Poster: Deployment-quality and Accessible Solutions for Cryptography Code Development

Sazzadur Rahaman1, Ya Xiao1, Sharmin Afroze1, Ke Tian1, Miles Frantz1, Danfeng (Daphne) Yao1, Na Meng1, Barton P. Miller2, Fahad Shaon3, Murat Kantarcioglu3

1Computer Science, Virginia Tech, Blacksburg, VA
2 Computer Science, University of Wisconsin-Madison, Madison, WI
3 Computer Science, University of Texas at Dallas, Dallas, TX
{sazzad14, yax99, sharminafrose, ketian, frantzme, danfeng, nm8247}@vt.edu, bart@cs.wisc.edu, {FahadShaon, muratk}@utdallas.edu

Abstract—The prevalence of Cryptographic API misuses seriously threatens software security. Automatic screening of cryptographic misuse vulnerabilities has been a popular and important line of research over the years. However, the vision of producing a scalable detection tool that developers can routinely use to screen millions of lines of code is still unattained.

Our main technical enabler is a high precision and high throughput approach based on specialized program analysis. Specifically, we design inter-procedural program slicing on top of a new on-demand flow-, context- and field-sensitive data flow analysis. Our current prototype named CryptoGuard can detect a wide range of Java cryptographic API misuses with a precision of 98.61%, when evaluated on 46 complex Apache Software Foundation projects (including, Spark, Ranger, and Obfiz). Our evaluation on 6,181 Android apps also generated many security insights. We created a comprehensive benchmark named CryptoAPI-Bench with 38-unit basic cases and 74-unit advanced cases for in-depth comparison with leading solutions (e.g., SpotBugs, CrySL, Coverity). To make CryptoGuard widely accessible, we are in the process of integrating CryptoGuard with the Software Assurance Marketplace (SWAMP). SWAMP is a popular no-cost service for continuous software assurance and static code analysis.

I. INTRODUCTION

Cryptography offers provable security guarantees in the presence of adversaries. Various software libraries and frameworks provide a variety of cryptographic APIs to support secure coding. Cryptographic API misuses, such as exposed secrets, predictable random numbers, and vulnerable certificate verification, seriously threaten software security [5]–[7], [10].

The research solution that we aim to transfer to practice in this project addresses the pervasive problem of cryptographic coding vulnerabilities in real-world software. Specifically, our ongoing goals are to produce high quality code screening tools and make them accessible to the developers in various convenient form, including, standalone, IDE plugin (e.g., Eclipse, IntelliJ IDEA), build tool plugin (e.g., Gradle, Maven), code screening as a service (e.g., Software Assurance Marketplace aka, SWAMP).

We have made substantial progress toward building a high accuracy and low runtime static analysis solution for detecting 16 types of cryptographic and SSL/TLS API misuse vulnerabilities. The main technical enabler is the use of highly optimized forward and backward program slicing algorithms, which are built on top of on-demand flow-, context- and field-sensitive data-flow analysis [11].

Detection accuracy. CryptoGuard’s data-flow analysis adopts a set of refinement insights that systematically discard false alerts. These refinement insights (RI) are deduced by observing common programming idioms and language restrictions to remove irrelevant elements, i.e., resource identifiers, vulnerabilities, etc. We also categorize their severity into high, medium, and low, based on i) attacker’s gain and ii) attack difficulty. Vulnerabilities from predictable secrets, SSL/TLS MitM and insecure Hash are immediately exploitable, hence are classified as high risks. Vulnerabilities from predictability and CPA provide substantial advantages to attackers by significantly reducing attack efforts. They are at medium-level risks. Brute-forcing ciphers, requiring non-trivial effort, is low risk.

![Fig. 1. Reduction of false positives with refinement insights in 46 Apache projects and 6,181 Android apps. Top 6 rules with maximum reductions are shown [11].](image-url)

1Available at https://github.com/CryptoGuardOSS/cryptoguard
arguments about states of operations, constants on infeasible paths, and bookkeeping values. For eight of our rules, these refinement algorithms reduce the total number of alerts by 76% in Apache and 80% in Android (Figure 1). Our manual analysis shows that CRYPTO GUARD has a precision of 98.61% on Apache [11].

**Runtime overhead and coverage.** Existing flow-, context- and field-sensitive analysis techniques build a super control-flow graph of the entire program, which has a significant impact on runtime. In contrast, our on-demand slicing algorithms run much faster, which start from the slicing criteria and only propagate to the methods that have the potential to impact security. Hence, a large portion of the code base is not touched. For Apache projects, the average runtime was 3.3 minutes with a median of around 1 minute. For Android apps, we terminated unfinished analysis after 10 minutes. The average runtime was 3.2 minutes with a median of 2.85 minutes [11].

**Comparison with other tools.** We construct CRYPTOAPI-BENCH², a comprehensive benchmark with 112 cases for comparing the quality of cryptographic vulnerability detection tools. CRYPTOAPI-BENCH covers 16 types of cryptographic misuses. In CRYPTOAPI-BENCH, there are 38 basic test cases and 74 advanced test cases. Experimental evaluation shows that CRYPTO GUARD outperforms the state-of-the-art open source and commercial solutions in this space, including CrySL [9], SpotBugs [2], and the free online version of Coverity [1], in terms of precision and recall [11]. For runtime comparison, we ran CrySL and CRYPTO GUARD on 10 randomly selected Apache projects. Unfortunately, CrySL crashed and exit prematurely for 7 of them. For the 3 completed projects, CrySL is slower, but comparable on 2 projects (5 vs. 3 seconds, 25 vs. 19 seconds). However, it is 3 orders of magnitude slower than CRYPTO GUARD on kerberos-codec [11]. During our experiments, we use CrySL 1.0 (commit id 10e86fd8b), SpotBugs 3.0.1 (from SW AMP) and the results from Coverity was obtained before Jan 07, 2019.

**New security findings.** Using our research prototype CRYPTO GUARD, we have successfully screened 46 large open source projects on Apache Software Foundation and 6,181 Android apps from Google Play Market. We discovered a wide range of security issues in real-world coding practices. In Apache projects, there is a widespread insecure practice of storing plaintext passwords in code or in configuration files. Insecure uses of SSL/TLS APIs are set as the default configuration in some cases. For some of them, end-users are susceptible to use insecure default configurations due to lack of proper warning or documentations. In Android apps, 95% of the vulnerabilities come from the libraries that are packaged with the applications. Some libraries are from renowned software firms.

However, these security issues are the tip of the insecure coding iceberg. Through our disclosure interactions with developers and observations from StackOverflow forum [3], [4], [10], we found that a substantial number of developers did not appear to understand the concepts or implications of security API usage. The unfortunate reality is that most developers, with tight project deadlines and short product turnaround time, are not willing to spend effort on hardening their code for long-term benefits. Thus, it is unrealistic to assume that developers will better themselves on their own without any external help.

**Ongoing and future work.** We aim to transition secure cryptographic coding research solutions to practice. Our on-going effort is to make code screening convenient and accessible to mass developers. We are integrating CRYPTO GUARD with Software Assurance Marketplace (SWAMP), an one of the most popular free-of-cost service for continuous software assurance and static code analysis. In SWAMP, programmers can access over 30 scanning tools for a wide variety of languages and platforms. Typically, developers upload their codes or binaries to SWAMP for analysis. There is also a locally installable version of the SWAMP, called SWAMP-in-a-Box, for users that cannot upload their code to an external facility. Each week, the SWAMP performs thousands of assessments, and hundreds of copies of SWAMP-in-a-Box have been downloaded. After successful integration, SWAMP will be able to offer a comprehensive cryptographic misuse detection service to thousands of its users. We plan to create CRYPTO GUARD plugins for popular Java IDE environments, namely IntelliJ IDEA and Eclipse. The only cryptography-related IDE is Eclipse’s CogniCrypt plugin, which is for a code assistant tool (i.e., auto-complete of crypto APIs) [8], not for vulnerability detection. We also plan to create CRYPTO GUARD plugins for Apache Maven and Gradle. Enabling crypto code screening in the early stages of the software development cycle will be more effective. Orthogonally, we plan to upgrade relevant Java static analysis tools (namely, Soot) to newer versions of Java, which will generate impact beyond the specific crypto problem ³.

References


²Available at https://github.com/CryptoGuardOSS/cryptoapi-bench

³The current Soot does not support Java 9 or above.