MNM
TEAM
Munich Network Management Team

Ludwig-Maximilians-Universität München

Prof. Dr. D. Kranzlmüller
Dr. N. gentschen Felde

Willkommen in der Gruppenphase des Systempraktikums. Ihre Aufgabe in der Projektphase ist es, einen Client für das Brettspiel *Baschni* in der Programmiersprache `C` zu entwickeln. Die Übungsblätter werden Sie schrittweise zu diesem Ziel führen.

Der Lehrstuhl stellt im Rahmen des Systempraktikums einen Gameserver und ein Webinterface bereit. Der Gameserver implementiert den Spielablauf, Ihr Client die Spielelogik. Der Gameerver ist gewissermaßen der Spielleiter und somit insbesondere für die Regeleinhaltung verantwortlich, wohingegen Ihr Client in die Rolle eines Spielers schlüpft. Über das Webinterface unter `http://sysprak.priv.lab.nm.ifi.lmu.de` können Spiele verwaltet werden. Der Gameserver und das Webinterface sind nur aus dem MWN[1] erreichbar (Hinweis: `ssh -L` auf die CIP-Rechner[2].

Um das Testen Ihres Clients zu erleichtern, bietet das Webinterface zudem die Möglichkeit als menschlicher Spieler an einem Spiel teilzunehmen. Sie können jetzt testweise ein neues Spiel erstellen und zum Einstieg gegen ihre Kommilitonen oder sich selbst spielen. Sie werden einen Einblick gewinnen, wie das Spiel abläuft.

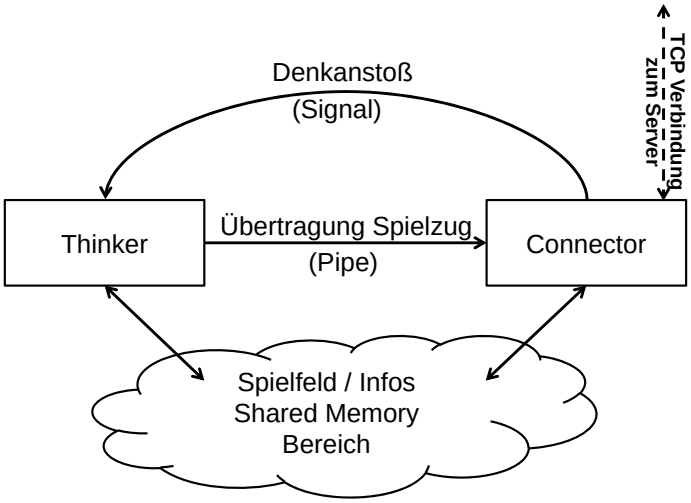

Abbildung 1: Übersicht über den Client

Der von Ihnen zu entwickelnde Client wird im Verlaufe des Systempraktikums schrittweise entwickelt. Maßgeblich besteht er aus zwei Prozessen, dem *Thinker* und dem *Connector*. Der *Connector* übernimmt die Kommunikation mit dem Gameserver und der *Thinker* berechnet den nächsten Spielzug.

Wird der Connector vom Gameserver zum nächsten Spielzug aufgefordert, so legt er alle Informationen, die er vom Gameserver bekommen hat um den nächsten Spielzug zu berechnen, in einen *geteilten Speicherbereich*. Danach sendet der Connector ein *Signal* an den Thinker. Der dadurch „aufgeweckte“ Thinker beginnt nun (mit Hilfe der aus dem *geteilten Speicherbereich* gelesenen Informationen) den nächsten Spielzug zu berechnen. Das Ergebnis der Berechnungen (der Spielzug) sendet der Thinker über eine *unnamed Pipe* an den Connector zurück, welcher den Spielzug an den Gameserver sendet. Abbildung 1 gibt Ihnen einen groben Überblick über den Aufbau des Clients.

[1] Informationen zum MWN: `http://www.lrz.de/services/netz/`
[2] Informationen zu ssh: `http://www.rz.informatik.uni-muenchen.de/FAQ/Aussenzugriff.faq.html/`

1 Baschni

Im Jahr 1875 stellte Martin Wiberg seine Rechenmaschine[3] vor, welche voll automatisch Logarithmentafeln drucken konnte. Im selben Jahr wurde auch das erste Mal „Baschni“ (zu deutsch: Türme) in einem Buch über russische Brettspiele erwähnt. Das Spiel war auch als „Stolbowje Schaschki“ (zu deusch: Türmchendame) bekannt, geriet jedoch bald wieder in Vergessenheit. In den 80er Jahren wurde es wieder entdeckt und erlebt seitdem, vor allem im russischsprachigen Raum, seine Renaissance. Bei dem Spiel handelt es sich um eine Abwandlung des russischen Damespiels, bei dem geschlagene Steine nicht vom Brett genommen werden, sondern unter dem schlagenden Stein platziert werden. Dadurch entstehen die Türme, welche dem Spiel seine Namen geben. Im Systempraktikum verwenden wir eine leicht abgewandelte Version.

1.1 Spielregeln

Das Spiel Bashni wird von zwei Spielern gespielt. Der „helle“ Mitspieler verfügt über die grünen Spielfiguren, der „dunkle“ Mitspieler über die schwarzen. Neben der Farbe unterscheiden sich die Spielfiguren noch nach ihrer Art. So gibt es normale Spielfiguren und Damen. Gespielt wird auf dem von den Spielen Schach und Dame bekannten Spielbrettern. Sie bestehen aus 8×8-Feldern, die abwechselnd hell und dunkel sind. Es wird nur auf den dunklen Feldern gespielt. Die Bezeichnung der Felder ist, ebenfalls wie beim Schach, eine Kombination aus einem Buchstaben und einer Zahl.

Die Spieler platzieren sich auf zwei gegenüberliegenden Seiten am Spielfeld. Die Felder von A1 bis H1 sind dabei die sogenannte Grundline des hellen Spielers und A8 bis H8 die des dunklen.

Bei der Startaufstellung zu Beginn einer Partie werden zwölf normale Spielfiguren pro Spieler auf seiner Seite aufgereiht. Die des hellen Mitspielers liegen auf A1, C1, E1, G1, B2, D2, F2, H2, A3, C3, E3 und G3. Die Spielfiguren des dunklen Mitspielers liegen auf H8, F8, D8, B8, G7, E7, C7, A7, H6, F6, D6 und B6.

Der erste Spielzug obliegt dem hellen Spieler. Danach wird abwechselnd gezogen. Ist ein Spieler an der Reihe, kann aber keinen Zug machen, so hat er verloren.

Ist ein Spieler an der Reihe, so wählt er zunächst einen seiner Türme aus, mit dem er den Zug durchführen möchte. Ein Turm besteht aus einem oder mehreren Spielsteinen, gleich welcher Farbe. Ein Turm gehört dem Mitspieler, dem auch der oberste Stein auf dem Turm gehört. Wie der Turm über das Spielfeld gezogen werden darf hängt von der Art des obersten Spielsteins ab.

Ein Spielzug untergliedert sich in Teilzüge. Die Türme bewegen sich bei jedem Teilzug diagonal über das Spielfeld. Bei direkt aufeinanderfolgenden Teilzügen eines Zuges darf sich die Teilzugrichtung nicht um 180° ändern. Es kann also nur in die selbe Richtung, nach links oder rechts weiter gezogen werden. Bei einem Teilzug kann kein oder genau ein gegnerischer Turm übersprungen werden. Dabei wird der oberste Stein des übersprungenen Turms geschlagen. Der geschlagene Stein wird unter dem eigenen Turm, mit dem der Teilzug durchgeführt wurde, platziert. Eigene Türme dürfen in keinem Fall übersprungen werden. Kann in einer Runde mindestens ein schlagender Zug durchgeführt werden, so muss ein beliebiger schlagender Zug durchgeführt werden, wobei es jedoch nicht der Zug mit den meisten Schlägen sein muss. Landet man nach einem schlagenden Teilzug auf einer Position, von der aus man weiter schlagen kann, so muss dies auch getan werden. Ein Zug besteht somit entweder aus genau einem nicht schlagenden Zug oder aus einem bis mehreren schlagenden Zügen.

Ein Turm, bei dem ein normaler Spielstein oben aufliegt, darf nur ein Feld „nach vorne“ (Richtung Gegner) gezogen werden. So darf z. B. der helle Mitspieler von B4 nach A5 und C5 ziehen, nicht jedoch auf A3 oder C3. Analog darf der dunkle Mitspieler z. B. von B4 nach A3 oder C3 ziehen, nicht jedoch auf A5 oder C5. Ist eines der direkt an den Turm angrenzenden Felder mit einem gegnerischen Turm belegt und das unmittelbar folgende Feld frei, so kann ein schlagender Teilzug durchgeführt werden. Dabei kann vorwärts wie auch rückwärts geschlagen werden. Erreicht der Turm die gegnerische Grundlinie, so wird der oberste Spielstein zu einer Dame befördert.

Ein Turm, bei dem eine Dame oben auf liegt, darf in alle Richtungen ziehen. Es darf dabei beliebig weit gezogen werden, solange maximal ein gegnerischer Turm und kein eigener Turm übersprungen wird. Im Gegensatz zu den original Spielregeln kann man seinen Turm bei einem schlagenden Teilzug im Systempraktikum auf einem beliebigen freien Feld hinter dem geschlagenen Turm platzierten.

Ist ein Spieler an der Reihe, kann aber keinen Zug machen, so hat er verloren und die Partie endet. Im Gegensatz zu den original Spielregeln sind im Systempraktikum beliebig viele Stellungswiederholungen möglich und führen nicht zu einem

[3] `https://youtu.be/kF3eIVmu3zQ`

Unentschieden. Stattdessen endet die Partie spätestens nach dem 200. Zug oder nach 50 Zügen, in denen keine neue Dame erzeugt wurde. In diesen Fällen wird ein Gewinner nach einem Punktesystem ermittelt. Haben beide Teilnehmer die gleiche Punktezahl, so Endet das Spiel unentschieden. Die Punkte eines Spielers ergeben sich aus Summe von der Anzahl seiner normalen Türme und der dreifachen Anzahl an Dame-Türmen.

2 Protokolldefinition des Gameservers

Ihr zu implementierender Client kann mit dem Gameserver über das hier beschriebenen zeilenorientierte Protokoll kommunizieren. Zeilen werden, wie unter Unix üblich, mit einem „newline character“ (`'\n'`) abgeschlossen. Der MNM-Gameserver ist wie folgt zu erreichen:

- *Hostname:* `sysprak.priv.lab.nm.ifi.lmu.de`
- *Port:* `1357` (TCP)

Wenn der Gameserver eine Zeile mit einem `+` als erstes Zeichen schickt, ist dies eine positive Antwort. Im Folgenden ist nur der positive Verlauf einer Kommunikation angegeben. An jedem Schritt kann eine Negativantwort auftreten, diese ist erkennbar an dem `-` als erstes Zeichen der Zeile. Ein `-` ist stets gefolgt von einer Fehlermeldung. Im Anschluss an die Fehlermeldung wird die Verbindung getrennt.

Wenn nicht innerhalb von im Gameserver festgelegten Zeitgrenzen auf Befehle geantwortet oder eine zu lange „Denkzeit“ benötigt wird (s. u.), schickt der Gameserver:
S: - TIMEOUT ⟨⟨ *Begründung* ⟩⟩

Die folgende Protokolldefinition kürzt eine Zeile, welche vom Client an den Gameserver geschickt wird, mit **C:** für *Client* ab. Eine Zeile, welche vom Gameserver an den Client übermittelt wird, wird mit **S:** für *Server* abgekürzt.

In ⟨⟨ ⟩⟩ eingeschlossene Werte werden obligatorisch durch die ihnen entsprechenden Werte ersetzt. Werte, die in *[[ ]]* eingeschlossen sind, geben optionale Werte an, d. h. sie können auch weggelassen werden.

Das Protokoll gliedert sich in folgenden drei Phasen:

1. *Prolog* – hier wird dem Spiel beigetreten und Informationen über das Spiel ausgetauscht
2. *Spielverlauf* – hier wird gewartet bis man an der Reihe ist, bzw. das Spiel beendet wird
3. *Spielzug* – hier übermittelt der Gameserver ein Spielfeld und erwartet einen Spielzug

Diese Phasen werden in den folgenden Unterkapiteln ausführlich beschrieben. Das Zustandsdiagramm in Abbildung 2 gibt eine grobe Übersicht über den Ablauf der drei Protokollphasen.

2.1 Protokollphase *Prolog*

⟨⟨ *Aufbau der TCP-Verbindung durch Client* ⟩⟩
S: `+ MNM Gameserver` ⟨⟨ *Gameserver Version* ⟩⟩ `accepting connections`
C: `VERSION` ⟨⟨ *Client Version* ⟩⟩
S: `+ Client version accepted - please send Game-ID to join`
C: `ID` ⟨⟨ *Game-ID* ⟩⟩
S: `+ PLAYING` ⟨⟨ *Gamekind-Name* ⟩⟩
S: `+` ⟨⟨ *Game-Name* ⟩⟩
C: `PLAYER` *[[Gewünschte Spielernummer]]*
S: `+ YOU` ⟨⟨ *Spielernummer* ⟩⟩ ⟨⟨ *Spielername* ⟩⟩
S: `+ TOTAL` ⟨⟨ *Spieleranzahl* ⟩⟩

Nun kommt für jeden der anderen Spieler die Zeile:
S: `+` ⟨⟨ *Spielernummer* ⟩⟩ ⟨⟨ *Spielername* ⟩⟩ ⟨⟨ *Bereit* ⟩⟩

S: `+ ENDPLAYERS`

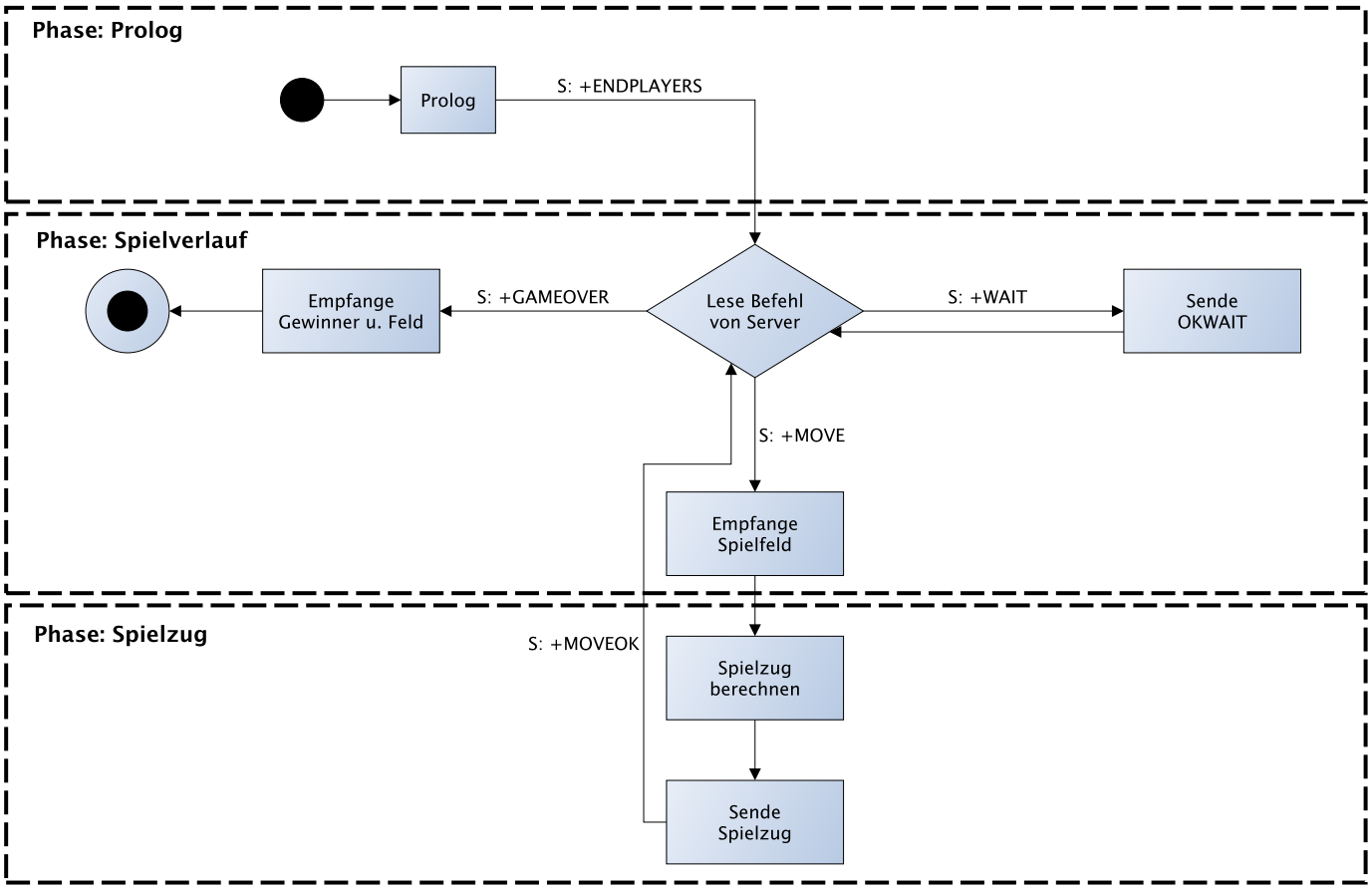

Abbildung 2: Protokollübersicht

Der Gameserver akzeptiert den Client, wenn die Major-Versionsnummern (links vom Punkt) von ⟨⟨ *Gameserver Version* ⟩⟩ und ⟨⟨ *Client Version* ⟩⟩ gleich sind. Der Gameserver setzt jeweils ein `v` voran. Die Major-Versionsnummer des Gameserver ist dieses Semester immer 2, die Minor-Versionsnummer kann sich jedoch ändern. Kompatibel sind z. B. ein Gameserver mit der Version `v2.1` und ein Client mit der Version `2.42`. Nicht kompatibel wären hingegen `v2.1` und `3.1` oder `v2.1` und `v2.1` (Client hat ein `v` Präfix).

Da der Gameserver nicht nur Baschni spielen kann, teilt ⟨⟨ *Gamekind-Name* ⟩⟩ die Spielart mit. Ist diese anders als `Bashni` (englische Schreibweise von *Baschni*), muss der Client sich mit einer Meldung, die auf dieses Problem hinweist, beenden.

⟨⟨ *Game-Name* ⟩⟩ und ⟨⟨ *Spielername* ⟩⟩ können im Webinterface beim Erstellen eines Spiels angegeben werden, bzw. enthalten Standartwerte, wenn die Angabe ausbleibt.

Wenn man einen Spieler mit einer bestimmten Nummer übernehmen möchte, kann dieser Wunsch mit *[[Gewünschte Spielernummer]]* angegeben werden. Lässt man die Spielernummer und das Leerzeichen nach `PLAYER` weg, so bekommt man einen freien Computerspieler vom Gameserver zugeteilt.

Die ⟨⟨ *Spieleranzahl* ⟩⟩ wird bei einem Baschni-Spiel immer `2` sein.

Ist ein Spieler bereits verbunden, so ist ⟨⟨ *Bereit* ⟩⟩ `1`, ansonsten `0`. Auch wenn nicht alle Spieler bereit sein sollten, so wird mit der nächsten Protokollphase fortgefahren.

2.2 Protokollphase *Spielverlauf*

In dieser Phase können eine *Idle*, *Move* sowie eine *Game over* Befehlssequenz vorkommen. Wurde das erste Mal in die Spielverlaufsphase gewechselt oder eine der Befehlssequenzen beendet, so entscheidet die nächste Zeile (die im Diagramm an den Kanten annotiert ist) über die nächste abzuarbeitende Befehlssequenz.

2.2.1 *Idle* Befehlssequenz

S: `+ WAIT`
C: `OKWAIT`

`WAIT`-Befehle kommen in regelmäßigen Abständen und müssen rechtzeitig mit einem `OKWAIT` quittiert werden. Ansonsten beendet der Gameserver die Verbindung und das Spiel gilt im Turnier als verloren. Wie dem Diagramm in Abbildung 2 zu entnehmen ist, bleiben die `WAIT`-Befehle z. B. in der Protokollphase *Spielzug* oder während der *Move* Befehlssequenzen aus.

2.2.2 *Move* Befehlssequenz

S: `+ MOVE` ⟨⟨ *Maximale Zugzeit* ⟩⟩
S: `+ PIECESLIST`
S: `+ w@A1`
S: `+ b@C1`
S: `+ W@E1`
S: `+ B@G1`
... gekürzte Ausgabe
S: `+ ENDPIECESLIST`
C: `THINKING`
S: `+ OKTHINK`

Der `MOVE`-Befehl fordert zum Zug auf. Nach der in Millisekunden angegebenen ⟨⟨ *Maximale Zugzeit* ⟩⟩ muss die anschließende Protokollphase „Spielzug“ abgeschlossen sein. Bitte berücksichtigen Sie bei Ihrer Implementierung die Latenzzeiten der Verbindung. Zwischen `PIECESLIST` und `ENDPIECESLIST` werden Informationen zu jedem Spielstein ausgegeben. Das `+` steht dabei für eine postive Nachricht vom Server. Durch ein Leerzeichen getrennt folgt die Art und die Position des Spielsteins. Die Steine des weißen Spielers sind mit einem `w` gekennzeichnet, die des schwarzen Spielers mit einem `b`. Handelt es sich um Damesteine, so wird die korrespondierende Majuskel anstatt der Minuskel verwendet. Durch ein `@` getrennt folgt die Positionsangabe des Spielsteins. Steine, die zuerst in der Liste auftauchen, sind in den Türmen weiter unten angesiedelt. Es kann jedoch nicht davon ausgegangen werden, dass alle Spielsteine eines Turms direkt hintereinander in der Liste stehen.

Der Client muss direkt nach der Übermittlung des Spielfeldes `THINKING` schicken und hat hierfür wenig Zeit. Nach der Gameserver-Antwort `OKTHINK` befinden wir uns in der Protokollphase „Spielzug“.

2.2.3 *Game over* Befehlssequenz

S: `+ GAMEOVER`
S: `+ PIECESLIST`
S: `+ w@A1`
S: `+ b@C1`
S: `+ W@E1`
S: `+ B@G1`
... gekürzte Ausgabe
S: `+ ENDPIECESLIST`
S: `+ PLAYER0WON` ⟨⟨ *'Yes' oder 'No'* ⟩⟩
S: `+ PLAYER1WON` ⟨⟨ *'Yes' oder 'No'* ⟩⟩
S: `+ QUIT`
⟨⟨ *Abbau der TCP-Verbindung durch Gameserver* ⟩⟩

Nach einem `GAMEOVER` stellen die weiteren Zeilen den Spielzustand dar, mit dem das Spielende erreicht wurde. Dieser wird an alle Spieler geschickt. Zudem wird angegeben, welcher Spieler gewonnen haben. Es ist auch möglich, dass eine Partie in einem Unentschieden endet. In dem Fall ist bei beiden Spielern der Gewinnstatus des Spiels als `Yes` gesetzt. Nach `QUIT` beendet der Server die Verbindung.

2.3 Protokollphase *Spielzug*

C: `PLAY` ⟨⟨ *Spielzug* ⟩⟩
S: `+ MOVEOK`

Ein ⟨⟨ *Spielzug* ⟩⟩ besteht aus Koordinaten, die mit `:` getrennt sind. Die Koordinate wird in der Schachnotation angegeben, also z.B. `C3` oder `G5`. Die erste Koordinate wählt den Turm aus, mit dem der Spielzug durchgeführt werden soll. Die zweite Koordinate gibt das Zug- oder Sprungziel des ersten Teilzuges an, die dritte die des zweiten, und so weiter. Soll z. B. mit dem Turm an Position C5 im ersten Teilzug von C5 nach E7 gesprungen werden und im zweiten Teilzug (mit dem selben Turm) von E7 nach G5, so wird dies als `C5:E7:G5` kodiert. Sollte der übermittelte Zug nicht gültig sein, wird eine entsprechende Fehlermeldung übermittelt und die Verbindung getrennt.

3 Bugreports

Der Gameserver wird jedes Jahr um neue Funktionen und Spiele erweitert. Trotz aller Bemühungen kann es passieren, dass noch Fehler im System sind. Melden Sie diese Bugs bitte gleich an sysprak-bugs@nm.ifi.lmu.de. Für organisatorische Anliegen wenden sie sich bitte an sysprak-admin@nm.ifi.lmu.de.

Damit Fehler auch behoben werden können, ist eine genau Beschreibung des Problem nötig. Bitte versuchen Sie den Fehler so genau wie möglich zu beschreiben und dabei auf mindestens folgende Punkte einzugehen:

- Die URL, auf der das Problem auftrat.
- Die Game-ID.
- Die Schritte, die nötig sind, um das Problem reproduzieren zu können.
- Eine ausführliche Beschreibung des unerwarteten Ergebnisses (z. B. Screenshot der Webseite, wörtliche Fehlermeldungen, .pcap Dateien, o. ä.).
- Eine Beschreibung dessen, was Sie erwartet hätten.