Poster: Scientific Comparison on Accuracy and Scalability of Cryptographic API Misuse Detection

Sharmin Afrose¹, Ya Xiao¹, Sazzadur Rahaman², Barton P. Miller³, Danfeng (Daphne) Yao¹

¹Computer Science, Virginia Tech, Blacksburg, VA
²Computer Science, University of Arizona, Tucson, AZ
³Computer Science, University of Wisconsin-Madison, Madison, WI
sharminafrose@vt.edu, yax99@vt.edu, sazz@cs.arizona.edu, bart@cs.wisc.edu, danfeng@vt.edu

Abstract—Studies showed that misuses of cryptographic APIs are common in real-world code. To detect misuses, several open-sourced and commercial tools are developed for screening Java programs. We develop two comprehensive benchmarks (CryptoAPI-Bench, ApacheCryptoAPI-Bench) that enable the first scientific in-depth comparison on accuracy and scalability of misuse detection. CryptoAPI-Bench contains 181 unit test cases covering basic cases, as well as complex cases (i.e., interprocedural, field sensitive, multiple class test cases, and path sensitive cases). The ApacheCryptoAPI-Bench consists of 121 cryptographic cases from 10 Apache projects. Both benchmarks include correct test cases for testing false-positive. We evaluate four tools, namely, SpotBugs, CryptoGuard, CrySL, and another tool (anonymous) using both benchmarks. We present their performance and comparative analysis. The ApacheCryptoAPI-Bench also examines the scalability of the tools. Our benchmarks are beneficial for advancing state-of-the-art solutions in the space of misuse detection.

Index Terms—Cryptographic APIs, Benchmarks, Java

I. INTRODUCTION

Various studies have shown that a vast majority of Java and Android applications misuse cryptographic libraries and APIs, causing devastating security and privacy implications [1]–[5]. The prominent causes for cryptographic misuses are the deficiency in understanding of security API usage [2], [6], complex API designs [6], [7], the lack of cybersecurity training [2], insecure code generation tools and insecure/misleading suggestions in Stack Overflow [2], [8]. The reality is that most developers, with tight project deadlines and short product turnaround time, spend little effort on improving their knowledge or hardening their code for long-term benefits [9]. Recognizing these practical barriers, automatic cryptographic code generation, and misuse detection tools [3] play a significant role in assisting developers with writing and maintaining secure code. The security community has produced several impressive static (e.g., CryptoLint [1], CrySL [10], FixDroid [11], MalloDroid, CryptoGuard [3], Parfait [12]) and dynamic code screening tools (e.g., Crylogger, SMV-Hunter, and AndroSSL) to detect API misuses in Java. The static analysis does not require a program to execute, rather it is performed on a version of the code (e.g., source code, intermediate representations or binary). Many abstract security rules are reducible to concrete program properties that are enforceable via generic static analysis techniques [3].

To advance and monitor the scientific progress in generating effective tools, a mechanism for comparative studies is required. Unfortunately, for detection of cryptographic API misuses, no suitable mechanism or benchmark exists. Such a benchmark needs to have several requirements: cover a wide range of misuse instances, cover interesting program properties (e.g., flow-, context-, field-, path-sensitivity, etc.), written in easily compilable source codes so that both source code and binary code analysis tools can be easily evaluated.

None of the existing benchmarks follows these criteria (e.g., DroidBench, Ghera). For example, DroidBench only contains binaries. Ghera has sources of provided Android apps. However, both DroidBench and Ghera barely cover cryptographic API misuses.

Our contribution is to develop two benchmarks [13], [14]. First, We develop a benchmark named CryptoAPI-Bench that utilizes various interesting program properties (e.g., field-, context-, and path-sensitivity) to produce a diverse set of test cases. We also provide another benchmark named ApacheCryptoAPI-Bench for checking the scalability property of the cryptographic vulnerability detection tools. We run CryptoAPI-Bench and ApacheCryptoAPI-Bench on four static analysis tools (i.e., SpotBugs [15], CryptoGuard, CrySL, and Tool A (anonymous) and perform a comparative analysis of these tools.

II. CRYPTO MISUSE CATEGORIES

In this section, we discuss 5 groups of 18 Java cryptographic API misuse categories. We got insights into these misuse categories from previous literature, NIST documents, and other blogs. We describe reasons for vulnerability and possible security solutions for these misuse categories. The considered misuse groups (categories) are: Predictable secrets (cryptographic key, password in PBE, password in KeyStore, credentials in string), vulnerability in SSL/TLS (hostname verifier, certificate validation, SSL socket, HTTP protocol), predictable PRNGs (predictable random number generator, seed in PRNG), vulnerable parameters (salt in PBE, mode of operation, initialization vector, iteration in PBE), vulnerable algorithms (symmetric ciphers, asymmetric ciphers, cryptographic hash, cryptographic MAC)
TABLE I

<table>
<thead>
<tr>
<th>Tools</th>
<th>CryptoAPI-Bench</th>
<th>ApacheCryptoAPI-Bench</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpotBugs</td>
<td>81.25</td>
<td>92.86</td>
</tr>
<tr>
<td>CryptoGuard</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>CrySL</td>
<td>58.82</td>
<td>71.43</td>
</tr>
<tr>
<td>Tool A*</td>
<td>100.00</td>
<td>92.86</td>
</tr>
</tbody>
</table>

III. DESIGN OF BENCHMARKS

CryptoAPI-Bench: In this section, we present the design of the CryptoAPI-Bench. We manually generate 181 unit test cases guided by 18 types of misuses. We divide all test cases into two types, i.e., basic cases and advanced cases that incorporate possible variations in the perspective of program analysis to detect cryptographic vulnerability.

ApacheCryptoAPI-Bench: We include the early version of real-world large 10 Apache projects to check the scalability property of different tools. The spark project is the largest among 10 considered projects containing 311,856 lines of code. Deltaspike contains the lowest number of lines of code (i.e., 5,116 LoC).

We enlist 121 test cases in ApacheCryptoAPI-Bench. Among them, 79 are basic cases, and 42 are advanced test cases. We detect 88 cryptographic misuses, i.e., true positive alerts. Regarding true negatives, we consider only the cases where a tool shows the case as a false alert. With this consideration, we show 33 true negative cases.

IV. EVALUATION

Performance Evaluation: Our experimental evaluation revealed some interesting insights. For complex cases, specialized tools (e.g., CryptoGuard, CrySL) detect more cryptographic misuses and cover more rules than general-purpose tools (e.g., SpotBugs, Tool A). Currently, none of these tools supports path-sensitive analysis. In the real world codes, the number of basic cases is much higher than advanced cases.

V. CONCLUSION

Scientific, reproducible, and in-depth comparisons are essential components. In this paper, we present CryptoAPI-Bench and ApacheCryptoAPI-Bench to evaluate the accuracy, scalability, and security guarantees of various cryptographic misuse detection tools.

REFERENCES