

REU Supplemental Funding Request: An Information Flow Analyzer for Java

Context. This proposal is for an undergraduate summer research project on secure information flow in Java programs. The work will support ongoing research of David Naumann and Anindya Banerjee in our CyberTrust project entitled “CT-ISG Collaborative Research: Access Control and Downgrading in Information Flow Assurance.” This REU request is under Naumann’s award, CNS-0627338, and Naumann will supervise the student.

The overall goal is to develop tools and methods for high assurance of secure information flow in systems implemented using object-oriented languages like Java and C#, focusing on policies that involve controlled downgrading and are linked to access policies.

In previous work we developed an approach to confidentiality and integrity policy that takes into account code based access control (stack inspection) which is used in the Java and .NET runtime systems. We devised a type-based static analysis and showed that it enforces the policy as formalized by a noninterference condition [2, 3, 6]. Our PhD student Qi Sun designed an inference algorithm so that users only need to specify information flow policy at external interfaces [10]. He implemented a prototype, called **SecJ**, as part of his recently completed dissertation [9]. We have also investigated how to extract interface specifications from Java access controls [8].

Working with another PhD student, we recently found an expressive and robust way to specify sophisticated information flow policies in which specified functions of a secret may be downgraded under specified conditions including but not limited to access control state [1, 4]. Enforcement of our Conditioned Gradual Release policies is by a modular combination of type-based static analysis with program verification.

The **SecJ** prototype was used to conduct experiments with security type inference including adaptations of some of the case studies developed using **Jif**.¹ That system, like **SecJ**, provides a security-typed language based on Java, but the two are at quite different points in the design space. **Jif** has been under development for almost a decade and now has a stable distribution that is being used by a number of research groups and in advanced security courses. The **SecJ** tool has been distributed to researchers outside Naumann’s and Banerjee’s institutions, on an ad hoc basis, but is not yet supported by a web site.

Objectives. Funding is requested to support an undergraduate student who will continue development of the **SecJ** tool. The first objective for this summer is to add a graphical user interface (GUI) to provide easy use of the activities supported by the tool: policy specification, modular inference for Java packages, and developer-guided refinement of inferred library APIs to support extensibility of security-critical applications. Moreover, the distribution will be polished and made available on a web site, possibly allowing remote invocation of the tool.

The second objective for this summer is to connect **SecJ** with the **ESC/Java2** verification system,² to support Conditioned Gradual Release (CGR) policies. The **SecJ** prototype already has a rudimentary declassification mechanism that simply creates a loop-hole in security type checking. To this hook, policies and enforcement need to be attached. It is premature to include this feature as part of the distribution. For this summer, the goal is to

¹<http://www.cs.cornell.edu/jif/>

²<http://jmleclipse.projects.cis.ksu.edu/docs/esc-java.shtml>

have a prototype for our own use in experiments that will guide the subsequent development of both theory and implementation for CGR.

The third objective for this summer is to train a student to do more advanced research in the future. Naumann is working on the self-composition technique [7] which is needed for practical CGR enforcement using off-the-shelf verification tools. (This also ties with Naumann's work on the JML specification language under a different NSF award, CNS-0708330.) He has previously worked with an undergraduate, Dustin Long, conducting experiments with this technique, but it requires substantial background.

Project plans. During the first month of summer, the supported student will implement a GUI for `SecJ` as a standalone Java application. During the second month he will develop the web site for the tool and get feedback from other students to improve the design of the GUI and the library of sample programs and policies. During the last two weeks, if all goes well, he will focus on scripting a loosely coupled connection between `SecJ` and `ESC/Java`. (If not, he will still be busy with the GUI.) At the same time, he will polish the GUI and web site based on feedback from other students in our Lab for Secure Systems.

Qi Sun, original developer of `SecJ`, is now at Google but has already shown his commitment to supporting the tool and helping with further development. His thesis provides adequate design and implementation documentation for our purposes. One of my current students, Stan Rosenberg, is familiar with `SecJ` and `ESC/Java`. He is continuing to work on declassification and will be available during the summer to help the undergraduate in this project.

Previous experience. I supervised at least one or two undergrads every summer for the six years for various projects, funded by Stevens Institute and REU supplements. (Last summer I took a break from this.) The CodeBlue system originated as an award-winning senior design project supervised by Naumann; the undergraduates involved are coauthors on a Globecom paper [5]. In the subsequent three summers, Prof. Susanne Wetzels and I jointly supervised undergraduates porting the system from C++ to Java, extending its functionality, and improving its performance. The first of these students, Jared Cordasco, is now in the PhD program.

During 2005–2006, undergraduate Dustin Long worked with me on the self-composition technique, as research assistant funded by Telcordia Technologies and NSF Research Experiences for Undergraduates (REU). This led to his B.S. degree with thesis: *A Multi-tiered Security Methodology: an Investigation into Lightweight Java Access Controls and Verifying Secure Information Flow*.

Project management and deliverables. The student will develop Java code as well as web content and scripts. A final report will be required as well as weekly status reports. The student will meet with me 3–4 hours each week during the first month. During the remaining six weeks I will travel intermittently. Prof. Wetzels has agreed to meet the student when I am away so there is face-to-face contact every week. I will always be in contact by email and the student will be interacting with Qi Sun and Stan Rosenberg.

The student will be required to be on campus during the 35 hour work week. Meetings will take place in the Lab for Secure Systems and the student can work in the department lab.

Selection process and criteria. I will publicize the position on my web page and the CS undergraduate mailing list. I will interview students individually to assess their programming skill and familiarity with Java and Linux. Just as important is the student's demonstrated ability to carry out sustained and productive activity on their own initiative. For this reason I will probably choose a student who has distinguished herself in one of my courses. For this project, I prefer a student at the sophomore level, since one objective is training for more advanced work the following summer.

One student has already been identified as candidate for the position. For background, he is currently developing small applications in Jif.

References

- [1] A. Banerjee, D. Naumann, and S. Rosenberg. Towards a logical account of declassification. In *ACM Workshop on Programming Languages and Analysis for Security*, pages 61–66, 2007.
- [2] A. Banerjee and D. A. Naumann. Using access control for secure information flow in a Java-like language. In *Computer Security Foundations Workshop (CSFW)*, pages 155–169. IEEE Computer Society Press, 2003.
- [3] A. Banerjee and D. A. Naumann. Stack-based access control for secure information flow. *Journal of Functional Programming*, 15(2):131–177, 2005. Special issue on Language Based Security.
- [4] A. Banerjee, D. A. Naumann, and S. Rosenberg. Expressive declassification policies and modular static enforcement. In *29th IEEE Symposium on Security and Privacy*, 2008. To appear.
- [5] D. Hromin, M. Chladil, N. Vanatta, D. Naumann, S. Wetzel, F. Anjum, and R. Jain. CodeBlue: a Bluetooth interactive dance club system. In *IEEE Globecom*, 2003.
- [6] D. A. Naumann. Verifying a secure information flow analyzer. In J. Hurd and T. Melham, editors, *18th International Conference on Theorem Proving in Higher Order Logics TPHOLS*, volume 3603 of *Lecture Notes in Computer Science*, pages 211–226. Springer, 2005.
- [7] D. A. Naumann. From coupling relations to mated invariants for secure information flow. In *European Symposium on Research in Computer Security (ESORICS)*, volume 4189 of *LNCS*, pages 279–296, 2006.
- [8] M. Pistoia, A. Banerjee, and D. A. Naumann. Beyond stack inspection: A unified access-control and information-flow security model. In *28th IEEE Symposium on Security and Privacy*, pages 149–163, 2007.
- [9] Q. Sun. Constraint-based secure information flow inference for object-oriented programs. Technical Report CS-TR-2007-6, Stevens Institute of Technology, 2007.
- [10] Q. Sun, A. Banerjee, and D. A. Naumann. Modular and constraint-based information flow inference for an object-oriented language. In R. Giacobazzi, editor, *Static Analysis Symposium (SAS)*, volume 3148 of *LNCS*, pages 84–99. Springer-Verlag, 2004.