Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Konzeption und Umsetzung eines web-basierten Schulungssystems aus dem Bereich IT-Security

Conception of a web-based training system within the area of IT security

Bachelorarbeit

im Studiengang Informatik

von

Oscar Altrogge

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: M. Sc. Wasja Brunotte

Hannover, 14.08.2020

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 14.08.2020

Oscar Altrogge

Zusammenfassung

Die Verwendung von schützenswerten Daten ist allgegenwärtig, seien es Bankdaten beim Einkaufen, persönliche Korrespondenz, empfindliche Gesundheitsdaten, vertrauliche Geschäftsdokumente oder die Integrität immer stärker vernetzter Fahrzeuge. Security by obscurity stellt hier keinen Schutz vor Angriffen und den damit einhergehenden Verlusten dar. Durch zunehmende rechtliche Bestimmungen wie beispielsweise die DSGVO und einen höheren Anspruch auf Seiten der Endnutzer, steigen die Anforderungen an die Umsetzung von IT-Security. Folglich steigen auch die Anforderungen an die IT-Security-Kompetenzen der Entwickler. Universitäten und Firmen haben ein Interesse daran, ihre Studenten und Mitarbeiter im Bereich IT-Security erfolgreich zu schulen und fortzubilden, um diesen Ansprüchen gerecht zu werden. Den Teilnehmern fehlt im Rahmen von Schulungen oft die Möglichkeit, an praktischen Übungen zu arbeiten, da gerade die Umsetzung der Umgebungen zum Experimentieren aufwendig ist. Doch gerade die praktische Anwendung verspricht einen höheren Wissensgewinn. In dieser Arbeit wird eine Plattform konzipiert und prototypisch umgesetzt, die versucht genau diese Lücke zu schließen. Die entstandene Software ermöglicht es Teilnehmern, unter Verwendung ihrer eigenen Hardware und ohne zusätzliche Installationen, an interaktiven Übungen und Experimenten begleitend zur Schulung teilzunehmen. Für Trainer bietet sie zudem den Vorteil einer einmaligen Vorbereitung der Aufgaben, unabhängig der Anzahl der Teilnehmer oder ihrer Ausstattung. Zur Demonstration der Möglichkeiten der Plattform werden zudem zwei Lernmodule entwickelt.

Abstract

Conception of a web-based training system within the area of IT security

The use of sensitive data is omnipresent, be it banking data while shopping, personal correspondence, sensitive health data, confidential business documents or the integrity of increasingly connected vehicles. Security by obstructiy does not provide any protection against attacks and the losses associated with them. Due to increasing legal obligations, such as the GDPR, and higher demands on the part of the end users, the requirements for the implementation of IT security are rising. Consequently, the demands on the IT security competencies of the developers are increasing as well. Universities and companies have an interest in training and educating their students and employees in the field of IT security to meet these requirements. The participants often lack the possibility to work on practical exercises in the context of training courses, because the implementation of the environments for experimentation is particularly complex. But especially the practical application promises a higher knowledge gain. In this thesis, a platform is designed and prototypically implemented, which tries to close exactly this gap. The resulting software enables participants to perform interactive exercises and experiments accompanying the training, using their own hardware without additional installations. For trainers, it also offers the advantage of a one-time preparation of the training, independent of the number of participants or their equipment. In addition to the platform, two learning modules were also developed to demonstrate the possibilities.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Problemstellung

IT-Security spielt eine zunehmend wichtiger werdende Rolle. Durch zunehmende rechtliche Bestimmungen wie beispielsweise die Datenschutz-Grundverordnung (DSGVO) und einen höheren Anspruch auf Seiten der Endnutzer, steigen die Anforderungen an die Umsetzung von IT-Security. Durch Schulungen wird versucht, Entwickler auf die aktuelle Bedrohungslage und potentielle Schwachstellen aufmerksam zu machen. Nicht nur aus unternehmerischer Sicht ist es dabei wichtig, den Wissensgewinn der Schulungen zu optimieren [20, 23, 22].

Bereits vor hunderten von Jahren befassten sich Philosophen mit dem Ansatz, dass Lernen stark mit der praktischen Anwendung verbunden ist. So findet sich im Konfuzianismus sinngemäß der folgende Gedanke:

> “Wenn ich höre, vergesse ich;
> Wenn ich sehe, erinnere ich;
> Wenn ich mache, verstehe ich;” *nach [31]*

Auch Untersuchungen aus dem Bereich der Lernforschung beschäftigen sich mit diesem Ansatz. Der Einsatz von Multimedia-Inhalten wird dort ebenfalls als Unterstützung beim Verstehen und Lernen neuer Inhalte diskutiert. Dies wurde in der Vergangenheit in unterschiedlichen Arbeiten untersucht [37, 6, 26, 15].

Interaktive Inhalte, die begleitend zu Schulungen auch Aufgaben und Experimente anbieten, sollten folglich als sinnvolle Erweiterung angeboten werden, um dieses Potential auszuschöpfen. Teilnehmern von Schulungen bleibt derzeitig diese Möglichkeit oft verwehrt, da eine Vorbereitung der

Aufgaben und eine Anpassung auf den Teilnehmer im Rahmen einer Schulung einen hohen Mehraufwand bedeutet. Eine Fragmentierung innerhalb der Teilnehmer bezüglich ihres Zugangs zu identischer, technischer Ausstattung trägt ihr Übriges dazu bei.

1.2 Zielsetzung

Eine Lösung für dieses Problem wäre die Bereitstellung einer Plattform, die parallel zu den Schulungsinhalten auch interaktive Beispiele und Programmieraufgaben zur Verfügung stellt, ohne Vorbedingungen an die Teilnehmer oder deren Software-Umgebung zustellen. Hierdurch könnte der Vorbereitungsaufwand für praktische Aufgaben reduziert werden, da die Individualisierung der Aufgaben durch die Plattform automatisiert werden könnte. Zudem würden Unterschiede in den Geräten der Teilnehmer durch einen externen und standardisierten Zugang wegfallen. Dies ermöglicht es, mit einmaligem Aufwand praktische Beispiele in einer beherrschbaren Umgebung bereitzustellen.

Konkret soll hier eine web-basierte Plattform konzipiert und umgesetzt werden, die dem Nutzer über den vorinstallierten Browser die Möglichkeit gibt, einen praktischen Teil begleitend zu einer Schulung zu absolvieren. Dies soll geschehen ohne dabei gesondert Rücksicht auf das genutzte Betriebssystem, User-Rechte, zusätzliche Software oder den technischen Wissensstand des Teilnehmers nehmen zu müssen. Die Plattform soll dem Teilnehmer sowohl Aufgaben als auch komplexe IT-Umgebungen bereitstellen, die es ihm ermöglichen, selbstbestimmt IT-Security-Inhalte zu erlernen. Eine Installation von zusätzlicher Software soll dabei weitgehend, falls möglich vollständig, vermieden werden. Schulungen werden von Trainern veranstaltet, welche auch die Inhalte dieser Schulungen kuratieren. Die entstehende Plattform soll dem Trainer die Möglichkeit geben, mit einmaligem Aufwand die Inhalte seiner Schulung anzubieten und bestehende Inhalte mit interaktiven Elementen zu erweitern.

Darüber hinaus sollen zur Demonstration der Möglichkeiten für Trainer und Teilnehmer exemplarisch zwei Module entstehen, die verschiedene Aspekte der Plattform nutzen und hervorheben. Dabei steht das Konzept und die Möglichkeit eines späteren Ausbaus im Fokus.

1.3 Struktur der Arbeit

Die Arbeit ist in sechs Kapitel strukturiert. In diesem ersten Kapitel wird das betrachte Problem, das angestrebte Ziel und die Struktur der Arbeit vorgestellt.

Kapitel 2 befasst sich mit den Grundlagen, welche sowohl bei der Konzeption und Umsetzung der Plattform als auch bei der Gestaltung von

zwei prototypischen Lernmodulen eine Rolle spielen. Folglich werden hier Techniken genauso wie Begriffe der IT-Security behandelt.

Im Anschluss an die Grundlagen wird in Kapitel 3 eine Auswertung der Arbeiten und Anwendungen, die bereits zu diesem Thema existieren, behandelt. Es findet bereits hier eine Abgrenzung zu diesen Verwandten Arbeiten statt, um das Gap klar darzustellen und Anforderungen an das angestrebte Produkt zu formulieren.

Nach der Analyse folgt in Kapitel 4 die Konzeption und Umsetzung beginnend mit dem Aufteilen und Aufsetzten der Repositories und dem Abwägen verschiedener technischer Möglichkeiten. Darauf folgt die Entwicklung der Abläufe, um Inhalte in die Plattform einzubinden und Umgebungen für Übungsaufgaben zu starten.

Im darauf folgenden Kapitel 5 wird die prototypische Umsetzung von zwei Lernmodulen aufgezeigt, die verschiedene Stärken der zuvor erstandenen Plattform nutzen.

Das abschließende Kapitel 6 enthält eine Zusammenfassung der Arbeit und einen Ausblick auf mögliche Erweiterungen, um den entstandenen Prototypen zu einem fertigen Produkt weiterzuentwickeln.

Kapitel 2

Grundlagen

Dieses Kapitel beschäftigt sich mit dem für die praktische Umsetzung notwendigen Hintergrundwissen. Zunächst geht es um die Grundlagen, die bei der Konzeption der Projekt-Struktur zum Einsatz kommen, gefolgt von den im Projekt verwendeten Techniken und Methoden, die einen reibungslosen der Plattform ermöglichen. Zum Verständnis der Lernmodule folgen zuletzt noch Ausführungen und Begrifflichkeiten aus dem Bereich der IT-Security.

2.1 Software Engineering-Grundlagen

Der folgende Abschnitt beschäftigt sich mit Begriffen, die beim Aufsetzten des Projekts berücksichtigt werden müssen und sowohl Software-Qualität als auch die allgemeine Struktur betreffen.

2.1.1 Code Conventions

Code Conventions fallen in den Bereich der Software-Qualität. Sie stellen Regeln auf, die die vorhandene Freiheiten in den Programmiersprachen einschränken. Die betroffenen Freiheiten sind unter anderem: Variabel- und Klassennamen, Strukturen und Formatierung. Es wird dabei das Ziel verfolgt, innerhalb eines Projektes einen einheitlichen Stil unabhängig vom Softwareentwickler beizubehalten. Durch das Befolgen der selbst auferlegten Regeln ergibt sich ein konsistentes und homogenes Code-Bild, welches wiederum bei Reviews und für Dritte leichter verständlich ist, auch wenn dafür weiterhin Kenntnisse über die verwendeten Techniken notwendig sind [16].

Beispiele für Code Conventions sind:

- Einrückungen werden mittels Tabulator vorgenommen.
- Getter-Methoden tragen das Präfix get, Setter-Methoden hingegen das Präfix set.

- Jede Klasse wird in eine Datei auslagert, die den Klassennamen als Dateinamen trägt.

In *Software Engineering Practices for Minimizing Technical Debt* werden Code Conventions von Vinay Krishna1 und Dr. Anirban Basu auch als Möglichkeit vorgeschlagen, um eine Reduzierung der Technical Debt zu erreichen. Technial Debt beschreibt dabei die Wahl eines Vorgehens mit kurzfristig wenig Aufwand, der mit der Zeit jedoch zu Einbußen führt. So können Code Conventions zu einem niedrigerem Aufwand bei Erweiterung und Wartung führen [19].

Einzelne Entwickler lassen sich dann nicht mehr durch den geschriebenen Code identifizieren, da die Formalien zu einem gleichen Code-Bild führen sollten. Einen Verlust von Verantwortung ist dabei jedoch keine Gefahr, da die Autoren weiterhin in der Versionsverwaltung identifizierbar bleiben.

Liegen die Richtlinie in einem entsprechendem Format vor, kann durch Editoren, Compiler und verschiedene Arten der Automatisierung auch der Computer die Einhaltung von Code Conventions prüfen.

2.1.2 Automatisierung durch CI/CD-Pipelines

Eine der möglichen Automatisierungen sind Pipelines. Sie werden nach dem Hochladen eines neuen Commits in die Versionsverwaltung von alleine oder manuell ausgeführt.

“In continuous intergration, software building and testing are automated. Continuous deployment takes this practice one step further by also automatically deploying software changes to production. “[21]

Allgemein bieten Pipelines die Möglichkeit, sich wiederholende Aufgaben aller Art zu automatisieren. Dabei wird ein Skript angelegt, welches je nach Konfiguration manuell oder automatisch ausgeführt wird. Dies geschieht jeweils auf Grundlage des aktuellen Standes in der Versionsverwaltung bzw. kann manuell auch für vorherige Stände ausgeführt werden. Mögliche Automatisierungen betreffen hier beispielsweise das Überprüfen von Qualitätsmerkmalen, das Kompilieren auf unterschiedlichen Architekturen, die Durchführung von Unit- und Integrationstests sowie das automatische Deployment der fertigen Anwendung auf Staging- und Produktionssysteme. Zugleich schafft man mit dem Build-Server für alle Entwickler eine einheitliche Plattform, auf der das Arbeitsergebnis funktionieren muss und unterbindet damit Aussagen wie “Auf meinem Rechner läuft es“. [2]

Ziel ist es, monotone und repetitive Aufgaben von den Schultern der Entwickler zu nehmen und ihnen die Möglichkeit zu geben, sich um die tatsächliche Entwicklung und, im Falle von DevOps (siehe [13]), den Betrieb der Software zu kümmern. Zeitgleich kann Fachwissen, welches beispielsweise zum Bereitstellen nötig ist, einmalig vorbereitet werden und dann beliebig oft auch von anderen Team-Mitgliedern ohne Vorkenntnisse ausgeführt werden.

Durch automatisch gestartete Pipelines kann auch sichergestellt werden, dass kein relevanter Schritt vergessen oder übergangen wird [2, 27].

Bei einer wachsenden Codebasis in einem Projekt kann die Dauer bis zum vollständigen Durchlaufen einer Pipeline für das gesamte Projekt durchaus eine längere Zeit in Anspruch nehmen. Ein naheliegender Schritt bei großen und komplexen Projekten ist es, eine Unterteilung in diverse kleine Elemente vorzunehmen, um bei verhältnismäßig kleinen Code-Änderungen die Pipeline nicht für das gesamte Projekt durchlaufen lassen zu müssen.

2.1.3 Monolithen und Micro-Services

Herkömmlicherweise wird ein Projekt, sofern möglich, in einer großen Codebasis umgesetzt. Dabei arbeiten alle Entwickler in der gleichen Projektumgebung. Am Ende entsteht eine Software, die alle Anforderungen an das Projekt abdeckt. Man spricht in diesem Fall auch von einem Monolithen. Bei steigender Verknüpfung und Projektgröße werden Monolithen jedoch schwer durchschaubar und wartungsintensiv. Wie bereits bei den Pipelines 2.1.2 ausgeführt, sind auch längere Zeiten bis zur Bereitstellung eine mögliche Folge dieser Architektur [11].

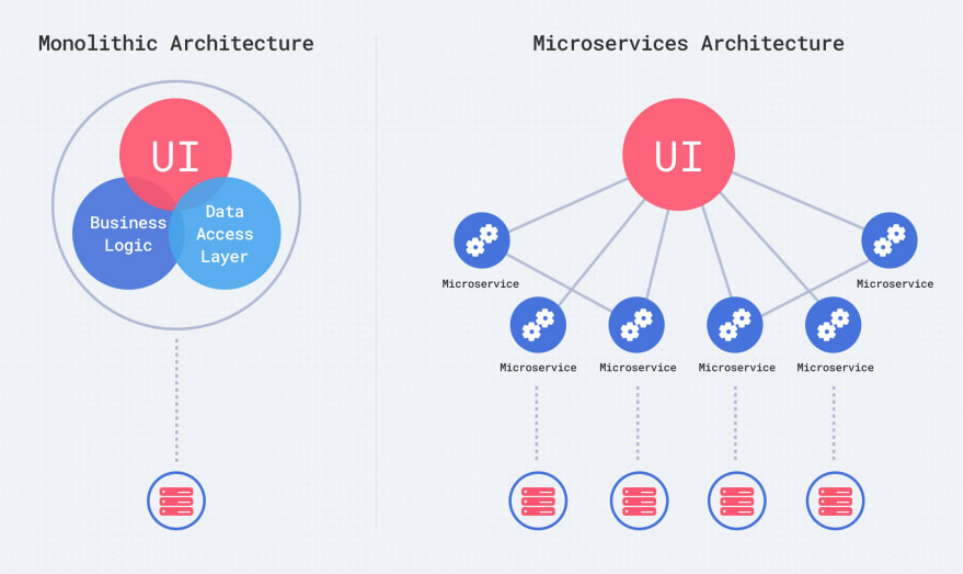

Abbildung 2.1: Micro-Services vs. Monolith [5]

Micro-Services verfolgen dagegen den Ansatz, ein Projekt in kleinere Services mit definierten Zuständigkeiten zu zerlegen (Abbildung 2.1). Jeder Service hat einen klar definierten Aufgabenbereich, den er abdecken, aber nicht überschreiten soll. Dabei sollte der Aufgabenbereich klein genug gewählt sein um überschaubar zu bleiben. Zudem sollten die einzelnen

Dienste möglichst unabhängig voneinander operieren und nur über definierte Schnittstellen mit anderen Micro-Services kommunizieren [12, 27].

Durch klar definierte Schnittstellen und die ansonsten vorgesehene Unabhängigkeit einzelner Micro-Services ist es, anders als in Monolithen, sehr einfach möglich, Services in unterschiedlichen Programmiersprachen zu schreiben und Services gegebenenfalls auch vollständig auszutauschen, ohne weitere Auswirkung auf das übrige Projekt. Schnittstellen sollten deshalb möglichst unabhängig von der darunter liegenden Implementierung gewählt werden. Beispielsweise durch den Einsatz von HTTP/REST Schnittstellen [27].

Die Unterteilung bedarf jedoch eines größeren administrativen Aufwands, der häufig auch Orchestrierung genannt wird. Im Vergleich zu Monolithen stellt die Orchestrierung dabei andere Anforderungen an die Technik und die Produktivsysteme im Hintergrund [1].

2.2 Technische Grundlagen

Zur Bereitstellung der Software sind fortgeschrittene Techniken notwendig, die in diesem Abschnitt eingeführt werden.

2.2.1 Virtuelle Maschinen und Container

Bei der Entwicklung von Software kann es hilfreich und manchmal nötig sein, auf bestehende Implementierungen, andere Dienste oder Schnittstellen zuzugreifen. Damit stellt die Software Anforderungen sowohl an die Build-Umgebung, als auch an das spätere Produktivsystem. Benötigt werden beispielsweise installierte Frameworks, verwendete Command Line Interface Tools (CLI) und bestimmte Konfigurationen. Es gibt verschiedene Möglichkeiten diese so zu bündeln, dass sie Hardward-unabhängig vorliegen.

2.2.1.1 Virtuelle Maschinen

Virtuelle Maschinen sind ein Verfahren zur Abstraktion von Hardware. Dabei wird auf einem Host-System ein Hypervisor installiert, der wiederum den virtuellen Maschinen verschiedene Ressourcen wie CPU-Kerne/Threads, RAM, Festplattenspeicher, Netzwerk und Periphere zur Verfügung stellt. Durch die Abstraktion der Hardware sind virtuelle Maschinen Hardware-unabhängig und können so leicht auf ein anderes System umgezogen werden. Die virtuelle Maschine verfügt über ein vollständiges Betriebssystem mit eigenem Kernel, der beim Start des Gast-Systems mit hochfährt. Es ist möglich, wie in Abbildung 2.2 zu sehen, mehrere virtuelle Maschinen beziehungsweise Gast-Systeme auf einem Host zu starten. Durch die Zuweisung der Ressourcen kann eine bessere Gesamtauslastung des Host-Systems erreicht werden. Allerdings sind virtuelle Maschinen so abgeschottet, dass sie keinen Einfluss

auf andere virtuellen Maschinen und das Host-System nehmen können. Somit kann die virtuelle Maschine wie ein eigenständiger Computer behandelt werden. Durch Images und Snapshots lassen sich beliebige gesicherte Zustände wiederherstellen, sollte es zu Problemen kommen. Es ist möglich, virtuelle Maschinen so vorzubereiten, dass sie eine komplette Produktionsumgebung mit sich führen, beispielsweise einen Web-Server, eine Datenbank sowie Konfigurationen um direkt als Produktivsystem gestartet werden können [27, 10, 2].

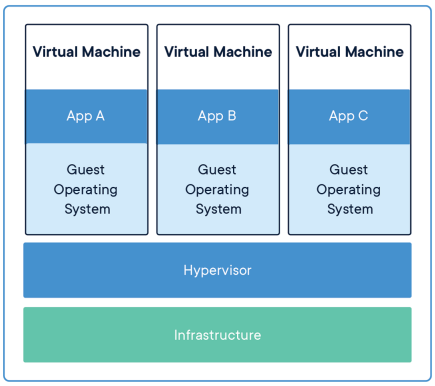

Abbildung 2.2: Host mit mehreren virtuellen Maschinen [10]

2.2.1.2 Container

Container hingegen stellen kein komplettes Betriebssystem zur Verfügung, sondern lediglich eine Software. Sie nutzen dabei den Kernel des Host Systems. In Abbildung 2.3 ist eine beispielhafte Container-Umgebung zu sehen. Durch Cross-Compiling lassen sich auch Container für andere Kernel erstellen. Auch Container sind voneinander abgeschottet und können nur über definierte Schnittstellen miteinander kommunizieren. Wie auch bei virtuellen Maschinen erhalten sie Ressourcen zugewiesen [10].

Bei Containern gibt es zunächst das Buildfile, welches eine Anleitung dafür ist, welche Software installiert und welche Dateien kopiert werden sollen. Aus dem Buildfile entsteht ein Image, welches wiederum als Ausgangspunkt für ein anderes Buildfile verwendet werden kann oder um eine Instanz des Containers zu starten. Es können beliebig viele Instanzen eines Images gestartet werden, die über eingebundene Verzeichnisse und Environment-Variablen angepasst werden können. Durch das Erweitern von bestehenden Images entstehen Schichten, die sich klar nachvollziehen lassen. In dem Buildfile ist genau beschrieben, wie sich das Image zusammensetzt. Virtuelle Maschinen sehen so etwas hingegen nicht vor, können mit Techniken wie Ansible aber auch ähnlich vorbereitet werden, jedoch unter höherem Aufwand [1].

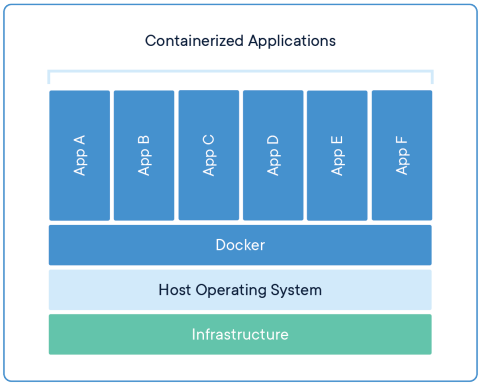

Abbildung 2.3: Host mit einer Container-Umgebung [10]

Docker gilt als erstes großes Projekt, um Container-Umgebungen zu ermöglichen. Die dort umgesetzten Techniken wurden von vielen anderen Anbietern übernommen und weite Teile der Open Container Initiative (OCI) stammen direkt von Docker. Einer der Vorteile hierbei ist, dass Container Images, die auf einer OCI-konformen Plattform erstellt wurden, auch auf anderen OCI-konformen Plattformen lauffähig sind [10, 17].

Beide Lösungen, sowohl virtuelle Maschinen als auch Container, bieten die Möglichkeit, eine Anwendung gemeinsam mit ihrer benötigten Umgebung auszuliefern und dank der Isolierung auch parallel auf einem System zu betreiben. Der Parallelbetrieb kann dazu führen, dass unterschiedliche Dienste versuchen auf den gleichen Port zu hören. Dies macht ein Ausweichen auf andere Ports oder weitergehende technische Maßnahmen wie beispielsweise Ingress Routing nötig.

2.2.2 Ingress Routing

Ingress Routing ermöglicht es, mehrere Dienste oder auch mehrere Instanzen eines Dienstes, die auf den gleichen Port hören, auf einem System zu starten, ohne diese extern auf anderen Ports anbieten zu müssen. Dabei wird ein Reverse Proxy gestartet, der die Kontrolle über den gewünschten Port übernimmt und anschließend eine entsprechende Weiterleitung vornimmt. Dabei werden die Hostnamen bzw. Domainnamen verwendet, um die unterschiedlichen Ziele zu unterscheiden. Dies ähnelt dem Verhalten von Webserververn, die über unterschiedliche Adressen auch unterschiedliche Websites/VHosts anbieten können. Webserver wie Apache und NGINX beispielsweise sind auch in der Lage als Reverse Proxy für HTTP Request zu arbeiten und diese Requests an andere Services auf anderen Ports weiterzuleiten. Diese können somit ebenfalls auf dem gleichen System über den Standard HTTP Port 80 und HTTPS Port 443 angeboten werden. Je nach gewähltem Reverse Proxy ist es auch möglich, über weitere Faktoren

wie Header-Informationen, Cookies oder Login-Daten die Weiterleitung zu beeinflussen [34].

2.3 Kryptographie

Zum tieferen Verständnis der prototypischen Module sind gewisse Grundlagen der IT-Security nötig. Ein wichtiger Teil dieser Grundlagen ist die Kryptographie. Bevor an dieser Stelle Kryptographie im Allgemeinen besprochen wird, werden zunächst einige Rollen eingeführt, die bei der Erklärung hilfreich sind.

2.3.1 Rollen zur Erklärung

Um Erklärungen von Prozessen verständlicher wiederzugeben, werden in der IT-Security oft Namen genutzt, um Rollen zu beschreiben, die an einem Prozess beteiligt sind. Dabei sollte jedoch stets bedacht werden, dass diese Rollen sowohl von Personen als auch von Computer-Systemen eingenommen werden können [29].

Man spricht oftmals von Alice und Bob, die schützenswerte Informationen austauschen möchten.
Beispiel: Alice möchte Bob eine Nachricht mit den aktuellen Bilanzen schicken.

Nun gibt es Gegenspieler, die versuchen, an diese schützenswerten Informationen zu gelangen. Dabei unterscheidet man aktive und passive Parteien. Ein Partei, die passiv an den Prozessen teilnimmt, also nur versucht an Information zu gelangen, die ihr nicht zustehen, wird oftmals als Eve (abgeleitet vom englischen Wort eavesdropper für Lauscher oder auch Abhörer) bezeichnet.
Beispiel: Eve hört Alice ab um die Bilanzen zu erhalten, die diese an Bob schickt.

Mallory (abgeleitet vom englischen Wort malicious für hinterhältig oder auch heimtückisch) bezeichnet eine Partei, die auch aktiv in die Prozesse eingreift und dabei beispielsweise Nachrichten fälscht oder austauscht.
Beispiel: Mallory fängt die von Alice gesendeten Bilanzen ab und sendet gefälschte Bilanzen an Bob.

Die Kryptographie verfolgt vier Ziele: Vertraulichkeit, Authentizität, Integrität und Verbindlichkeit. Zum einen versuchen Alice und Bob, die Nachrichten vertraulich zu halten, sodass Eve nicht mitlesen kann. Zum anderen soll Bob verifizieren können, dass eine Nachricht von Alice verfasst wurde und ihm diese nicht von Mallory zugespielt wurde. Darüber hinaus soll ebenfalls sichergestellt sein, dass die Nachricht unverändert ankommt und der Autor nicht abstreiten kann, dass diese von ihm stammt [14, 4].

Abzugrenzen ist hierbei noch die Steganographie. Ziel der Steganographie ist es, Information zu verbergen und nach Möglichkeit unauffindbar zu

platzieren. Dabei soll die Existenz der Information geheim gehalten werden. Es handelt sich hier ebenfalls um einen Teil der IT-Security, dieser wird im weiteren Verlauf jedoch nicht weiter betrachtet.

Grundlegend kann man zwischen symmetrischer und asymmetrischer Kryptographie unterscheiden. Dies hat Auswirkungen auf die Art des Schlüssels und des Algorithmus, der benötigt wird, um eine Nachricht zu ver- bzw. zu entschlüsseln.

2.3.2 Symmetrische Verfahren

Bei symmetrischen Verfahren wird sowohl zum Verschlüsseln als auch zum Entschlüsseln der gleiche Schlüssel verwendet. Dies setzt voraus, dass Alice und Bob sich zunächst auf einen gemeinsamen Schlüssel (ein Shared Secret) einigen und diesen vor Dritten schützen [30].

Ein Beispiel für ein simples symmetrisches Verfahren ist die Cäsar-Chiffre. Sie ersetzen einzelne Buchstaben mit einem anderen. Der Schlüssel ist dabei, um wie viele Stellen im Alphabet die Buchstaben verschoben werden [4].

Eine Herausforderung bei symmetrischen Verfahren ist der Schlüsselaustausch, da sowohl Alice als auch Bob die gleiche Information benötigen, um an der Kommunikation teilzunehmen. Hierfür gibt es das Diffie-Hellman-Verfahren, um auch über unsichere Kommunikationskanäle einen geheimen Schlüssel zu generieren. Auch wenn Eve alle Nachrichten mitliest, gelingt es bei diesem Verfahren den Schlüssel geheim zu halten. Dennoch besteht die Gefahr eines Man-in-the-Middle Angriffes, da Bob nicht verifizieren kann, ob er mit Alice die Schlüsselgenerierung durchführt oder ob Mallory diesen Vorgang aktiv beeinflusst [4].

2.3.3 Man-in-the-middle Angriff

Grundlegend ist ein Man-in-the-Middle Angriff mit dem Prinzip von “Stiller Post“ vergleichbar. Jedoch beeinflusst der Angreifer Mallory die Kommunikation aktiv nach seinem Willen und unterbricht die direkte Kommunikation zwischen Alice und Bob. Gerade wenn Alice und Bob zuvor keine Gelegenheiten hatten, sich auf einen gemeinsamen Schlüssel zu einigen, stehen sie vor einem Problem. Versuchen Alice und Bob mit Hilfe des Diffie-Hellman-Verfahrens sich auf einen gemeinsamen Schlüssel zu einigen, kann Mallory die nötigen Nachrichten ersetzen. So ist Mallory in der Lage, jeweils mit Alice und Bob eigene Schlüsselinformationen zu generieren, da keiner von beiden verifizieren kann, vom wem diese stammen. Kommt es nun dazu, dass Alice eine verschlüsselte Nachricht an Bob schicken möchte, verschlüsselt sie diese mit dem Schlüssel, den sie unwissentlich mit Mallory ausgehandelt hat. Mallory ist nun in der Lage die Nachricht zu lesen und entweder unverändert mit dem Schlüssel für Bob erneut zu verschlüsseln und weiter zu senden

oder auch die ursprüngliche Nachricht zu verfälschen beziehungsweise eine Nachricht nicht weiterzuleiten [24].

Durch Verwendung von Signaturen lassen sich solche Angriffe erkennen und verhindern, da eine Authentifizierung stattfinden kann. Hierfür werden jedoch die asymmetrischen Verfahren benötigt.

2.3.4 Asymmetrische Verfahren

Bei asymmetrischen Verfahren kommt ein Public und ein Private Key zum Einsatz. In diesem Fall erzeugt Alice einen Private Key, den sie schützen muss und nicht weiter geben darf, und einen dazugehörigen Public Key, welchen sie frei verteilen kann [30].

Nun gibt es zwei Möglichkeiten dieses Schlüsselpaar zu verwenden: verschlüsseln und signieren. Beim der Verschlüsselung nutzt Bob den Public Key, um die Nachricht zu verschlüsseln. Danach kann nur noch Alice die Nachricht mit dem Private Key wieder entschlüsseln, denn der Public Key kann die Verschlüsselung nicht wieder aufheben. Beim Signieren nimmt Alice ihren Private Key und wendet ihn auf die Nachricht an. Im Anschluss kann der Public Key genutzt werden, um diese Signatur zu prüfen. Da dieser frei zur Verfügung steht, kann der Inhalt der Nachricht nicht als geheim angesehen werden. Die Signatur ist also nur als Unterschrift von Alice anzusehen, nicht als Mittel zur Verschlüsselung [30].

Asymmetrische Verfahren sind im Allgemeinen nicht so performant wie symmetrische Verfahren. Gerade im Bereich von Embeded Systemen spielt dies eine relevante Rolle [25].

Um die Zuweisung eines Public Keys zu einer Identität zu gewährleisten, kommen Public Key Infrastructures (kurz PKI) zum Einsatz. Dabei stellt eine Zertifizierungsstelle, der alle Teilnehmer vertrauen, einen Public Key zur Verfügung. Möchte Alice nun ihren Public Key an ihre Identität binden, meldet sie sich bei der Zertifizierungsstelle. Die Zertifizierungsstelle überprüft ihre Identität und signiert im Anschluss den öffentlichen Schlüssel von Alice und ermöglicht es ihr, diesen nun weiterzuverteilen. Bob kann bei Erhalt einer Nachricht einfach mit Hilfe des öffentlichen Schlüssels der Zertifizierungsstelle prüfen, ob die Signatur unter Alice Public Key stimmt und diesem im Anschluss vertrauen.

2.3.5 Kombination symmetrischer/asymmetrischer Verfahren

Eine Kombination der beiden Verfahren kommt bei der Umsetzung von SSL/TLS zum Einsatz. Beim Verbindungsaufbau oder auch Handshake (Abbildung 2.4) kommen signierte Nachrichten zum Einsatz, die zwar zeitaufwendiger sind, aber eine Prüfung der Identität zulassen. Die Identität des Servers kann dabei durch eine PKI sichergestellt werden. Durch die Ver-

wendung von Verfahren zur Schlüsselgenerierung, wie Diffie-Hellman, können sich Server und Client auf einen gemeinsamen Schlüssel einigen. Dieser kann dann genutzt werden, um im Anschluss an den Verbindungsaufbau, schnell und performant mittels symmetrischer Verfahren die Kommunikation zu verschlüsseln [30, 8].

Ein Cipher Suite legt dabei die Kombinationen und Stärken von Algorithmen fest, die zum Einsatz kommen. Hierfür gibt unter anderem das Bundesamt für Sicherheit in der Informationstechnik (BSI) Empfehlungen aus, welche Cipher Suites dem aktuellen Stand der Technik entsprechen [8].

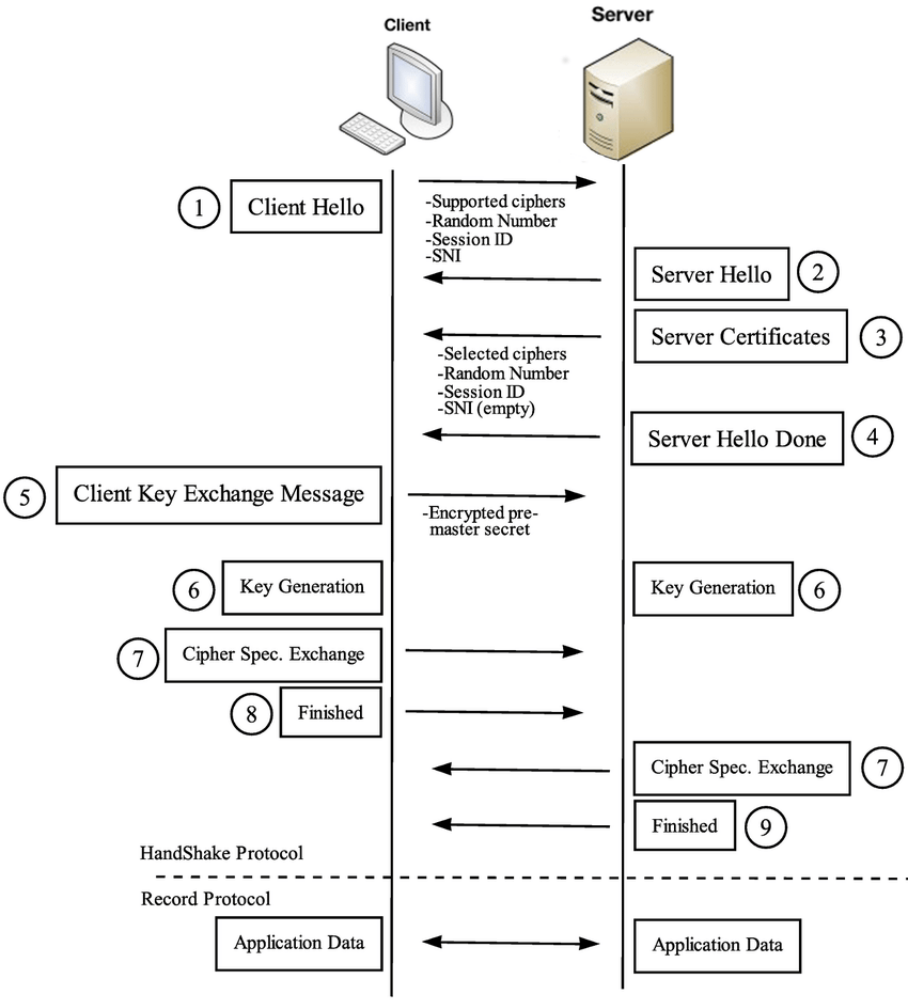

Abbildung 2.4: Ablauf des TLS handshake protocol [33]

2.4 Schwachstellen

Das Wissen zu Schwachstellen ist ein wichtiger Teil der IT-Security und zum Verständnis der Lernmodule relevant. Dabei sind in dieser Arbeit vor allem solche Schwachstellen relevant, die auf ungeprüften Eingaben beruhen.

2.4.1 Buffer-overflow Angriffe

Bei Buffer-overflow Angriffen macht man sich die interne Speicheranordnung und das Ablegen von Variablen in dieser zunutze. Aufeinanderfolgende Variablen erhalten in der Regel aufeinanderfolgende Speicheraddressen. Durch fehlende Prüfungen der Zugriffslängen oder das Verzichten auf Prüfwerte werden diese Angriffe begünstigt beziehungsweise ermöglicht. Dies kann durch weitere Umstände beeinflusst werden, beispielsweise durch die Compiler-Konfiguration. Wird ein Wert in den Buffer geschrieben, der die Länge des dafür reservierten Speicherplatz überschreitet, kann dies dazu führen, dass auch der Speicherplatz der nachfolgenden Variablen mit überschrieben wird [18].

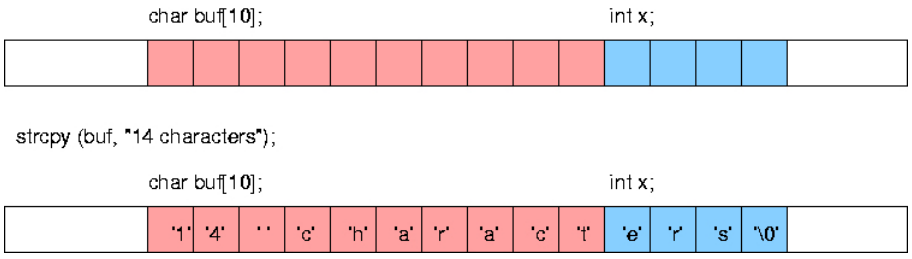

Abbildung 2.5: Beispiel für einen Buffer-overflow [3]

Schreibt man also anstatt der vorgesehenen zehn Zeichen weitere Zeichen in die erste Variable, ändert sich der Wert der zweiten Variable ebenfalls (siehe Abbildung 2.5). Auch das Fehlen des NULL-Bytes kann bereits zu Problemen führen, sollte das Programm einen String bis zum nächsten NULL-Byte lesen wollen. Durch Abwandelungen dieses Angriffes kann es auch möglich sein, je nach Compiler-Konfiguration, Rücksprungadressen anzupassen und tiefergehenden Zugang zu erlangen. Grundsätzlich beruht der Angriff darauf, dass Eingaben nicht ausreichend geprüft werden [18].

2.4.2 Injection

Auch bei einer Code oder Command Injection wird der Umstand ausgenutzt, dass Nutzereingaben ungeprüft im Programmablauf verwendet werden. Es gibt diverse Schwachstellen, die dieser Kategorie angehören, wie beispielsweise Cross-Site-Scripting (XSS) Attacken. Für die Arbeit von Bedeutung sind jedoch SQL-Injections.

Eine SQL-Anfrage an den Datenbank-Server wird zur Programm-Laufzeit zusammengestellt und interpretiert. Dies ermöglicht es beispielsweise Nutzern auf Webseiten, eine Suchfunktion anzubieten, die Datenbankeinträge mittels eines Freitext-Feldes durchsucht. Mallory könnte nun über das Suchfeld die Anfrage umgestalten, beispielsweise durch das Einschleusen von Steuerzeichen oder SQL-Befehlen. Ziel ist es, ein Verhalten hervorzurufen, das vom Entwickler in dieser Form nicht beabsichtigt war [35].

```
// anfälliger Code
Statement sm = connection.createStatement();
String query = "SELECT id FROM Users WHERE user='" + user
                    + "' AND pw='" + pw + "'";
ResultSet result = sm.executeQuery(query);

// gepatchter Code
PreparedStatement sm = connection.prepareStatement( "SELECT id
    ↪ FROM Users WHERE user=? AND pw=?");
sm.setString(1, user);
sm.setString(2, pw);
ResultSet result = sm.executeQuery();
```

Listing 2.1: Eine (gepatchte) SQL-Injection-Schwachstelle (angelehnt an [7])

Um diese Art von Angriff zu unterbinden, befreit man die Nutzereingaben von potentiell gefährlichen Zeichen oder Texten. Man spricht hier auch vom sanitizing (englisch für desinfizieren). Dabei werden Steuerzeichen entweder entfernt oder durch zusätzliche Zeichen escaped, beispielsweise um Anführungszeichen in Datenbankeinträgen zu ermöglichen, ohne die Anfrage zu beeinflussen. Moderne Datenbank-Frameworks unterstützen das Binden von Variablen in Anfragen und sanatizen diese automatisch, bevor eine Auswertung stattfindet (siehe Listing 2.1). Auch eingeschränkte Nutzerrechte können dabei helfen, die mögliche Angriffsfläche zu reduzieren. Genauso sind Einschränkungen durch Views auf eine Tabelle eine Möglichkeit, die potentiell betroffenen Daten bei manipulierten SQL-Anfragen zu begrenzen [35].

Kapitel 3

Gap-Analyse

Es gibt bereits verschiedene Ansätze, die versuchen, die in Kapitel 1 angesprochenen Probleme zu lösen. In diesem Abschnitt werden drei dieser Lösungen vorgestellt und im Anschluss Anforderungen an das Projekt ausgearbeitet. Dabei zeigen Teile der Anforderungen eine Abgrenzung und Weiterentwicklung gegenüber der hier gezeigten Lösungen.

3.1 Ein interaktives IT-Security-Labor

Die Masterarbeit *Virtualisation-based Infrastructure for Practical IT-Security Exercises* von Christian Thoms [36] beschäftigt sich mit einer Ergänzung zu einem bestehenden IT-Security-Labor. Darüber hinaus wurden drei neue Laborübungen geschaffen. Die einzelnen Aufgaben wurden in Docker-Container eingefasst. Jeder Container enthält dabei nur die nötige Software und Konfiguration, die für die Aufgabe relevant ist.

Die gewählte Umsetzung sieht dabei einen SSH-Container als Zugangspunkt zu den Aufgaben vor. Die Teilnehmer müssen hierfür im Vorfeld einen SSH-Key generieren und sich zunächst per SSH-Client an dem Zugangspunkt anmelden. Von dort aus stehen ihnen dann die Möglichkeiten und die dort installierte Software per Terminal zur Verfügung.

Innerhalb von Docker wurden durch Software-definierte Netzwerke die Aufgaben untereinander verknüpft. Der Zugangspunkt stellt die Verbindung ebenfalls über dieses Vorgehen her. Hierdurch wurde auch eine Abgrenzung einzelner Teilnehmer voneinander realisiert.

Zur Kontrolle des Lernfortschritts enthalten die Aufgaben Informationen, die im Rahmen eines Capture-The-Flag (CTF) Events genutzt werden. Dabei müssen Teilnehmer die Information in Form eines kurzen Textes finden oder Zugang zu diesem erlangen. Diese Information wird an einer zentralen Plattform abgeliefert und kontrolliert.

Dabei ermöglicht es Ansible[1] den Teilnehmern über die bestehende

[1] `https://www.ansible.com/`

Plattform ihre eigenen Umgebungen zu starten und zu steuern. Ansible ist ein Tool, in dem ein Soll-Zustand einer Serverumgebung beschrieben wird, den das System automatisch versucht zu erreichen.

Dieser Ansatz wird bereits in einem IT-Security Labor eingesetzt.

3.2 Container zur Bereitstellung vorkonfigurierter Umgebungen

In der Bachelorarbeit *Konzeption und Umsetzung einer IDE in einem Docker-Container* von Jonny Niebuhr [28] wird das Vorgehen geschildert, wie eine vollwertige Entwicklungsumgebung (oder auch IDE genannt) in einen Docker-Container eingefasst werden kann. Dabei wurde auf eine bestehende IDE zurückgegriffen. Diese wurde gemeinsam mit weiteren Software-Paketen, wie beispielsweise Sprachpakete und Compiler, in ein Docker-Image installiert. Das erstellte Docker-Image erhält dabei auch eine vorgefertigte Konfiguration.

Der Zugang zu der IDE findet dabei über eine Virtual Network Computing-Lösung (VNC) statt. VNC ist dabei eine Lösung zur Übertragung des Bildschirmes und der Steuerung mittels Maus- und Tastureingaben. Hierzu wird im Container ein VNC-Server installiert und per Port-Freigabe dem VNC-Client des Anwenders zur Verfügung gestellt.

In Kombination ergibt sich eine IDE, die bereits vorkonfiguriert und mit Software und Tools vorinstalliert ausgeliefert werden kann und dabei wenig bis keine Abhängigkeiten zum Host-System besitzt. Dem Nutzer stehen dabei alle Möglichkeiten der ursprünglichen IDE zur Verfügung. Als einzige Voraussetzung werden ein VNC-Client und ein Docker-Host benötigt.

Dieser Ansatz könnte in dieser Form auch genutzt werden, um praktische Aufgaben für IT-Security-Schulungen zu erstellen.

3.3 Online-Tools

Auch online stehen Tools zum Erlernen aktueller Techniken zur Verfügung, die sich nicht nur auf den Bereich der IT-Security beschränken. Umgebungen wie Freecodecamp[2], Play with Docker[3] und CrypTool-Online[4] bieten die Möglichkeit, im Kontext des Webbrowsers, verschiedene Technologien kennenzulernen und zu testen. Eine Installation zusätzlicher Software ist hier nicht nötig.

Betrachtet wird an dieser Stelle das CrypTool-Online, welches Teil des CrypTool-Projektes ist. Das CrypTool-Projekt stellt diverse Open-Source-

[2] `https://www.freecodecamp.org/`

[3] `https://labs.play-with-docker.com/`

[4] `https://www.cryptool.org/`

Lernprogramme mit dem Fokus auf Kryptographie und Kryptoanalyse zur Verfügung.

Die Plattform stellt dem Nutzer dabei keine Aufgaben sondern lässt ihn frei experimentieren, um den Einfluss unterschiedlicher Parameter zu untersuchen. Die einzelnen Seiten enthalten meist einen kurzen Einleitungstext und ein interaktives Beispiel. Darüber hinaus stehen dem Nutzer auf Wunsch auch tiefergehende Informationen sowie Links zu weiteren Quellen zur Verfügung. Weitere Features sind die Schritt-für-Schritt-Module zu RSA und AES sowie weitere kleine Tools wie ein Passwort-Generator.

3.4 Abgrenzung und Anforderungen

Aus den bereits vorhandenen Lösungen und Ansätzen, die in den vorangegangenen Abschnitten zusammengefasst wurden, lassen sich Anforderungen an das Projekt und dessen Konzept stellen. Die Anforderungen haben das Ziel, offene Punkte aus bestehenden Arbeiten aufzugreifen und positive Aspekte beizubehalten. Zeitgleich findet hiermit auch die Abgrenzung zu den bisherigen Arbeiten statt.

[R1] Die Plattform soll im Browser lauffähig sein

Die Verwendung eines SSH-Clients wie in 3.1 ist beispielsweise bei Windows nur über zusätzliche Software wir PuttY möglich. Auch ein Zugriff auf Software über VNC wie in wie in 3.2, setzt die Installation einer Client Applikation voraus. Sowohl in Universitäten als auch in Unternehmen ist die Installation von Software jedoch in der Regel Administratoren vorbehalten. Auch Hardware, die von den Teilnehmern selbst administriert wird, erlaubt gelegentlich keine Installation der nötigen Software. Dabei können private Bedenken genauso wie die Inkompatibilität zu bereits installierter Software oder Umgebungen eine Rolle spielen.

Ausgehend von dieser Annahme ist die Installation von zusätzlicher Software im Rahmen einer Schulung nur bedingt möglich und bedarf längerfristiger Vorbereitung. Bei der Konzeption sollte folglich beachtet werden, dass auf der Seite von Teilnehmern nur Software angewendet wird, die üblicherweise mit allen gängigen Betriebssystemen ausgeliefert wird.

[R2] Die Plattform soll in ihren Aufgaben nicht an eine bestimmte Programmiersprache gebunden sein

Ein Ansatz hierfür wäre die Umsetzung wie in 3.3 beschrieben. Die hier aufgezeigten Tools beschränken sich jedoch meist auf mathematische Aufgaben auf Basis von Javascript. Bei der Konzeption sollte keine Einschränkung auf die Möglichkeiten einer einzigen Programmiersprache stattfinden. Es sollen tiefergehende Arbeiten wie das Ausführen von Command Line Tools erhalten bleiben.

[R3] Die Teilnahme an Übungen soll ohne Eingreifen eines Trainers möglich sein

Der in 3.1 angesprochene Ansatz benötigt zudem einen SSH-Key im Vorfeld. Für den Trainer bedeutet dies einen Aufwand für jeden neuen Teilnehmer und somit zeitliche Einbußen. Das Konzept sollte also vorsehen, den Aufwand für den Trainer konstant zu halten, unabhängig von der Teilnehmerzahl.

[R4] Die Plattform soll skalierbar sein

[R5] Die Plattform soll, soweit möglich, die Ressourcen der Teilnehmer einsetzen

Auch die Anzahl der Teilnehmer kann eine Herausforderung darstellen. Dies betrifft vor allem die Bereitstellung der Lernumgebungen, die auf eine serverseitige Komponente angewiesen sind. Durch eine höhere Teilnehmerzahl ergibt sich dort ein höherer Ressourcenverbrauch. Für das Konzept sollte berücksichtigt werden, dass an geeigneten Stellen die Ressourcen der Teilnehmer verwendet werden, um eine Reduktion der Serverlast zu ermöglichen. Sollte eine Vermeidung der Mehrbelastung des Servers nicht erreichbar sein, dann sollte das Einbringen weiterer Serverressourcen möglichst leicht von statten gehen.

[R6] Die Plattform soll Übungen bestehend aus mehreren System erlauben

Die Umsetzung in 3.1 zeigt aber auf, dass die Möglichkeit besteht, Aufgaben aus mehreren Containern anzubieten, um ein breites Spektrum an Trainingssituationen zu ermöglichen. Im Konzept sollte also beachtet werden, dass Aufgaben auch aus komplexeren Umgebungen mehrerer Einheiten bestehen können.

[R7] Die Plattform soll auf Corporate Identity Richtlinien anpassbar sein

Eine höhere Akzeptanz bei Trainern und Auftraggebern soll durch die Anpassung an deren Corporate Identity erreicht werden. Als weitere Anforderung an die Plattform ergibt sich hierdurch eine erleichterte Anpassbarkeit der Optik.

[R8] Die Plattform soll um weitere Übungen erweiterbar sein

Wie eingangs erwähnt schreitet in der IT-Security der Stand der Technik stetig weiter fort. Folglich sollten Lernmodule stetig weiter entwickelt und ergänzt werden. Im Konzept und bei der Umsetzung sollte dies berücksichtigt werden.

Kapitel 4

Konzept und Implementation

Ziel dieses Kapitels ist es, ein Grundgerüst zu schaffen, in dem das Projekt umgesetzt und mit Lernmodulen verknüpft werden kann. Zunächst ist dabei die Frage zu klären, ob die Plattform in Form eines großen Monolithen geschrieben oder in kleinere Micro-Services unterteilt werden soll. Auch eine Entscheidung zur Wahl von Virtuellen-Maschinen oder Containern und der Ingress-Routing-Lösung muss in diesem Kapitel getroffen werden. Anschließend werden diese Entscheidungen bei der Implementierung der Plattform eingesetzt.

In dieser Arbeit erfolgt dabei eine Eingrenzung der betrachteten Produkte. Hierdurch soll innerhalb der zeitlichen Limitierung ein lauffähiger Prototyp entstehen. Dabei werden vorrangig Ansätze betrachtet, die dem Autor vertraut sind, um die Einarbeitungszeit zu minimieren. Teilweise sind auch Entscheidungen für bestimmte Produkte auf diesen Umstand zurückzuführen. Dieses Vorgehen wird möglich, da es sich um ein Greenfield-Projekt[1] handelt. Es geht dabei nicht zwangsläufig um die Wahl der besten Lösung sondern um die praktikabelste.

4.1 Monolith oder Micro-Services

Die Plattform soll wie in [R1] gefordert im Web-Browser lauffähig sein. Dies hat zur Folge, dass eine Website entwickelt werden muss, die wiederum von einem Web-Server angeboten wird. Dieser Web-Server lässt sich dann ebenfalls in einzelne Aufgabenbereiche unterteilen. Zudem soll die Plattform nach [R4] auch skalierbar sein. Der Einsatz des Micro-Service-Patterns verspricht hier die größere Flexibilität. Während bei Micro-Services die Dienste zu einzelnen Aufgabenbereichen modelliert und danach unabhängig voneinander skaliert werden können, ist dies bei Monolithen nicht der Fall. Wie in den Grundlagen angesprochen, würde ein Monolith alle Aufgabenbereiche des Servers vereinen und ließe sich nur als ganze Einheit skalieren.

[1] Greenfield-Projekt sind nicht durch bestehende Arbeitsergebnisse eingeschränkt

Benötigt also nur ein Teilbereich weitere Ressourcen, lässt sich dies nicht umsetzen, ohne dabei auch weitere Ressourcen für die anderen Teilbereiche anzubieten.

Auf Grundlage dieser Abwägung wurde das Projekt in kleinere Services unterteilt. Der Mehraufwand bei der Verwaltung von Micro-Services gegenüber dem eines Monolithen wird dabei in Kauf genommen. Im späteren Verlauf wird sich zeigen, dass dieser Mehraufwand im Vergleich zu der Verwaltung der Nutzer-Umgebungen zu vernachlässigen ist.

4.2 Virtuelle Maschinen oder Container

Auch bei der Entscheidung zwischen virtuellen Maschinen und Containern spielt, wie bereits bei der Wahl der Micro-Services, die Skalierbarkeit eine wichtige Rolle. Container lassen eine Skalierung zu, die deutlich feingranularer ist und mit geringerem Ressourcen-Aufwand umgesetzt werden kann als dies bei virtuellen Maschinen der Fall ist. Bei der Wahl von virtuellen Maschinen kommt zunächst die Frage auf, ob jeder Service eine einzelne virtuelle Maschine erhält oder eine virtuelle Maschine alle Services vereint. Erhält jeder Service seine eigene virtuelle Maschine, so wächst der Ressourcen-Bedarf rasant an. Der Start einer weiteren virtuellen Maschine bedeutet hier neben dem Start einer weiteren Instanz der Software auch den Start eines weiteren Betriebssystems. Werden hingegen alle Micro-Services in einer virtuellen Maschine ausgeliefert, so verfällt schnell der Vorteil der Micro-Services. Diese stünden hierdurch nicht mehr unabhängig voneinander zur Verfügung. Weitere Instanzen eines Services, in Form einer weiteren virtuellen Maschine, ziehen folglich auch weitere Instanzen aller anderen Services mit sich.

Ein weiterer Punkt, der für die Wahl von Containern spricht, ist die Wiederverwendbarkeit. Die Verwendung von Containern sieht einen Zusammenbau auf Grundlage einer simplen Blaupause in Form eines Skriptes vor. Durch Ergänzungen weiterer Layer zu einem Container-Image können diese erweitert und als neues Image abgelegt werden. Auch das Aktualisieren der einzelnen Bestandteile wird vereinfacht. Hierzu wird das Image einfach unter Hinzunahme des neuen Images erneut gebaut oder durch das Einspielen eines Patches aktualisiert. Virtuelle Maschinen verfügen, wie bereits in den Grundlagen erwähnt, über keine derartig standardisierte Möglichkeit.

Im Rahmen dieser Arbeit wird aufgrund der zuvor genannten Argumente die Umsetzung mittels Containern gewählt. Eine Bereitstellung der Container-Umgebung innerhalb einer virtuellen Maschine wäre weiterhin denkbar, ist allerdings für diese Arbeit nicht weiter relevant. Hier handelt es sich um eine Entscheidung, die bei der Inbetriebnahme getroffen werden kann.

Die Verwendung von Docker-Containern legt zudem den Einsatz von

docker-compose-Dateien nahe. In einer YAML-Datei werden dabei einzelne Container als Services aufgelistet und entsprechend konfiguriert. So wird auch die Orchestrierung der einzelnen Micro-Services erleichtert.

4.3 Die Wahl des Ingress-Routing

Die erstellten Container müssen nun noch der Öffentlichkeit zugänglich gemacht werden. Damit dies geschehen kann, wird Software zum Ingress-Routing eingesetzt. An dieser Stelle werden die beiden möglichen Lösungen Traefik[2] und NGINX[3] betrachtet.

NGINX ist ein Web-Server, der sich ebenfalls als Reverse-Proxy einsetzten lässt. Es handelt sich hier um eine Software, die bereits einige Jahre verfügbar ist und sich erfolgreich bewährt hat. Traefik ist ein verhältnismäßig junges Produkt, das sich auf den Einsatz mit Containern fokussiert. Über das Anhängen von Labeln an Container kann Traefik diese automatisch erkennen und in das Routing mit aufnehmen. Dieses lässt sich über Umwege auch mit NGINX realisieren, jedoch nicht im Rahmen des offiziellen Docker-Images.

Das Absichern der Kommunikation zwischen Teilnehmer und Plattform und innerhalb der Plattform kann über Zertifikate ebenfalls in diesem Schritt abgedeckt werden. Der Einsatz von Let's Encrypt [4] bietet sich an dieser Stelle an. Dabei handelt es sie um eine Zertifizierungsstelle, die kostenlos und automatisiert arbeitet, aber dennoch von Browsern als vertrauenswürdig eingestuft wird. Traefik hat eine solche Anbindung bereits integriert. Bei der Inbetriebnahme von Traefik wird die TLS-Konfiguration an Traefik übergeben. Diese enthält auch alle für Let's Encrypt relevanten Daten, genauso wie die in 2.3.5 beschriebenen Cipher Suites. Im späteren Verlauf ist es nur noch nötig, die Domain anzugeben und Traefik mit einem Label am Container darüber zu informieren, dass eine Verschlüsslung erwünscht ist. Traefik kümmert sich darauf hin eigenständig um die Ausstellung des nötigen Zertifikats. Die TLS-Konfiguration lässt sich auch bei NGINX detailliert einstellen. Hier fehlt jedoch die direkte Anbindung an einen Dienst wie Let's Encrypt. Zertifikate können auf anderem Wege automatisch generiert werden, der Pfad zu den einzelnen Zertifikaten muss NGINX jedoch für jede Domain bekannt geben werden.

Aufgrund seiner Spezialisierung auf Container, der Möglichkeit einer automatischen Erkennung der Routen und der einfachen Automatisierung der Zertifikatserstellung kommt in dieser Arbeit Traefik als Ingress-Routing-Lösung zum Einsatz.

[2] `https://containo.us/traefik/`
[3] `https://www.nginx.com/`
[4] `https://letsencrypt.org/`

4.4 Projekt-Architektur im Detail

Das Projekt wird wie folgt in Services aufgeteilt:

- Web-Interface
- User-Service
- Course-Service
- Environment-Service
- Einzelne Environments / Übungen

Dabei findet die Kommunikation der einzelnen Services untereinander per REST-Schnittstellen statt (siehe hier Abbildung 4.1). Dabei findet das Ingress-Routing wie in Abbildung 4.2 statt. Im Folgenden werden die einzelnen Services im Detail beschrieben.

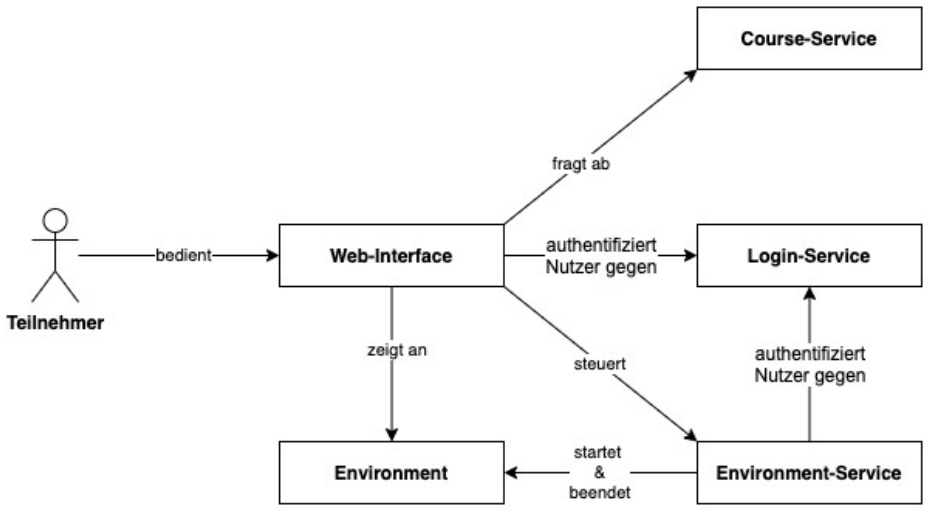

Abbildung 4.1: Projekt-Architektur und Interaktionen der Services

4.4.1 Web-Interface

Das Web-Interface bietet den zentralen Ausgangspunkt für die Teilnehmer einer Schulung. Über diesen im Browser verfügbaren Dienst erhalten Teilnehmer Zugriff auf alle nötigen Funktionen [R1]. Dies beinhaltet den Zugang zu Schulungsmaterialien und den einzelnen Kursen sowie deren praktischen Aufgaben. Das Starten einer Lernumgebung ist hier ebenso möglich wie deren Bedienung. Auch simple Aufgaben können in diesem Kontext direkt durch die vorhandene Javascript-Umgebung im Browser bereitgestellt werden.

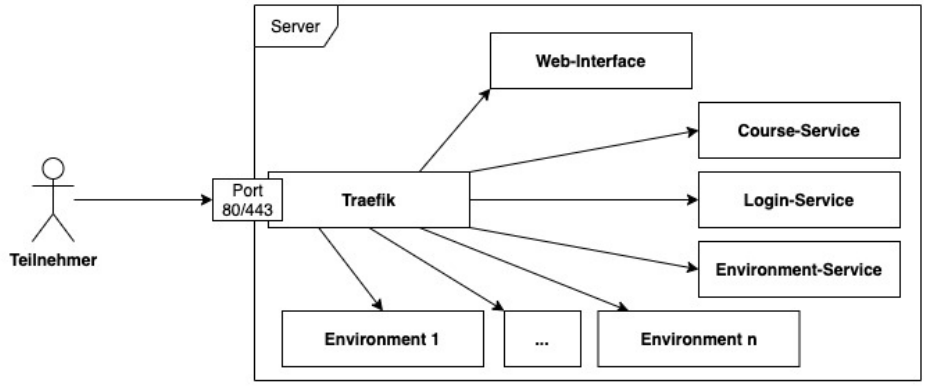

Abbildung 4.2: Ablauf des Ingress-Routing in der Projekt-Architektur

Diese werden zur Laufzeit auf den Geräten der Teilnehmer ausgeführt und entlasten somit auch den Server [R5].

4.4.2 User-Service

Der User-Service dient dem Verwalten von Teilnehmer-Profilen sowie Nutzerrechten. Sollte zukünftig eine Kommerzialisierung der Plattform in Betracht gezogen werden, könnte über diesen Dienst zudem eine Abrechnung integriert werden. Während der Entwicklung des Prototyps wurde auf all diese Funktionen verzichtet. Vorerst findet die Wahl des Nutzernamen ohne Prüfung direkt durch den Teilnehmer statt. Die Teilnahme findet derzeit folglich ungeprüft unter Nutzung eines echten Namens oder eines Pseudonyms statt. Auch die API-Endpunkte nehmen in dieser Umsetzung keine weitere Prüfung vor. Bei der Umsetzung ist die spätere Anbindung eines User-Services dennoch berücksichtigt worden.

4.4.3 Course-Service

Der Course-Service ist vorgesehen, um das Web-Interface mit Lernmodulen zu versorgen. Dieser kann als Quelle verwendet werden, um dem Web-Interface eine Liste mit Kursen bereitzustellen. Ebenso kann er die nötigen Definitionen für einzelne Aufgaben an den Environment-Service weiterreichen. Im Rahmen eines zentralen Deployments der Plattform, an der unterschiedliche Trainer ihre Kurse anbieten können, wäre eine Verwendung dieses Services ebenfalls denkbar. Dabei müsste jeder Trainer nur eine Instanz des Course-Service bereitstellen, die wiederum nur seine Lernmodule anbietet. Somit wird der Aufwand für Trainer weiter reduziert.

Auch die Entwicklung neuer Kurse lässt sich mit Hilfe dieser Struktur vereinfachen. Durch eine Instanz des Course-Service, der nur die aktuellen Entwicklungsstände enthält und nur einer vorbestimmten Nutzergruppe

zugänglich ist, ließen sich neue Kursinhalte im Kontext der vollwertigen Plattform austesten, ohne dabei weitere vollständige Instanzen der gesamten Plattform zu diesem Zweck starten zu müssen.

Durch das dynamische Hinzufügen weiterer Übungen im Rahmen eines Course-Service kann die gewünschte Erweiterbarkeit [R8] abgedeckt werden. In dieser Arbeit bleibt der Course-Service jedoch ein theoretisches Konstrukt, da während der Entwicklung des Prototyps die Anzahl der Kurse überschaubar bleiben wird. Die später gewählte Technik des Web-Interfaces lässt ein eventuelles Einfügen eines oder auch mehrerer Course-Services dennoch zu. Die Einbindung weiterer Lernmodule findet somit vorerst nur manuell über das Eintragen der Kurse direkt in das Web-Interface statt.

4.4.4 Einzelne Environments

Bei den Environments handelt es sich um die Umgebungen, die dem Teilnehmer im Rahmen einer Übung bereitgestellt werden. Sie enthalten alle nötigen Abhängigkeiten einer Aufgabe, egal ob es sich dabei um Code, Executables oder andere Dateien handelt. Die Environment eines Nutzers soll unabhängig von allen anderen Teilnehmern arbeiten können und vom Trainer einmalig, nicht auf den einzelnen Teilnehmer zugeschnitten, vorbereitet werden [R3]. Durch eine Verschiebung der einzelnen Aufgaben in ihre eignen Dienste kann die Programmiersprache freigewählt werden. Hiermit wird die Anforderung [R2] erfüllt. Dabei muss lediglich sichergestellt werden, dass die Aufgabe eine Web-Oberfläche anbietet, welche ihrerseits in das Web-Interface eingebunden werden kann [R1]. Die Beschreibung einer Environment findet in einer Template-Datei statt. Eine Anpassung dieses Templates findet dynamisch während des Starts einer Environment durch den Environment-Service statt.

4.4.5 Environment-Service

Der Environment-Service übernimmt die Aufgabe, Environments für die Teilnehmer zu starten und diese dem Web-Interface zugänglich zu machen. Hierzu passt er die Template-Datei an und fügt die nötigen Konfigurationen für das Ingress-Routing hinzu. Durch diese Automatisierung können Nutzer ohne weiteres Eingreifen eines Trainers ihre Environment starten [R3]. Sollten viele Teilnehmer gleichzeitig versuchen mit Environments zu arbeiten, kann dies die Ressourcen eines einzelnen Server ausreizen. Hier spielt die Skalierbarkeit erneut eine Rolle. Durch das Starten einer weiteren Instanz auf einem anderen Server können mehr Teilnehmer gleichzeitig auf der Plattform arbeiten [R4]. Als weitere Maßnahme zur Ressourcen-Optimierung überwacht der Environment-Service zudem die Lebensdauer der Environments. Nach einer vom Administrator festgelegten Zeit werden Environments durch den Service automatisch heruntergefahren.

4.5 Implementierung des Environment-Services

Die Umsetzung des Environment-Services findet in Form eines NodeJS-Projektes[5] statt. Hierbei kommt Javascript, genauer das Javascript Super-Set Typescript[6] zum Einsatz. Einer der großen Vorteile an dieser Stelle ist, dass Typescript im Gegensatz zu Javascript typsicher ist. Das Einführen von Coding Conventions wird hierdurch ebenfalls erleichtert. Diese lassen sich mit Hilfe von eslint[7] automatisch prüfen sowohl in der Entwicklungsumgebung als auch in einer Pipeline. Vor der Verwendung in einem Produktivsystem wird der Code zu Javascript transpiliert[8].

Im Rahmen von Node stehen verschiedene Frameworks zur Erstellung eines Web-Servers bereit. Der Web-Server wird benötigt, damit der Environment-Service eine HTTP-Schnittstelle für das Web-Interface bereitstellen kann. Dabei bietet Node selbst eine rudimentäre HTTP-API an. Diese hat jedoch nur einen deutlich reduzierten Funktionsumfang gegenüber der im folgenden betrachteten Lösung. In dieser Arbeit wird aufgrund der dort vorhandenen Modularisierung und Erweiterbarkeit NestJS[9] eingesetzt. NestJS liefert bereits einiges an Vorbereitung bezüglich Konfiguration und Tests. Hierdurch wird die Entwicklung eines Protoyps beschleunigt.

Die Tests in diesem Projekt beschränken sich derzeit auf die Hilfsfunktionen. Eine spätere Ausweitung der Tests inklusive end-to-end (e2e)-Tests sind möglich. Genauso wie die Überprüfung der Docker-Befehle/Aufrufe, entweder durch einen Mock-up-Service oder das aufwendigere Überwachen und Prüfen der laufenden Container. Auch diese können in der Pipeline automatisiert werden. Gerade im Sinne der Software Qualität ist die Automatisierung der Tests eine sinnvolle Maßnahme [32].

Als Format der Template-Dateien wurde das docker-compose-Format gewählt. Als besondere Konvention wird dem Format dabei hinzugefügt, dass ein Service mit dem Namen “main“ in der Datei existieren muss. Ein Beispiel für eine solche Environment-Template-Datei ist in Listing 4.1 dargestellt. Hierdurch kann der Environment-Service die nötigen dynamischen Anpassungen an den jeweiligen Teilnehmer vornehmen, wie in Listing 4.2 zu sehen.

```
1 version: '3'
2 services:
3   main:
4     image: 'theiaide/theia-cpp'
```

Listing 4.1: Beispiel einer Environment-Template-Datei

[5] https://nodejs.org/
[6] https://www.typescriptlang.org/
[7] https://eslint.org/
[8] transpilieren beschreibt die Überführung in eine andere Sprache
[9] https://nestjs.com/

```yaml
version: '3'
services:
  main:
    image: 'theiaide/theia-cpp'
    networks:
      - traefik-proxy
      - default
    labels:
      - silver-book.user=testuser
      - silver-book.course=testcourse
      - traefik.enable=true
      - traefik.docker.network=traefik-proxy
      - traefik.http.routers.silver-book-testcourse-testuser.rule
        ↪ =Host(`testcourse-testuser.env.example.com`)
      - traefik.http.routers.silver-book-testcourse-testuser.
        ↪ entrypoints=websecure
      - traefik.http.routers.silver-book-testcourse-testuser.tls.
        ↪ certresolver=le
networks:
  traefik-proxy:
    external: true
```

Listing 4.2: Individualisierte Environment-Datei aus Listing 4.1

Dabei wird hier ausgenutzt, dass es sich bei den Compose-Dateien um YAML-Dateien handelt, welche in ein Key-Value-Objekt umgewandelt werden können. Nun ist es möglich, über die entsprechenden Keys auf den main-Container zuzugreifen. Dem main-Container werden ein Netzwerk und diverse Labels hinzugefügt. Beim Start des Containers kann er nun automatisch von Traefik erkannt und je nach Konfiguration mit einem SSL-Zertifkat versehen werden. Im Anschluss steht er über den generierten Hostnamen zur Verfügung.

Die Verwendung der docker-compose-Datei ermöglicht es zudem, mehr als einen Container in einem Projekt zu definieren. Somit steht auch der Erstellung von Templates für komplexere Umgebungen mit mehr als einem System nichts im Weg [R6]. Der Environment-Service erhält zudem eine Option über die sich auch anonymisierte Domains konfigurieren lassen.

Der Environment-Service enthält im Rahmen dieser Arbeit drei wichtige Funktionen: das manuelle Starten, das manuelle Stoppen und das automatische Stoppen einer Environment. Ein manueller Start-Aufruf erwartet als Eingabeparameter den Nutzernamen und den gewünschten Kurs. Diese Informationen werden genutzt, um die Template-Datei auszuwählen und zu individualisieren. Dabei werden die einzelnen Container mit Labels versehen. Ein manueller Stopp-Aufruf erhält die gleichen Parameter wie zuvor der

manuelle Start. Daraufhin wird die Liste der laufenden Container abgerufen und nach den notwendigen Labels gefiltert. Die betroffenen Container werden beendet. Das automatische Stoppen prüft in konfigurierbaren Abständen alle laufenden Container der Plattform. Die einzelnen Container einer Environment erhalten hierzu beim Start ebenfalls ein Label mit einem Verfallsdatum.

Begleitend zum Projekt wurde auch eine Pipeline angelegt. In dieser wurde sowohl das Linting als auch der Build-Prozess des Container-Images aufgenommen. Auch das darauf folgende Deployment in eine Staging-Umgebung wurde hier automatisiert. Hierzu wurde ein Buildfile (Dockerfile) zur Erstellung des Container-Images angelegt. Das Dockerfile nimmt als Ausgangspunkt ein offizielles Node-Container-Image[10], kopiert den Source-Code hinein, lädt alle Projekt-Abhängigkeiten und transpiliert das Projekt. Das fertige Image wird von der Pipeline in eine Container-Registry hochgeladen. Außerdem wurde eine docker-compose-Datei für das Deployment angelegt, die wiederum das angelegte Image verwendet, entsprechend konfiguriert und startet. Die Pipeline ist dabei so konfiguriert, dass jeder Push auf den master-Branch in der Versionsverwaltung das Deployment aktualisiert. Hierdurch lässt sich der aktuelle Stand der Entwicklung leicht einsehen.

4.6 Implementierung des Web-Interfaces

Neben der Erstellung der Website mit einfachen HTML-Tags bietet sich die Wahl eines Frameworks zur Erstellung der Inhalte an. Große Frameworks sind derzeit beispielsweise Angular[11] und React[12]. In dieser Arbeit wird dabei React zum Einsatz kommen. Dabei handelt es sich um eine persönliche Präferenz, die zu dem schnellen Voranschreiten des Prototyps beitragen soll. Wie auch im Environment-Service wird hier Typescript verwendet, um die zuvor genannten Vorteile gegenüber Javascript zu verwenden.

Zusätzlich kommt der Site-Generator Gatsby[13] zum Einsatz. Gatsby erlaubt es, schnelle und optimierte Websites auf Basis von React umsetzten. Dabei findet ein Teil der Evaluierung des React-Codes bereits zur Buildtime statt, um dynamische Inhalte nicht erst zur Laufzeit herunterladen zu müssen. Deshalb eignet es sich hervorragend, um später die Darstellung der Kurse automatisch zu generieren. Dabei bietet Gatsby ein weites Spektrum an Quellen an um die Inhalte dynamisch zu laden. Beispiele sind hier das lokale Dateisystem, Datenbanken oder auch API-Endpunkte. Der im Konzept vorgesehene Course-Service ließe sich auf diese Weise ebenfalls abfragen.

[10] `https://hub.docker.com/_/node`
[11] `https://angular.io/`
[12] `https://reactjs.org/`
[13] `https://www.gatsbyjs.org/`

Durch den Einsatz von CSS (Cascading Style Sheets) lässt sich die Plattform auch an die Corporate Identity Richtlinien anpassen [R7].

Ein späteres Einbinden von Tests, beispielsweise mit Tools wie Cypress[14], ist ratsam. Mit diesen Mitteln lassen sich sowohl die Bedienung der Oberfläche simulieren als auch die optische Persistenz zwischen Versionen testen. Da während der Entwicklung des Prototyps nahezu das gesamte Web-Interface häufigen Änderungen unterliegt, wurde an dieser Stelle auf die Einführung einer Testumgebung verzichtet.

Die Oberfläche durchlief bei der Umsetzung diverse Iterationen. Besonders hervorzuheben ist dabei die Verwendung eines iframes[15], um die Web-Oberfläche der Environment direkt zu integrieren. Der Teilnehmer kann dabei selbst wählen, ob er die Darstellung innerhalb des Web-Interafaces bevorzugt oder gegebenenfalls auf einen größeren Arbeitsbereich ausweicht. Hierzu lässt sich die Environment in einem eigenen Browser-Fenster öffnen.

Wie auch im Environment-Service wird eine Pipeline verwendet. Die automatisierten Funktionen sind dabei nahezu identisch. Abweichend zum vorherigem Projekt wird in einem Docker-Container durch Gatsby die statische Version des Web-Interfaces generierte. Die einzelnen Web-Dateien werden im Anschluss in ein NGINX-Container-Image[16] eingefügt. NGINX übernimmt dabei als Web-Server das Ausliefern der generierten Dateien an die Web-Browser. Traefik kümmert sich hierbei um die Weiterleitung der Anfragen an den NGINX-Container.

[14] `https://www.cypress.io/`

[15] `https://www.w3schools.com/tags/tag_iframe.asp`

[16] `https://hub.docker.com/_/nginx`

Kapitel 5

Lernmodule

In diesem Kapitel wird die Umsetzung der beiden Lernmodule zur Kryptographie und zu Schwachstellen beschrieben. Dabei wurde das Modul zur Kryptographie lediglich als Erweiterung des Web-Interfaces umgesetzt. Das Modul zu Schwachstellen hingegen wird in Form von Environments umgesetzt um komplexere Aufgaben zu ermöglichen.

5.1 Modul zur Kryptographie

Im Rahmen dieser Arbeit wurde die Umsetzung der Caesar-Chiffre gewählt. Das Verfahren ist in seinem Ablauf recht simpel und für die Umsetzung sind keine weiteren Abhängigkeiten oder Environments nötig. Es kann ohne größeren Aufwand im Browser, im Kontext von Javascript, abgearbeitet werden. Aus diesem Grund werden diese Inhalte direkt als Teil des Web-Interfaces in Gatsby umgesetzt. Dazu wird eine React-Komponente angelegt, welche auf der Seite des Kurses importiert wird. Die Komponente enthält sowohl die verwendeten Berechnungen und die Logik zur Durchführung des Verfahrens als auch die erklärenden Texte und die grafische Benutzeroberfläche.

Die vollständige Verlagerung dieser Aufgabe in den Browser des Teilnehmers ermöglicht, neben einer höheren Reaktionsgeschwindigkeit der Inhalte, auch die Möglichkeit der Nutzung durch mehr Teilnehmer zur gleichen Zeit. Dies liegt zum einen daran, dass keine weitere Kommunikation zwischen Client und Server nötigt ist und zum anderen wird bei der Berechnung der Algorithmen die Hardware der Teilnehmer verwendet. Folglich müssen auch keine Server-Ressourcen aufgebracht werden, um jedem Teilnehmer eine Environment zu stellen. Somit ermöglicht der Teilnehmer mit seiner eigenen Rechenleistung eine Entlastung des Servers [R5].

Die hier erstellte Komponente wird in eine einzelne Klasse ausgelagert und in der Projektstruktur eigenständig, außerhalb der Struktur des Kurses, abgelegt. Die einzelnen Komponenten lassen sich ähnlich wie Docker-Images wiederverwenden. Sie könnte somit im späteren Verlauf auch von anderen

Kursen durch einen Import der entsprechenden Datei übernommen werden. Hierdurch werden unnötige Redundanzen verhindert und ein schnelleres Anlegen von Kursen ermöglicht, vorausgesetzt es existieren bereits entsprechende Komponenten.

Über ein Eingabefeld kann der Teilnehmer die Verschiebung im Alphabet auswählen. Diese Verschiebung wird in React als State hinterlegt. Bei einer Änderung dieses States kann durch React automatisch eine Aktualisierung der Anzeige vorgenommen werden. Über eine triviale Funktion wird ein Mapping vorgenommen, welches die Verschiebung eines Buchstabens um die zuvor gewählte Anzahl umsetzt. Über ein zweites Eingabefeld kann der Nutzer eine Nachricht eingeben, die mittels dieser Funktion verschlüsselt wird. Um dem Teilnehmer das Vorgehen besser verständlich zu machen, wird zudem eine Tabelle erzeugt, die das gesamte Alphabet und die entsprechende Substitution anzeigt. Dies soll es ihm ermöglichen, durch Experimentieren und beiliegende Erklärungen, spielerisch das Verfahren kennenzulernen. Trainer können diesem Beispiel die Umsetzung von Client-basierten-Aufgaben entnehmen.

Auf gleiche Art und Weise lassen sich vergleichbare Lerninhalte umsetzten. Die Grenzen eines solchen Moduls sind jedoch dort erreicht, wo Javascript an seine Grenzen stößt. In diesem Fall sind komplexere Umgebungen gefragt, die in dieser Arbeit als Environment während des Konzepts geplant wurden. Diese kommen im Modul zu Schwachstellen zum Einsatz.

5.2 Modul zu Schwachstellen

Das Schwachstellen-Modul besteht aus drei Übungen die das Konzept der Environments stufenweise weiter ausbauen.

5.2.1 Die Automatisierung

Die Environments wurden als eigenständige Projekte in der Versionsverwaltung angelegt. Somit kann jedes der Projekte ebenfalls mit einer eigenen Pipeline zur Automatisierung ausgestattet werden. So lässt sich bei einer Änderung das Image für die jeweilige Environment schnell anpassen und an einem zentralen Ort bereitstellen. Die resultierenden Container-Images können abschließend in der Container-Registry abgelegt und später abgerufen werden. Um die Aufgabe in der Schulungsplattform zu aktualisieren, muss lediglich das entsprechende Image mit dem Tag *latest* heruntergeladen werden. Durch das Anbringen weiterer Tags an die Images lassen sich auch alte Versionsstände der Aufgaben in die Plattform einbinden. Hierzu ist lediglich eine Anpassung der Template-Datei notwendig.

5.2.2 Terminal Environment zu Buffer-overflow-Attacken

Zur Demonstration eines Buffer-overflows ist ein simples C-Programm erstellt worden. Um dem Teilnehmer Zugriff auf das entstehende Executable zu geben, soll ihm ein Terminal im Browser angeboten werden. Nach diversen Iterationen mit verschiedenen Web-Terminal-Lösungen fiel die Wahl auf das tsl0922/ttyd-Image [1] als Ausgangspunkt. Dieses Image bietet ein vollwertiges Terminal-Fenster über den Browser an. Das Programm wird mit den nötigen Parametern kompiliert und dem Image als weiterer Layer hinzugefügt.

Der Teilnehmer soll bei dieser Aufgabe durch Experimentieren herausfinden, wie sich ein Buffer-overflow auf den Programmablauf auswirken kann. Dies soll dem Teilnehmer einen ersten Hinweis zum Verständnis eines Buffer-overflows geben. Trainer können dieser Environment entnehmen, wie ein simples Terminal-Programm in einen Kurs aufgenommen werden kann.

5.2.3 Entwicklungsumgebung zu Buffer-overflow-Attacken

Nach der Demonstration dieser Schwachstelle soll dem Teilnehmer auch die Gelegenheit gegeben werden, sich selbst einen Eindruck vom Code dieser Schwachstelle zu verschaffen. Die hierzu erstellte Environment enthält sowohl den ursprünglichen Code als auch zwei Skripte. Der Source-Code bleibt dabei zur vorherigen Aufgabe identisch. Die beiden Skripte enthalten die nötigen Befehle, um das C-Programm einmal mit und einmal ohne die im Compiler integrierten Sicherheitsmaßnahmen gegen Buffer-overflows zu kompilieren.

Um dem Teilnehmer die Arbeit am Code zu erleichtern, soll ihm hier eine Entwicklungsumgebung bereitgestellt werden. Bei der Umsetzung wurden zwei Web-IDEs betrachtet: zum einen Visual Studio Code[2], ein Code-Editor von Microsoft, der sich durch Plugins sehr anpassbar zeigt; zum anderen Theia[3], ein Projekt, das sich an Visual Studio Code orientiert, dabei aber auch auf die Ausführung in Cloud-Umgebungen ausgelegt ist.

Beide Projekte sind mit aktuellen Web-Technologien umgesetzt und können ihre Dienste im Web-Browser anbieten. Theia ist ebenfalls als Container-Image und mit einer vorinstallierten C/C++ Umgebung verfügbar[4]. Dies erleichtert die Umsetzung der Aufgabe erheblich. Folglich wird für die Umsetzung an dieser Stelle das Theia-Container-Image inklusive der C-Umgebung verwendet.

Der Teilnehmer kann in dieser Umgebung einen Einblick in die Compiler-Einstellungen erlangen. Desweiteren kann er eigenständig experimentieren, wie sich die Anordnung der Variablen im Code auf diese Schwachstelle auswirkt und weitere Versuche unternehmen, diese Schwachstelle ohne

[1] `https://hub.docker.com/r/tsl0922/ttyd`
[2] `https://code.visualstudio.com/`
[3] `https://theia-ide.org/`
[4] `https://hub.docker.com/r/theiaide/theia-cpp`

die Hilfe des Compilers abzusichern. Trainer können dieser Environment entnehmen, wie eine komplexere IDE in einen Kurs aufgenommen werden kann.

5.2.4 Environment für SQL-Injections

Das Setup für eine SQL-Injection ist um einiges komplexer als für die bisherigen Environments. Eine Umsetzung im Browser selbst über SQLite-Datenbanken wäre ebenfalls denkbar gewesen. Da Web-SQL zum derzeitigen Stand jedoch nicht von Browsern wie Firefox und Safari unterstützt wird, findet die Umsetzung mit Environments statt [9]. Zudem ermöglicht dies die Demonstration komplexerer Environments bestehend aus mehreren Containern.

Im Rahmen dieser Arbeit wurde eine minimalistische Web-Oberfläche in PHP umgesetzt. Der PHP-Code enthält eine Suchfunktion, die intern eine SQL-Anfrage auslöst. Diese ist bewusst anfällig für SQL-Injections angelegt. Durch die Verwendung von Steuerzeichen in der Suche der Web-Oberfläche kann die Anfrage missbraucht werden, um beliebige Datenbankbefehle abzusetzen. Es können beispielsweise versteckte Einträge sichtbar gemacht oder auch Zeilen gelöscht werden.

Insgesamt sind bei der Umsetzung drei Container im Einsatz (siehe Abbildung 5.1). Der erste ist ein NGINX-Webserver, der die Anfragen des Browsers entgegennimmt. Dieser wird in der Template-Datei als main-Service hinterlegt, um später durch Traefik verfügbar zu sein. Der zweite Container ist für die Auswertung der PHP-Skripte zuständig. Die entsprechenden Anfragen werden vom ersten Container weitergeleitet. Von diesem Container werden die SQL-Anfragen an den Datenbank-Server weitergeleitet. Dieser befindet sich im dritten Container und stellt eine SQL-Datenbank zur Verfügung. Durch die Wahl von Service-Namen in der docker-compose-Datei (Listing 5.1) können die Dienste sich gegenseitig finden und miteinander kommunizieren.

Der Nutzer erhält in dieser Environment die Gelegenheit, frei mit dem Thema der SQL-Injection zu arbeiten. Die Anzeige des zusammengesetzten SQL-Kommandos zusammen mit dem Ergebnis soll dabei das Kennenlernen dieser Schwachstelle erleichtern. Da dem Nutzer ein Knopf zur Wiederherstellung der Datenbank zur Verfügung steht, können problemlos Befehle ausgeführt werden, die Einfluss auf die Tabellenstruktur nehmen. Sollte es dennoch zu Fehlern kommen, kann die Environment vom Teilnehmer neu gestartet werden, um in den Ausgangszustand zurückzukehren. Trainer können dieser Environment entnehmen, wie Aufgaben bestehend aus mehreren Systemen beziehungsweise Containern umgesetzt werden können.

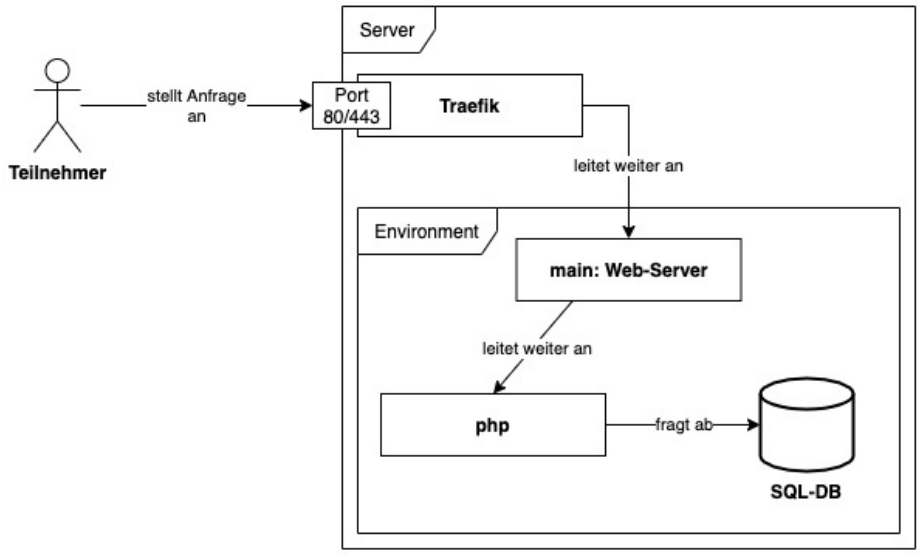

Abbildung 5.1: Interaktionen und Routing der SQL-Injection-Environment

```yaml
version: '3'

services:
  main:
    image: example.com/sql-injection/nginx
    depends_on:
      - php

  php:
    image: example.com/sql-injection/php

  mysql:
    image: mysql:5.7
    environment:
      MYSQL_DATABASE: cookie
      MYSQL_ROOT_PASSWORD: CourseSQLsecret
```

Listing 5.1: Template-Datei der SQL-Injection-Environment

Kapitel 6

Zusammenfassung und Ausblick

Zuletzt sollen in diesem Kapitel die Ergebnisse und Erkenntnisse dieser Arbeit zusammengefasst und kritisch beleuchtet werden. Der anschließende Ausblick geht vorrangig auf die Möglichkeiten der Erweiterung sowie den Umgang mit der in dieser Arbeit erarbeiteten Plattform ein.

6.1 Zusammenfassung

IT-Security spielt eine zunehmend wichtiger werdende Rolle. Derzeit finden Schulungen in diesem Bereich jedoch oftmals auf theoretischer Ebene statt. Der Einsatz von interaktiven Inhalten könnte hier zu einem größeren Wissensgewinn führen. Hindernisse dabei sind häufig Einschränkungen der Software-Umgebung von Schlungunsteilnehmern und der Aufwand für die Trainer der Schulungen.

Die Zielsetzung, dass Teilnehmern im Rahmen von IT-Security-Schulungen praktische Aufgaben zugänglich gemacht werden können, wurde in dieser Arbeit erreicht. Die dazu aufgestellten Anforderungen der Gap-Analyse wurden dabei alle berücksichtigt. Die umgesetzte Plattform ermöglicht den Teilnehmern eine installtionsfreie Teilnahme an einer IT-Security-Schulung unter einmaligem Aufwand für den Trainer. Zudem wurden zwei Lernmodule umgesetzt, welche diese Aspekte an Beispielen demonstrieren. Die Lernmodule sind als eigenständige Projekte umgesetzt und können auch außerhalb der Plattform verwendet werden.

Der zeitlich begrenzte Rahmen setzte der vollständigen Umsetzung des Konzeptes Grenzen. Diverse Punkte wurden daher in dieser Arbeit bewusst nur minimalistisch umgesetzt. Die Plattform befindet sich derzeit im Zustand eines Prototyps. Es ist dennoch eine solide Grundlage geschaffen worden, auf der sich eine Weiterentwicklung begründen lässt. Mit überschaubarem Aufwand kann ein Feinschliff erfolgen, um die Plattform aus dem Status eines

Prototyps in ein Produktivsystem zu überführen. Während der Umsetzung ergaben sich zudem einige Stellen, die Möglichkeiten zur Erweiterung des Konzepts und der Funktionen bieten. Diese werden im folgenden Ausblick aufgezeigt.

6.2 Ausblick

Allem voran ist anzumerken, dass eine Überprüfung stattfinden muss, ob durch die praktischen Übungen auf der Plattform ein tatsächlicher Lernzuwachs bei den Teilnehmern ergibt. Ohne eine solche Überprüfung lassen sich keine pauschalen Aussagen über die Effektivität der unterschiedlichen Module und Möglichkeiten treffen [38].

Vor einem ersten der Öffentlichkeit zugänglichem Deployment sollte zunächst ein verlässlicher User-Service umgesetzt werden. Dieser bedarf jedoch der Anpassung an den jeweiligen Betreiber der Plattform, da eine Nutzerverwaltung auf etliche Arten bereits vorliegen kann. So wären Anbindungen an Internetplattformen wie beispielsweise Facebook, Google, Github, Gitlab, genauso möglich wie ein eigenes OAuth-Verfahren, einen zentralen Single-Sign-on im Firmennetzwerk oder eine vollständige Eigenentwicklung.

Die derzeitige Umsetzung beruht ausschließlich auf Docker-Containern und setzt einen vorkonfiguriertes Traefik-Deployment zum Ingress Routing voraus. Eine Umstellung auf Kubernetes würde hauptsächlich eine Änderung der Docker-compose-Dateien zu Kubernetes-Deployments bedeuten. Im Rahmen von Kubernetes stehen standardisierte Schnittstellen für Ingress Routing zur Verfügung. Traefik steht hier ebenfalls als Drop-In-Möglichkeit zur Verfügung. Darüber hinaus bliebe das Vorgehen der Plattform weitestgehend unverändert. Zeitgleich würde sich durch eine solche Umstellung, die Wahl eines Cloud-Anbieters oder der Betrieb eines eigenen Kubernetes-Cluster anbieten. Die Wahl von Containern ermöglicht hier ein flexibles und Anbieter-unabhängiges Vorgehen. Eine solcher Schritt würde die Skalierbarkeit zudem wesentlich verbessern.

Um eine portable Version dieser Schulungsplattform aufzubauen, wäre ein Deployment auf Raspberry Pis[1] denkbar. Beachtet werden müsste in diesem Fall die stark eingeschränkte Ressourcenmenge des Raspberry Pi. Durch das Starten des Environment-Services auf mehreren Raspberry Pis könnte dabei ein Stück weit Abhilfe geschaffen werden. Dank der kleinen und leichten Bauform, dem geringen Stromverbrauch und dem verbauten WiFi, welches auch zum Anlegen eines Ad-hoc-WLANs verwendet werden kann, sind diese hervorragend für den Transport geeignet. Sinnvoll wäre diese Umsetzung in Verbindung mit dem zuvor vorgeschlagenen Kubernetes-Cluster. Mit diesen Mitteln könnte ein Koffer gebaut werden, der am Tag der Schulung in der Raummitte aufgestellt wird und den Teilnehmern ein

[1] Ein Raspberry Pi ist ein kleiner Einplatinen Computer mit ARM Prozessor

WLAN sowie die Schulungsplattform bereitstellt. Damit wird eine noch größere Unabhängigkeit von den Rahmenbedingungen, wie beispielsweise den Internetzugängen, geschaffen. Um ein gültiges Wildcard-Zertifikat und eine Domainauflösung sollte sich zuvor gekümmert werden. Auch hier könnten weitere Container zum Einsatz kommen.

Eine Erweiterung durch die Verbindung mit einem Capture-the-Flag-Wettbewerb als Fortschrittskontrolle und Gamification-Ansatz könnte ebenfalls vielversprechend sein. Eine eventuelle Kombination mit der in 3.1 vorgestellten Arbeit/Plattform könnte hier als Ausgangpunkt dienen. Dabei könnte beim ersten Start des Containers eine geheime Flag generiert und versteckt werden, die auf dem jeweiligen Nutzernamen beruht und nur vom System bekannt und geprüft werden kann. Vergleichbar wäre dies mit dem Vorgehen eines One-Time-Password (OTP). Dies unterbindet zeitgleich die Weitergabe der Flag an andere Teilnehmer. Lediglich die Weitergabe des Lösungsweges ist möglich. Dies zwingt die Teilnehmer folglich weiterhin zu der erwünschten Anwendung der Techniken. Die Annahme der Flag könnte entweder mit Hilfe eines zusätzlich konzipierten Micro-Services oder im Rahmen des User-Service geschehen.

Die in dieser Arbeit umgesetzten Lernmodule beruhen derzeit auf Techniken, die dank JavaScript entweder direkt für den Browser entwickelt oder im Voraus auf die Verwendung mit Chromium ausgelegt wurden. Durch das Integrieren von Tools wie xpra und unter Verwendung von HTML5 könnten auch herkömmliche Linux-GUI-Programme in die Plattform aufgenommen werden. Dabei würde eine Umleitung der X11-Umgebung des Programms beziehungsweise des Betriebssystems stattfinden. Dies würde den Zugang zu einer noch wesentlich größeren Menge an Software ermöglichen. So könnten auch Schulungen zu Software über den Browser stattfinden, ohne vorherige Installation und mit der Option einer vorkonfigurierten und zurücksetzbaren Umgebung.

Zuletzt soll hier angemerkt werden, dass das ausgearbeitete Schulungssystem sich nicht nur für den Einsatz im IT-Security-Bereich eignet. Durch das Hinzufügen weiterer Module aus anderen Bereichen kann die Plattform für diverse andere Demonstrationen und Aufgaben genutzt werden. So wären Einführungen in Command-Line-Tools und Programmiersprachen denkbar. Auch um mit der Umsetzung von Software-Qualität zu experimentieren, ließen sich Umgebungen schaffen.

Literaturverzeichnis

[1] J. Arundel and J. Domingus. *Cloud Native DevOps mit Kubernetes : Bauen, Deployen und Skalieren moderner Anwendungen in der Cloud.* dpunkt.verlag, Heidelberg, 2019.

[2] A. Axelrod. *Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects.* Apress, Berkeley, CA, 2018.

[3] T. P. Baker. Buffer Overflows: Why are they a security problem? `https://www.cs.fsu.edu/~baker/opsys/notes/bufferoverflow.html`. letzter Zugriff: 12.08.2020.

[4] M. Barakat, C. Eder, and T. Hanke. An introduction to cryptography. *Timo Hanke at RWTH Aachen University*, 2018.

[5] A. Barashkov. Microservices vs. Monolith Architecture. `https://dev.to/alex_barashkov/microservices-vs-monolith-architecture-4l1m`. letzter Zugriff: 12.08.2020.

[6] R. Brünken, T. Seufert, and S. Zander. Förderung der Kohärenzbildung beim Lernen mit multiplen Repräsentationen: Fostering Coherence Formation in Learning with Multiple Representations. *Zeitschrift für pädagogische Psychologie*, 19(1/2):61–75, 2005.

[7] W. Brunotte. Security Code Clone Detection entwickelt als Eclipse Plugin. Masterarbeit, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2018.

[8] BSI. BSI Technische Richtlinie TR-02102-2: Kryptographische Verfahren: Empfehlungen und Schlüssellängen. Technical report, 2020.

[9] A. Deveria and L. Schoors. Can I use... Support tables for HTML5, CSS3, etc. `https://caniuse.com/#search=websql`. letzter Zugriff: 12.08.2020.

[10] Docker-Inc. What is a Container? `https://www.docker.com/resources/what-container`. letzter Zugriff: 12.08.2020.

[11] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. *Microservices: Yesterday, Today, and Tomorrow*, pages 195–216. Springer International Publishing, Cham, 2017.

[12] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina. Microservices: How To Make Your Application Scale. In A. K. Petrenko and A. Voronkov, editors, *Perspectives of System Informatics*, pages 95–104, Cham, 2018. Springer International Publishing.

[13] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. DevOps. *IEEE Software*, 33(3):94–100, 2016.

[14] W. Ertel and E. Löhmann. *Angewandte Kryptographie*. Hanser;, München, 2020.

[15] D. Euler. (Multi) Mediales Lernen-Theoretische Fundierungen und Forschungsstand. *Unterrichtswissenschaft*, 22(4):291–311, 1994.

[16] K. Frühauf, J. Ludewig, and H. Sandmayr. Qualitätssicherung. In *Software-Projektmanagement und-Qualitätssicherung*, pages 59–74. Springer, 1988.

[17] S. Fu, J. Liu, X. Chu, and Y. Hu. Toward a standard interface for cloud providers: The container as the narrow waist. *IEEE Internet Computing*, 20(2):66–71, 2016.

[18] E. Haugh and M. Bishop. Testing C Programs for Buffer Overflow Vulnerabilities. In *NDSS*. Citeseer, 2003.

[19] V. Krishna and A. Basu. Software Engineering Practices for Minimizing Technical Debt. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*, 2013.

[20] C. J. Langevin. Rising Demand for Private-Sector Cyber Skills.

[21] M. Leppänen, S. Mäkinen, M. Pagels, V. Eloranta, J. Itkonen, M. V. Mäntylä, and T. Männistö. The highways and country roads to continuous deployment. *IEEE Software*, 32(2):64–72, 2015.

[22] H. Li, L. Yu, and W. He. The Impact of GDPR on Global Technology Development. *Journal of Global Information Technology Management*, 22(1):1–6, 2019.

[23] M. C. Libicki, D. Senty, and J. Pollak. *Hackers wanted: An examination of the cybersecurity labor market*. Rand Corporation, 2014.

[24] A. Mallik. Man-in-the-middle-attack: Understanding in simple words. *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, 2(2):109–134, 2019.

[25] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos. Lightweight Cryptography for Embedded Systems – A Comparative Analysis. In J. Garcia-Alfaro, G. Lioudakis, N. Cuppens-Boulahia, S. Foley, and W. M. Fitzgerald, editors, *Data Privacy Management and Autonomous Spontaneous Security*, pages 333–349, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[26] S. Mühleisen. *Lernen verstehen*, pages 178–188. VS Verlag für Sozialwissenschaften, Wiesbaden, 2003.

[27] S. Newman. *Building microservices*. OR̂eilly, Beijing, 2015.

[28] J. Niebuhr. Konzeption und Umsetzung einer IDE in einem Docker-Container. Bachelorarbeit, Hochschule für Angewandte Wissenschaften Hamburg, 2017.

[29] R. Oppliger. Disillusioning Alice and Bob. *IEEE Security Privacy*, 15(5):82–84, 2017.

[30] N. Pohlmann. *Cyber-Sicherheit : das Lehrbuch für Konzepte, Prinzipien, Mechanismen, Architekturen und Eigenschaften von Cyber-Sicherheitssystemen in der Digitalisierung, Springer eBooks, Computer Science and Engineering*. Springer Vieweg, Wiesbaden, 2019.

[31] quoteinvestigator. Tell Me and I Forget; Teach Me and I May Remember; Involve Me and I Learn. `https://quoteinvestigator.com/2019/02/27/tell/#note-21846-1`. letzter Zugriff: 12.08.2020.

[32] K. Schneider. *Abenteuer Softwarequalität : Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement*, pages 199–200. dpunkt.verlag, Heidelberg, 2012. Abenteuer Software-Qualität, Das Q-Buch; 2. Auflage.

[33] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment. A multi-level framework to identify HTTPS services. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, page 241, 2016.

[34] P. Sommerlad. Reverse Proxy Patterns. In *EuroPLoP*, pages 431–458, 2003.

[35] Z. Su and G. Wassermann. The Essence of Command Injection Attacks in Web Applications. *SIGPLAN Not.*, 41(1):372–382, Jan. 2006.

[36] C. Thoms. Virtualisation-based Infrastructure for Practical IT-Security Exercises. Masterarbeit, Gottfried Wilhelm Leibniz Universität Hannover, 2020.

[37] J. Thonhauser. *Aufgaben als Katalysatoren von Lernprozessen: eine zentrale Komponente organisierten Lehrens und Lernens aus der Sicht von Lernforschung, allgemeiner Didaktik und Fachdidaktik.* Waxmann Verlag, 2008.

[38] B. Weidenmann. "Multimedia": Mehrere Medien, mehrere Codes, mehrere Sinneskanäle? *Unterrichtswissenschaft*, 25(3):197–206, 1997.

Abbildungsverzeichnis

Listings