Whitepaper

Overview

Introduction to iDRAC Delegated Authorization

This white paper provides an overview of iDRAC Delegated Authorization. It explains Delegated Authorization and how it addresses the limitations with the current iDRAC web-based authorization architecture. It provides a solution overview and briefly describes the OAuth 2.0 authorization framework along with some key terminology from the perspective of an iDRAC.

May 2021

ID-463

Table of contents

Introduction to Delegated Authorization

Delegated Authorization is a way to access REST APIs in a modern secure manner. It is an important concept in modern HTTP APIs, with one of the most popular implementations being the IETF standard for OAuth 2.0 (RFC 6749). OAuth 2.0 is an industry-standard for handling web-based authorization and offers excellent solutions to get HTTP services to interoperate in an elegant and secure way.

Delegated auth does away with providing a username and password for every device. Instead users can log into a centralized “Authorization server” and get issued a “token”. These tokens can then be used on configured systems instead of a username and password. There are several Enterprise level authorization servers including Ping Identity, Microsoft ADFS, and open source solutions such as Keycloak. These authorization servers allow a wide variety of authentication services including AD and LDAP to suit consumers’ various needs.

Since Delegated Authorization does not involve sharing usernames or passwords with clients, it provides enhanced security and remains unaffected by password rotation schemes. This simplifies password rotation policies since new passwords do not need to be “rolled out” to scripts or console applications.

Limitations with the legacy architecture

With previous iDRAC releases, the only way to access iDRAC REST APIs was to include a username and password as authentication for every HTTP request. Optionally a username and password could be used to create a session, but the initial credentials exchange was still required.

This is true for both iDRAC local user accounts as well as for directory services such as Active Directory and LDAP. In both cases, the iDRAC must either handle user credentials to perform authentication for local users, or handle AD/LDAP user credentials to be passed over to AD/LDAP servers for authentication.

In a datacenter environment, scripts and console applications handle the remote server management. Since usernames and passwords are required for all iDRAC REST APIs, these scripts and applications need a username and password to operate.

It is apparent that the legacy solution is not very secure and is fraught with maintenance and scalability issues, some of which are outlined below.

Security

1. A security breach may result in denial-of-service attacks and loss of sensitive customer data such as usernames and passwords.
2. The legacy back-end applications need to deal with user credentials for storage, retrieval, and updates in a database.
3. User credentials may need to be transmitted over the network for AD/LDAP configurations.
4. Inadvertent leakage of sensitive user information in troubleshooting logs on proxies and servers etc.
5. Password rotation policies mandate client scripts and applications to be updated on a periodic basis.

Maintenance/Scalability

1. User account creation and management.
2. iDRAC provides an array of fine-grained user privileges but not all of them can be leveraged due to the limited number of local user accounts that can be set up on the iDRAC.
3. In the scenario where there are multiple iDRAC’s to be configured, all the user creation and management tasks need to be performed on a per-iDRAC basis.

4. Maintain and enhance authentication modules to comply with the latest security standards through application of patches etc.

Key terms for Delegated Authorization

Starting iDRAC version 4.20.20.00, iDRAC introduces support for Delegated Authorization.

To configure iDRAC for Delegated Authorization, an iDRAC admin needs to understand certain key OAuth 2.0 terminology. Initially, this terminology can be difficult to grasp. To simplify things, this section briefly describes various key OAuth 2.0 terms from an iDRAC perspective. In the next section, we will use these terms in describing the basic workflow.

Client	A client is a third-party application, console, or script that accesses the iDRAC REST APIs with the user's consent.
Authorization Server	The Authorization Server is the only device that directly handles usernames and passwords. It issues tokens to the client after successful user authentication.
Resource Server	The iDRAC is the Resource Server.
Resource Owner	The User is the Resource Owner.
Token	Tokens are used by iDRAC in place of usernames and passwords.
Offline Tokens	Tokens that Users provide to applications or scripts that can be used without requesting consent from the User each time.

Solution Overview

To address limitations with the legacy architecture, it becomes imperative to move to a centralized authorization solution and *delegate* authorization responsibilities to specialized servers. These authorization servers are specifically designed to provide user identity and access management solutions.

From an iDRAC perspective, the notion of Delegated Authorization is that a script or console application ("Client") can, on behalf of a user ("Resource Owner"), obtain authorization from an "Authorization Server" in the form of a "token". This token is then presented in a designated HTTP request header as part of API accesses to the iDRAC ("Resource Server"). The token contains details about the Authorization Server issuing the token, the iDRAC that this token is valid for, and user details such as the username and privileges that the user should have on the iDRAC.

Most importantly, the tokens are secure and trustworthy as they are signed by the Authorization Server's private encryption key. The public key is known to the iDRAC and is therefore used to verify the authenticity of the tokens.

So, with Delegated Authorization a User (“Resource Owner”) delegates access to the resources a user owns to a designated client application, without enabling the client application to impersonate the user.

This means that an iDRAC user may enable a third-party client application to invoke the iDRAC web API on users’ behalf without users having to share their username and password with the client application. To avoid constant requests for consent, a User may elect to provide the client with an “Offline Token”. A client will exchange this offline token with the Authorization Server for a normal token to access the iDRAC.

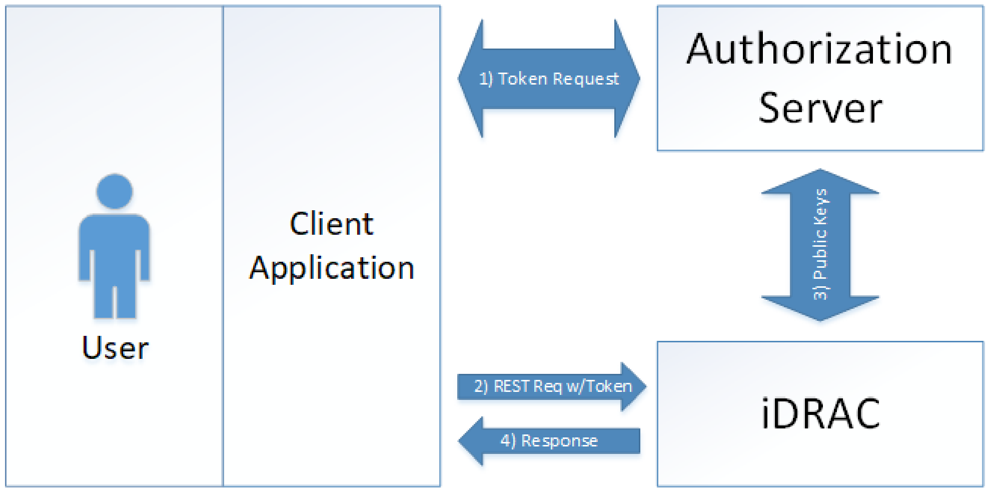

Fig 1

Initially iDRAC is configured to use Authorization Server’s public key.

Step 1: Client requests and obtains an access token on behalf of the user.
Step 2: Client specifies the access token on the API request.
Step 3: iDRAC validates token using the Authorization Server’s public key
Step 4: Client receives the response from the iDRAC.

Summary/Conclusion

Delegated Authorization greatly simplifies user account set up and access management. A centralized user management system eliminates password propagation and using short-lived tokens, the need to share credentials. No longer are usernames and passwords passed to client applications or even the iDRAC. Additionally, it provides users’ options to revoke client access as per users’ needs. Once Delegated Auth is configured, iDRAC API access is seamless using access token.

By providing centralized authentication and auditing, it enables users to focus on their business workflows and bring security standards to the forefront regarding user authorization and access. Delegated Auth is by far the most secure and standard way to access iDRAC Remote Services.