#### **Lab: Information Flow Tracking**

(Week 3)

Yulei Sui

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

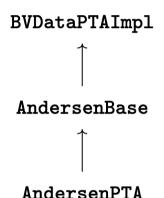
#### **Assignment-1**

- Assignment-1 (20 points)
  - readSrcSnkFromFile and reachability: Implement a context-sensitive graph traversal on a CodeGraph (i.e., ICFG) and collect feasible paths from a source node to a sink node on SVF's ICFG.
  - solveWorklist: Implement field-sensitive Andersen's inclusion-based constraint solving for points-to analysis on SVF's ConstraintGraph
  - aliasCheck: Implement taint analysis in class ICFGTraversal. Checking
    aliases of the two variables at source and sink. Two variables are aliases if their
    points-to sets have at least one overlapping element.

#### **Assignment-1**

- Assignment-1 (20 points)
  - readSrcSnkFromFile and reachability: Implement a context-sensitive graph traversal on a CodeGraph (i.e., ICFG) and collect feasible paths from a source node to a sink node on SVF's ICFG.
  - solveWorklist: Implement field-sensitive Andersen's inclusion-based constraint solving for points-to analysis on SVF's ConstraintGraph
  - aliasCheck: Implement taint analysis in class ICFGTraversal. Checking
    aliases of the two variables at source and sink. Two variables are aliases if their
    points-to sets have at least one overlapping element.
  - 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

#### **Assignment Structure**



 You will be working on AndersenPTA's solveWorklist method.

#### **Assignment Structure**

## BVDataPTAImpl



AndersenBase



AndersenPTA

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

#### **Assignment Structure**

# BVDataPTAImpl AndersenBase AndersenPTA

- You will be working on AndersenPTA'S solveWorklist method.
- Constraint graph is the field consCG.
- More APIs about points-to operations and constraint graph are here:

```
https://github.com/SVF-tools/
Software-Security-Analysis/wiki/
SVF-CPP-API
```

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

```
SVF::AndersenBase
                         :getPts(NodeID ptr)
                                                           // get points-to set of ptr
                         ::addPts(NodeID ptr. NodeID obi)
                                                          // add obj to the points-to set of pointer ptr
                         ::unionPts(NodeID ptr. NodeID ptr) // union two points-to sets
                         :pushIntoWorklist(NodeID id)
                                                          // push the node to worklist
                         ::popFromWorkist()
                                                          // pop a node from the worklist
                         ::isInWorklist(NodeID id)
                                                          // return true if the node is in the worklist
                         ::isWorklistEmpty()
                                                          // return true if the worklist is empty
SVF::AndersenPTA
                         ::addCopyEdge(NodeID src. NodeID dst)
                                                                   // add a copy edge from src to dst
                                                            // get the node based on its ID
SVF::ConstraintGraph
                        ::getConstraintNode(NodeID id)
SVF::ConstraintNode
                         ::getStoreInEdges()
                                                         // get incoming store edges of the node
                         ::getStoreOutEdges()
                                                         // get outcoming store edges of the node
                         ::getLoadInEdges()
                                                         // get incoming load edges of the node
                         ::getLoadOutEdges()
                                                         // get outcoming load edges of the node
                         ::getCopvInEdges()
                                                         // get incoming copy edges of the node
                         ::getCopvOutEdges()
                                                         // get outcoming copy edges of the node
                         ::getGepInEdges()
                                                         // get incoming gep edges of the node
                         ::getGepOutEdges()
                                                        // get outcoming gep 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
```

#### **Debugging Tips**

- MAYALIAS and NOALIAS denote the expected results (oracle) that your implementation should yield (e.g., if your results failed in a MAYALIAS case, it means that your points-to set is incomplete).
- The AndersenPTA::alias(NodeID, NodeID) method is used to evaluate whether two pointers (ConstraintNodes/SVFVars) are aliases (i.e., their points-to sets intersect). You can get the ID of a ConstraintNode/SVFVar via getId().
- Add -print-pts as an extra option for your ass1 executable when you try to print out the final points-to set of each node to validate your MAYALIAS and NOALIAS results.
- Use -print-constraint-graph to print out the final ConstraintGraph or -dump-constraint-graph to dump it into a dot file for viewing in VSCode.
- Use the toString() method in SVFVar, SVFStmt, ConstraintNode, or ICFGNode to understand the mapping from SVFIR to LLVMIR and C.

#### C++ File Reading

Implement method readSrcSnkFormFile in Assignment-1.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