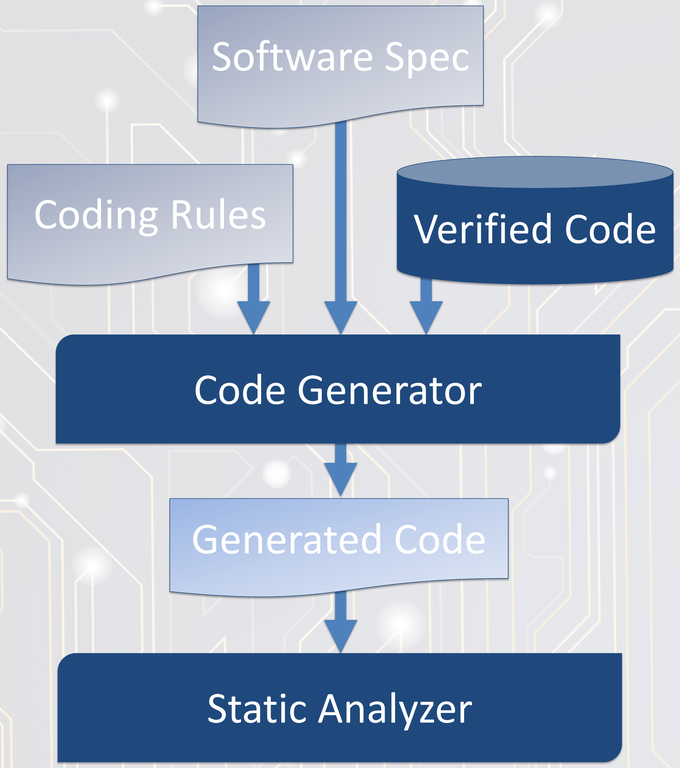
Static Analysis of Programmatically Generated Network Software: Challenges and Synergies

Jonathan Myers[1], Raymond McDowell[1], Christopher Rouff[1], Doug Williams[2], Daniel Bennett[3] [1]JHU/APL [2]Leidos [3]Shavano Systems

System Design



Targeted CWEs
CWE-120 Buffer Copy without Checking Size of Input
CWE-124 Buffer Underwrite ('Buffer Underflow')
CWE-127 Buffer Under-read
CWE-129 Improper Validation of Array Index
CWE-131 Incorrect Calculation of Buffer Size
CWE-190 Integer Overflow or Wraparound
CWE-456 Missing Initialization of a Variable
CWE-476 NULL Pointer Dereference
CWE-676 Use of Potentially Dangerous Function
CWE-754 Improper Check for Unusual or Exceptional Conditions
CWE-805 Buffer Access with Incorrect Length Value
CWE-822 Untrusted Pointer Dereference
CWE-825 Expired Pointer Dereference

Motivation

- Automatically generated code *should* be better-suited to analysis and verification than hand-written code, but
  - Numerous false positives: static analyzers may misunderstand paradigms used by code generators
  - False negatives: current analyzers may not identify the types of errors introduced by code generators
- We developed code gen system and analysis system in tandem:
  - Generated code is created with static analysis in mind
  - Static analyzers are tuned to the code generation system
- This allows rigorous analysis of generated code and high confidence in its quality

Methods

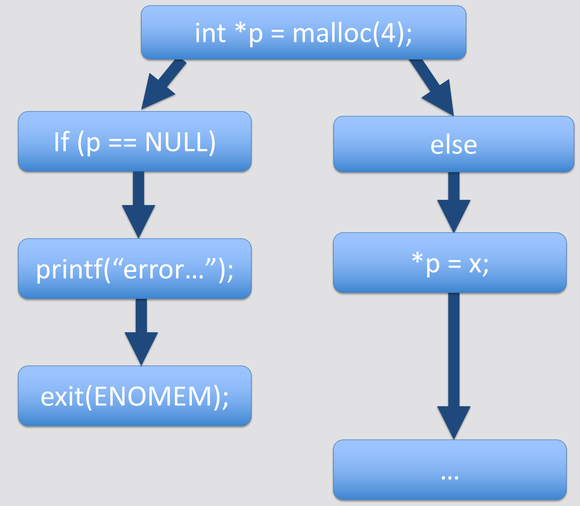
- Formal Verification of Safety
  - Use Frama-C's WP plugin to perform software verification on select components, verify safety
  - Create verified memory-safe buffer library for use by generated code
- Coding Standards and Parse Tree Analysis
  - Craft coding rules that restrict language features to a safe subset
  - Enforce coding rules through extensions to Clang Static Analyzer
  - Severely limit use of pointers or raw C buffers in unverified code

Example

```
/*@ precondition: \valid(p+(0..sz)); */
void foo(int* p, uint sz) {
   for (uint i = 0; i < sz; i++) {
     p[i]++;
   }
}
```

- A function which is memory-safe if its code contract is respected
- Can be verified with Frama-C WP
- Calling code must be verified for precondition

Example



- Absence of CWE-754 (Improper Check for Unusual or Exceptional Conditions) can be determined using only a parse tree
- Code generation system always performs safety checks in predictable fashion
- Static analyzer will raise an error if the safety check is not in its expected location

JOHNS HOPKINS APPLIED PHYSICS LABORATORY

This work was performed as part of AGNES: Automatic Generation of Network Element Software, a project funded by the Office of Naval Research and a collaborative effort between Leidos, Shavano Systems and JHU/APL.