**CSE 5382-001: Secure Programming**
**Fall 2018, © TJ, UTA 2018**

**Programming Assignment 1**
**Due: September 14, 2018, 23:59 UTA time**
**Understanding and Using Static Code Analysis Tools**

The goal of this assignment is to familiarize you with using open source static code analysis tools to find bugs and vulnerabilities in source code and to recognize that not all tools are created equal. Each tool uses different analysis techniques and is programmed with different rules for detection of problems. Also, the settings, and in some cases the plugins, used in a tool will have an effect on what types of problems it can identify.

This will be an individual assignment (no teams).

Part 1 (85 Points Max):
For the first part of this assignment, you will analyze a simple web server written in Java that is posted on Blackboard along with this document. The analysis will be performed with at least two tools, one of which must be FindBugs v3.0.0 (or newer) with the FindBugs Security Plugin v1.4.6 (or newer). The second tool can be a tool of your choosing (e.g. PMD v5.5.2, JLint 3.0, SonarLint 2.1, etc.). Some tools, such as FindBugs, analyze the binary (Java bytecode) while others analyze source directly (the documentation for the tools should provide direction as to whether the tool analyzes source or binary). As a result, you will be required to build the source for the simple web server in order to analyze it with FindBugs.

The following web site is a good starting point for locating static analysis tools for various languages: https://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html.

Prior to using a tool to analyze the simple web server, manually review the code and see if you can identify any problems. Document any problems you find. Explain where in the code the problem lies (file and line number) and why you think it's a problem. Don't worry if you don't catch very many, since we have not yet covered discussion of various vulnerabilities and how to find them. Consider this a baseline assessment of your existing code review skills and ability to spot security problems. As you look at the code, think about ways an attacker can exploit weaknesses in the web server.

Run the analysis across the entire code base (in this case, a single Java class). Each tool typically has a variety of ways it can be invoked (command line, from scripts, integrated into build systems such as Ant or Maven, from integrated development environments such as Eclipse, etc.). The method of invocation you use is up to you, but you should be able to understand the requirements for the tool, the types of files on which it operates, and what it produces. **Make sure you turn on scanning for security-related vulnerabilities and/or enable security rule sets as appropriate in the chosen tool.** Document any tool findings and indicate which ones you found in your manual analysis.

Experiment with changing the sensitivity levels within each of the tools (this setting might be called one of many things and has to do with how aggressive the tool is in performing an analysis). Notice the increase/decrease in the number of findings reported. Also, experiment with turning on/off

various rule sets in each tool. Document your observations in the report. NOTE: Based on the small code sample being used, you may or may not notice a difference in the findings.

The graduate teaching assistant should be available to provide assistance as needed in setting up your environment and building the simple web server if you are struggling to get it working.

Part 2 (15 Points Max):
For the second part of this assignment, pick some code you've written in the past (possibly homework for another class). It can be in any language (as long as you can find a capable static code analysis tool for it). C/C++ or Java would be preferable. Run an analysis tool on it to see what kind of findings you get and try fixing few of the bugs. The goal here is to make you aware of vulnerability escapes in your own prior work which may lead to malicious exploitation. It will help you uncover mistakes so that you can adjust programming practices to avoid them in the future.

Submission:
You will submit a report in PDF or MS Word that documents the following:

- Results of your manual review of the simple web server (conducted prior to running the tools).
- Your tool choices and versions.
- Your approach to invoking the tools. Include screen shots and steps.
- Your understanding of how the tools work (compare and contrast each tool) as well as differences in types of flaws reported. This will come from reviews of the documentation for each tool.
- Include attachments of tool output reports, fixes (for your own code), preferably in HTML, PDF, Excel, or ASCII text (XML as a last resort).