AWS Security Blog

Managing temporary elevated access to your AWS environment

by James Greenwood, Bikash Behera, and Kevin Higgins | on 12 NOV 2021 | in Advanced (300), AWS Identity and Access Management (IAM), Security, Identity, & Compliance | Permalink | Comments | Share

> **September 27, 2023:** We updated this post to include a list of newer temporary elevated access solutions that integrate with AWS IAM Identity Center.

> **September 9, 2022:** This blog post has been updated to reflect the new name of AWS Single Sign-On (SSO) – AWS IAM Identity Center. Read more about the name change here.

> **March 23, 2022**: In the section "Logging session activity," we fixed an error in the CloudTrail example and added a note of explanation.

In this post you'll learn about temporary elevated access and how it can help mitigate risks relating to human access to your AWS environment. You'll also learn about solutions for temporary elevated access, including a minimal reference implementation that you can download and use as a starting point to build a solution tailored for your organization.

Introduction

While many modern cloud architectures aim to eliminate the need for human access, there often remain at least some cases where it is required. For example, unexpected issues might require human intervention to diagnose or fix, or you might deploy legacy technologies into your AWS environment that someone needs to configure manually.

AWS provides a rich set of tools and capabilities for managing access. Users can authenticate with multi-factor authentication (MFA), federate using an external identity provider, and obtain temporary credentials with limited permissions. AWS Identity and Access Management (IAM) provides fine-grained access control, and AWS IAM Identity Center makes it simple to manage access across your entire organization using AWS Organizations.

For higher-risk human access scenarios, your organization can supplement your baseline access controls by implementing *temporary elevated access*.

What is temporary elevated access?

The goal of temporary elevated access is to make sure that each time a user invokes access, there is an appropriate business reason for doing so. For example, an appropriate business reason might be to fix a specific issue or deploy a planned change.

Traditional access control systems require users to be authenticated and authorized before they can access a protected resource. Becoming authorized is typically a one-time event, and a user's authorization status is reviewed periodically—for example as part of an access recertification process.

With *persistent access*, also known as *standing access*, a user who is authenticated and authorized can invoke access at any time just by navigating to a protected resource. The process of invoking access does not consider the reason why they are invoking it on each occurrence. Today, persistent access is the model that IAM Identity Center supports, and is the most common model used for IAM users and federated users.

With *temporary elevated access*, also known as *just-in-time access*, users must be authenticated and authorized as before—but furthermore, each time a user invokes access an additional process takes place, whose purpose is to identify and record the business reason for invoking access on this specific occasion. The process might involve additional human actors or it might use automation. When the process completes, the user is only granted access if the business reason is appropriate, and the scope and duration of their access is aligned to the business reason.

Why use temporary elevated access?

You can use temporary elevated access to mitigate risks related to human access scenarios that your organization considers high risk. Access generally incurs risk when two elements come together: high levels of privilege, such as ability to change configuration, modify permissions, read data, or update data; and high-value resources, such as production environments, critical services, or sensitive data. You can use these factors to define a risk threshold, above which you enforce temporary elevated access, and below which you continue to allow persistent access.

Your motivation for implementing temporary elevated access might be internal, based on your organization's risk appetite; or external, such as regulatory requirements applicable to your industry. If your organization has regulatory requirements, you are responsible for interpreting those requirements and determining whether a temporary elevated access solution is required, and how it should operate.

Regardless of the source of the requirement, the overall goal is to reduce risk.

> **Important**: While temporary elevated access can reduce risk, the preferred approach is to automate your way out of needing human access in the first place. Aim to use temporary elevated access only for infrequent activities that cannot yet be automated. From a risk perspective, the best kind of human access is the kind that doesn't happen at all.

The AWS Well-Architected Framework provides guidance on using automation to reduce the need for human user access:

- OPS06-BP07 — Fully automate integration and deployment
- OPS10-BP07 — Automate responses to events
- REL08-BP05 — Deploy changes with automation

How can temporary elevated access help reduce risk?

In scenarios that require human intervention, temporary elevated access can help manage the risks involved. It's important to understand that temporary elevated access does not replace your standard access control and other

security processes, such as access governance, strong authentication, session logging and monitoring, and anomaly detection and response. Temporary elevated access supplements the controls you already have in place.

The following are some of the ways that using temporary elevated access can help reduce risk:

1. Making sure that users only invoke elevated access when there is a valid business reason. Users are discouraged from invoking elevated access habitually, and service owners can mitigate potentially disruptive operations during critical time periods.

2. Visibility of access to other people. With persistent access, user activity is logged—but no one is routinely informed when a user invokes access, unless their activity causes an incident or security alert. With temporary elevated access, every access invocation is typically visible to at least one other person. This can arise from their participation in approvals, notifications, or change and incident management processes which are multi-party by nature. With greater visibility to more people, inappropriate access by users is more likely to be noticed and acted upon.

3. A reminder to be vigilant. Temporary elevated access provides an overt reminder for users to be vigilant when they invoke high-risk access. This is analogous to the kind security measures you see in a physical security setting. Imagine entering a secure facility. You see barriers, fences, barbed wire, CCTV, lighting, guards, and signs saying "You are entering a restricted area." Temporary elevated access has a similar effect. It reminds users there is a heightened level of control, their activity is being monitored, and they will be held accountable for the actions they perform.

4. Reporting, analytics, and continuous improvement. A temporary elevated access process records the reasons why users invoke access. This provides a rich source of data to analyze and derive insights. Management can see why users are invoking access, which systems need the most human access, and what kind of tasks they are performing. Your organization can use this data to decide where to invest in automation. You can measure the amount of human access and set targets to reduce it. The presence of temporary elevated access might also incentivize users to automate common tasks, or ask their engineering teams to do so.

Implementing temporary elevated access

Before you examine the available solutions, first take a look at a logical architecture for temporary elevated access, so you can understand the process flow at a high level.

A typical temporary elevated access solution involves placing an additional component between your identity provider and the AWS environment that your users need to access. This is referred to as a *temporary elevated access broker*, shown in Figure 1.

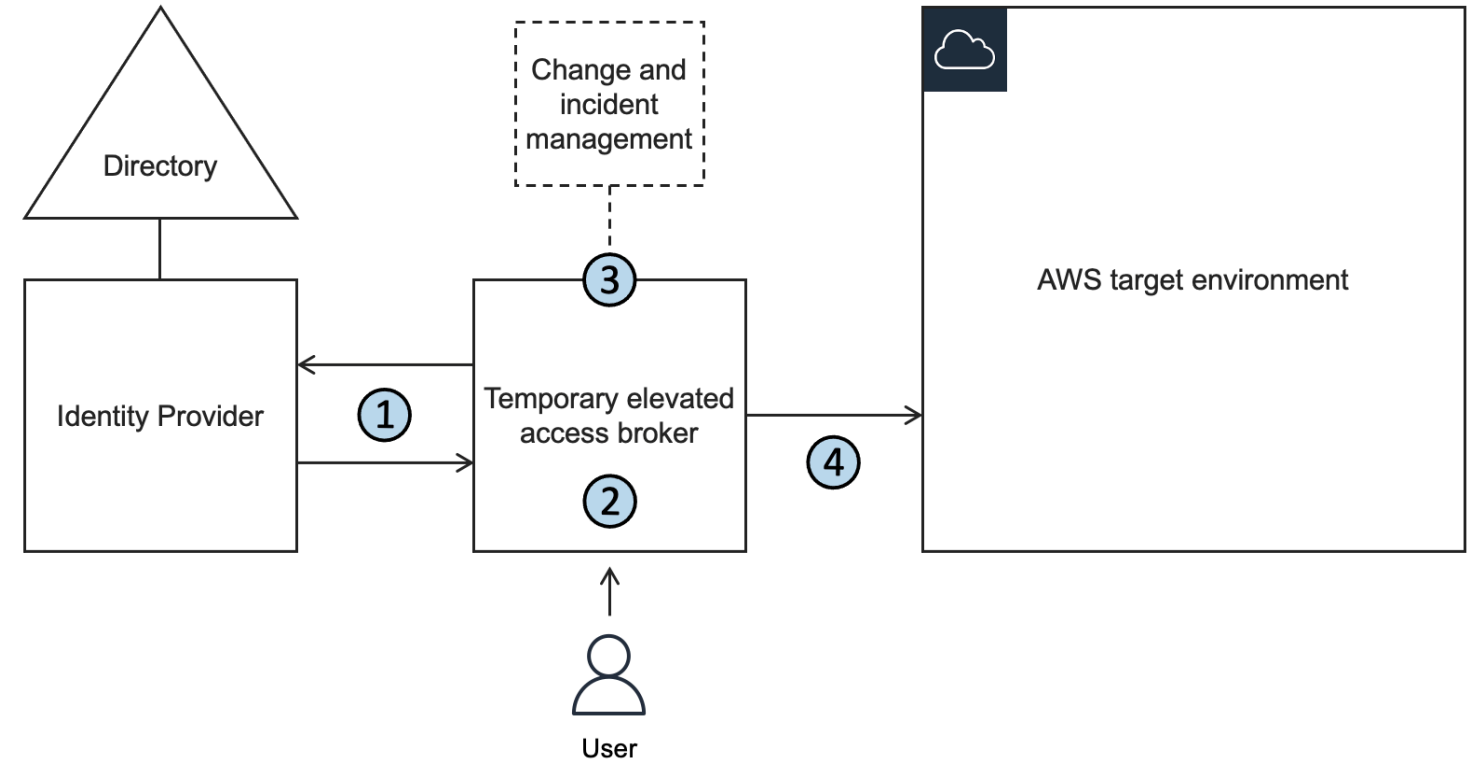


Figure 1: A logical architecture for temporary elevated access

When a user needs to perform a task requiring temporary elevated access to your AWS environment, they will use the broker to invoke access. The broker performs the following steps:

1. Authenticate the user and determine eligibility. The broker integrates with your organization's existing identity provider to authenticate the user with multi-factor authentication (MFA), and determine whether they are eligible for temporary elevated access.

> **Note**: *Eligibility* is a key concept in temporary elevated access. You can think of it as pre-authorization to invoke access that is contingent upon additional conditions being met, described in step 3. A user typically becomes eligible by becoming a trusted member of a team of admins or operators, and the scope of their eligibility is based on the tasks they're expected to perform as part of their job function. Granting and revoking eligibility is generally based on your organization's standard access governance processes. Eligibility can be expressed as group memberships (if using *role-based access control*, or RBAC) or user attributes (if using *attribute-based access control*, or ABAC). Unlike regular authorization, eligibility is not sufficient to grant access on its own.

2. Initiate the process for temporary elevated access. The broker provides a way to start the process for gaining temporary elevated access. In most cases a user will submit a request on their own behalf—but some broker designs allow access to be initiated in other ways, such as an operations user inviting an engineer to assist them. The scope of a user's requested access must be a subset of their eligibility. The broker might capture additional information about the context of the request in order to perform the next step.

3. Establish a business reason for invoking access. The broker tries to establish whether there is a valid business reason for invoking access with a given scope on this specific occasion. Why does this user need this access right now? The process of establishing a valid business reason varies widely between organizations. It might be a simple

approval workflow, a quorum-based authorization, or a fully automated process. It might integrate with existing change and incident management systems to infer the business reason for access. A broker will often provide a way to expedite access in a time-critical emergency, which is a form of *break-glass access*. A typical broker implementation allows you to customize this step.

4. Grant time-bound access. If the business reason is valid, the broker grants time-bound access to the AWS target environment. The scope of access that is granted to the user must be a subset of their eligibility. Further, the scope and duration of access granted should be necessary and sufficient to fulfill the business reason identified in the previous step, based on the *principle of least privilege*.

Temporary elevated access solutions

To get started with temporary elevated access, you can use one of the following solutions:

- **Vendor-managed and supported solutions that integrate with IAM Identity Center –** AWS Security Partners offer temporary elevated access solutions that integrate with IAM Identity Center, an AWS service that helps you securely create or connect your workforce identities and manage their access centrally across AWS accounts and applications. AWS has validated the integration of select partner offerings and assessed their capabilities against a common set of customer requirements. For more information, see temporary elevated access.
- **A self-managed and self-supported solution that integrates with IAM Identity Center –** a starting point for customers who are willing to deploy, tailor, and maintain the capability by themselves, this solution integrates with IAM Identity Center. For more information, see Temporary elevated access management with IAM Identity Center.
- **A self-managed and self-supported minimal reference implementation –** another option for customers who are willing to deploy, tailor, and maintain the capability by themselves, this solution integrates directly with an external identity provider using OpenID Connect, and federates users directly into IAM roles in your accounts.

The rest of this blog post describes the self-managed and self-supported minimal reference implementation.

A minimal reference implementation for temporary elevated access

Let's now look at a minimal reference implementation for temporary elevated access that you can download and deploy to your AWS environment. Information about deploying, running, and extending the solution is available in the Git repo README page.

> **Note**: You can use this reference implementation to complement the persistent access that you manage for IAM users, federated users, or manage through IAM Identity Center. For example, you can use the multi-account access model of IAM Identity Center for persistent access management, and create separate roles for temporary elevated access using this reference implementation.

To establish a valid business reason for invoking access, the reference implementation uses a single-step approval workflow. You can adapt the reference implementation and replace this with a workflow or business logic of your choice.

To grant time-bound access, the reference implementation uses the identity broker pattern. In this pattern, the broker itself acts as an intermediate *identity provider* which conditionally federates the user into the AWS target environment granting a time-bound session with limited scope.

Figure 2 shows the architecture of the reference implementation.

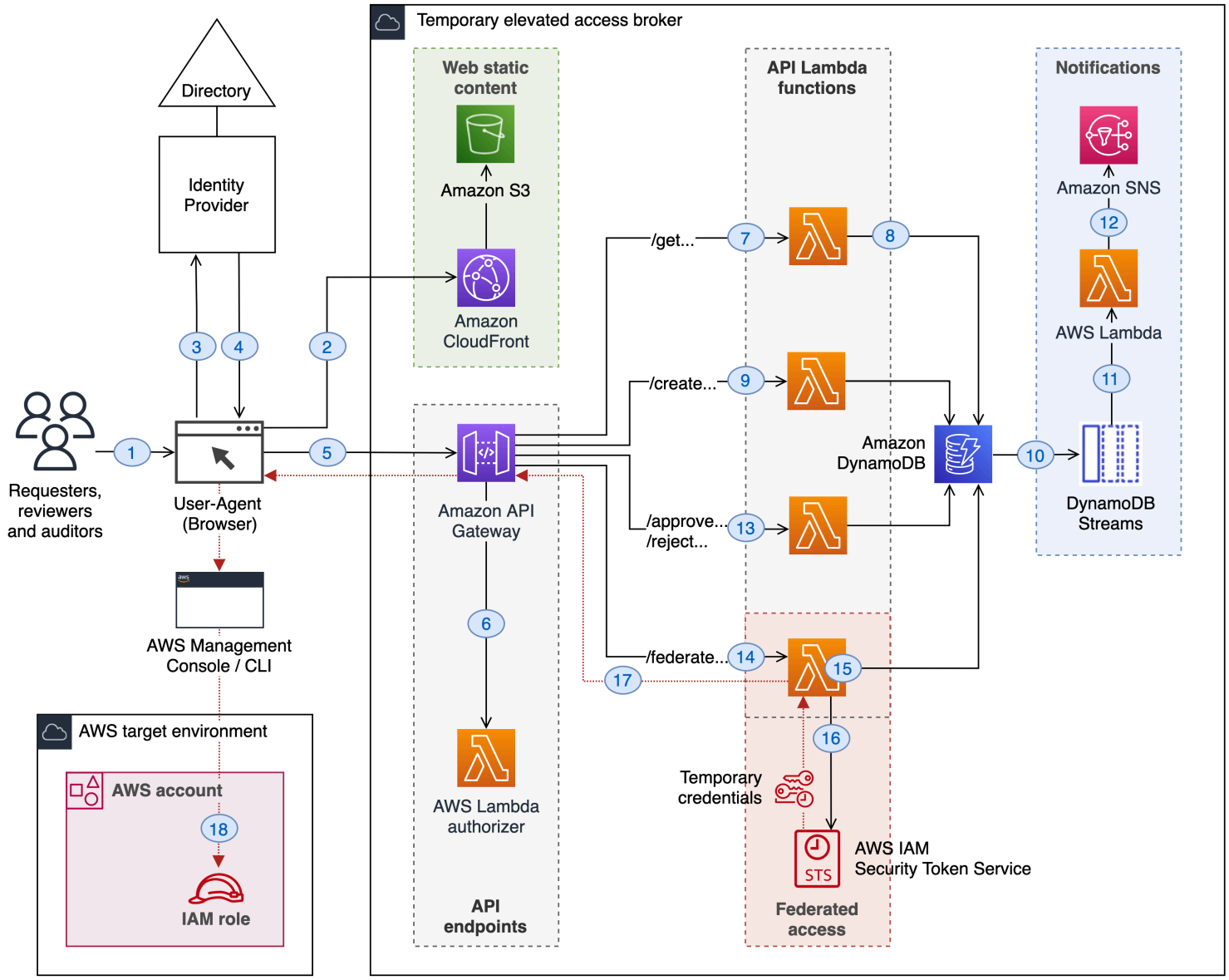


Figure 2: Architecture of the reference implementation

To illustrate how the reference implementation works, the following steps walk you through a user's experience end-to-end, using the numbers highlighted in the architecture diagram.

Starting the process

Consider a scenario where a user needs to perform a task that requires privileged access to a critical service running in your AWS environment, for which your security team has configured temporary elevated access.

Loading the application

The user first needs to access the temporary elevated access broker so that they can request the AWS access they need to perform their task.

1. The user navigates to the temporary elevated access broker in their browser.
2. The user's browser loads a web application using web static content from an Amazon CloudFront distribution whose target is an Amazon S3 bucket.

The broker uses a web application that runs in the browser, known as a Single Page Application (SPA).

> **Note**: CloudFront and S3 are only used for serving web static content. If you prefer, you can modify the solution to serve static content from a web server in your private network.

Authenticating users

3. The user is redirected to your organization's identity provider to authenticate. The reference implementation uses the OpenID Connect Authorization Code flow with Proof Key for Code Exchange (PKCE).
4. The user returns to the application as an authenticated user with an *access token* and *ID token* signed by the identity provider.

The access token grants delegated authority to the browser-based application to call server-side APIs on the user's behalf. The ID token contains the user's attributes and group memberships, and is used for authorization.

Calling protected APIs

5. The application calls APIs hosted by Amazon API Gateway and passes the access token and ID token with each request.
6. For each incoming request, API Gateway invokes a Lambda authorizer using AWS Lambda.

The Lambda authorizer checks whether the user's access token and ID token are valid. It then uses the ID token to determine the user's identity and their authorization based on their group memberships.

Displaying information

7. The application calls one of the `/get...` API endpoints to fetch data about previous temporary elevated access requests.
8. The `/get...` API endpoints invoke Lambda functions which fetch data from a table in Amazon DynamoDB.

The application displays information about previously-submitted temporary elevated access requests in a request dashboard, as shown in Figure 3.

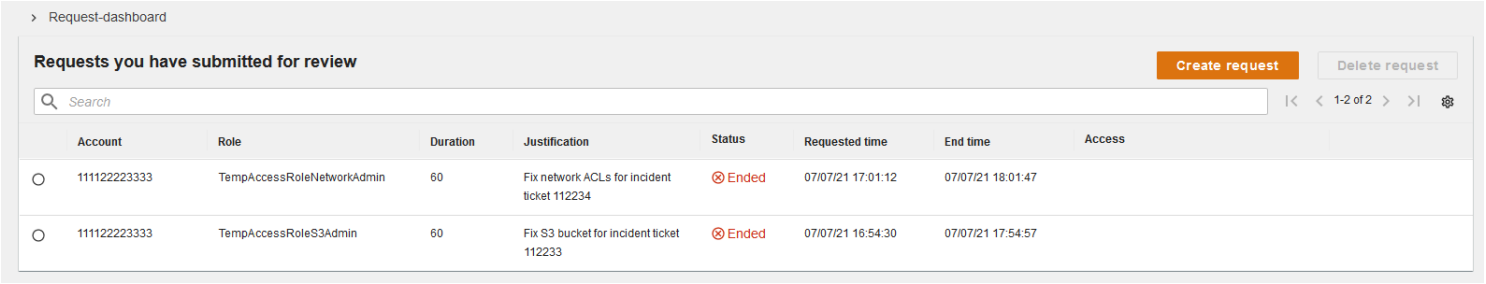


Figure 3: The request dashboard

Submitting requests

A user who is eligible for temporary elevated access can submit a new request in the request dashboard by choosing **Create request**. As shown in Figure 4, the application then displays a form with input fields for the IAM role name and AWS account ID the user wants to access, a justification for invoking access, and the duration of access required.

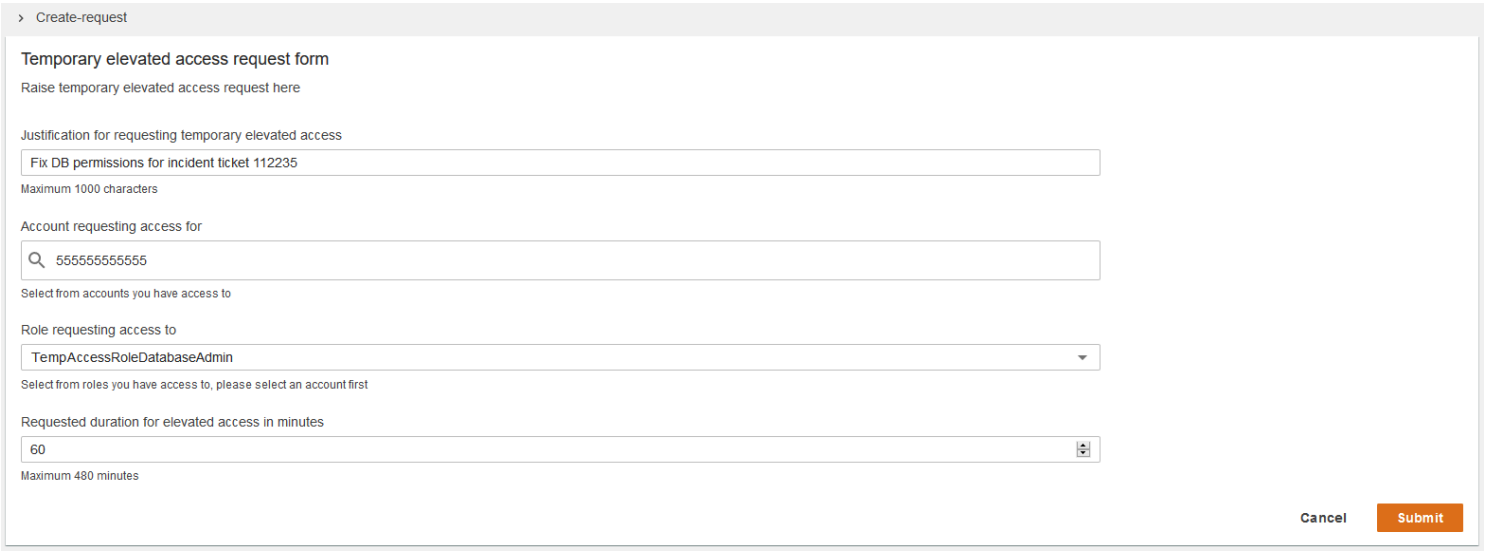


Figure 4: Submitting requests

The user can only request an IAM role and AWS account combination for which they are eligible, based on their group memberships.

> **Note**: The duration specified here determines a time window during which the user can invoke sessions to access the AWS target environment if their request is approved. It does not affect the duration of each session. Session duration can be configured independently.

9. When a user submits a new request for temporary elevated access, the application calls the `/create...` API endpoint, which writes information about the new request to the DynamoDB table.

The user can submit multiple concurrent requests for different role and account combinations, as long as they are eligible.

Generating notifications

The broker generates notifications when temporary elevated access requests are created, approved, or rejected.

10. When a request is created, approved, or rejected, a DynamoDB stream record is created for notifications.
11. The stream record then invokes a Lambda function to handle notifications.
12. The Lambda function reads data from the stream record, and generates a notification using Amazon Simple Notification Service (Amazon SNS).

By default, when a user submits a new request for temporary elevated access, an email notification is sent to each authorized reviewer. When a reviewer approves or rejects a request, an email notification is sent to the original requester.

Reviewing requests

A user who is authorized to review requests can approve or reject requests submitted by other users in a review dashboard, as shown in Figure 5. For each request awaiting their review, the application displays information about the request, including the business justification provided by the requester.

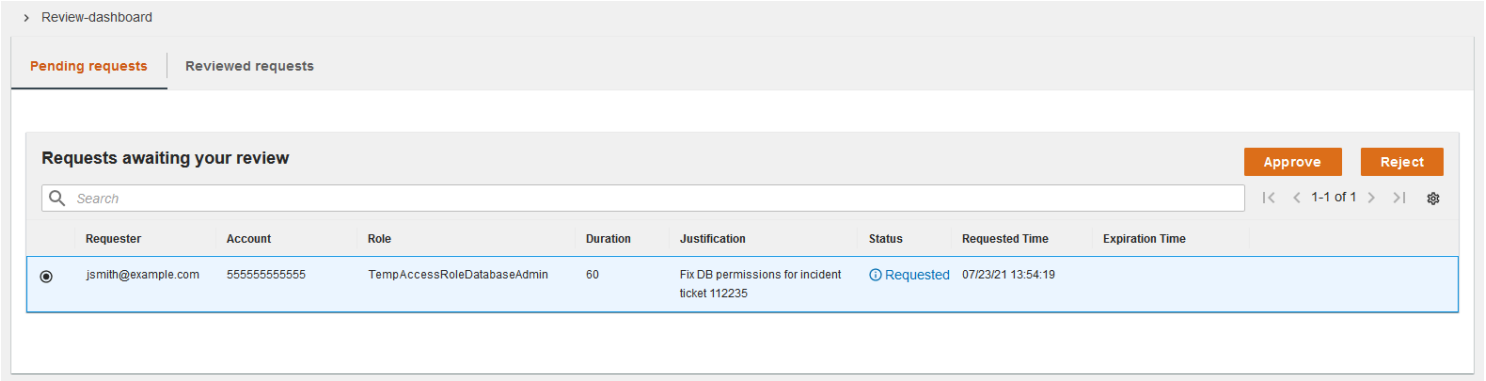


Figure 5: The review dashboard

The reviewer can select a request, determine whether the request is appropriate, and choose either **Approve** or **Reject**.

13. When a reviewer approves or rejects a request, the application calls the `/approve...` or `/reject...` API endpoint, which updates the status of the request in the DynamoDB table and initiates a notification.

Invoking sessions

After a requester is notified that their request has been approved, they can log back into the application and see their approved requests, as shown in Figure 6. For each approved request, they can invoke sessions. There are two ways they can invoke a session, by choosing either **Access console** or **CLI**.

- If the user chooses **Access console**, then the broker federates them into the AWS Management Console.
- If the user chooses **CLI**, then the broker provides temporary credentials for use with the AWS Command-Line Interface (AWS CLI).

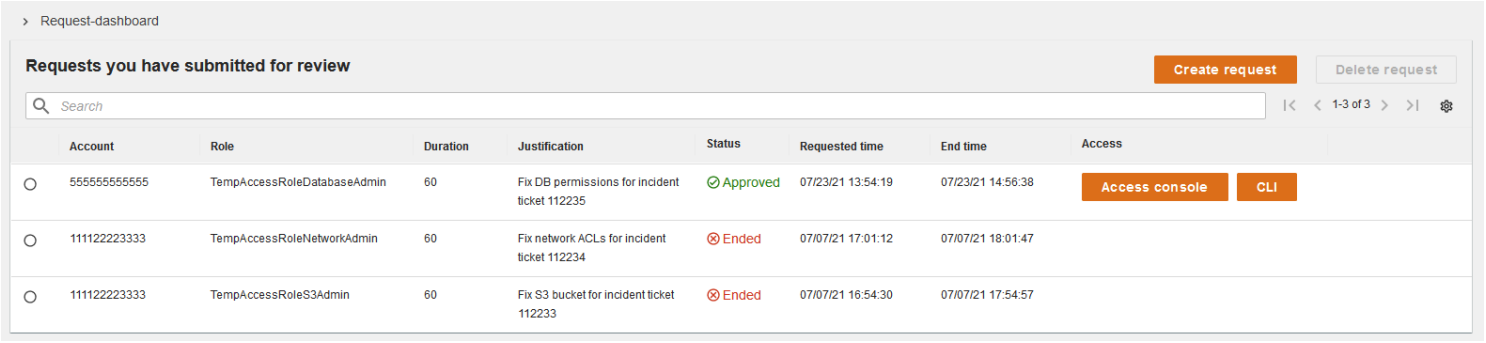


Figure 6: Invoking sessions

Both options grant the user a session in which they assume the IAM role in the AWS account specified in their request.

When a user invokes a session, the broker performs the following steps.

14. When the user chooses **Access console** or **CLI**, the application calls one of the `/federate...` API endpoints.
15. The `/federate...` API endpoint invokes a Lambda function, which performs the following three checks before proceeding:
    a. *Is the user authenticated?* The Lambda function checks that the access and ID tokens are valid and uses the ID token to determine their identity.
    b. *Is the user eligible?* The Lambda function inspects the user's group memberships in their ID token to confirm they are eligible for the AWS role and account combination they are seeking to invoke.
    c. *Is the user elevated?* The Lambda function confirms the user is in an elevated state by querying the DynamoDB table, and verifying whether there is an approved request for this user whose duration has not yet ended for the role and account combination they are seeking to invoke.
16. If the three checks succeed, the Lambda function calls sts:AssumeRole to fetch temporary credentials on behalf of the user for the IAM role and AWS account specified in the request.
17. The application returns the temporary credentials to the user.
18. The user obtains a session with temporary credentials for the IAM role in the AWS account specified in their request, either in the AWS Management Console or AWS CLI.

Once the user obtains a session, they can complete the task they need to perform in the AWS target environment using either the AWS Management Console or AWS CLI.

The IAM roles that users assume when they invoke temporary elevated access should be dedicated for this purpose. They must have a trust policy that allows the broker to assume them. The trusted principal is the Lambda execution role used by the broker's `/federate...` API endpoints. This helps make sure that the only way to assume those roles is through the broker.

In this way, when the necessary conditions are met, the broker assumes the requested role in your AWS target environment on behalf of the user, and passes the resulting temporary credentials back to them. By default, the

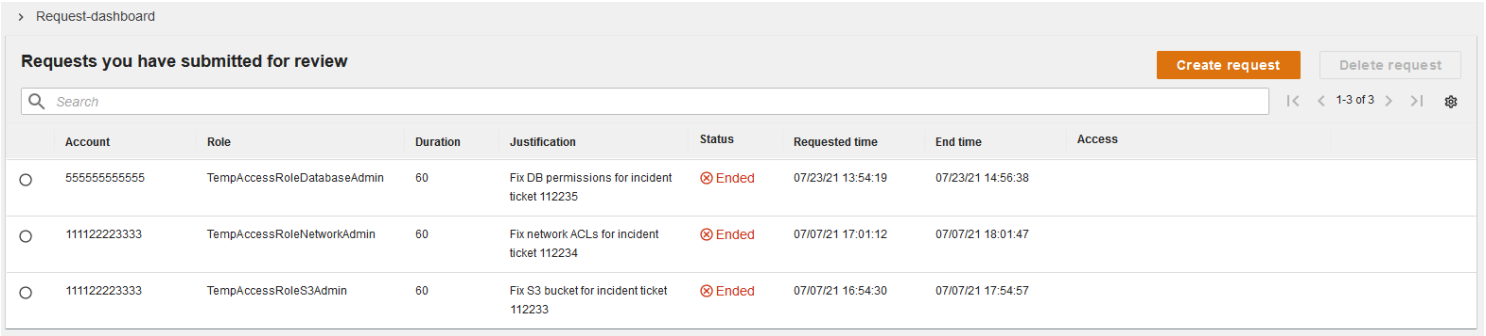
temporary credentials last for one hour. For the duration of a user's elevated access they can invoke multiple sessions through the broker, if required.

Session expiry

When a user's session expires in the AWS Management Console or AWS CLI, they can return to the broker and invoke new sessions, as long as their elevated status is still active.

Ending elevated access

A user's elevated access ends when the requested duration elapses following the time when the request was approved.



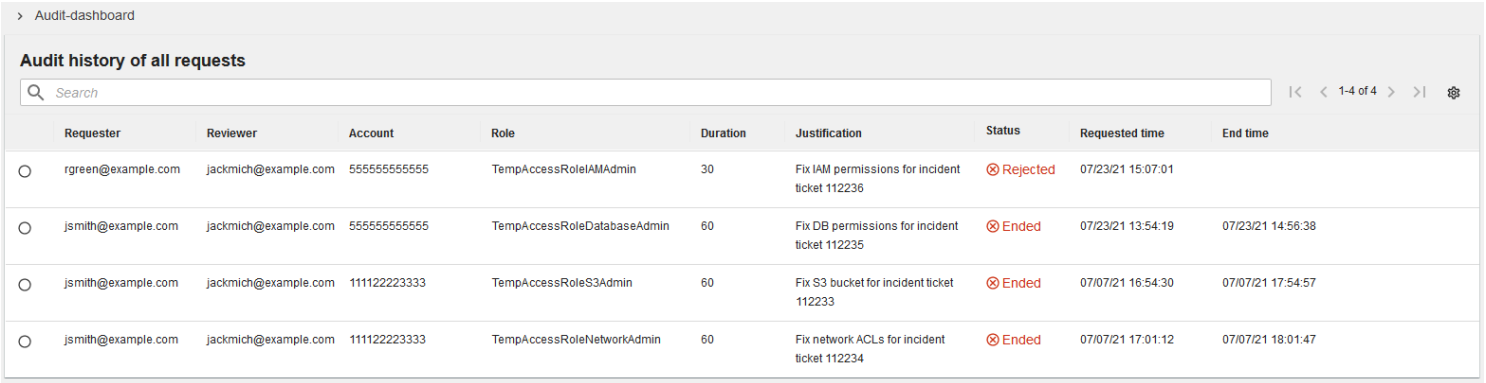
Account	Role	Duration	Justification	Status	Requested time	End time	Access
555555555555	TempAccessRoleDatabaseAdmin	60	Fix DB permissions for incident ticket 112235	Ended	07/23/21 13:54:19	07/23/21 14:56:38	
111122223333	TempAccessRoleNetworkAdmin	60	Fix network ACLs for incident ticket 112234	Ended	07/07/21 17:01:12	07/07/21 18:01:47	
111122223333	TempAccessRoleS3Admin	60	Fix S3 bucket for incident ticket 112233	Ended	07/07/21 16:54:30	07/07/21 17:54:57	

Figure 7: Ending elevated access

Once elevated access has ended for a particular request, the user can no longer invoke sessions for that request, as shown in Figure 7. If they need further access, they need to submit a new request.

Viewing historical activity

An audit dashboard, as shown in Figure 8, provides a read-only view of historical activity to authorized users.



Requester	Reviewer	Account	Role	Duration	Justification	Status	Requested time	End time
rgreen@example.com	jackmich@example.com	555555555555	TempAccessRoleIAMAdmin	30	Fix IAM permissions for incident ticket 112236	Rejected	07/23/21 15:07:01	
jsmith@example.com	jackmich@example.com	555555555555	TempAccessRoleDatabaseAdmin	60	Fix DB permissions for incident ticket 112235	Ended	07/23/21 13:54:19	07/23/21 14:56:38
jsmith@example.com	jackmich@example.com	111122223333	TempAccessRoleS3Admin	60	Fix S3 bucket for incident ticket 112233	Ended	07/07/21 16:54:30	07/07/21 17:54:57
jsmith@example.com	jackmich@example.com	111122223333	TempAccessRoleNetworkAdmin	60	Fix network ACLs for incident ticket 112234	Ended	07/07/21 17:01:12	07/07/21 18:01:47

Figure 8: The audit dashboard

Logging session activity

When a user invokes temporary elevated access, their session activity in the AWS control plane is logged to AWS CloudTrail. Each time they perform actions in the AWS control plane, the corresponding CloudTrail events contain the unique identifier of the user, which provides traceability back to the identity of the human user who performed the actions.

The following example shows the `userIdentity` element of a CloudTrail event for an action performed by user `someone@example.com` using temporary elevated access.

```
        "type": "AssumedRole",
        "principalId": "AROACKCEVSQ6C2EXAMPLE:someone@example.com-TempAccessRoleS3Admin",
        "arn": "arn:aws:sts::111122223333:assumed-role/TempAccessRoleS3Admin/someone@exar
        "accountId": "111122223333",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/TempAccessRoleS3Admin",
                "accountId": "111122223333",
                "userName": "TempAccessRoleS3Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-07-02T13:24:06Z"
            }
        }
    }
```

> **Note**: The `mfaAuthenticated` attribute in the CloudTrail logs shows `false` for users who authenticate with an external identity provider, regardless of authentication strength. Currently, this attribute is set to `true` only when users use MFA natively in AWS. We strongly recommend that you enforce MFA in your identity provider so that each user accessing the broker uses strong authentication.

Security considerations

The temporary elevated access broker controls access to your AWS environment, and you should manage the broker carefully with the goal of preventing unauthorized access. It is also an inline dependency for accessing your AWS environment and must operate with sufficient resiliency.

The broker should be deployed in a dedicated AWS account with a minimum of dependencies on the AWS target environment for which you'll manage access. It should use its own access control configuration following the principle of least privilege. Ideally the broker should be managed by a specialized team and use its own deployment pipeline, with a two-person rule for making changes—for example by requiring different users to check in code and

approve deployments. Special care should be taken to protect the integrity of the broker's code and configuration and the confidentiality of the temporary credentials it handles.

See the reference implementation README for further security considerations.

Extending the solution

You can extend the reference implementation to fit the requirements of your organization. Here are some ways you can extend the solution:

- **Customize the UI**, for example, to use your organization's branding.
- **Keep network traffic within your private network**, for example, to comply with network security policies.
- **Change the process for initiating and evaluating temporary elevated access**, for example, to integrate with a change or incident management system.
- **Change the authorization model**, for example, to use groups with different scope, granularity, or meaning.
- **Use SAML 2.0**, for example, if your identity provider does not support OpenID Connect.

See the reference implementation README for further details on extending the solution.

Conclusion

In this blog post you learned about *temporary elevated access* and how it can help reduce risk relating to human user access. You learned that you should aim to eliminate the need to use high-risk human access through the use of automation, and only use temporary elevated access for infrequent activities that cannot yet be automated. Finally, you learned about solutions for temporary elevated access, including a self-managed and self-supported minimal reference implementation that you can download and customize to fit your organization's needs.

If you have feedback about this post, submit comments in the **Comments** section below. If you have questions about this post, start a new thread on the AWS IAM forum or contact AWS Support.

Want more AWS Security how-to content, news, and feature announcements? Follow us on Twitter.

TAGS: AWS CLI, AWS IAM, AWS Management Console, Identity broker, Privileged access, Security Blog, Temporary elevated access

Comments

ALSO ON **AWS SECURITY BLOG**

Announcing initial services available ...

2 months ago • 1 comment

English | French | German | Italian | Spanish Last month, we shared that we ...

How to use the AWS Secrets Manager ...

a month ago • 1 comment

AWS Secrets Manager is a service that helps you manage, retrieve, and ...

Generate AI powered insights for ...

7 months ago • 1 comment

In part 1, we discussed how to use Amazon SageMaker Studio to analyze ...

How to u 2.0 in Am

5 months ago

Implementi authenticati authorizatio

3 Comments

4 Prashanth ▾

Join the discussion...

♡ Share

Best Newest Oldest

kriptone Digital Security

a month ago

Thanks Nice and Informative Blog.
Computer security service

0 0 Reply

Nolarond

a year ago

Do you have reccomendations on how to structure groups (in the identity provider, for example AD) that contain the elevated access accounts?

0 0 Reply

gt_gamer

a year ago

I implemented every step to the letter (multiple times), yet dashboard is not rendering. Any help would be appreciated. Thank you.

Sam

0 0 Reply