CS510 Assignment #2 (Due Feb-26 before class)

February 24, 2015

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, 6, 9, 10, 3, 4, 5, 14, 15, 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 (30p)

```
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++;
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) (20 points) Assume the input is "aadb", 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. Explain how the attacker can gain control using the vulnerability in the code and how much control the attacker gets over the control-flow of the application.

3 Forward Computation of Dynamic Slices (20p)

```
1.  int x, y , power;
2.  float z;
3.  input (x,y);
4.  if (y<0)
5.     power= -y;
6.  else
7.     power=y;
8.  z=1;
```

```
9.  while (power!=0) {
10.    z=z*x;
11.    power=power-1;
12. }
13. if (y<0)
14.    z=1/z;
15. output(z);
```

Please apply the forward computation algorithm to compute the dynamic slice for variable z for the input of $x = 42$ and $y = 4$. Show stepwise details including current slices and CDS status.

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 transfer function for analyzing individual paths (7p).
- Argue that your analysis terminates in the presence of loops. You can use example if needed.(6p)
- Extend your analysis to compute aggregate values directly. (7p)

5 Delta Debugging (20p)

A function bool *geegg(string s)* returns

- **true** if the string s contains three g characters or more or
- **true** if s contains two e characters or more and
- **false** otherwise.

For instance, *geegg("good eggs tomorrow")* returns *true*, *geegg("no eggs today")* returns *false*.

Apply the delta-debugging algorithm on the 22-character input
"delta-debugging-is-fun"
to find a 1-minimal input that still causes *geegg()* to return **true**. Record the individual inputs and test outcomes.