Instructions for code generation

整个代码生成过程可以简要的分成三块：定义代码模板、生成语句、组合语句

key files

- templates.py
  - 定义代码模板的地方，如：

```
#template for Wild pointer
PTR_INIT_PAIRS = [
    ("int* $ptr_var;", None),
    ("int* $ptr_var2;", "$ptr_var2 = new int(rand());"),
    ("int $target_var;", "$target_var = rand();")
]

UNSAFE_DEREFERENCE_LINES = ["*$ptr_var = $target_var;"]

SAFE_DEREFERENCE_LINES= ["*$ptr_var2 = $target_var;"]
```

（图1）

定义了野指针解引用的错误，（示例代码如下）

```
int entity_0;                  // Tag.BODY
int* entity_7;                 // Tag.BODY
int* entity_5;                 // Tag.BODY
entity_7 = new int(rand());    // Tag.BODY
entity_0 = rand();             // Tag.BODY
*entity_5 = entity_0;          // Tag.PTR_DEREF_UNSAFE
return 0;                      // Tag.BODY
```

（图2）

  - 图1代码模板中，*PTR_INIT_PAIRS*是**解引用前的铺垫代码**，对应图2的前五句
  - 图1中的后两句, *UNSAFE_DEREFERENCE_LINES*和*SAFE_DEREFERENCE_LINES*是**解引用代码**，也就是有可能产生漏洞的地方。
- sa_tag.py
  - 由于神经网络的输入必须是digital而不能是literal language，所以需要将训练集的代码逐句编码，即每一句代码根据其类型赋予一个tag值。神经网络训练时，一个代码文件的tag值序列作为输入。

```
# Function wrapping lines
OTHER = 0
# Lines inside body that aren't buffer writes
BODY = 1
# Buffer write that requires control flow analysis
BUFWRITE_COND_SAFE = 2
# Buffer write that requires control flow analysis
BUFWRITE_COND_UNSAFE = 3
# Buffer write that is provably safe even without
BUFWRITE_TAUT_SAFE = 4
# Buffer write that is provably unsafe even withou
BUFWRITE_TAUT_UNSAFE = 5
# Pointer dereference safe
PTR_DEREF_SAFE = 6
# Pointer dereference unsafe
PTR_DEREF_UNSAFE = 7
```

(sa_tag.py)

- 所以上图2中的七行代码最后在训练时的输入是[1,1,1,1,1,7,1]

- generate.py
  - 要修改的地方：
    - 110行，main函数内的generator（一个函数指针数组，存放的是各漏洞代码生成函数的地址），我们把自己的漏洞代码生成函数写好之后，把函数指针加在这里就可以了
  - 要添加的东西（以野指针解引用为例）
    - 没别的，就是添加好自己的漏洞代码生成函数
    - 生成过程很简单，就把填空词给生成好，然后填入templates.py中定义的模板，写好这句代码的safe（是否有漏洞）
    - 比如：

```
def gen_unsafe_deref_example(include_cond_bufwrite = True):
    anon_vars = _get_anon_vars()
    ptr_var, ptr_var2, target_var = anon_vars[:3]
    substitutions = {
        'ptr_var': ptr_var,
        'ptr_var2': ptr_var2,
        'target_var': target_var
    }
    safe = False
    ptr_init_pairs = templates.PTR_INIT_PAIRS
    dereference_lines = templates.UNSAFE_DEREFERENCE_LINES

    return _assemble_ptr_example(ptr_init_pairs, dereference_lines, safe, substitutions)
```

通过_get_anon_vars()函数（自带）生成了十个填空词，然后取了三个，用substitution填空进去。

- 然后_assemble_ptr_example函数，是自己写的把这些代码语句组合起来的一个函数。

注意事项

- 尽量复用作者的接口，按照它的规范来。