Übung Web-App

Dozent: Roy Meissner roy.meissner@uni-leipzig.de

Story zur Anwendung

Sie wollen eine öffentliche Chat-Anwendung für Ihren Verein anbieten, ähnlich einer Chat-Gruppe. Die Anwendung soll im Browser laufen.

Alle Nutzer*innen werden einen Feed der letzten Nachrichten sehen und können selbst neue Nachrichten senden.

Ihr Verein ist klein/unwichtig und daher kein interessantes Angriffsziel. Nur Personen in Ihrem Verein kennen die Adresse der Anwendung.

„Geheime“ Themen (bspw. monetär) werden nur bei Vereinssitzungen, mit physischer Anwesenheit besprochen.

Vorbereitung

1. Klonen Sie sich die beiden Git-Repositories:
   - https://gitlab.dit.htwk-leipzig.de/roy.meissner/chat-app-ui
   - https://gitlab.dit.htwk-leipzig.de/roy.meissner/chat-app-service
2. Stellen Sie sicher, dass sie docker und docker-compose nutzen können
3. Wechseln Sie in eines der Repositories
4. Führen Sie aus:
   1. docker-compose pull
   2. docker-compose up -d
5. Stellen Sie sicher, dass sie http://localhost:8081 aufrufen können
6. Testen Sie die App kurz aus (simulieren Sie mehrere Nutzer*innen mit mehreren Browser-Tabs)
7. Rufen Sie das BSI Kompendium auf und navigieren Sie zu „APP:Anwendungen“

Analyse der Anwendung

- Aus welchen Teilen könnte die Anwendung potentiell bestehen?
- Welche Punkte unter APP des BSI Kompendium müssten Sie damit beachten?
- Welche Gefährdungslagen unter APP3.1 und APP3.2 treffen auf diese Anwendung zu?
  - Analysieren Sie dazu die Ausgabe von `docker-compose logs -f`
  - Rufen Sie im Browser die Entwicklerkonsole (Netzwerk-Tab) auf und versuchen Sie herauszufinden, welche Frameworks genutzt werden (ggf. incl. Version)
  - Könnte die Anwendung automatisiert genutzt werden? Bspw. von einem Bot?
  - Können Sie durch das Angebot der Anwendung gegen Gesetze verstoßen?
- Wie relevant sind ... für diese Anwendung?
  - APP3.1, Punkt 2.4
  - APP3.2, Punkt 2.1
  - APP3.2, Punkt 2.3
  - APP3.2, Punkt 2.4
  - APP3.2, Punkt 2.5

Annahmen

- Rufen Sie sich die Story zur App in Erinnerung
- Welche Annahmen wurden dort getroffen?
- Wie realistisch sind diese Annahmen?

- Warum sollten Sie eine Authentisierung einführen und sich damit entsprechende Sicherheitsrisiken ins Portfolio holen?
  - Bietet die Anwendung weiter Missbrauchspotential? Welche Rollen kommen dafür in Frage?
  - Könnte die Anwendung weiter automatisiert genutzt werden? Wenn ja, wie verhindern Sie dies?

- Reichen die diskutierten Punkte aus, um Missbrauch zu vermeiden?

Konzeptionelle Verbesserung

- Was müsste getan werden, um den Dienst abzusichern?

- Mögliche Diskussionspunkte:
  - Accounts & Passwörter – Passwortrichtlinien? Second Factor? Mehrere Nutzer*innen, ein Account?
  - Verhinderung automatisierter Nutzung – Könnte dies auch valide Nutzung beeinträchtigen?
  - Gezielte Ermöglichung automatisierter Nutzung – RSS Feeds, Backup, ...
  - Verschlüsselte Kommunikation – TLS/HTTPS zw. Client/Server? Ende-zu-Ende Verschlüsselung?
  - Erkennung von illegalen Inhalten – „Uploadfilter“? Kann man diese missbrauchen? Ist das Zensur?
  - Server-Protokolle – verraten diese etwas? Welche Informationen sind nötig, welche nicht?

- Womit unterstützt Sie das BSI Kompendium?

Einfache Verbesserungen

- Sehen sich die Optionen unter https://www.npmjs.com/package/http-server#user-content-available-options an. Welche Punkte könnten Sie ad hoc nutzen?
- Stoppen Sie den frontend und starten Sie diesen manuell, über
  - „docker run –it —rm –p 8081:8081 gitlab.dit.htwk-leipzig.de:5050/roy.meissner/chat-app-ui:latest /bin/sh“ – siehe docker-compose.yaml
  - Führen Sie auf der Shell „http-server /nodeApp/dist -p 8081“ aus und rufen Sie den Frontend im Browser auf
  - Führen Sie die gefundene Option ein
- Erstellen Sie sich ein eigenes TLS-Zertifikat (HTTPS). Im Container:
  - openssl req -newkey rsa:4096 -x509 -sha512 -days 365 -nodes -out certificate.pem -keyout privatekey.pem
  - Nutzen Sie das Zertifikat über die beschriebenen Optionen (—ssl –C –K)

- Optionen:
  - rsa:4096 – Schlüssel Algorithmus (asymmetrische Verschlüsselung)
  - x509 – Standard für Zertifikate
  - sha512 – Hash Algorithmus
  - days - Gültigkeitsdauer
  - nodes – privaten Schlüssel nicht verschlüssen
- Sehen Sie sich das Zertifikat an
  - openssl x509 -noout -in certificate.pem -text
- Extrahieren Sie den Public-Key
  - openssl x509 -pubkey -noout -in certificate.pem