CS5218 Assignment2 Report

Guo Shijia A0191309E

March 29, 2019

1 Taint Analysis Design

The data flow equations for taint variables analysis are:

$$TV_{entry}(l) = \begin{cases} \emptyset, & if \quad l = init(S_F) \\ \cup\{TV_{exit}(l^{'})|(l^{'}, l) \in flow(S_F)\}, & otherwise \end{cases}$$

$$TV_{exit}(l) = \big(TV_{entry}(l) \backslash kill_I(B^l)\big) \cup gen_I(B^l), where B^l \in block(S_F)$$

kill and gen functions are show below:

$$gen_I([z := a]^l) = \begin{cases} \{z\}, & if \quad \exists a^{'} \in FV(a), a^{'} \in TV_{entry}(l) \cup TV_{before}(l) \\ \emptyset, & otherwise \end{cases}$$

$$gen_I([skip]^l) = \emptyset$$

$$gen_I([b]^l) = \emptyset$$

$$kill_I([z := a]^l) = \begin{cases} \{z\}, & if \quad \forall a^{'} \in FV(a), a^{'} \notin TV_{entry}(l) \cup TV_{before}(l) \\ \emptyset, & otherwise \end{cases}$$

$$kill_I([skip]^l) = \emptyset$$

$$kill_I([b]^l) = \emptyset$$

Combine above formula, the monotone framework is defined as:

$$L = P(Var_*)$$

$$\subseteq = \subseteq$$

$$\cup = \cup$$

$$\perp = \emptyset$$

$$_{l} = \emptyset$$

$$E = \{init(S_*)\}$$

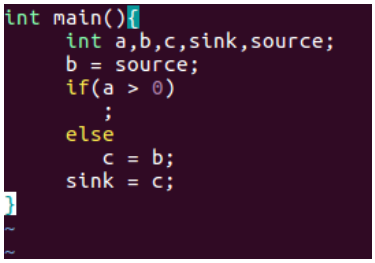
$$F = flow(S_*)$$

$$\mathcal{F} = \{f : L \to L | \exists l_k, l_g : f(l) = (l \backslash l_k) \cup l_g\}$$

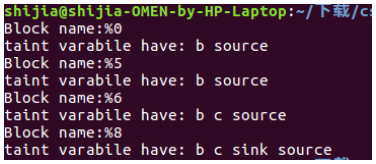
$$f_\ell(l) = \left(l \backslash kill(B^\ell)\right) \cup gen(B^\ell), where B^\ell \in blocks(S_*)$$

2 Taint Analysis Algorithm Implementation

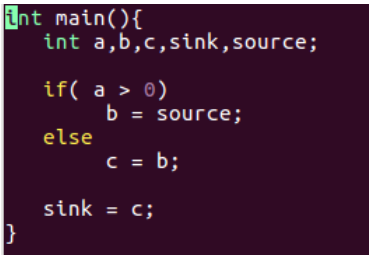
In order to perform taint variable analysis, i would use the chaotic iteration algorithm introduced in the lectures. we have below observations from the taint variables analysis: 1)if an untaint variable is assigned by an expression containing a taint variable, then the variable will be taint. 2) if a taint variables is assigned by a constant or an expression do not contains any taint variables, the the variable will be untaint. 3) if in a block, a variables is taint, then in the Successors of that block, the variable will still taint.
Based on those observations, my implementation as follows: 1) we pre-define an order to process the each block 2)we gather the interaction relationships between variables in each block. 3)we maintain the taint variables in each blocks and continue to update, until got the fixed point.
Here are an important point is how we gather the interaction relationships between variables in each block. More specifically, we focus on 3 instructions: Store, Load and Icmp. Below is the process flow: 1) when we encounter load instructions, we store the variables into a set A. 2) when we encounter Icmp instructions, we clear set A. 3) when we encounter Store instructions, if store an constant to a variable, we remove the record whose key equals that variable, else we update that record as (variable, set A).
Then we process the blocks follow the pre-defined order, use the predecessor block's information and interaction relationships to update the information until we got the fixed point.
the compile command:
clang++-3.5 -o assignment2 Assignment2.cpp ‘llvm-config-3.5 –cxxflags‘ ‘llvm-config-3.5 –ldflags‘ ‘llvm-config-3.5 –libs‘ -lpthread -lncurses -ldl
the execute command:
./assignment2 testcase1.ll
some testcases and result:



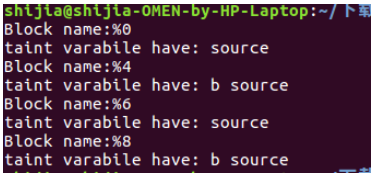
(a) source code



(b) result

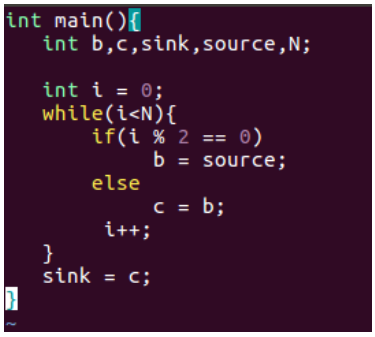


(c) source code

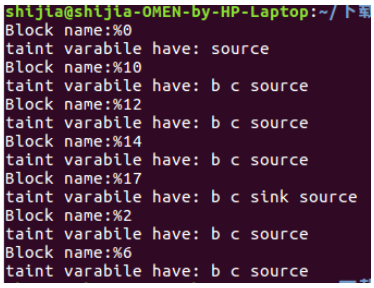


(d) result

Figure 1: Task2 Example testcase

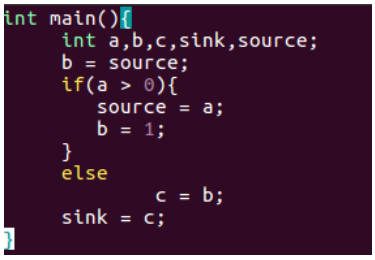


(a) source code

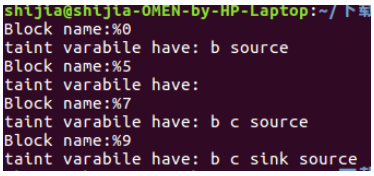


(b) result

Figure 2: Task3 Example testcase

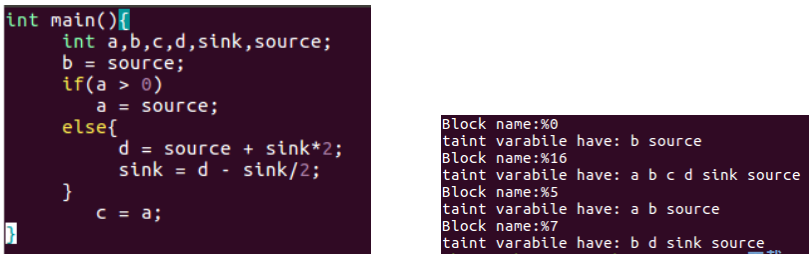


(a) source code



(b) result

Figure 3: untaint testcase



(a) source code (b) result

Figure 4: contains arithmetic operations testcase