Tutorial: Principles and Practices of Secure Crypto Coding in Java*

Sazzadur Rahaman  Na Meng  Danfeng (Daphne) Yao
Department of Computer Science
Virginia Tech
\{sazzad14, nm8247, danfeng\}@vt.edu

Abstract—Various software libraries and frameworks provide a variety of APIs to support secure coding. However, misusing these APIs can cost developers tremendous time and effort, introduce security vulnerabilities to software, and cause serious consequences like data leakage or Denial of Service (DoS) on servers. No prior tutorial educates people on the best practice of secure coding, the pitfalls that should be avoided, and the remediation of insecure code.

To increase the security awareness of developers and improve the quality of their software products, we propose a 90-minute tutorial to teach participants the principles and practices of Java secure coding. In this tutorial, we will introduce the principles of using security APIs, analyze typical API misuse cases to explain the causes and effects. We will also introduce a tool that we recently developed to automatically detect API misuse in Java.

There are five parts in our tutorial. To reveal the existing status of secure coding practice, we will first introduce the findings in our recent study on StackOverflow posts relevant to Java security. Second, we will discuss the recommended principles of API usage by security experts. Third, to correlate the principles with existing practice, we will also discuss some API misuse examples for the hash digest, message encryption and decryption, key generation, and SSL/TLS connection. Fourth, we will ask participants to examine extra code examples and discuss the security property of each example. Finally, We will give an overview of the available tools and resources, demonstrate a tool named RigorITYJ that we recently developed to automatically detect API misuse in Java. We will also help participants install and use RigorITYJ on their own machines and ask them for trials.

By actively involving participants in code discussion and tool trial, we aim to raise the security awareness among developers, improve their secure coding capabilities, and equip them with the tools they need for secure coding.

1. Introduction

Various software libraries or frameworks (e.g., BouncyCastle [6]) provide a variety of features to enable secure coding. For instance, Java platform defines the Java Cryptography Architecture (JCA), which contains APIs for hashes, keys, certificates, digital signatures, and encryption [10]. Java Secure Socket Extension (JSSE) includes APIs for standard secure communication protocols like SSL/TLS [1]. Misusing these libraries and frameworks not only slows development time, but also leads to security vulnerabilities in the resulting software [4], [15].

Our recent study on StackOverflow posts revealed a worrisome reality in the software development industry [11]. Based on the discussions among developers on StackOverflow, we found that a substantial number of developers did not appear to understand the concepts or implications of security API usage. Such lack of domain knowledge creates frustration in developers, who sometimes ended up using APIs in easy but completely insecure ways. Additionally, we collected substantial empirical evidence showing that (1) the security API usage is over complicated and poorly documented; (2) the error reporting systems of Java platform security APIs cause confusion; and (3) the understanding on multi-language support for secure coding is rather weak. This indicates a strong need of better secure coding education for developers, to raise their security awareness when designing, implementing, and using security APIs.

Till date, the number of studies offering automatic verification for secure use of cryptography is fairly limited (e.g., CryptoLint for Android [8], FixDroid for Android IDE [12]. RigorITYJ [13] and CogniCrypt [17] for Java, TaintCrypt for C/C++ [14], etc.). Moreover, there is no prior tutorial session focusing secure cryptographic coding practice in Java. Existing online tutorials and documents introduce some best practices of using security APIs (e.g., [1], [2], [3]). Nevertheless, these documents do not include many code examples to thoroughly demonstrate the correct API usage, neither do they discuss or explain developers’ frequent mistakes when using the APIs. Online forums like StackOverflow [16] contain sufficient code snippets to demonstrate and recommend security API usage, but some of the recommended usage is incorrect or insecure [5].

To create mass awareness, we propose a 90-minute tutorial on the principles and practices of secure coding with proper usage of cryptographic APIs. In this tutorial, we will include hands-on components and audience interactions to ensure the quality of participants’ learning experience. Specifically, there are five parts in the tutorial.

Part I: Review of Existing Secure Coding Practices. To emphasize the importance of secure coding practice,
2. Prerequisites and Target Audience

To ensure the quality of hands-on experience, we expect people to have the basic knowledge of Java programming. Participants are highly recommended to bring their laptops and try out our research tool.

Security researchers, software developers, students, and other software professionals at all levels are welcome to attend the tutorial. For the participants that have little domain knowledge of security, our tutorial will cover a set of interesting information about the principles of secure coding, the existing status of secure coding practice, the gap between the principles and practices, and the potential solutions to narrow the gap (e.g., developer education and tool support). For security experts and security library builders, this tutorial will be also valuable, because it provides a platform for people to share their expertise and learn about the needs of better software support for secure coding practice.

References