Tutorial: Secure Coding Practices, Automated Assessment Tools and the SWAMP

Barton P. Miller
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706
bart@cs.wisc.edu

Elisa Heymann
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706
elisa@cs.wisc.edu

I. Introduction

Security is crucial to the software that we develop and use. With the incredible growth of both Web and Cloud services, security is becoming even more critical.

Securing your network is not enough. Every service that you deploy is a window into your data center from the outside world, and a window that could be exploited by an attacker.

This tutorial is relevant to anyone wanting to learn about minimizing security flaws in the software they develop or manage. We share our experiences gained from performing vulnerability assessments of critical services and middleware. You will learn important skills for software developers and analysts concerned with security.

Software assurance tools – tools that scan the source or binary code of a program to find weaknesses – are the first line of defense in assessing the security of a software project. These tools can catch flaws in a program that affect both the correctness and safety of the code. The SWAMP is a open and free facility to provide access to a large collection of tools for a variety of languages and environments. This tutorial is also relevant to anyone wanting to learn how to use these automated assessment tools to minimize security flaws in the software they develop or manage.

II. Technical Topics

Our tutorial focuses on the programming practices that can lead to security vulnerabilities, and on automated tools for finding security weaknesses. This tutorial features several interactive secure coding quizzes where the audience will be challenged to find as many vulnerabilities as they can in short code fragments, based on vulnerabilities that we found in real world software. What the audience finds (and does not find) will then be discussed. These quizzes will motivate the following section of the tutorial.

We will start the tutorial with some basic vocabulary presenting terms that will help the student focus the key concepts being taught. This will include such terms as attack surface, impact surface, vulnerability, exploit, and mitigation.

The first major technical area of our tutorial is a presentation of the most common vulnerabilities found in middleware and services. Descriptions of each type of vulnerability will be presented with examples. The examples will show how each type of vulnerability occurs within code, pointing out how common usage patterns for system library routines, kernel calls, and common programming techniques can result in the vulnerability. The coding examples are presented in C, C++, Java, Python and Perl.

Along with the description of the vulnerabilities, we will show how the vulnerability can be mitigated or eliminated through the use of specific programming and design techniques. An important part of our discussion of each vulnerable technique is a description of the thought processes used by the attacker in developing an exploit.

The second technical area of our tutorial is a presentation about automated assessment tools. We will introduce the different types of analysis tools, how these tools work, their output and their limitations. We then talk about control flow analysis and data flow analysis, as they are the tools' core to answer if certain code is safe or not.

The next section of the tutorial explains how to use different commercial and open source tools for C/C++ and Java, and how to process the tools' output. For that we use simple test applications extracted from the NIST/NSA Juliet test suite, where each of these applications contain specific weaknesses, and the version of the same code with the weakness fixed. The weaknesses we address are drawn from a collection of the most commonly occurring ones in real code, such as Relative Path Traversal, OS Command Injection, Cross-Site Scripting (XSS), Improper Neutralization of Script in an Error Message Web Page, Integer Overflow, Sensitive Information Uncleared Before Release, Uncaught Exception, and Use of Hard-coded Password.

The last section of this tutorial shows how users can benefit from the Software Assurance Marketplace (SWAMP) (https://continuousassurance.org). SWAMP users can access both commercial and open source software assessment tools. This section will have a hands-on exercise (see section below).

III. Target Audience

This tutorial is targeted at developers wishing to minimize the security flaws in the software that they develop. It covers the defensive side of security – how to prevent problems by showing many types of vulnerabilities that occur in real code and what techniques can be used to prevent them, and how to use automated analysis tools to detect flaws in their software. The target audience for this tutorial is anyone involved with the development, deployment, assessment, or management of critical software.

IV. Hands-on Exercise

The tutorial will include an opportunity for the attendees to experiment with software assurance tools available in the DHS-funded Software Assurance Marketplace (SWAMP), an open and free facility.

The student will progress through a structured exercise:

1. Signing up for the SWAMP or accessing it via them github, Google, or InCommon credentials.
2. Uploading software to the SWAMP.
3. Running a variety of software assurance tools.
4. Viewing and interpreting the results.
5. Fixing problems found and iterating over the above steps.

The exercise will be packed in a VirtualBox image, which will be available to attendees before the tutorial session (and available on the web and memory sticks at the tutorial). The VirtualBox image will be pre-configured and ready to run with example code and step-by-step instructions.

Note that these exercises are newly developed to be streamlined for time constraints of a tutorial session.

V. Learning Outcomes

The goals for this tutorial are to teach software developers and designers to:

- Visualize code and software design from a security perspective.
- Learn specific techniques for writing secure code.
- Learn how software assurance tools can to help improve the security of their code.
- Learn about specific tools resources available to them and get initial experience using these resources.

Audience prerequisites: To gain maximum benefit from this tutorial, attendees should be familiar with the process of developing software and at least one of the C, C++ Java or scripting programming languages. This tutorial does not assume any prior knowledge of security assessment or vulnerabilities.

VI. Previous Tutorials

Miller and Heymann teach a wide variety of tutorials on software vulnerability assessment, secure programming, and software assurance tools. They are constantly updating and expanding their tutorial content.

Tutorials taught in the past year include:

- “Secure Coding Practices and Automated Assessment Tools”. Half day. Technical University of Munich, Germany. March 2018.
- “Secure Coding Practices and Automated Assessment Tools”. 3 day. Hands-on. Total Soft Bank (TSB), Busan, South Korea. November 2017.
- “Secure Coding Practices”. URISC@SC17. Denver, CO, November 2017. (1 hour long)
- “Secure Coding Practices and Automated Assessment Tools”. Half day. O'Reilly Security Conference, New York, October, 2017.
- “Automated Assessment Tools: Theory and Practice”. Half day. NFS Cybersecurity summit. Arlington, VA, August 2017. Hands-on.
- “Secure Coding Practices & Automated Assessment Tools”. 3 days. FAA. New Jersey, June 2017. Hands-on: Java (WebGoat) and C (filetool) with the SWAMP.
- “Secure Coding Practices & Automated Assessment Tools”. Half day. OSCON: O'Reilly Open Source Convention, Austin, TX, May 2017.
- “Secure Coding Practices and Automated Assessment Tools”. Two days. Card Access Engineering, Utah, April 2017.
- “Secure Coding Practices and Automated Assessment Tools”. Half day. Universidad de la República. Montevideo, Uruguay, March 2017.

In additional, Miller and Heymann are developing an online curriculum cover these topics. The prototype, under-development website can be seen at: http://research.cs.wisc.edu/mist/SoftwareSecurityCourse/

Barton Miller is the Vilas Distinguished Achievement and the Amar & Belinder Sohi Professor of Computer Science at the University of Wisconsin-Madison. He is Chief Scientist for the DHS Software Assurance Marketplace research facility and is Software Assurance Lead on the NSF Cybersecurity Center of Excellence. In addition, he co-directs the MIST software vulnerability assessment project in collaboration with his colleagues at the Autonomous University of Barcelona. He also leads the Paradyn Parallel Performance Tool project, which is investigating performance and instrumentation technologies for parallel and distributed applications and systems. His research interests include systems security, binary and malicious code analysis and instrumentation extreme scale systems, parallel and distributed program measurement and debugging, and mobile computing. Miller's research is supported by the U.S. Dept. of Homeland Security, U.S. Dept. of Energy, National Science Foundation, NATO, and various corporations.

In 1988, Miller founded the field of Fuzz random software testing, which is the foundation of many security and software engineering disciplines. In 1992, Miller (working with his then-student, Prof. Jeffrey Hollingsworth), founded the field of dynamic binary code instrumentation and coined the term "dynamic instrumentation". Dynamic instrumentation forms the basis for his current efforts in malware analysis and instrumentation.

Elisa Heymann is a Senior Scientist on the NSF Cybersecurity Center of Excellence at the University of Wisconsin-Madison, and an Associate Professor at the Autonomous University of Barcelona. She co-directs the MIST software vulnerability assessment at the Autonomous University of Barcelona, Spain.

She was also in charge of the Grid/Cloud security group at the UAB, and participated in two major Grid European Projects: EGI-InSPIRE and European Middleware Initiative (EMI). Heymann's research interests include security and resource management for Grid and Cloud environments. Her research is supported by the NSF, Spanish government, the European Commission, and NATO.