**Assignment 4 (20 points), SE 421 due: Wednesday, 9/22/2021**

**Name (Last, First): Ogbondah, Chimzim**

**Submission Requirement**: Submit a PDF named HW4-lastname-firstname. Include the top two lines with your last and the first name. Include the problem statement followed by your answer.

**Pre-requisite**:

1. Review Lecture Notes 5, 6, 7
2. Review Atlas Demo Slides.
3. Read the problem statements carefully.
4. Configure Atlas Shell, please follow Atlas Shell Confuration.pdf shared with this assignment.
5. For question 7, follow the Chat App Setup Guide shared with this assignment.
6. Please read the supplement resources shared with this assignment.

**Resources:**

1. You have sample queries (in lecture notes and Atlas demo slides) that you can modify to create the new queries required for answers.
2. The Java doc for the C commons queries is also provided to you.

**Use of Atlas**: Using Atlas map XINU and create the universal graph. It is required for all except the last problem. For the last problem, use the Java code given as zip file. In case you want to use Atlas for the last problem, you will first create a project and map the Java code in Atlas.

**Problem 1 (2 point):** Report the results for the following queries.

1. universe.eval.nodes.size
   1. 29308
2. universe.eval.edges.size
   1. 106232
3. functions("\*").eval.nodes.size
   1. 276

**Problem 2 (2 points):** Report and correct the error for the following queries.

var x = cfg(functions("dskqopt")).eval.nodes

show(x)

The number of total nodes in the function dskqopt is 46 and correcting the error to get this value entails changing show(x) to x.size or by using the command **var x = cfg(function(“dskqopt”)).nodes(XCSG.Node).eval.nodes.size**

**Problem 3 (3 points):** Report the results for the following queries.

1. functions("ds\*").induce(edges(XCSG.Call)).eval.nodes.size
   1. **12**
2. functions("ds\*").induce(edges(XCSG.Call)).eval.edges.size
   1. 7
3. show(functions("ds\*").induce(edges(XCSG.Call))) – *Report in one sentence your understanding of what is shown. Do not include the graph*
   1. It is showing all the edge calls from all the classes that start with ds

**Problem 4 (3 points):**

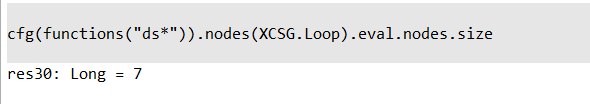
1. show(cfg(functions("dskqopt")).nodes(XCSG.Loop)) Report in one sentence what is shown.

*Do not include the graph*

**It is showing all the loops inside of the dskqopt class**

1. Manually count and *report* the total number of loops in all the functions that start with “ds.”
   1. 7
2. Write a query to count the total number of loops in functions that start with “ds”

*Include a screen shot to show the execution of your query and its result*.



**Problem 5 (5 points):**

**Alternative1**

var dskenqCfg = cfg(functions("dskenq"))

var dskenqCallSites = dskenqCfg.nodes(XCSG.SimpleCallSite)

show(dskenqCallSites)

**Alternative2**

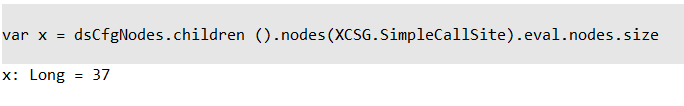
var dskenq = functions("dskenq")

var dskenqCfgNodes = dskenq.children().nodes(XCSG.ControlFlow\_Node)

var x = dskenqCfgNodes.children ().nodes(XCSG.SimpleCallSite)

show(x)

1. Which of the above alternatives can correctly count the number of *call sites*?
   1. Alternative 2
2. Manually count and *report* the total number of in *call sites* in functions that start with “ds” (e.g. dswrite)
   1. 37
3. Give a query sequence to count the number of *call sites* in functions that start with “ds”
   1. Var ds = functions(ds\*)
   2. Var ds = dsCfgNodes = ds.children().nodes(XCSG.ControlFlow\_Node)



**Problem 6:** Import ChatClient and ChatServer projects contained in chatapp.zip and map the ChatClient project*.* Analyze only the ChatClient project to find the confidentiality breach vulnerability. You have multiple options: run the client, read the code, use Atlas to understand the code.

**Effort (4 points)**: Describe your threat modeling including the leaks and triggers you considered. Were you able to confirm a threat model? If not, why? Write a short paragraph about the aspects of program you examined. (*For this part, you will be given credit based on the quality of your effort and how well you have described*)

* + - Since the program doesn’t check to ensure that each username is unique in the event that someone wants a transaction history of their messages, the messages could be sent two anyone else who potentially has the same username as you. This breaches confidentiality since messages and usernames are being breached.

**Correct detection of the vulnerability (1 point)**