AUTHENTICATION & AUTHORIZATION

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

AUTHENTICATION

- Identität eines Benutzers feststellen

- Beispiel:
- Anmeldung mit Benutzername und Passwort
- Fingerabdruck oder Gesichtserkennung auf Mobilgeräten
- Zwei-Faktor-Authentifizierung (2FA), z. B. SMS-Codes oder Authenticator-Apps

AUTHORIZATION

- Zugriffsrechte prüfen

- Beispiel:
- Ein „User“ darf sein eigenes Profil bearbeiten, aber nicht das eines anderen Nutzers
- Ein „Admin“ darf Benutzer löschen oder hinzufügen

PASSWORT-HASHES

- Warum hashen?
- Klartext-Passwörter sind anfällig für Datenlecks

- Hash-Verfahren
- BCrypt: Geeignet für Passwörter, da es zeitaufwändig ist (Schutz vor Brute-Force-Angriffen)
- Argon2: Gewinner des Hash-Wettbewerbs 2015. Sehr sicher
- PBKDF2 (Password-Based Key Derivation Function 2) : Besonders resistent gegenüber Brute-Force und Rainbow Table Angriffen ist

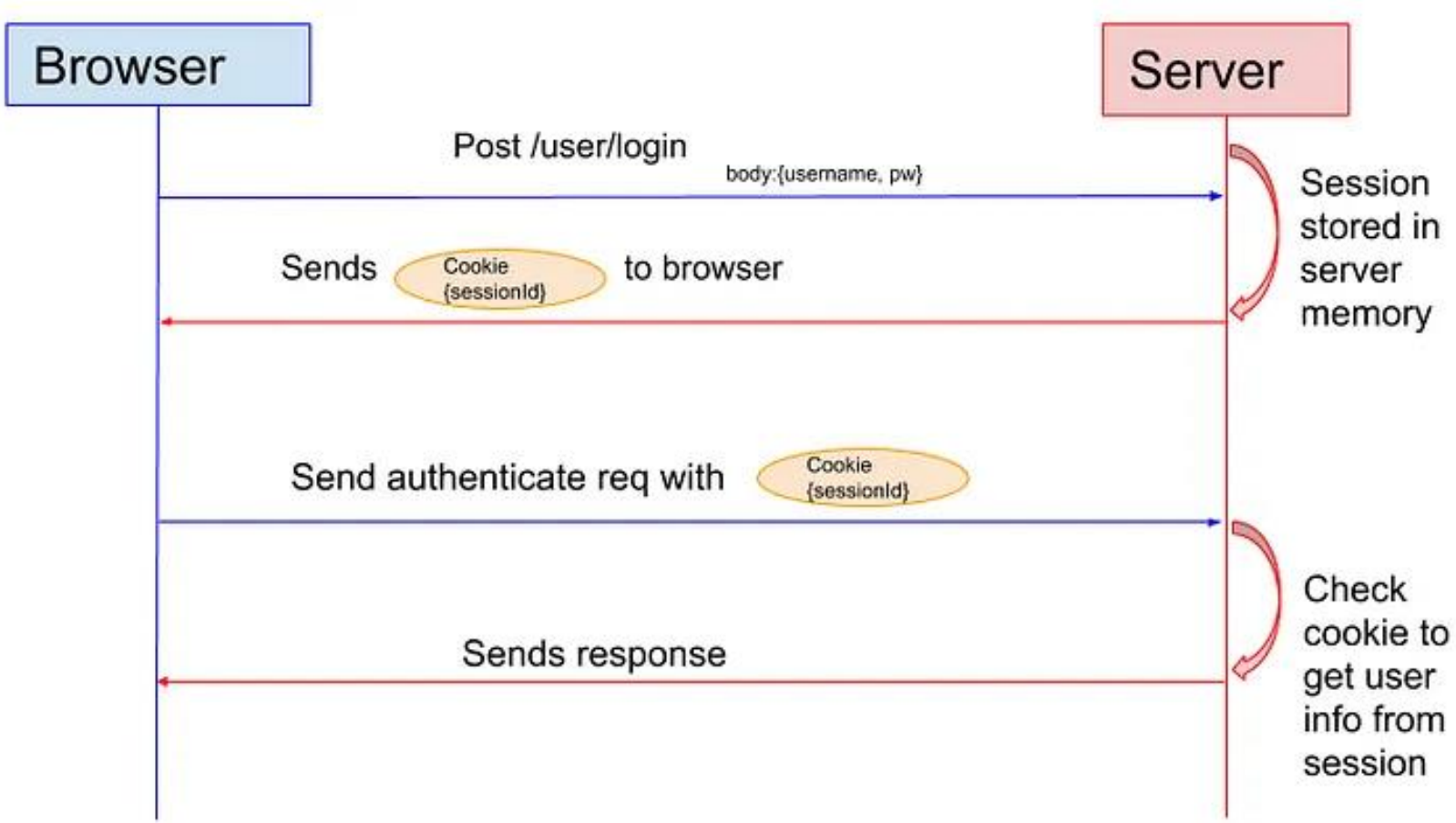
SESSION-BASIERTE AUTHENTIFIZIERUNG

- Der Server erstellt nach erfolgreichem Login eine Session-ID
- Die Session-ID wird im Browser als Cookie gespeichert oder als Token zurückgegeben
- Bei jedem weiteren Request sendet der Browser das Cookie / die App den Token mit

- Zentralisierte Kontrolle: Der Server verwaltet alle Sitzungen

- Skalierungsprobleme bei vielen Benutzern

SESSION-BASIERTE AUTHENTIFIZIERUNG



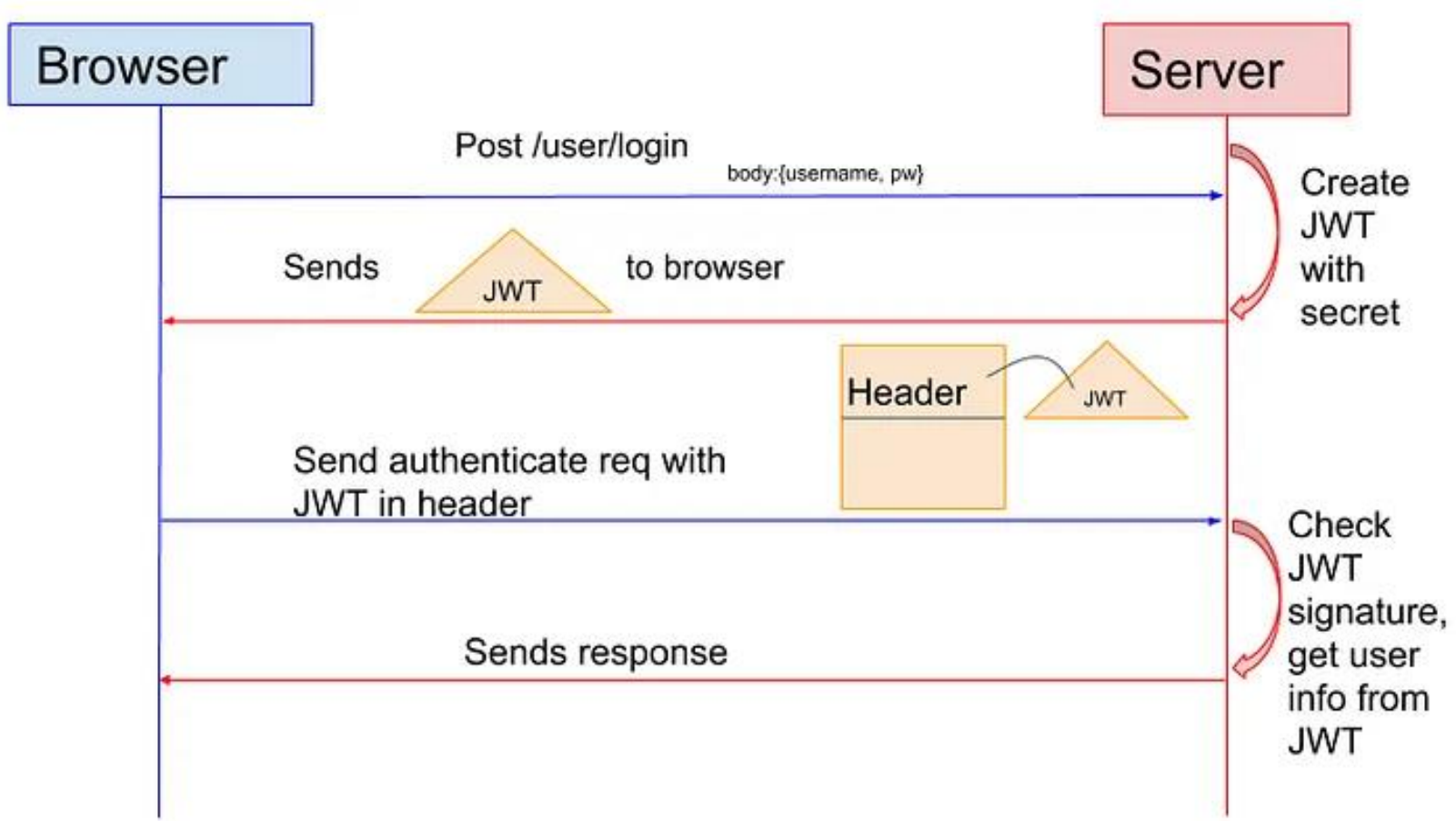
TOKEN-BASIERTE AUTHENTIFIZIERUNG

- Der Server stellt nach erfolgreichem Login ein Token (z. B. JWT) aus
- Das Token wird auf dem Client gespeichert
- Bei jeder Anfrage sendet der Client das Token mit

- Server muss keine Sitzungsdaten speichern
- Tokens können dezentral überprüft werden

- Sicherheitsrisiken bei unsicherer Speicherung (z. B. XSS im Browser Local Storage)
- Schwierige Token-Invalidierung bei Logout

TOKEN-BASIERTE AUTHENTIFIZIERUNG



JWT (JSON WEB TOKENS)

- Ein JWT ist ein kompakter, URL-sicherer Token-Standard, der Informationen zwischen zwei Parteien als JSON-Objekt kodiert und optional signiert oder verschlüsselt überträgt
- Ein JWT besteht aus drei Teilen, die durch Punkte (.) getrennt sind

- Header: Enthält die Informationen über den Typ des Tokens und den verwendeten Signaturalgorithmus
- Payload: Enthält die Claims, also die Daten, die im Token übertragen werden
- Signature: Stellt sicher, dass der Token nicht manipuliert wurde
- Wird erstellt, indem der Header, die Payload und ein geheimer Schlüssel durch z.B. den Algorithmus (z. B. HMAC SHA-256) verarbeitet werden

BROWSER-BASIERTE ANWENDUNGEN

- Häufige Speicherung von Authentifizierungsdaten in Cookies
- Ein Benutzer meldet sich bei einer Website an, der Server sendet eine Session-ID im Cookie zurück

- Sicherheitsaspekte:
- **HttpOnly-Cookies:** Nur serverseitig zugänglich, nicht durch JavaScript lesbar (Schutz vor XSS)
- **Secure-Flag:** Cookie wird nur über HTTPS übertragen
- **SameSite-Attribute:** Schutz vor CSRF-Angriffen
- **CSRF-Tokens:** Schutz vor CSRF-Angriffen

DESKTOP & MOBILE APPS

- Verwendung von Tokens (z. B. JWTs), die in einem sicheren Speicher wie der Secure Enclave oder dem Android Keystore abgelegt werden

- Beispiel:
- Eine mobile App speichert einen Bearer-Token und sendet diesen bei API-Aufrufen im Authorization-Header mit

OAUTH 2.0 UND OPENID CONNECT (OIDC)

- OAuth 2.0 (Autorisierung)
- Ermöglicht es Drittanbieter-Apps, auf Ressourcen zuzugreifen, ohne Passwörter zu teilen

- Beispiel:
- Login mit Google

- OpenID Connect (OIDC) (Authentifizierung)
- Baut auf OAuth 2.0 auf und fügt eine Authentifizierungsschicht hinzu

OAUTH 2.0 UND OPENID CONNECT (OIDC)