CS510 Assignment #2 (Due March 2nd in class)

February 8, 2017

1 Dynamic Control Dependence (20p)

```
1.    if (p1)
2.       return;
3.    while (p2) {
4.       if (p3)
5.          break;
6.       if (p4) {
7.          s1;
8.       } else {
9.          s2;
10.         continue;
11.      }
12.      s3;
13.   }
14.   if (p5 ||
15.       p6) {
16.          s4;
17.   }
```

Consider the above code snippet. Assume the execution trace is 1, 3, 4, 6, 7, 12, 3, 4, 6, 9, 10, 3, 4, 5, 14, 16, 17.

Construct the dynamic control dependence subgraph, i.e., the graph that reveals control dependences between executed statements. Please show stepwise control dependence stack state and the dependence detected.

2 Dynamic Data Dependence (20p)

```
1. void (* F) ();
2. char A[1];
3. char B[10];
4. int i,j;

5. i=j=0;
6. read(B, 10); //read 10 bytes
```

```
7. F= &foo();
8. while (j<10) {
9.   if (B[j]=='b')
10.     break;
11.  j=j++;
12.  if (j>0)
13.     i++;
14.  (*F) ();
15. }
16. A[i]=B[j];
17. (*F) ();
```

Data provenance tracking is a technique that tracks the set of INPUT VALUES that a variable or an executed statement is dependent on. For example, assume a program execution is

```
1. read (buf, 2) with input 10 and 20;
2. x=buf[0];
3. y=x+buf[1];
```

.

The provenance of x and y are $\{10\}$ and $\{10, 20\}$, respectively. Data provenance can be used to defend against code injection attacks by not allowing a function call to have a non-empty provenance.

(a) (10 points) Sketch a forward online algorithm that computes data provenance forwards along program execution, considering both data and control dependences.

(b) (10 points) Assume the input is "cb", apply your algorithm to the program at the beginning to detect code injection vulnerabilities. Note that function pointer F and array A are next to each other on the stack so that $A[1]$ shares the same memory location with the first byte of F.

3 Formulating Dynamic Analysis (20p)

Please write the formal semantics for tracking dynamic control dependences on-the-fly. Please use the language that supports functions, which is the language on page 15 of the slides for program semantics. The output of the analysis shall be a trace of dynamic control dependences, which includes all the dynamic control dependences exercised during execution.

- Define the semantics configuration (5p).
- Define the semantics rules (15p).

4 Static Analysis (20p)

Design an analysis to determine the sign of the possible values of a variable, all negative numbers by the symbol -, zero by the symbol 0, and all positive numbers by the symbol +. Assume only int type is supported. Only two kinds of binary operations are possible: addition and subtraction. There may be predicates and loops.

- Determine the abstract domain and the abstract semantics that analyzes individual paths (7p).
- Argue that your analysis terminates in the presence of loops. You can use example if needed.(6p)
- Rewrite your analysis in the form of data flow analysis such that the analysis does not need to traverse the individual paths. (7p)

5 Static Analysis (20p)

```
1.  int y, t, p;
2.  float x,z;
3.  x=random(0.0, 1.0); /*x is a random sample from [0.0,1.0];*/
4.  y=random(-50,50);   /*y is a radom sample in [-50,50];*/
5.  if (y<0)
6.     p= -y;
7.  else
8.     p=y;
9.  z=100;
10.  while (p>0) {
11.    z=z*x;
12.    p=p-1;
13. }
14. output(z);
15. output(p);
```

- Apply range analysis to the above program and compute the range of variables using the (data flow) worklist algorithm. Please do not use the per-path analysis, but rather compute the aggregate results directly (10p).
- Does the worklist algorithm terminate on the program(3p)? Does it guarantee termination in general, why(3p)? Is range analysis distributive?If not, give an example (4p).