Notes about Juliet Test Suite

Michal

7 November 2021

1 What it contains

It contains example programs (called test cases) that include weaknesses (potential vulnerabilities) categorized by **Common Weakness Enumeration** (CWE) (e.g. top 25 CWEs).

Example programs are **artificial**, meaning they were created with the sole purpose of presenting flaws/vulnerabilities/weaknesses.

Typically, each CWE directory, contains multiple examples. And typically each of examples have the **baseline** version (marked with **_01**) which contains the simplest form of a program, and multiple more complex versions (marked with **_02+**) created to more closely resemble normal (non-artificial) programs.

Baseline programs seem the most interesting from my point of view.

Each program has (only) 1 **"bad"** function containing weakness, and (at least) 1 **"good"** function with the same code except the weakness. *OMITBAD* and *OMITGOOD* preprocessor macros may be supplied to compiler to compile bad/good functions only.

1.1 Example: CWE121_Stack_Based_Buffer_Overflow__CWE131_memcpy

Bad function

```
void CWE121_Stack_Based_Buffer_Overflow__CWE131_memcpy_01_bad()
{
    int * data;
    data = NULL;
    /* FLAW: Allocate memory without using sizeof(int) */
    data = (int *)ALLOCA( 10 );
    {
        int source[10] = {0};
        /* POTENTIAL FLAW: Possible buffer overflow if data was not allocated
    correctly in the source */
        memcpy(data, source, 10*sizeof(int));
        printIntLine(data[0]);
    }
}
```

Good function

```
/* goodG2B uses the GoodSource with the BadSink */
static void goodG2B()
{
    int * data;
    data = NULL;
    /* FIX: Allocate memory using sizeof(int) */
    data = (int *)ALLOCA( 10*sizeof(int) );
    {
        int source[10] = {0};
        /* POTENTIAL FLAW: Possible buffer overflow if data was not allocated
    correctly in the source */
        memcpy(data, source, 10*sizeof(int));
        printIntLine(data[0]);
```

```c
    }
}

void CWE121_Stack_Based_Buffer_Overflow__CWE131_memcpy_01_good()
{
    goodG2B();
}
```

2 Use cases

2.1 Main use case

Usually it is used for testing efficiency of static source/binary scanners. This could look like:

- Compile 100 programs with **bad** only function.
- Compile 100 programs with **good** only function.
- Scan all 200 programs with a binary scanner (without running programs at all)
- Any of the 100 **bad** programs where weakness wasn't found is a false negative.
- Any of the 100 **good** programs where weakness was found is a false positive.

2.2 CHERI heap-safety use case

The creators of Cornucopia heap-safety algorithm for CHERI used it differently. They compiled *use-after-free* and *double-free* CWE programs and ran them on CHERI-based system. Any program that completed execution without raising exception was classified as false negative (meaning that CHERI didn't manage to detect and prevent it).

2.3 How can we use it

For a strictly online/run-time based anomaly detection system it may not have a great value.

Juliet Test Suite includes large number of different potential vulnerabilites, but it doesn't really exploit them, it doesn't change the flow of the program. In our case using **good** only programs for training and **bad** programs for testing would be no different than compiling a set of any distinct programs and randomly splitting them (into 'bad' and 'good').

3 Customization

All files and their functions have consistent naming convention. Together with provided python scripts (that come with the Juliet Test Suite), it's flexible in terms of what programs can be generated.

It provides 118 CWEs of which 68 are available for non-Windows systems (68 have Makefiles). A python script (**create_per_cwe_files.py**) is provided for automatic generation/customizing of these Makefiles.

4 Useful documents

User guide for v1.2 (looks like the only document that actually explains how to use the suite, including compilation information)

Changes in v1.3 (1.3 is the most recent version, this document was last updated in 2019, the "1995" is not a year btw)