UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Praktikum 9

Projekt Teil 1: Angreifen einer Client/Server-Anwendung

Zu bearbeiten bis zum 06. Juli 2023

Cybersecurity im Sommersemester 2023

Jan Wichelmann, Anja Köhl

Einführung

Die Unternehmensführung von *Immature Technical Solutions* beschloss neue Geschäftsmodelle zu erkunden, und gründete eine Online-Bank. Hierfür beauftragte sie einige ihrer verbliebenen Praktikanten, eine geeignete Software zu entwickeln. Diese sollte folgende Anforderungen erfüllen:

- Client/Server-System, wobei der Client frei bei der Bank zum Download bereit steht und beliebig portabel ist;
- Klassischer Login mit Benutzername und Passwort;
- Anzeige des Kontostandes nach Login;
- Registrierung eines Gerätes durch Angeben eines Verifikationscodes, der über einen zweiten Kanal versendet wurde (Two-Factor);
- Durchführung von Überweisungen mithilfe der so registrierten Geräte.

Dieser zusätzliche Schutz sollte sicherstellen, dass selbst bei Diebstahl von Benutzername und Passwort keine Überweisungen im Namen des Opfers durchgeführt werden können, da hierzu auch der Verifikationscode (und damit Zugriff auf den zweiten Kanal) notwendig wäre.

Die Entwicklung ging gut voran, und innerhalb von wenigen Tagen hatten die Praktikanten (unter Benutzung der für ihre hohe Sicherheit bekannten Programmiersprache *Java*) eine funktionierende Lösung entwickelt, die an die frisch gewonnenen Kunden weitergegeben und trotz der eher schlichten Benutzungsoberfläche von diesen auch gut angenommen wurde.

Kurz darauf begannen sich jedoch Beschwerden über angeblich unrechtmäßige Überweisungen an dubiose Konten zu häufen, welche die Bank mit Hinweis auf das durchdachte Schutzsystem bei den Überweisungen selbstverständlich entschieden zurückwies; stattdessen vermutete man die Leichtgläubigkeit der Kunden bei verlockenden E-Mail-Angeboten (*sicheres Investment mit 200% Rendite!*) als Grund. Die Vorstandsvorsitzende ging sogar so weit, in einem Video die sichere Funktionsweise des Programms zu demonstrieren, in dem sie vergeblich versuchte mit ihrem Konto eine Überweisung ohne Geräte-Registrierung durchzuführen. Zudem zeigte sie, dass dank der Deckelung des maximalen Betrags pro Überweisung selbst bei Besitz eines registrierten Gerätes ein Diebstahl größerer Summen sehr zeitaufwändig ist, und daher bestimmt bemerkt werden würde. Wenige Minuten später hatten Unbekannte ihr Konto leergeräumt, und das Geld sicher auf einer Inselgruppe geparkt.

Nun doch etwas beunruhigt, beschloss der Bankvorstand eine unabhängige Sicherheitsüberprüfung der besagten Software in Auftrag zu geben, und etwaige Sicherheitslücken bei dieser Gelegenheit schließen zu lassen.

Ziel des Projekts

Das Ziel über die kommenden Wochen ist es, eine ausführliche Analyse von Schwachstellen in der vorliegenden Implementierung durchzuführen, und diese anschließend abzusichern.

In den nächsten Abschnitten finden Sie zuerst eine Beschreibung des zugrundeliegenden Kommunikationsprotokolls (das später im Rahmen der Schließung von Sicherheitslücken beliebig modifiziert werden darf) und einige Hinweise zum Aufbau der Testumgebung und zur Implementierung; schließlich folgt die Aufgabe für den ersten Praktikumstermin.

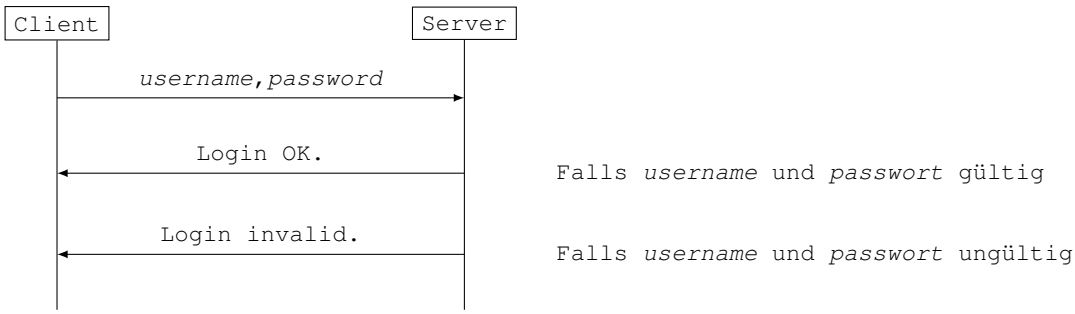
Das Protokoll

Das Protokoll unterstützt fünf mögliche Interaktionen, die im Folgenden im Detail beschrieben sind:

1. Login.

   Voraussetzungen: *Keine*.

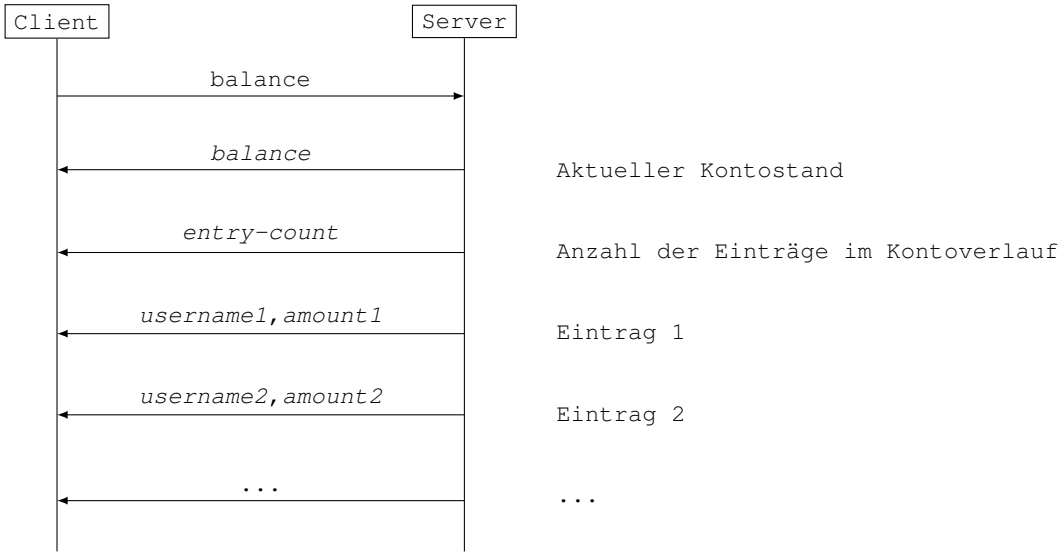
   Der Client sendet seinen Benutzernamen und das Passwort. Der Server prüft diese; falls die Logindaten gültig sind, wird eine Erfolgsnachricht gesendet und der Benutzer ist für diese Sitzung angemeldet. Falls die Logindaten ungültig sind, wird eine Fehlernachricht gesendet und der Login wiederholt.



2. Balance.

   Voraussetzungen: Login.

   Nach Aufforderung des Clients sendet der Server dessen Kontostand und Transaktionsverlauf.

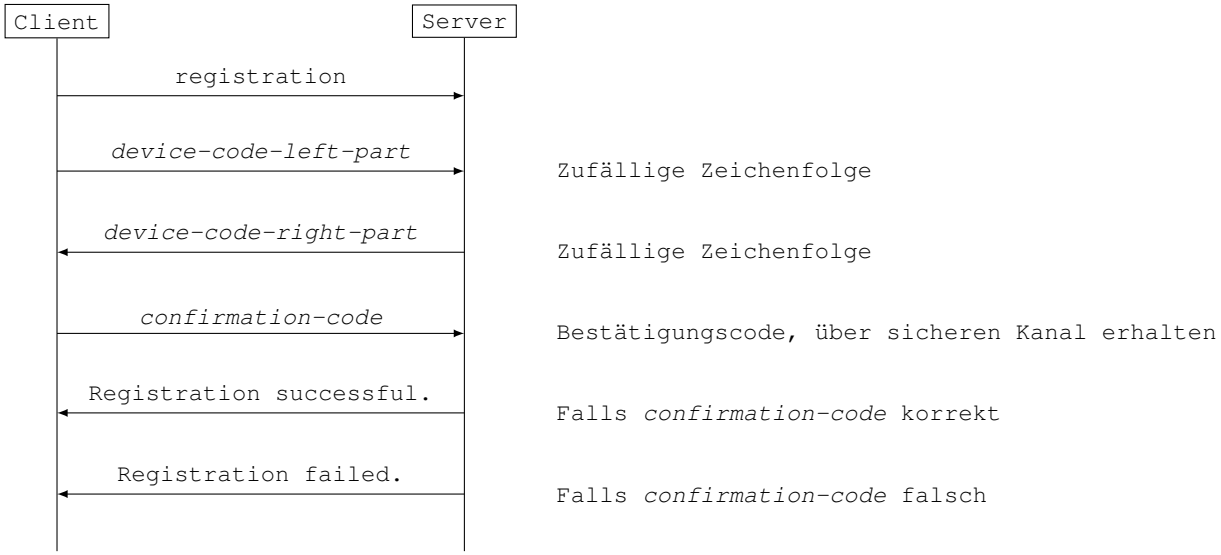


3. Registration.

   Voraussetzungen: Login.

   Client und Server vereinbaren einen Device-Code, indem jeder eine Hälfte generiert und diesem dem anderen sendet. Der Server sendet daraufhin einen Bestätigungscode über einen sicheren (aber sehr beschränkten) Kanal an den Client, welchen der Client angeben muss, um die Registrierung zu beenden und den generierten Device-Code damit gültig zu machen. Dieser Device-Code kann dann für spätere Authentifizierungen genutzt werden.

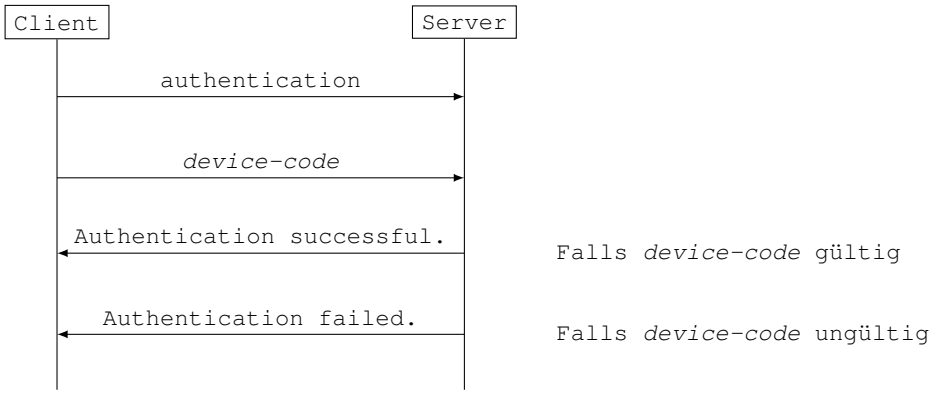
Nach der Registrierung ist der Benutzer direkt authentifiziert und kann den Transaction-Befehl benutzen, d.h. der Authentication-Schritt fällt weg.



4. Authentication.

   Voraussetzungen: Login.

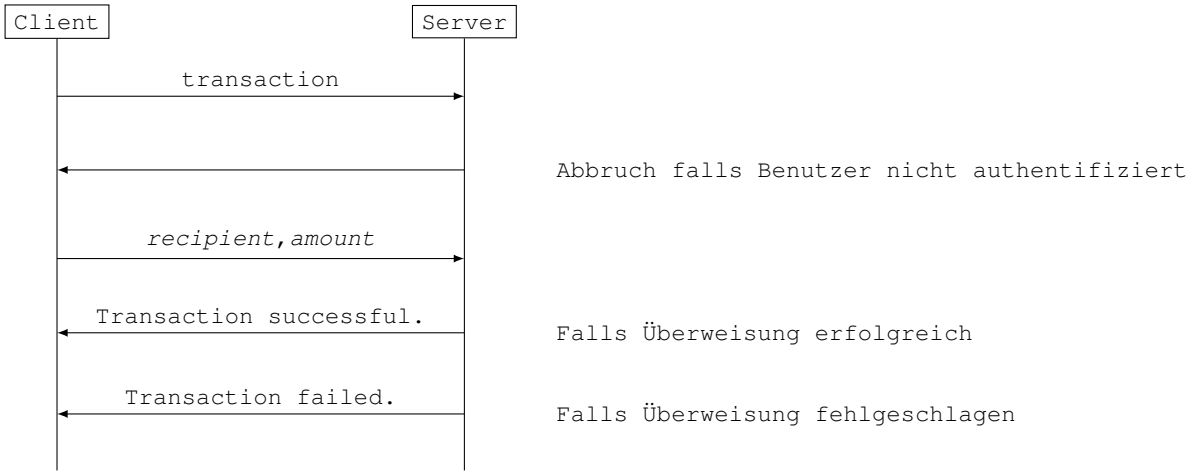
   Der Client sendet seinen Device-Code. Der Server überprüft diesen auf Gültigkeit und sendet wahlweise eine Bestätigung oder eine Fehlernachricht. Danach ist der Benutzer authentifiziert und kann den Transaction-Befehl benutzen.



5. Transaction.

   Voraussetzungen: Login; Authentication *oder* Registration.

   Nach der Authentifizierung gibt der Benutzer den Namen eines Empfängers und die zu überweisende Geldmenge an. Der Server führt die Überweisung durch und sendet eine Bestätigung bei Erfolg, bzw. eine Fehlermeldung falls die Überweisung fehlgeschlagen ist (unbekannter Empfänger, zu wenig Geld auf dem Konto).



Hinweise zur Implementierung

Die Implementierung ist mit Java 17 erstellt, der Quelltext und etwaige Abhängigkeiten sind vollständig im Moodle zu finden. Es wird empfohlen, der Übersichtlichkeit halber in einer IDE zu arbeiten.

Der Server erstellt ein Socket auf einem festgelegten Port, und wartet dort auf neue Clients. Jede neue Client-Verbindung wird dann in einen eigenen Thread `ClientThread` ausgelagert, der die Kommunikation mit dem Client durchführt. Der Thread wird beendet, sobald die Client-Verbindung getrennt wird. Das Warten auf Clients und die Erstellung von Threads sollten kein relevantes Sicherheitsrisiko darstellen (höchstens für die Verfügbarkeit der Anwendung, was aber nicht Untersuchungsgegenstand ist), d.h. die Klasse `ServerMain` kann im weiteren Verlauf des Projekts ignoriert werden. Das gleiche gilt für die Klasse `LabEnvironment`, welche lediglich einige für die Praktikumsumgebung spezifische Hilfsfunktionen enthält. Die Klasse `Database` verwaltet sämtliche Nutzerdaten (Namen, Device-Codes, Kontostände...), die bei Start aus einer JSON-Datei gelesen und danach im Speicher gehalten werden. Eine Speicherfunktion ist nicht vorgesehen, d.h. etwaige Änderungen während der Laufzeit gehen absichtlich bei Programmende verloren.

Beim Client steuert die Klasse `ClientMain` den allgemeinen Ablauf des Protokolls, und erlaubt dem Benutzer die Wahl der nächsten durchzuführenden Aktion; die einzelnen Aktionen werden von Hilfsklassen (erkennbar an dem Suffix `Task` im Namen) implementiert.

Es liegen folgende Bash-Skripte bei (diese dienen zur Referenz und zur Automatisierung der Praktikumsumgebung, und müssen nicht verwendet werden):

- `config.sh`: Legt die Pfade zu den `javac` und `java`-Binaries fest; muss für das aktuelle System angepasst werden, falls die nachfolgenden Skripte verwendet werden.
- `compile.sh`: Kompiliert Client und Server, und erlaubt das Erzeugen einer neuen Datenbank mit Zugangsdaten.
- `server.sh`: Startet den Server mit der gegebenen Datenbank.
- `client.sh`: Startet den Client und baut eine Verbindung mit dem angegebenen Server auf.

Die Skripte erwarten jeweils einige Parameter, welche in der beiliegenden README-Datei dokumentiert sind.

Wir empfehlen die Nutzung einer IDE zum Bearbeiten und Ausführen des Codes. Die Basisimplementierung wurde mit IntelliJ IDEA erstellt, vorkonfigurierte Projektdateien liegen anbei.

Hinweise zur Testumgebung

Sie können wie gewohnt über das CTF-System einen Praktikumsserver erreichen, der Ihnen die Steuerung der Ihnen zugeordneten Bankserver-Instanz erlaubt. Gleichzeitig werden Ihnen dort die Confirmation Codes und weitere für die Bearbeitung der Aufgaben relevante Zugangsdaten angezeigt.

Auf den jeweiligen Bankservern befinden sich folgende Accounts:

- Ein Account für Ihre Gruppe (`group`, Passwort `0000`).
- Accounts, von denen im Rahmen der zu untersuchenden Szenarien Geld gestohlen werden soll.
- Ein Account `dummy`, für Tests und als Hilfsmittel für die Angriffsszenarien.

Beachten Sie, dass die Bankserver nur über das VPN erreichbar sind.

Aufgabe 1 Angriffe

Wir untersuchen drei Angriffsszenarien, die sich im Wissen und den Fähigkeiten des Angreifers unterscheiden, und in beliebiger Reihenfolge bearbeitet werden können:

1. Der Angreifer kennt $\texttt{username}_1$ und befindet sich zwischen Client und Server (Man-in-the-Middle, siehe Abbildung unten). ◀ 70 / 0
2. Der Angreifer kennt $\texttt{username}_2$ und $\texttt{password}_2$. ◀ 70 / 0
3. Der Angreifer kennt $\texttt{username}_3$. ◀ 70 / 0

Klicken Sie zur Simulation des Szenarios 1 auf den entsprechenden Knopf im Hauptserver; dieser wird dann unter Benutzung des Clientprogramms folgende Protokollschritte ausführen:

Client-Registrierung

- Verbindungsaufbau mit Ihrem System (täuscht Bankserver vor)
- Login `victim1`
- Kontoverlauf
- Registrierung (Generierung eines neuen Device-Codes *device-code*)
- Überweisung
- Verbindungstrennung

Client-Authentifizierung

- Verbindungsaufbau mit Ihrem System (täuscht Bankserver vor)
- Login `victim1`
- Kontoverlauf
- Authentifizierung (unter Benutzung des Device-Codes *device-code*)
- Überweisung
- Verbindungstrennung

Die Verbindung über Ihr System simuliert einen erfolgreichen ARP-Spoofing-Angriff. `victim1` ist allerdings nicht unendlich geduldig: Der Client wird nach spätestens 15 Sekunden automatisch beendet, Ihr Man-in-the-Middle-Angriff muss also schnell genug sein!

Jede Gruppe erhält einen entsprechenden Satz Zugangsdaten; zum erfolgreichen Abschließen eines Angriffsszenarios i muss ein beliebiger positiver Betrag $1 \leq x_i \leq 10$ von Konto $\texttt{victim}_i$ auf Ihr Gruppenkonto überwiesen werden. Jedes Angriffsszenario bringt hierbei eine bestimmte Zahl von Punkten; Sie müssen zum Bestehen in mindestens einem Szenario Geld stehlen.

Hinweise:

- Um die eventuell notwendigen Brute-Force-Angriffe ohne zu starke Belastung des Netzwerks zu ermöglichen, bestehen die Passwörter nur aus Ziffern (wie bei Banken durchaus üblich).
- Falls Sie beim Man-in-the-Middle-Szenario keine Pakete vom Client empfangen können, lohnt es sich vermutlich, einmal die Firewall-Einstellungen zu überprüfen.

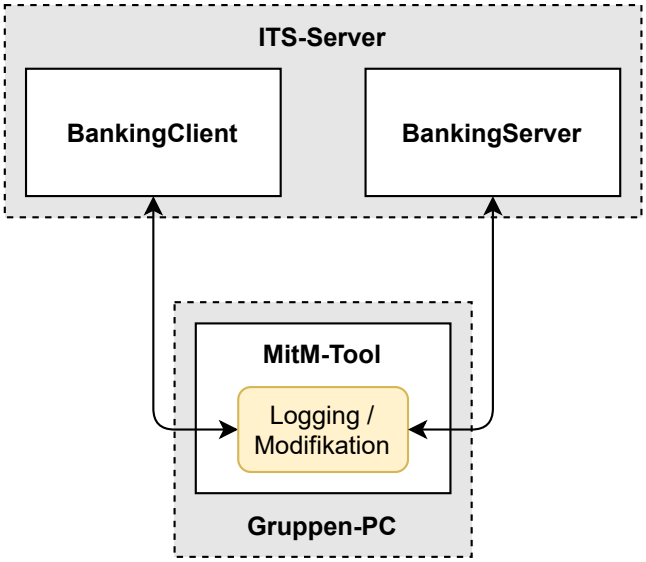


Abbildung 1: Man-in-the-Middle-Aufbau von Angriffsszenario 1. Client und Server kommunizieren nicht direkt, sondern nur indirekt über Ihr System. Das gibt Ihnen die Möglichkeit, sämtlichen Datenverkehr mitzuschneiden und zu verändern.