UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Praktikum 9

Projekt Teil 2: Absichern einer Client/Server-Anwendung

Zu bearbeiten bis zum 06. Juli 2023

Cybersecurity im Sommersemester 2023

Jan Wichelmann, Anja Köhl

Aufgabe 1 Absichern

Aufbauend auf der Schwachstellenanalyse vom ersten Projektteil soll die vorliegende Implementierung nun abgesichert werden. Grundsätzlich dürfen beliebige Änderungen an Programm und Protokoll vorgenommen werden; um die Kompatibilität zur Praktikumsumgebung zu bewahren und die Einarbeitungszeit der anderen Gruppen später klein zu halten, sind jedoch einige Einschränkungen nötig:

1. Die Benutzeroberfläche des Clients muss sich genauso verhalten wie im Originalprogramm, Ein-/Ausgabeformate dürfen also nicht verändert werden (mehr Debug-Ausgaben sind in Ordnung). Dies stellt sicher, dass die automatischen Tests auf dem Praktikumsserver funktionieren.
2. Der Aufbau des Programms bleibt grob erhalten; dies betrifft die aktuelle Aufteilung in Tasks im Client und die Bearbeitung von Verbindungen in `ClientThread`-Instanzen im Server. Es sollte also keine vollständige Neuentwicklung abgegeben werden. Weiterhin sollte das Programm natürlich nach wie vor die Funktionen Login, Kontostand, Authentifizierung/Registrierung und Überweisung unterstützen.
3. Nutzen Sie keine externen Programme, die nicht definitiv vorinstalliert sind[1]; falls Sie Java-Bibliotheken hinzufügen, legen Sie diese bitte im jeweiligen `lib/`-Ordner ab.
4. Nehmen Sie an folgenden Skripten, Dateien und Funktionen keinerlei Änderungen vor:
   - `compile.sh`
   - `client.sh`
   - `server.sh`
   - `src/LabEnvironment.java`
   - `safePrintln()` und `safeDebugPrintln()` in `src/Utility.java`
5. Abgesehen von der Datenbank-Datei im Server werden alle Teile und Konfigurationsdateien des Programms den angreifenden Gruppen zur Verfügung gestellt. Etwaige geheime Daten müssen also bei der Generierung in der Datenbank-Datei abgelegt werden.
6. Das Datenbankformat darf folglich erweitert werden, jedoch dürfen keine Felder entfernt werden. Falls Sie Felder hinzufügen, sollten diese in der `Database.generate`-Funktion entsprechend sinnvoll initialisiert werden.
7. Der Server darf zur Laufzeit in keine nicht-temporären Dateien schreiben (also auch nicht in die Datenbank). Erlaubt sind Zugriffe auf temporäre Dateien (`/tmp`-Ordner), die nach jeder Ausführung verloren gehen können. Der Client darf im per Kommandozeile übergebenen Gerätecode-Ordner beliebige Dateien ablegen; bedenken Sie jedoch, dass diese Dateien nach Ausführung von Szenario 1 wieder gelöscht werden.
8. Die Passwort-Generierung in der `Database.generate`-Funktion darf nicht verändert werden (10 Ziffern für Szenario 1, 6 Ziffern für Szenarien 2 und 3).

[1] Die Laufzeitumgebung ist ein Docker-Container basierend auf `openjdk:17-alpine`.

9. Die Überprüfung einer bis zu 10-stelligen PIN soll nach wie vor maximal rund fünf Sekunden dauern.
10. Confirmation Codes dürfen nach wie vor maximal 4 Zeichen lang sein und nur aus ASCII-Buchstaben und -Ziffern bestehen.
11. Verlassen Sie sich nicht darauf, dass die im Server sichtbare IP und der Port eines Clients dessen tatsächlichen Verbindungsdaten entspricht: Die Server werden in Docker-Containern gehostet, die ein eigenes internes Netzwerk mitbringen, und damit oft andere Adressen haben, als von außen sichtbar ist.
12. Halten Sie die Anzahl der Ausgaben im Server niedrig; idealerweise nutzen Sie für alle Debugausgaben die `Utility.debugPrintln`-Funktion. Zu viele „normale“ Ausgaben via `Utility.safePrintln` können die Kommunikation zwischen den Docker-Containern überlasten, sodass diese sich in der Folge aufhängen.
13. Selbstverständlich sollte der Quelltext nach wie vor gut lesbar sein (kein *security by obscurity*).

Bitte reichen Sie Ihre Lösungen hierfür über den Praktikumsserver ein, inklusive einer Textdatei `Changes.txt` mit einer Beschreibung der an Implementierung und Protokoll vorgenommenen Änderungen.

Zum Bestehen des Praktikums muss ein ernsthafter Versuch erkennbar sein, die im ersten Projektteil gefundenen Lücken zu schließen.

Aufgabe 2 Testen

Stellen Sie zusätzlich zu Ihren eigenen Tests sicher, dass Ihre Implementierung auch wie vorgesehen auf dem Praktikumsserver funktioniert. Lösen Sie dazu nach dem Hochladen die automatisierte Validierung aus. Diese führt die folgenden Schritte durch, und gleicht dabei die Ausgabe des Clients mit einer erwarteten Ausgabe ab:

1. Starten des Praktikumsservers
2. Starten des Client, Verbindungsaufbau mit Praktikumsserver
3. Login `victim1`
4. Kontoverlauf
5. Registrierung (Generierung eines neuen Device-Codes *`device-code`*)
6. Überweisung
7. Verbindungstrennung, Beenden des Client
8. Starten des Client, Verbindungsaufbau mit Praktikumsserver
9. Login `victim1`
10. Kontoverlauf
11. Authentifizierung (unter Benutzung des Device-Codes *`device-code`*)
12. Überweisung
13. Verbindungstrennung, Beenden des Client
14. Beenden des Praktikumsservers

Wenn die Validierung erfolgreich beendet wird, sollte Ihre Implementierung mit der Praktikumsumgebung voll kompatibel sein.

Falls sich Probleme ergeben sollten, die Sie auch nach erneuter Lektüre der Einschränkungen aus Aufgabe 1 nicht erklären können, wenden Sie sich an einen der Betreuer.