深圳大学计算机与软件学院

软件理论实验室

MemLock: Memory Usage Guide Fuzzing

https://github.com/wcventure/MemLockFuzzer

扫描二维码获取更多信息

https://icse2020-memlock.github.io/

MemLock: Memory Usage Guided Fuzzing

Cheng Wen, Haijun Wang, Yuekang Li, Shengchao Qin, Yang Liu, Zhiwu Xu, Hongxu Chen, Xiaofei Xie, Geguang Pu and Ting Liu







Uncontrolled memory consumption

- **A kind of critical software security weaknesses**
  - The software does not properly control the allocation of a limited resource thereby enabling an actor to influence the amount of resources consumed, eventually leading to the exhaustion of available resources (from CWE-400).

- **Memory consumption vulnerability:**
  - CWE-674: Uncontrolled Recursion
  - CWE-789: Uncontrolled Memory Allocation
  - CWE-401: Missing Release of Memory after Effective Lifetime.

CVE-2018-17985

```
1  struct demangle_component *
2  cplus_demangle_type (struct d_info *di) {
3
4    // "peek" is a single character extracted from the input directly
5    char peek = d_peek_char (di);
6
7    switch (peek){
8      ...
9      case 'P':
10       ret = d_make_comp (di,
11         DEMANGLE_COMPONENT_POINTER,
12         cplus_demangle_type (di), NULL);
13       break;
14     case 'C':
15     ...
16   }
17   ...
18 }
```

Code Snippet from *cp-demangle.c* in *Binutils v2.31*

CVE-2018-4866

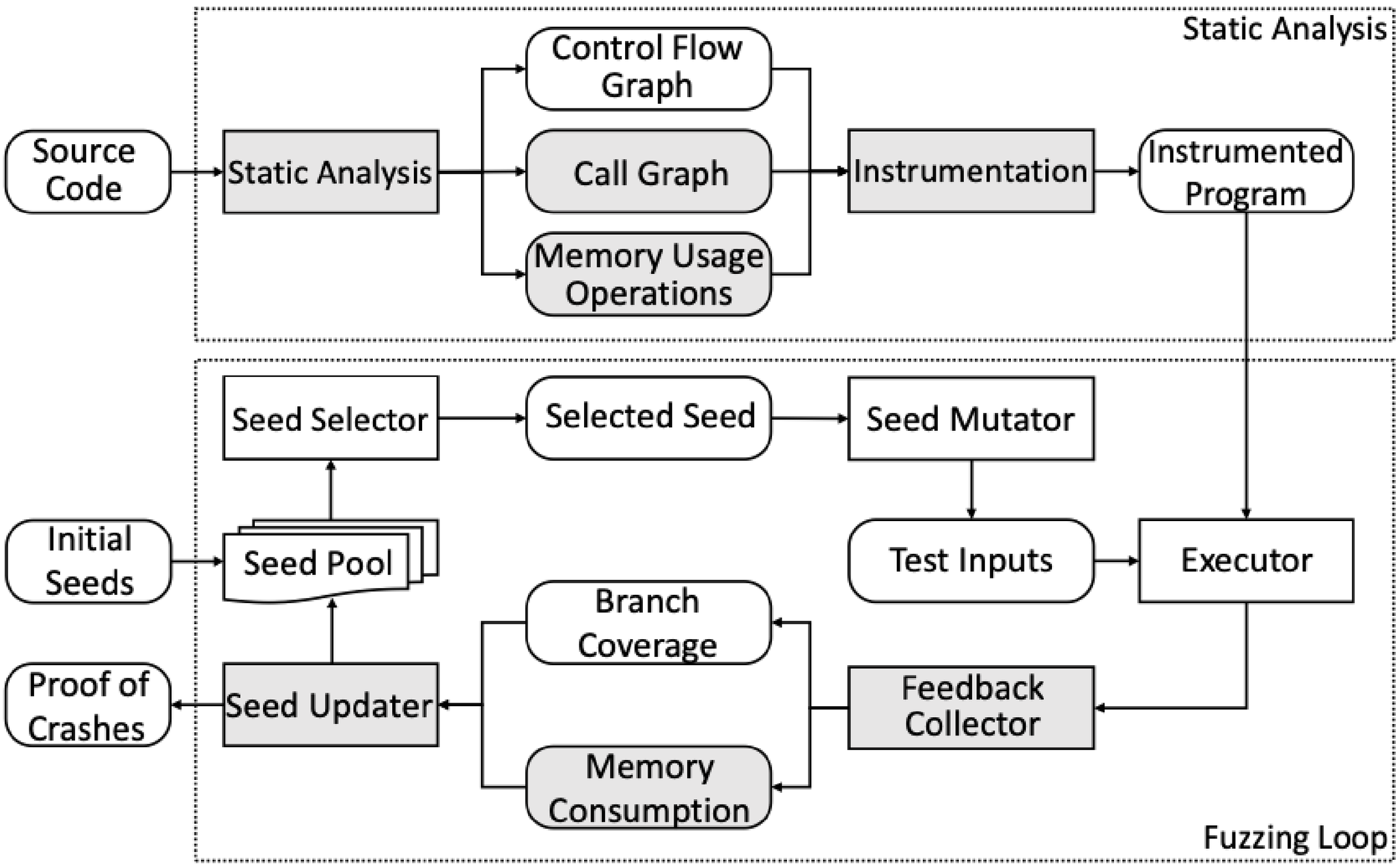
```
1  class EXIV2API DataBuf {
2  public:
3    // Constructor with an initial buffer size
4    explicit DataBuf(long size): pData(new byte[size]), size(size) {}
5    ...
6    byte* pData; // Pointer to the buffer
7    size_t size; // The current size of the buffer
8  };
9
10 void Jp2Image::readMetadata() {
11   while (io_->read((byte*)&subBox, sizeof(subBox)) ==
     ↪  sizeof(subBox) && subBox.length ) {
12     subBox.length = getLong((byte*)&subBox.length, bigEndian);
13     DataBuf data(subBox.length); // Allocation without checking
14     ...
15     io_->seek(position - sizeof(box) + box.length, BasicIo::beg);
16   }
17 }
```

Code Snippet from *jp2image.cpp* in *Exiv2 v0.26*

MemLock: Memory Usage Guided Fuzzing

- Use light-weight program **instrumentation** to collect the memory consumption information.
- Guide the fuzzing process with new-dimensional **feedback** guidance information--memory consumption.
- Through a novel **seed updating scheme** to retain the most interesting input for each path.

MemLock Overview



Instrumentation

- **Instrument the program based on the static analysis of**
  - Control Flow Graph (CFG)

    add instrumentation into every edge of the CFG to obtain the coverage information.
  - Call Graph

    add instrumentation into both the entry and the exit of the function call to obtain the recursive depth.
  - Memory Usage Operations

    add instrumentation into the memory allocation/deallocation functions and obtain its parameters and return value.

Feedback

- **Fuzzing is Guided by two-dimensional information**
  - Coverage

    guides the exploration of different program paths.
  - Memory consumption

    guides the search for those program paths that exhibit more memory consumption.

Seed Updating

