## **Lab: Information Flow Tracking**

(Week 3)

Yulei Sui

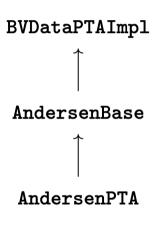
School of Computer Science and Engineering University of New South Wales, Australia

## Quiz-1 + Lab-Exercise-1 + Assignment-1

- A set of quizzes on WebCMS (5 points)
  - LLVM compiler and its intermediate representation
  - Code graphs (including ICFG and PAG)
- Lab-Exercise-1 (5 points)
  - Implement a graph traversal on a general graph
- Assignment-1 (20 points)
  - Control-flow: Implement a context-sensitive graph traversal on a CodeGraph (i.e., ICFG) and print feasible paths from a source node to a sink node on the graph
  - Data-flow: Implement Andersen's inclusion-based constraint solving for points-to analysis
  - Implement a taint checker using control-flow analysis and data-flow analysis.

## Quiz-1 + Lab-Exercise-1 + Assignment-1

- A set of guizzes on WebCMS (5 points)
  - LLVM compiler and its intermediate representation
  - Code graphs (including ICFG and PAG)
- Lab-Exercise-1 (5 points)
  - Implement a graph traversal on a general graph
- Assignment-1 (20 points)
  - Control-flow: Implement a context-sensitive graph traversal on a CodeGraph (i.e., ICFG) and print **feasible** paths from a source node to a sink node on the graph
  - Data-flow: Implement Andersen's inclusion-based constraint solving for points-to analysis
  - Implement a taint checker using control-flow analysis and data-flow analysis.
  - Specification and code template: https:
  - //github.com/SVF-tools/Software-Security-Analysis/wiki/Assignment-1 SVF APIs for control- and data-flow analysis https:
  - //github.com/SVF-tools/Software-Security-Analysis/wiki/SVF-CPP-API



 You will be working on AndersenPTA's solveWorklist method.

## BVDataPTAImpl



AndersenBase



AndersenPTA

- You will be working on AndersenPTA'S solveWorklist method.
- Constraint graph is the field consCG.

## BVDataPTAImpl



AndersenBase



AndersenPTA

- You will be working on AndersenPTA's solveWorklist method.
- Constraint graph is the field consCG.
- Address edge processing is done for you.

# **BVDataPTAImpl** AndersenBase AndersenPTA

- You will be working on AndersenPTA'S solveWorklist method.
- Constraint graph is the field consCG.
- Address edge processing is done for you.
- Note in the API there is a getDirectInEdges/getDirectOutEdges but no getCopyIn/OutEdges. This is intentional, use the Direct variant.
- You will reuse this assignment for assignment 4, make sure it is clean. :)

## **APIs for Implementing Andersen's analysis**

```
::getPts(NodeID ptr)
                                                                      //get points-to set of ptr
   SVF:: AndersenBase
                                  ::addPts(NodeID ptr, NodeID obi)
                                                                      // add obj to point-to set of object ptr
                                  ::unionPts(NodeTD ntr. NodeTD ntr)
                                                                      // union two point-to sets
                                  :: pushIntoWorklist(NodeID id)
                                                                     // push the node to worklist
                                  ::popFromWorklist()
                                                                      // pop a node from the worklist
                                  ::isInWorklist(NodeID id)
                                                                      // return true if the node in the worklist
                                  ::isWorklistEmptv()
                                                                      // return true if the worklist is empty
    SVF:: AndersenPTA
                                  ::addCopyEdge(NodeID src. NodeID dst) // add a copy edge from src to dst
                                  ::getConstraintNode(nodeId id)
                                                                    //get the node based on its id
SVF::ConstraintGraph
                                   :: dump()
                                                                    // dump the ConsG
                                                                 // get incoming store edges of the node
                                  ::getStoreInEdge()
                                                                 //get outgoing store edges of the node
                                   ::getStoreOutEdge()
SVF::ConstraintNode
                                   ::getDirectOutEdge()
                                                                 // get outgoing copy edges of the node
                                  ::getDirectInEdge()
                                                                 // get incoming copy edges of the node
```

```
https://github.com/SVF-tools/Software-Security-Analysis/wiki/SVF-CPP-API#worklist-operations
https://github.com/SVF-tools/Software-Security-Analysis/wiki/SVF-CPP-API#points-to-set-operations
https://github.com/SVF-tools/Software-Security-Analysis/wiki/SVF-CPP-API#alias-relations
https://github.com/SVF-tools/Software-Security-Analysis/wiki/SVF-CPP-API#constraintgraph-constraintnode-and-constraintedge
```

## **Assignment 1: Taint Tracker**

- Implement method readSrcSnkFromFile in Assignment-4.cpp using C++ file reading to configure sources and sinks.
- Implement method printICFGPath to collect the tainted ICFG paths and add each path (a sequence of node IDs) as a string into std::set<std::string> paths similar to Assignment 2
- Implement method aliasCheck to check aliases of the variables at source and sink.

## **Coding Task**

- Code template and specification: https://github.com/SVF-tools/ Teaching-Software-Analysis/wiki/Assignment-4
- Make sure your previous implementations in Assignment-2.cpp and Assignment-3.cpp are in place.
  - Class TaintGraphTraversal in Assignment 4 is a child class of 'ICFGTraversal'. TaintGraphTraversal will use the DFS method implemented in Assignment 2 for control-flow traversal.
  - Andersen's analysis implemented in Assignment 3 will also be used for checking aliases between two pointers.

## C++ File Reading

Implement method readSrcSnkFormFile in Assignment-4.cpp to parse the two lines from SrcSnk.txt in the form of

```
source -> { source src set getname update getchar tgetstr }
sink -> { sink mysql_query system require chmod broadcast }
```

Please refer to the following links (among many others) for C++ file reading:

- https://www.tutorialspoint.com/cplusplus/cpp\_files\_streams.htm
- https://www.cplusplus.com/doc/tutorial/files/
- https://linuxhint.com/cplusplus\_read\_write/
- https://opensource.com/article/21/3/ccc-input-output