|  |  |
| --- | --- |
| **Table 1.** Taint inheritance of different SIMPLE language elements. | |
| **Expression** | **Evaluated Value Has Taint If** |
| e + e’ |  |
| e – e’ |  |
| e \* e’ |  |
| e / e’ |  |
| e == e’ |  |
| e != e’ |  |
| e >= e’ |  |
| e <= e’ |  |
| e && e’ |  |
| e || e’ |  |
| -e |  |
| !e |  |
| mem[e] |  |
| var |  |
| c | NEVER |
| get\_int() | NEVER |
| get\_secret\_int() | ALWAYS |
| l = e |  |
| if (e) then {s\*} else {s\*} | NEVER: Control flow leaks not checked |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 2.** Taint inheritance for && expression. | | | | | |
| **&&** | | **e'** | | | |
| **FALSE** | **FALSE** | **TRUE** | **TRUE** |
| **e** | **FALSE** | FALSE | FALSE | FALSE | FALSE |
| **FALSE** | FALSE | FALSE | FALSE | FALSE |
| **TRUE** | FALSE | FALSE | TRUE | TRUE |
| **TRUE** | FALSE | FALSE | TRUE | TRUE |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 3.** Taint inheritance for || expression. | | | | | |
| **||** | | **e'** | | | |
| **FALSE** | **FALSE** | **TRUE** | **TRUE** |
| **e** | **FALSE** | FALSE | FALSE | TRUE | TRUE |
| **FALSE** | FALSE | FALSE | TRUE | TRUE |
| **TRUE** | TRUE | TRUE | TRUE | TRUE |
| **TRUE** | TRUE | TRUE | TRUE | TRUE |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table 4. Taint inheritance for \* expression (similar to &&) | | | | | |
| **\*** | | **e'** | | | |
| **0** | **0** | **!0** | **!0** |
| **e** | **0** | 0 | 0 | 0 | 0 |
| **0** | 0 | 0 | 0 | 0 |
| **!0** | 0 | 0 | !0 | !0 |
| **!0** | 0 | 0 | !0 | !0 |

tables.h typedefs an unsigned int called value\_t. We can change value\_t to a struct containing the unsigned int, “value”, and the bool, “tainted”, which indicates whether the value is tainted.

Then we need to ensure that in eval.c, which returns a type of value\_t, instead of just adding, subtracting, or referencing a the value\_t, we perform the arithmetic operations on value\_t.value, and perform the taint-flow operations on value\_t.tainted to determine the flow of the taint.

Then, in eval\_stmts, when a PRINT command is encountered, we simply have to check the taint to see if we print out any special print statements, or if we can print out the correct value.

I’ve already \*slightly\* modified the code in tables, so if you do a git pull, what you pull won’t exactly match what we originally downloaded for the assignment.

Anyway, this looks good, and I feel like it’s going to be a breeze.

-John