



Kurs: CY-B-3 Sichere Programmierung

Dozent: Michael Heigl

Absichern eines terminalbasierten Online-Schachservers (Chess Online)

Autoren:

Sandro Schamberger (22102471) [33%]

Florian Hagengruber (22101608) [33%]

Christian Joiko (22111097) [33%]

Inhaltsverzeichnis

1. Einleitung.....	3
2. Funktionsweise der schwachen Version	5
2.1 Login und Menü	5
3.1 Datenbank.....	8
3.2 Passwort und Aktivierungscode	11
3.3 Kommunikation	15
3.4 Code Analysetools und Styleguides.....	17
3.5 Sonstige Schwachstellen	18
4. Beheben der Schwächen.....	19
4.1 Datenbank.....	19
4.2 Passwort und Aktivierungscode	21
4.3 Kommunikation	24
4.4 Künftige Fixes.....	25
5. Fazit.....	26
Literaturverzeichnis	27
Abbildungsverzeichnis	30

1. Einleitung

Die Digitalisierung betrifft momentan fast jeden Bereich unseres Lebens. Sei es nun am Arbeitsplatz, im Auto oder im eigenen Heim, die Digitalisierung ist mittlerweile omnipräsent. Doch so viele Vorteile der digitale Wandel auch bringt, so wird sie jedoch stets von einem schwerwiegenden Problem begleitet, dass leider noch nicht dieselbe Aufmerksamkeit erhält wie die Digitalisierung selbst. Bei dieser bössartigen Nebenwirkung handelt es sich um die Cyber Sicherheit oder besser gesagt der Mangel davon. Mit jeder digitalen Neuerung und jedem neuen Gerät bilden wir eine weitere Angriffsfläche, auf der wir von Cyberkriminellen attackiert werden können, weshalb diese Dienste bestmöglich geschützt werden müssen. Viele denken hierbei wahrscheinlich an kritische Infrastruktur wie Krankenhäuser oder Stromversorgung, wobei jedoch alle Aspekte hinreichend abgesichert werden müssen, auch wenn sie noch so unscheinbar sind.

Ein Beispiel hierfür kommt aus dem Bereich der Freizeitgestaltung, die Videospielindustrie. Doch warum ist Informationssicherheit hier so wichtig? Dank der Digitalisierung ist diese Freizeitaktivität mittlerweile weit verbreitet, sodass im Jahr 2021 bereits jeder Zweite in Deutschland ab 16. Jahren gelegentlich Videospiele spielt [1]. Allein 2021 wurden 6,17 Milliarden Euro für Videospielsoftware ausgegeben [2]. Somit entsteht ein sehr großes Risiko für Firmen, da Vorfälle in der IT-Sicherheit einen großen Reputationsschaden mit sich führen können, wodurch der Anteil der Firma an dieser Geldsumme drastisch sinken kann. Doch nicht nur die Anbieter und Produzenten selbst sind von Cyberangriffen betroffen, sondern auch die Konsumenten, welche ihre persönlichen Daten, ihre Zahlungsinformationen und ihr investiertes Vermögen in die Hände der Firmen geben. Werden Nutzerdaten nicht ausreichend geschützt, so könnten diese gestohlen werden und zu Schäden beim Nutzer führen.

Wie dringlich ein solcher Schutz gegen Hackangriffe wirklich ist, wurde den Unternehmen und Nutzern immer wieder aufs Neue verdeutlicht: 2011 gab es einen Angriff auf das Unternehmen Sony [3]. Durch das Eindringen einer unbefugten Person wurde das PlayStation-Network für 23 Tage abgeschaltet [4]. Da PSN-Spieler sich somit nicht mehr mit dem Netzwerk verbinden konnten, verzeichnete das Unternehmen einen Schaden von 171 Millionen US-Dollar [5]. Jedoch nahm nicht nur das Unternehmen Sony Schaden an dem Angriff, auch persönliche Daten der damals 77 Millionen Nutzer, darunter Namen, Anschrift, Geburtsdatum sowie Passwörter und Kreditkarteninformationen wurden entwendet [3].

Doch wie schwer ist es eigentlich, ein Videospiel mit all seinen verknüpften Diensten und Nutzerdaten gegen Cyberkriminelle und Hacker zu schützen? Um diese Frage zu beantworten, wird in diesem Projekt ein Server, mit dem Nutzer gegeneinander Schach spielen können, genauer unter die Lupe genommen. Dabei werden zuerst mögliche Schwachstellen analysiert und aufgrund ihres Gefahrenpotentials für Nutzer und Firma bewertet. Anschließend werden die erkannten Schwachstellen repariert, um eine möglichst sichere Version zu erstellen. Durch den Vergleich der beiden Versionen und die Aufzählung der möglichen Angriffsvektoren wird versucht, diese Frage zu beantworten.

2. Funktionsweise der schwachen Version

2.1 Login und Menü

Bevor ein Benutzer per „Chess Online“ Schach spielen kann, muss er mit dem Server eine Verbindung aufbauen. Dafür wird das Protokoll „Telnet“ verwendet, womit sich ein Client unter Eingabe der IP-Adresse des Servers und der Portnummer 8080 verbinden kann. Nachdem die Verbindung erfolgreich hergestellt wurde, wird dem Benutzer das Menü angezeigt. Hier wird der Spieler aufgefordert sich anzumelden oder zu registrieren, falls dieser noch keinen Account besitzt (siehe Abb. 1).

```
-Enter a move by giving the coordinates of the starting point and the goal point
-During a match you can either enter a legal Move or "--Help" for further commands
(1)Login   (2)Registration   (3)Exit
Please enter the number that corresponds to your desired menu:
```

Abbildung 1: Benutzer wird im Menü zum Login geführt

Für den Fall, dass der Benutzer noch kein Konto besitzt, kann er den Menüpunkt „Registration“ auswählen, um sich eines anzulegen. Für die Registrierung wird zuerst eine gültige Mailadresse eines Studenten oder Mitarbeiters der Technischen Hochschule Deggendorf benötigt. Die eingegebene E-Mail wird zur Kontrolle auf ihren Aufbau hin überprüft, ob sie eine Mailadresse der THD sein kann (siehe Abb. 2). Ist dies der Fall, so wird der Benutzer nach einem Passwort gefragt. Schlussendlich wird dem Nutzer dann mitgeteilt, dass ihm per E-Mail ein Aktivierungscode, bestehend aus vier Ziffern, gesendet wurde (siehe Abb. 3). Die Mailadresse, das Passwort und der Aktivierungscode werden außerdem zusammen mit einem automatisch generierten Nutzernamen in der Tabelle „Spieler“ der Datenbank abgespeichert (siehe Abb. 4).

```

Please enter the number that corresponds to your desired menu: 2
Enter your email address: evil.hacker@gmail.com
Your input was not a valid email address
Enter your email address: sandro.schamberger@stud.th-deg.de
Enter a new password: MyPassword

```

Abbildung 2: Registrierung des Accounts und Abweisung einer invaliden Mailadresse

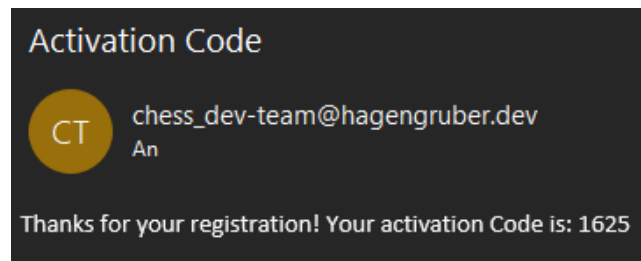


Abbildung 3: Empfang des Aktivierungscode



Abbildung 4: Modell der Datenbank

Sollte der Nutzer bereits ein Konto besitzen, so kann er den Menüpunkt „Login“ benutzen, um sich anzumelden. Dafür werden seine Mailadresse und sein Passwort benötigt. Ist dies seine erste Anmeldung, so wird er zudem noch nach seinem Aktivierungscode gefragt (siehe Abb. 5). Wurde dieser richtig eingegeben, so ist der Benutzer eingeloggt und kann nun die Funktionen des Servers benutzen. Zu diesen zählen das Ausloggen durch den Menüpunkt „Logout“, das Laden oder Starten eines Spiels gegen eine KI, dem Beitreten eines Online-Matches und dem Verlassen der Applikation. Im Zuge dieser Arbeit wird hauptsächlich der Vorgang eines Online-Matches betrachtet, da dieser Aspekt die meisten Risiken birgt.

```

Please enter the number that corresponds to your desired menu: 1
email address: sandro.schamberger@stud.th-deg.de
password: MyPassword
Enter your activation Code: 60877

```

Abbildung 5: Anmeldung des Nutzers mit Abfrage des Aktivierungscodes

2.2 Während eines Online-Matches

Entschließt sich ein eingeloggter Nutzer, dass dieser online eine Partie gegen einen anderen Spieler austragen möchte, so betritt der Spieler durch Eingabe des dazugehörigen Menüzeichens eine Warteschlange. Betritt ein weiterer Spieler die Warteschlange, so werden die Daten der beiden Kontrahenten in einer Queue gespeichert und ein Spiel wird gestartet. Nun können die Nutzer abwechselnd durch die Eingabe einer Zeichenkette, bestehend aus dem Anfangsfeld und dem Zielfeld, ihre Figuren bewegen (e.g. „G1E1“). Neben diesen Aktionen können die Spieler Befehle wie *--stats*, *--surrender* und *--draw* verwenden.

Eine Runde kann aus verschiedenen Gründen enden. Die verschiedenen Möglichkeiten hierfür sind das manuelle Beenden mittels der oben genannten Befehle *--surrender* und *--draw* (nur wenn sich beide Spieler für ein Remis einigen) oder das Besiegen des gegnerischen Königs

Am Ende der Runde wird das Ergebnis außerdem in der Datenbank vermerkt. Dazu zählen das Speichern des Endergebnisses in der Tabelle „Spiele“, sowie das Abändern der Statistiken der Spieler. Dabei wird den beiden Spielern je nach Situation der Wert des Attributs „siege“, „niederlagen“ oder „remis“ um eins erhöht und die ELO-Bewertung wird für beide Spieler neu kalkuliert (siehe Abb. 6) [6].

```
def recalculate_elo(self, victor_id, loser_id):
    victor_elo = int(self.database.fetch_general_data('elo', 'Spieler', 'WHERE id = ' + str(victor_id))[0][0])
    loser_elo = int(self.database.fetch_general_data('elo', 'Spieler', 'WHERE id = ' + str(loser_id))[0][0])
    elo_difference = max(victor_elo, loser_elo) - min(victor_elo, loser_elo)
    elo_difference = elo_difference / 400
    expected_value = 1 / (1 + 10**elo_difference)
    changed_elo = victor_elo + 20 * (1 - expected_value)
    elo_change = changed_elo - victor_elo
    self.database.add_elo(victor_id, elo_change)
    self.database.remove_elo(loser_id, elo_change)
```

Abbildung 6: Berechnung der ELO-Änderung

3. Analyse der Schwachstellen

Im folgenden Kapitel wird der Code der schwachen Version analysiert und die Funktionen des Programms werden auf Angriffsmöglichkeiten und Schwächen untersucht. Die Schwächen werden dafür in Gruppen zu ihrer jeweiligen Programmfunktion eingeteilt. Dabei werden, nach einer kurzen Einführung zur Schwachstelle, zuerst die betroffenen Codestellen genannt und der Grund für die Schwachstelle aufgezeigt. Danach wird ein möglicher Angriff mit seinen potenziellen Konsequenzen zur Veranschaulichung des Problems vorgestellt. Zuletzt werden die jeweilige Klassifizierung der CWE und eine Bepunktung des Schweregrads angegeben. Die Punkteverteilung verläuft nach den Richtlinien der CWE [7]. Zur Berechnung der Punktzahl wird der „Axelsec-Rechner“ verwendet.

3.1 Datenbank

Da „Chess Online“ zur Verwaltung der Nutzerdaten, der Spielhistorie und der Speicherstände eine Datenbank verwendet, sollte jedem Programmierer sofort die Gefahr einer SQL-Injection in den Sinn kommen. Die SQL-Injection ist nicht ohne Grund sowohl im Jahr 2020 als auch im Jahr 2022 auf Platz 3 der „CWE Top 25 Most Dangerous Software Weaknesses“ [8]. Die Möglichkeit einer SQL-Injection kann somit katastrophale Folgen haben, wie nachfolgend aufgezeigt wird.

Bei dem Projekt werden die meisten Datenbankbefehle unabhängig von Eingaben des Nutzers ausgeführt. Jedoch werden für manche Abfragen Nutzereingaben zwingend benötigt, genauer gesagt bei den Datenbankabfragen, welche während der Anmeldung und der Registrierung benötigt werden. Die Ursache für die SQL-Injection ist eine Kombination aus dem Verwenden von schwachen Softwarekonstruktionen, obwohl sicherere vorhanden wären und dem Fehlen der Input Validation. Ersteres bezieht sich hierbei auf die

Verwendung der Funktion *executescript()* und dem *%s-Platzhalter*. Des Weiteren wird die Datenbank unverschlüsselt und ohne Schutzvorrichtungen am Server abgelegt, was ebenfalls zu Problemen führen kann, wenn ein Angreifer Zugriff auf den Server erlangt.

Sollte ein Hacker die Datenbank per SQL-Injection angreifen wollen, so muss er lediglich die richtige Anzahl der Parameter für das SQL-Statement herausfinden, sodass dieses ausgeführt wird und nicht abgebrochen werden kann. Da dafür jedoch ein wenig Trial-and-Error genügt, kann der Angreifer schnell, einfach und, im Falle der Registrierung, ohne Account und funktionierende Mailadresse per SQL-Injection die Datenbank in vollem Umfang nach seinen Wünschen manipulieren oder sich bei anderen Accounts ohne Passwort anmelden. Einzig die Validierung der Mailadresse erschwert dem Angreifer das Herausfinden einer korrekten Eingabe, da seine Injection die Überprüfung des Domänenteils überstehen muss und dann bereits beim Überprüfen, ob diese Mailadresse bereits registriert ist, ausgeführt werden muss. Jedoch ist dies nicht allzu schwer und kann durch Umsteigen auf die anderen drei Eingabefelder sogar vollständig umgangen werden. Nachfolgend wird eine mögliche Eingabe für beide Vorgänge mit ihren Folgen für die Anwendung aufgezeigt, wobei ein Stern (*) für eine beliebige Texteingabe steht:

Registrierung		
<u>Feld</u>	<u>Eingabe</u>	<u>Konsequenzen</u>
E-Mail	*@th-deg.de	Löscht die Tabelle Spieler mit allen Accountdaten
Passwort	*, '*', '*'); DROP TABLE Spieler;--	
E-Mail	*' OR 1 = 1; UPDATE Spieler SET siege = 1000 WHERE nutzername = 'c.joiko';--@th-deg.de	Setzt die Anzahl der Siege des Spielers mit dem Nutzernamen „c.joiko“ auf 1000
Passwort	<i>Eingabe egal</i>	

Anmeldung		
<u>Feld</u>	<u>Eingabe</u>	<u>Konsequenzen</u>
E-Mail	<i>Eingabe egal</i>	Einloggen mit erstem registriertem Account
Passwort	* ' OR True;--	

Der gemeinsame CWSS-Score für alle Schwächen der Datenbank beträgt **73,9**. Eine detaillierte Version der Berechnung sowie die zur Berechnung herangezogenen CWEs befinden sich in der Datei *Berechnung CWSS-Score (Datenbank).xlsx* im Anhang.

CWSS Vector	
(TI:C,1/AP:P,0,9/AL:S,0,9/IC:L,0,9/FC:T,1/RP:N,1/RL:N,0,7/AV:V,0,8/AS:N,1/IN:A,1/SC:A,1/BI:h,0,9/DI:h,1/EX:H,1/EC:N,1/P:W,1)	

CWSS Score		Rating
Base Finding Subscore	86,4	
Attack Surface Subscore	0,90	
Environmental Subscore	0,95	
Final Score	73,9	High

Abbildung 7: CWSS-Score und CWSS-Vektor der CWE der Datenbank

3.2 Passwort und Aktivierungscode

Das Passwort ist wahrscheinlich eines der ältesten und am weitest verbreiteten Authentifizierungsverfahren. Sie sind einfach als Schutz zu implementieren und können trotz ihrer Simplität dennoch großen Schutz bieten, vorausgesetzt sie werden auch richtig verwendet. Noch immer verwenden fast 50% der Deutschen meistens Passwörter mit einer Länge von maximal 10 Zeichen [9]. In Kombination mit zu einfach gewählten Passwörtern wie „hallo123“ oder dem Notieren an unsicheren Orten verlieren sie relativ schnell jegliche Bedeutung beim Schutz der eigenen Daten, was nachfolgend auch bei „Chess Online“ aufgezeigt wird.

Das erste Problem entsteht bereits bei der Erstellung des Passwortes bei der Registrierung eines neuen Accounts. Das Passwort wird bei der Eingabe am Terminal angezeigt und kann somit bereits durch einen Blick auf den Bildschirm erlangt werden. Zudem wird der Nutzer nicht um eine Bestätigung des Passwortes gebeten, was in Kombination mit der fehlenden Möglichkeit zum Ändern des Passwortes bei einem Tippfehler schnell zu einem Problem führt. Doch die eingeschränkte Effektivität der Passwörter ist nicht nur Schuld des Benutzers, sondern auch des Entwicklers. Die Passwörter werden kein einziges Mal verschlüsselt und liegen permanent als Plaintext vor. Zudem wurde von Programmierern das Passwort für den E-Mail-Server im Quelltext der Datei mail.py vergessen (siehe Abb. 8).

```
self.USERNAME = "chess_dev-team@hagengruber.dev"  
self.PASSWORD = "halonthxitol36598#!/89gotls"
```

Abbildung 8: Hard-coded Credentials in mail.py

Neben den Passwörtern weisen auch die Aktivierungs-codes zum Freischalten eines neuen Accounts einen verheerenden Mangel auf. Die 4-stelligen Codes sind sehr einfach per Brute-Force-Attacke zu knacken, was aufgrund der fehlenden Limitierung der Eingabeversuche zu einem kompletten Verlust der Authentifizierungsfunktion des Aktivierungs-codes führt. Außerdem wird zur Generierung die kryptographisch unsichere Funktion *random.randint()* verwendet.

Wie bereits erwähnt kann der Aktivierungscode in der schwachen Version mit geringem Ressourcen- und Zeitaufwand durch Brute-Forcing herausgefunden werden. Dies ermöglicht einem Angreifer eine beliebige Anzahl an Accounts anzulegen, da die Prüfung der Echtheit einer E-Mail durch den Aktivierungscode verifiziert wird.

Um eine solche Brute-Force-Attacke durchzuführen, muss der Angreifer eine Anwendung schreiben, die eine automatische TCP-Verbindung zum Server aufbaut, sich anschließend versucht anzumelden und die Zahlen 1000 bis 9999 für den Aktivierungscode einsetzt. Hierbei ist zu beachten, dass bei einem falschem Aktivierungscode das Menü in der Funktion *get_menu_choice([...])* durch *self.get_menu_choice(self.view.get_menu_choice())* rekursiv aufgerufen wird (siehe in Anhang *Quellcode Ausschnitt get_menu_choice.py*).

Wenn mehr als 490 Versuche, den Aktivierungscode zu prüfen, fehlgeschlagen sind, stoppt der Server die Verbindung mit dem Client durch einen *RecursionError*. Dies muss der Angreifer abfangen und eine erneute Verbindung aufbauen, was Performanceeinbrüche nach sich zieht (siehe in Anhang *Quellcode brute-force-Attacke.py*). Die Zeit in Sekunden eines solchen Angriffes lässt sich approximativ in Abhängigkeit des Aktivierungs-codes wie folgt ermitteln:

$$\frac{x - 1000}{z} = y$$

Abbildung 9: Formel zur Berechnung der Brute-Force-Dauer des Aktivierungscode

Die Variable x steht hierbei für den Code, y für die Sekundenanzahl und z für die durchschnittliche Anzahl an Versuchen pro Sekunde. Beispielsweise kann man den Code 2316 in ca. 43 Sekunden knacken, wenn man von 30,45 Versuchen in der Sekunde ausgeht.

Durch das unkontrollierte Erstellen von Accounts kann z. B. versucht werden, die Performance der Datenbank und dadurch das Schutzziel Verfügbarkeit durch ständige Registrierungen zu beeinträchtigen. Auch erhöht sich die Gefahr einer DOS bzw. einer DDOS-Attacke, da der Angreifer z. B. mit einer großen Anzahl an Accounts gegen die KI spielen könnte. Hierbei ist der Rechenaufwand zu betrachten, der durch das Spielen mit der KI benötigt wird, da durch genügend Spieler der Server ausgelastet werden könnte und somit das Schutzziel Verfügbarkeit bedroht wird. Ein solcher Denial-of-Service-Angriff ist ähnlich wie die Brute-Force-Attacke aufgebaut: der Angreifer muss automatisiert so viele TCP-Verbindungen wie möglich zum Server aufbauen und diesen somit überlasten, was im nachfolgenden Angriff mittels Threads realisiert worden ist. Um die Effektivität des Angriffes zu steigern, spielt jeder Thread gegen die KI in einer Endlosschleife, was die Rechenlast des Servers zusätzlich steigern soll.

Folgende Hardware ist für den Angriff verwendet worden:

- Prozessor: Intel(R) Core(TM) i7-7700K 4.20 GHz
- RAM: 16 GB DDR4
- Betriebssystem: Windows 10 Home

Die Effizienz des Angriffes ist anhand der Zeit in Sekunden gemessen, die ein Spieler benötigt, um sich am Server anzumelden und die KI den ersten Zug macht, welche nachfolgend als Zug-dauer bezeichnet wird. Anhand der Datei *Auswirkung DOS Attacke*

Sprungweite 1.xlsx im Anhang kann man erkennen, dass schon 49 erzeugte Spieler die Zugdauer von ca. 5 Sekunden auf 120 Sekunden bringen und die CPU dauerhaft auslasten. Da im Laufe des Spieles die Anwendung immer mehr Züge zu berechnen hat, steigt die Antwortzeit der KI in der Mitte des Spieles bei nur einem Nutzer schon auf ca. 5 Minuten. Bei 49 aktiven Spielern ist daher eine Antwortzeit von dutzenden Minuten bis zu Stunden nicht auszuschließen.

Auch ist eine Statistik von 150 Spielern erhoben worden (siehe in Anhang *Auswirkung DOS Attacke Sprungweite 20.xlsx*). Hierbei ist zu beachten, dass die höchste Zug-dauer bei fast 700 Sekunden liegt, was das Spielen gegen die KI enorm beeinträchtigt.

Ein geplanter Absturz des Servers ist dennoch nicht erzielt worden, jedoch stört die DOS-Attacke den Spielfluss außerordentlich und somit wird die Verfügbarkeit ebenso beeinträchtigt.

Der gemeinsame CWSS-Score für alle Passwort-bezogenen Schwächen beträgt **58,2**. Eine detaillierte Version der Berechnung sowie die zur Berechnung herangezogenen CWEs befinden sich in der Datei *Berechnung CWSS-Score (Passwort).xlsx* im Anhang.

CWSS Vector		
(TI:h,0,9/AP:RU,0,7/AL:A,1/IC:N,1/FC:T,1/ RP:n,1/RL:N,0,7/AV:V,0,8/AS:N,1/ IN:A,1/SC:A,1/ BI:M,0,6/DI:h,1/EX:H,1/EC:L,0,9/P:W,1)		

CWSS Score		Rating
Base Finding Subscore	90	
Attack Surface Subscore	0,90	
Environmental Subscore	0,72	
Final Score	58,3	Medium

Abbildung 10: CWSS-Score und CWSS-Vektor der CWE der Passwörter

3.3 Kommunikation

Tagtäglich kommunizieren Milliarden von Geräten mittels verschiedenster Protokolle über lokale und globale Netzwerke miteinander. Dabei tauschen sie unterschiedlichste Informationen aus: von Textnachrichten über Geräteinformationen, bis hin zu Authentifizierungsmethoden und Passwörter. Diese Daten müssen geschützt werden, da ein Angreifer mit Zugriff auf das jeweilige Netzwerk die Kommunikation einfach mithören und manipulieren könnte. Unsichere Protokolle wie FTP oder HTTP gilt es daher gegen solche Angriffe und Verstöße der Schutzziele wie Vertraulichkeit oder Integrität zu wappnen.

Gleiches gilt für die Kommunikation zwischen dem „Chess Server“ und einem Client: in der schwachen Version des Spieles findet der Datenaustausch mittels TCP statt, einem Protokoll, welches gesendete Daten nicht verschlüsselt oder vor Manipulation schützt. Sensible Informationen wie E-Mail-Adressen oder Passwörter können abgefangen und ausgelesen werden. Dies könnte ein Angreifer mittels einer Man-in-the-Middle-Attacke erreichen. Hierzu müsste er nur im selben Netzwerk wie der Server oder der Client sein und die TCP-Pakete mit z. B. Wireshark auslesen. Dadurch erhält der Hacker sämtliche Nachrichten, die der Server und der Spieler miteinander austauschen und kann somit Nutzernamen oder Passwörter aufzeichnen. Wie in Abbildung 11 und Abbildung 12 zu sehen ist, wurde erfolgreich die E-Mail „florian.hagengruber@stud.th-deg.de“ sowie das Passwort „aPassword“ während des Anmeldevorgangs ausgelesen.

0000	dc a6 32 67 b2 d6 d0 57 7b 28 78 73 08 00 45 00	..2g...W {(xs...E.
0010	00 4c c1 45 40 00 80 06 b5 33 c0 a8 01 72 c0 a8	.L.E@... .3...r..
0020	01 70 04 a6 30 39 4f 58 64 93 4c c7 03 94 50 18	.p..090X d.L...P.
0030	01 fe 01 1c 00 00 66 6c 6f 72 69 61 6e 2e 68 61fl orian.ha
0040	67 65 6e 67 72 75 62 65 72 40 73 74 75 64 2e 74	gengrube r@stud.t
0050	68 2d 64 65 67 2e 64 65 0d 0a	h-deg.de ..

Abbildung 11: Mittels Wireshark abgefangene Mailadresse

0000	dc a6 32 67 b2 d6 d0 57 7b 28 78 73 08 00 45 00	..2g...W {(xs...E.
0010	00 33 c1 47 40 00 80 06 b5 4a c0 a8 01 72 c0 a8	·3·G@... ·J...r...
0020	01 70 04 a6 30 39 4f 58 64 b7 4c c7 03 9e 50 18	·p...090X d·L...P·
0030	01 fe dc 81 00 00 61 50 61 73 73 77 6f 72 64 0daP assword·
0040	0a	·

Abbildung 12: Mittels Wireshark abgefangenes Passwort

Jedoch ist nicht nur das Mitlesen der Pakete möglich. Ein Angreifer könnte mittels TCP-Spoofing Daten mit der IP-Adresse eines verbundenen Clients an den Server senden und so seine Identität fälschen. Hierbei ist lediglich zu beachten, dass die Sequenznummer der gesendeten TCP-Pakete erraten werden muss, was jedoch mittels Hochzählen der Sequenznummer und Brute-Force zu bewerkstelligen wäre. Durch das Fälschen der Identität hält der Server den Angreifer fälschlicherweise für den verbundenen Client und somit kann dieser im Namen des Clients die Funktionalitäten bedienen.

Der gemeinsame CWSS-Score für alle Schwächen der Kommunikation zwischen Clients und Server beträgt **63,7**. Eine detaillierte Version der Berechnung sowie die zur Berechnung herangezogenen CWEs befinden sich in der Datei *Berechnung CWSS-Score (Kommunikation).xlsx* im Anhang.

CWSS Vector		
(TI:h,0,9/AP:RU,0,7/AL:A,1/IC:N,1/FC:T,1/ RP:n,1/RL:N,0,7/AV:V,0,8/AS:N,1/ IN:T,0,9/SC:A,1/ BI:M,0,6/DI:h,1/EX:H,1/EC:N,1/P:W,1)		

CWSS Score		Rating
Base Finding Subscore	90	
Attack Surface Subscore	0,89	
Environmental Subscore	0,80	
Final Score	63,7	Medium

Abbildung 13: CWSS-Score und CWSS-Vektor der CWE der Passwörter

3.4 Code Analysetools und Styleguides

Als weitere Kontrollinstanz neben dem manuellen Analysieren des Codes wurden verschiedene statische Code Analysetools verwendet, um mögliche Schwachstellen im Programm zu entdecken. Folglich werden die Ergebnisse der Analysen mit dem Namen des verwendeten Tools sowie die Fehlerraten der „False-Negative“ Fälle aufgelistet.

Tool	Gefundene CEWs	„False-Negative“-Rate
Prospector	<ul style="list-style-type: none">• CWE-259: Use of Hard-coded Password• CWE-798: Use of Hard-coded Credentials	98,11% (104 nicht gefundene CEWs)
Bandit	<ul style="list-style-type: none">• CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')• CWE-798: Use of Hard-coded Credentials• CWE-259: Use of Hard-coded Password• CWE-330: Use of Insufficiently Random Values	96,23% (102 nicht gefundene CEWs)
Snyk	/	100% (106 nicht gefundene CWEs)

Abbildung 14: Tabelle der Analyse der Code-Analysis-Tools

Auch wurden die aktuellen Styleguide Richtlinien nach PEP 8 eingehalten. Dabei wurde der Code mit dem Tool PyLint analysiert und dementsprechend verbessert. In der starken Version der Anwendung konnte ein PyLint-Score von **9,74** erreicht werden.

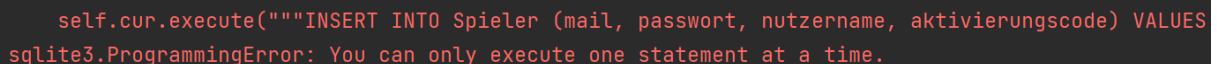
3.5 Sonstige Schwachstellen

Eine weitere zu betrachtende Schwachstelle ist die Manipulation des Arbeitsspeichers. Wenn ein Angreifer Zugriff auf den Server hat und dabei Tools wie z. B. die Cheat Engine [10] einsetzt, hat dieser uneingeschränkt Zugriff auf den Arbeitsspeicher. Dadurch wird das Hinzufügen, Verändern oder Löschen des RAMs und dadurch auch der Variablen und des gespeicherten Codes ermöglicht. Eine mögliche Konsequenz dessen wäre das unautorisierte Ändern von Statuswerten der Benutzer, wie etwa die ELO. Auch kann schädlicher Code initiiert werden, was dem Angreifer z.B. das Erstellen einer Reverse Shell erlaubt.

4. Beheben der Schwächen

4.1 Datenbank

Um die in 3.1 angesprochene SQL-Injection zu beheben, sollten zuerst die verwendeten unsicheren Softwarekonstruktionen durch sichere Alternativen ersetzt werden. Die erste Änderung ist das Ersetzen der Funktion *executescript()* durch *execute()*. Der Unterschied zwischen diesen beiden Funktionen ist, dass *execute()* nur einen SQL-Befehl ausführen kann, während *executescript()* mehrere Statements verarbeitet [11]. Da unsere Funktionen zur Manipulation der Datenbank immer nur ein Statement ausführen müssen, ist dies ein sehr einfacher Schutz vor angehängten, unerwünschten SQL-Statements des Angreifers. Nach Anwendung des Fixes sind die beiden Injections, welche zuvor während der Registrierung möglich waren, nicht mehr möglich (siehe Abb. 15).



```
self.cur.execute("""INSERT INTO Spieler (mail, password, nutzername, aktivierungscode) VALUES  
sqlite3.ProgrammingError: You can only execute one statement at a time.
```

Abbildung 15: Fehlermeldung nach Ausführung der ersten SQL-Injection

Leider schützt diese Verbesserung noch nicht vor dem Login ohne Passwort, weshalb eine weitere Verbesserung vorgenommen werden muss. Diese betrifft die Platzhalter in den SQL-Statements, welche später durch die Eingaben des Nutzers ersetzt werden. Die schwache Version verwendet momentan *%s-Platzhalter*, welche laut der Dokumentation des sqlite3-Moduls in Python unsicher und anfällig für SQL-Injections sind. Als Verbesserung wird hierfür die Verwendung der moduleigenen *?-Platzhalter* vorgeschlagen, welche nach Einbindung die dritte SQL-Injection verhindern sollten [11]. Da dies jedoch nicht der Fall ist, wird zusätzlich der Input des Users auf maliziöse Absichten überprüft, bevor er verarbeitet wird. Um dies zu erreichen, wird ein Array mit verbotenen Zeichen eingefügt und über dieses iteriert, ob in der eingegebenen Mailadresse ein verbotenes

Zeichen vorhanden ist (siehe Abb. 16). Es wird hierbei nur die E-Mail überprüft, da invalide Zeichen im Passwort durch das vorherige Hashen eliminiert werden. Um die SQL-Injection zu verhindern, ohne den zulässigen Zeichensatz für Passwörter unnötig einzuschränken wird sich im Projekt auf vier Sonderzeichen beschränkt (siehe Abb. 17). Neben dem Blacklisting wurde außerdem noch ein Whitelisting Ansatz zur Validation der Benutzereingaben implementiert. Dieses Vorgehen gewährleistet aktiv und sicher den Schutz gegen SQL-Injection.

```
for c in self.forbidden:
    if c in mail:
        return "Wrong Credentials. Please Try again"
```

Abbildung 16: Überprüfung der Mail auf invalide Zeichen

```
self.forbidden = ['"', "--", "'", ";"]
```

Abbildung 17: Definition der verbotenen Zeichen

Gegen die ungesicherte und unverschlüsselte Ablage der Datenbank sollte ebenfalls vorgegangen werden. Dazu könnte beispielsweise SQLCipher verwendet werden [12]. Zur weiteren Absicherung sollte ein regelmäßiges Backup der Datenbank gemacht werden, damit im Falle der ersten beiden SQL-Injections der Schaden wieder behebbar wäre und die Integrität nicht vollständig verloren geht. Diese beiden Fixes wurden jedoch aus Zeitgründen und der höheren Relevanz und Gefährlichkeit der SQL-Injection nicht ins Programm eingebaut. Außerdem sind Mutexe bei der Verbindung mit der Datenbank hinzugefügt worden. Dies beugt Race Conditions durch das Verhindern eines gleichzeitigen Zugriffes auf die Datenbank vor.

4.2 Passwort und Aktivierungscode

Für die Behebung der in 3.2 angesprochenen Schwächen wurden mehrere einzelne Maßnahmen durchgeführt, um die Passworteingabe und das Senden eines Aktivierungscodes sicherer zu gestalten.

Zuerst wurde die Validation der Eingaben genauer betrachtet. Da der längste im Spiel einzugebende Befehl, **--Surrender**, 12 Zeichen lang ist, wurde die Länge aller Eingaben, außer des Passwortes und der Emailadresse, auf 12 Zeichen beschränkt. Somit gibt es weniger Möglichkeiten einen Angriff durchzuführen.

```
def __init__(self):  
    self.min_len = 13  
  
def check_input_length(self, user_input):  
    """returns bool if the length is ok"""  
    return len(user_input) < self.min_len
```

Abbildung 18: Input Validation in security.py

Entsprechende Eingaben wurden auf die zu verarbeitenden Datentypen gecastet, um fehlerhafte Benutzereingaben weiter zu vermeiden.

Da keine verfügbare Passwortrichtlinie den Anforderungen an das Projekt entsprochen hat wurde eine Passwortrichtlinie nach den Vorgaben des Bundesamts für Sicherheit in der Informationstechnik (BSI) [13] erstellt. Somit sollte das Passwort aus mindestens 10 und maximal 20 Zeichen bestehen. Weiterhin müssen jeweils 2 Großbuchstaben, Kleinbuchstaben, Zahlen und Sonderzeichen verwendet werden. Auch die Größe des Aktivierungscodes wurde auf 10 Byte erweitert. Dabei wurde die Funktion *os.urandom()* benutzt.

In Hinblick auf die erfolgreiche Durchführung einer SQL-Injection wurden die in Abbildung 17 unter *forbidden* genannten Zeichen in die Passwortrichtlinie übernommen, für ein Passwort verboten und führen direkt zu der erneuten Aufforderung ein valides Passwort einzugeben.

```
def __init__(self):
    self.upper = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
                  'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
    self.lower = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
                  'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
    self.numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    self.specials = ['!', '?', '$', '%', '&', '#', '@']
    self.allowed_mail_chars = self.upper + self.lower + self.numbers + ['@', '-', '.']
    self.forbidden = ['"', "--", "'", ";"]
    self.length = {'min': 10, 'max': 20}

def check_password(self, password):...

def check_password_guideline(self, characters, password):
    """returns bool whether the password complies with the policy"""
    if characters['upper'] >= 2 and characters['lower'] >= 2 \
        and characters['nums'] >= 2 and characters['special'] >= 2:
        return self.length['min'] <= len(password) <= self.length['max']

    return False
```

Abbildung 19: Passwortrichtlinie in security.py

Eine weitere Schutzmaßnahme bildet die Maskierung der Passworteingabe. Durch die Bibliothek *getpass* kann die Eingabe des Passwortes ganz ausgeblendet werden. Hierdurch wird ein Ausspionieren des Passwortes weiter eingeschränkt.

```
elif 'password' in message.lower():
    password = getpass('')
    self.conn.sendall(password.encode())
```

Abbildung 20: getpass zur Maskierung des Passworts

Um Fehler während der Registrierung zu verhindern, wird das Passwort doppelt abgefragt. Somit kann eine falsche Eingabe des Passwortes vermieden werden.

```
def __init__(self):
    self.argon = PasswordHasher(time_cost=16, memory_cost=2 ** 15,
                                parallelism=2, hash_len=32, salt_len=16)

def hash(self, user_input):
    """Returns the hash from the user input"""
    return self.argon.hash(user_input)

def verify(self, user_hash, user_input):
    """Returns bool if the user input is equal from the Hash in database"""
    try:
        return self.argon.verify(user_hash, user_input)
    except argon2.exceptions.VerifyMismatchError:
        return False
```

Abbildung 21: Argon2-Hashfunktion in security.py

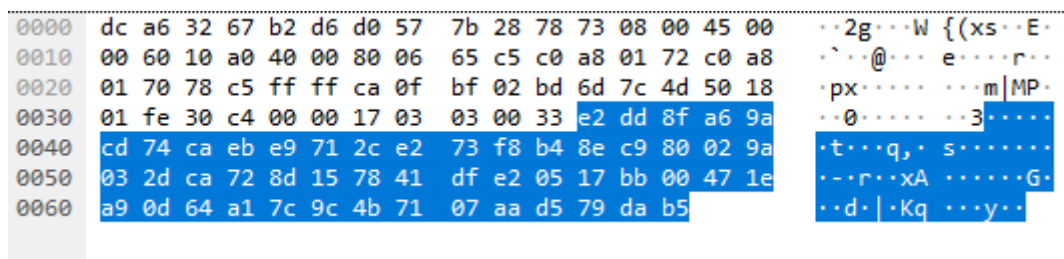
Zum Schutz vor Brute-Force-Attacken gegen Passwörter und Aktivierungs-codes wurde die maximale Anzahl an Eingabeversuchen auf 3 beschränkt. Sollte der Benutzer beim Login entweder das Passwort oder den Aktivierungscode zu oft falsch eingeben, wird der Nutzer gesperrt. Um wieder freigeschaltet zu werden, muss er sich an das Entwicklerteam wenden.

Zuletzt werden die Passwörter und Aktivierungs-codes mit dem im Unterricht vorgestellten Argon2-Hashverfahren gehasht. Dieses Verfahren hat den Vorteil, dass man verschiedene Einstellungen tätigen und dem Hash einen Salt mitgeben kann. Mit der Funktion *verify()* können die in der Datenbank gespeicherten Passwörter sehr einfach verglichen werden.

Auch die Anmeldedaten des SMTP-Servers in der *mail.py* Datei wurden in das Verzeichnis *certs* als JSON-File ausgelagert. Vorausgesetzt dieser Ordner ist Zugangsbeschränkt wurde so das Plaintext-Problem somit behoben.

4.3 Kommunikation

Um die in 3.3 beschriebene Schwachstelle, das unverschlüsselte Senden von Informationen zu beheben, wurde ein TLS-Wrapper verwendet. Dieser Ver- und Entschlüsselt die TCP-Pakete mittels eigen signierter SSL-Zertifikate. Hierbei ist zu beachten, dass sowohl Server als auch jeder Client ein von derselben Zertifizierungsstelle signiertes Zertifikat, sowie einen Schlüssel haben muss. Die Geheimhaltung dieser Zertifikate und Schlüssel ist Voraussetzung für die sichere Kommunikation. Wenn nun ein Angreifer eine Man-in-the-Middle-Attacke ausführen möchte, sieht er zwar die versendeten Pakete, die Payload ist jedoch verschlüsselt (siehe Abb. 22).



The image shows a Wireshark packet capture of a TLS-encrypted packet. The packet is displayed in hexadecimal and ASCII. The hexadecimal data is as follows:

Offset	Hex Data
0000	dc a6 32 67 b2 d6 d0 57 7b 28 78 73 08 00 45 00
0010	00 60 10 a0 40 00 80 06 65 c5 c0 a8 01 72 c0 a8
0020	01 70 78 c5 ff ff ca 0f bf 02 bd 6d 7c 4d 50 18
0030	01 fe 30 c4 00 00 17 03 03 00 33 e2 dd 8f a6 9a
0040	cd 74 ca eb e9 71 2c e2 73 f8 b4 8e c9 80 02 9a
0050	03 2d ca 72 8d 15 78 41 df e2 05 17 bb 00 47 1e
0060	a9 0d 64 a1 7c 9c 4b 71 07 aa d5 79 da b5

The ASCII representation on the right shows the corresponding characters, with many being non-printable (represented by dots) due to the encryption. The packet structure includes a TLS record header (0000-000f) and encrypted data (0010-0060).

Abbildung 22: Mittels Wireshark abgefangenes Paket mit verschlüsselten Daten

Auch das TCP-Sniffing und die Manipulation von Paketen werden durch TLS behoben, da das Protokoll einen Schutz der Authentifikation und Integrität bietet.

4.4 Künftige Fixes

Einige schwerwiegende Schwächen wurden bereits analysiert und gefixt, jedoch sind nach wie vor Schwachstellen im Programm zu finden, die in künftigen Versionen der Anwendung behoben werden. So müssen die Zertifikate, Schlüssel und SMTP-Zugangsdaten durch Zugriffsbeschränkungen vor unautorisierten Personen geschützt werden. Durch diese Maßnahme stellt man die Vertraulichkeit sowie Integrität der Kommunikation zwischen den Teilnehmern sicher und schützt zusätzlich den E-Mail-Account vor unbefugten Zugriffen. Auch der Server muss zugriffsbeschränkt sein, damit Angreifer keinen Zugriff auf die Quelldateien und die Datenbank haben. Außerdem wird dadurch die Manipulation des Arbeitsspeichers verhindert.

Auch wird in zukünftigen Updates der Anwendung ein Logger implementiert werden, der sowohl Login-Versuche, Registrierungen und TCP-Verbindungen, als auch noch mögliche Fehler im Programm aufzeichnen soll. Dadurch erhält man einen Überblick über Indikatoren für Angriffe, die Informationen helfen aber auch für das Debugging. Die Dateien, die dabei erzeugt werden, sollten nach Möglichkeit auf separaten Systemen gespeichert werden und müssen auch gegen unautorisierte Zugriffe geschützt werden.

Eine weitere Schwäche, die noch behoben werden muss, ist die Integritätsprüfung der Spielstände beim Laden eines Matches gegen die KI. Wenn ein Angreifer Zugriff auf die Datenbank bekommt, kann man die Spielstände beliebig manipulieren und sogar Schadcode in das Programm initiieren. Durch Prüfung der Integrität dieser Speicherstände wird eine solche Manipulation bzw. Angriff verhindert.

5. Fazit

Schlussendlich lässt sich sagen, dass das Absichern eines Servers einige Probleme mit sich bringt. Jede Komponente, jedes Feature und auch die Kommunikation zwischen den Komponenten, welche in das Programm verbaut wird, verbessern das Projekt nicht nur, sondern fügen zugleich einen Angriffsvektor für Angreifer hinzu, was sie zu einer Art „zweischneidigem Schwert“ macht. Doch auch wenn bereits kleine Softwareprojekte wie „Chess Online“ schon unzählige Fehlerquellen beinhalten, welche meistens katastrophale Folgen haben können, so lässt sich der Hauptteil der Fehler bereits durch kleine und unkomplizierte Fixes drastisch ihrer Schwere verringern oder gar komplett eliminieren. Somit lässt sich feststellen, dass die Sicherheit bei der Erstellung von Software stets eine hohe Priorität darstellt.

Literaturverzeichnis

- [1] Bitkom (2022): „Anteil der Computer- und Videospieler in Deutschland in den Jahren 2013 bis 2022“ auf Statista;
Zugriff: 13. Dezember 2022;
<https://de.statista.com/statistik/daten/studie/315860/umfrage/anteil-der-computerspieler-in-deutschland/>

- [2] game.de (2022): „Umsatz im Markt für Computer- und Videospiele (ohne Hardware) in Deutschland von 2009 bis 2021 (in Millionen Euro)“ auf Statista;
Zugriff: 13. Dezember 2022;
<https://de.statista.com/statistik/daten/studie/317808/umfrage/umsatz-im-markt-fuer-computer-und-videospiele-in-deutschland/>

- [3] heise.de (2011): „Welcome Back PSN: The Winners“; Zugriff: 14. Dezember 2022;
<https://kotaku.com/welcome-back-psn-the-winners-5804318>

- [4] kotaku.com (2011): „Angriff auf Playstation Network: Persönliche Daten von Millionen Kunden gestohlen“;
Zugriff: 14. Dezember 2022;
<https://www.heise.de/newsticker/meldung/Angriff-auf-Playstation-Network-Persoенliche-Daten-von-Millionen-Kunden-gestohlen-1233136.html>

- [5] pcmag.com (2011): „PlayStation Hack to Cost Sony \$171M; Quake Costs Far Higher“;
Zugriff: 14. Dezember 2022;
<https://www.pcmag.com/archive/playstation-hack-to-cost-sony-171m-quake-costs-far-higher-264796>

- [6] Wikipedia (o.J.): “ELO-Zahl: Anpassung nach einer Partie“;
Zugriff: 13. Dezember 2022;
https://de.wikipedia.org/wiki/Elo-Zahl#Anpassung_nach_einer_Partie

- [7] The MITRE Corporation (2014): „Scoring CWEs“;
Zugriff: 19. Dezember 2022;
https://cwe.mitre.org/cwss/cwss_v1.0.1.html

- [8] The MITRE Corporation (2022): „2022 CWE Top 25 Most Dangerous Software Weaknesses“;
Zugriff: 13. Dezember 2022;
https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html

- [9] Web.de (2022): „Wie lang sind Ihre am häufigsten verwendeten Passwörter?“ auf Statista;
Zugriff: 13. Dezember 2022;
<https://de.statista.com/statistik/daten/studie/988439/umfrage/laenge-von-passwoertern-in-deutschland/>

- [10] Cheat Engine
Zugriff: 19. Dezember 2022;
<https://www.cheatengine.org/>

- [11] Python Software Foundation (o.J.): “sqlite3”-Dokumentation
Zugriff: 13. Dezember 2022;
<https://docs.python.org/3/library/sqlite3.html>

- [12] Zetetic (o.J.): About SQLCipher
Zugriff: 13. Dezember 2022;
<https://www.zetetic.net/sqlcipher/about/>

[13] Bundesamt für Informationsschutz (o.J.): Sichere Passwörter erstellen

Zugriff: 21. Dezember 2022;

https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehlungen/Accountschutz/Sichere-Passwoerter-erstellen/sichere-passwoerter-erstellen_node.html

Abbildungsverzeichnis

Abbildung 1: Benutzer wird im Menü zum Login geführt.....	5
Abbildung 2: Registrierung des Accounts und Abweisung einer invaliden Mailadresse	6
Abbildung 3: Empfang des Aktivierungscode	6
Abbildung 4: Modell der Datenbank	6
Abbildung 5: Anmeldung des Nutzers mit Abfrage des Aktivierungscode.....	6
Abbildung 6: Berechnung der ELO-Änderung	7
Abbildung 7: CWSS-Score und CWSS-Vektor der CWE der Datenbank	10
Abbildung 8: Hard-coded Credentials in mail.py.....	11
Abbildung 9: Formel zur Berechnung der Brute-Force-Dauer des Aktivierungscode.....	13
Abbildung 10: CWSS-Score und CWSS-Vektor der CWE der Passwörter.....	14
Abbildung 11: Mittels Wireshark abgefangene Mailadresse	15
Abbildung 12: Mittels Wireshark abgefangenes Passwort.....	16
Abbildung 13: CWSS-Score und CWSS-Vektor der CWE der Passwörter.....	16
Abbildung 14: Tabelle der Analyse der Code-Analysis-Tools	17
Abbildung 15: Fehlermeldung nach Ausführung der ersten SQL-Injection	19
Abbildung 16: Überprüfung der Mail auf invalide Zeichen.....	20
Abbildung 17: Definition der verbotenen Zeichen.....	20
Abbildung 18: Input Validation in security.py	21
Abbildung 19: Passwortrichtlinie in security.py	22
Abbildung 20: getpass zur Maskierung des Passworts.....	22
Abbildung 21: Argon2-Hashfunktion in security.py	23
Abbildung 22: Mittels Wireshark abgefangenes Paket mit verschlüsselten Daten	24