FindBugs Lab

Adapted from [Software Security, Appendix A, by Gary McGraw, Addison-Wesley, 2006]

Introducing FindBugs and Static Analysis

FindBugs is a static analysis tool for Java that can find security vulnerabilities and other bugs. It is a free tool, and it is also part of an industrial security tool called Fortify:
http://www8.hp.com/us/en/software-solutions/static-code-analysis-sast/index.html

Watch this Google Tech Talk by Prof. Bill Pugh (University of Maryland, College Park), one of the creators of FindBugs:
https://www.youtube.com/watch?v=GgK20Yv9QRk
The talk lasts about one hour.

Auditing Source Code Manually

Any kind of manual source code review requires patience, an eye for detail, and extensive knowledge about the types of problems that can cause a program to fail. A security audit is no different, but instead of thinking "what could go wrong?" the auditor must consider "what could an attacker *force* to go wrong?" The auditor's role is to pare down this infinite search space and identify the most dangerous problems and weaknesses in a program. Experience is invaluable but can only be gained through practice.

1. Download gradesystem.zip from CMS. This is part of the source code of one of the projects from this course in Fall 2011. Please do not redistribute this code.
2. Examine the application source code. Do not look for specific issues yet; just become familiar with the application by asking yourself the same types of questions that an auditor starting a code audit would ask, such as:
    - How large is the application?
    - What is the basic design of the application?
    - Who are the likely attackers?
    - What would an attacker hope to achieve?
    - How are the developers trying to protect the application?
    - What sorts of techniques might an attacker use to subvert the application?
3. Examine the source code of Client_Second.java and ConnectionClass.java.
    - How and when is this code executed?
    - Can an attacker exploit any vulnerabilities in this code?

Questions for Thought

1. How many vulnerabilities can you consciously look for as you manually audit code: 5, 10, 100, 1000?
2. How often should a security audit be performed? If you performed an audit today and fixed the problems, what would your confidence be in the code 7, 30, or 180 days later? How much new code would developers write in 7, 30, or 180 days?
3. If you had to set up a process for manually auditing code in your company, how would you estimate the amount of effort and time required to do it effectively?
4. If you cannot audit all of the code, how should you choose which section of code to audit? How confident would you be in the results of the audit?
5. Identify five programming styles or techniques that make auditing easier or harder.

Auditing Source Code Automatically

1. Download and unzip the latest version of FindBugs from http://findbugs.sourceforge.net/downloads.html.
2. Launch the FindBugs GUI by running FBROOT/bin/findbugs, where FBROOT is the directory in which you unzipped the file in step 1.
3. In the FindBugs GUI, choose File->New Project. Give the project a name of your choice. Add to the "Classpath for Analysis" the gradesystem.zip file you downloaded in the previous part of the lab. Add to "Source Directories" the same gradesystem.zip file. Run the analysis.
4. How many security bugs are identified by FindBugs in this system? How many other bugs are identified?
5. Use the View menu to view only Scary bugs. Then only the Scariest bugs. Then All Bug Ranks. How many Scary (or worse) bugs are there?
6. Experiment with viewing bugs in different ways using the "Group Bugs By" interface. Drag-and-drop the different bug attributes ("Bug Kind", "Category", etc.) to see how bugs can be sorted in various ways.
7. Examine the bug "read of unwritten field logDisplay in Client_Second.connectClient".
    - What is the potential problem that results from this bug?
    - Did the analysis miss an initialization of this field? Or did the programmers fail to initialize it? How should this bug be fixed?
8. Examine any of the security bugs identified by the analysis in ConnectionClass.java.
    - What is the potential problem that results from this bug? (Hint: read the Wikipedia page on SQL injection, if you don't already know what that is.)
    - Do you know how to fix this bug?
9. [Optional] Use the FindBugs Cloud reviewing system to make notes on the severity of bugs and how to fix them.
10. Use File->Save As to save your analysis/review/classification results as HTML. Open the HTML file in a browser to see the FindBugs report on this system.

Questions for Thought

1. Can you think of (or write) a line of code that would be acceptable in one program but would cause a serious security problem in another program?
2. What makes one security issue more important than another? How do you determine the importance of a security issue?

3. Would you have found all of the bugs that FindBugs identified, if you had to use manual inspection?
4. How many of those do you think are really bugs?
5. How many of those bugs do you think are really security vulnerabilities?
6. How do you know they aren't *all* security vulnerabilities?

Further Exploration

There are FindBugs plugins for many major IDEs (Eclipse, NetBeans, etc.). Install one of those plugins and activate automatic analysis of your source code. Note how FindBugs now becomes part of the development process: it identifies problems with your code just like the Java compiler does.

Fair Warning

Your alpha, beta, and final milestones must all be accompanied with a FindBugs scan in which you have reviewed and classified all bugs deemed Scary or Scariest. Integrate FindBugs into your development process from the very beginning!