

# So ein Saftladen: IT-Sicherheit in handlungsorientierten Fallbeispielen

Kokula, R., Kufner, H., Leesch, B., Reinold, K., Scholz, J., Wich, S., Winter, P.

DOI: 10.18420/ibis-03-01-07

## Zusammenfassung

IT-Sicherheit ist ein Thema, das immer dann in den öffentlichen Diskurs kommt, wenn Systeme gehackt oder anderweitig gestört werden. Auch in den schulischen Lehrplänen gewinnt es zunehmend an Bedeutung. Der Artikel skizziert eine Annäherung an das Thema in Form von Fallbeispielen auf der Basis des OWASP Juice Shop.

## Einleitung

IT-Sicherheit ist ein für Gesellschaft und Wirtschaft, aber auch für den privaten Anwender relevantes Thema, das zunehmend an Bedeutung gewinnt. Die nachfolgende Abbildung stellt wesentliche Risiken zusammen, denen IT-Systeme ausgesetzt sind.

Insbesondere die Verhinderung und Abwehr gezielter Angriffe ist in Zeiten von Cyberkriminalität und Cyberkrieg von globaler Relevanz. So schreibt das Bundesamt für Sicherheit in der Informationstechnik (BSI 2024):

„[...] ganz Deutschland ist aufgerufen, eigene Angriffsflächen zu ermitteln und zu



Abbildung 1: Gefahren für IT-Systeme (Brichzin et al. 2024) © Cornelsen Verlag GmbH, Berlin 2024

schützen. Das ist in historisch gewachsenen IT-Landschaften eine große Herausforderung, aber notwendig, denn die Angreifer suchen beständig nach neuen Angriffsvektoren.“ (BSI 2024, S. 8)

Zur Vermeidung von erfolgreichen Angriffen ist die Schaffung eines breiten gesellschaftlichen Problembewusstseins ein zentraler Faktor:

„Im alltäglichen Umgang mit IT-Systemen ist Awareness eine elementare Sicherheitsmaßnahme. Das bedeutet zunächst, dass ein Problembewusstsein für Cyber-Sicherheit geschaffen werden muss. Darauf aufbauend kann man eine Verhaltensänderung hin zu sicherem digitalen Umgang erreichen. Security Awareness Maßnahmen sind dann erfolgreich, wenn sie die Zielgruppen befähigen und den einzelnen Menschen für mehr Cyber-Sicherheit motivieren.“ (BSI 2025)

Die allgemeinbildende Relevanz des Themas Informationssicherheit zeigt sich auch darin, dass das Thema Einzug in die aktuellen Lehr- und Bildungspläne findet. Exemplarisch seien

hier die Pläne von Nordrhein-Westfalen, Mecklenburg-Vorpommern und Bayern zitiert:

„Die Schülerinnen und Schüler [...] untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen.“ (QUA-LiS 2024)

„Die Schülerinnen und Schüler [...] konzipieren Maßnahmen zur Realisierung von Datensicherheit für konkrete Anwendungsfälle, insbesondere Zugriffskontrolle.“ (Institut für Qualitätsentwicklung 2019)

„Die Schülerinnen und Schüler [...] analysieren exemplarisch ein Informatiksystem (z. B. Smartphone-App, Smarthome-System, Informatiksystem eines Unternehmens) hinsichtlich der Umsetzung wichtiger Schutzziele der Informationssicherheit und bewerten das Erreichen dieser Ziele. [Sie] beschreiben verschiedene Arten der Gefährdung eines Informatiksystems und analysieren ein mögliches Angriffsszenario. [Sie] erläutern Maßnahmen, die die Informationssicherheit gewährleisten sollen. In diesem Kontext werden ihnen technische und wirtschaftliche Grenzen bewusst. [Sie] erörtern verschiedene Perspektiven einer Fragestellung der Informationssicherheit, z. B. Offenlegung oder Nichtoffenlegung von Schwachstellen. Dabei berücksichtigen sie individuelle und gesellschaftliche Auswirkungen.“ (ISB 2025)

Es wird deutlich, dass man sich nicht auf eine gesellschaftlich-kulturell-wirtschaftliche und anwendungsbezogene Perspektive beschränken kann, um ein Bewusstsein für die Gefährdungsszenarien von IT-Systemen zu erreichen. Um ein Verständnis für die zugrundeliegenden Wirkprinzipien zu erzeugen, muss auch - zumindest exemplarisch - der technologische Hintergrund aufgezeigt werden; es ist dazu notwendig, dass die Lernenden Fälle auch aus der Perspektive des Angreifenden analysieren. Dabei besteht das Problem, dass nach den sogenannten „Hackerparagraphen“ BGB §202a und §202c das unbefugte Ausspähen und allein die Vorbereitung darauf strafbare Handlungen sind, wofür die Lernenden im Rahmen der Unterrichtssequenz auch sensibilisiert werden sollten. Experimente müssen also in einem System stattfinden, das den Zugriffsversuch explizit zulässt. Ein solches System ist der nachfolgend vorgestellte OWASP Juice Shop. Der Fokus der Angriffe (vgl. Abb. 1) liegt dabei auf den besonders relevanten Bereichen Diebstahl, Datenspionage und Manipulation von IT-Systemen.

OWASP steht für Open Worldwide Application Security Project. Es handelt sich um eine gemeinnützige Stiftung, die sich für die Verbesserung der IT-Sicherheit einsetzt und die ihre Projekte frei, offen und kostenlos bereitstellt. Unter anderem führt OWASP mit der „OWASP Top 10“ eine anerkannte, regelmäßig aktualisierte Liste der kritischsten Angriffsszenarien für Webanwendungen (OWASP Top Ten, 2025). Der Juice-Shop (OWASP Juice Shop, 2025), der im Rahmen der nachfolgenden Fallbeispiele vorgestellt wird, bietet einen „Spiel- und Experimentierplatz“, um sich praktisch mit gängigen

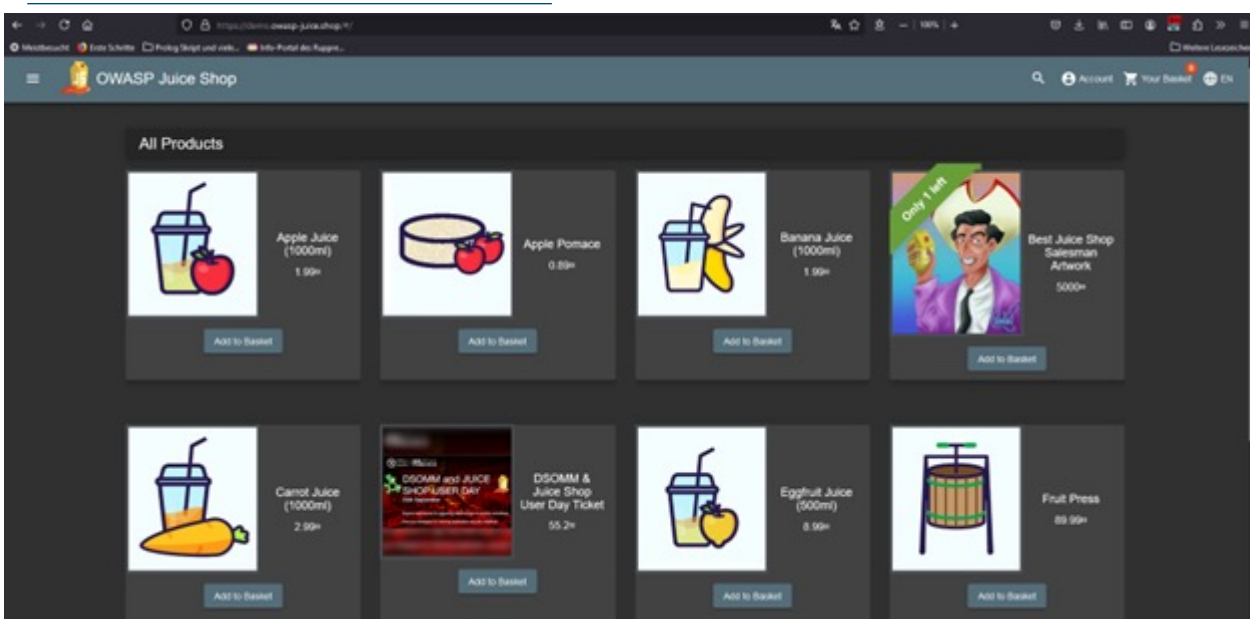


Abbildung 2: demo.owasp-juice.shop (OWASP Foundation/ CC BY-SA 4.0)

Angriffsszenarien vertraut zu machen. Unter <http://demo.owasp-juice.shop> steht eine regelmäßig aktualisierte Deployment-Test-Version ohne garantierte Verfügbarkeit bereit. Eine weitere Instanz findet man unter <https://juice-shop.herokuapp.com>. Für intensivere Auseinandersetzung empfiehlt es sich, auf Basis von node.js eine lokale Instanz bzw. einen eigenen Juice-Shop-Server zu installieren.

Die nachfolgend beschriebenen Fallbeispiele sind für den konkreten Unterrichtseinsatz konzipiert, z. B. in Form einer Expertenarbeit in Kleingruppen. Je nach Kursgröße können bestimmte Fallbeispiele weggelassen bzw. weitere ergänzt werden. Jedes Szenario steht dabei exemplarisch für einen typischen Angriffsvektor, der so oder in ähnlicher Form bei echten Cyberangriffen verwendet wurde. Ziel ist es, dass Schülerinnen und Schüler erkennen, dass die Vektoren sehr vielfältig sein können und Angreifer sich typischerweise über eine gefundene Schwachstelle Zugang zu weiteren Teilen der Systeme verschaffen können.

## Angriffsszenario 1: Anmelden als Admin mittels SQL-Injection

**Beschreibung des Szenarios:** Bei Webanwendungen werden oft im Frontend Eingabefelder benutzt, die im Backend in SQL-Statements übersetzt werden. Das Beispiel zeigt eine typische Anmeldemaske und die dadurch ausgelöste SQL-Abfrage zur Überprüfung der Logindaten.

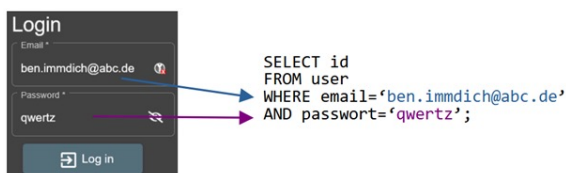


Abbildung 3: Anmeldemaske und beispielhafte Übersetzung in SQL-Anweisung (Klaus Reinold/ OWASP Foundation/ CC BY-SA 4.0)

ten.

**Angriffsvektor:** Werden die eingegebenen Daten aus dem Frontend direkt und ungeprüft in einen SQL-Befehl eingesetzt, so kann ein Angreifer dies gezielt ausnutzen. Dazu nutzt man Elemente der Syntax von SQL, beispielsweise das Apostroph zur Klammerung von Textdaten, das Semikolon zur Abtrennung zweier SQL-Befehle oder einen doppelten Bindestrich, um den Rest einer Zeile als Kommentar zu kennzeichnen. Trägt man nun im Feld der E-Mail-Kennung die auf den ersten Blick sinnlose Phrase ' OR TRUE; -- ein, so wird erst bei genauerer

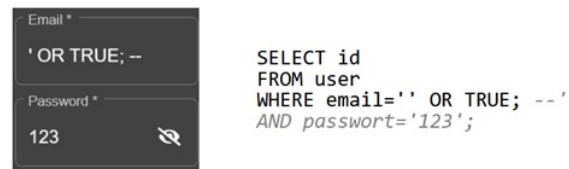


Abbildung 4: SQL-Injection (Klaus Reinold/OWASP Foundation/ CC BY-SA 4.0)

Betrachtung des entsprechenden SQL-Befehls klar, was man dadurch erreicht:

Es entsteht eine SQL-Abfrage mit einer Bedingung, bei der der erste Teil (email= ' ') wohl für keinen Benutzer wahr ist. Durch die Verknüpfung OR TRUE entsteht aber insgesamt eine Bedingung, die für alle Datensätze zutrifft. Durch den doppelten Bindestrich wird der Rest der Abfrage auskommentiert, also irrelevant. Man erhält somit eine Ergebnistabelle, die alle Benutzer-IDs umfasst. Es lässt sich vermuten, dass das Entwicklungsteam aufgrund der Eindeutigkeit von Benutzernamen davon ausgegangen ist, dass höchstens ein Datensatz in der Ergebnistabelle vorhanden sein kann. Die Anmeldung erfolgt daraufhin für den ersten Datensatz der Benutzertabelle, welches häufig – so auch hier – ein Administrator ist. Jackpot für den Angreifer, er ist nun als privilegierter Nutzer im System angemeldet, ganz ohne die Eingabe eines richtigen Passworts.

**Fazit:** Statt sich am System anzumelden und damit Zugang zu erlangen, könnte der Angreifer auch die Verfügbarkeit des Systems angreifen und ' OR TRUE; SHOW TABLES; -- ausprobieren, um die Tabellenbezeichner zu ermitteln. Mit dem TRUNCATE-Befehl könnte man diese Tabellen dann leeren. Die Folge wäre eine Katastrophe für den Anbieter, da alle seine Kunden in der Tabelle gelöscht würden und sich niemand mehr am System anmelden könnte.

Zum Schutz vor SQL-Injections nutzt man heute standardmäßig Prepared Statements. Deren Grundidee ist es, SQL-Befehl und Daten nicht in einer Zeichenkette an den SQL-Interpreter zu übergeben, sondern Befehl und Daten zu trennen. Erst nach Interpretation des Befehls werden dann die Daten eingefüllt und der Befehl ausgeführt. Trotz seit langem bekannter Gegenmaßnahmen stehen Injections aber immer noch in der Liste der gängigsten und erfolgreichsten Angriffsszenarien.

## Angriffsszenario 2: „Security through obscurity“

**Beschreibung des Szenarios:** In der IT-Sicherheit gibt es verschiedene Ansätze, um Systeme und Informationen vor unbefugten Zugriffen zu schützen. Um sich komplizierte Sicherheitsmechanismen zu sparen, ist ein sehr einfacher Ansatz „Security through obscurity“: Dabei wird die Sicherheit eines Systems dadurch gewährleistet, dass der genaue Ort von Daten unbekannt ist oder ein verwendeter Algorithmus geheim gehalten wird. Dass dies kein sinnvoller Ansatz ist, besagt das Kerckoffsche Prinzip (Koch 2024), nach welchem die Sicherheit eines Systems niemals von der Geheimhaltung des Designs abhängen darf.

Dies hat mehrere Gründe:

- Ein geheimer Speicherort kann gefunden werden und ein geheimer Algorithmus kann durch Reverse Engineering rekonstruiert werden.
- Sollte ein Speicherort oder Algorithmus bekannt werden, ist es deutlich schwieriger, zu einem alternativen Speicherort oder Algorithmus zu wechseln, als einen neuen Schlüssel zu erstellen.
- Je mehr Fachleute sich mit einem Sicherheitsverfahren beschäftigen, desto eher werden Schwachstellen in diesem entdeckt.

Das Einführungsbeispiel in diesem Szenario ist die sogenannte Steganographie. Dabei wird In-

formation so verborgen, dass diese für Dritte bei einer oberflächlichen Betrachtung nicht zu sehen ist. Es lassen sich beispielsweise mehrere Bilder zu einer Datei kombinieren, von denen im Browser nur eines angezeigt wird.

**Angriffsvektor:** Dieses Verfahren findet Anwendung in der „About us“-Seite des „Saftladens“. Betrachtet man mit Hilfe der Entwicklerwerkzeuge den Quellcode der Webseite im Browser genauer (STRG+UMSCHALT+I oder F12), so stellt man fest, dass unerwartet eines der sieben Bilder bei gleichen Proportionen eine abweichende Dateiendung besitzt: *Bild 5* ist vom Typ *png*. Speichert man diese Datei, indem man den Pfad aus dem Quellcode kopiert und in einem neuen Tab aufruft, so kann man sie danach genauer untersuchen.

Mit frei verfügbaren Programmen wie *OpenStego*<sup>1</sup> lassen sich durch Steganographie überlagerte Bilder wieder trennen. Öffnet man hier die Datei *Bild 5*, so findet man das Bild eines Charakters aus der Fernsehserie „Rick and Morty“. In diesem Beispiel ist das geheime Bild eher ein versteckte Hommage; in anderen Fällen könnte an dieser Stelle ebenso ein Passwort stehen. Mit *OpenStego* lassen sich mit wenigen Klicks auch eigene Dateien in anderen Dateien verstecken.

**Fazit:** Der Ansatz „Security through obscurity“ ist mit einem Schlüssel unter der Fußmatte eines Hauses zu vergleichen. Solange niemand

<sup>1</sup><https://www.openstego.com>

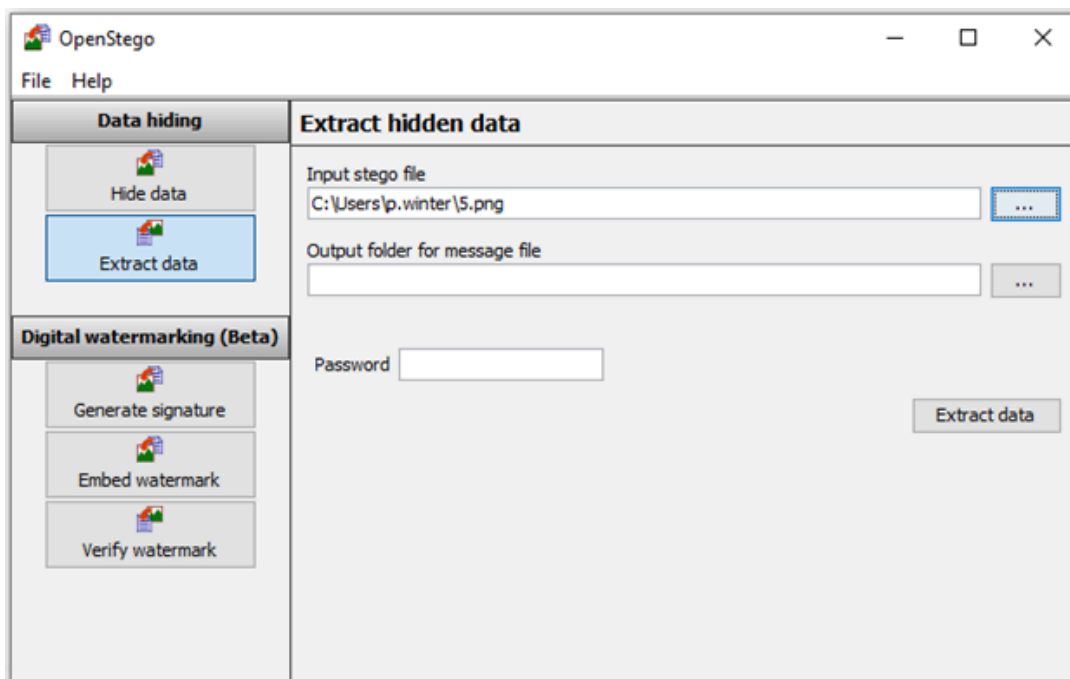


Abbildung 5: Screenshot OpenStego (Philipp Winter/ CC BY-SA 4.0; Samir Vaidya GPL2.0)

genauer danach sucht, ist das Haus sicher. Weiß man aber, wonach man suchen muss, sieht es schnell ganz anders aus. Der „Saftladen“ nutzt an vielen Stellen solche vermeintliche „Sicherheitsmaßnahmen“ und es werden in den folgenden Szenarien noch weitere betrachtet.

### Angriffsszenario 3 – Inhalte und Struktur des Webservers auslesen

**Beschreibung des Szenarios:** Im Saftladen lassen sich Artikel erwerben, indem man sie zunächst in den Warenkorb legt. Vor dem Abschluss des Kaufs bietet sich die Möglichkeit, einen Coupon-Code einzugeben, der einen prozentualen Nachlass gewährt. Der Social-Media-Account des Shops veröffentlicht regelmäßig solche Codes, um Kunden anzulocken, ein beispielhafter Code ist `k#pDlG7sn`. Nun wäre es interessant, selbst einen möglichst hochwertigen Rabattcoupon zu erzeugen. Dazu benötigt man Kenntnis über die Struktur der Rabattcoupons. Oft sind nicht über einen Link zugängliche Dateien auf dem Server eine Quelle für solche Informationen. Es wird bei der Bearbeitung dieser Sicherheitslücke wiederum deutlich, dass „Security Through Obscurity“ keine sinnvolle Strategie zum Schutz einer Website ist.

**Angriffsvektor:** Grundsätzlich kann eine Website als Ordnerstruktur gesehen werden, in der die verschiedenen Ordner und Dateien jeweils über eine eindeutige URL erreichbar sind. Um eventuell ungesicherte Dateien an bestimmten URLs zu finden, lohnt es sich, bei der Navigation durch die Seite einen gezielten Blick auf die Pfade zu werfen – so wird auf der Seite About Us ein Link zu [Check out our boring terms of use if you are interested in such lame stuff](#) angegeben. Nach Klick auf den Link wird die Datei <https://juice-shop.herokuapp.com/ftp/legal.md> geöffnet, worin uninteressanter Text steht. Jedoch zeigt die URL, dass es einen Ordner namens `ftp` auf dem Server gibt, der die Datei `legal.md` enthält. Eingabe der URL ohne den Dateinamen eröffnet einen Blick in das entsprechende Verzeichnis (Abb. 6), das offensichtlich nicht für den öffentlichen Zugriff gedacht ist.

**Fazit:** Fehlkonfigurationen beim Webserver erlauben es mit relativ wenig Aufwand, Außenstehenden einen Einblick in die Inhalte des Webservers zu erhalten. Diese Einblicke in Struktur und Inhalte können dann Ausgangspunkt für weitere Aktivitäten sein.

### Angriffsszenario 4 – Fälschen von Rabattcoupons

**Beschreibung des Szenarios:** Beim Betrachten der auf dem Server hinterlegten Daten scheinen die Dateien `coupons_2013.md.bak` und `package.json.bak` Hinweise zur Coupongenerierung zu enthalten. Leider erscheint bei Aufruf dieser Dateien einer Fehlermeldung, dass der Zugriff nur auf Dateien im Format `.md` und `.pdf` erlaubt ist. Dieser Schutz lässt sich durch Einsatz des sogenannten *Poison Null Byte* umgehen.

**Angriffsvektor:** Die Länge einer Zeichenkette wird entweder durch die explizite Angabe der Länge oder durch die Verwendung eines speziellen Symbols am Ende des Strings erreicht. Letzteres ist als Null Byte (`%00`) bekannt. Dieses Null Byte kann verwendet werden, um die Beschränkung auf bestimmte Dateitypen zum Download zu umgehen. Dazu wird die URL der Datei so angepasst, dass ein Null Byte nach der eigentlichen Dateiendung eingefügt und danach eine passende Endung ergänzt wird, welche die Dateiendung in den „Augen“ des Servers ändert und so die Sperre umgeht. So wird aus

```
~/ftp/coupons_2013.md.bak
```

per Null Byte

```
~/ftp/coupons_2013.md.bak%00.md
```

Diese Herangehensweise funktioniert jedoch noch nicht, da URLs ein eigenes Encoding haben – das `%`-Symbol (vom Null Byte `%00`) muss daher auch mit der Zeichenkette `%25` encodiert werden. Der Aufruf von

```
~/ftp/coupons_2013.md.bak%2500.md
```

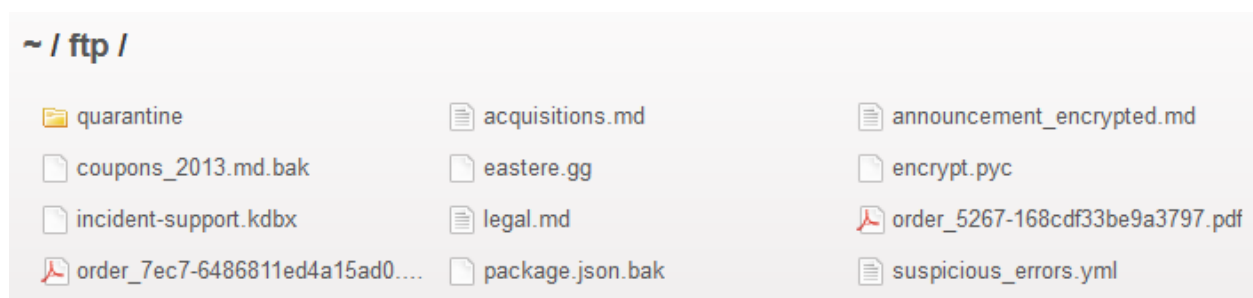


Abbildung 6: Ein Blick hinter die Kulissen des OWASP Juice Shops (Han-Min Kufner/ CC BY-SA 4.0)

hat den Download der Datei zur Folge. Analog kann für andere interessante Dateien im Ordner verfahren werden.

Die Datei package.json.bak ist von großem Wert, da sie offenlegt, welche Codebibliotheken bei der Entwicklung der Website verwendet wurden. Für die Coupongenerierung ist es naheliegend, eine Bibliothek für Hashing-Algorithmen oder andere kryptografische Verfahren zu ver-

im Admin-Bereich einsehen; als eingeloggter Admin agiert man quasi wie ein normaler User. Allerdings kann man sich trotzdem diesen Zugang erschleichen, z.B. um in diesem Szenario 5-Sterne-Feedback zu löschen und dadurch den Shop schlechter darzustellen.

**Angriffsvektor:** Zum Admin-Bereich kann man nur Zugriff erhalten, indem man sich wie in Szenario 1 beschrieben als Admin anmeldet. Der



Abbildung 7: Nutzung eines Online-Tools zur Codierung/Decodierung mit dem z85-Verfahren <https://cryptii.com/pipes/z85-encoder> (Han-Min Kufner/ CC BY-SA 4.0)

muten. Ein Blick auf die Liste der verwendeten Pakete enthält auch z85, ein Codierungsverfahren.

Probeweises Dekodieren eines Coupons, z.B. „l}6D\$gC7ss“ aus der Coupon-Backup-Datei mit dem z85-Verfahren ergibt die Zeichenkette „DEC13-15“.

Weitere Coupon-Decodierungen verdeutlichen, dass der Code mit einem Monatskürzel und dem zugehörigen Jahr beginnt und nach dem Bindestrich angibt, wie viel Prozent der Rabatt betragen soll. Nun lassen sich nach Herzenswunsch Coupons erstellen, so z.B. ein Coupon für den aktuellen Monat (DEC24 für den Autor) mit 99% Nachlass. Dazu kodiert man den String „DEC24-99“ und erhält: l}6D#g+yZF.

**Fazit:** Bei diesem Szenario zeigt sich erneut, dass ein bloßes Verstecken von kritischer Struktur keine nachhaltige Strategie zur Sicherung einer Website sein kann. Auch wenn solche Information die Seite nicht unmittelbar kompromittieren muss, so lassen sich dadurch oft wichtige weitere Angriffsvektoren identifizieren und es wird zunehmend schwieriger, alle Eventualitäten abzusichern.

## Angriffsszenario 5: Unbefugter Zugriff auf den Admin-Bereich

**Beschreibung des Szenarios:** Das Abtrennen des Admin-Bereichs vom Administrator-Account kann bereits schützen, da ein Angreifer, der sich mit Angriffsszenario 1 erfolgreich als Admin eingeloggt hat, weniger Schaden anrichten kann. Im Juice Shop kann man die Liste der registrierten User und das Käufer-Feedback nur

Bereich ist zwar nicht als Verweis anklickbar im Menü hinterlegt; man kann jedoch die URL einfach erraten, durch Vergleich mit den URLs verschiedener Bereiche. Um zum Administrationsbereich zu gelangen, wird das Stichwort „administration“ an den Teil „/#/“ angehängt. Nun

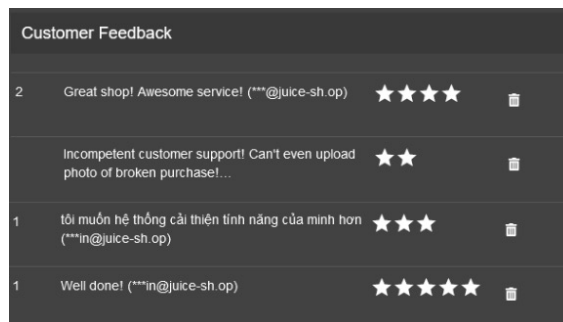


Abbildung 8: Ausschnitt des Customer-Feedbacks (Reinhild Kokula/ OWASP Foundation/ CC BY-SA 4.0)

sieht man links die Tabelle der registrierten User und rechts die Tabelle „Customer Feedback“. Durch Klicken auf die Mülltonne kann man ungewolltes Feedback nun löschen und dadurch das Szenario lösen.

**Fazit:** Gängige Maßnahmen sind hier der Schutz des Administratorpassworts (Angriffsszenario 1), der Verzicht auf direkte Verweise zum Adminbereich im Menü (siehe Angriffsszenario 2) sowie das Vermeiden von gängigen URL-Endungen („.../#/administration“). Das Entwicklungsteam geht also davon aus, dass diejenigen, die als Administrator eingeloggt sind und die URL zum Administrationsbereich kennen, dies auch rechtmäßig tun. Allerdings ist die URL zu vorhersehbar, um tatsächlich Schutz zu gewährleisten. Eine einprägsame URL wie „.../#/

noitartsnimda“ (administration rückwärts) ist aber nicht zielführend, da dies wieder Security through obscurity darstellt (siehe Angriffsszenario 2). Bereits das Wählen einer komplizierteren Zieladresse, z.B. ein zufällig generierter String, kann das Finden des Administrationsbereichs erheblich erschweren. Zusätzlich könnte der Zugang durch ein weiteres, entsprechend gegen Angriffsszenario 1 abgesichertes Passwort eingeschränkt werden.

## Angriffsszenario 6: Manipulierte Bewertungen

**Beschreibung des Szenarios:** Klickt man auf den Menü-Button in der linken oberen Ecke und dann auf den Menüpunkt "Customer Feedback",

Abbildung 9: Feedback-Formular (Julian Scholz/ OWASP Foundation/ CC BY-SA 4.0)

so gelangt man auf die Feedback-Seite. Wenn man nicht eingeloggt ist, wird das Feedback automatisch unter dem Namen „anonymous“ abgeschickt. Ziel dieses Angriffsszenarios ist es, das Feedback unter dem Namen eines bereits registrierten Benutzers abzuschicken.

**Angriffsvektor:** Beim Drücken des Submit-Buttons wird ein POST-Request ausgeführt, der folgende Information als JSON-Daten überträgt. Dieser kann mittels des Netzwerkanalyse-Tabs über die Entwicklerwerkzeuge ausgelesen werden:

```
{
  "captchaID": 7
  "captcha": "5"
  "comment": "Das ist ein Test. "
  "rating": 5
}
```

Bei diesem POST-Request werden keine Benutzerinformationen mitgesendet. Nun betrachtet man die JSON-Daten der Antwort auf diesen POST-Request:

```
{
  "status": "success",
  "data": {
    "id": 7,
    "comment": "Das ist ein Test",
    "rating": 5,
    "updateAt": "2024-12-09T13:07:38.035Z",
    "createAt": "2024-12-09T13:07:38.035Z",
    "UserId": null
  }
}
```

In der Antwort befindet sich zusätzlich die UserID. Nun betrachtet man den Quelltext der Webseite mit den Entwicklerwerkzeugen genauer. Zur betreffenden Stelle im Quelltext, die untersucht werden soll, gelangt man über einen Rechtsklick auf das Formular und anschließend über den Menüpunkt Untersuchen des Kontextmenüs. Hier die betreffende Zeile:

```
<input id="userId" class="ng-untouched ng-pristine ng-valid" ngcontent-nnm-c23="" hidden="" type="text">
```

Durch das Schlüsselwort "hidden" wird dieses Eingabefeld dem Benutzer jedoch nicht angezeigt (siehe Angriffsszenario 2). Manipuliert man die Zeile und entfernt das Schlüsselwort "hidden", so erscheint das Eingabefeld im Feedbackformular.

Jetzt kann Feedback unter einer anderen Benutzeridentität versendet werden. Betrachtet man

Abbildung 10: Feedback-Formular mit ID-Feld (Julian Scholz/ OWASP Foundation/ CC BY-SA 4.0)

nochmals die Daten des POST-Requests, so wird tatsächlich die User-ID mit übergeben:

```
{
  "UserID": "1",
```

```

"captchaID": 7,
"captcha": "5",
"comment": "Das ist ein Test.",
"rating": 5
}
    
```

**Fazit:** Aus Sicherheitsgründen sollte vor der Veröffentlichung des Feedbacks geprüft werden, dass wirklich nur Daten des aktuell angemeldeten Benutzers manipuliert werden (Integrität). Da das bei der Entwicklung aber schnell einmal vergessen wird, können Schwachstellen entstehen.

## Angriffsszenario 7: Umgehen eines CAPTCHAs

**Beschreibung des Szenarios:** Webanwendungen sind täglich unerwünschten automatisierten Angriffen durch Bots ausgesetzt, beispielsweise dem automatisierten Ausfüllen von Formularen oder Erstellen von sogenannten „Fake-Konten“. Diese Angriffe können mit CAPTCHAs verhindert werden. CAPTCHA (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part - Ein vollständig automatisierter öffentlicher Turing-Test zur Unterscheidung zwischen Computern und Menschen) ist ein Verfahren, welches sicherstellen soll, dass eine Interaktion mit einem System von einem Menschen und nicht von einem automatisierten Bot durchgeführt wird. CAPTCHAs werden eingesetzt, um jegliche Art von automatisiertem Missbrauch, einschließlich Brute-Force-Angriffen, zu stoppen. Sie funktionieren, indem sie einen Test präsentieren, der für Menschen *leicht* zu bestehen, für Computer jedoch *schwer* zu bewältigen ist. Daher kann man mit einiger Si-

cherheit feststellen, ob ein Mensch eine Anfrage tätigt.

**Angriffsvektor:** Ziel des Angriffs soll es sein, innerhalb von 10 Sekunden zehn oder mehr Kundenfeedbacks möglichst automatisiert zu übersenden. Das Customer Feedback Formular (vgl. Abbildung 9) enthält ein CAPTCHA, um es vor Missbrauch zu schützen. In diesem Fall muss der Benutzer eine einfache Rechenaufgabe lösen, bevor er das Feedback abschicken kann. Das manuelle Abschicken eines Feedbacks inklusive Lösung des CAPTCHAs in weniger als zwei Sekunden stößt hierbei schon an seine Grenzen. Die Herausforderung dieses Szenarios ist daher, den Automatisierungsschutz zu überwinden.

Dies gelingt mit den folgenden Schritten:

1. Vorbereitung: Öffnen Sie über das Hauptmenü das Customer Feedback-Formular und die Entwicklerwerkzeuge.

```

{
  "captchaId": 8,
  "captcha": "8+5-5",
  "answer": "8"
}
    
```

2. Wechseln Sie in den Tab Netzwerk (oder Netzwerkanalyse).
3. Sie sollten eine GET-Anfrage an die Adresse <https://juice-shop.herokuapp.com/rest/captcha/> bemerken, die das CAPTCHA für das Feedback-Formular abrufen. Die HTTP-Antwort sieht ähnlich aus wie:

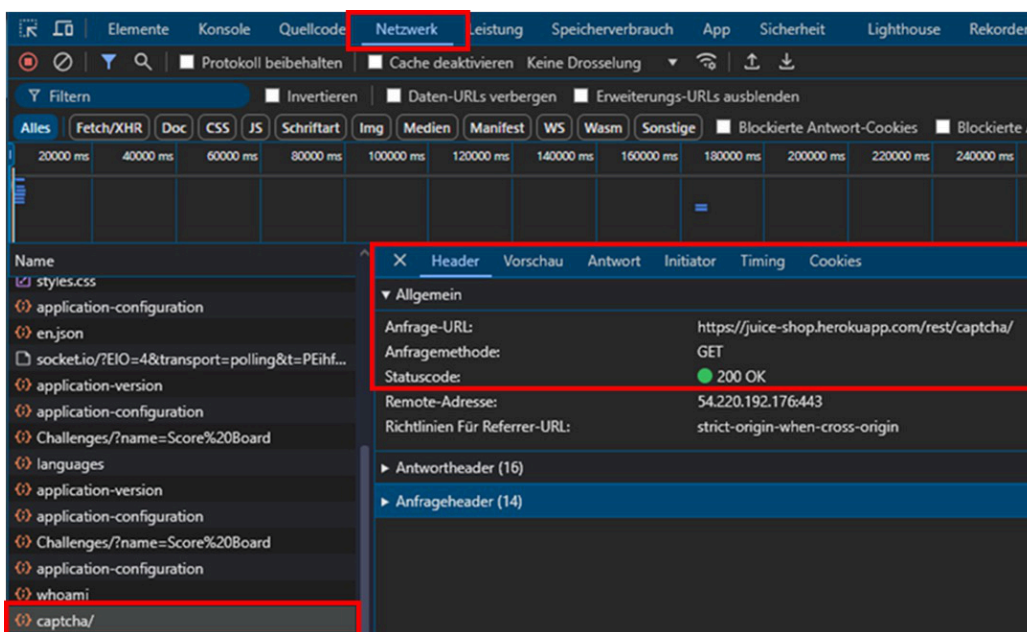


Abbildung 11: Entwicklerwerkzeuge im Browser (Benedikt Leesch/CC BY-SA 4.0)

```

{
  captchaId: 9,
  captcha: "10",
  comment: "Testkommentar
           (anonymous)",
  rating: 3
}

```

4. Füllen Sie das Formular aus und senden Sie es ab. Beobachten Sie währenddessen den Netzwerk-Tab. Die CAPTCHA-ID und die Lösung werden zusammen mit dem Feedback als POST-Anfrage (Feedbacks/) übermittelt:
5. Währenddessen wurde ein neues CAPTCHA mit neuer Aufgabe abgerufen.
6. Kopieren Sie nun die POST-Anfrage als Fetch:  
*Rechtsklick Feedbacks/ -> Kopieren -> Als fetch kopieren*
7. Wechseln Sie nun in den Tab Konsole. Dort können Sie den kopierten Fetch einfügen und mit Enter bestätigen. Der Server akzeptiert die *POST-Anfrage*, obwohl die CAPTCHA Lösung nicht mehr zu der aktuellen Rechnung passt. Wiederholen Sie dies mehrmals und die Aufgabe ist gelöst.
8. *Automatisierung*: Schreiben Sie ein kurzes Javascript, welches mit einer Wiederholung automatisiert die *POST-Anfrage* an den Server schickt. Führen Sie das Skript im

Tab *Konsole* aus. Sie können das folgende Beispiel als Vorlage verwenden.

*Tipp*: Fügen Sie das Skript in einem Editor zusammen.

**Fazit**: Dieses Beispiel zeigt: CAPTCHAs sind nicht unbesiegbar. Das Entwicklungsteam muss sicherstellen, dass CAPTCHAs nicht wiederverwendet werden können. Aber auch aktuellere CAPTCHAs, die auf Bildverarbeitung setzen, müssen aufgrund der rasanten Weiterentwicklung der Künstlichen Intelligenz in Zukunft neu gedacht werden.

## Angriffsszenario 8: Anfragen vor dem Senden abfangen

**Beschreibung des Szenarios**: In den Angriffsszenarien 4 und 6 wird über die Entwicklerwerkzeuge eine bereits gesendete Anfrage im Netzwerk-Tab ausgelesen und anschließend über die Konsole eine modifizierte Version erneut gesendet. Wollte man nun eine Bewertung des Onlineshops manipulieren, so müsste man zunächst mindestens einmal eine 1-Sterne-Bewertung abgeben, um den Aufbau und Inhalt der Anfrage auslesen zu können; selbst einen Stern hat dieser wortwörtliche „Saftladen“ nicht verdient! Das Ziel ist es, die Webseite dazu zu bringen, eine ungültige Bewertung zu akzeptieren, ohne zuvor eine gültige Anfrage gesendet zu haben. Die übliche Bewertungsskala (1 bis 5 Sterne) wird dabei durch eine manipulierte Anfrage umgangen. Gerade bei der Vorberei-

```

const sendFeedback = async () => {
  for (let i = 0; i < 10; i++) {
    try {
      const response = await fetch(...
        ...);
      if (response.ok) {
        console.log(`Anfrage ${i + 1}:Erfolgreich gesendet!`);
      } else {
        console.error(`Anfrage ${i + 1}:Fehler ${response.status}`);
      }
    } catch (error) {
      console.error(`Anfrage ${i + 1}: Fehler`, error);
    }
  }
};
sendFeedback();

```

Ersetzen Sie **fetch** durch die kopierte Post-Anfrage.

Das Skript wiederholt die Anfrage zehnmal.

**//Ersetzen**

Error-Handling: Fehler oder ungültige Antworten werden in der Konsole ausgegeben.

Quelltext 1: Automatisierung des CAPTCHA-Angriffs

tung von größeren Angriffen möchte man vor-

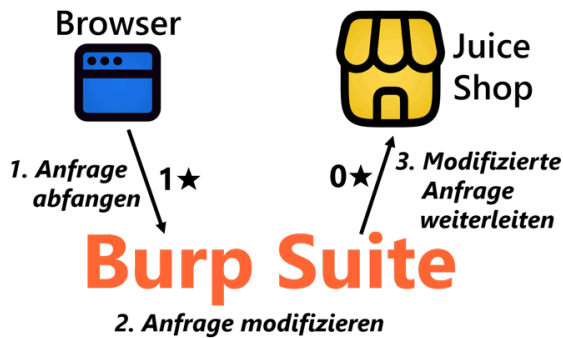


Abbildung 13: Improper Input Validation Angriff: Datenfluss der Anfrage (Sebastian Wich/ CC BY-SA 4.0)

her „unter dem Radar“ bleiben und das Abschicken von nicht notwendigen Anfragen in jedem Fall vermeiden. Im Folgenden werden die in Abbildung 13 gezeigten Schritte zur Ausführung eines Angriffs genauer erläutert.

**Angriffsvektor:** Mit dem Programm *Burp Suite*<sup>2</sup> wird der Datenverkehr zwischen dem Browser und der Webseite abgefangen. Der Angreifer navigiert zur Bewertungsfunktion, wählt eine Bewertung (z. B. 1 Stern) aus und schickt diese ab. Burp Suite fängt die Anfrage clientseitig ab, bevor sie den Server erreicht, sodass der Angreifer sie einsehen und bearbeiten kann.

Beispiel des relevanten Teils einer abgefangenen Anfrage:

```
{
  "captchaID":0,
  "captcha":"10",
  "comment":"Das ist ein törichter Saftladen!",
  "rating":1
}
```

In der abgefangenen Anfrage wird der Parameter „rating“ identifiziert, der die Anzahl der Sterne (hier 1) überträgt. Statt des ursprünglich gewählten Wertes wird ein ungültiger Wert, wie z. B. 0, eingetragen.

Die manipulierte Abfrage wird anschließend an den Server weitergeleitet. Dieser Angriff nutzt wiederum vor allem eine fehlende Validierung auf Serverseite aus. Wenn die Webseite nur auf Client-Seite im Browser überprüft, ob die Bewertung innerhalb der erlaubten Werte liegt, kann dies leicht umgangen werden. Wie bei den vorangegangenen Beispielen beschrieben, sollten Eingabedaten stets auf Serverseite validiert werden.

**Fazit:** Burp Suite ermöglicht neben dem Abfangen von Anfragen auch viele weitere Möglichkeiten für Sicherheitsfachleute, wie etwa automatisierte Schwachstellenanalyse, Manipulation von Daten, Durchführung von Angriffssimulationen wie Brute-Force- und Wörterbuchan-

griffe sowie die umfassende Sicherheitsprüfung von Webanwendungen.

## Fazit

Webanwendungen sind ein zentraler Bestandteil moderner IT-Infrastrukturen, jedoch auch ein häufiges Ziel für Angriffe. Strukturierte Sicherheitsmaßnahmen sind zur Abwehr erforderlich. Durch eine Kombination aus manuellen Tests und automatisierten Werkzeugen (Burp Suite) lassen sich Sicherheitslücken präzise identifizieren und beheben, was die Stabilität und Verlässlichkeit von Anwendungen entscheidend erhöht. Durch eine Simulation und Thematisierung der Angriffsszenarien im Informatikunterricht wird Sicherheit als integraler Bestandteil der Softwareentwicklung offenbart und damit ein Problembewusstsein für Cyber-Sicherheit geschaffen. Darauf aufbauend lässt sich eine Verhaltensänderung hin zu einem sicheren Umgang entwickeln. Dabei sollten neben den beschriebenen Angriffen auf die Technik auch der Faktor Mensch als Schwachstelle bewusstgemacht werden, der durch Social Engineering (Phishing etc.) Angriffsvektoren bietet, aber auch durch unbeabsichtigte Fehler eine Gefahr darstellt.

Der OWASP Juice Shop bietet neben den skizzierten Fallbeispielen eine große Anzahl weiterer lehrreicher Angriffsvektoren für gezielte Angriffe auf ein Softwaresystem. In diesem Sinne ist er ein interessanter und herausfordernder Spiel- und Übungsplatz zum Thema Informationssicherheit und – für diesen Zweck – überhaupt kein Saftladen!

## Quellen

Alle Webseiten/Links wurden zuletzt geprüft am 21.01.2025.

Brichzin, Peter u. a. (2024): Informatik 6 erhöhtes Niveau. Cornelsen, Berlin.

Bundesamt für Sicherheit in der Informationstechnik (2024): Die Lage der IT-Sicherheit in Deutschland [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2024.pdf?\\_\\_blob=publicationFile&v=5](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2024.pdf?__blob=publicationFile&v=5)

Bundesamt für Sicherheit in der Informationstechnik (2025): Awareness [https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Faktor-Mensch/Awareness/awareness\\_node.html](https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Faktor-Mensch/Awareness/awareness_node.html)

<sup>2</sup> <https://portswigger.net/burp>

Koch, A. (2024). Symmetrische Kryptologie und ihre Veranschaulichung. Informatische Bildung in Schulen 2(2). <https://doi.org/10.18420/ibis-02-02-07>

Ministerium für Bildung, Wissenschaft und Kultur des Landes Mecklenburg-Vorpommern, Institut für Qualitätsentwicklung (Institut für Qualitätsentwicklung) (2019). Rahmenplan Informatik für die Qualifikationsphase der gymnasialen Oberstufe. [https://www.bildung-mv.de/export/sites/bildungserver/downloads/unterricht/rahmenplaene\\_allgemeinbildende\\_schulen/Informatik/RP\\_INFO\\_SEK2.pdf](https://www.bildung-mv.de/export/sites/bildungserver/downloads/unterricht/rahmenplaene_allgemeinbildende_schulen/Informatik/RP_INFO_SEK2.pdf)

Qualitäts- und UnterstützungsAgentur - Landesinstitut für Schule (QUA-LiS) (2024). Kernlehrplan Informatik für die Gymnasiale Oberstufe. <https://www.schulentwicklung.nrw.de/lehrplaene/lehrplannavigator-s-ii/gymnasiale-oberstufe/informatik/informatik-klp/index.html>

Staatsministerium für Schulqualität und Bildungsforschung (ISB), München (2025). LehrplanPLUS Gymnasium. Fachlehrplan Informatik 12 (erhöhtes Anforderungsniveau). <https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/12/informatik/erhoeht>

## Lizenz



Dieser Artikel steht unter der Lizenz CC BY NC 4.0 zur Verfügung.

## Kontakt

Reinhild Kokula  
Gymnasium Münchberg

Han-Min Kufner  
Carl-Friedrich-Gauß-Gymnasium Schwandorf

Benedikt Leesch  
Deutschherren-Gymnasium Aichach

Klaus Reinold  
Studienseminar am Rupprecht-Gymnasium München  
[reinold@rupprecht-gymnasium.de](mailto:reinold@rupprecht-gymnasium.de)

Julian Scholz  
Gymnasium Tutzing

Sebastian Wich  
Gymnasium LSH Marquartstein  
[sebastian@wichematik.education](mailto:sebastian@wichematik.education)

Philipp Winter  
Carl-Spitzweg-Gymnasium Germering  
[philipp.winter@tum.de](mailto:philipp.winter@tum.de)