Ergebnisbericht

Juice-Shop

Saftiger Vertrieb GmbH

Adam Apfel
Obststraße 44
Safthausen

Deining, 3. Oktober 2025

Projektnummer: 0001

Berichtsversion 1.0

VERTRAULICH

Schutzpunkt GmbH
Schloßstraße 7 A
92364 Deining

Kontakt:
Max Bäumler
Mobil: +49 9184 8081 966
E-Mail: *max.baeumler@schutzpunkt.com*
Web: *www.schutzpunkt.com*

Geschäftsführer: Max Bäumler
Registergericht Nürnberg: HR B 44700
USt-IdNr.: DE455642304

Inhaltsverzeichnis

1 Dokumentenkontrolle

1.1 Team

Kontakt	Details	Rolle
Max Bäumler	Mobil: +49 9184 8081 966 E-Mail: max.baeumler@schutzpunkt.com	Lead Pentester

1.2 Änderungshistorie

Version	Beschreibung	Datum
0.1	Initiale Version	03.10.2025
1.0	Finalisierung des Berichts	03.10.2025

2 Management Summary

Das Ziel dieses Penetrationstests war es, die Sicherheit der Webanwendung **Juice-Shop** zu bewerten, Schwachstellen zu identifizieren und potenzielle Risiken für die kritischen Vermögenswerte des Unternehmens zu bewerten. Alle Aktivitäten wurden zwischen **Dienstag, 30. September 2025** und **Donnerstag, 2. Oktober 2025** durchgeführt. Insgesamt wurden **3 Personentage** für diesen Test aufgewendet. Diese Bewertung ist Teil umfassenderer Bemühungen, die Sicherheit und Widerstandsfähigkeit der Systeme und Daten der Organisation dauerhaft zu gewährleisten.

Wichtigste Ergebnisse

- **Gesamtrisiko**: Basierend auf den Testergebnissen kann **das Gesamtrisiko im Vergleich zu ähnlichen Tests für andere Kunden als durchschnittlich eingestuft werden**. Es wurden insgesamt **5 offene Schwachstellen** mit unterschiedlicher Kritikalität identifiziert.
- **Riskante Schwachstellen**: Die Schwachstellen **H1, M2,** führen möglicherweise zu: Unmittelbaren finanziellen Verlusten durch verwundbare Gutscheincodes. Ferner unbefugtem Zugriff, Datenschutzverletzungen, und Betriebsstörungen durch unberechtigten Zugriff auf eine Administrationsschnittstelle.
- **Weniger riskante Schwachstellen**: Unserer Einschätzung nach sind die restlichen Schwachstellen als weniger riskant zu bewerten, jedoch nicht zu vernachlässigen. Sie können in bestimmten Szenarien ausgenutzt werden, aber deren Ausnutzung verursacht weniger wahrscheinlich unmittelbaren Schaden.

Empfehlungen

1. **Sofortmaßnahmen**: Die Schwachstellen **H1, M2,** sollten priorisiert und so schnell wie möglich behoben werden, da diese nach Einschätzung der Tester das größte Risiko darstellen.
2. **Quick Wins**: Die Schwachstellen **H1, M1, M3,** können vermutlich mit geringem Aufwand behoben werden.
3. **Kontinuierliche Verbesserung der Sicherheit**: Regelmäßige Fehlerbehebungen, Tests und Sicherheitsüberprüfungen sollten Teil eines Plans zur kontinuierlichen Verbesserung der IT-Sicherheit sein. Es wird dringend empfohlen, wichtige Systeme nach der Behebung von Schwachstellen erneut zu testen, um deren Wirksamkeit sicherzustellen.

Einschränkungen und Umfang des Tests

Der Penetrationstest wurde innerhalb eines definierten Testumfangs durchgeführt und konzentrierte sich auf die Webanwendung Juice-Shop. Die Ergebnisse gelten nur für den Zeitraum, in dem der Test durchgeführt wurde, da sich die Sicherheitslage im Laufe der Zeit ändern kann. Es ist wichtig zu beachten, dass möglicherweise nicht alle Schwachstellen entdeckt wurden, da der Test durch den Umfang und die knappe zur Verfügung stehende Zeit begrenzt war.

Fazit

Insgesamt ist das Sicherheitsniveau zwar im Allgemeinen normal, jedoch erfordern die identifizierten Schwachstellen, insbesondere die riskanten Schwachstellen, sofortige Aufmerksamkeit, um eine mögliche Ausnutzung zu vermeiden. Regelmäßige Penetrationstests sowie die umgehende Behebung der festgestellten Mängel sind für die Aufrechterhaltung einer sicheren Umgebung unerlässlich.

Durch die Beseitigung der identifizierten Risiken kann das Unternehmen seine Abwehr gegen Cyber-Bedrohungen erheblich verbessern und die Sicherheit seiner digitalen Ressourcen erhöhen.

2.1 Schwachstellenübersicht

Im Rahmen dieses Tests wurden offene Schwachstellen mit folgender Kritikalität identifiziert: **1 Hoch**, **3 Mittel** und **1 Info** .

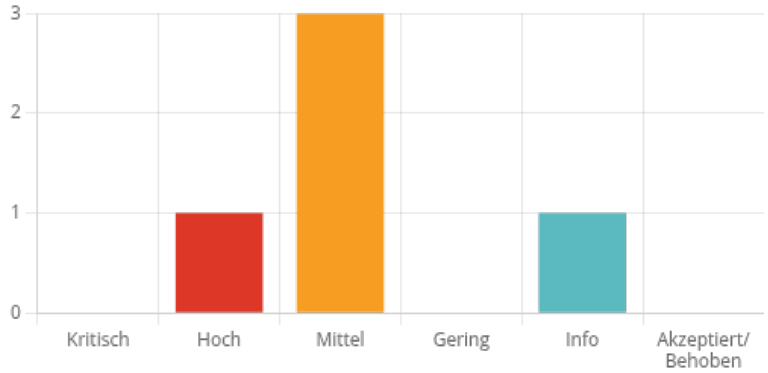

Abbildung 1 - Verteilung der identifizierten Schwachstellen

2.2 Identifizierte Schwachstellen

Die folgende Tabelle bietet einen Überblick über die identifizierten Schwachstellen.

#	CVSS	Beschreibung	IA	QW	Status	Seite
H1	8.6	Login Bypass durch Fehlerbasierte SQL-Injection (SQLi)	‼️	✳️	Offen	8
M1	6.1	Cross-Site Scripting (XSS) – Reflektiert		✳️	Offen	13
M2	5.3	Fehler in der Rabattgutschein-Logik	‼️		Offen	17
M3	5.3	Verzeichnisauflistung aktiviert		✳️	Offen	22
I1	0.0	Informationspreisgabe über Stack-Traces			Offen	25

‼️ IA = Sofortmaßnahme (Immediate action). ✳️ QW = Quick Win.

3 Allgemeine Rahmenbedingungen

In diesem Abschnitt werden die allgemeinen Rahmenbedingungen des gesamten Auftrags dokumentiert.

3.1 Zielsetzung

Das Ziel dieses Penetrationstests ist es, Sicherheitslücken, Fehlkonfigurationen und potenzielle Angriffswege zu identifizieren, welche die Vertraulichkeit, Integrität und Verfügbarkeit der Web-Anwendung **Juice-Shop** bedrohen, sowie gleichzeitig die Einhaltung relevanter Sicherheitsrichtlinien und Best Practices sicherzustellen.

Ziel ist es, so viele Schwachstellen wie möglich in gegebener Zeit abzudecken, während sich an die im Kick-Off Termin vereinbarten Regeln gehalten wird, um eine minimale Beeinträchtigung des Geschäftsbetriebs zu gewährleisten.

3.2 Testzeitraum

Alle Aktivitäten wurden zwischen **Dienstag, 30. September 2025** und **Donnerstag, 2. Oktober 2025**. Insgesamt **3 Personentage** wurden für diesen Test aufgewendet.

3.3 Prüfumfang (Scope)

Der Projektumfang definiert genau, was Teil des Tests ist.

Im Umfang

Die folgenden Anwendungen sind Teil des Prüfumfangs.

System	Beschreibung
https://juice-shop.lab	OWASP Juice-Shop (Testsystem)

Außerhalb des Umfangs

Es sind keine Teilbereiche der Anwendungen ausdrücklich vom Prüfumfang ausgeschlossen.

3.4 Benutzerkonten und Berechtigungen

Benutzerkonten

Vom Kunden wurden keine Benutzer bereitgestellt. Es gab allerdings die Funktion zur Selbstregistrierung. Folgende Accounts wurden selbst registriert und verwendet.

Benutzer	Rolle	Beschreibung
max.baeumler@schutzpunkt.com	Regulärer Benutzer	Selbst registriert

Gutscheincodes

Ferner wurden folgende Gutscheincodes vom Kunden bereitgestellt.

Code	Rabatt	Gültigkeit
q:<Irh7ZKp	10%	September 2025
pEw8ph7ZKu	10%	Oktober 2025
pEw8ph7ZKu	15%	Oktober 2025
pes[Ch7ZKp	10%	November 2025

3.5 IP-Adressen

Von diesen Systemen aus wurden die Angriffe durchgeführt.

System	Beschreibung
80.151.38.120	IP-Adresse unseres Büros in Deining

3.6 Testgrundlage und -ansatz

Der Test wurde unter Zuhilfenahme des OWASP WSTG Frameworks durchgeführt.

Folgender Testansatz wurde gewählt:

- **Informationsgrundlage:** Grey-Box
- **Aggressivität:** Deliberative
- **Abdeckung:** Limited
- **Vorgehensweise:** Obviously
- **Zugang:** Network Connection
- **Herkunft:** External

3.7 Einschränkungen

Während des Tests traten keine Einschränkungen auf.

4 Findings

8.6 H1: Login Bypass durch Fehlerbasierte SQL-Injection (SQLi)

Vektor	*CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:L*
Status	Offen
Sofortmaßnahme empfohlen	Ja ‼
Quick Win zur Behebung	Ja ✳
Tags	WSTG-INPV-05, CWE-89, ATT&CK-T1190
Betroffene Komponenten	https://juice-shop.lab/rest/user/login

Zusammenfassung

Die Anwendung gibt bei der Verarbeitung nicht vertrauenswürdiger Eingaben Datenbankfehlerinformationen preis und ermöglicht dadurch fehlerbasierte SQL-Injection. Angreifer können SQL-Interpreterfehler auslösen und die zurückgegebenen Fehlermeldungen nutzen, um die Loginfunktion zu umgehen.

Auswirkung

Dies führt dazu, dass jeder Besucher der Website unter Zuhilfenahme dieser Schwachstelle als Administrator angemeldet ist.

Ferner ermöglicht die Offenlegung detaillierter Datenbankfehler eine gezielte Ableitung von Abfrageaufbau und Schema. Angreifer können Fehlerinhalte nutzen, um sensible Daten zu extrahieren, Datensätze zu verändern oder logische Kontrollen zu umgehen, die von Abfrageergebnissen abhängen. Anhaltende Offenlegung von SQL-Fehlern erhöht die Wahrscheinlichkeit einer vollständigen Datenkompromittierung für betroffene Tabellen und verbundene Entitäten.

Empfehlung

Keine detaillierten Datenbankfehler an Endnutzer anzeigen und alle nutzerkontrollierten Eingaben sicher behandeln.

Technische Beschreibung

Nicht vertrauenswürdige Eingaben werden ohne Parametrisierung in SQL-Abfragen eingebunden, sodass fehlerhafte SQL-Ausdrücke die Datenbank erreichen. Bei Auswertungsfehlern erzeugt die Datenbank detaillierte Fehlermeldungen, die die Anwendung in HTTP-Antworten oder der UI zurückgibt und damit interne Abfragekontexte offenlegt. Fehlerbasierte SQL-Injection nutzt diese offengelegten Fehler, um iterativ die Struktur der zugrunde liegenden Abfragen und des Schemas zu

ermitteln. Offengelegte Inhalte umfassen häufig SQLSTATE-Codes, Treibernamen, Ausnahme- und Stack-Informationen, Tabellen- oder Spaltenbezeichner, Funktionsnamen sowie Details zu Typkonflikten. Diese Signale erlauben präzise Anpassungen von Eingabewerten und offenbaren Spalten, Typen und Constraints, bis eine sinnvolle Datenabfrage oder -manipulation möglich wird. Die Hauptursache ist zeichenkettenbasierter Query-Aufbau und eine zu permissive Fehlerbehandlung, die niedrige Datenbankausnahmen an Endnutzer weitergibt. Sekundäre Faktoren sind uneinheitliche Eingabevalidierung und fehlende durchgesetzte parametrisierte Datenzugriffsmuster im gesamten Code. Das nachfolgende Beispiel zeigt eine generische Serverantwort, die einen Datenbankfehler offenlegt, ohne umgebungsspezifische Details preiszugeben.

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/plain; charset=utf-8

Fehler bei der Abfrageausführung auf /items?id=<value>
SQLSTATE[42000]: Syntax error or access violation: tatsächlicher DB-Fehlerauszug (z. B. Spalte nicht gefunden, Typkonflikt)
Treiber: Treiber/Version, falls offengelegt | Query-Fragment: Fragment, falls offengelegt
```

Typische Indikatoren für fehlerbasierte SQL-Injection sind Datenbankfehlercodes in Antworten, geleakte Bezeichner und Datentypen sowie Traces, die direkt auf Query-Konstruktionspfade verweisen. Sind diese Signale vorhanden, verringern sie die Unsicherheit für einen Angreifer erheblich und beschleunigen den Weg zur Datenkompromittierung.

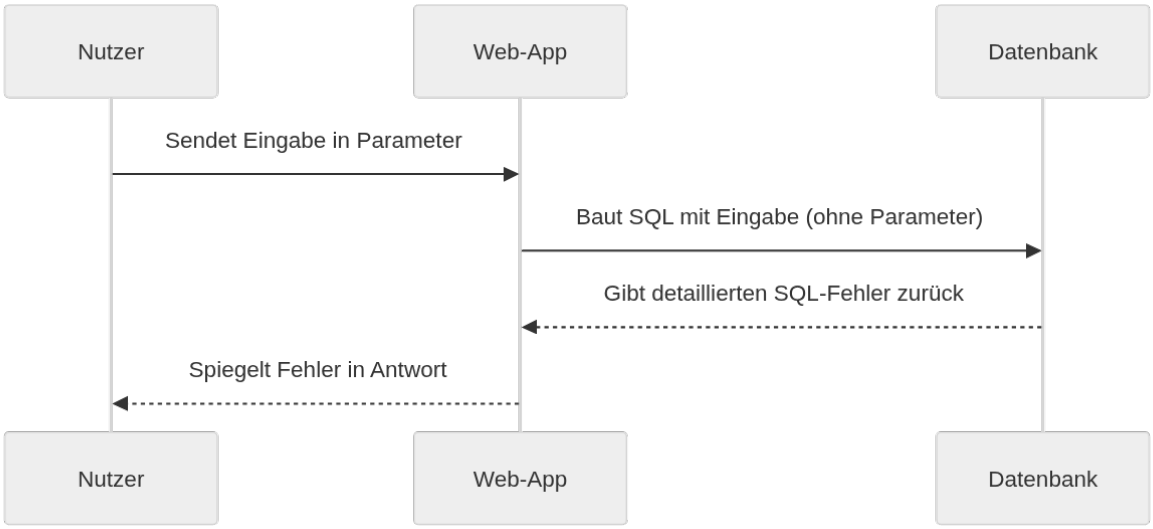

Abbildung 2 - Signalfluss bei fehlerbasierter SQL-Injection

Nachweise

In Abbildung 3 ist zu erkennen, dass ein SQL-Fehler ausgelöst wird, wenn die E-Mail-Adresse ein `'` enthält.

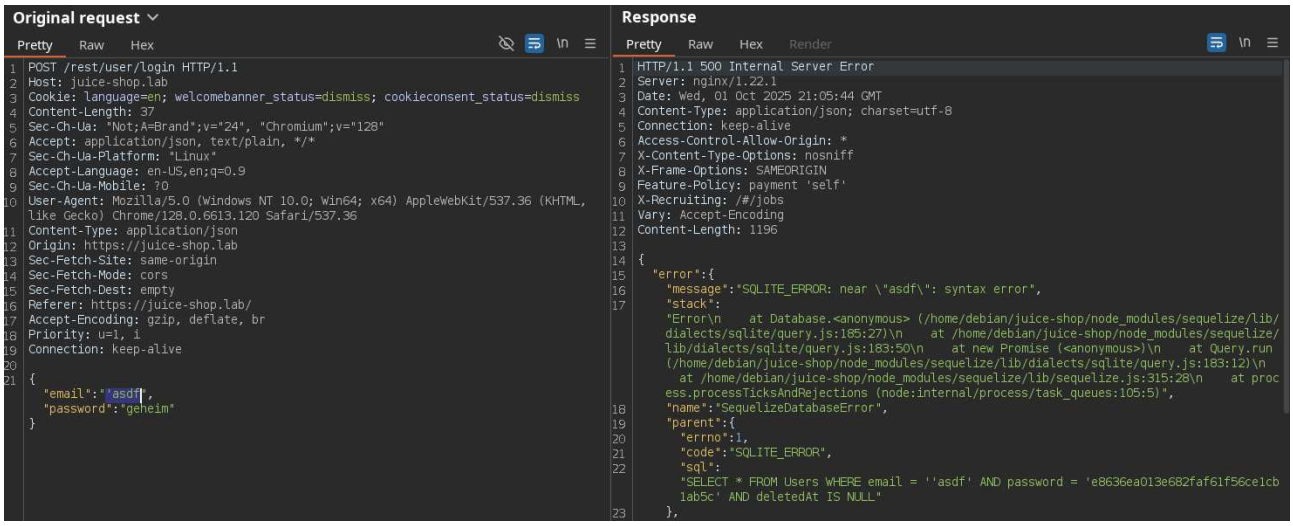

Abbildung 3 - SQL Fehler bei E-Mail

Durch geschicktes Wählen des Nutzernamens `' or 1 = 1; -- -` findet eine SQL-Injektion statt, wodurch sich der Login umgehen lässt. Schließlich ist man als Administrator eingeloggt (siehe Abbildung 4).

Folgendes passiert im Hintergrund beim Zusammenbau der Anfrage:

1. `'` schließt das Feld der E-Mail-Adresse
2. Danach wird SQL-Code verwendet
3. `or 1 = 1` ist eine wahre Aussage
4. `;` beendet die SQL-Anfrage
5. Der Rest der ursprünglichen Anweisung wird auskommentiert mit `--`

```
SELECT * FROM Users WHERE email = '' or 1 = 1; -- -' AND password =
'e8636ea013e682faf61f56ce1cb1ab5c' AND deletedAt IS NULL
```

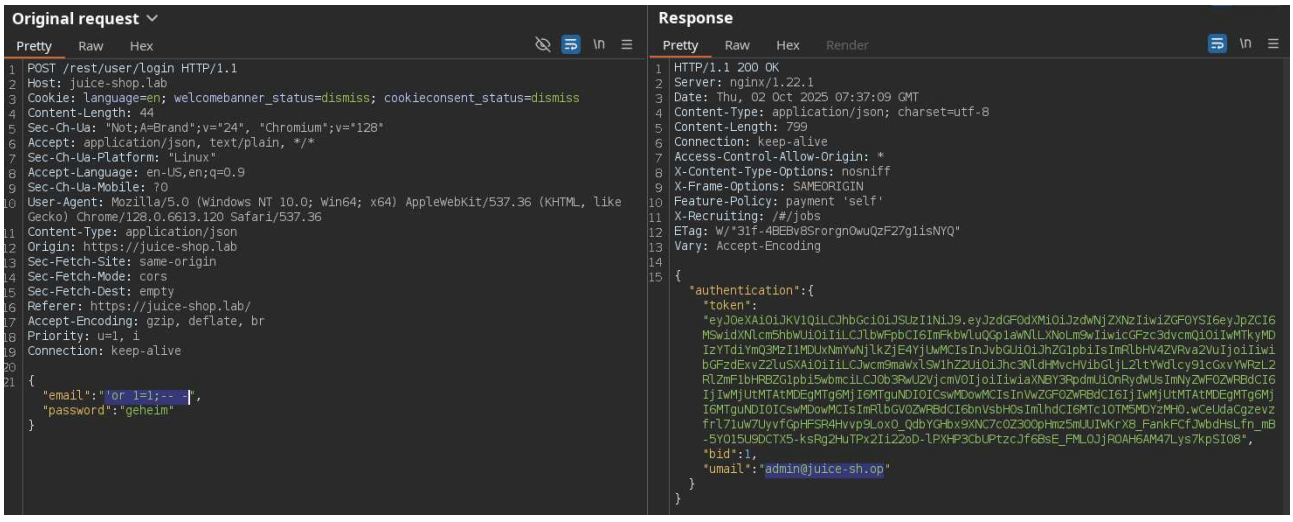

Abbildung 4 - SQL-Injektion Login Bypass

Dies erlaubt dann erfolgreichen Zugriff auf die Administrationsoberfläche unter *https://juice-shop.lab/#/administration* (siehe Abbildung 5). Hier können beispielsweise alle anderen Benutzer eingesehen und Rezensionen gelöscht werden.

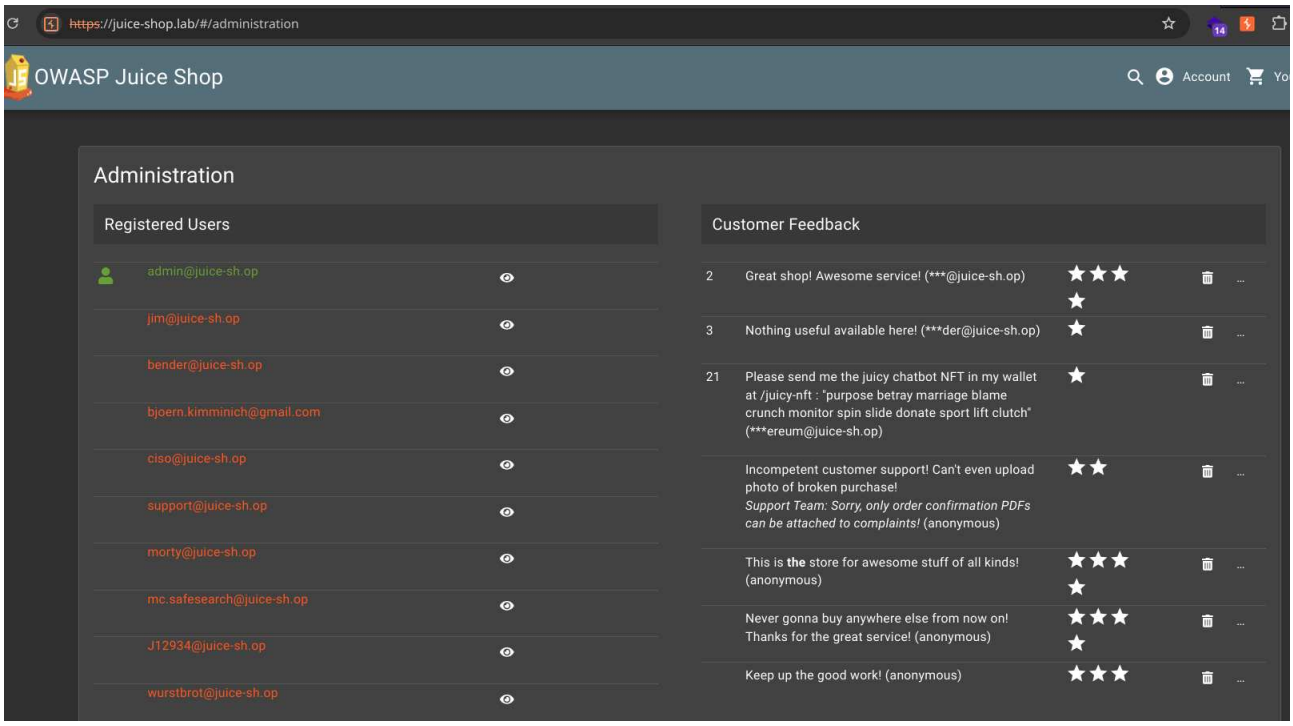

Abbildung 5 - Zugriff Administrationsoberfläche

Technische Empfehlung

Ersetzen Sie zeichenkettenbasierten SQL-Aufbau durch vorbereitete Statements (prepared Statements) mit Bind-Variablen für jede Abfrage mit nutzerkontrollierten Daten. Erzwingen Sie Parametrisierung über eine zentrale Data-Access-Schicht oder ORM-Konfiguration und untersagen Sie Ad-hoc-Query-Konstruktion in Code-Reviews und CI-Prüfungen. Implementieren Sie generische Fehlerbehandlung, die Datenbankausnahmen auf standardisierte Antworten (z. B. HTTP 500) abbildet, ohne SQLSTATE, Bezeichner oder Stack-Traces offenzulegen, und protokollieren Sie Details serverseitig mit Korrelations-IDs. Deaktivieren Sie ausführliche Fehlerseiten und detaillierte Ausnahmemeldungen in Produktionskonfigurationen für Web-Framework und Datenbanktreiber. Validieren und normalisieren Sie Eingaben nach strengen Typ- und Längen-Constraints, und weisen Sie unerwartete Formate frühzeitig zurück. Inventarisieren und refaktorisieren Sie alle Endpunkte mit Eingaben in Abfragen, und fügen Sie Tests hinzu, die Nicht-Offenlegung von Fehlern und Parametrisierung sicherstellen. Ergänzen Sie statische Analyse- oder Lint-Regeln zur Erkennung von stringbasiertem SQL und zur Durchsetzung vorbereiteter Statements im gesamten Code. Beispiel (konzeptionell):

```
// Vorher: verwundbar
String sql = "SELECT * FROM items WHERE id = " + id_variable; // String-Konkatenation
stmt.executeQuery(sql);

// Nachher: parametrisiert
PreparedStatement ps = conn.prepareStatement("SELECT * FROM items WHERE id = ?");
ps.setInt(1, id_variable);
ps.executeQuery();
```

Referenzen

- *https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection*
- *https://owasp.org/www-community/attacks/SQL_Injection*

- *https://cwe.mitre.org/data/definitions/89.html*
- *https://portswigger.net/web-security/sql-injection*

6.1 M1: Cross-Site Scripting (XSS) – Reflektiert

Vektor	*CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N*
Status	Offen
Sofortmaßnahme empfohlen	Nein
Quick Win zur Behebung	Ja ✳
Tags	WSTG-CLNT-01, CWE-79
Betroffene Komponenten	https://juice-shop.lab/#/search?q=

Zusammenfassung

Die Anwendung reflektiert benutzerbereitgestellte Eingaben ohne Ausgabeenkodierung in die Antwort. Dadurch kann ein Angreifer einen Link oder eine Anfrage gestalten, der im Browser des Opfers skriptgesteuerten Code innerhalb der Anwendung ausführt. Der Angriff ist flüchtig und erfordert Benutzerinteraktion, kann jedoch zu Kontenmissbrauch, Datenexposition und unbefugten Aktionen über die Sitzung des Opfers führen.

Auswirkung

Erfolgreiche Ausnutzung ermöglicht die Ausführung angreiferkontrollierten Skripts im Browser des Opfers innerhalb der Anwendung. Mögliche Folgen umfassen Sitzungsmissbrauch, unbefugte Aktionen im Namen des Benutzers, Offenlegung von auf der Seite sichtbaren Daten sowie Manipulation vertrauenswürdiger UI-Elemente zur Durchführung glaubwürdiger Phishing-Angriffe innerhalb der Anwendung. Der Angriff erfordert Benutzerinteraktion (z. B. Aufruf eines präparierten Links oder Absenden von Daten) und kann alle Benutzer betreffen, die mit der präparierten Eingabe interagieren.

Empfehlung

Kontextsensitive Ausgabeenkodierung für alle reflektierten Daten konsequent anwenden und, wenn möglich, eine restriktive Content Security Policy erzwingen.

Technische Beschreibung

Reflektiertes Cross-Site Scripting tritt auf, wenn unkontrollierte Benutzereingaben unmittelbar und ohne geeignete Ausgabeenkodierung für den jeweiligen Rendering-Kontext in einer Serverantwort zurückgegeben werden. Typische Quellen sind Query-Parameter, Pfadsegmente, Formularfelder oder Header, die per String-Konkatenation in HTML, Attribute, URLs oder Inline-Skripte eingefügt werden. Die Hauptursache ist das Fehlen oder Umgehen kontextsensitiver Ausgabeenkodierung sowie unsichere Render-Muster, die nicht vertrauenswürdige Eingaben wie vertrauenswürdiges Markup oder Skript behandeln. Framework-Auto-Escaping kann deaktiviert, falsch konfiguriert oder durch unsichere Template-Konstrukte umgangen sein. Ungeeignete Inhaltsbehandlung, etwa die Rückgabe von HTML mit nicht vertrauenswürdigen bei permissivem Response-Typ, kann die Exposition erhöhen, auch wenn die eigentliche Ursache die fehlende korrekte Enkodierung bleibt. Eine restriktive Content Security Policy (CSP) kann die Auswirkungen begrenzen, behebt jedoch nicht das zugrunde liegende

Problem. Das folgende Beispiel zeigt einen Parameter, der ohne Enkodierung in HTML-Inhalt reflektiert wird und illustriert, wie nicht vertrauenswürdige Eingaben im Browser gerendert werden:

```
GET /search?q=<untrusted_input> HTTP/1.1
Host: host
Accept: text/html
```

```
<!doctype html>
<html>
  <body>
    <p>Suche nach: <untrusted_input></p>
  </body>
</html>
```

Nachweise

- Betroffener Endpunkt und Parameter:
  - URL: *https://juice-shop.lab/#/search*
  - Parameter: *q*

In Abbildung 6 ist zu erkennen, dass die Eingabe `asdf` des Suchfelds im Kontext Search Results der Webapplikation gespiegelt wird.

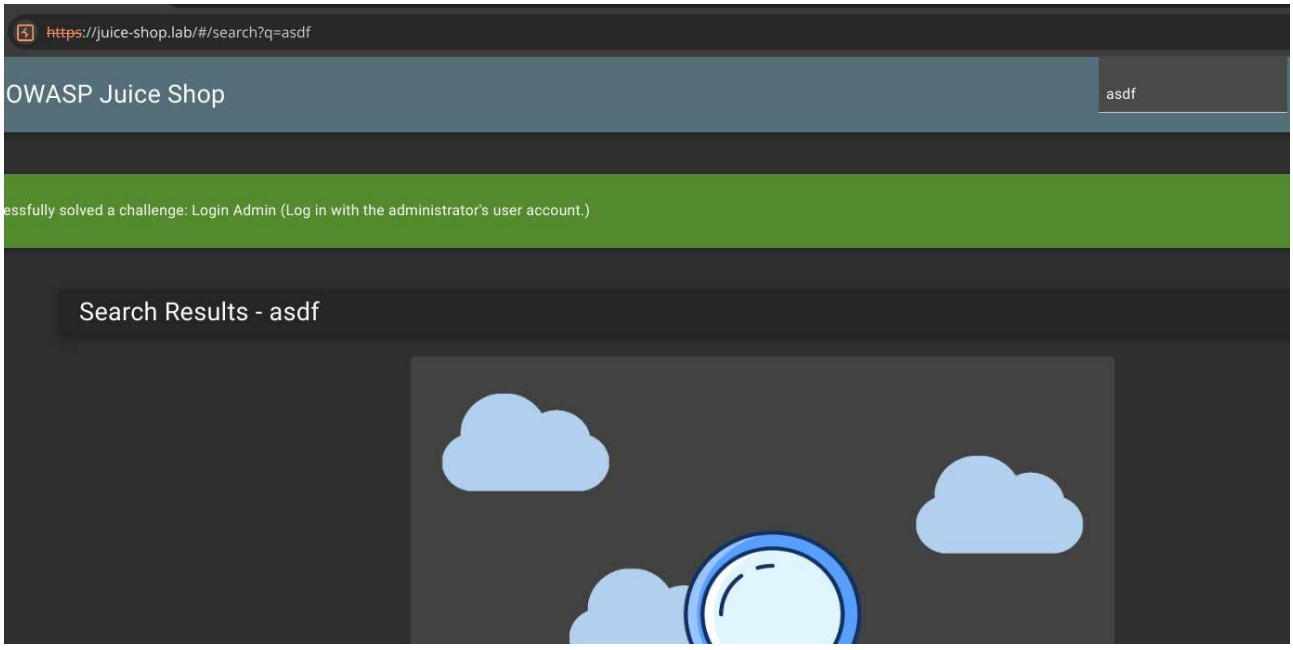

Abbildung 6 - Reflektierter Input aus Suchfeld

Durch Eingabe des folgenden JavaScript-Codes wird eine Pop-up-Meldung ausgegeben, welche die Ausführung dieses Codes bestätigt (siehe Abbildung 7).

```
<iframe%20src%3D"javascript:alert('Schutzpunkt%20XSS')">
```

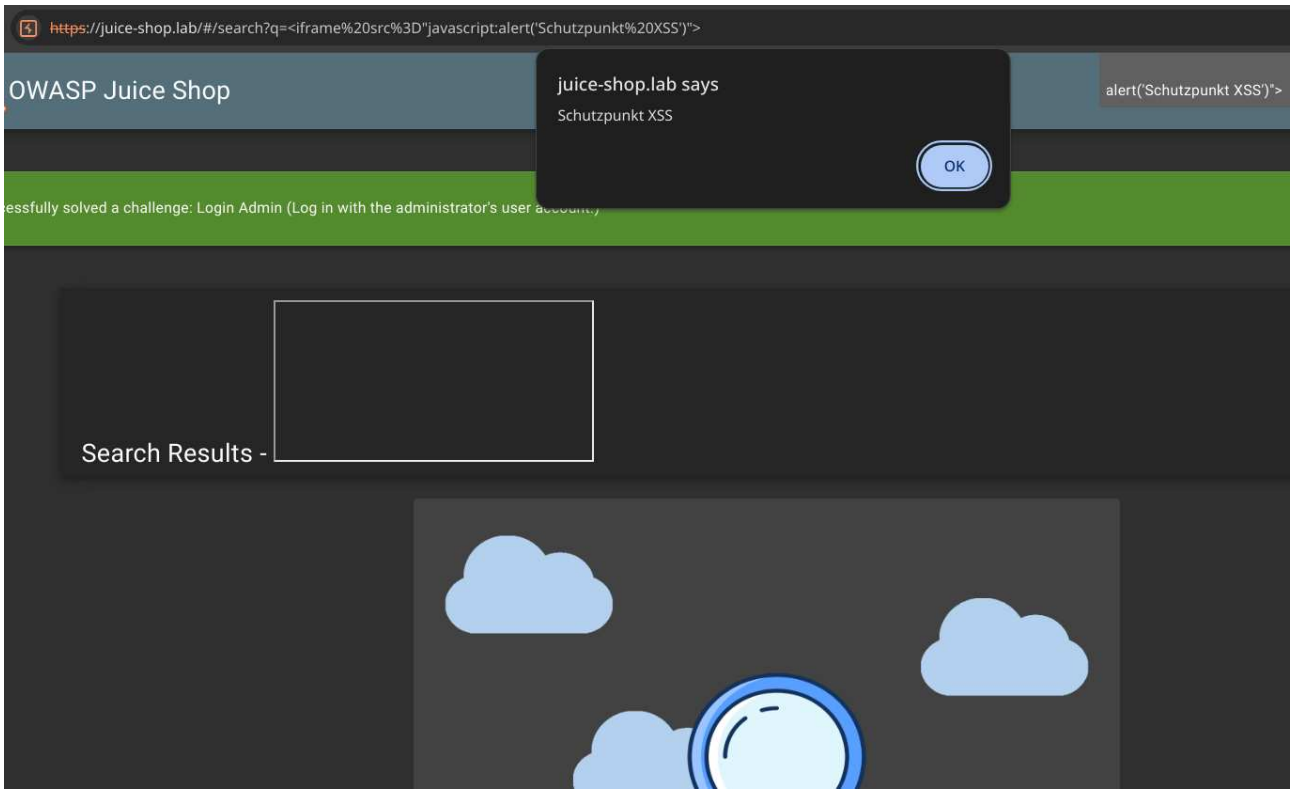

Abbildung 7 - Reflektiertes XSS

Abbildung 8 zeigt die betroffene Stelle im Frontend Code der Anwendung.

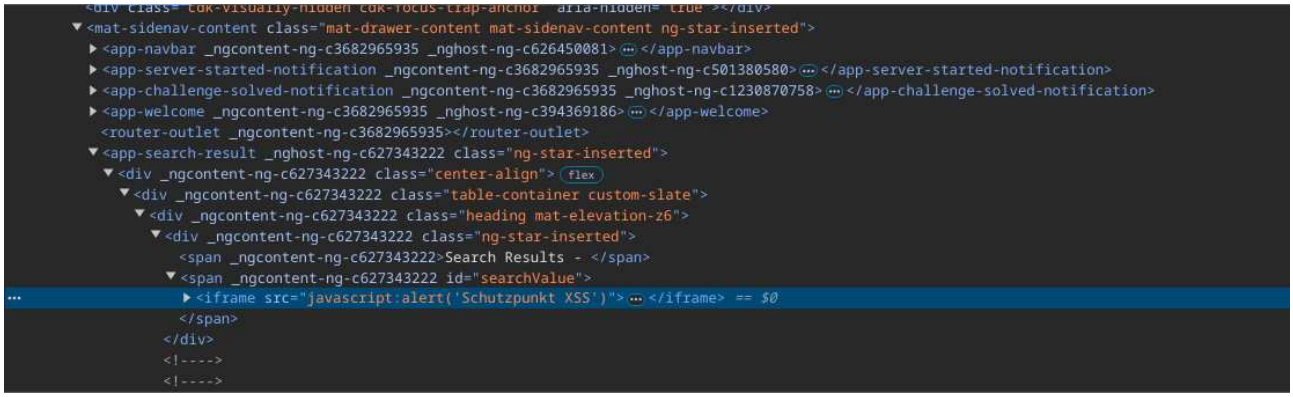

Abbildung 8 - Stelle im Code mit XSS

Technische Empfehlung

Identifizieren Sie alle Stellen, an denen nicht vertrauenswürdige Eingaben in Antworten reflektiert werden, und wenden Sie vor dem Rendern kontextsensitive Ausgabeenkodierung an. Nutzen Sie Framework-Funktionen mit standardmäßigem Auto-Escaping und vermeiden Sie die Konkatenation nicht vertrauenswürdiger Eingaben in HTML, Attributen, URLs, CSS oder JavaScript-Kontexten. Wenden Sie den passenden Encoder für den jeweiligen Kontext an, etwa HTML-Text, HTML-Attribut, URL-Parameter oder JavaScript-String, anstatt sich auf generische oder Blacklist-Filter zu verlassen. Vermeiden Sie das Rendern nicht vertrauenswürdiger Eingaben als HTML und verwenden Sie sichere Templating- und DOM-APIs, die Textknoten statt Markup verarbeiten. Setzen Sie eine restriktive CSP um, die Inline-Skripte verbietet und Skriptquellen auf geprüfte Ursprünge beschränkt, um Auswirkungen bei verbleibenden Lücken zu begrenzen. Ergänzen Sie Unit- und Integrationstests, die

prüfen, dass reflektierte Felder enkodiert sind und gefährliche Zeichen in ihren jeweiligen Kontexten unschädlich gerendert werden.

Beispiel (illustrativ):

```
// Template- oder serverseitiges Rendering
String safe = org.owasp.encoder.Encode.forHtml(userInput);
out.print(safe);
```

Referenzen

- *https://owasp.org/www-community/attacks/xss/*
- *https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html*
- *https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting*
- *https://cwe.mitre.org/data/definitions/79.html*

5.3 M2: Fehler in der Rabattgutschein-Logik

Vektor	*CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N*
Status	Offen
Sofortmaßnahme empfohlen	Ja ‼️
Quick Win zur Behebung	Nein oder zu wenige Informationen
Tags	WSTG-BUSL-01, WSTG-BUSL-02, CWE-840, CWE-345, ATT&CK-T1190
Betroffene Komponenten	https://juice-shop.lab/rest/basket/6/coupon/

Zusammenfassung

Die Rabattgutschein-Logik der Anwendung erlaubt unbeabsichtigte Preisreduktionen aufgrund unzureichender serverseitiger Validierung und schwacher Durchsetzung von Einlöse-Regeln. Angreifer können Gutscheine über die vorgesehenen Rabattgrenzen hinaus anwenden. Die Schwachstelle ist ein Konstruktionsfehler in der Gutscheinlogik und kein kosmetisches Problem.

Auswirkung

Unautorisierte Rabattanwendungen können zu direkten Umsatzeinbußen, Margenerosion und verfälschter Finanzberichterstattung führen. Bestände können zu nicht beabsichtigten Preisen abfließen, und Marketingbudgets sowie Kampagnenanalysen werden unzuverlässig. Missbrauch kann zu Betrugsmustern wie Arbitrage, Weiterverkauf oder Account Farming eskalieren.

Voraussetzungen

Ein oder mehrere Rabattgutscheine erhalten oder gesammelt zu haben, um ein Schema ableiten zu können.

Empfehlung

Neues Gutscheinkonzept implementieren oder zumindest Gutscheinregeln umfassend serverseitig validieren.

Technische Beschreibung

Das Rabattsystem validiert und erzwingt die Gutschriftsregeln nicht strikt auf dem Server, wodurch ein Missbrauch von Gutscheinvorteilen möglich wird. Typische Lücken sind fehlende Prüfungen für Einmalnutzung, keine Bindung eines Gutscheins an ein spezifisches Konto oder eine Bestellung, ausbleibende Validierung von Rabatten, fehlerhafte Behandlung von Stapelungsregeln und das Nichtverhindern negativer oder nahezu nuller Gesamtsummen.

Wenn der Gutscheinstatus clientseitig aus veränderbaren Parametern abgeleitet wird, können Angreifer Werte manipulieren oder Anfragen wiederholen, um wiederholte Rabatte zu erzielen. Schwache Eingabevalidierung kann zudem unbeabsichtigte Kombinationen erlauben.

Nachweise

Vom Kunden wurden folgende Gutscheincodes bereitgestellt.

Code	Rabatt	Gültigkeit
q:<Irh7ZKp	10%	September 2025
pEw8ph7ZKp	10%	Oktober 2025
pEw8ph7ZKu	15%	Oktober 2025
pes[Ch7ZKp	10%	November 2025

Dekodierung

Auffällig ist hierbei, dass diese sich anhand des Datums und des Prozentsatzes ähneln. Der Teil `h7ZK` scheint hierbei immer gleich zu bleiben, was auf eine Kodierung der Gutscheine schließen lässt. Diese erwies sich als Base85 Kodierung, wie in Abbildung 9 zu sehen ist und ist wie folgt aufgebaut: `MMMYY-%%` wobei `%` für den Nachlass in Prozent steht.

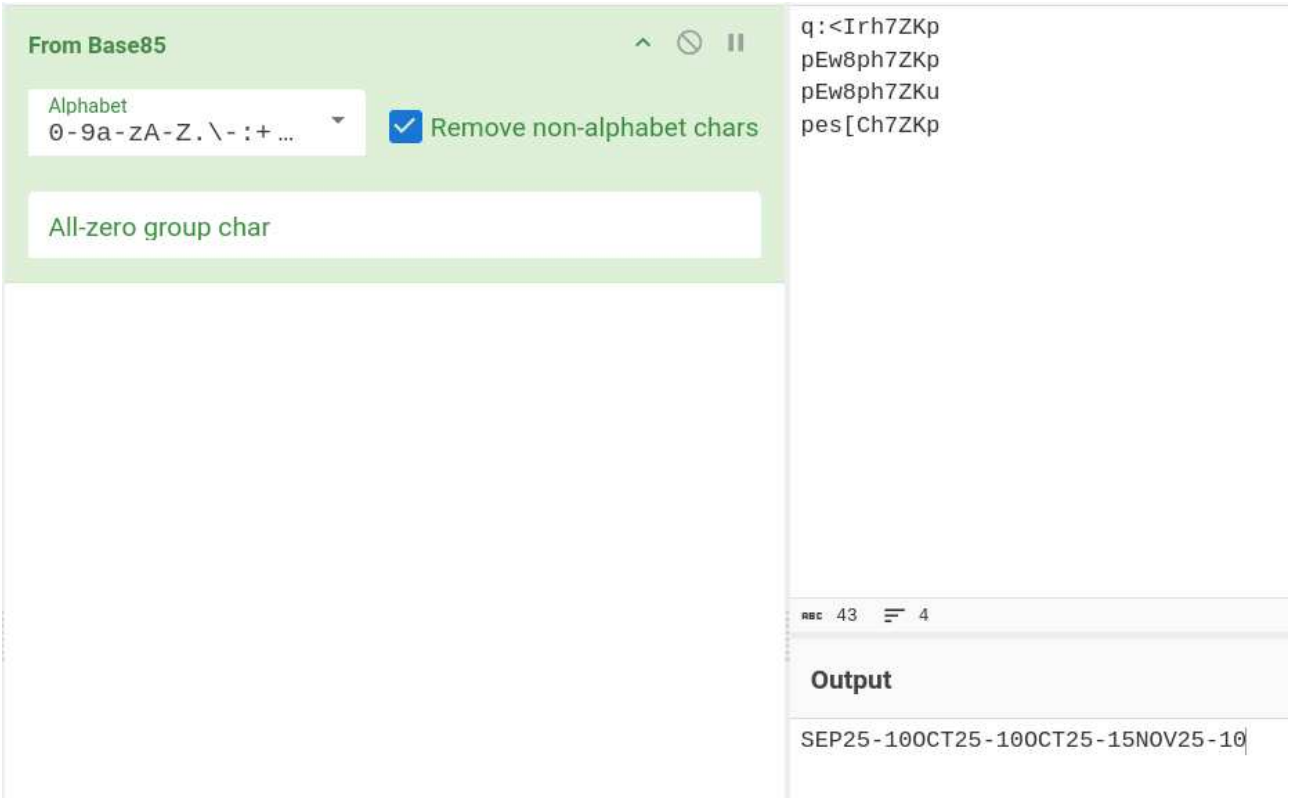

Abbildung 9 - Base85 Dekodierte Gutscheine

Erstellung eigenes eigenen Gutscheins

Anschließend wurde ein neuer Gutschein `OCT25-99` erstellt (Abbildung 10). Der Wert dieses Gutscheins ist `pEw8ph7Z*G`

Abbildung 10 - Selbst erstellter Gutschein

Einlösen des Gutscheins

Nachfolgendes Request-Response-Paar zeigt den Versuch, diesen selbst generierten Gutschein einzulösen. In der Checkout-Übersicht ist zu sehen, dass dies erfolgreich war (siehe Abbildung 11).

Request

```
PUT /rest/basket/6/coupon/pEw8ph7Z*G HTTP/1.1
Host: juice-shop.lab
[...SNIP...]
Content-Type: application/json
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Platform: "Linux"
Origin: https://juice-shop.lab
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://juice-shop.lab/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive
```

```
Content-Length: 2

{}
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.22.1
Date: Thu, 02 Oct 2025 13:00:07 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 15
Connection: keep-alive
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
ETag: W/"f-EKshmF+cUf70Vv3BHGSC98QSEKM"
Vary: Accept-Encoding

{"discount":99}
```

Checkout-Übersicht

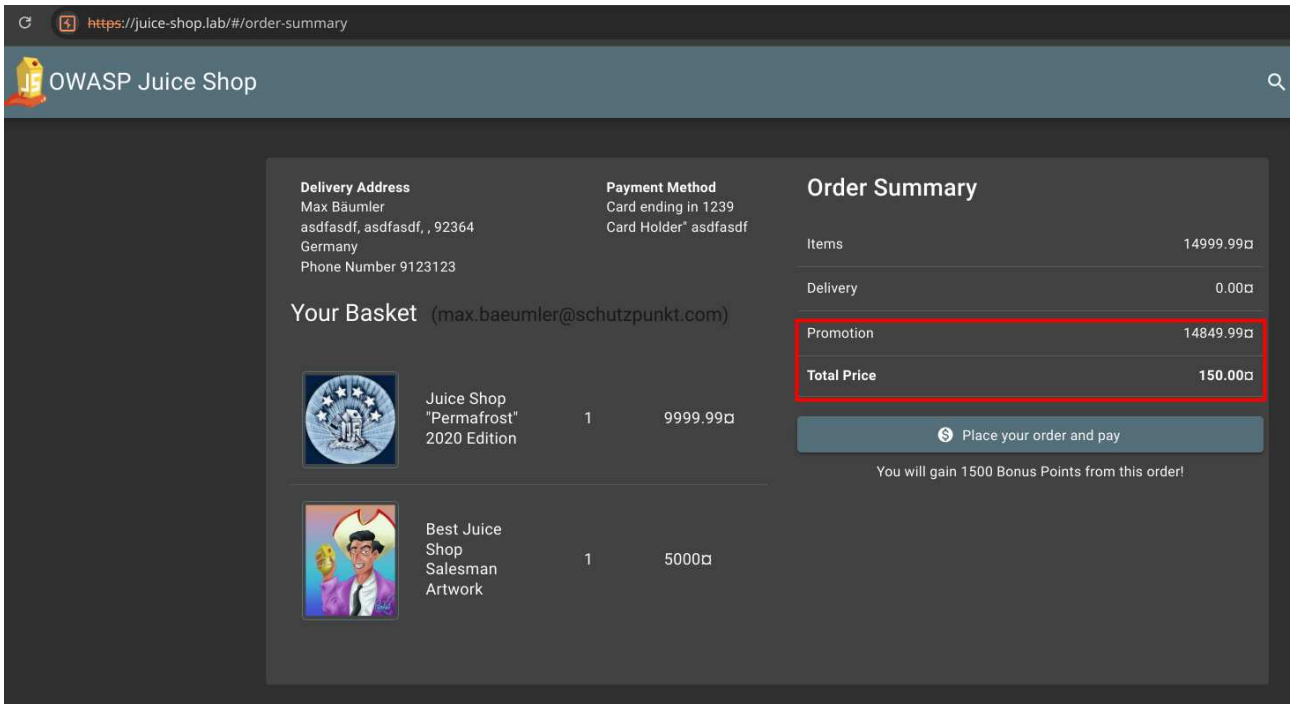

Abbildung 11 - Rabatt von 99 % erhalten

Technische Empfehlung

Serverseitig überprüfen, ob ein Gutschein wirklich erlaubt ist. Für November beispielsweise checken, ob der Gutschein wirklich exakt `NOV25-10` ist und nicht aus dem Text den Rabatt berechnen. Es wird jedoch eine Neuimplementierung der Gutscheinlogik empfohlen, da durch Unachtsamkeit immer wieder das gleiche Problem entstehen kann. Möchte man beispielsweise einem Kunden einen

Gutschein mit mehr Rabatt gewähren, ist das aktuell nicht möglich, sondern erlaubt allen Nutzern, sich diesen Rabatt zu erschleichen.

Gutscheinbedingungen vor jeder Preisänderung serverseitig validieren und erzwingen, einschließlich Berechtigung (Nutzer, Segment, Geografie), Gültigkeitsfenster, Mindestbestellwert, Artikel-/Kategorie-Scope, Stapelungsgrenzen und maximaler Rabatt. Gutscheininstanzen an ein eindeutiges Subjekt wie Nutzer-ID oder Bestell-ID binden und den Zustand mit atomaren Transaktionen persistieren, um Einmal- oder Limitnutzung sicherzustellen. Alle Gutschein-Lebenszyklusereignisse mit Korrelations-IDs protokollieren und auf Anomalien wie ungewöhnlich hohe Einlöseraten oder schnelle Sequenzen überwachen. Unit- und Integrationstests für Regelbewertung, Idempotenz und Konkurrenzsituationen hinzufügen, um Regressionen zu vermeiden.

Referenzen

- *https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Business_Logic_Testing/01-Testing_for_business_logic*
- *https://cheatsheetseries.owasp.org/cheatsheets/Business_Logic_Security_Cheat_Sheet.html*
- *https://portswigger.net/web-security/logic-flaws*
- *https://owasp.org/Top10/A04_2021-Insecure_Design/*

5.3 M3: Verzeichnisauflistung aktiviert

Vektor	*CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N*
Status	Offen
Sofortmaßnahme empfohlen	Nein
Quick Win zur Behebung	Ja ✳
Tags	CWE-548, CWE-200
Betroffene Komponenten	https://juice-shop.lab/ftp

Zusammenfassung

Der Webserver erlaubt die Verzeichnisauflistung auf einem oder mehreren öffentlich zugänglichen Pfaden. Dadurch werden Dateinamen und die Struktur von Verzeichnissen für jeden Besucher sichtbar, auch ohne Authentifizierung. Solche Listings führen zu Informationspreisgabe und erleichtern Informationsbeschaffung sowie zielgerichtete Angriffe.

Auswirkung

Unbeschränkte Verzeichnisauflistung kann sensible Dateien, Konfigurationsreste, Backups und Zugangsdaten in webzugänglichen Pfaden offenlegen. Offengelegte Dateinamen und Strukturen ermöglichen schnellere Informationsbeschaffung, zielgerichtete Angriffe und das Auffinden übersehener administrativer Endpunkte. Informationspreisgabe kann nachfolgende Angriffe unterstützen, etwa das Auslesen von Konfigurationsdateien mit Zugangsdaten, die Ausnutzung veralteter Komponenten, die in Listings sichtbar werden, oder das unbefugte Herunterladen proprietärer Inhalte. Werden Listings von Suchmaschinen indexiert, besteht die Exposition langfristig und über die unmittelbare Zielgruppe hinaus.

Empfehlung

Verzeichnisauflistung auf allen öffentlich zugänglichen Webpfaden deaktivieren.

Technische Beschreibung

Verzeichnisauflistung tritt auf, wenn der Server für ein Verzeichnis ohne Standard-Indexdatei (z. B. index.html) automatisch eine Übersichtsseite generiert und zurückliefert. Dies ergibt sich meist durch aktivierte Funktionen wie Apache mod_autoindex, Nginx autoindex oder IIS Directory Browsing, beziehungsweise durch nicht gehärtete Standardkonfigurationen. Ist die Auflistung aktiviert, liefert eine Anfrage an einen Verzeichnispfad (zum Beispiel /assets/) eine Seite mit Dateien, Größen, Zeitstempeln und Unterverzeichnissen. Solche Listings offenbaren interne Dateibenennungen, Build-Artefakte, Backups, temporäre Dateien und Konfigurationsfragmente, die nicht öffentlich zugänglich sein sollten. Angreifer und automatisierte Crawler können Verzeichnisse iterieren, um die Anwendungsstruktur zu kartieren und sensible Dateien schneller zu finden. Öffentlich erreichbare Listings erfordern keine Authentifizierung und können von Suchmaschinen gecacht oder indexiert werden, was die unbeabsichtigte Exposition verstärkt. Dieses Verhalten ist häufig unbeabsichtigt und bleibt bestehen, bis die Verzeichnisauflistung explizit deaktiviert oder eine Standard-Indexdatei bereitgestellt wird.

Beispiel für Anfrage und Antwort, die eine aktivierte Auflistung zeigen.

```
GET /assets/ HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html>
  <head><title>Index of /assets/</title></head>
  <body>
    <h1>Index of /assets/</h1>
    <a href="backup.zip">backup.zip</a>
    <a href=".env">.env</a>
    <a href="config.old">config.old</a>
  </body>
</html>
```

Nachweise

- Beobachtete URLs mit aktivierter Verzeichnisauflistung:
  - *https://juice-shop.lab/ftp*

Wie in Abbildung 12 zu sehen, sind unter oben genanntem Pfad auch mehrere sensible Dateien zu lesen wie:

- incident-support.kdbx
- suspicious_errors.yml
- coupons_2013.md.bak

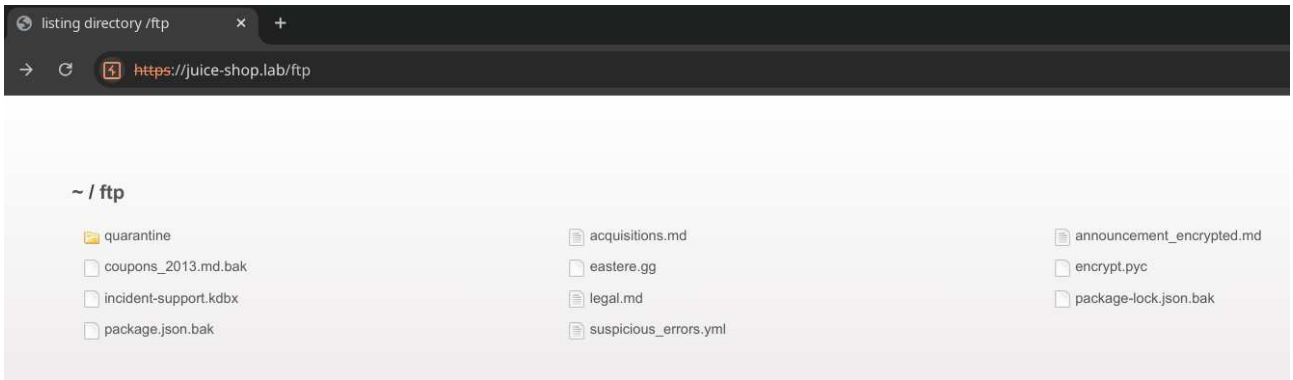

Abbildung 12 - Directory Listing Aktiviert

Technische Empfehlung

Verzeichnisauflistung serverseitig für alle öffentlichen Websites und Anwendungen abschalten und minimale Exposition für notwendige Pfade durchsetzen. In Apache HTTP Server die Auflistung global oder pro Verzeichnis deaktivieren, indem autoindex entfernt und eine restriktive Options-Direktive verwendet wird.

```
# httpd.conf oder .htaccess
Options -Indexes
```

```
# Sicherstellen, dass mod_autoindex nicht geladen ist, falls nicht benötigt
# LoadModule autoindex_module modules/mod_autoindex.so  # auskommentieren oder entfernen
```

In Nginx autoindex für relevante Locations oder Server-Blöcke ausschalten.

```
server {
  location / {
    autoindex off;
  }
}
```

In IIS Directory Browsing auf Site- oder Anwendungsebene deaktivieren oder via web.config erzwingen.

```
<!-- web.config -->
<configuration>
  <system.webServer>
    <directoryBrowse enabled="false" />
  </system.webServer>
</configuration>
```

Wenn eine Auflistung aus legitimen Gründen erforderlich ist, nur für authentifizierte und rollenbeschränkte Benutzer zulassen und strikt von öffentlichen Routen isolieren. Sicherstellen, dass keine sensiblen Dateien in webzugänglichen Verzeichnissen liegen, und bei Bedarf eine Standard-Indexseite bereitstellen, um automatisch generierte Listings zu verhindern.

Referenzen

- *https://cwe.mitre.org/data/definitions/548.html*
- *https://owasp.org/Top10/A05_2021-Security_Misconfiguration*
- *https://httpd.apache.org/docs/current/en/mod/mod_autoindex.html*
- *https://nginx.org/en/docs/http/ngx_http_autoindex_module.html*

0.0 I1: Informationspreisgabe über Stack-Traces

Vektor	*CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N*
Status	Offen
Sofortmaßnahme empfohlen	Nein
Quick Win zur Behebung	Nein oder zu wenige Informationen
Tags	WSTG-ERRH-02, CWE-209
Betroffene Komponenten	https://juice-shop.lab/rest/<anything>

Zusammenfassung

Die Anwendung zeigt bei Fehlern detaillierte Fehlermeldungen (Stack-Traces) für Endnutzer an. Diese offenbaren interne Codepfade, Framework- und Bibliotheksversionen, Konfigurationsdetails sowie Dateisystempfade. Solche Informationen unterstützen effiziente Aufklärung und können für zielgerichtete Angriffe genutzt werden. Das Verhalten weist auf unzureichendes Fehlerhandling und aktivierte Debug-Einstellungen in Produktionsumgebungen hin.

Auswirkung

Angreifer können anhand der Inhalte von detaillierten Fehlermeldungen (Stack-Traces) Frameworks, Versionen und Bibliotheken enumerieren und dadurch nach versionsspezifischem Schadcode suchen. Dateipfade und Klassennamen offenbaren die Anwendungsstruktur und erleichtern gezielte Tests von Komponenten und fehleranfälligen Schnittstellen. Detaillierte Ausnahme-Meldungen können Konfigurationswerte und Betriebskontext offenlegen und erhöhen die Wahrscheinlichkeit einer Datenexposition über sekundäre Schwachstellen.

Empfehlung

Deaktivieren Sie die Anzeige von Stack-Traces in Produktion und geben Sie generische Fehlermeldungen aus, während detaillierte Diagnosen serverseitig protokolliert werden.

Technische Beschreibung

Diese Schwachstelle entsteht, wenn unbehandelte Ausnahmen oder falsch konfigurierte Fehlerbehandler rohe Stack-Traces in HTTP-Antworten oder UI-Seiten zurückgeben. Ausführliche Fehlerausgaben sind in Entwicklungs- oder Debug-Modi oft standardmäßig aktiviert und werden versehentlich in Produktion beibehalten. Frameworks liefern häufig Standard-Fehlerseiten, die Ausnahme-Meldungen, Klassennamen, Dateipfade, Zeilennummern und Versionsinformationen von Abhängigkeiten ausgeben. Die Rückgabe dieser Details an Clients verschafft Angreifern präzise Einblicke in die interne Architektur, verwendete Technologien und potenzielle Schwachpunkte. Ursachen sind fehlendes globales Exception-Handling, fehlende Umgebungsumschaltung für Debug-Flags, direkte Serialisierung von Exceptions und unzureichende Normalisierung von API-Fehlern. Typische Indikatoren sind 500-Antworten mit mehrzeiligen Traces, Fehlerseiten mit Framework-Branding und Versionsangaben sowie Offenlegung von Verzeichnis- oder Dateipfaden. Das Problem

ist eine Inhalts- und Informationspreisgabe, keine Verfügbarkeits- oder Autorisierungsstörung, erhöht aber direkt die Angreiferfähigkeit, andere Schwachstellen auszunutzen. Die Vermeidung erfordert die Weiterleitung aller Fehler über standardisierte Handler, die generische Meldungen liefern und detaillierte Infos ausschließlich serverseitig protokollieren. Das folgende Beispiel veranschaulicht eine repräsentative geleakte Antwort.

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/plain; charset=utf-8

java.lang.NullPointerException: Cannot read property 'id' of undefined
    at com.example.controllers.UserController.getUser(UserController.java:42)
    at
org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1012)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:661)
/opt/app/services/user-service/src/main/java/com/example/controllers/UserController.java:42
Framework: Spring Boot 2.6.3
```

Nachweise

Ruft man einen der REST-API unbekannten Endpunkt hier *asdf* unter der URL *https://juice-shop.lab/rest/* auf, so erhält man einen Stack-Trace der Anwendung (Abbildung 13). Dieser enthält folgende Informationen:

- Versionsnummer + Technologie: *Express.js 4.21.0*
- Technologie:*Angular.js*
- Pfad: `/home/debian/juice-shop`

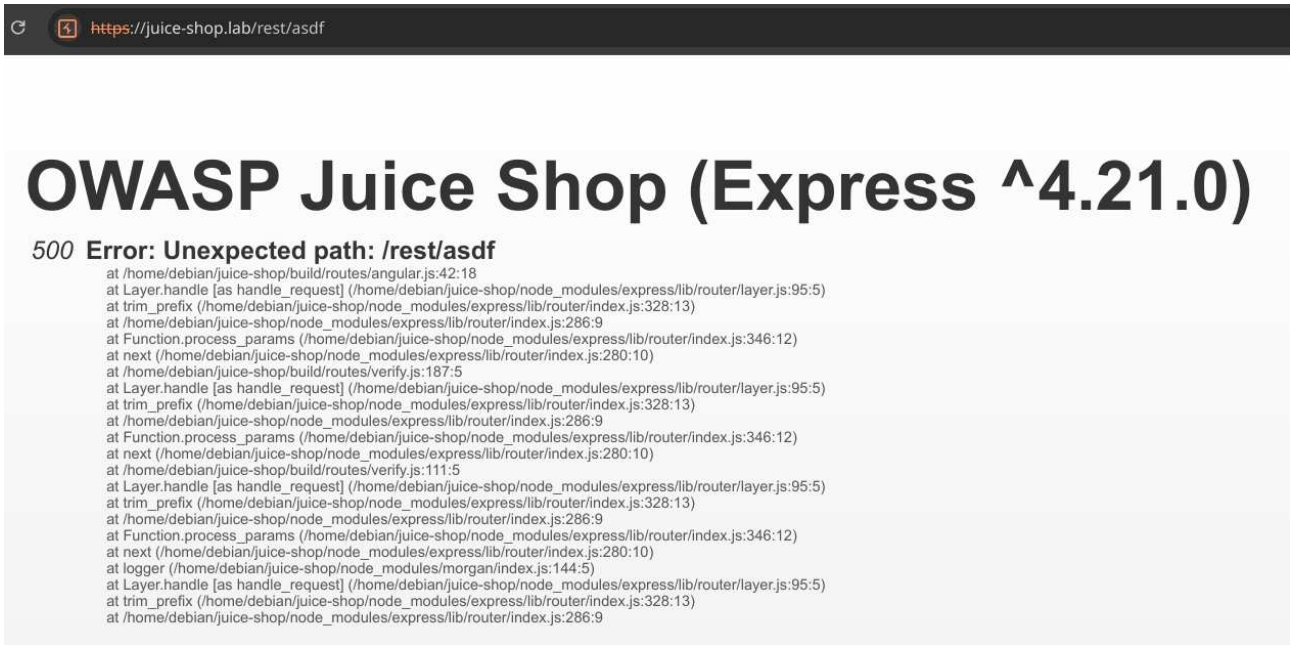

Abbildung 13 - Stack-Trace an unbekanntem REST-Endpunkt

Technische Empfehlung

Implementieren Sie zentrales Exception-Handling, das alle unbehandelten Fehler auf generische Antworten ohne Stack- oder Pfadangaben abbildet. Stellen Sie über Umweltumschaltung sicher, dass Debug- oder Entwickler-Fehlerseiten in Produktion deaktiviert sind, z. B. `debug=false` und äquivalente Einstellungen in allen Diensten. Standardisieren Sie API-Fehlerformate auf ein minimales Schema, wie

einen undurchsichtigen Fehlercode und eine Korrelations-ID, und geben Sie keine Exception-Meldungen zurück. Protokollieren Sie vollständige Diagnosen einschließlich Stack-Traces ausschließlich serverseitig mit geeignetem Zugriffsschutz und Aufbewahrung, nicht in Client-Antworten. Überprüfen Sie Webserver- und Framework-Konfigurationen, sodass keine standardmäßigen ausführlichen Fehlerseiten extern sichtbar sind. Validieren Sie das Verhalten mit automatisierten Tests, die das Fehlen von Stack-Traces und internen Pfaden in allen Fehlerszenarien sicherstellen.

```
// Beispiel für zentrales Fehlerhandling (generische Antwort)
app.use(function errorHandler(err, req, res, next) {
  const korrelationsId = generateId();
  logError({ korrelationsId, err }); // serverseitiges Log enthält err.stack
  // Keine Rückgabe von err.stack oder internen Pfaden an den Client
  res.status(500).json({ error: "Ein unerwarteter Fehler ist aufgetreten.",
korrelationsId });
});
```

Referenzen

- *https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/10-Error_Handling/02-Testing_for_Stack_Trace*
- *https://cwe.mitre.org/data/definitions/209.html*
- *https://owasp.org/Top10/A05_2021-Security_Misconfiguration/*
- *https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html*

5 Haftungsausschluss

Dieser Auftrag wurde mit dem Ziel durchgeführt, potenzielle Sicherheitslücken zu identifizieren und umsetzbare Empfehlungen bereitzustellen. Es ist jedoch Folgendes zu beachten:

- **Keine Gewähr für Vollständigkeit:** Obwohl der Test darauf ausgelegt ist, Schwachstellen innerhalb des Scopes zu identifizieren, gibt es keine Garantie dafür, dass alle Schwachstellen, Bedrohungen oder Risiken entdeckt wurden. Die Ergebnisse sind nicht als vollständig zu betrachten, und im Laufe der Zeit können neue Schwachstellen entstehen.
- **Zeit- und Materialansatz:** Der Auftrag folgt einem Zeit- und Materialansatz, bei dem der Testaufwand anhand der aufgewendeten Zeit, der eingesetzten Ressourcen und der Komplexität der durchgeführten Aufgaben abgerechnet wird. Die Ergebnisse sind daher im Kontext des Testzeitraums und der bereitgestellten Ressourcen zu betrachten.
- **Zeitliche Gültigkeit der Ergebnisse:** Die identifizierten Findings und Schwachstellen sind nur für den Zeitraum gültig, in dem der Test durchgeführt wurde. Das Sicherheitsniveau kann sich schnell ändern, und nach Abschluss des Tests können neue Schwachstellen auftreten.
- **Retests und kontinuierliche Verbesserung:** Retests werden ausdrücklich empfohlen, da sie zusätzliche Schwachstellen aufdecken können, die in der Erstbewertung nicht erkannt wurden oder zukünftig auftreten. Sicherheit ist ein kontinuierlicher Prozess, und regelmäßige Tests sind entscheidend, um ein starkes Sicherheitsniveau aufrechtzuerhalten.
- **Schutzsysteme können die Ergebnisse beeinflussen:** Aktive Schutzsysteme (z. B. Intrusion-Detection-Systeme, Intrusion-Prevention-Systeme, Firewalls usw.) können die Testergebnisse beeinflussen. Diese Systeme können den Testprozess beeinträchtigen und möglicherweise zu unvollständigen oder irreführenden Findings führen.
- **Falschpositive:** Im Rahmen der Bewertung können Falschpositive auftreten – Fälle, in denen Schwachstellen identifiziert werden, sich jedoch später als nicht ausnutzbar oder nicht vorhanden herausstellen. In unserer Praxis teilen wir alle gesammelten Informationen, einschließlich potenzieller Falschpositive, da sie dennoch hilfreiche Erkenntnisse liefern können. Prominente Beispiele sind Findings, die auf gemeldeten Versionsnummern basieren, bei denen Sicherheitskorrekturen zurückportiert wurden, oder Anwendungen, die zwar verwundbare Bibliotheken verwenden, jedoch nicht deren verwundbare Komponenten nutzen.

Die bereitgestellten Ergebnisse und Empfehlungen basieren auf dem Verständnis und dem Umfang der Prüfung und sollten als Teil eines umfassenderen, kontinuierlichen Sicherheitsverbesserungsprozesses verwendet werden.

Abbildungsverzeichnis

A Anhang

A.1 Zusätzliche Dateien zum Bericht

Folgende Dateien werden separat zum Bericht zur Verfügung gestellt

- Aufbereitete Ergebnisse des Port-Scanners *nmap*
  - juice-shop.lab.html
  - juice-shop.lab.nmap
  - juice-shop.lab.gnmap
  - juice-shop.lab.xml
- Ergebnisse des Schwachstellen-Scanners *nuclei*
  - nuclie-juice-shop.json