Making a Decompiler out of a Compiler

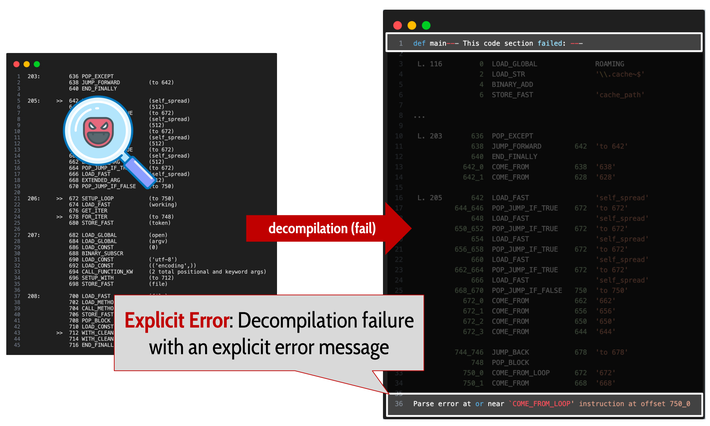
Ali Ahad, Yonghwi Kwon
University of Maryland

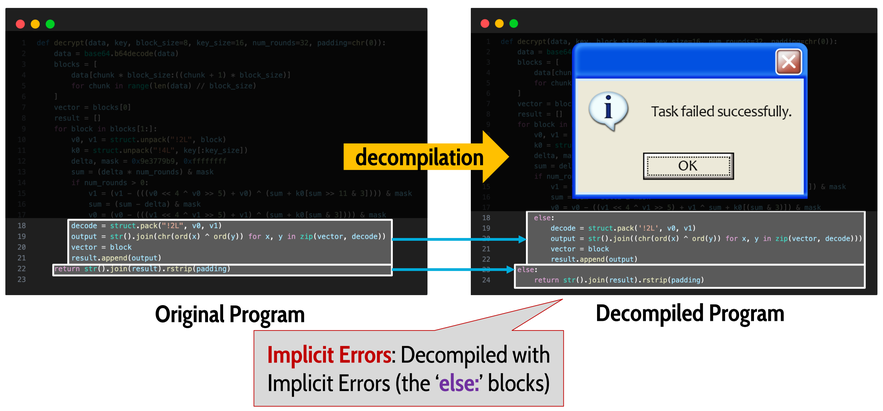
Abstract:

We propose a novel technique for Python decompilation that aims to produce semantically equivalent source code by leveraging the Python compiler. It is based on the observation that two equivalent binary files, produced from the same compiler, tend to have semantically equivalent source files. Hence, to decompile a binary, we only synthesize a source file that, after compilation, is equivalent to the binary in question. Unfortunately, synthesizing such a file randomly is both inefficient and time-consuming. Meanwhile, relying only on `large' corpora of code-to-binary mappings is unrealistic as they fail on unseen binaries. To address these, we introduce iterative Python source synthesis and mutations with similarity feedback to synthesize a source file that, after compilation, produces an equivalent binary. By doing so, our work aims to solve the limitations of current decompilation techniques and learning-based techniques.

Motivation:

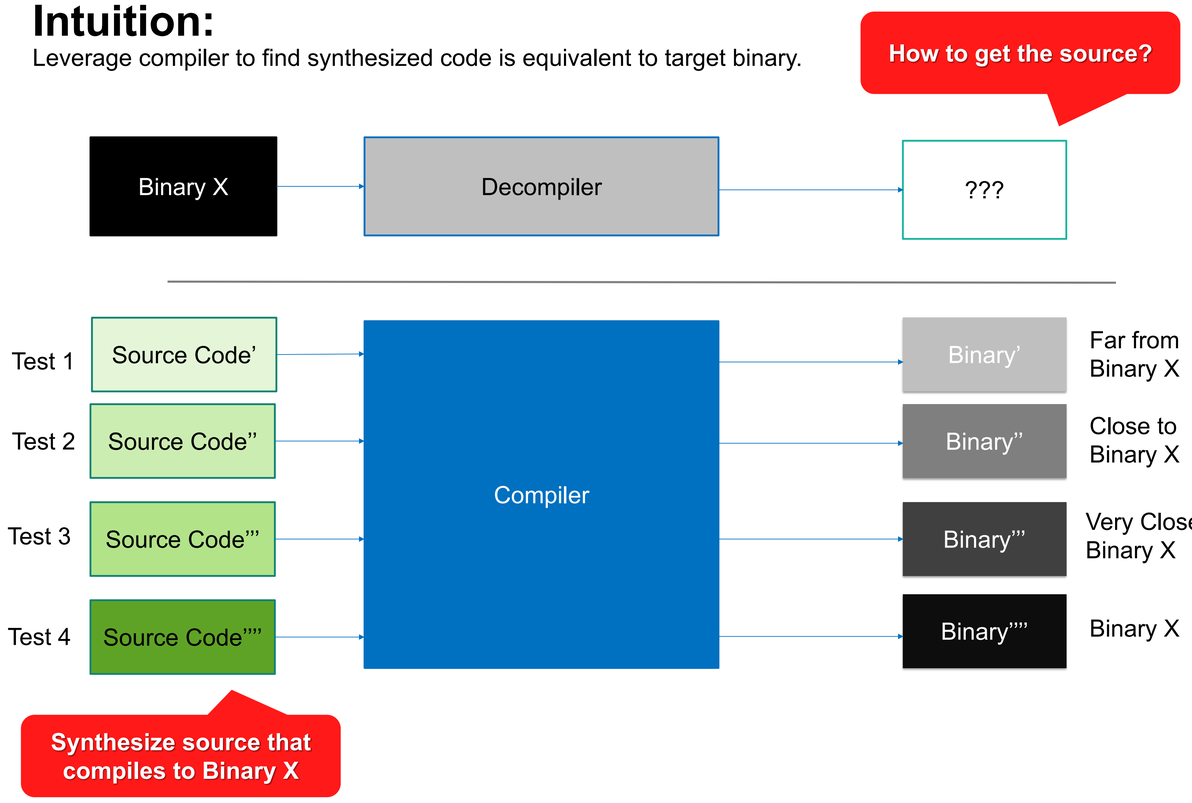
- Decompilers based off of heuristics have errors.
- Malware writers can exploit to trick forensic analyst or make forensic analysis time consuming.



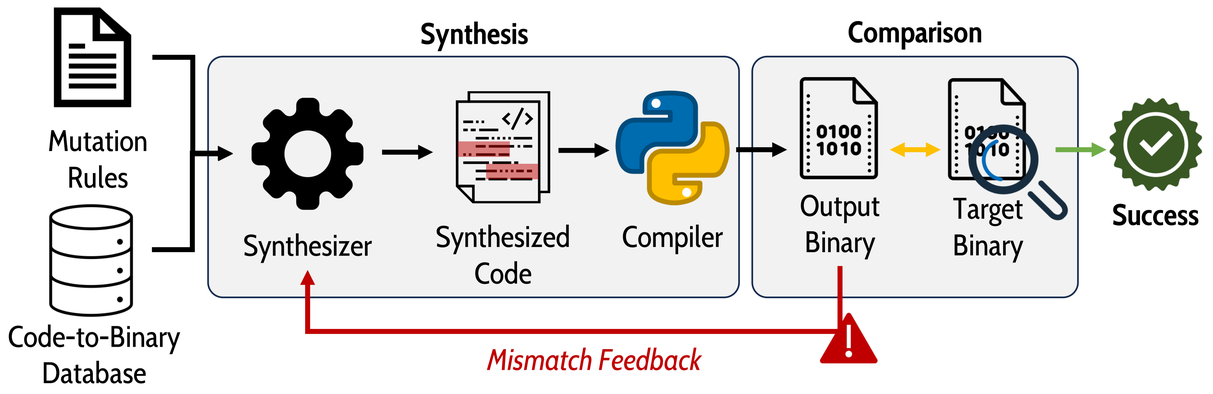


Intuition:

Leverage compiler to find synthesized code is equivalent to target binary.



Overall Solution:



UNIVERSITY OF MARYLAND