Fakultät Informatik

Hochschule Reutlingen
Alteburgstraße 150
72762 Reutlingen

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science (B.SC.)
in Kooperation mit Novatec GmbH

API Security Testing mit OWASP Zap und RAML

Mert Yücel

Studiengang:	Wirtschaftsinformatik
Prüfer/in:	Prof. Dr. Martin Schmollinger, Prof. Dr. Muster Mustermann
Betreuer/in:	Anreas Falk, Jan Horak
Beginn am:	1. September 2018
Beendet am:	15. Januar 2019

Kurzfassung

REST-APIs sind heutzutage weit verbreitet und dank ihrer Einfachheit, Skalierbarkeit und Flexibilität werden sie weitgehend als Standardprotokoll für die Web-APIs angesehen. Es scheint plausibel anzunehmen, dass die Ära der Desktop-basierten Anwendungen kontinuierlich zurückgeht und im Zuge dessen, die Benutzer von Desktop- zu Web- und weiteren mobilen Anwendungen wechseln.

Bei der Entwicklung von REST-basierten Web-Anwendungen wird ein REST-basierter Web Service benötigt, um die Funktionalitäten der Web-Anwendung richtig testen zu können, da die gängigen Penetrationstest-Werkzeuge für REST-APIs nicht direkt einsatzfähig sind, wird die Sicherheit solcher APIs jedoch immer noch zu selten überprüft und das Testen dieser Arten von Anwendungen ist eine sehr große Herausforderung. Hauptsächlich für das erstmalige Testen ist es für den Betreiber von Webanwendungen sehr unüberschaubar. Verschiedene Werkzeuge, Frameworks und Bibliotheken sind dazu da, die Testaktivität automatisieren zu können. Die Nutzer wählen diese Dienstprogramme basierend auf ihrem Kontext, ihrer Umgebung, ihrem Budget und ihrem Qualifikationsniveau. Einige Eigenschaften von REST-APIs machen es jedoch für automatisierte Web-Sicherheitsscanner schwierig, geeignete REST-API-Sicherheitstests für die Schwachstellen durchzuführen.

Diese Bachelorarbeit untersucht wie die Sicherheitstests heutzutage realisiert werden und versucht qualitativ-deskriptiv aufzudecken, ob auf solche Sicherheitstests verlässlich sind. Es werden verschiedene existierende Tools verglichen, die das Testen von RESTful APIs unterstützen. Dann wird ihre Vor- und Nachteile herausgefunden und gegeneinander abgewägt. Es wird auch Gewißheit verschaffen, wie die technische Schwachstellen(wie z.B. Konzeptionsfehlern, Programmierfehlern und Konfigurationsfehlern) von Webanwendungen dargelegt. Parallel zu den Schwachstellen werden die jeweiligen Gefahren und Angriffspunkte für eine Webanwendung anhand von ”OWASP Top 10 - Risiken” erörtert.

Im Rahmen dieser Bachelorarbeit wird ein Plugin für das Open Source Werkzeug OWASP Zap entwickelt, um Schwachstellen automatisch zu untersuchen, um ein gewisses Maß an Sicherheit gewährleisten zu können. Dieses Plugin muss so implementiert werden, so dass die RAML-Dokumentationen importiert werden können, um automatisierte Penetrationstests ausführen zu können. Es wird auch eine Möglichkeit geschaffen, dass das Plugin in OWASP Zap sowohl über die UI, aber auch über Kommandozeile ansprechbar ist, um dies auch in CI-Umgebungen einbinden zu können.

Danksagung

Bla bla..

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

ER error rate. 37

FR Fehlerrate. 37

IKT Informations- und Kommunikationstechnologie. 3

RDBMS Relational Database Management System. 37

1. Einleitung

Bei LaTeX werden Absätze durch freie Zeilen angegeben. Da die Arbeit über ein Versionskontrollsystem versioniert wird, ist es sinnvoll, pro *Satz* neue Zeile im `.java`-Dokument anzufangen. So kann einfacher ein Vergleich von Versionsständen vorgenommen werden.

Die Arbeit ist in folgender Weise gegliedert: In Kapitel 2 werden die Grundlagen dieser Arbeit beschrieben. Schließlich fasst Kapitel 2 die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

1.1. Motivation

bla bla..

1.2. Zielsetzung

bla bla..

1.3. Aufbau der Arbeit

1. Vorerst werden in Kapitel 2..
2. In einer Anforderungsanalyse in Kapitel 3..
3. Kapitel 4 vermittelt..
4. Es folgt irgendwas in Kapitel 5. In diesem Teil..
5. Kapitel 6 dokumentiert
6. Die Gesamtarbeit..

2. Grundlagen

Im folgenden Kapitel werden auf die mehrere Grundlagen eingegangen. Zu Beginn wird ein Überblick über die Informationssicherheit geschaffen und Grundlegende Begriffe, Schutzziele, Schwachstellen, Bedrohungen und Angriffe aufgezeigt. Anschließend werden die Sicherheitsrichtlinie und Sicherheitsinfrastruktur vorgestellt. Zuletzt wird ein Überblick über die Technologien des Projektes gegeben.

2.1. Informationssicherheit

> *"Meine Nachricht für Unternehmen, die sie denken, nicht angegriffen worden ist: Sie suchen nicht hart genug."* James Snook

Informationssicherheit ist eine wichtige Herausforderung im Bereich der neuen Informationstechnologien. Sie ist heutzutage in nahezu allen Bereichen von zentraler Bedeutung. Die Sicherheit von Informationen, Daten, Geschäft und Investitionen gehören zu den Schlüsselprioritäten und die Verfügbarkeit von Informationen in der richtigen Zeit und Form ist heutzutage in jeder Organisation ein Muss. IT-Sicherheit hat das Zweck, Unternehmen und deren Werte zu schützen, indem sie böswillige Angriffe zu identifizieren, zu reduzieren und zu verhindern.

Das schnelle Wachstum von Webanwendungen hängt zweifellos von der Sicherheit, dem Datenschutz und der Zuverlässigkeit der Anwendungen, Systeme und unterstützenden Infrastrukturen ab. Obwohl IT-Sicherheit heutzutage sehr große Rolle in unserem Leben spielt, liegt das durchschnittliche Sicherheitswissen von IT-Fachkräften und Ingenieuren weit hinter. Das Internet ist jedoch für seine mangelnde Sicherheit bekannt, wenn sie nicht genau und streng spezifiziert, entworfen und getestet werden. In den letzten Jahren war es offensichtlich, dass der Bereich der IT-Sicherheit leistungsfähige Werkzeuge und Einrichtungen mit überprüfbaren Systemen und Anwendungen umfassen muss, die die Vertraulichkeit und Privatsphäre in Webanwendungen wahren, die die Probleme definieren[Fur08, S. 1].

Wegen der Realisierung eine lückenlose Abwehr von Angriffen in der Wirklichkeit nicht möglich ist, inkludieren das Gebiet der IT-Sicherheit hauptsächlich auch Maßnahmen und Vorgehensweise, um die potenziellen Schäden zu vermindern, um die Risiken beim Einsatz von Informations- und Kommunikationstechnologie (IKT)-Systemen zu verringern. Die Angriffsversuche müssen rechtzeitig und mit möglichst hoher Genauigkeit erkannt werden. Dazu muss auf eingetretene Schadensfälle mit geeigneten technischen Maßnahmen reagiert werden. Es sollte klargestellt werden, dass Techniken der Angriffserkennung und Reaktion ebenso zur IT-Sicherheit gehören, wie methodische Grundlagen, um IKT-Systeme so zu entwickeln, dass sie mittels Design ein hohes Maß an Sicherheit bieten. Durch IT-Sicherheit können eine Möglichkeit geschaffen werden, um die Entwicklung vertrauenswürdiger Anwendungen und Dienstleistungen mit innovativen

Geschäftsmodellen beispielsweise im Gesundheitsbereich, bei Automotive-Anwendungen oder in zukünftigen, intelligenten Umgebungen zu verbinden[Eck13, S. 20–21].

2.1.1. Grundlagen der IT-Sicherheit

2.1.1.1. Offene- und geschlossene Systeme

Ein IT-System besteht aus ein geschlossenes und ein offenes System mit der Begabung zur Speicherung und Verarbeitung von Informationen. Die offene Systemen (z. B. Windows) sind vernetzte, physisch verteilte Systeme mit der Möglichkeit zum Informationsaustausch mit anderen Systemen[Eck13, S. 22–23]. Tom Wheeler definiert offene Systeme als *"jene Hard- und Software-Implementierungen, die der Sammlung von Standards entsprechen, die den freien und leichten Zugang zu Lösungen verschiedener Hersteller erlauben. Die Sammlung von Standards kann formal definiert sein oder einfach aus De-facto-Definitionen bestehen, an die sich die großen Hersteller und Anwender in einem technologischen Bereich halten"*[Whe13, S. 4].

Ein geschlossenes System (z. B. Datev) baut auf der Technologie eines Herstellers auf und ist zu Konkurrenzprodukten nicht kompatibel. Es dehnt sich auf einen bestimmten Teilnehmerkreis aus und beschränkt ein bestimmtes räumliches Gebiet[Eck13, S. 22–23].

2.1.1.2. Soziotechnische Systeme

IT-Systeme werden in unterschiedliche Strukturen (gesellschaftliche, unternehmerische und politische Strukturen) mit verschiedenem technischen Know-how und für sehr verschiedene Ziele in Betrach gezogen[Eck13, S. 23]. IT-Sicherheit wird den Schutz eines soziotechnischen Systems gewährleistet. Darauffolgend ergibt sich dann auch das Ziel der IT Sicherheit. Die Unternehmen bzw. Institutionen und deren Daten sollen gegen Schaden und Bedrohungen geschützt werden. Insbesondere soll auf den Schutz von IT-Systemen angewiesen sein[STS18].

2.1.1.3. Information und Datenobjekte

IT-Systeme haben die Begabung Informationen zu speichern und zu verarbeiten. Die Information wird in Form von Daten bzw. Datenobjekten repräsentiert. **Passiven Objekten** (z.B. Datei, Datenbankeintrag) haben die Fähigkeit, Informationen zu speichern. **Aktive Objekten** (z.B. Prozesse) haben die Fähigkeit, sowohl Informationen zu speichern als auch zu verarbeiten[Eck13, S. 23]. **Subjekte** sind die Benutzer eines Systems und alle Objekte, die im Auftrag von Benutzern im System aktiv sein können[Eck13, S. 24].

Informatik wird als die Wissenschaft, Technik und Anwendung der maschinellen Verarbeitung, Speicherung, Übertragung und Darstellung von Information identifiziert. **Informationen** sind einen abstrakten Gehalt ("Bedeutungsinhalt", "Semantik") eines Dokuments, einer Aussage, Beschreibung, Anweisung oder Mitteilung[Bro13, S. 5] und sie werden durch die Nachrichten übermittelt[BKRS13, S. 18]. Information wird umgangssprachlich sehr oft für Daten verwendet aber es gibt Unterschied zwischen Daten und Information. Der Mensch bildet die Informationen in Daten ab, indem er die Nachrichten übertragen oder verarbeiten. Die Daten, die maschinell bearbeitbare Zeichen sind, werden durch in einer Nachricht enthaltene Information die Bedeutung

der Nachricht darstellen. Auf der Ebene der Daten geschiet die Übertragung oder Verarbeitung und das Resultat wird vom Mensch als Information interpretiert[BSI08].

2.1.1.4. Funktionssicher

In den Sicheren Systeme sollen alle korrekten Spezifikationen korrekt funktioniert werden und eine hohe Zuverlässigkeit und Fehlersicherheit gewährleistet werden. Die Isolierung von der Außenwelt weicht konstant durch die stetig zunehmende Vernetzung jeglicher Systeme mit Informationstechnik auf. Das Zweck von Funktionssicherheit (engl. safety) ist, dass die Umgebung vor dem Fehlverhalten des Systems schützen. Bei der Entwicklungsphase müssen systematische Fehler vermieden werden. Durch die Überwachung im laufenden Betrieb müssen die Störungen erkannt werden und solche Störungen müssen dominiert werden, um ein funktionssichere Zustand zu erreichen[MHTK15].

2.1.1.5. Informationssicher

Das Hauptziel von Informationssicherheit (engl. security) ist, dass die Informationen zu schützen, die sowohl auf Papier, in Rechnern oder auch in Köpfen gespeichert sein. IT-Sicherheit kümmert sich ersten um den Schutz von Werten u. Ressourcen und deren Verarbeitung[BSIDI11, S. 81], um unautorisierte Informationsveränderung oder -gewinnung zu verhindern[Eck13, S. 26].

2.1.1.6. Datensicherheit und Datenschutz

Datensicherheit bedeutet, dass der Zustand eines Systems der Informationstechnik, in dem die Risiken, die im laufenden Betrieb dieses Systems bezüglich von Gefährdungen anwesend sind, durch Maßnahmen auf ein bestimmtes Menge eingeschränkt sind. Datenschutz (engl. privacy) hat die Aufgabe, durch Schutz der Daten vor Missbrauch in ihren Verarbeitungsphasen der Beeinträchtigung fremder und eigener schutzwürdiger Belange zu begegnen.[Ebe94, S. 14–15].

2.1.1.7. Verlässligkeit

Verlässlichkeit (engl. dependability) eines Systems bedeutet, dass es keine betrügerische Zustände akzeptieren und es soll gewährleistet werden, indem spezifizierte Funktion verlässlich funktioniert[Eck13, S. 27].

2.1.2. Schutzziele

2.1.2.1. Authentizität

Bei dem Begriff ”Authentizität” handelt es sich um die Authentizität eines Objekts bzw. Subjekts (engl. authenticity), die die Echtheit und Glaubwürdigkeit des Objekts oder Subjekts, die anhand einer eindeutigen Identität und charakteristischen Eigenschaften umfasst[Eck13, S. 28].

Erkennung von Angriffen können gewährleistet werden, indem innere Maßnahmen zu vollständigt wird, die der Authentizität von Subjekten und Objekten überprüft[Spi85, S. 13], diesbezüglich muss Beweis erbracht werden, dass eine behauptete Identität eines Objekts oder Subjekts mit dessen charakterisierenden Eigenschaften übereinstimmt[Eck13, S. 28].

2.1.2.2. Informationsvertraulichkeit

Unter Informationsvertraulichkeit versteht man, dass zu bearbeitende Daten nur den Personen zugänglich sind, die Berechtigung haben. Wenn die Geheimhaltung vernünftig ist, können Schaden entstehen. In jedem einzelnen Unternehmensbereich muss durch die vollständige Maßnahmen den unautorisierte Zugriff in interne Datenbestände verhindert werden[Gor03, S. 205].

2.1.2.3. Datenintegrität

Durch die Integrität wird die Korrektheit von Daten und der korrekten Funktionsweise von Systemen sichergestellt. Wenn der Begriff Integrität auf Daten benutzt wird, bedeutet er, dass die Daten vollständig und unverändert sind. Er wird in der Informationstechnik weiter gefasst und auf ”Informationen” angewendet. Der Begriff ”Information” wird für Daten angewendet, die nach bestimmten Attribute, wie z. B. Autor oder Zeitpunkt der Erstellung zugeordnet können. Wenn die Daten ohne Erlaubnis verändert werden, bedeuten dass die Angaben zum Autor verfälscht oder Zeitangaben zur Erstellung manipuliert wurden[BFSI07].

2.1.2.4. Verfügbarkeit

Ein System versichert die Verfügbarkeit (eng. availability), indem authentifizierte und autorisierte Subjekte in der Wahrnehmung ihrer Berechtigungen nicht unautorisiert beeinträchtigt werden können. Wenn in einem System unterschiedliche Prozesse eines Benutzers oder verschiedene Benutzern um gemeinsame Ressourcen zugreifen, kann Ausführungsverzögerungen vorkommen. Durch normalen Verwaltungsmaßnahmen resultierende Verzögerungen werden als keine Verletzung der Verfügbarkeit dargestellt, aber wenn CPU mit einem hoch prioren Prozess monopolisiert, diesbezüglich kann absichtlich einen Angriff auf die Verfügbarkeit hervorrufen. Somit kann ein hohes Maß von Daten plötzlich entstehen, das zu Stausituationen in einem Netz führt[Eck13, S. 33].

2.1.2.5. Verbindlichkeit

Verbindlichkeit ist eine Möglichkeit, die eine IT-Transaktion während und nach der Durchführung unzweifelhaft gewährleisten. Durch die Nutzung von qualifizierten digitalen Signaturen kann die Verbindlichkeit erreicht werden. Die Dauer der Zuordenbarkeit ist abhängig von der Aufbewahrung der Logdateien und wird durch den Datenschutz angeordnet[SePe11].

2.1.2.6. Anonymisierung

Nach § 3 Abs. 6 Bundesdatenschutzgesetz bedeutet Anonymisierung, dass *"das Verändern personenbezogener Daten derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können"*[DsBa18].

2.1.3. Schwachstellen, Bedrohungen, Angriffe

2.1.3.1. Schwachstellen und Verwundbarkeit

Hauptgrund für die Gefährdung der Erreichung der Schutzziele sind **Schwachstellen**. Wenn die Schwachstellen ausgenutzt werden, werden die Interaktionen mit einem IT-System autorisiert, die nicht dem definierten Soll-Verhalten entsprechen. Mittels der Ausnutzung von Schwachstellen können das IT-System angegriffen werden. Somit können die unberechtigt auf eine Ressource zugegriffen werden[Now11, S. 19–20]. Unter der Begriff "Verwundbarkeit" (engl. vulnerability) versteht man, dass sie eine Schwachstelle ist und über solche Verwundbarkeiten können die Sicherheitsdienste des Systems unautorisiert modifiziert werden[Eck13, S. 38].

2.1.3.2. Bedrohungen und Risiko

Eine Bedrohung (engl. threat) bedeutet Ausnutzung einer oder mehreren Schwachstellen oder Verwundbarkeiten, die einen Verlust der Datenintegrität, der Informationsvertraulichkeit oder der Verfügbarkeit zu führen, oder die Authentizität von Subjekten zu gefährden[Eck13, S. 39]. Im Kontext der Informationssicherheit versteht man unter einem Risiko die Wahrscheinlichkeit des Eintritts eines Schadenereignisses und die Höhe des potentiellen Schadens ist[Now11, S. 15].

2.1.3.3. Angriff und Typen von (externe) Angriffen

Personen oder Systeme, die versuchen eine Schwachstelle auszunutzen, wird als **Angreifer** genannt. Ein **Angriff** ist dabei der Versuch, ein IT-System unautoriziert zu verändern oder zu nutzen und unterscheidet als aktive- und passive Angriffe. Wenn die Vertraulichkeit durch unberechtigte Informationsgewinnung verletzt wird, wird dies als **passive Angriffe** genannt und **aktive Angriffe** manipulieren die Daten oder schleusen sie in das System ein, um die Verfügbarkeit und Integrität zu gefährden[Now11, S. 20].

2.1.3.3.1. Hacker und Cracker

Hacker sind technisch erfahrene Personen im Hard- und Softwareumfeld. Sie können Schwachstellen finden, um unbefugtes einzudringen oder Funktionen zu verändern[Sch17]. Dieser Begriff wird für kriminelle Personen verwendet, die die Lücken im IT-System finden und dies unerlaubt für kriminelle Zwecke wie z. B. Diebstahl von Informationen nutzen[Sill11]. Ebenfalls ist ein **Cracker** ein technisch sehr erfahrener Angreifer und er hat das Zweck entweder nur an seinen eigenen Vorteil oder will den Schaden einer dritten Person, diesbezüglich geht ein größere Schadenrisiko als Hackern, deswegen strengen Unternehmen noch mehr an[Eck13, S. 45].

2.1.3.3.2. Skript Kiddie

Unter dem Begriff "Script-Kiddie" versteht man ein nicht-ernsthafte Hacker, der die ethischen Prinzipien professioneller Hacker ablehnen, die das Streben nach Wissen, Respekt vor Fähigkeiten und ein Motiv der Selbstbildung beinhalten. Script Kiddies verkürzen die meisten Hacking-Methoden, um schnell ihre Hacking-Fähigkeiten zu erlangen. Sie machen sich nicht viel Gedanken oder Zeit, um Computerkenntnisse zu erwerben, sondern bilden sich schnell aus, um nur das Nötigste zu lernen. Skript-Kiddies können Hacking-Programme verwenden, die von anderen Hackern geschrieben wurden, weil ihnen oft die Fähigkeiten fehlen, eigene zu schreiben. Script Kiddies versuchen, Computersysteme und Netzwerke anzugreifen und Websites zu zerstören. Obwohl sie als unerfahren und unreif angesehen werden, können Skript-Kiddies so viel Computerschaden verursachen wie professionelle Hacker[TSC11].

2.1.3.3.3. Geheimdienste

Die National Security Agency (NSA) ist ein US-Geheimdienst, der für die Erstellung und Verwaltung von Informationssicherung und Signalintelligenz (SIGINT) für die US-Regierung verantwortlich ist. Die Aufgabe der NSA besteht in der globalen Überwachung, Sammlung, Entschlüsselung und anschließenden Analyse und Übersetzung von Informationen und Daten für ausländische nachrichtendienstliche und nachrichtendienstliche Zwecke[TNSA14].

2.1.3.3.4. Allgemeine Krimanilität

Spyware: Spyware ist eine Art von Malware (oder "bösartige Software"), die Informationen über einen Computer oder ein Netzwerk ohne die Zustimmung des Benutzers sammelt und weitergibt. Es kann als versteckte Komponente echter Softwarepakete oder über herkömmliche Malware-Vektoren wie betrügerische Werbung, Websites, E-Mail, Instant-Messages sowie direkte File-Sharing-Verbindungen installiert werden. Im Gegensatz zu anderen Arten von Malware wird Spyware nicht nur von kriminellen Organisationen, sondern auch von skrupellosen Werbern und Unternehmen, die Spyware verwenden, genutzt, um Marktdaten von Nutzern ohne deren Zustimmung zu sammeln. Unabhängig von der Quelle wird Spyware vor dem Benutzer verborgen und ist oft schwer zu erkennen, kann jedoch zu Symptomen wie einer verschlechterten Systemleistung und einer hohen Häufigkeit unerwünschter Verhaltensweisen führen[CPSW12].

Phishing: Phishing ist eine Cyberkriminalität, bei der ein Ziel oder Ziele per E-Mail, Telefon oder SMS von jemandem kontaktiert werden, der sich als legitime Institution ausgibt, um Personen dazu zu bringen, sensible Daten wie persönlich identifizierbare Informationen, Bank- und Kreditkartendetails und Passwörter bereitzustellen. Die Informationen werden dann für den Zugriff auf wichtige Konten verwendet und können zu Identitätsdiebstahl und finanziellen Verlusten führen[PHS17].

Erpressung: Bei der Erpressung geht es sich um Schadsoftware, die auf fremden Rechnern eindringt, somit wird die Daten auf der Festplatte des fremden Rechners so verschlüsselt, dass diese Daten für den Benutzer nicht mehr verfügbar sind Danach fordert der Angreifer für die Entschlüsselung der Daten fordert einen Geldbetrag, der über ein Online-Zahlungssystem entrichten ist[Eck13, S. 48].

Bot-Netze: Unter der Begriff "Bot-Netz" versteht man, dass es eine Reihe verbundener Computer, die gemeinsam auf einer Aufgabe (wie z. B. Massenhafte Versand von E-Mails) abgezielt wurden[BN17].

2.2. Technologien des Projektes

Für die Entwicklung des OWASP Zap RAML Plug-In sind verschiedene Technologien erforderlich. Die benötigten Technologien werden in diesem Abschnitt näher erläutert.

2.2.1. Objektorientierte Programmierung

Die objektorientierte Programmierung (OOP) bezieht sich auf eine Art von Computerprogrammierung (Softwaredesign), bei der Programmierer nicht nur den Datentyp einer Datenstruktur definieren, sondern auch die Arten von Operationen (Funktionen), die auf die Datenstruktur angewendet werden können. Auf diese Weise wird die Datenstruktur zu einem Objekt, das sowohl Daten als auch Funktionen enthält. Darüber hinaus können Programmierer Beziehungen zwischen einem Objekt und einem anderen erstellen. Zum Beispiel können Objekte Eigenschaften von anderen Objekten erben[OOP15].

2.2.1.1. Java

Java ist eine universelle Programmiersprache, die von Sun Microsystems entwickelt wurde. Java definiert als objektorientierte Sprache ähnlich wie C ++, wird jedoch vereinfacht, um Sprachfunktionen zu eliminieren, die häufige Programmierfehler verursachen. Die Quellcodedateien (Dateien mit der Erweiterung `.java`) werden in ein Format namens Bytecode (Dateien mit der Erweiterung `.class`) kompiliert, das dann von einem Java-Interpreter ausgeführt werden kann. Kompilierter Java-Code kann auf den meisten Computern ausgeführt werden, da Java-Interpreter und Laufzeitumgebungen, die als Java Virtual Machines (VMs) bezeichnet werden, für die meisten Betriebssysteme vorhanden sind, einschließlich UNIX, Macintosh OS und Windows[J18].

2.2.2. RESTful Web Services

Representational State Transfer (REST) ist ein Architekturstil, der Einschränkungen wie die einheitliche Schnittstelle angibt, die bei Anwendung auf einen Webdienst wünschenswerte Eigenschaften wie Leistung, Skalierbarkeit und Änderbarkeit hervorbringen, mit denen Services im Web am besten funktionieren. Im REST-Architekturstil werden Daten und Funktionen als

Ressourcen betrachtet und der Zugriff über URIs (Uniform Resource Identifiers) erfolgt. Der REST-Architekturstil beschränkt eine Architektur auf eine Client/Server-Architektur und ist so ausgelegt, dass ein zustandsloses Kommunikationsprotokoll (wie z. B. HTTP) verwendet wird. Im REST-Architektur-Stil tauschen Clients und Server Repräsentationen von Ressourcen unter Verwendung einer standardisierten Schnittstelle und eines standardisierten Protokolls aus[RWS13].

Die folgenden Prinzipien ermutigen RESTful-Anwendungen, einfach, leicht und schnell zu sein:

Ressourcenidentifikation durch URI: Ein RESTful Webservice macht eine Reihe von Ressourcen verfügbar, die die Ziele der Interaktion mit ihren Clients identifizieren. Ressourcen werden durch URIs identifiziert, die einen globalen Adressierungsraum für die Ressourcen- und Serviceerkennung bereitstellen.

Einheitliche Schnittstelle: Ressourcen werden mit einem festen Satz von vier Operationen zum Erstellen, Lesen, Aktualisieren und Löschen bearbeitet: PUT, GET, POST und DELETE. **PUT** erstellt eine neue Ressource, die mit **DELETE** gelöscht werden kann. **GET** ruft den aktuellen Status einer Ressource in einer Darstellung ab. **POST** überträgt einen neuen Status auf eine Ressource.

Selbstbeschreibende Nachrichten: Ressourcen sind von ihrer Darstellung entkoppelt, sodass auf ihren Inhalt in verschiedenen Formaten wie HTML, XML, Nur-Text, PDF, JPEG, JSON und anderen zugegriffen werden kann. Metadaten über die Ressource sind verfügbar und werden beispielsweise verwendet, um das Zwischenspeichern zu steuern, Übertragungsfehler zu erkennen, das geeignete Repräsentationsformat auszuhandeln und eine Authentifizierung oder Zugriffssteuerung durchzuführen.

Stateful Interaktionen durch Hyperlinks: Jede Interaktion mit einer Ressource ist staatenlos; Das heißt, Anforderungsnachrichten sind in sich geschlossen. Stateful Interaktionen basieren auf dem Konzept der expliziten Zustandsübertragung. Es gibt verschiedene Techniken, um den Status auszutauschen, z. B. URI-Umschreiben, Cookies und versteckte Formularfelder. Der Zustand kann in Antwortnachrichten eingebettet sein, um auf gültige zukünftige Zustände der Interaktion zu zeigen.

2.2.3. Microservice Architekturen

Microservices sind ein Modularisierungskonzept und dienen dazu, große Softwaresysteme in kleinere Teile zu unterteilen. Sie beeinflussen somit die Organisation und Entwicklung von Softwaresystemen. Microservices können unabhängig voneinander eingesetzt werden. Das heißt, dass Änderungen an einem Microservice unabhängig von Änderungen anderer Microservices in Produktion genommen werden können. Sie können in verschiedenen Technologien implementiert werden und es gibt keine Einschränkung für die Programmiersprache oder die Plattform[Wol16, S. 45–46].

2.2.3.1. Spring Boot

Das Spring Framework, das bereits seit über einem Jahrzehnt besteht, hat sich als Standardframework für die Entwicklung von Java-Anwendungen etabliert.

2.2.3.1.1. Spring MVC Komponente

Das Spring Web MVC-Framework stellt eine Model-View-Controller (MVC) -Architektur und fertige Komponenten bereit, mit denen flexible und lose gekoppelte Webanwendungen entwickelt werden können. Das MVC-Muster führt zu einer Trennung der verschiedenen Aspekte der Anwendung (Eingangslogik, Geschäftslogik und UI-Logik), während eine lose Kopplung zwischen diesen Elementen bereitgestellt wird[MVC12].

- Das Modell kapselt die Anwendungsdaten und besteht im Allgemeinen aus POJO (Plain Old Java Object).
- Die Ansicht ist verantwortlich für das Rendern der Modelldaten und generiert im Allgemeinen HTML-Ausgabe, die der Browser des Clients interpretieren kann.
- Der Controller ist verantwortlich für die Verarbeitung von Benutzeranforderungen und das Erstellen eines geeigneten Modells und übergibt es an die Ansicht zum Rendern.

2.2.3.1.2. Spring Rest Docs

Spring REST Docs verwendet Snippets, die von Tests erstellt wurden, die mit Spring MVCs Testframework, Spring WebTestClient von WebFlux oder REST Assured 3 geschrieben wurden. Dieser Test-Driven-Ansatz hilft dabei, um die Genauigkeit der Service-Dokumentation zu gewährleisten. Das Ziel von Spring REST Docs ist es, Dokumentationen für RESTful Services zu erstellen, die genau und lesbar sind. Bei der Dokumentation eines RESTful Services geht es hauptsächlich um die Beschreibung seiner Ressourcen. Zwei wichtige Teile der Beschreibung jeder Ressource sind die Details der HTTP-Anforderungen, die sie verbraucht, und die HTTP-Antworten, die sie erzeugt[Wil18].

2.2.4. Modellbasierte Ansätze für REST-Schnittstellen

Für viele verschiedene Einsatzgebiete gibt es verschiedenen Modellierungssprachen für die Erstellung und Beschreibung von REST-Schnittstellen. In diesem Abschnitt werden auf die Sprachen OPenAPI und RAML eingehen.

2.2.4.1. Open API

OpenAPI (früher Swagger) ist ein API-Beschreibungsformat für REST-APIs. Mit einer OpenAPI-Datei kann das gesamte API beschrieben werden[Swa17];

- Verfügbare Endpunkte (`/user`) und Vorgänge für jeden Endpunkt (`GET /user, POST /user`),
- Ein- und Ausgabe für jede Operation,
- Authentifizierungsmethoden,
- Kontaktinformationen, Lizenzen, Nutzungsbedingungen und andere Informationen.

2.2.4.2. RAML

RAML steht für RESTful API Modeling Language und ist eine Möglichkeit, praktisch RESTful APIs so zu beschreiben, dass sie sowohl für Menschen als auch für Computer gut lesbar sind. RAML konzentriert sich auf sauber zu beschreibende Ressourcen, Methoden, Parameter, Antworten, Medientypen und andere HTTP-Konstrukte, die die Basis für viele moderne APIs bilden. Das Ziel ist es, dass dem aktuellen API-Ökosystem zu helfen und unmittelbare Probleme zu lösen und dann immer bessere API-Muster zu fördern. Mittels RAML wird einen klar definierten Format in einem für Menschen lesbaren Format erhalten, um tatsächlich als Quellcode zu existieren[RAML13].

RAML beschreibt eine API in einer Weise:

- **Richtig:** Eine API-Spezifikation ist ein Vertrag: Was erfasst wird, muss das Verhalten der API korrekt beschreiben.
- **Konsistent:** RAML unterstützt stark die Erfassung von Mustern und ermöglicht die Entdeckung und den Austausch von Mustern.
- **Lesbar und beschreibbar:** Optimiert das Erstellen und Lesen von Spezifikationen.

2.2.5. Open Source Werkzeug OWASP Zap

Der OWASP Zed Attack Proxy (ZAP) ist eines der beliebtesten kostenlosen Sicherheitstools der Welt und wird von Hunderten von internationalen Freiwilligen aktiv gepflegt. Es hilft automatisch bei der Entwicklung und beim Testen von Anwendungen Sicherheitslücken in Webanwendungen zu finden. Es ist auch ein großartiges Werkzeug für erfahrene PenTester für manuelle Sicherheitstests[OZ18].

Funktion	Beschreibung
Intercepting Proxy	ZAP ermöglicht alle Anforderungen, die an eine Web-App gestellt werden und alle Antworten abzufangen und zu prüfen. Unter anderem können dadurch auch AJAX calls abgefangen werden.
Spider	Spider können neue URLs auf Webseiten entdecken und aufrufen. Der Spider in ZAP überprüft alle gefundenen Links auf Sicherheitsprobleme.
Automatischer, aktiver Scan	ZAP kann automatisiert Web-Apps auf Sicherheitslücken überprüfen und Angriffe durchführen. Diese Funktion dürfen nur für eigene Webanwendungen genutzt werden.
Passiver Scan	Mit passiven Scans werden Webanwendungen überprüft, aber nicht angegriffen.
Forced Browse	ZAP kann testen, ob bestimmte Verzeichnisse oder Dateien auf dem Webserver geöffnet werden können.
Fuzzing	Mit dieser Technik werden ungültige und unerwartete Anfragen an den Webserver gesendet.
Dynamic SSL Certificates	ZAP kann SSL-Anfragen entschlüsseln. Dazu verwendet das Tool den Man-in-the-Middle-Ansatz.

Smartcard und Client Digital Certificates Support	ZAP kann Smartcard-gestützte Webanwendungen testen, sowie TLS-Handshakes prüfen, zum Beispiel zwischen Mail-Servern.
WebSockets	Mit WebSockets lassen sich auch Anwendungen testen, die eine einzige TCP-Verbindung für die bidirektionale Kommunikation nutzen.
Plug-n-Hack	Diese Technologie wurde von Mozilla entwickelt, um festzulegen, wie Sicherheitstools wie ZAP mit Browsern zusammenarbeiten können, um optimale Sicherheitstests durchzuführen.
Powerful REST based API	Webentwickler können eine eigene grafische Oberfläche für ZAP entwickeln, um das Tool dem eigenen Unternehmen anzupassen.
Add-Ons und Erweiterungen	In ZAP lassen sich Erweiterungen integrieren, sowie Vorlagen für bestimmte Tests. Dazu steht ein eigener Shop zur Verfügung

Tabelle 2.1.: Die wichtigsten Funktionen von ZAP

Quelle: Entnommen aus Security Insider:
https://www.security-insider.de/sicherheit-fuer-webanwendungen-mit-zed-attack-proxy-a-728630/

3. Sicherheitsrisiken von Webanwendungen

In diesem Kapitel wird die Sicherheit von Webanwendungen anhand von Bedrohungen, Gegenmaßnahmen, Schwachstellen und Angriffen analysiert.

3.1. Schwachstellen

3.1.1. OWASP Top 10 Risiken

3.1.1.1. Injection

Injektion-Schwachstellen wie SQL-, NoSQL-, OS- und LDAP-Injection treten auf, wenn nicht vertrauenswürdige Daten als Teil eines Befehls oder einer Datenabfrage von einem Interpreter verarbeitet werden[OWA17, S. 6].

Angriffsvektoren: Der Angreifer sendet einfache textbasierte Angriffe, die die Syntax des Zielinterpreters missbrauchen. Fast jede Datenquelle kann einen Injection-Vektor darstellen, einschließlich interner Quellen[OWA17, S. 7].

Schwachstellen: Injection-Schwachstellen tauchen auf, wenn eine Anwendung nicht vertrauenswürdige Daten an einen Interpreter weiterleitet. Sie sind weit verbreitet, besonders in veraltetem Code. Sie finden sich in SQL-, LDAP-, XPath und NoSQL-Anfragen, in Betriebssystembefehlen sowie in XML, SMTP-Headern, Parametern, etc. Injection-Schwachstellen lassen sich durch Code-Prüfungen einfach, durch externe Tests aber in der Regel nur schwer entdecken. Angreifer setzen dazu Scanner und Fuzzer ein[OWA17, S. 7].

Technische Auswirkungen: Injection kann zu Datenverlust oder -verfälschung, Fehlen von Zurechenbarkeit oder Zugangssperre führen. Unter Umständen kann es zu einer vollständigen Systemübernahme kommen[OWA17, S. 7].

Mögliche Angriffsszenarien:

Szenario 1:
Die Anwendung nutzt ungeprüfte Eingabedaten bei der Konstruktion der verwundbaren SQL-Abfrage:

```
String query = SSELECT * FROM accounts WHERE
custID='"+ request.getParameter(id") + "";
```

Szenario 2:
Blindes Vertrauen in den Einsatz eines Frameworks:

```
Query HQLQuery = session.createQuery("FROM accounts
WHERE custID='" + request.getParameter(id") + "");
```
Der Angreifer verändert in beiden Fällen den 'id'-Parameter im Browser und sendet: ' or '1'='1.

Zum Beispiel: http://example.com/app/accountView?id=' or '1'='1

Das ändert die Logik der Anfrage so, dass in diesem Fall alle Datensätze der Tabelle ‚accounts' zurückgegeben werden. Schlimmstenfalls werden durch Injections Daten verändert oder sogar Stored Procedures gestartet[OWA17, S. 7].

3.1.1.2. Fehler in Authentifizierung und Session-Management

Anwendungsfunktionen, die die Authentifizierung und das Session-Management umsetzen, werden oft nicht korrekt implementiert. Dies erlaubt es Angreifern Passwörter oder Session-Token zu kompromittieren oder die Schwachstellen so auszunutzen, dass sie die Identität anderer Benutzer annehmen können[OWA17, S. 6].

Angriffsvektoren: Angreifer nutzen Lücken bei der Authentifizierung oder im Sessionmanagement (z.B. ungeschützte Nutzerkonten, Passwörter, Session-IDs), um sich eine fremde Identität zu verschaffen[OWA17, S. 8].

Schwachstellen: Obwohl es sehr schwierig ist, ein sicheres Authentifizierungs- und Session-Management zu implementieren, setzen Entwickler häufig auf eigene Lösungen. Diese haben dann oft Fehler bei Abmeldung und Passwortmanagement, bei der Wiedererkennung des Benutzers, bei Timeouts, Sicherheitsabfragen usw. Das Auffinden dieser Fehler kann sehr schwierig sein, besonders wenn es sich um individuelle Implementierungen handelt[OWA17, S. 8].

Technische Auswirkungen: Diese Fehler führen zur Kompromittierung von Benutzerkonten. Ein erfolgreicher Angreifer hat alle Rechte des Opfers. Privilegierte Zugänge sind oft Ziel solcher Angriffe[OWA17, S. 8].

Mögliche Angriffsszenarien:

Szenario 1:
Eine Flugbuchungsanwendung fügt die Session-ID in die URL ein[OWA17, S. 8]:

http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

Ein authentifizierter Anwender möchte dieses Angebot seinen Freunden mitteilen. Er versendet obigen Link per E-Mail, ohne zu wissen, dass er seine Session-ID preisgibt. Nutzen seine Freunde den Link, können sie seine Session sowie seine Kreditkartendaten benutzen.

Szenario 2:
Anwendungs-Timeouts sind falsch konfiguriert. Ein Anwender benutzt einen öffentlichen PC, um die Anwendung aufzurufen. Anstatt die „Abmelden"-Funktion zu benutzen, schließt der Anwender nur den Browser. Der Browser ist auch eine Stunde später noch authentifiziert, wenn ein potentieller Angreifer ihn öffnet[OWA17, S. 8].

3.1.1.3. Verlust der Vertraulichkeit sensibler Daten

Viele Anwendungen schützen sensible Daten, wie Kreditkartendaten oder Zugangsinformationen nicht ausreichend. Angreifer können solche nicht angemessen geschützten Daten auslesen oder modifizieren und mit ihnen weitere Straftaten, wie beispielsweise Kreditkartenbetrug, oder Identitätsdiebstahl begehen. Vertrauliche Daten benötigen zusätzlichen Schutz, wie z.B. Verschlüsselung während der Speicherung oder Übertragung sowie besondere Vorkehrungen beim Datenaustausch mit dem Browser[OWA17, S. 6].

Angriffsvektoren: Angreifer brechen i.d.R. nicht die Verschlüsselung selbst. Stattdessen stehlen sie Schlüssel, führen Seiten-Angriffe aus oder stehlen Klartext vom Server, während der Übertragung oder aus dem Browser des Kunden heraus[OWA17, S. 9].

Schwachstellen: Fehlende Verschlüsselung vertraulicher Daten ist die häufigste Schwachstelle. Die Nutzung von Kryptographie erfolgt oft mit schwacher Schlüsselerzeugung und -verwaltung und der Nutzung schwacher Algorithmen, insbesondere für das Password Hashing. Browser Schwachstellen sind verbreitet und leicht zu finden, aber nur schwer auszunutzen. Ein eingeschränkter Zugriff lässt externer Angreifer Probleme auf dem Server i.d.R. nur schwer finden und ausnutzen[OWA17, S. 9].

Technische Auswirkungen: Fehler kompromittieren regelmäßig vertrauliche Daten. Es handelt sich hierbei oft um sensitive Daten wie personenbezogene Daten, Benutzernamen und Passwörter oder Kreditkarteninformationen[OWA17, S. 9].

Mögliche Angriffsszenarien:

Szenario 1:
Eine Anwendung verschlüsselt Kreditkartendaten automatisch bei der Speicherung in einer Datenbank. Das bedeutet

aber auch durch SQL-Injection erlangte Kreditkartendaten in diesem Fall automatisch entschlüsselt werden. Die Anwendung hätte die Daten mit eine Public Key verschlüsseln sollen und nur nachgelagerte Anwendungen und nicht die Webanwendung selbst hätten die Daten mit dem Private Key entschlüsseln dürfen[OWA17, S. 9].

Szenario 2:
Eine Webseite schützt die authentisierten Seiten nicht mit SSL. Der Angreifer stiehlt das Sitzungscookie des Nutzers durch einfaches Mitlesen der Kommunikation (z.B. in einem offenen WLAN). Durch Wiedereinspielen dieses Cookies übernimmt der Angreifer die Sitzung des Nutzers und erlangt Zugriff auf die privaten Daten des Nutzers[OWA17, S. 9].

3.1.1.4. XML External Entities (XXE)

Viele ältere oder schlecht konfigurierte XML-Prozessoren werten externe Entitätsverweise in XML-Dokumenten aus. Wenn bei der Verarbeitung solcher Dokumente nicht berücksichtigt werden, wird eine unberechtigte Befehlsausführung den Abfluss interner Informationen riskiert[OWA17, S. 6].

Angriffsvektoren: Der Angreifer können anfällige XML-Prozessoren ausnutzen, wenn sie XML hochladen oder feindliche Inhalte in ein XML-Dokument aufnehmen können, um anfälligen Code, Abhängigkeiten oder Integrationen auszunutzen[OWA17, S. 10].

Schwachstellen: Viele ältere XML-Prozessoren erlauben standardmäßig die Angabe einer externen Entität, eines URI, der dereferenziert und während der XML-Verarbeitung ausgewertet wird. Static Application Security Testing kann dieses Problem durch Untersuchen der Abhängigkeiten und der Konfiguration erkennen[OWA17, S. 10].

Technische Auswirkungen: Diese Fehler können verwendet werden, um Daten zu extrahieren, eine Remote-Anforderung vom Server auszuführen, interne Systeme zu scannen, einen Denial-of-Service-Angriff durchzuführen sowie andere Angriffe auszuführen[OWA17, S. 10].

Mögliche Angriffsszenarien:

Szenario 1:
Der Angreifer versucht, Daten vom Server zu extrahieren[OWA17, S. 10]:

```
<?xml version="1.0ëncoding=ÏSO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd»]>
<foo>&xxe;</foo>
```

Szenario 2:
Ein Angreifer testet das private Netzwerk des Servers, indem er die obige Entity-Zeile wie folgt ändert[OWA17, S. 10]:

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private»]>
```

Szenario 3:
Ein Angreifer versucht einen Denial-of-Service-Angriff, indem er eine möglicherweise endlose Datei einfügt[OWA17, S. 10]:

```
<!ENTITY xxe SYSTEM "file:///dev/random»]>
```

3.1.1.5. Broken Access Control

Einschränkungen, was authentifizierte Benutzer tun dürfen, werden häufig nicht ordnungsgemäß durchgesetzt. Angreifer können diese Mängel ausnutzen, um auf nicht autorisierte Funktionen und Daten zuzugreifen[OWA17, S. 6].

Angriffsvektoren: Die Nutzung der Zugriffskontrolle ist eine Kernkompetenz von Angreifern. Static Application Security Testing (SAST)-Tools können das Fehlen einer Zugriffskontrolle erkennen, können jedoch nicht überprüfen, ob sie funktionsfähig ist, wenn sie vorhanden ist[OWA17, S. 11].

Schwachstellen: Schwachstellen der Zugriffskontrolle treten häufig auf, da keine automatisierte Erkennung und keine effektiven Funktionstests durch Anwendungsentwickler vorhanden sind. Die Erkennung der Zugriffskontrolle ist normalerweise nicht für automatisierte statische oder dynamische Tests geeignet[OWA17, S. 11].

Technische Auswirkungen: Die technische Auswirkung besteht darin, dass Angreifer als Benutzer oder Administrator fungieren, indem sie jeden Datensatz erstellen, darauf zugreifen, aktualisieren oder löschen[OWA17, S. 11].

Mögliche Angriffsszenarien:

Szenario 1:
Die Anwendung verwendet unverifizierte Daten in einem SQL-Aufruf, der auf Kontoinformationen zugreift[OWA17, S. 11]:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Ein Angreifer modifiziert einfach den "acct" Parameter im Browser, um die gewünschte Kontonummer zu senden. Wenn dies nicht ordnungsgemäß überprüft wurde, kann der Angreifer auf das Konto eines Benutzers zugreifen.

http://example.com/app/accountInfo?acct=notmyacct

Szenario 2:
Ein Angreifer zwingt einfach zu Ziel-URLs. Für den Zugriff auf die Administrationsseite sind Administratorrechte erforderlich. Wenn ein nicht authentifizierter Benutzer auf eine Seite zugreifen kann, ist dies ein Fehler. Wenn ein nicht-Administrator auf die Verwaltungsseite zugreifen kann, ist dies auch ein Fehler[OWA17, S. 11].

3.1.1.6. Sicherheitsrelevante Fehlkonfiguration

Sicherheit erfordert die Festlegung und Umsetzung einer sicheren Konfiguration für Anwendungen, Frameworks, Applikations-, Web- und Datenbankserver sowie deren Plattformen. Sicherheitseinstellungen müssen definiert, umgesetzt und gewartet werden, die Voreinstellungen sind oft unsicher. Des Weiteren umfasst dies auch die regelmäßige Aktualisierung aller Software[OWA17, S. 6].

Angriffsvektoren: Angreifer benutzen Standardkonten, inaktive Seiten, ungepatchte Fehler, ungeschützte Dateien und Verzeichnisse etc., um unautorisierten Zugang zum oder Kenntnis über das Zielsystem zu erlangen[OWA17, S. 12].

Schwachstellen: Sicherheitsrelevante Fehlkonfiguration kann auf jeder Ebene der Anwendung, inkl. Plattform, Web- und Anwendungsserver, oder Datenbank vorkommen. Die Zusammenarbeit zwischen Entwicklern und Administratoren ist wichtig, um eine sichere Konfiguration aller Ebenen zu gewährleisten. Automatisierte Scanner können oft fehlende Sicherheitspatches, Fehlkonfigurationen, Standardkonten, nicht benötigte Dienste, usw. erkennen[OWA17, S. 12].

Technische Auswirkungen: Diese Fehler geben Angreifern häufig unautorisierten Zugriff auf Systemdaten oder -funktionalitäten. Manchmal führen sie zur kompletten Kompromittierung des Zielsystems[OWA17, S. 12].

Mögliche Angriffsszenarien:

Szenario 1:
Die Administratorkonsole mit Standardkonto wurde automatisch installiert und nicht entfernt. Angreifer entdecken dies, melden sich über das Standardkonto an und kapern das System[OWA17, S. 12].

Szenario 2:
Directory Listings wurden nicht deaktiviert. Angreifer nutzen dies, um in den Besitz aller Dateien zu kommen. Sie laden alle existierenden Java-Klassen herunter, dekompilieren diese und entdecken einen schwerwiegenden Fehler in der Zugriffskontrolle[OWA17, S. 12].

Szenario 3:
Die Konfiguration des Anwendungsserver erlaubt es, Stack Traces an Benutzer zurückzugeben. Dadurch können potentielle Fehler im Backend offengelegt werden. Angreifer nutzen zusätzliche Informationen in Fehlermeldungen aus[OWA17, S. 12].

Szenario 4:
Der Applikationsserver wird mit Beispielapplikationen ausgeliefert, die auf dem Produktivsystem nicht entfernt wurden. Diese Beispielapplikationen besitzen bekannte Sicherheitsschwachstellen, die Angreifer ausnutzen können um den Server zu kompromittieren[OWA17, S. 12].

3.1.1.7. Cross-Site Scripting (XSS)

XSS-Schwachstellen treten auf, wenn eine Anwendung nicht vertrauenswürdige Daten entgegennimmt und ohne entsprechende Validierung oder Umkodierung an einen Webbrowser sendet. XSS erlaubt es einem Angreifer Scriptcode im Browser eines Opfers auszuführen und somit Benutzersitzungen zu übernehmen, Seiteninhalte zu verändern oder den Benutzer auf bösartige Seiten umzuleiten[OWA17, S. 6].

Angriffsvektoren: Der Angreifer sendet textbasierte Angriffsskripte, die Eigenschaften des Browsers ausnutzen. Fast jede Datenquelle kann einen Angriffsvektor beinhalten, auch interne Quellen wie Datenbanken.[OWA17, S. 13].

Schwachstellen: XSS ist die am weitesten verbreitete Schwachstelle in Webanwendungen. XSSSchwachstellen treten dann auf, wenn die Anwendung vom Benutzer eingegebene Daten übernimmt, ohne sie hinreichend zu validieren und Metazeichen als Text zu kodieren. Es gibt drei Typen von XSSSchwachstellen: 1) Persistent, 2) nichtpersistent/reflektiert, und 3) DOM-basiert (lokal). Die meisten XSS-Schwachstellen sind verhältnismäßig einfach mit Hilfe von Tests oder Code-Analyse zu erkennen.[OWA17, S. 13].

Technische Auswirkungen: Angreifer können Skripte im Browser des Opfers ausführen und die Session übernehmen, Webseiten entstellen, falsche Inhalte einfügen, Benutzer umleiten, den Browser des Benutzers durch Malware übernehmen.[OWA17, S. 13].

Mögliche Angriffsszenarien:

Szenario 1:
Die Anwendung übernimmt nicht vertrauenswürdige Daten, die nicht auf Gültigkeit geprüft oder escaped werden, um folgenden HTML-Code zu generieren :[OWA17, S. 13]:

```
(String) page += «input name='creditcard' type='TEXT'
value='"+ request.getParameter("CC") + ">";
```

Der Angreifer ändert den Parameter 'CC' in seinem Browser auf[OWA17, S. 13]:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'
```

Dadurch wird die Session-ID des Benutzers an die Seite des Angreifers gesendet, so dass der Angreifer die aktuelle Benutzersession übernehmen kann. Beachten Sie bitte, dass Angreifer XSS auch nutzen können, um jegliche CSRF-Abwehr der Anwendung zu umgehen. A8 enthält weitere Informationen zu CSRF[OWA17, S. 13].

3.1.1.8. Unsichere Deserialisierung

Unsichere Deserialisierung führt häufig zur Remote-Code-Ausführung. Selbst wenn Deserialisierungsfehler keine Remotecodeausführung zur Folge haben, können sie zum Ausführen von Angriffen verwendet werden, einschließlich Wiedergabeangriffe, Injektionsangriffe und Angriffe auf erweiterte Rechte[OWA17, S. 6].

Angriffsvektoren: Die Ausnutzung der Deserialisierung ist schwierig, da die Standard-Exploits selten ohne Änderungen oder Anpassungen des zugrunde liegenden Exploit-Codes funktionieren[OWA17, S. 14].

Schwachstellen: Dieses Problem ist in den Top 10 enthalten und basiert auf einer Branchenumfrage und nicht auf quantifizierbaren Daten. Einige Tools können Deserialisierungsfehler erkennen, aber es wird häufig menschliche Hilfe benötigt, um das Problem zu überprüfen. Es wird erwartet, dass die Prävalenzdaten für Deserialisierungsfehler zunehmen werden, wenn Werkzeug entwickelt werden, um sie zu identifizieren und zu beheben[OWA17, S. 14].

Technische Auswirkungen: Die Auswirkungen von Deserialisierungsfehlern können nicht unterschätzt werden. Diese Fehler können zu Remote-Code-Execution-Angriffen führen, einer der schwerwiegendsten Angriffe, die möglich sind[OWA17, S. 13].

Mögliche Angriffsszenarien:

Szenario 1:
Ein PHP-Forum verwendet die PHP-Objektserialisierung, um ein SSuper-Cookießu speichern, das die Benutzer-ID, die Rolle, den Kennwort-Hash und den anderen Status des Benutzers enthält. [OWA17, S. 13]:

```
(String) page += «input name='creditcard' type='TEXT'
value='"+ request.getParameter("CC") + ">";
```

Der Angreifer ändert den Parameter 'CC' in seinem Browser auf[OWA17, S. 13]:

```
a:4:i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:üser";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";
```

Ein Angreifer ändert das serialisierte Objekt, um sich Administratorrechte zu gewähren:

```
a:4:i:0;i:1;i:1;s:5:Älice";i:2;s:5:ädmin";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";
```

3.1.1.9. Nutzung von Komponenten mit bekannten Schwachstellen

Komponenten wie z.B. Bibliotheken, Frameworks oder andere Softwaremodule werden meistens mit vollen Berechtigungen ausgeführt. Wenn eine verwundbare Komponente ausgenutzt wird, kann ein solcher Angriff zu schwerwiegendem Datenverlust oder bis zu einer Serverübernahme führen. Applikationen, die Komponenten mit bekannten Schwachstellen einsetzen, können Schutzmaßnahmen unterlaufen und so zahlreiche Angriffe und Auswirkungen ermöglichen[OWA17, S. 6].

Angriffsvektoren: Ein Angreifer erkennt Komponenten mit Schwachstellen mittels Scan, oder manueller Analyse. Er passt den Exploit an und führt den Angriff aus. Bei tief eingebetteten Komponenten ist dies schwieriger[OWA17, S. 15].

Schwachstellen: So gut wie jede Anwendung ist von diesem Problem betroffen, da die meisten Entwicklungs-Teams wenig darauf achten, dass die benutzten Komponenten bzw. Bibliotheken aktuell sind. Häufig kennen sie nicht einmal alle Komponenten, oder machen sich keine Gedanken über deren Version. Die rekursive Abhängigkeit von weiteren Bibliotheken verschlechtert die Situation weiter.[OWA17, S. 15].

Technische Auswirkungen: Die ganze Bandbreite von Schwachstellen ist möglich, inkl. Injection, Fehler in der Zugriffskontrolle, XSS usw. Die Auswirkungen können von minimal bis hin zur vollständigen Übernahme des Servers und der Daten reichen[OWA17, S. 15].

Mögliche Angriffsszenarien:

Szenario 1:
Die durch Schwachstellen in Komponenten verursachten Lücken können von minimalen Risiken bis zu ausgeklügelter

Malware führen, die für gerichtete Angriffe geeignet ist. Die Komponenten laufen meist mit allen Anwendungsrechten, wodurch ein Mangel in jeder Komponente schwerwiegend sein kann[OWA17, S. 15].

3.1.1.10. Insufficient Logging & Monitoring

Insufficient Logging und Monitoring in Kombination mit fehlender oder ineffektiver Integration mit Vorfallreaktionen ermöglicht Angreifern, um weitere Systeme anzugreifen und die Daten zu manipulieren, zu extrahieren oder zu zerstören[OWA17, S. 6].

Angriffsvektoren: Angreifer verlassen sich auf das Fehlen von Überwachung und rechtzeitiger Reaktion, um ihre Ziele zu erreichen, ohne entdeckt zu werden[OWA17, S. 16].

Schwachstellen: Eine Strategie, um zu bestimmen, ob Sie eine ausreichende Überwachung haben, besteht darin, die Protokolle nach dem Durchdringungstest zu untersuchen. Die Handlungen der Tester sollten ausreichend protokolliert werden, um zu verstehen, welche Schäden sie verursacht haben[OWA17, S. 16].

Technische Auswirkungen: Die meisten erfolgreichen Angriffe beginnen mit der Prüfung auf Schwachstellen[OWA17, S. 16].

Mögliche Angriffsszenarien:

Szenario 1:
Eine Open Source-Projektforumsoftware, die von einem kleinen Team betrieben wurde, wurde mit einem Fehler in der Software gehackt. Die Angreifer konnten das interne Quellcode-Repository mit der nächsten Version und den gesamten Foreninhalt löschen. Obwohl die Quelle wiederhergestellt werden konnte, führte das Fehlen von Überwachung, Protokollierung oder Alarmierung zu einem viel schlimmeren Verstoß. Das Forumssoftwareprojekt ist aufgrund dieses Problems nicht mehr aktiv[OWA17, S. 16].

Szenario 2:
Ein Angreifer verwendet Scans für Benutzer, die ein allgemeines Kennwort verwenden. Sie können alle Konten mit diesem Passwort übernehmen. Für alle anderen Benutzer hinterlässt dieser Scan nur ein falsches Login. Nach einigen Tagen kann dies mit einem anderen Passwort wiederholt werden[OWA17, S. 16].

3.1.2. Weitere Risiken

3.1.3. Common Vulnerability Scoring System

3.1.4. Common Vulnerability Exposures

3.2. Technische Schwachstellen

3.2.1. Programmierfehler

3.2.2. Konfigurationsfehler

3.2.3. Konzeptionsfehler

3.2.4. Fehler resultierend aus menschlichem Verhalten

4. Penetrationstest

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

4.1. Grundlegendes Konzept

4.1.1. Black-Box

4.1.2. White-Box

4.1.3. Gray-Box

4.2. Kriterien für Penetrationstests

4.3. Ablauf eines Penetrationstest

4.4. Penetrationstest-Tools

4.4.1. ABC

4.5. Automatisierte Penetrationstest-Tools

4.5.1. ABC

4.6. Vor- und Nachteile zwischen manuelle und automatisierte Penetrationstest-Tools

5. Entwicklung einer Spring-Boot-Webanwendung mit Sicherheitslücken

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

6. Entwicklung von Plug-in für das OWASP-ZAP

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

7. Was ist die Vorteile von diesem Plug-in gegenüber anderen Plug-ins?

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

8. Fazit

Hier bitte einen kurzen Durchgang durch die Arbeit.

A. Benutzerhandbuch für das OWASP-Zap RAML Plug-in

> Probleme kann man niemals mit derselben Denkweise lösen, durch die sie entstanden sind.
>
> *(Albert Einstein)*

Pro Satz eine neue Zeile. Das ist wichtig, um sauber versionieren zu können. In LaTeX werden Absätze durch eine Leerzeile getrennt.

Folglich werden neue Abstäze insbesondere *nicht* durch Doppelbackslashes erzeugt. Der letzte Satz kam in einem neuen Absatz.

Wörter am besten mittels `\enquote{...}` „einschließen“, dann werden die richtigen Anführungszeichen verwendet.

Beim Erstellen der Bibtex-Datei wird empfohlen darauf zu achten, dass die DOI aufgeführt wird.

A.1. Quellcode

Listing A.1 zeigt, wie man Programmlistings einbindet. Mittels `\lstinputlisting` kann man den Inhalt direkt aus Dateien lesen.

Quellcode im `<listing />` ist auch möglich.

A.2. Abbildungen

Die Abbildung A.1 und A.2 sind für das Verständnis dieses Dokuments wichtig. Im Anhang zeigt Abbildung A.4 auf Seite 38 erneut die komplette Choreographie.

Es ist möglich, SVGs direkt beim Kompilieren in PDF umzuwandeln. Dies ist im Quellcode zu latex-tipps.tex beschrieben, allerdings auskommentiert.

A.3. Tabellen

Tabelle A.1 zeigt Ergebnisse und die Tabelle A.1 zeigt wie numerische Daten in einer Tabelle representiert werden können.

Listing A.1 lstlisting in einer Listings-Umgebung, damit das Listing durch Balken abgetrennt ist

```
<listing name="second sample">
  <content>not interesting</content>
</listing>
```

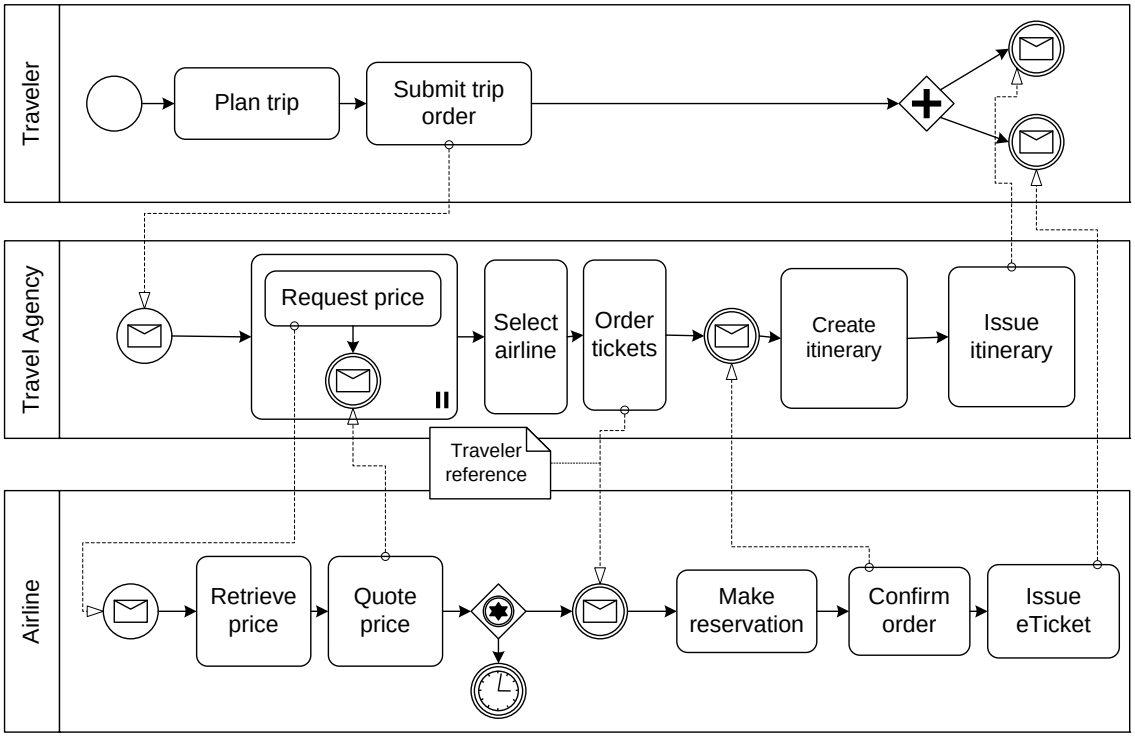


Abbildung A.1.: Beispiel-Choreographie

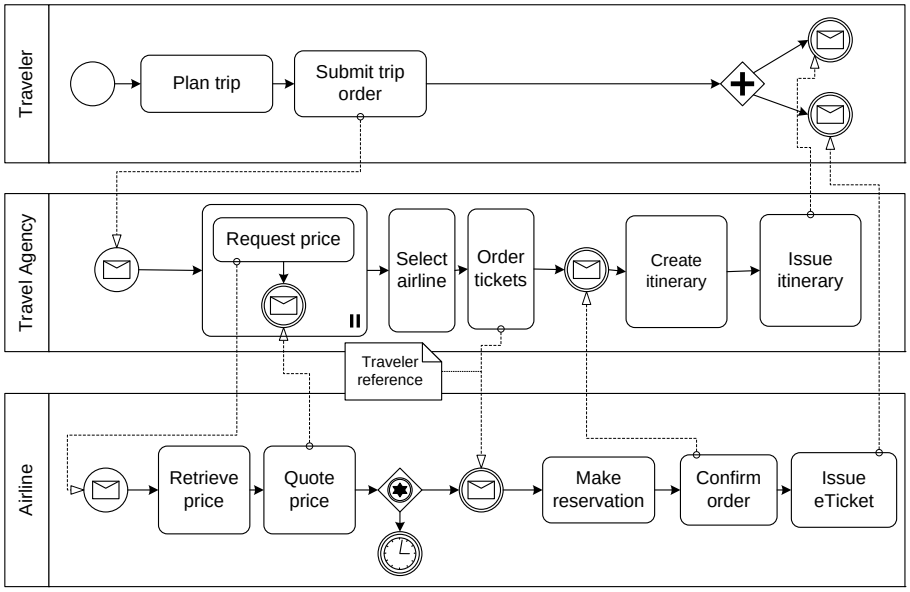
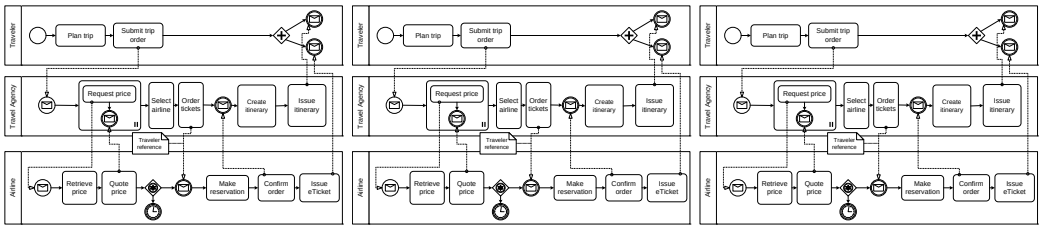


Abbildung A.2.: Die Beispiel-Choreographie. Nun etwas kleiner, damit `\textwidth` demonstriert wird. Und auch die Verwendung von alternativen Bildunterschriften für das Verzeichnis der Abbildungen. Letzteres ist allerdings nur Bedingt zu empfehlen, denn wer liest schon so viel Text unter einem Bild? Oder ist es einfach nur Stilsache?



(a) **(b)** **(c)** Subcaption if needed

Abbildung A.3.: Beispiel um 3 Abbildung nebeneinader zu stellen nur jedes einzeln referenzieren zu können. Abbildung A.3b ist die mittlere Abbildung.

zusammengefasst		Titel
Tabelle	wie	in
tabsatz.pdf	empfohlen	gesetzt
Beispiel	ein schönes Beispiel für die Verwendung von „multirow“	

Tabelle A.1.: Beispieltabelle – siehe http://www.ctan.org/tex-archive/info/german/tabsatz/

A.4. Pseudocode

Algorithmus A.1 zeigt einen Beispielalgorithmus.

	Parameter 1		Parameter 2		Parameter 3		Parameter 4	
Bedingungen	**M**	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
W	1,1	5,55	6,66	,01				
X	22,22	0,0	77,5	,1				
Y	333,3	,1	11,11	,05				
Z	4444,44	77,77	14,06	,3				

Tabelle A.2.: Beispieltabelle für 4 Bedingungen (W-Z) mit jeweils 4 Parameters mit (M und SD). Hinweiß: immer die selbe anzahl an Nachkommastellen angeben.

Algorithmus A.1 Sample algorithm

procedure SAMPLE(a,v_e)
  parentHandled $\leftarrow (a = \text{process}) \vee \text{visited}(a'), (a', c, a) \in \text{HR}$
    // $(a', c'a) \in \text{HR}$ denotes that a' is the parent of a
  if parentHandled $\wedge (\mathcal{L}_{in}(a) = \emptyset \vee \forall l \in \mathcal{L}_{in}(a) : \text{visited}(l))$ **then**
    visited$(a) \leftarrow$ true
    $$\text{writes}_\circ(a, v_e) \leftarrow \begin{cases} \text{joinLinks}(a, v_e) & |\mathcal{L}_{in}(a)| > 0 \\ \text{writes}_\circ(p, v_e) & \exists p : (p, c, a) \in \text{HR} \\ (\emptyset, \emptyset, \emptyset, false) & \text{otherwise} \end{cases}$$
    if $a \in \mathcal{A}_{basic}$ **then**
      HANDLEBASICACTIVITY(a,v_e)
    else if $a \in \mathcal{A}_{flow}$ **then**
      HANDLEFLOW(a,v_e)
    else if $a =$ process **then** // Directly handle the contained activity
      HANDLEACTIVITY(a',v_e), $(a, \perp, a') \in \text{HR}$
      writes$_\bullet(a) \leftarrow$ writes$_\bullet(a')$
    end if
    for all $l \in \mathcal{L}_{out}(a)$ **do**
      HANDLELINK(l,v_e)
    end for
  end if
end procedure

Und wer einen Algorithmus schreiben möchte, der über mehrere Seiten geht, der kann das nur mit folgendem **üblen** Hack tun:

Algorithmus A.2 Description

code goes here
test2

A.5. Abkürzungen

Beim ersten Durchlauf betrug die Fehlerrate (FR) 5. Beim zweiten Durchlauf war die FR 3. Die Pluralform sieht man hier: error rates (ERs). Um zu demonstrieren, wie das Abkürzungsverzeichnis bei längeren Beschreibungstexten aussieht, muss hier noch Relational Database Management Systems (RDBMS) erwähnt werden.

Mit `\gls{...}` können Abkürzungen eingebaut werden, beim ersten Aufrufen wird die lange Form eingesetzt. Beim wiederholten Verwenden von `\gls{...}` wird automatisch die kurz Form angezeigt. Außerdem wird die Abkürzung automatisch in die Abkürzungsliste eingefügt. Mit `\glspl{...}` wird die Pluralform verwendet. Möchte man, dass bei der ersten Verwendung direkt die Kurzform erscheint, so kann man mit `\glsunset{...}` eine Abkürzung als bereits verwendet markieren. Das Gegenteil erreicht man mit `\glsreset{...}`.

Definiert werden Abkürzungen in der Datei *content ausarbeitung.tex* mithilfe von `\newacronym{...}{...}{...}`.

Mehr Infos unter: http://tug.ctan.org/macros/latex/contrib/glossaries/glossariesbegin.pdf

A.6. Definitionen

Definition A.6.1 (Title)
Definition Text

Definition A.6.1 zeigt …

A.7. Fußnoten

Fußnoten können mit dem Befehl `\footnote{...}` gesetzt werden[1]. Mehrfache Verwendung von Fußnoten ist möglich indem man zu erst ein Label in der Fußnote setzt `\footnote{\label{...}...}` und anschließend mittels `\cref{...}` die Fußnote erneut verwendet[1].

A.8. Verschiedenes

Ziffern (123 654 789) werden schön gesetzt. Entweder in einer Linie oder als Minuskel-Ziffern. Letzteres erreicht man durch den Parameter `osf` bei dem Paket `libertine` bzw. `mathpazo` in `fonts.tex`.

Kapitälchen werden schön gesperrt...

I. Man kann auch die Nummerierung dank paralist kompakt halten
II. und auf eine andere Nummerierung umstellen

[1] Diese Fußnote ist ein Beispiel.

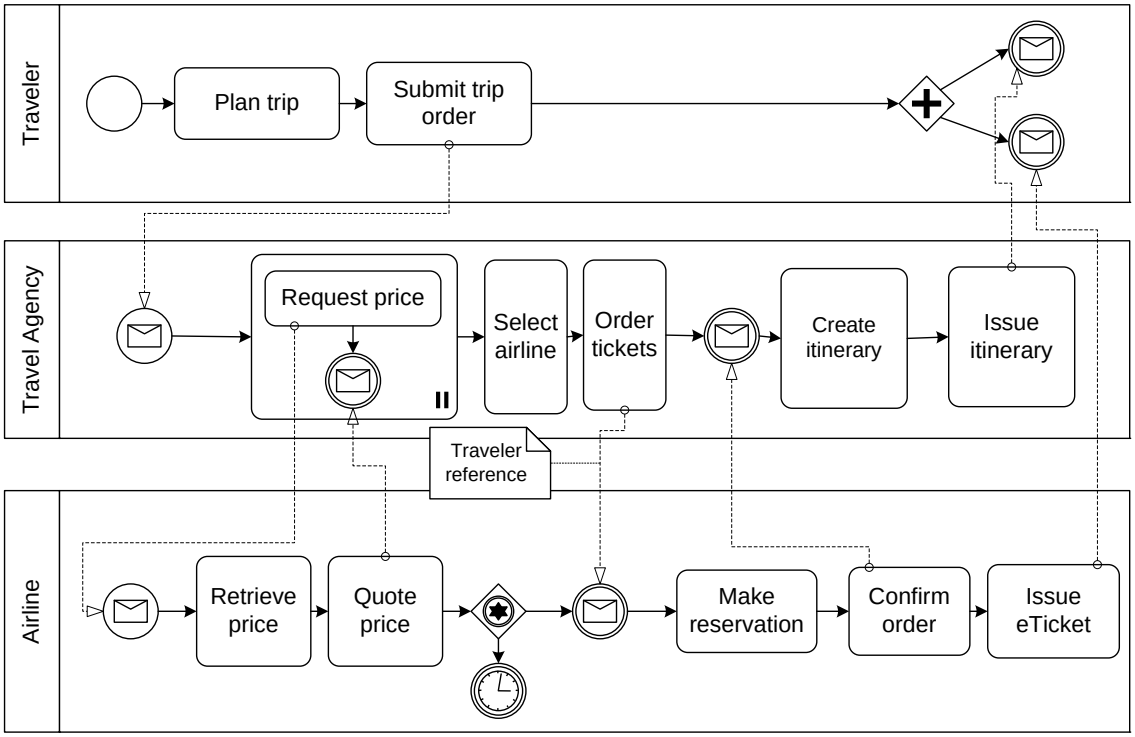


Abbildung A.4.: Beispiel-Choreographie I

A.9. Weitere Illustrationen

Abbildungen A.4 und A.5 zeigen zwei Choreographien, die den Sachverhalt weiter erläutern sollen. Die zweite Abbildung ist um 90 Grad gedreht, um das Paket `pdflscape` zu demonstrieren.

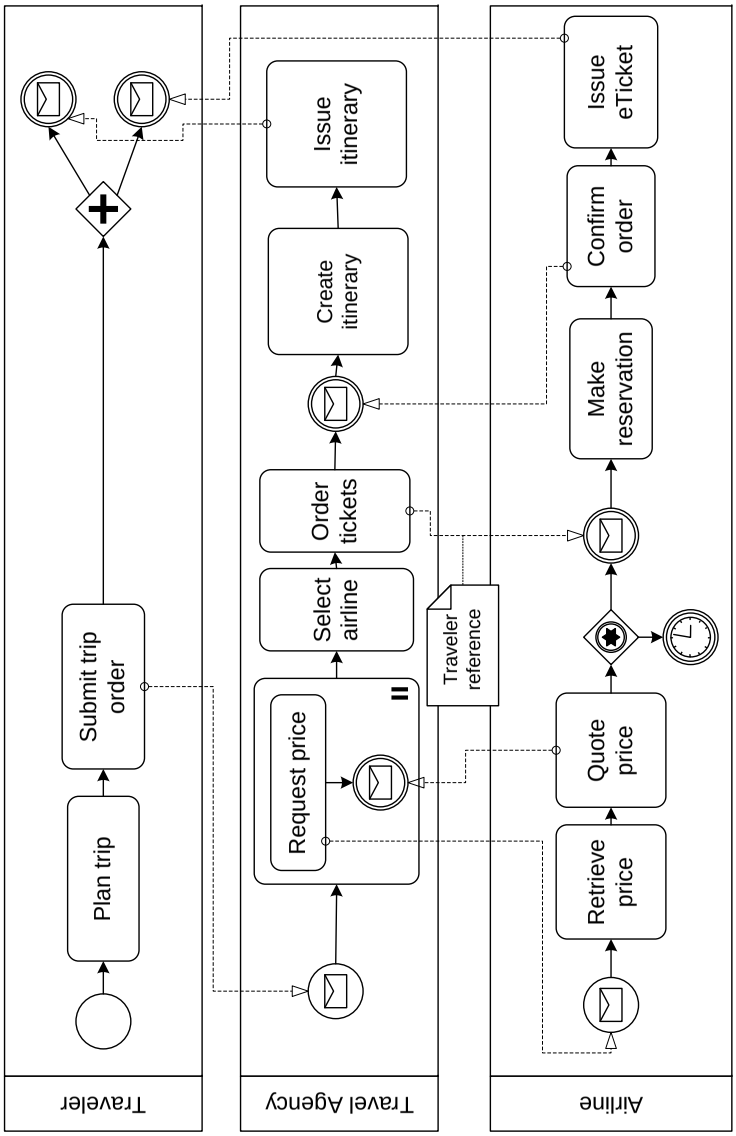


Abbildung A.5.: Beispiel-Choreographie II

A.10. Plots with pgfplots

Pgfplot ist ein Paket um Graphen zu plotten ohne den Umweg über gnuplot oder matplotlib zu gehen.

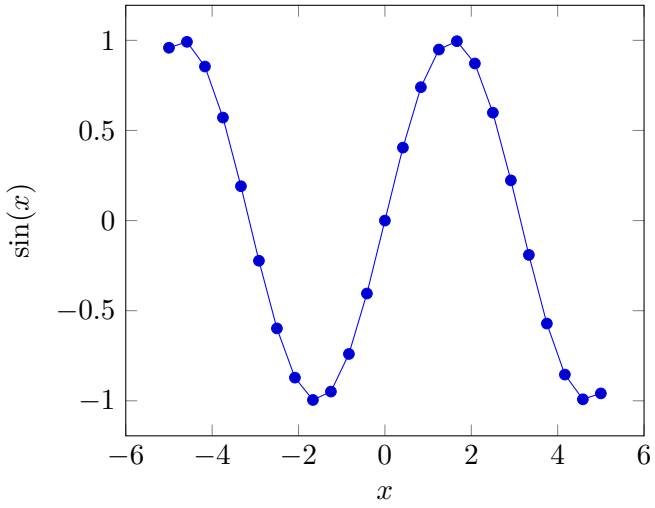


Abbildung A.6.: $\sin(x)$ mit pgfplots.

A.11. Figures with tikz

TikZ ist ein Paket um Zeichnungen mittels Programmierung zu erstellen. Dieses Paket eignet sich um Gitter zu erstellen oder andere regelmäßige Strukturen zu erstellen.

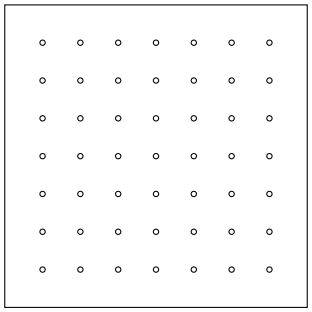


Abbildung A.7.: Eine tikz-Graphik.

Literaturverzeichnis

[BFSI07] Bundesamt für Sicherheit in der Informationstechnik. *Glossar und Begriffsdefinitionen*. 2007. URL: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/Glossar/glossar_node.html (zitiert auf S. 6).

[BKRS13] J. Blieberger, J. Klasek, A. Redlein, G.-H. Schildt. *Informatik*. Springer-Verlag, 2013 (zitiert auf S. 4).

[BN17] M. von Symantec. *Was ist ein Botnet?* 2017. URL: https://de.norton.com/internetsecurity-malware-what-is-a-botnet.html (zitiert auf S. 9).

[Bro13] M. Broy. *Informatik Eine grundlegende Einführung: Band 1: Programmierung und Rechnerstrukturen*. Bd. 1. Springer-Verlag, 2013 (zitiert auf S. 4).

[BSI08] Bildungsstandards Informatik. *Information und Daten*. 2008. URL: https://www.informatikstandards.de/index.htm?section=standards&page_id=10 (zitiert auf S. 5).

[BSIDI11] „Internet-Sicherheit: Einführung, Grundlagen, Vorgehensweise". In: *Bundesamt für Sicherheit in der Informationstechnik* (2011) (zitiert auf S. 5).

[CPSW12] Cyberpedia. *What is Spyware?* 2012. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-spyware (zitiert auf S. 8).

[DsBa18] Datenschutzbeauftragter. *Pseudonymisierung – was ist das eigentlich?* 2018. URL: https://www.datenschutzbeauftragter-info.de/pseudonymisierung-was-ist-das-eigentlich/ (zitiert auf S. 7).

[Ebe94] J. Eberspächer. „Sichere Daten, sichere Kommunikation". In: *Secure Infor-mation, Secure Communication. Datenschutz und Datensicherheit in Tele-kommunikations- und Informationssystemen. Privacy and Information Security in Communication and Information Systems, Berlin, Heidelberg, New York* (1994) (zitiert auf S. 5).

[Eck13] C. Eckert. *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. Walter de Gruyter, 2013 (zitiert auf S. 4–7, 9).

[Fur08] S. Furnell. *Securing information and communications systems: Principles, technologies, and applications*. Artech House, 2008 (zitiert auf S. 3).

[Gor03] W. Gora. *Handbuch IT-Sicherheit: Strategien, Grundlagen und Projekte*. Addison-Wesley, 2003 (zitiert auf S. 6).

[J18] V. Beal. *Java*. 2018. URL: https://www.webopedia.com/TERM/J/Java.html (zitiert auf S. 9).

[MHTK15] N. Menz, P. Hoepner, J. Tiemann, F. Koußen. „Safety und Security aus dem Blickwinkel der öffentlichen IT". In: *Kompetenzzentrum Öffentliche IT* (2015) (zitiert auf S. 5).

[MVC12] T. Point. *Spring - MVC Framework*. 2012. URL: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm (zitiert auf S. 11).

[Now11] T. Nowey. „Einleitung“. In: *Konzeption eines Systems zur überbetrieblichen Sammlung und Nutzung von quantitativen Daten über Informationssicherheitsvorfälle*. Springer, 2011 (zitiert auf S. 7).

[OOP15] V. Beal. *OOP – Object Oriented Programming*. 2015. URL: https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html (zitiert auf S. 9).

[OWA17] OWASP. „OWASP Top 10“. In: *The Ten Most Critical Web Application Security Risks*. Creative Commons, 2017 (zitiert auf S. 15–21).

[OZ18] OWASP. *OWASP Zed Attack Proxy Project*. 2018. URL: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project (zitiert auf S. 12).

[PHS17] Phishing.org. *What is Phishing?* 2017. URL: http://www.phishing.org/what-is-phishing (zitiert auf S. 8).

[RAML13] RAML. *About RAML*. 2013. URL: https://raml.org/about-raml (zitiert auf S. 12).

[RWS13] O. Docs. *What Are RESTful Web Services?* 2013. URL: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html (zitiert auf S. 10).

[Sch17] P. Schmitz. *Was ist ein Hacker?* 2017. URL: https://www.security-insider.de/was-ist-ein-hacker-a-596399/ (zitiert auf S. 7).

[SePe11] Oliver Wege. *Verbindlichkeit*. 2011. URL: https://www.secupedia.info/wiki/Verbindlichkeit (zitiert auf S. 6).

[Sill11] P. M. D. H. Siller. *Hacker – Definition*. 2011. URL: https://wirtschaftslexikon.gabler.de/definition/hacker-53395 (zitiert auf S. 7).

[Spi85] P. P. Spies. „Datenschutz und Datensicherung im Wandel der Informationstechnologien“. In: *Datenschutz und Datensicherung im Wandel der Informationstechnologien*. Springer, 1985 (zitiert auf S. 6).

[STS18] Eric Weis. *Was ist der Unterschied zwischen IT Sicherheit und Informationssicherheit*. 2018. URL: https://www.brandmauer.de/blog/it-security/unterschied-it-sicherheit-und-informationssicherheit (zitiert auf S. 4).

[Swa17] Swagger. *What Is OpenAPI?* 2017. URL: https://swagger.io/docs/specification/about/ (zitiert auf S. 11).

[TNSA14] Techopedia. *National Security Agency (NSA)*. 2014. URL: https://www.techopedia.com/definition/15146/national-security-agency-nsa (zitiert auf S. 8).

[TSC11] Techopedia. *Script Kiddie*. 2011. URL: https://www.techopedia.com/definition/4090/script-kiddie (zitiert auf S. 8).

[Whe13] T. Wheeler. *Offene Systeme: ein grundlegendes Handbuch für das praktische DV-Management*. Springer-Verlag, 2013 (zitiert auf S. 4).

[Wil18] A. Wilkinson. *Spring REST Docs*. 2018. URL: https://docs.spring.io/spring-restdocs/docs/2.0.2.RELEASE/reference/html5/ (zitiert auf S. 11).

[Wol16] E. Wolff. *Microservices: flexible software architecture*. Addison-Wesley Professional, 2016 (zitiert auf S. 10).

Alle URLs wurden zuletzt am 10.01.2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift