Two possible options

- Translate all PHP constraints to C code and then generate llvm bytecode.
  - Put the constraints into the condition of an synthesized error, e.g. assert(False);
  - KLEE detects the error and generates input to trigger.
  - Only need to support a subset of PHP code in the translation.
  - Totally relies on KLEE to solve constraints.
- Try to get symbolic return value of built-in functions and combine to PHP constraints.
  - No direct APIs to get symbolic return value
  - Also need to put built-in functions to conditions and analyze the constraints generated by KLEE.

Current Progress

- There are tools to compile whole C/C++ libraries to LLVM bytecode.
- Try to play with KLEE by applying to PHP interpreter.
  - Compile PHP interpreter with clang and get its LLVM bytecode
  - Failed!
  - KLEE is not completely implemented
  - Some intrinsic (C built-in) functions are not supported and raise error: (inline) assembly code.
  - Trying to apply to only one source file still has this problem.
    - Also include all library in compilation.
    - Hard to figure out which library/file causes this error

Current Progress (cont.)

php-src/ext/standard/string.c

-
  ```
  PHP_FUNCTION(str_replace)
  {
          php_str_replace_common(INTERNAL_FUNCTION_PARAM_PASSTHRU, 1);
  }
  ```
- `static void php_str_replace_common(INTERNAL_FUNCTION_PARAMETERS, int case_sensitivity)`

-
  ```
  struct _zend_execute_data {
          const zend_op       *opline;           /* executed opline                */
          zend_execute_data   *call;             /* current call                   */
          zval                *return_value;
          zend_function       *func;             /* executed function              */
          zval                 This;             /* this + call_info + num_args    */
          zend_execute_data   *prev_execute_data;
          zend_array          *symbol_table;
          void               **run_time_cache;   /* cache op_array->run_time_cache */
  };
  ```