SaTC: CORE: Small: Collaborative: Evaluating Performance and Security of Executable Steganography for Surreptitious Programs

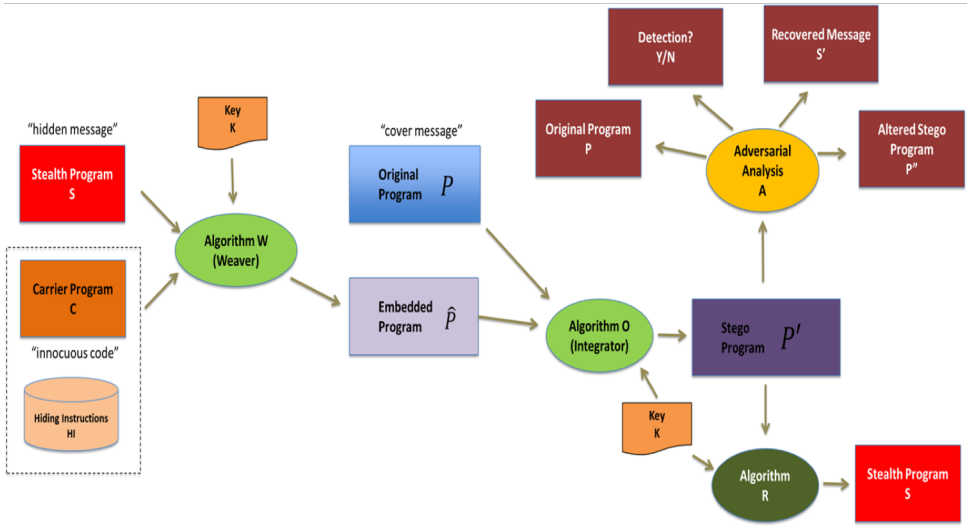
Award 1811560 (University of Nebraska at Omaha) PI: William Mahoney
Award 1811578 (University of South Alabama) PI: J. Todd McDonald

Objectives:

Contribute to the greater body of basic research analyzing the limits and uses of software protection for legitimate programs in public and private sectors

- Determine the feasibility of using executable steganography: hiding code (the hidden message) within code (the cover message)
- Develop and evaluate algorithms for interleaving (weaving) assembly level code into other assembly forms
- Utilize code weaving for integrating code to watermark and fingerprint software, supporting intellectual property protection for legitimate software developers

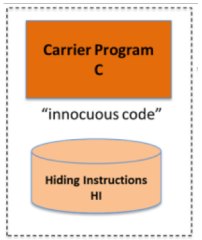
Methodology for Executable Steganography



Scientific/Technical Approach:

- Profile limits / capacity of different opcode types / create database of hiding instructions and limits for code weaving
- Develop and extend integration techniques for stealth programs that are transformed into embedded programs
- Analyze effectiveness of code weaving (executable stego) and integration as a software watermarking approach
- Perform adversarial analysis on executable stego programs

I	Position of a 1-byte immediate value
J	Position of a 2-byte immediate value
K	4-byte immediate value
L	8-byte immediate value
W	Position of a 1-byte relative offset
X	Position of a 2-byte relative offset
Y	4-byte relative offset
Z	8-byte relative offset



Accomplishments:

- Developed hiding instruction database
- Developed x86-64 instruction steganography (Weaver)
- Developed analysis framework for profiling protected applications
- Developed initial analyzer for identifying marks of Weaver insertion and profiled initial benchmark code examples

Challenges:

- Identified limitations of approach after study of hiding instruction capacities and implementation of Weaver
- Now refocusing our embedding algorithm to a block-level, architecture independent method based on LLVM intermediate representation