025-Sicherheitsaudit-Webanwendung

IT-Sicherheitsaufgabe
Anforderungsniveau: Mittel

1 Ziel der Aufgabe

Sie prüfen den OWASP Juice Shop – eine öffentlich verfügbare, absichtlich verwundbare Webanwendung – und schreiben darüber einen Auditbericht. Nicht die Menge der Funde zählt, sondern deren Belastbarkeit: vier bis fünf Befunde, jeder sauber belegt, selbst bewertet und im Betriebskontext eingeordnet, sind uns lieber als zwanzig abgeschriebene Werkzeugmeldungen.

Dass der Prüfgegenstand verwundbar ist, wissen Sie vorher, und Lösungswege stehen im Netz. Das ist Absicht: Die Aufgabe ist nicht das Finden, sondern das Berichten. Trennen Sie sauber zwischen Nachweis und Vermutung? Bilden Sie einen CVSS-Vektor selbst und begründen ihn? Erkennen Sie einen Fehlalarm Ihres Werkzeugs als solchen? Ordnen Sie Maßnahmen nach tatsächlichem Risiko statt nach Punktzahl? Das Ausnutzen ist hier Mittel, nicht Zweck.

Veranschlagte Bearbeitungszeit: 4–6 Stunden

2 Aufgabenbeschreibung

1. **Prüfgegenstand starten**

   Der Juice Shop ist eine Angular-Einzelseitenanwendung vor einem Express-Server auf Node.js mit SQLite als Datenbank. Ein Befehl genügt:

   ```
   docker run --rm -p 127.0.0.1:3000:3000 bkimminich/juice-shop
   # danach erreichbar unter http://localhost:3000/

   # Stand des Prüflings festhalten -- beides gehört in den Bericht:
   curl -s http://localhost:3000/rest/admin/application-version
   docker image inspect --format '{{index .RepoDigests 0}}' bkimminich/juice-shop
   ```

   Die Anwendung bringt unter `/#/score-board` einen Punktestand mit, der die eingebauten Übungsaufgaben auflistet. Sie dürfen ihn benutzen, er ersetzt aber keinen Befund: eine abgehakte Übung ist kein Nachweis, und ein Bericht, der nur diese Liste nacherzählt, ist keiner.

2. **Prüfrahmen festlegen**

   Halten Sie vor dem ersten Test fest, was Sie prüfen und was nicht. Die Abgrenzung gehört an den Anfang des Berichts: Geltungsbereich ausschließlich Ihre lokale Instanz, keine Tests gegen fremde Systeme; Prüftiefe Black Box über Oberfläche und HTTP-Schnittstelle (der Quelltext ist öffentlich – wenn Sie hineinsehen, schreiben Sie es dazu); ausgeschlossen sind Lasttests, Denial of Service und Angriffe auf die Docker-Laufzeit. Dazu Prüfzeitraum, Werkzeuge mit Version sowie Fassung und Bilddigest des Prüfgegenstands aus Schritt 1.

   Für die Priorisierung gilt folgende Annahme, die Sie im Bericht übernehmen: Der Shop läuft öffentlich erreichbar mit rund 2 000 Kundenkonten, die Zahlungsabwicklung liegt bei einem externen Dienstleister, und die Maßnahmen setzt ein Betriebsteam von zwei Personen um.

3. **Untersuchen**

Lassen Sie zuerst ein automatisches Werkzeug laufen und arbeiten Sie danach von Hand weiter. Der ZAP-Baseline-Durchlauf braucht ein gemeinsames Docker-Netz, um den Prüfling zu erreichen:

```
docker network create audit
docker run --rm -d --name juiceshop --network audit \
       -p 127.0.0.1:3000:3000 bkimminich/juice-shop

mkdir -p werkzeuge
docker run --rm --network audit -v "$PWD/werkzeuge:/zap/wrk:rw" \
       -t ghcr.io/zaproxy/zaproxy:stable \
       zap-baseline.py -t http://juiceshop:3000 -J zap.json -I

# Handarbeit, zum Beispiel:
curl -sI http://localhost:3000/ | sort
curl -s 'http://localhost:3000/rest/products/search?q=apple' | head -c 400
```

Sichten Sie mindestens die folgenden Bereiche und halten Sie zu jedem fest, was Sie geprüft haben – *auch ohne Befund*:

- **Zugriffskontrolle:** Erreicht ein angemeldetes Konto die Daten eines anderen, wenn die Kennung in Adresse oder Rumpf ersetzt wird? Sind Verwaltungsfunktionen ohne die passende Rolle erreichbar? Prüfen Sie die Endpunkte direkt – eine ausgeblendete Schaltfläche ist keine Zugriffskontrolle.
- **Einschleusung:** Suchparameter, Filter, Anmeldefelder.
- **Authentifizierung und Sitzung:** Kennwortrichtlinie, Begrenzung der Fehlversuche, Auskunft über vorhandene Konten, Prüfung und Gültigkeitsdauer des Tokens, Cookie-Attribute.
- **Preisgabe:** Fehlerseiten mit Stapelüberwachung oder Datenbankmeldung, Felder in Antworten, die der Aufrufer nicht sehen sollte, offen erreichbare Verzeichnisse und Dateien.
- **Konfiguration und Kopfzeilen:** etwa `Content-Security-Policy`, `Referrer-Policy`, `frame-ancestors`. Eine fehlende Kopfzeile kann ein Befund sein; eine vorhandene, die nichts bewirkt, ebenfalls.

Konten legen Sie sich selbst an. Nutzen Sie dafür zwei erfundene Adressen unter `example.invalid` und das Kennwort `hunter2`.

4. **Befunde belegen und bewerten**

Wählen Sie vier bis fünf Befunde aus und arbeiten Sie diese vollständig aus. Jeder bekommt denselben Aufbau:

```
### F-02  Fremde Warenkörbe über die Kennung in der Adresse lesbar

Schweregrad:  Mittel (CVSS 3.1: 6.5 / AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N)
Kategorie:    A01:2021 Broken Access Control
CWE:          CWE-639 Authorization Bypass Through User-Controlled Key
Betroffen:    GET /rest/basket/{id}
Geprüft am:   2026-09-14, Werkzeug: curl 8.7.1
Reproduktion: 1. Anmelden als a@example.invalid, eigene Korbkennung notieren
              2. Dieselbe Anfrage mit der Korbkennung von b@example.invalid
              3. Antwort 200 mit den Positionen des fremden Korbs
Beleg:        belege/F-02-anfrage.txt, belege/F-02-antwort.txt
Auswirkung:   Jedes angemeldete Konto liest die Körbe aller Kunden.
Vektor,
begründet:    PR:L, weil ein gewöhnliches Konto genügt. S:U, weil der
```

```
              Zugriff die Anwendung nicht verlässt. C:H wegen vollständiger
              Kaufhistorien, I:N, da nur gelesen wird.
Maßnahme:     Serverseitig prüfen, dass die Kennung zur Sitzung gehört.
Aufwand:      klein
```

Der CVSS-Vektor wird selbst gebildet, nicht aus einer Werkzeugmeldung übernommen; maßgeblich ist die Spezifikation unter `first.org/cvss/v3.1/specification-document`. Die Begründung ist Pflicht und nennt für jede strittige Metrik den Grund. Bei mindestens zwei Befunden erwarten wir, dass Sie eine Metrik ausdrücklich gegen eine naheliegende Alternative abwägen – etwa `PR:N` gegen `PR:L` oder `S:U` gegen `S:C`.

Legen Sie die Belege als Textdateien oder Bildschirmfotos unter `belege/` ab und verweisen Sie aus dem Bericht darauf. Ohne Beleg kein Befund.

5. **Fehlalarme und Vermutungen trennen**

   Jeder übernommene Werkzeugbefund wird nachgeprüft und als bestätigt oder als Fehlalarm gekennzeichnet. Widerlegen Sie **mindestens einen** Fehlalarm aus dem ZAP-Durchlauf: Nennen Sie die Meldung, zeigen Sie mit einem Beleg, dass die beschriebene Auswirkung nicht eintritt, und erklären Sie in zwei Sätzen, warum das Werkzeug sie trotzdem meldet.

   Alles, was Sie für wahrscheinlich, aber nicht für nachgewiesen halten, gehört in einen eigenen Abschnitt „Beobachtungen ohne Nachweis“ – mit dem, was Ihnen zum Nachweis fehlt. Diese Punkte zählen nicht zu den vier bis fünf Befunden, wir lesen sie aber genauso aufmerksam.

6. **Priorisieren und berichten**

   Bringen Sie die Befunde in die Reihenfolge, in der das Betriebsteam aus Schritt 2 sie abarbeiten sollte. Diese Reihenfolge folgt nicht allein dem CVSS-Wert, sondern dem Risiko im beschriebenen Kontext sowie Aufwand und Wirkung der Maßnahme. Wo Ihre Reihenfolge von der Punktzahl abweicht, begründen Sie das in einem Satz – solche Abweichungen sind erwünscht, nicht verdächtig.

   Der Bericht enthält eine Zusammenfassung für die Leitung von höchstens einer halben Seite ohne Fachjargon, den Prüfrahmen aus Schritt 2, eine Befundübersicht als Tabelle (Kennung, Titel, CVSS, Kategorie, Rang), die Einzelbefunde im Aufbau aus Schritt 4, die Abschnitte „Geprüft, nichts gefunden“, „Fehlalarme“ und „Beobachtungen ohne Nachweis“ sowie einen Maßnahmenplan mit Aufwandsschätzung und einem Vorschlag für den Nachtest.

3 Anforderungen

- Vier bis fünf ausgearbeitete Befunde aus mindestens drei verschiedenen Kategorien der OWASP Top 10 (2021). Mehr Befunde verbessern die Bewertung nicht.
- Jeder Befund ist ohne Rückfrage nachstellbar und trägt einen begründeten CVSS-v3.1-Vektor, eine CWE-Kennung und die Zuordnung zur OWASP-Kategorie.
- Mindestens ein widerlegter Fehlalarm aus dem automatischen Durchlauf, mit Beleg. Die Rohausgabe liegt bei; ein unkommentierter ZAP-Bericht als Anhang genügt nicht.
- Alle Bereiche aus Schritt 3 sind im Bericht erwähnt – auch die ohne Befund. Nachweis und Vermutung stehen in getrennten Abschnitten; Formulierungen wie „vermutlich anfällig“ kommen im Befundteil nicht vor.
- Keine echten Zugangsdaten, Schlüssel oder personenbezogenen Daten in Bericht, Belegen oder Historie – nur erfundene Werte wie `example.invalid`, `hunter2` oder `fk_test_XXXXXXXX`. Zugangsdaten, die Sie im Prüfgegenstand vorfinden, kürzen Sie im Bericht auf die ersten zwei Zeichen.

- Der Bericht bleibt unter zehn Seiten.

4 Abgabe

- Ein Git-Repository mit nachvollziehbarer Commit-Historie und dem Auditbericht als `bericht.md` oder PDF.
- Verzeichnis `belege/` mit den Nachweisen, benannt nach der Befundkennung.
- Verzeichnis `werkzeuge/` mit der Rohausgabe des ZAP-Durchlaufs und Ihren `curl`-Aufrufen beziehungsweise Skripten, sodass sich die Prüfungen ohne Abtippen wiederholen lassen.
- Eine `README.md` mit Startanleitung, geprüfter Fassung samt Bilddigest, Werkzeugliste mit Versionen, aufgewendeter Zeit je Prüfabschnitt und einer Notiz, was Sie mit mehr Zeit zusätzlich geprüft hätten.

5 Bewertungskriterien

- **Belegqualität (35 %):** Lässt sich jeder Befund ohne Rückfrage nachstellen? Zeigt der Beleg die Auswirkung oder nur einen Statuscode? Sind Nachweis und Vermutung sauber getrennt?
- **Bewertung (25 %):** Passen Vektor, CWE und Schweregrad zum Sachverhalt? Trägt die Begründung der Metriken, oder ist der Vektor geraten?
- **Umgang mit Werkzeugen (20 %):** Ist der Fehlalarm wirklich einer, und ist die Widerlegung belegt? Wurde über den automatischen Durchlauf hinaus geprüft?
- **Bericht und Priorisierung (20 %):** Aufbau, Verständlichkeit für Technik und Leitung, Sprache. Ist die Maßnahmenreihenfolge im gegebenen Kontext begründet und umsetzbar?

Positiv fällt auf, wer einen Befund gegen die eigene Punktzahl priorisiert und das überzeugend begründet, und wer eine Schwachstelle findet, die der automatische Durchlauf nicht meldet.