Hochschule Düsseldorf
University of Applied Sciences

Prof. Dr. Holger Schmidt
Professur für Informatik, insb. IT-Sicherheit
Fachbereich Medien – Hochschule Düsseldorf

Fachbereich Medien
Faculty of Media

Übungsblatt 6 **Entwicklung sicherer Software – SS 2021** Datum: 26.05.2021

Aufgabe 6.1 Unterstütztes Selbstlernen

Bereiten Sie die nächste Vorlesung basierend auf [Hook and Eaves, 2020, Seiten 43 – 54] vor, indem Sie

- die Texte lesen und verstehen,
- sich Fragen und Notizen aufschreiben sowie
- den dargestellten Quellcode selbst praktisch ausprobieren.

Anmerkungen:

Um das Textverständnis zu verbessern, müssen Sie typischerweise weitere Quellen zu Rate ziehen. Halten Sie Ihre aktuellen Implementierungsergebnisse bereit, sodass wir Ihren Quellcode besprechen können.

Der Quellcode aus Hook and Eaves [2020] ist unter folgender URL bereitgestellt:
`https://leanpub.com/s/XCxd95GsH0-8tLTunJ6rhA.zip`

Aufgabe 6.2

Erweitern Sie Ihre Software um eine einfache Hashing/MAC/HMAC-Komponente basierend auf [Hook and Eaves, 2020, Seiten 43 – 54]. Ihre Komponente soll folgende Verfahren unterstützen:

- `SHA-256`
- `AESCMAC`
- `HMACSHA256`

Dazu soll Ihre Software bei jedem Speichervorgang einen Hashwert abspeichern und diesen beim Ladevorgang verifzieren sowie ggf. einen Hinweis auf einen manipulierten Textinhalt ausgeben. Testen Sie Ihre Erweiterung mit neuen JUnit-Tests und analysieren Sie die Testabdeckung mithilfe von JaCoCo. Dokumentieren Sie Ihren Quellcode und Test-Quellcode weiterhin mit Javadoc und Doxygen.

Aufgabe 6.3

Entwerfen Sie ein z. B. XML-basiertes Dateiformat als Container zur Speicherung aller relevanten Daten, die Ihre Software aktuell und zukünftig verarbeiten kann. Dazu zählen z. B. der Ciphertext, das verwendete Verschlüsselungsverfahren, der Hashwert, das verwendete Hashing-Verfahren, Schlüssellänge, Padding-Verfahren und Block-Modus. Implementieren Sie das Dateiformat und integrieren Sie eine Komponente zum Speichern und Laden des Dateiformats in Ihre Software.
Testen Sie Ihre Erweiterung mit neuen JUnit-Tests und analysieren Sie die Testabdeckung mithilfe von JaCoCo. Dokumentieren Sie Ihren Quellcode und Test-Quellcode weiterhin mit Javadoc und Doxygen.

Aufgabe 6.4

Mit Message Digests (aka Hashes) und Message Authentication Codes (MAC) haben wir zwei kryptografische Maßnahmen für die Überprüfung der Datenintegrität kennengelernt. Hashed MACs (HMAC) sind dabei spezielle MACs, die Hashing-Algorithmen einsetzen.

Beantworten Sie folgende Fragen:

a) Betrachten Sie zunächst Message Digests im Zusammenhang mit der Verschlüsselung eines Klartextes. Sie können nun einen Hash über den Klartext generieren und Klartext sowie Hash konkateniert verschlüsseln. Alternativ können Sie nur den Klartext verschlüsseln und den Hash nicht, oder Sie verschlüsseln den Klartext erst und generieren einen Hash über den Ciphertext. Analysieren Sie die drei Varianten hinsichtlich Vor- und Nachteilen. Welche Variante ist im Hinblick auf den Schutz der Integrität des Klartextes korrekt?

b) Welches Problem lösen Message Digests nicht, sodass es sinnvoll ist MACs einzusetzen?

Literatur

David Hook and Jon Eaves. *Java Cryptography: Tools and Techniques*. Leanpub, 2020. Version from 2020-10-10.