Single Sign On bei Webanwendungen

Daniel Ebert
Sebastian Scherer

27. Oktober 2020

1 Grundlagen

Dem Titel nach beschäftigt sich unsere Arbeit mit Single Sign On (SSO) für Webanwendungen. Ziel des SSO-Mechanismus ist es, dass sich Benutzer nur einmal unter Zuhilfenahme eines einzigen Authentifizierungsverfahrens identifizieren muss. Danach übernimmt der SSO-Mechanismus die Aufgabe, den Anwender zu authentifizieren und die erkannte Identität zu bestätigen. Dies hat den Vorteil, dass sich der Benutzer sich nur einmal identifizieren muss und seine Identität an weitere Systeme weitergegeben werden kann, ohne dass sich dieser erneut anmelden muss.

SSO ist somit für die Sicherheit eine Methode, um die Psychological Acceptability der Benutzer zu erhöhen, da sie nicht dazu gezwungen werden sich jedes Mal manuell zu authentifizieren.

Die folgenden Untersektionen beinhalten sogleich unsere vorläufige Gliederung.

1.1 Einführung

Einleitendes Kapitel in das Thema SSO, welches die Definition des Mechanismus, die Einordnung des Themas, die grundsätzliche Problemstellung und -abgrenzung, die Use-Cases für SSO bei Webanwendungen, das Ziel der Arbeit und unser Vorgehen enthält.

1.2 Aufbau von SSO

Es gibt mehrere Möglichkeiten SSO umzusetzen. Da der Umfang der Ausarbeitung 20 Seiten betragen soll, ist es nicht möglich alle SSO-Architekturen vorzustellen. Unsere praktische Umsetzung basiert auf Keycloak und dem OpenID Connect (OIDC) Standardprotokoll für Autorisierung und Authentifizierung und daher werden wir diese Architektur vorstellen. Wir haben uns für OIDC entschieden, da dieses speziell für Webanwendungen entwickelt worden ist und außerdem das modernere Protokoll ist. [13]

User sind Entitäten, die in der Lage sind, sich in das Keycloak System einzuloggen. Clients sind Entitäten, die Keycloak zur Authentifizierung eines Benutzers beauftragen können.

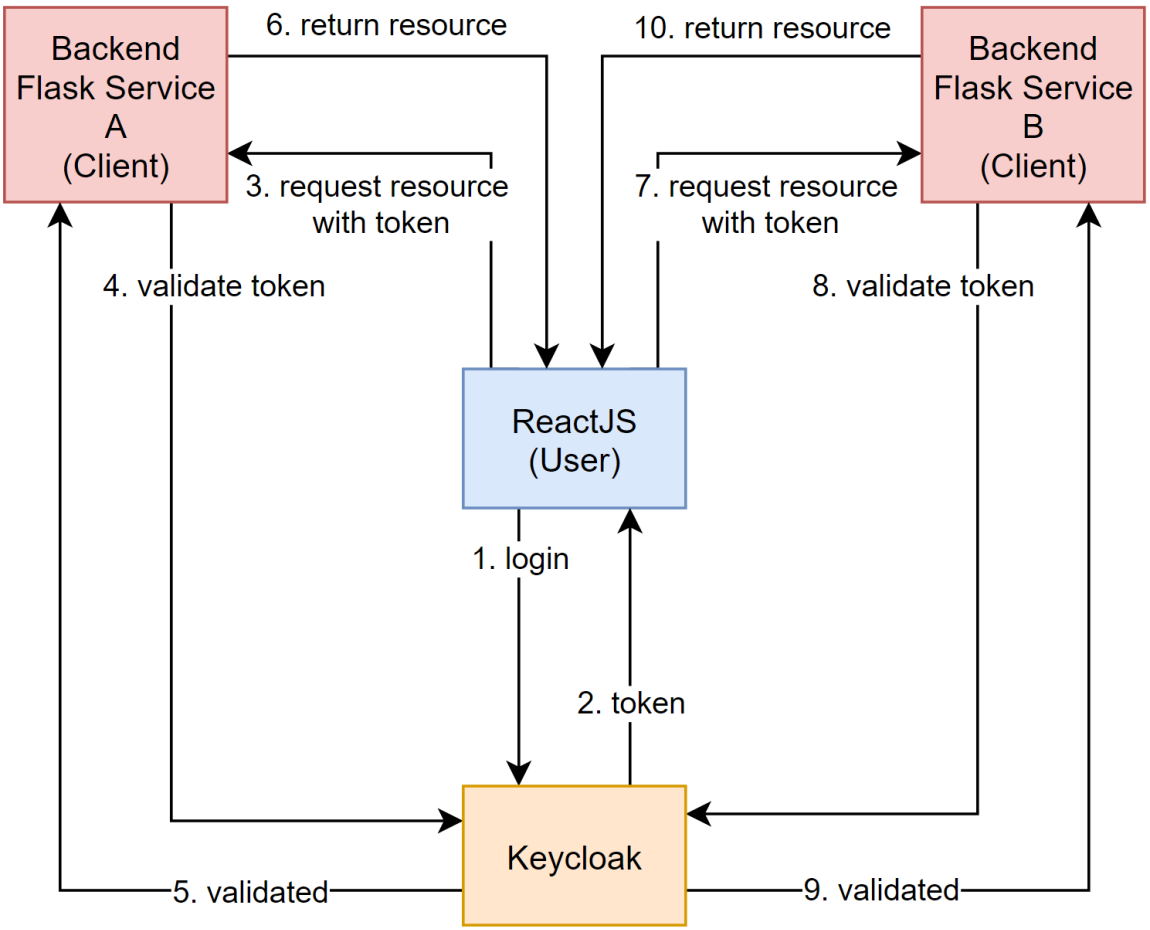

Abbildung 1: Beispiel Ablauf und Infrastruktur mit Keycloak

Clients stellen Ressourcen zur Verfügung, deren Zugriff auf die authentifizierten und autorisierten User beschränkt ist. In Keycloak erhalten User nach einem erfolgreichen Log-In ein Token. User können sich mit diesem Token bei mehreren Clients authentifizieren, ohne sich erneut z.B. mit E-Mail und Passwort einloggen zu müssen. Tokens sind von Keycloak signiert und können deshalb auf ihre Echtheit überprüft werden.

1.3 SSO-Threat Model

In dieser Sektion wollen wir die gängigen Schwachstellen und die passenden Gegenmaßnahmen erläutern. Die verbreitetsten Schwachstellen sind:

- Complete Mediation [5]
- Brute Force Angriffe auf Passwörter [5]
- Clickjacking [5]
- SSL/HTTPS Anforderungen [5]
- CSRF Angriffe [5]

- Kompromittierte Zugangstoken [5]
- Open Redirectors [5]

1.4 Umsetzung von SSO innerhalb einer Beispielwebanwendung

SSO soll an einer Beispielwebanwendung mit Keycloak gezeigt werden. Keycloak ist eine Open-Source-Lösung für Identity und Access Management (IAM), die auf die Entwicklung moderner Anwendungen und Services ausgerichtet ist. Die Community-Version von Red Hat macht es einfach, die Authentifizierung an Anwendungen und Diensten sicher zu implementieren.

Die Umsetzung basiert auf der in Figure 1 gezeigten Architektur. Im praktischen Teil setzen wir einen Keycloak Server im Standalone Mode auf. Wir erstellen eine Single Page Application mit ReactJS, bei der sich User in Keycloak registrieren und einloggen können und Ressourcen von zwei Backend Systemen über eine HTTP REST API abrufen können. Die Backend Systeme sind in Python geschrieben und verwenden Flask. Diese Backend und Frontend Systeme werden in Keycloak angebunden und integriert. Dabei fokussieren wir uns auf die SSO Aspekte von Keycloak wie User Registration, User Login, User Logout und Session Management. Keycloak Features die weniger mit SSO zu tun haben, wie z.B. Identity Brokering, Rollen und Gruppen, externe Datenbanken und Log Events werden nicht näher bearbeitet. Der Keycloak Server, als auch die Backend und Frontend Systeme laufen in getrennten Docker Containern. Wir verwenden Docker-Compose um die Container hinsichtlich z.B. des Netzwerks oder des persistenten Speichers hin zu konfigurieren.

Literatur

[1] Tayibia Bazaz und Aqeel Khalique. „A Review on Single Sign on Enabling Technologies and Protocols“. In: *International Journal of Computer Applications* 151 (Okt. 2016), S. 18–25. DOI: 10.5120/ijca2016911938.

[2] Kavindu Dodanduwa und Ishara Kaluthanthri. *Role of Trust in OAuth 2.0 and OpenID Connect*. 2018. arXiv: 1808.10624 [cs.CR].

[3] D. Fett, R. Küsters und G. Schmitz. „The Web SSO Standard OpenID Connect: In-depth Formal Security Analysis and Security Guidelines“. In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. 2017, S. 189–202. DOI: 10.1109/CSF.2017.20.

[4] Daniel Fett, Ralf Kuesters und Guido Schmitz. *A Comprehensive Formal Security Analysis of OAuth 2.0*. 2016. arXiv: 1601.01229 [cs.CR].

[5] T. Lodderstedt, M. McGloin und P. Hunt. *OAuth 2.0 Threat Model and Security Considerations*. RFC 6819. RFC Editor, Jan. 2013, S. 1–71. URL: https://www.rfc-editor.org/rfc/rfc6819.txt.

[6] C. Mainka u. a. „SoK: Single Sign-On Security — An Evaluation of OpenID Connect“. In: *2017 IEEE European Symposium on Security and Privacy (EuroS P)*. 2017, S. 251–266. DOI: 10.1109/EuroSP.2017.32.

[7] Vladislav Mladenov, Christian Mainka und Jörg Schwenk. *On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect*. 2016. arXiv: 1508.04324 [cs.CR].

[8] Nick Heijmink. *Secure Single Sign-On A comparison of protocols*. [Online; accessed 27-October-2020]. URL: https://www.ru.nl/publish/pages/769526/z_researchpaper_sso_final_nick_heijmink_s4250559.pdf.

[9] Dr. Yashpal Singh Parul Garg. „SSO (Single Sign On) Implementation“. In: *International Journal of Science and Research (IJSR)*. 2016, S. 988–990.

[10] Paul Madsen. *OpenID Connect 1.0 for Enterprise*. [Online; accessed 27-October-2020]. URL: https://www.pingidentity.com/content/dam/pic/downloads/resources/white-papers/en/openid-connect-white-paper.pdf?id=b6322a80-f285-11e3-ac10-0800200c9a66.

[11] Vedala Radha und D. Reddy. „A Survey on Single Sign-On Techniques“. In: *Procedia Technology* 4 (Dez. 2012), S. 134–139. DOI: 10.1016/j.protcy.2012.05.019.

[12] Red Hat, Inc. *Keycloak Documentation*. [Online; accessed 27-October-2020]. URL: https://www.keycloak.org/documentation.html.

[13] Red Hat, Inc. *OpenID Connect vs. SAML*. [Online; accessed 27-October-2020]. URL: https://www.keycloak.org/docs/latest/securing_apps/#openid-connect-vs-saml.

[14] Justin Richer und Antonio Sanso. *OAuth 2 in Action*. 1. Aufl. Shelter Island, NY 11964: Manning Publications, 2017.