Single Sign-On (SSO) Implementation Using Keycloak, RADIUS, LDAP, and PacketFence for Network Access

by Perpustakaan Referensi

Submission date: 17-Jan-2025 01:32PM (UTC+0700)
Submission ID: 2565800057
File name: JurnalTeknika-Justinus_-_Justinus_Andjarwirawan.docx (101.2K)
Word count: 3504
Character count: 20721

Single Sign-On (SSO) Implementation Using Keycloak, RADIUS, LDAP, and PacketFence for Network Access

Justinus Andjarwirawan[1*]

[1*]Program Studi Informatika, Universitas Kristen Petra, Surabaya, Jawa Timur
Email: [1*]justin@petra.ac.id

(Naskah masuk: dd mmm yyyy, direvisi: dd mmm yyyy, diterima: dd mmm yyyy)

Abstract

The increasing demand for secure, seamless authentication mechanisms in public and private networks has fueled the need for more robust network access control (NAC) systems, as well as Single Sign-On (SSO) which is critical for organizations that require seamless and secure access across different platforms. This paper explores SSO in a fully open source implementations with Keycloak, RADIUS and LDAP; extending to captive portal implementations with PacketFence for Wi-Fi authentication. Specifically, this paper highlights the integration of PacketFence with FreeRADIUS for captive portal authentication, leveraging Keycloak for identity management and providing users with secure Wi-Fi access. Real-world examples, such as authenticating campus network users over Wi-Fi with 802.1X and captive portals, illustrate how these systems work in tandem to provide scalable and secure network access control.

Keywords: SSO, RADIUS, LDAP, OAuth 2.0, NAC..

I. INTRODUCTION

With the increasing demand for secure network access in educational institutions and large enterprises, there is a need for unified, scalable, and secure authentication mechanisms. Traditional authentication solutions often involve cumbersome, decentralized systems, leading to security risks and poor user experiences. Combining technologies like Keycloak, FreeRADIUS, and PacketFence provides a powerful, integrated solution for secure SSO and network access control.

Keycloak handles identity management, while FreeRADIUS, and LDAP provides a scalable infrastructure for managing users across multiple platforms. PacketFence adds a layer of Network Access Control (NAC) by providing a captive portal for Wi-Fi users, giving administrators fine-grained control over network access.

II. RELATED WORK

Several studies have explored the integration of NAC systems and IAM solutions to provide more efficient authentication methods. In [1], research on the growing importance of identity federation and SSO frameworks for enterprise networks is emphasized, particularly as organizations expand their networks and adopt cloud-based infrastructures. Another study [2] highlights the role of NAC systems like PacketFence in managing wireless user access through dynamic VLAN assignment, endpoint security, and guest access.

Keycloak's capability to manage user identities and provide OAuth2/OpenID Connect support for SSO authentication has been well-documented in [3], where the advantages of its role in providing centralized identity management are discussed. Previous research has focused on integrating Keycloak with cloud platforms, but its application in on-premise WiFi access scenarios, as demonstrated in this journal, remains under-explored.

III. IMPLEMENTATION DESIGN

The scenario of this implementation involves Wi-Fi users connecting to a campus network with an authentication system running behind a captive portal as a Network Access Control, and an identity system to manage user roles and access rights to the institution's information system as well as a Single Sign-On to provide sessions for all applications that require an authentication. Behind the process, the authentication goes both to the captive portal and Keycloak, where PacketFence forwards to, providing both authentication and Single Sign-On. Optionally, PacketFence may authenticate directly to FreeRadius but will not apply a Single Sign-On. The deployment is shown in Image 1.

The architecture of the whole system is described as follows:

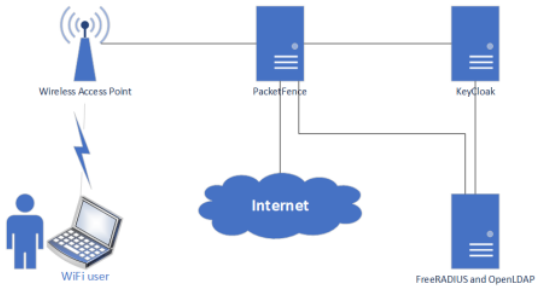

Image 1. Implementation Design

A. PacketFence

PacketFence[4] is a fully-featured, open-source Network Access Control (NAC) solution that provides a captive portal for managing and securing network access, including Wi-Fi and wired networks. It integrates seamlessly with authentication services such as FreeRADIUS and LDAP, allowing it to authenticate users via SSO or other mechanisms like 802.1x[4]. PacketFence is commonly deployed in educational institutions, campuses, and enterprises where guest and user network access needs to be tightly controlled.

PacketFence features:

- Captive Portal Authentication: PacketFence offers customizable captive portals for Wi-Fi users, forcing them to authenticate before gaining network access.

- RADIUS and LDAP Integration: It integrates with RADIUS and LDAP for centralized user authentication.

- Guest and BYOD Management: PacketFence supports Bring Your Own Device (BYOD) policies and allows guest users to authenticate via a web portal.

- Compliance and Auditing: It provides mechanisms for auditing network access, ensuring compliance with security policies.

PacketFence can also be used as LAN users authentication portal, as PacketFence may integrate with port-based authentication 802.1x.

B. RADIUS

In this scenario, a Wi-Fi captive portal is delivered using PacketFence integrated with FreeRADIUS[5] for the RADIUS, LDAP with OpenLDAP (slapd), and Keycloak. This setup allows campus users to authenticate via the captive portal before gaining network access. RADIUS and LDAP may use plain text to authenticate which can lead to a security issue. This scenario may somehow still be implemented because of legacy protocols and hashing algorithms[8], as well as in a migration from a legacy PAM based authentication[9]. LDAP stores users' passwords in its own format, therefore LDAP checks the input password in plain text. Keycloak serves as the SSO provider to serve sign-in session for general applications which are mostly web-based, while FreeRADIUS acts as the backend for authenticating users against LDAP.

C. Keycloak

1. Keycloak[6] is an open-source identity and access management solution supporting multiple authentication protocols (OAuth2, OpenID Connect, SAML) and providing SSO across various services, deployed on a Linux virtual machine (Ubuntu 24.04 LTS) in a containerized environment (Docker). Once installed, Keycloak is configured with a new realm for managing users and applications.

2. User Federation: In the Keycloak admin console, LDAP is configured as the user federation provider. This will allow Keycloak to authenticate users against an LDAP backend, which is OpenLDAP. Keycloak by itself may not need an LDAP backend because Keycloak can provide users locally.

3. Creating an OAuth2 client for PacketFence: a client is created in Keycloak for PacketFence, enabling OAuth2 as the authentication method. PacketFence will use this client to authenticate Wi-Fi users through the captive portal.

4. SSO Integration: Keycloak is configured for SSO across network services. For web applications and network access, users will authenticate once using their Keycloak credentials.

D. Packetfence configuration

Once PacketFence is installed, it is configured with the network interfaces for captive portal services. At least two network interfaces are required: one for management and one for providing access to clients through the captive portal.

E. Captive portal configuration

1. Enabling the Captive Portal: In the PacketFence web interface, the captive portal for the Wi-Fi interface (eth1 in this case) is enabled. Users connecting to the Wi-Fi network will be redirected to this portal for authentication.

2. Configuring FreeRADIUS in PacketFence: PacketFence uses FreeRADIUS to authenticate users via the captive portal. The RADIUS server is configured to communicate with an LDAP backend for authenticating users.

3. OAuth2 integration with Keycloak: PacketFence is configured to use Keycloak as the OAuth2 provider.

4. Redirecting Wi-Fi users to Captive Portal: all Wi-Fi users are redirected to the PacketFence captive portal upon connecting to the Wi-Fi network. This can be enforced by configuring the wireless access points (APs) to use PacketFence as the authentication source.

F. OAuth 2.0

OAuth, short for "Open Authorization," is a widely adopted standard for secure authorization in modern applications, enabling users to grant third-party applications access to their resources without sharing their credentials. Traditional authentication mechanisms that require users to provide credentials to third-party services pose significant security risks, such as password leakage and identity theft. OAuth addresses these challenges by providing a secure, token-based authorization framework that separates the process of granting access from user authentication.

OAuth allows users to approve one application to interact with another on their behalf without directly exposing their credentials. For example, a user might use their Google account to log into a third-party app, granting that app limited access to their Google Drive data without sharing their Google password.

OAuth has evolved over time, with the most recent version being OAuth 2.0, which offers improved simplicity and flexibility and it is the most widely used authorization framework today.

OAuth operates based on a set of key components, which together enable secure authorization:

- Resource Owner (User): The individual or entity that owns the data or resource being accessed. In most cases, the user of a service (e.g., an app or a website) is the resource owner.
- Client (Third-Party Application): The application that requests access to the resource owner's data. This is often a third-party service or app that the user interacts with, such as a mobile app requesting access to the user's photos stored in a cloud service. In this case PacketFence the third-party app to the Keycloak's OAuth 2.0.
- Authorization Server: The server that authenticates the source owner and issues access tokens to the client. It verifies the user's identity and ensures that the client has the correct permissions to access the resource.
- Resource Server: The server hosting the protected resources (e.g., user data). It validates the access token issued by the authorization server before granting the client access to the resources.
- Access Token: A credential issued by the authorization server that the client uses to access the resource server. The token is typically short-lived and is passed between the client and the resource server for each request.

The process of OAuth authorization involves a series of steps known as the "OAuth flow." OAuth 2.0 defines several grant types, or authorization flows, that outline how different kinds of clients (web apps, mobile apps, etc.) can interact with the authorization and resource servers. The most common flows include:

Authorization Code Grant Flow.

The authorization code flow is the most secure and widely used OAuth flow, particularly for web applications. The process involves the following steps:

- Authorization Request: The client (third-party app) redirects the user to the authorization server with a request for access to specific resources. The client also provides a redirect URI, where the user will be sent back after authentication.
- User Authentication: The user logs in to the authorization server (if not already logged in) and consents to granting the requested access to the client.
- Authorization Code Issued: Once the user consents, the authorization server redirects the user back to the client with an authorization code. This code is short-lived and can only be used once.
- Exchange Authorization Code for Token: The client sends the authorization code to the authorization server's token endpoint, along with its own credentials (e.g., client and client secret), to obtain an access token.
- Access Token Issued: The authorization server validates the authorization code and issues an access token to the client.
- Access Resource: The client uses the access token to request the desired resources from the resource server. If the access token is valid, the resource server grants access to the requested resource.

Implicit Grant Flow

The implicit grant flow is typically used by single-page applications (SPAs) or mobile apps, where the client cannot securely store sensitive information like a client secret. The steps are similar to the authorization code flow but with a key difference: instead of an authorization code, the access token is issued directly after the user authenticates. Risk is the main drawback of the implicit flow is that the access token is exposed in the browser's address bar, making it more vulnerable to interception.

Client Credentials Grant Flow

The client credentials flow is used for machine-to-machine (M2M) communication, where the client is a trusted entity that does not involve user authentication (e.g., a backend service accessing another service). The client authenticates directly with the authorization server using its credentials (client ID and secret) and receives an access token.

Resource Owner Password Credentials Flow

In this flow, the user provides their credentials (username and password) directly to the client. The client then exchanges these credentials for an access token from the authorization server. This flow is considered insecure because it requires the user to trust the client with their credentials. It is only recommended for use in scenarios where the client is highly trusted (e.g., a company's own mobile app).

An access token is the credential that the client uses to access resources on behalf of the user. Tokens are typically short-lived and can have a variety of formats, such as JWT (JSON Web Token). The token includes information like the scope of access, expiration time, and resource owner information.

A refresh token is a long-lived token that can be used by the client to obtain a new access token after the current one expires. Refresh tokens are often used in the authorization code flow to maintain continuous access without requiring the user to log in again.

OAuth enables Single Sign-On (SSO) functionality by allowing users to authenticate once with an identity provider (e.g., Google, Facebook, FreeRadius with LDAP) and then use that authentication to access multiple services without logging in again.

G. LDAP Configuration

LDAP installation and configuration are straightforward as it is a directory service which holds users' identities and their organization units along with access rights information which is mandatory for an institution to operate.

The objecClass to use in the scenario are account and posixAccount because many attributes in these objectClasses can be used to store access rights information, as well as userPassword for the authentication which later will be accessed by FreeRADIUS or KeyCloak. LDAP server is provided whereas KeyCloak may use its own users, because the LDAP itself can be used by other applications. In this case LDAP can be assigned to centralized users' credentials. New users, modifications and deletions only to be operated from LDAP, and the rest of the systems will follow.

The LDIF format for an LDAP user is as followed:

```
dn: uid=<user>,dc=example,dc=com
uid: <user>
cn: Full Name
objectClass: account
objectClass: posixAccount
uidNumber:
gidNumber:
homeDirectory:
```

H. FreeRADIUS Configuration

Integrate FreeRADIUS with OpenLDAP

1. Configuring RADIUS for LDAP Authentication: LDAP module in RADIUS must be enabled in order to authenticate to LDAP instead of RADIUS directly to its default authentication method, which is PAM or Linux's system users.

```
ldap {
   server = "ldap://ldap.example.com"
   identity = "uid=admin,cn=users,
       cn=accounts,dc=example,dc=com"
   password = "admin_password"
   base_dn = "dc=example,dc=com"
}
```

2. EAP and 802.1x Setup: Enabling EAP (Extensible Authentication Protocol) in FreeRADIUS to handle 802.1X requests from PacketFence. The `eap.conf` file is configured to support PEAP or EAP-TTLS, which is commonly used for Wi-Fi authentication.

```
eap {
  default_eap_type = peap
  peap {
tls_certfile = /etc/ssl/certs/radius.crt
tls_keyfile  = /etc/ssl/private/radius.key
tls_ca_certfile = /etc/ssl/certs/ca.pem
  }
}
```

The 802.1x can also be used in switch devices to authenticate LAN users, using the same backend as the Wi-Fi access points, providing a single authentication system.

3. Linking FreeRADIUS with PacketFence: FreeRADIUS is set up to accept authentication requests from PacketFence. Add the PacketFence IP address and shared secret to the `clients.conf` file in FreeRADIUS.

```
client packetfence {
    ipaddr = 192.168.1.10
    secret = shared_secret
}
```

IV. AUTHENTICATION TEST

Once everything is configured, the captive portal is tested by connecting a Wi-Fi client to the network:

1. Connecting to the Wi-Fi Network: The user connects to the campus Wi-Fi. The wireless access point forwards the authentication request to PacketFence. Any other application may authenticate to KeyCloak to gain access sessions for applications as KeyCloak authenticate to LDAP as well.

2. Captive Portal Redirection: The user is redirected 8 the PacketFence captive portal, where they are prompted to log in with a username and password. The gateway or router must be able to redirect packet to PacketFence as PacketFence may not always placed as a gateway.

3. OAuth2 Authentication with Keycloak: normally a user is redirected to the Keycloak login page. After logging in, Keycloak issues an OAuth2 token, which is verified by PacketFence. But in this case a user only needs to authenticate once through PacketFence's captive portal. To establish communication between PacketFence and Keycloak:

- OAuth2 Plugin in PacketFence: PacketFence supports OAuth2-based authentication. The captive portal was configured to use Keycloak as the OAuth2 provider.
- Token Exchange: After users authenticate via Keycloak, the system provides an access token, which PacketFence uses to grant access to the network.
- Session Management: Once the user is authenticated via SSO, the session is managed through PacketFence, ensuring that the user stays connected to the network without the need for repeated authentication.

4. FreeRADIUS Authentication: PacketFence forwards the authentication request to FreeRADIUS, which queries LDAP

to authenticate the user. This scenario is used when there is no need of an SSO, only as a means to authenticate.

5. Network Access Granted: Once authenticated, the user is granted access to the Wi-Fi network. When using the KeyCloaks portal, besides granted to the network the user also gains SSO sessions for all applications.

During the implementation, several challenges were encountered:

• Session Timeout: users occasionally experienced unexpected logouts due to short-lived sessions in Keycloak. This issue was resolved by adjusting the session timeout settings in both Keycloak and PacketFence.

• User Experience: Users expressed concerns about being redirected multiple times between the captive portal and the SSO login page. This was mitigated by improving the flow and reducing unnecessary redirects.

• Network Performance: During peak times, the captive portal caused delays in authentication, primarily due to the large volume of RADIUS requests. This issue was addressed by optimizing the PacketFence configuration and increasing the server capacity.

The integration of PacketFence with Keycloak provided a seamless and secure SSO experience for WiFi users. Keycloak's SSO mechanism allowed users to authenticate with their existing corporate credentials, simplifying the login process and improving user satisfaction. The use of PacketFence ensured that unauthorized devices could not access the network, thereby enhancing security.

Performance metrics collected during the testing phase demonstrated a minimal impact on network latency, and the system successfully managed hundreds of concurrent logins during peak hours. Figure 2 shows that there is no significant change in latency (Y axis, in milliseconds) on the number of concurrent users (X axis), only a slight unstable latency on higher number of concurrent users. Moreover, the centralized identity management system facilitated easier user management and reduced administrative overhead.

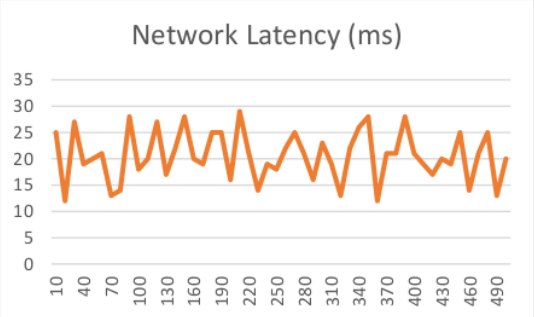

Figure 2. Network latency and concurrent users

V. SECURITY CONSIDERATIONS

When implementing captive portal authentication with PacketFence and FreeRADIUS, several security factors must be considered:

1. TLS Encryption: All communications between clients, PacketFence, FreeRADIUS, and Keycloak should be encrypted using TLS to prevent man-in-the-middle attacks. An SSL certificate with its CA chain bundle can be used.

2. Using LDAPS instead of LDAP: even when LDAP server is in the same server with FreeRADIUS, the best is always run the encrypted service of LDAP which is LDAPS. Placing LDAP, KeyCloak dan RADIUS in the same server is to provide more security as RADIUS or Keycloak may authenticate to LDAP in plain text. This also an SSL certificate can be used.

3. OAuth2 Token Security: Ensuring that OAuth2 tokens are securely handled and transmitted between Keycloak, PacketFence, and clients. Using HTTPS for all OAuth2 endpoints. If an access token is intercepted by an attacker, they can use it to gain unauthorized access to resources. This is why tokens should be encrypted and transferred only over secure channels (e.g., HTTPS). Attackers may also create malicious apps that mimic legitimate ones, tricking users into granting access to their data. Users should be cautious about the permissions they grant and ensure they trust the client before consenting. Short-lived access tokens reduce the window of opportunity for attacks. Additionally, refresh tokens should be securely handled, and users should have the ability to revoke tokens if suspicious activity is detected.

4. Session Management: Keycloak provides session management features, allowing administrators to monitor active sessions and revoke access when necessary.

5. Network Segmentation: Using VLANs to segment guest and authenticated users on the network, ensuring that unauthorized devices cannot access sensitive areas of the network. Wireless access points under its controller, are capable of running multiple VLANs.

VI. CONCLUSION

Combining PacketFence with FreeRADIUS and Keycloak provides a powerful solution for managing Wi-Fi network access with SSO capabilities. A single web portal will open user's access to not only Wi-Fi access but also the organization's web-based applications. By leveraging LDAP for centralized identity management, this setup ensures that both guest and regular users can securely authenticate via a captive portal. Whether an application authenticates direcly to LDAP with FreeRadius or through PacketFence and Keycloak, they will all authenticate to the same particular credentials. With this configuration, educational institutions and enterprises can easily manage and control network access while providing a seamless user experience through SSO.

REFERENCES

[1] Krawczyk, H., & Pirogova, A. (2022). The role of Identity Federation in Modern Network Architectures. Journal of Network Security, 48(3), 101-115.

[2] Arnaud, F., & Leclerc, M. (2023). Dynamic Network Access Control Using PacketFence: A Case Study. Journal of Information Systems, 29(2), 65-78.

[3] Kovac, M., & Petrovic, J. (2021). Single Sign-On and Identity Management with Keycloak. International Journal of Cybersecurity, 34(5), 88-97.

[4] PacketFence Project. (2023). "PacketFence Documentation". Available at: https://packetfence.org

[5] FreeRADIUS Project. (2023). "FreeRADIUS Documentation". Available at: https://freeradius.org/documentation

[6] Red Hat. (2023). "Keycloak Documentation". Available at: https://www.keycloak.org/documentation

[7] IEEE 802.1X-2020. (2020). "Port-Based Network Access Control". Available at: https://standards.ieee.org/standard/802_1X-2020.html

[8] Cristescu, Gabriel-Catalin. (2016). "Implementing an AAA-RADIUS solution based on legacy authentication protocols". 12th IEEE International Symposium on Electronics and Telecommunications (ISETC).

[9] Andjarwirawan, Justinus. & Palit, Henry N. (2017). "Linux PAM to LDAP Authentication Migration". International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT).

Single Sign-On (SSO) Implementation Using Keycloak, RADIUS, LDAP, and PacketFence for Network Access

ORIGINALITY REPORT

12%	10%	9%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Surabaya University Student Paper	4%
2	appmaster.io Internet Source	1%
3	docs.pivotal.io Internet Source	1%
4	blog.ostorlab.co Internet Source	1%
5	www.codemotion.com Internet Source	1%
6	Prabath Siriwardena. "Advanced API Security", Springer Science and Business Media LLC, 2020 Publication	1%
7	www.momentslog.com Internet Source	1%
8	Yvonne Wilson, Abhishek Hingnikar. "Solving Identity Management in Modern	1%

Applications", Springer Science and Business Media LLC, 2023
Publication

9	Submitted to University of Wales Swansea Student Paper	1 %
10	Submitted to Saxion HS Student Paper	1 %
11	Advanced API Security, 2014. Publication	1 %

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		