M2 CySec **UGA / Grenoble INP**

Exercises on code analysis techniques

Abstract Interpretation (value set analysis)

In the following we consider abstract interpretation on programs using the interval abstract domain.

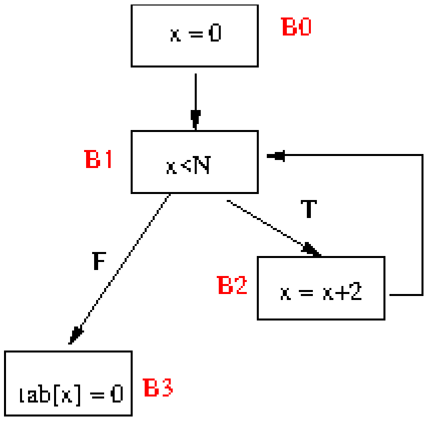
Exercise 1

We consider the following C code and its control-flow graph :

```
#define N 3

int x ;
int Tab[N] ;

x = 0 ;
while (x<N)
      x = x+2 ;
tab[x] = 0
```



Q1. Compute the value sets at each entry/exit points of each basic blocks without using any acceleration technique (i.e., widening/narrowing).

Q2. Same as Q1, but using widening/narrowing operators.

Q3. Same as Q2 by replacing the constant 3 by the constants1000 and 1001.

Q4 . What can we conclude about potential program vunerabilities ?

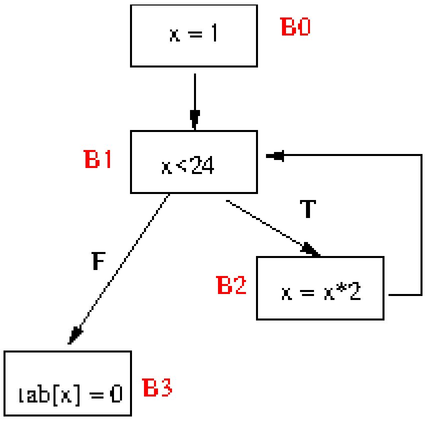
Exercise 2

We consider the following C code and its control-flow graph :

```
#define N 33

int x ;
int Tab[N] ;

x =  1;
while (x<N)
      x = x*2 ;
tab[x] = 0
```



Q1. Compute the value sets at each entry/exit points of each basic blocks using acceleration techniques (i.e., widening/narrowing).

Q2 . What can we conclude about potential program vulnerabilities ?

Q3. How could we get more precise results with Frama-C ?

Symbolic Execution

Exercise 3

We consider the following code, where variable x is a user input :

```
#define N ...
unsigned x, y z ;
int T[N] ;

read(x) ;
z = 2*x ;
if (z<x+20) {
    y = z -10
    if (y > 12)
         T[y] = 0 ;
    else
         T[x] = 0 ;
} else {
         T[z+ 3] = 0 ;
}
```

Q1. Give its sets of execution paths and corresponding path predicates

Q2. Is there a valid input valuation for each of these path predicates ?

Q3. How to extend theses path predicates in order to detect potential buffer overflows ?

Exercise 4

We consider the following code example , where x is a **positive user input** :

```
#define N 3

int x ;
int Tab[N] ;

while (x<N)
     x = x+2 ;
tab[x] = 0
```

Q1. Is a symbolic tool like PathCrawler able to find **all** the execution paths triggering the vulnerability ? Explain your answer (giving the set of path predicates to consider).

Q2 . Same question with N=1000

Exercise 5

Give some program (small) code examples containing vulnerabilities that would **not** be found by an automated symbolic execution engine.