Send Idira Audit Service events to AWS Security Hub in OCSF format

This topic describes how to configure Palo Alto Networks Idira® to deliver Idira® Audit Service by Palo Alto Networks events to AWS Security Hub in the Open Cybersecurity Schema Framework (OCSF) format, so identity threat detections appear in the AWS-native security console.

> **This integration uses a preview AWS Security Hub API.** The `BatchImportFindingsV2` and `GetFindingsV2` APIs are not yet generally available, so the deployment ships a pre-built dependency package rather than installing dependencies at deploy time. When these APIs reach general availability, a later release removes the pre-built package and the direct-transport fallback.

Overview

The integration runs as a scheduled AWS Lambda function in your AWS account. On each run it reads audit events from the Idira Audit Service API, transforms them to OCSF v1.1.0 and then to the AWS Security Finding Format (ASFF), and imports them into AWS Security Hub. Identity findings then appear alongside AWS-native sources for unified triage, correlation, and compliance reporting.

Event delivery is near-real-time within the Idira Audit polling interval. It is not streaming: the Idira Audit API is polling-only, bounded to one request per minute. The default schedule runs the Lambda once per hour.

This guide covers three phases:

- **Configure the Idira Audit Service source** to authorize event reads.
- **Deploy the integration to your AWS account** using AWS CloudFormation or Terraform.
- **Provide credentials and verify** that findings reach AWS Security Hub.

Before you begin

Complete these tasks before you start. The first step of each procedure references this section.

- Confirm you have an Idira Identity Security Platform tenant with the Audit Service enabled.
- Confirm the tenant is not already at its integration limit. An Idira environment supports a maximum of two third-party SIEM integration slots. If two are already configured, free one before you continue.
- Enable AWS Security Hub in the target AWS Region before deployment.
- Obtain an AWS account with permission to create a Lambda function, an IAM role, an EventBridge rule, an SSM parameter, and a Secrets Manager secret.
- Create an S3 bucket in the target Region to hold the Lambda deployment package.
- Install AWS CLI v2 and authenticate it with sufficient permissions. Python 3.12 or later is required only for local development.

Configure the Idira Audit Service source

This procedure authorizes the integration to read Idira Audit Service events. Record the six credential values it produces. You enter them in AWS Secrets Manager after deployment.

To configure the source:

1. In the Idira Identity administration console, register a confidential OAuth 2.0 client (a service account) for the integration.
2. Set the grant type to `client_credentials` and request the scope `isp.audit.events:read`.
3. Record the client ID and client secret the console generates. Store the secret in your secrets manager. Do not paste it into this guide, a ticket, or a shared file.
4. Record the OAuth 2.0 web application name. The token endpoint for your tenant is `https://<tenant>.id.cyberark.cloud/oauth2/token/<web_app>`.
5. In the Idira Audit console, open **Integrations** and generate or retrieve the SIEM API key. The integration sends this key in the `x-api-key` header.
6. Record the Idira Audit API base URL and the Idira Identity URL for your tenant.

You now have the six values the integration needs: `api_base_url`, `identity_url`, `client_id`, `client_secret`, `web_app`, and `api_key`.

Deploy the integration to your AWS account

Both deployment options create the same resources: a Lambda function, an IAM execution role, an EventBridge schedule rule, an SSM parameter that stores the polling cursor, and a Secrets Manager secret for the Idira credentials.

Choose one option.

Option A: AWS CloudFormation. Run the build script with your S3 bucket. It packages the Lambda, uploads it, and deploys the stack.

```
./deploy/build.sh <s3-bucket> [stack-name] [region] [profile]
```

Option B: Terraform. Upload the Lambda package first (run `deploy/build.sh`, or upload manually), then apply.

```
cd deploy/terraform
terraform init
terraform apply -var="artifacts_bucket=<s3-bucket>"
```

To limit which Idira services are ingested, set the `ApplicationCodes` parameter (CloudFormation) or the `application_codes` variable (Terraform) to a comma-separated list, for example `PAM,EPM`. Leave it empty to ingest all services. To change the run frequency, set the schedule expression, for example `rate(30 minutes)`.

Provide Idira credentials in AWS Secrets Manager

The deployment creates a placeholder secret at `/idira-audit/<stack>/credentials`. Populate it with the six values from the source procedure before the first run.

To populate the secret:

1. Retrieve the secret ARN from the stack outputs.

2. Write the six values as a JSON document to the secret with `aws secretsmanager put-secret-value`.
3. Confirm the values are `api_base_url`, `identity_url`, `client_id`, `client_secret`, `web_app`, and `api_key`.

```
aws secretsmanager put-secret-value \
  --secret-id "<secret-arn>" \
  --secret-string '{
    "api_base_url":  "https://<tenant>.cyberark.cloud",
    "identity_url":  "https://<tenant>.id.cyberark.cloud",
    "client_id":     "your-client-id",
    "client_secret": "your-client-secret",
    "web_app":       "your-web-app-name",
    "api_key":       "your-api-key"
  }'
```

Verify the integration

Confirm the integration works end to end after you deploy the stack and populate the secret.

1. In the Idira Audit Service, generate a test event that maps to a configured OCSF class, for example an authentication event (class `3002`).
2. Wait for at least one polling interval, or invoke the Lambda manually to trigger a run.
3. In the AWS Security Hub console, open **Findings** and filter by product name `Idira Audit`, company name `Idira`, or generator ID prefix `IdiraAudit/`.
4. Confirm the finding appears with the expected OCSF class and a severity that matches the Idira event severity.

The integration tracks its position with a cursor stored in SSM Parameter Store at `/idira-audit/<stack>/cursor`. To reprocess events, set the cursor value to `UNSET` and set the `FETCH_EVENTS_DAYS` environment variable to the lookback window you want.

Troubleshoot

No findings appear in AWS Security Hub.
Confirm AWS Security Hub is enabled in the target Region. Confirm the OAuth 2.0 client uses the `client_credentials` grant and the `isp.audit.events:read` scope, and that the Secrets Manager values are correct. Confirm the environment is within the two-SIEM-slot limit.

Findings appear later than expected.
This is expected behavior within limits. The Idira Audit API is polling-only at one request per minute, and the default schedule runs hourly. Lower the schedule frequency if you need shorter delivery windows. If delivery still exceeds the two-minute target after a run, confirm the cursor is advancing and no page of 500 events is being reprocessed.

Findings appear with the wrong severity.
Confirm the mapping from Idira event severity to the OCSF `severity_id` field matches the mapping in the integration configuration.

A deployment fails during dependency packaging.
The build depends on a preview Security Hub SDK that is not on PyPI. Do not run `pip install` for these dependencies at deploy time: the public wheel of the same version number lacks the preview APIs and the deployment fails at runtime. Use the vendored package that ships with the repository.

A credential appears in a log or configuration file.
Stop and rotate the credential. Store secrets in AWS Secrets Manager and reference them by ARN. Do not record a client secret, token, or key in this guide or any shared file.

Scope and limitations

- This integration surfaces and correlates identity findings in AWS Security Hub. It does not trigger response actions in Idira. Security Hub to Idira response orchestration is a separate future phase.
- The integration reads Idira Audit Service events. It does not read the Idira identity threat detection and response alert stream.
- Delivery is near-real-time within the one-request-per-minute polling interval. It is not streaming.
- The integration consumes one of the two third-party SIEM integration slots available per Idira environment.

- The integration depends on a preview AWS Security Hub API and ships vendored dependencies as an interim measure until that API reaches general availability.