symbolic analysis can be used to identify code that has been modified or obfuscated in an attempt to hide its original source, by looking for structural changes or inconsistencies.

**Stream of Topics**

Additional topics related to this type of analysis may include some of the following:

* Using static libraries like, Pefile or LIEF
* Alternative symbolic analyzers and intermediary languages (IL), Binary Ninja
* Performing symbolic analysis with IL and machine learning
* Solving capture-the-flags (CTFs) with angr

**Additional Resources**

* [Video Presentation](https://youtu.be/XlfOQTlfXuA)
* [Github - Angr](https://github.com/angr/angr)
* [Github – Jake Springer, Angr CTF](https://github.com/jakespringer/angr_ctf)
* [Scdbg using LibEmu](https://github.com/dzzie/SCDBG)
* [Alternative Intermediary Languages for Symbolic Analysis - Binja](https://docs.binary.ninja/dev/bnil-overview.html)