Web Pentest - Demo

Kunde:
Juice Shop GmbH
2025-08-29
v1.0

Kontakt:
Alexandre Labbe
+43 123 456789
alexandre@alabbe.fr

Inhaltsverzeichnis

Methodik und Umfang

Dieses Dokument ist ein Bericht über eine Sicherheitsüberprüfung der Webanwendung *Juicy Shop*. Diese wurde durchgeführt, um Sicherheitslücken 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** durchgeführt, ohne Zugangsdaten oder Vorwissen über die Anwendung, mit dem Ziel, unbekannte Schwachstellen zu identifizieren. Die Tests wurden nicht-invasiv durchgeführt, um so viele Fehlkonfigurationen und Sicherheitslücken wie möglich aufzudecken.

Die Bewertung war **zeitlich begrenzt**. Daher ist die folgende Liste der gefundenen Schwachstellen **nicht vollständig**. Es kann nicht garantiert werden, dass zukünftig keine weiteren Schwachstellen entdeckt werden.

Der Test umfasste die folgenden URLs:\

- **https://demo.owasp-juice.shop/**

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

Zusammenfassung

Die Suchfunktion im Shop wurde als anfällig für **SQL-Injection** identifiziert. Dies erlaubt Angreifern, Inhalte der Datenbank auszulesen, einschließlich sensibler Daten wie **Zugangsdaten**.

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.

Ein geheimer Endpunkt wurde in der Anwendung gefunden. Obwohl er nicht auf einer Seite verlinkt ist, ist er öffentlich zugänglich. Er enthält **sensible technische Informationen** über die Anwendung, die von Angreifern für Folgeangriffe genutzt werden können.

Die Werte der Rabattgutscheine konnten vom Tester decodiert werden. Dies ermöglicht Angreifern, **gültige Gutscheine mit hohen Rabatten zu erstellen**. Dadurch könnten sie Produkte im Shop zu stark reduzierten Preisen oder sogar **kostenlos** kaufen.

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.

Die betroffene Komponente implementiert einen **schwachen Captcha-Mechanismus**, um Bots daran zu hindern, Feedback einzureichen. Durch das Umgehen dieser Schutzmaßnahme könnten Angreifer die Formularübermittlung massiv automatisieren und gefälschte Feedbacks generieren.

Übersicht der Schwachstellen

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

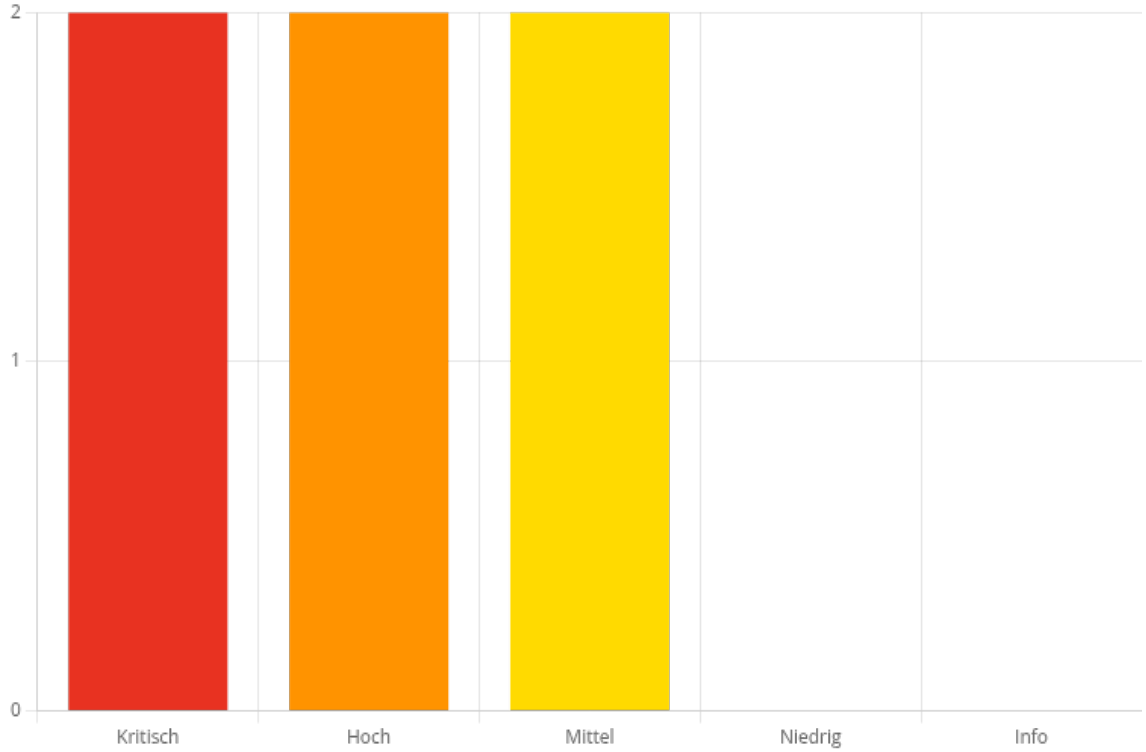

Abbildung 1 - Verteilung der identifizierten Schwachstellen

Schwachstelle	Kritikalität
SQL-Injection in der Suchfunktion	Critical
Fehlerhafte Authentifizierung	Critical
Sensible Daten öffentlich verfügbar	High
Fälschbare Rabattgutscheine	High
Fehlerhafte Zugriffskontrolle im Feedback-Bereich	Medium
Schwache Captcha-Mechanismen im Feedback-Bereich	Medium

1. SQL-Injection in der Suchfunktion

Status der Behebung:
Kritikalität: Critical
CVSS-Score: 9.1
Empfehlung: OWASP Juicy Shop **Recommendation:** Benutzereingaben sollten sorgfältig geprüft werden, bevor sie in Datenbankabfragen verwendet werden.

Überblick

Die Suchfunktion im Shop wurde als anfällig für **SQL-Injection** identifiziert. Dies erlaubt Angreifern, Inhalte der Datenbank auszulesen, einschließlich 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:

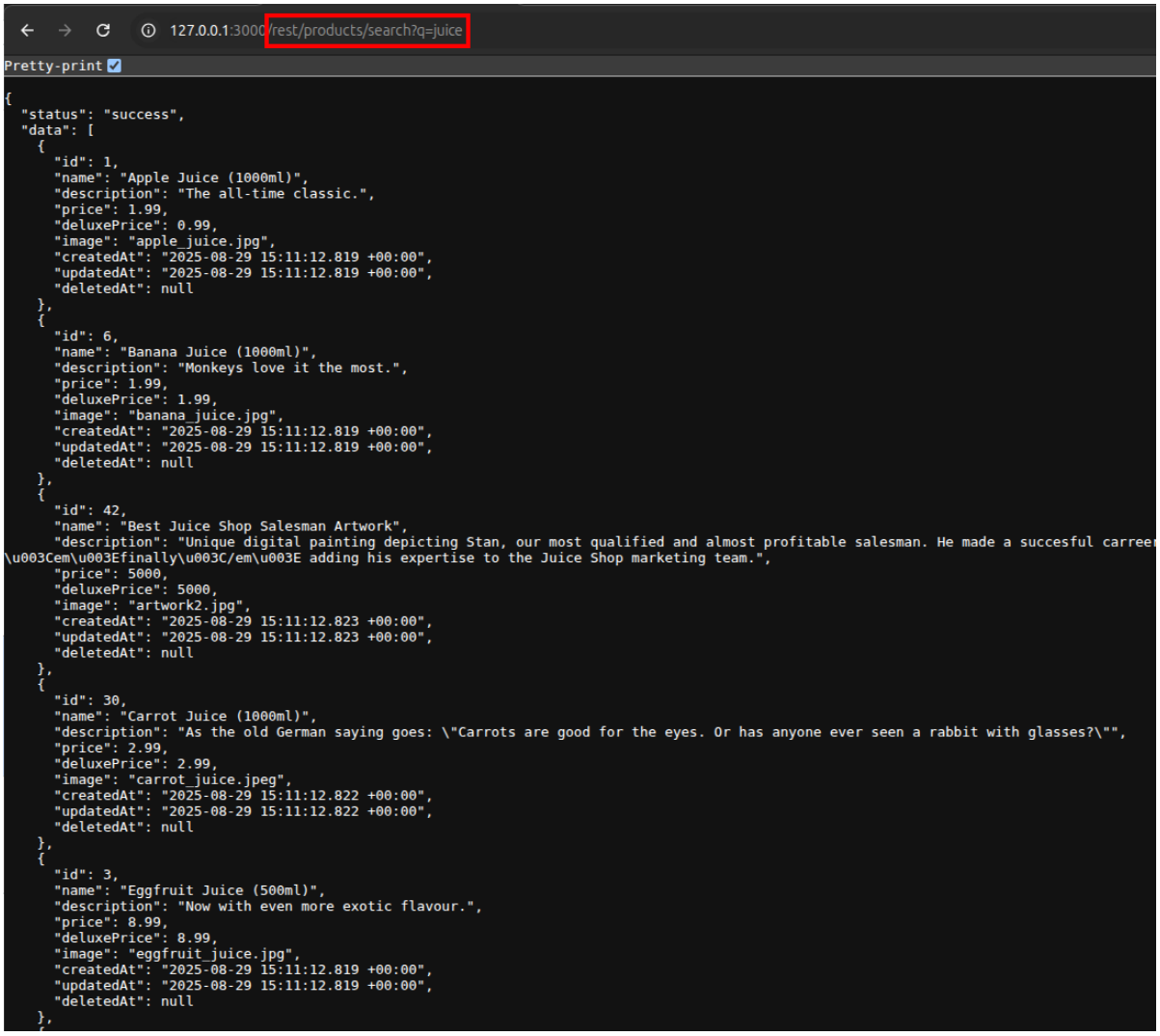

Abbildung 2 - HTTP-Antwort GET /rest/products/search?q=

Es wurde jedoch festgestellt, dass das Hinzufügen eines einfachen Anführungszeichens `'` einen Fehler in der Anwendung verursacht und eine SQLite-Fehlermeldung anzeigt.

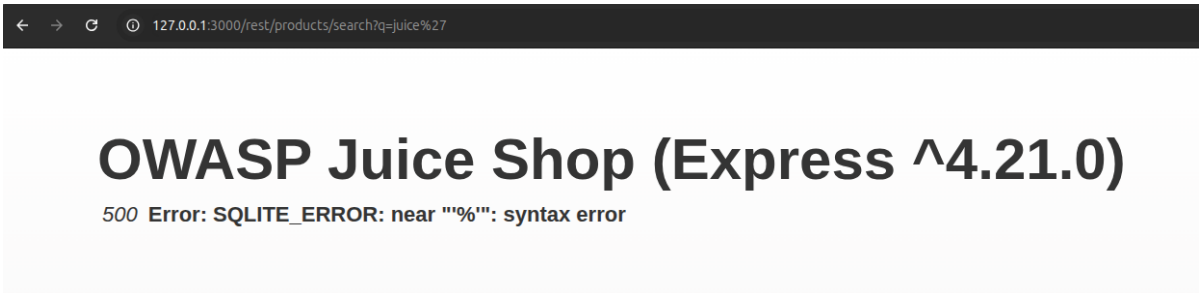

Abbildung 3 - SQLite-Fehler durch SQLi-Payload verursacht

Das einfache Anführungszeichen wird häufig 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 bestätigen und sie auszunutzen, um sensible Daten aus der Datenbank zu extrahieren.

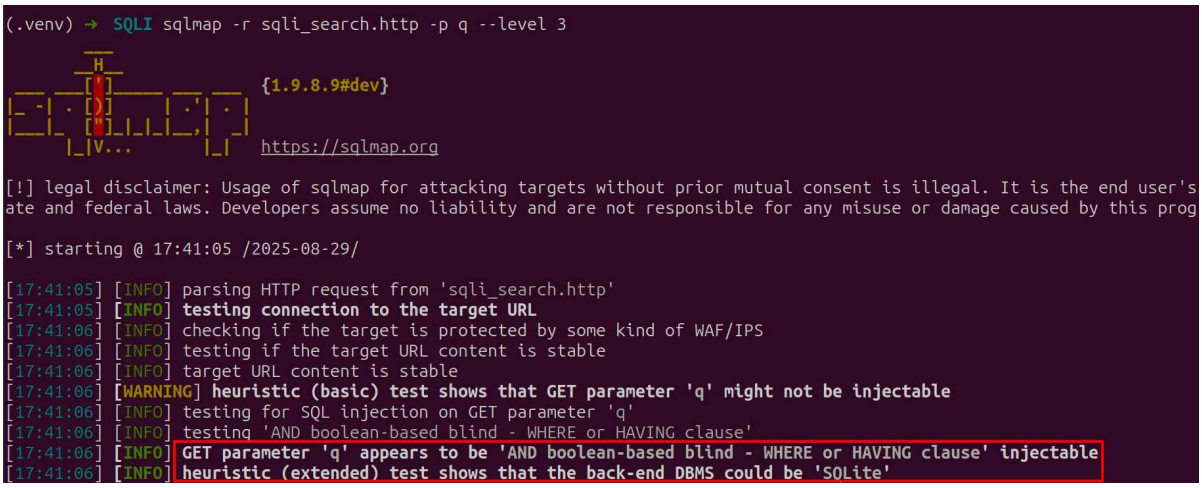

Abbildung 4 - Erkennung der SQLi mit sqlmap

Der Inhalt einer Tabelle namens `Users` konnte extrahiert werden, einschließlich MD5-Hashes, die offline geknackt werden können, um Klartext-Passwörter zu erhalten.

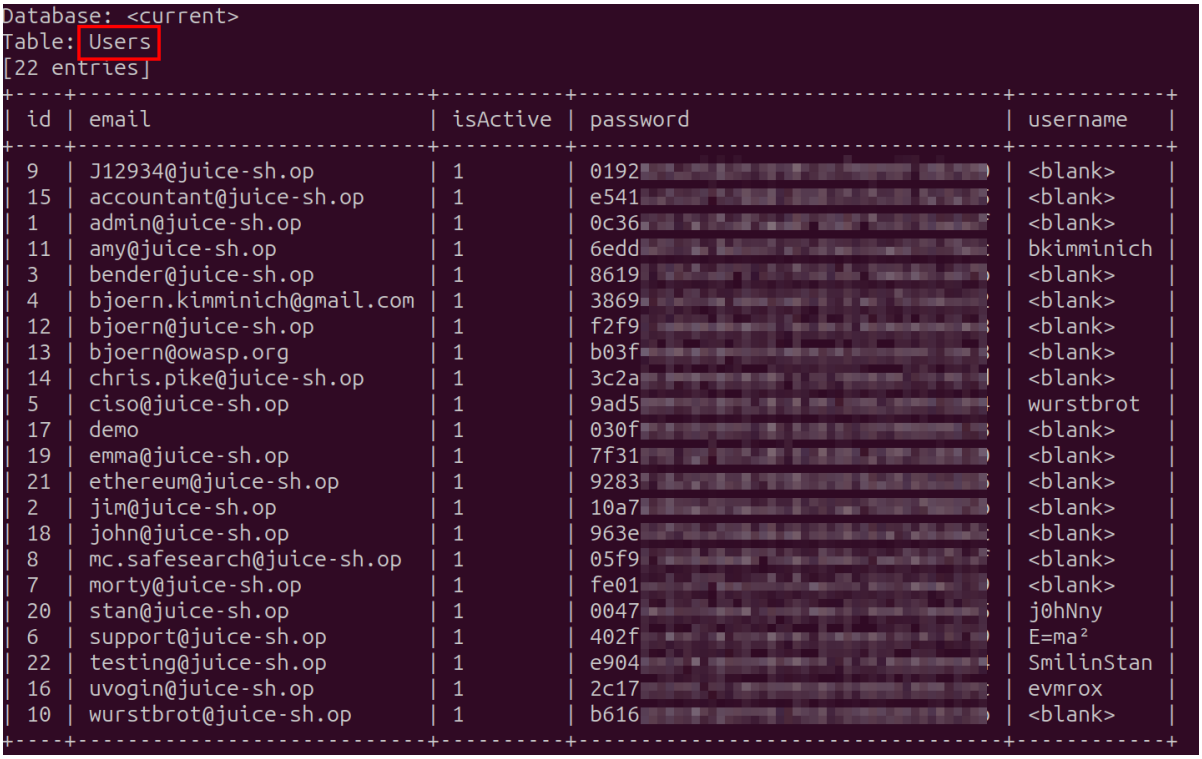

Abbildung 5 - Ausnutzung der SQLi mit sqlmap

Schließlich konnte auch der Quellcode der Anwendung extrahiert werden. Der folgende Screenshot zeigt, dass die Eingabedaten tatsächlich nicht bereinigt werden, bevor sie in die SQL-Abfrage eingefügt werden.

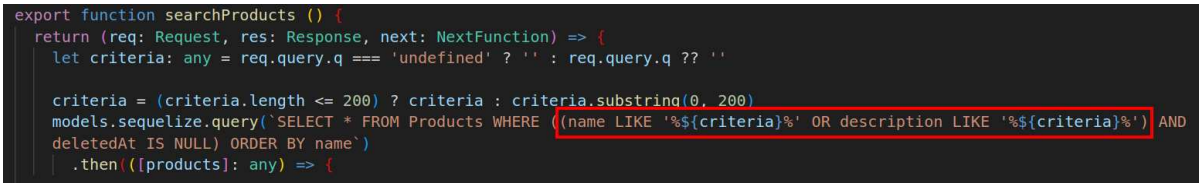

Abbildung 6 - Quellcode der SQLi-Schwachstelle

Empfehlung

- Benutzereingaben sollten niemals blind vertraut und müssen vor der Einbindung in SQL-Abfragen bereinigt werden.
- Es sollten **Prepared Statements** verwendet werden, um zu verhindern, dass die SQL-Syntax durch bösartige Payloads verändert wird.

Zusätzliche Informationen

- 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.

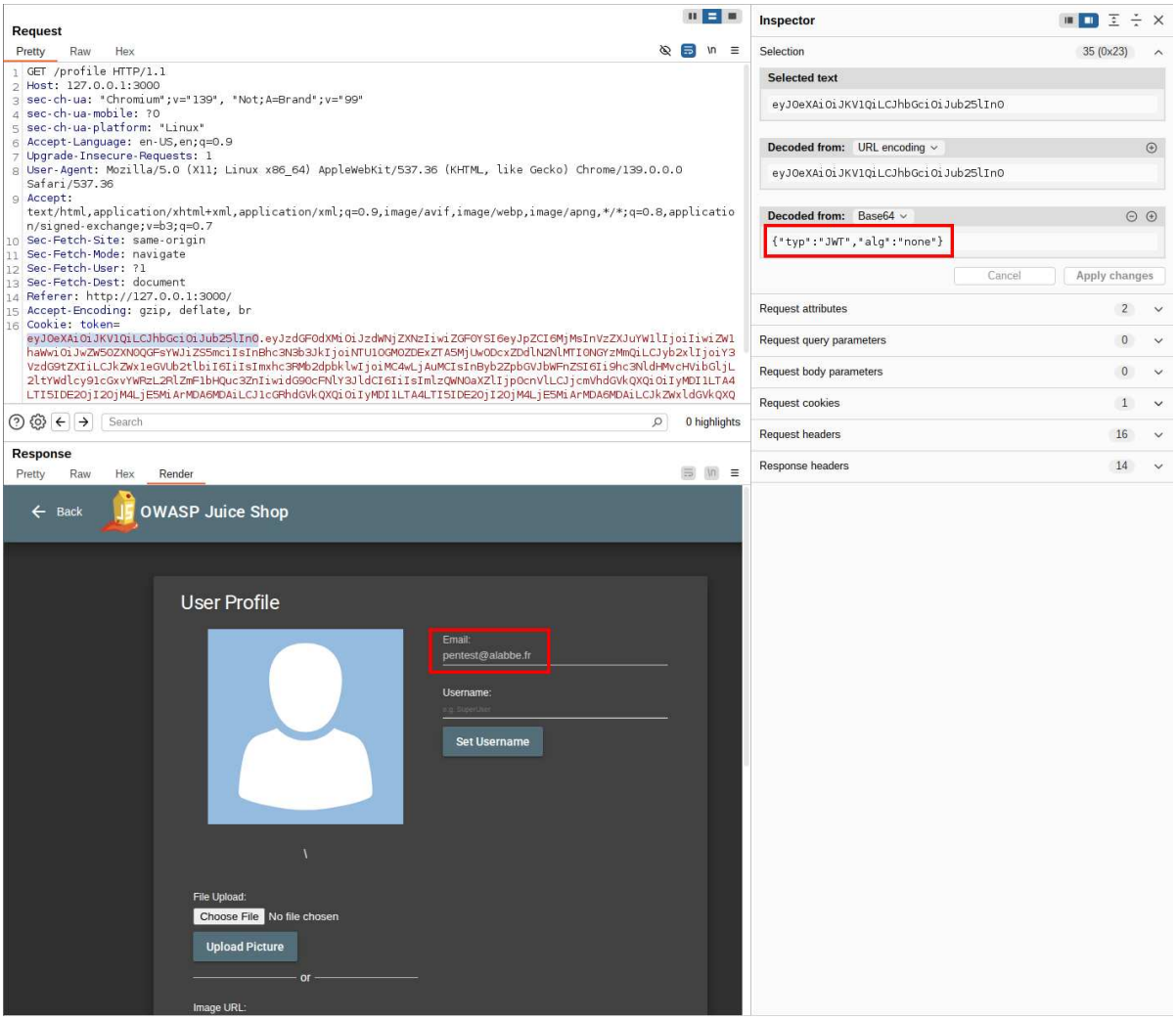

Abbildung 7 - JWT-Verschlüsselungsalgorithmus durch „none“ ersetzt

Anschließend konnte durch das Entfernen der Signatur der Datenanteil des Tokens von einem Angreifer geändert werden, einschließlich der **Session-ID**. Dies ermöglicht die **Imitation eines beliebigen Benutzers** in der Anwendung.

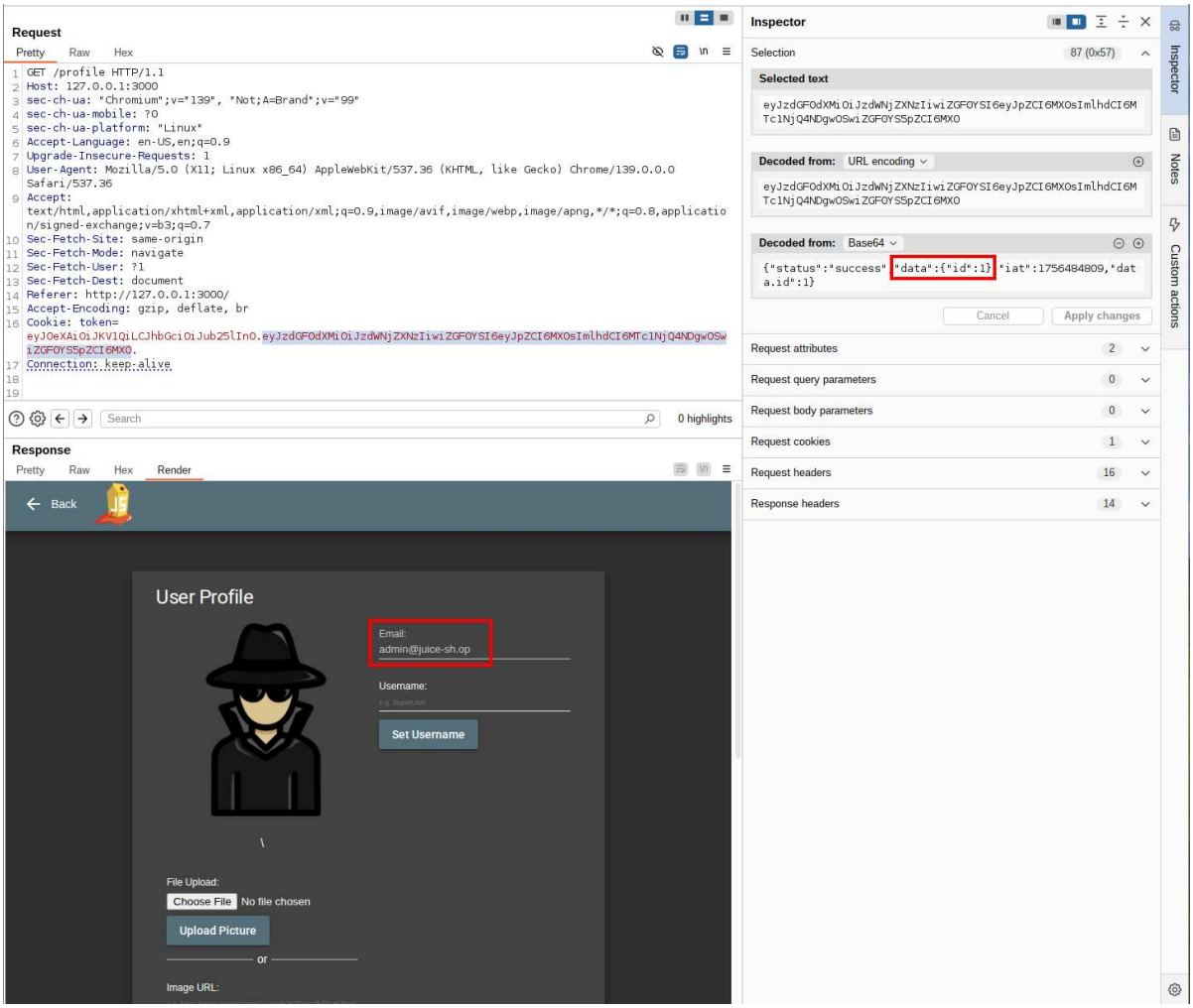

Abbildung 8 - Manipulation des Feldes „id“ in den JWT-Daten

Empfehlung

- Keine benutzerdefinierten JWT-Implementierungen verwenden
  - Stattdessen **Open-Source-Implementierungen mit bewährten Sicherheitsmechanismen** einsetzen
- Bei authentifizierten Endpunkten muss während der JWT-Überprüfung der Signaturalgorithmus auf eine **möglichst kleine Liste zulässiger Algorithmen** beschränkt werden

Zusätzliche Informationen

- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

3. Sensible Daten öffentlich verfügbar

Status der Behebung:
Kritikalität: High
CVSS-Score: 8.6
Empfehlung: OWASP Juicy Shop **Recommendation:** Zugriffskontrollen hinzufügen, um den Zugriff auf sensible und administrative Endpunkte zu beschränken.

Überblick

Ein geheimer Endpunkt wurde in der Anwendung gefunden. Obwohl er nicht auf einer Seite verlinkt ist, ist er öffentlich zugänglich. Er enthält **sensible technische Informationen** über die Anwendung, die von Angreifern für Folgeangriffe genutzt werden können.

Beschreibung

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

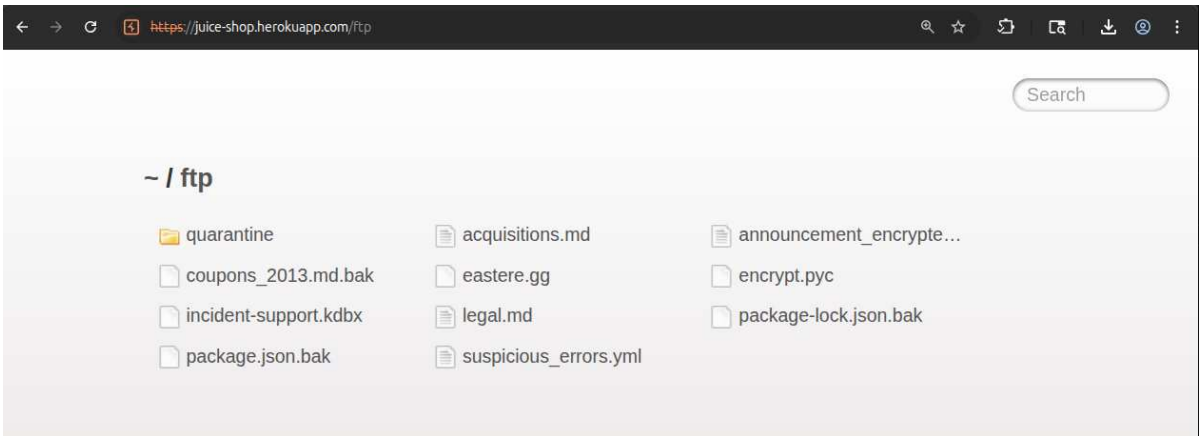

Abbildung 9 - Sensible Dateien öffentlich verfügbar

Solche Dateien enthalten Informationen, die Angreifer verwenden können, um ein besseres Verständnis der Funktionsweise der Anwendung zu erlangen und weitere Angriffe durchzuführen. Auf den ersten Blick können nur Dateien mit den Endungen `.pdf` und `.md` heruntergeladen werden. Dieser Schutzmechanismus kann jedoch umgangen werden, indem ein Null-Byte am Ende des Dateinamens hinzugefügt wird, gefolgt von einer gültigen Dateierweiterung.

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

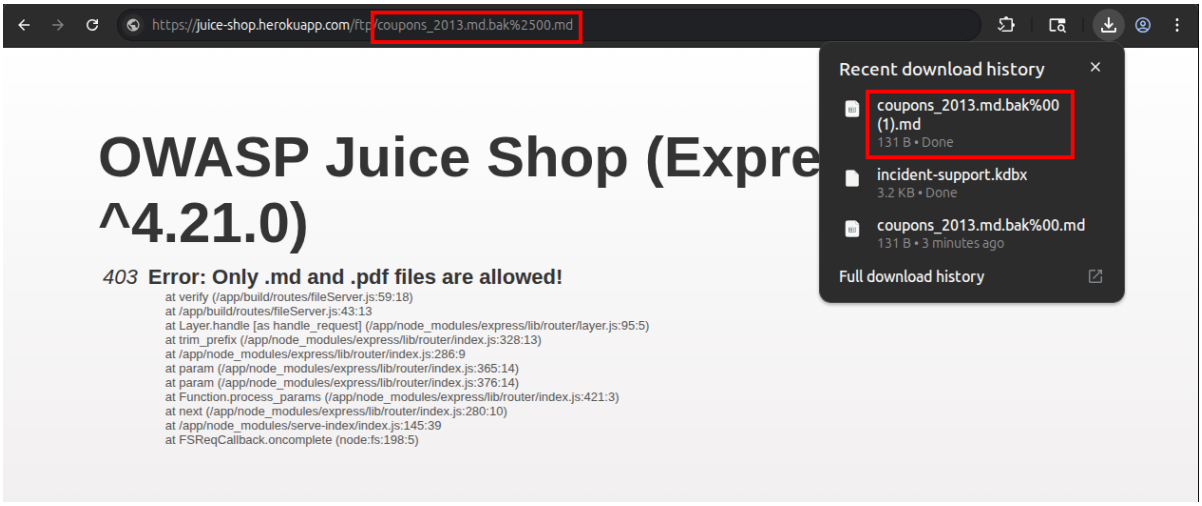

Abbildung 10 - Umgehung der Dateiendungskontrolle mittels Null-Byte

Empfehlung

- Zugriffskontrollen implementieren, um den Zugriff auf sensible Endpunkte zu beschränken.
- Null-Bytes neutralisieren, wenn URL-Parameter geparst werden.

Zusätzliche Informationen

- https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

4. Fälschbare Rabattgutscheine

Status der Behebung:
Kritikalität: High
CVSS-Score: 8.2
Empfehlung: OWASP Juicy Shop **Recommendation:** Generieren Sie Gutscheinwerte mit nicht vorhersagbaren Algorithmen.

Überblick

Die Werte der Rabattgutscheine konnten vom Tester decodiert werden. Dies ermöglicht Angreifern, **gültige Gutscheine mit hohen Rabatten zu erstellen**. Dadurch könnten 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 öffentlich verfügbar*) 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 erwähnten 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
l}6D$gC7ss
```

Dies ist kein Verschlüsselungsalgorithmus und erfordert daher keinen geheimen Schlüssel zur Decodierung. Das folgende Python-Skript wurde für 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.

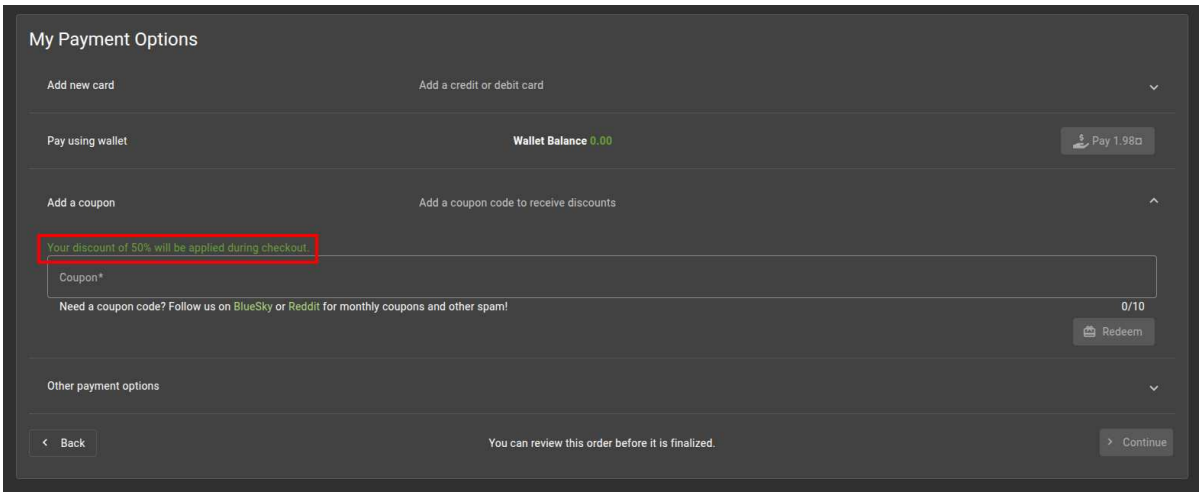

Abbildung 11 - Der gefälschte Gutschein wurde erfolgreich eingelöst

Empfehlung

- Die Werte der Gutscheine sollten einzigartig und nicht vorhersagbar sein, indem Zufälligkeit im Generierungsalgorithmus hinzugefügt wird.

Zusätzliche Informationen

- 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 {
      "UserId":1,
      "captchaId":7,
      "captcha":"10",
      "comment":"test (***in@juice-sh.op)",
      "rating":1
  }
7
```

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 {
      "status":"success",
      "data":{
          "id":14,
          "UserId":1,
          "comment":"test (***in@juice-sh.op)",
          "rating":1,
          "updatedAt":"2025-08-29T19:52:32.995Z",
          "createdAt":"2025-08-29T19:52:32.995Z"
      }
  }
```

Abbildung 12 - Imitation eines Benutzers beim Absenden eines Feedbacks

Empfehlung

- Zugriffskontrollen für die Feedback-Übermittlungsroute implementieren
- Das Feld `UserId` aus dem Anfrage-Body entfernen und die Identität direkt aus der Benutzersitzung ableiten

Zusätzliche Informationen

- https://owasp.org/Top10/A01_2021-Broken_Access_Control/

6. Schwache Captcha-Mechanismen im Feedback-Bereich

Status der Behebung:
Kritikalität: Medium
CVSS-Score: 5.3
Empfehlung: OWASP Juicy Shop **Recommendation:** Installieren oder implementieren Sie einen starken Captcha-Schutz.

Überblick

Die betroffene Komponente implementiert einen **schwachen Captcha-Mechanismus**, um Bots daran zu hindern, Feedback einzureichen. Durch das Umgehen dieser Schutzmaßnahme könnten Angreifer die Formularübermittlung massiv automatisieren und gefälschte Feedbacks generieren.

Beschreibung

Das Kunden-Feedback-Panel ist durch ein Captcha geschützt. Ziel dieser Maßnahme ist es, die Automatisierung der Formularübermittlung zu verhindern. Ohne diesen Mechanismus könnte ein bösartiger Bot beispielsweise die Funktion mit Spam überfluten, um legitime Feedbacks zu verdrängen.

Es wurde jedoch festgestellt, dass die implementierte Captcha-Lösung 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 für eine Maschine trivial lösbar und kann leicht aus dem HTML-Code ausgelesen werden.

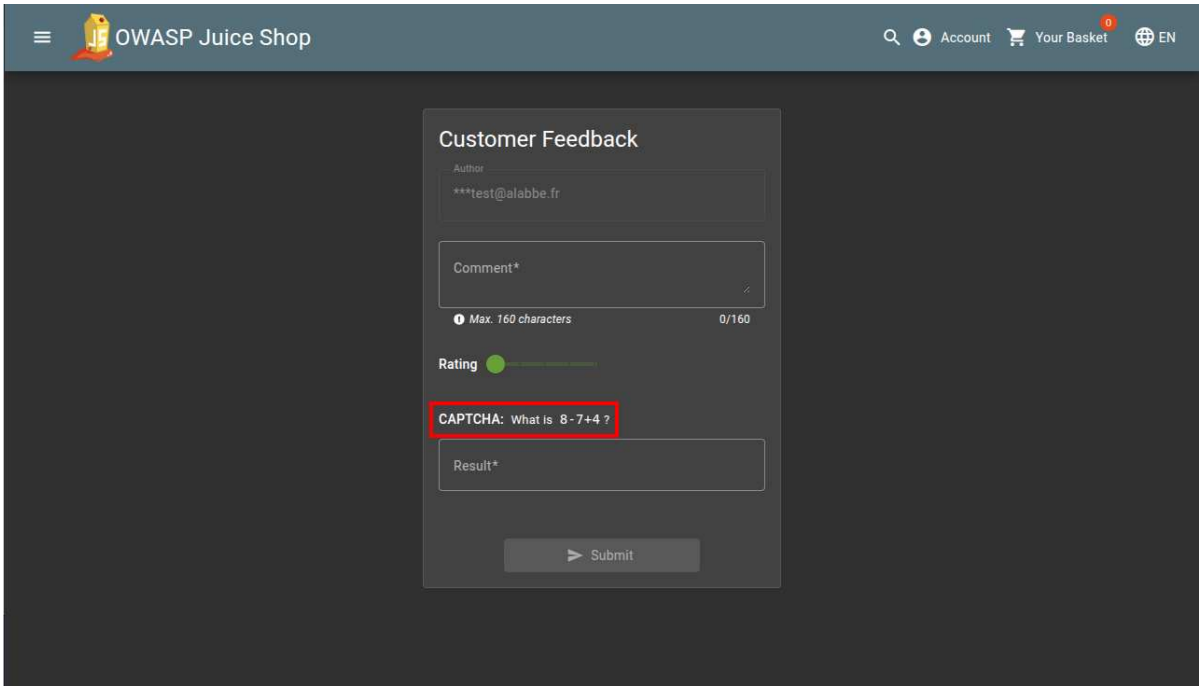

Abbildung 13 - Schwacher Captcha-Schutz mit einfacher mathematischer Aufgabe

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

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 können:

```
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 feddbacks have been posted by this bot.")

fuzz()
```

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

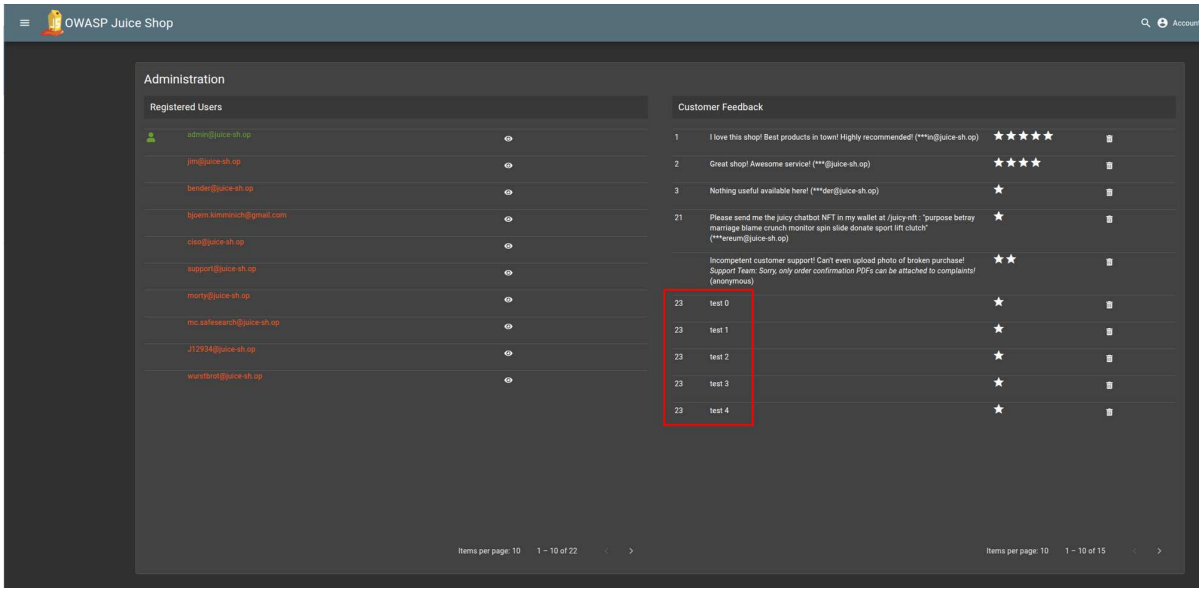

Abbildung 15 - Feedback-Spam im Administrator-Interface

Empfehlung

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

Zusätzliche Informationen

- https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-009_CAPTCHA_Defeat

Änderungsverlauf

Version	Datum	Beschreibung	Autor
1.0	2025-08-29	Endgültige Version	Alexandre Labbe