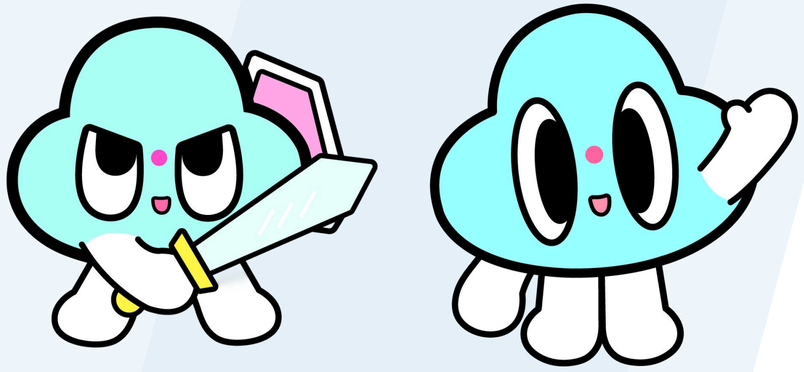
BEAVER DAM

AWSKRUG

OIDC의 도입과 함께 심화되어 온 공급망 공격의 변천사

AWS KRUG #Security · 2026.07



발표자 소개

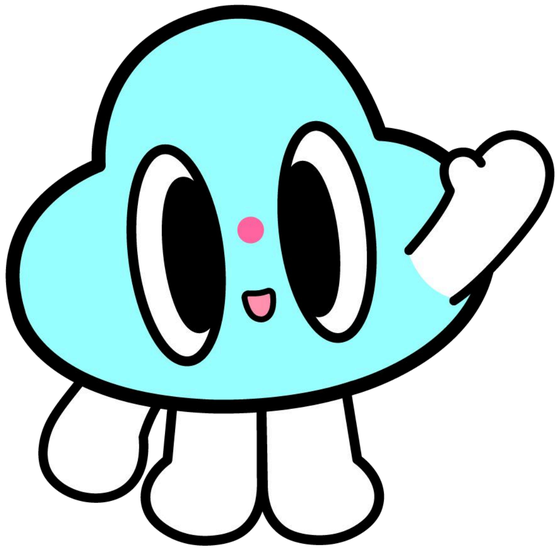
용시우

BeaverDam Community Builder

BeaverDam Community Builder

(주)에이쓰리시큐리티

목차



1

OIDC?

OAuth 비교 · Flow · 참여자 · claim의 유연성

OIDC가 무엇인가요?

	OAuth 2.0	OpenID Connect
목적	권한 위임 (Authorization)	**신원 검증 (Authentication)**
발급물	Access Token	**ID Token + Access Token**
질문	“접근해도 되니?”	**“너는 누구니?”**

OAuth = 호텔 키카드　　OIDC = 여권 + 키카드

OIDC 가 무엇인가요?

OAuth 2.0 vs OpenID Connect

OAuth 2.0

```
{
  "access_token": "ya29.a0Af...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "drive.readonly"
}
```

OpenID Connect

```
{
  "access_token": "ya29.a0Af...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "openid profile email",
  "id_token": "eyJhbGciOi..."  ★
}
```

★ = OIDC가 추가한 필드

OIDC Token의 Claim

OAuth ID Token = OIDC Token

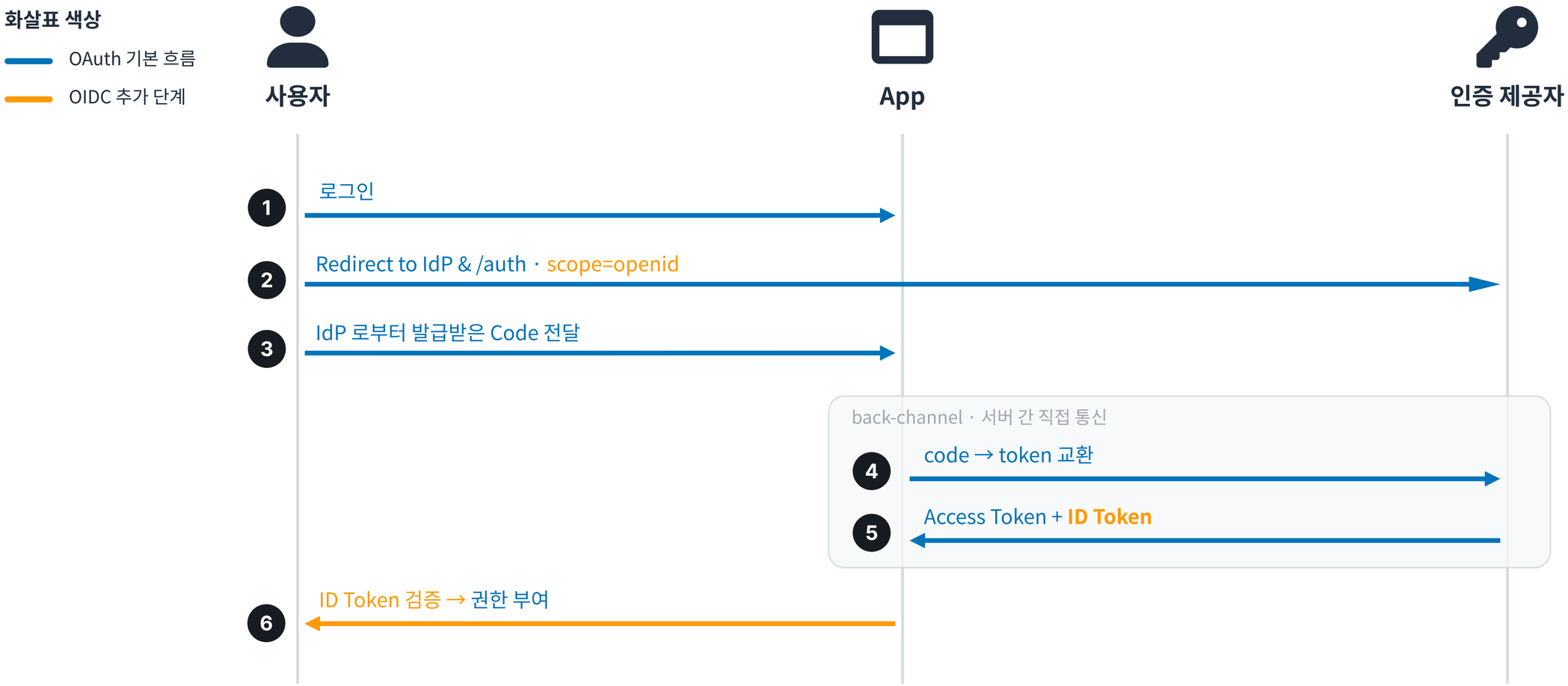
```
eyJhbGciOiJSUzI1Ni . eyJpc3MiOiJhY2NvdW50cy5nb29nbGUuY29tIiwic3ViIjoi . SflKxwRJSMeKKF2QT4
```

Header | Payload (Claims) | Signature

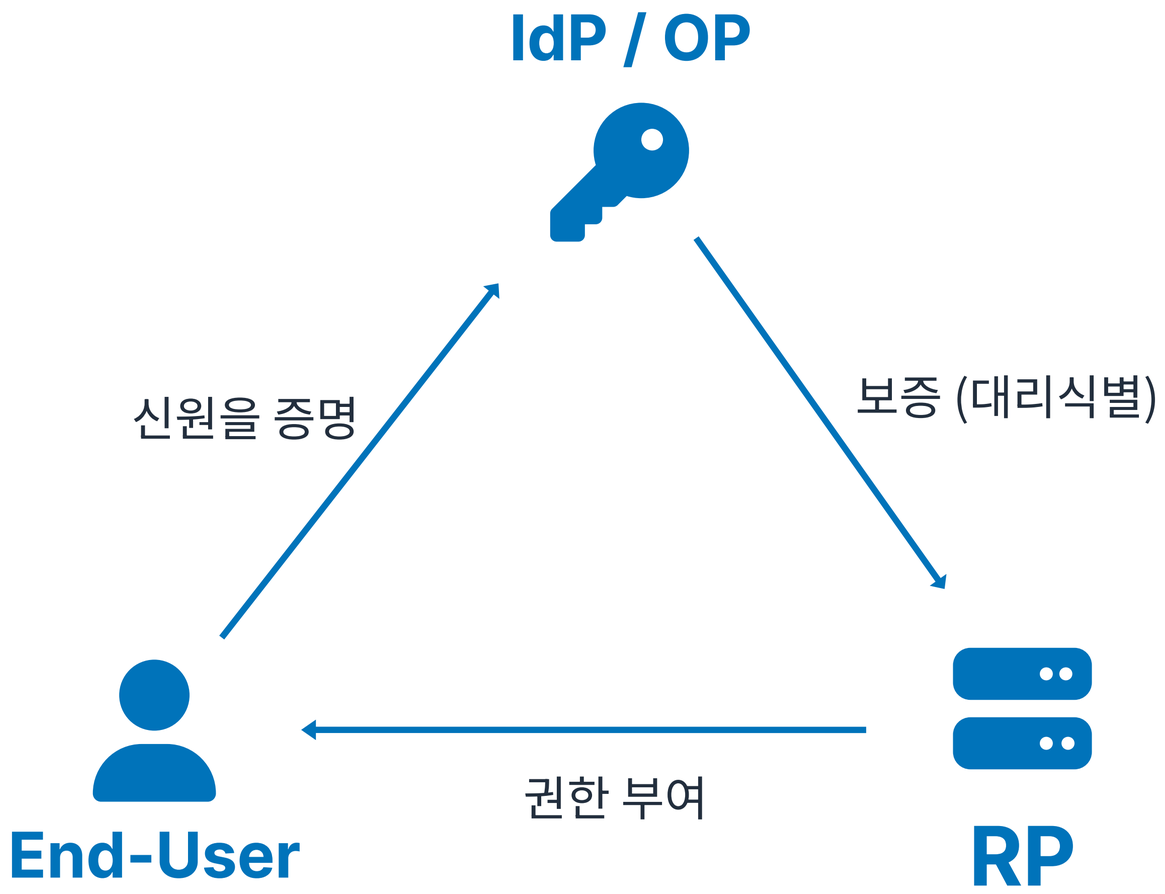
```
{
  "iss": "accounts.google.com",
  "sub": "110248...",
  "aud": "my-app.example.com",
  "exp": 1752537600,
  "iat": 1752534000
}
```

iss	발급자 — 누가 만들었나
sub	사용자 — 누구에 대한 것인가
aud	대상 — 어디에 쓸 것인가
exp	만료 시각
iat	발급 시각

인증/인가 Flow


화살표 색상
OAuth 기본 흐름
OIDC 추가 단계
사용자
App
인증 제공자
1
로그인
2
Redirect to IdP & /auth · scope=openid
3
IdP 로부터 발급받은 Code 전달
back-channel · 서버 간 직접 통신
4
code → token 교환
5
Access Token + ID Token
6
ID Token 검증 → 권한 부여

OIDC 인증의 핵심 참여자



OIDC로 무엇을 할 수 있나요?

End-User	IdP	RP
신원을 증명	신원을 보증	토큰을 신뢰
사람 (브라우저)	Google	Slack
워크플로우	GitHub Actions	AWS STS
Pod	EKS OIDC	AWS STS
워크플로우	GitHub Actions	npm

2

OIDC와 공급망

정적 시크릿 · 타임라인 · 신뢰 구조

우리 공급망 — 소스에서 배포까지

소스 → CI Runner → Artifact → ArgoCD → 배포, 여러 홉에서 OIDC로 신원을 증명한다

정적 시크릿의 시대

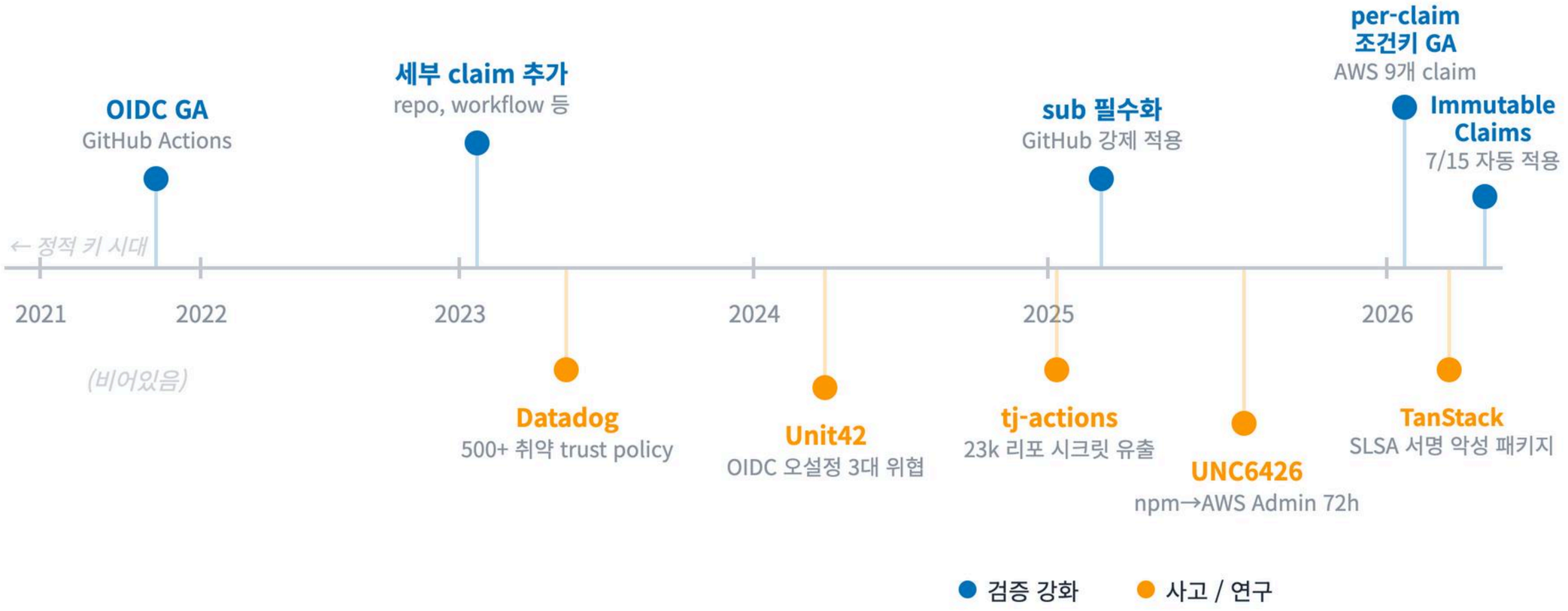
Usage

Add the following step to your workflow:

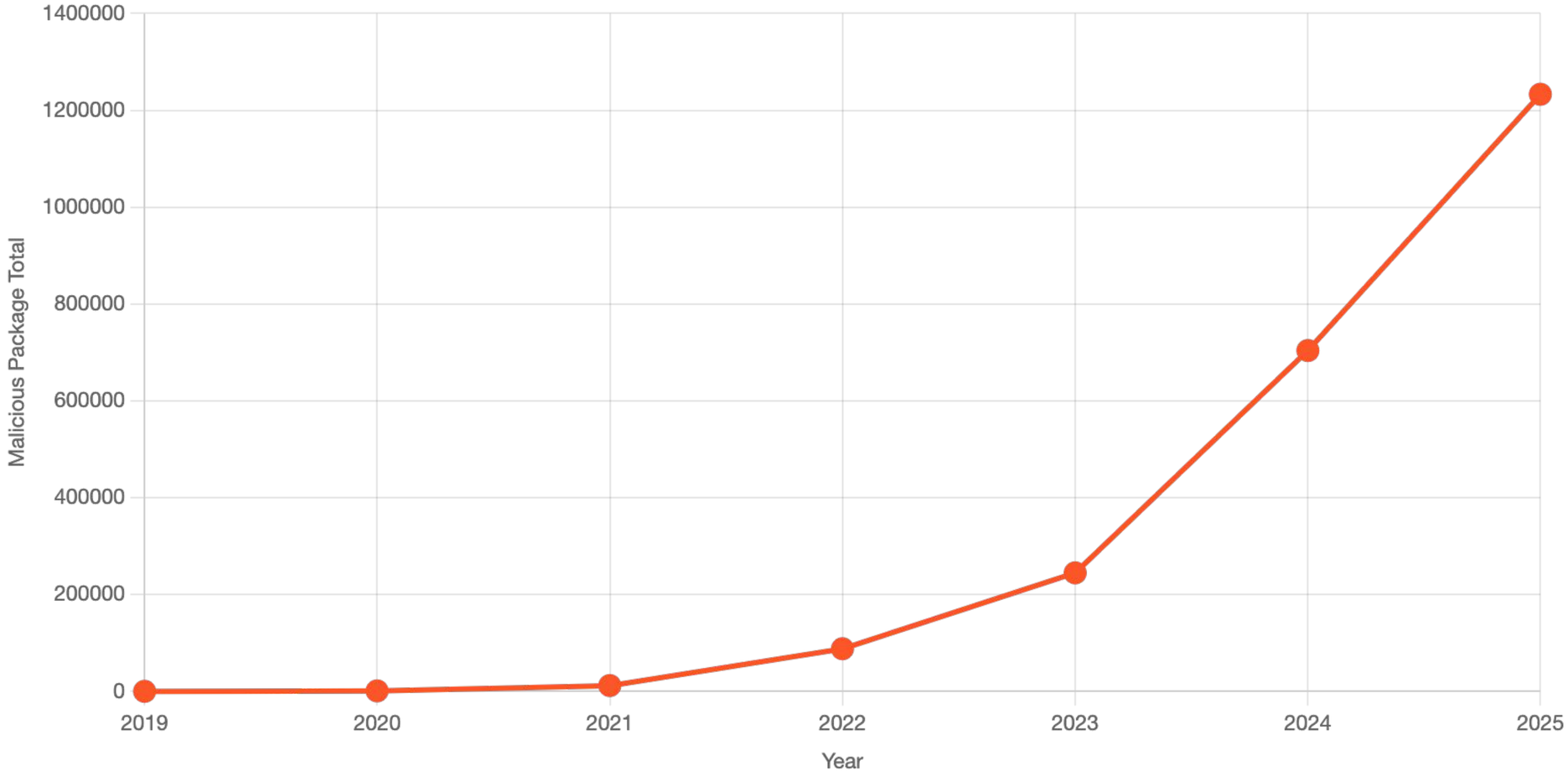
```yaml
- name: Configure AWS Credentials
  uses: aws-actions/configure-aws-credentials@v1
  with:
    aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
    aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
    aws-region: us-east-2
```

For example, you can use this action with the AWS CLI available in GitHub's hosted virtual environments. You can also run this action multiple times to use different AWS accounts, regions, or IAM roles in the same GitHub Actions workflow job.

OIDC 검증 체계 vs 공급망 사고



Annual Open Source Malware Growth (Sonatype)



발급은 하나, 검증은 제각각

같은 GitHub 토큰 · 각 RP의 Trust Policy가 보는 claim이 다르다

GitHub OIDC 토큰

```
{
  "iss": "token.actions.github...",
  "aud": "sts / pypi / npm",
  "sub": "repo:org/app:ref:main",
  "repository": "org/app",
  "workflow_ref": "publish.yml",
  "job_workflow_ref": "reusable.yml",
  "runner_environment": "hosted"
}
```

reusable를 쓰면 **caller(workflow_ref) ≠ called(job_workflow_ref)**

세 RP 모두 자신의 Trust Policy로 검증 — 보는 claim이 다르다

AWS STS

검증 `aud · sub`

repo·branch를 sub로 고정

PyPI

검증 `job_workflow_ref (called)`

실제 실행된 reusable을 확인

npm

검증 `workflow_ref (caller)`

호출한 워크플로우만 확인 (called 미검증)

누가 무엇을 검증하는가

각 RP의 Trust Policy · job_workflow_ref 한 claim으로 갈린다

GitHub 토큰 claim	AWS	PyPI	npm
`aud`	검증	검증	검증
`sub`	검증	—	—
`repository`	—	검증	검증
`workflow_ref` (caller)	—	—	검증
`job_workflow_ref` (called)	—	검증	미검증
`environment`	—	검증(옵션)	검증(옵션)

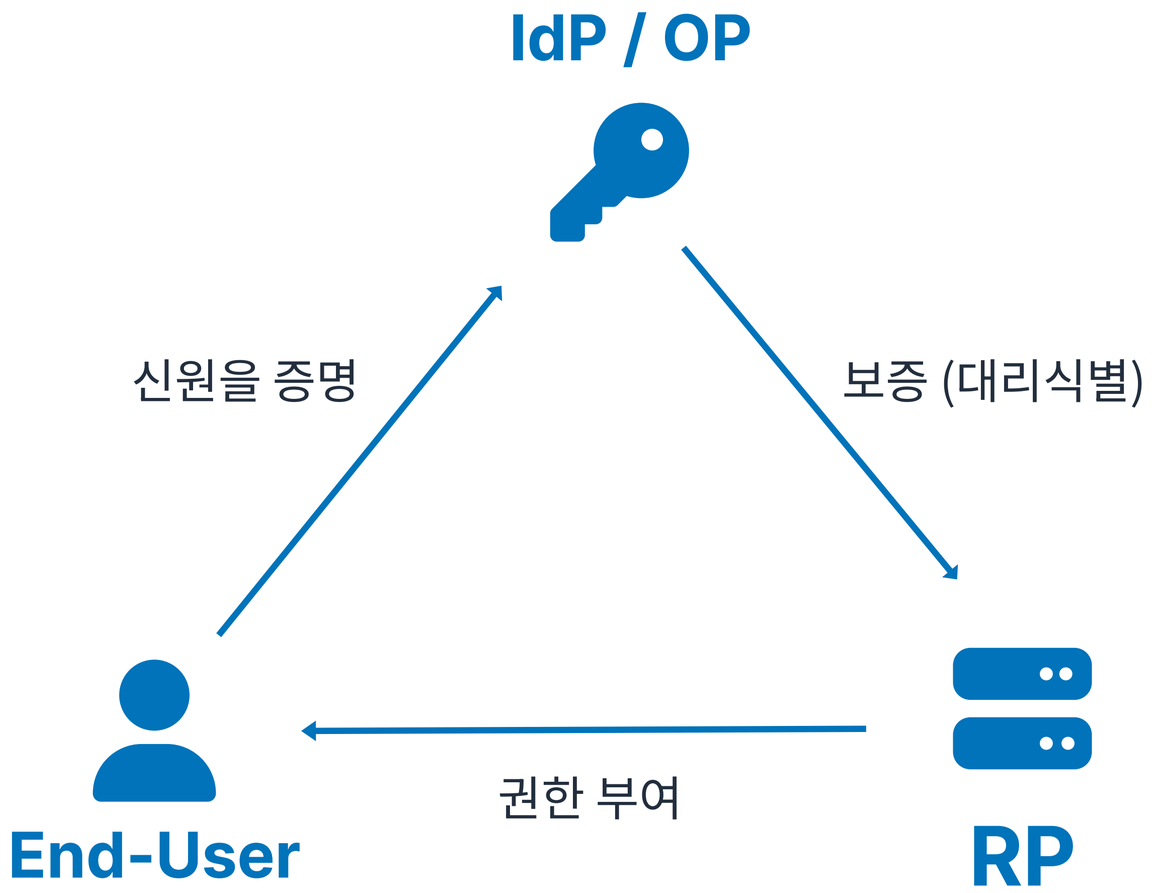
job_workflow_ref = 실제 실행된 reusable. npm은 caller만 보고 이건 안 본다 ↔ PyPI는 이것만 본다.

3

OIDC 의 공격 표면

신뢰 사슬 · 실행 주체 · RP 검증 · claim

OIDC의 신뢰가 보장되기 위한 3요소



신뢰가 새는 세 요인

실행 주체

워크플로우

claim

토큰 재료

RP 검증

trust policy

Workflow 위협 표면

Workflow 침해로 이어질 수 있는 공격 표면

PPE

`Poisoned Pipeline`

신뢰 컨텍스트에서
공격자 코드 실행

Unit42 OH-MY-DC
DEF CON 32

위험한 트리거

`pull_request_target`

fork PR가
base 권한으로 실행

Valsorda Compromise
Survey · 18건 중 5+

mutable 태그

`@v1 → 악성 재지정`

reusable · action을
SHA로 미고정

tj-actions CVE-2025-30066
23,000+ 리포

과도한 권한

`id-token: write`

permissions
광범위 노출

러너 토큰 탈취면 확대

Workflow 오염으로 인한 Runner 및 OIDC Token 탈취

이후 어떤 trust policy 검증도 무력화

RP 검증 과정에서의 위협 표면

잘못된 검증으로 인한 권한 상승 지점

서명 유효성 검증 오류

`iss / JWKS thumbprint`

잘못된 thumbprint · issuer
서명 검증을 우회

GitHub Changelog 2023-07
AWS thumbprint 문제

Claim 검증 오류

`StringLike sub: repo:org/*`

sub 미고정 · 와일드카드로
아무 workflow나 통과

Datadog · 500+ role
275+ AWS 계정

Token 재사용

`id-token 유출 → 재생`

만료 전 재사용 가능
RP는 일회성 미검증

tj-actions CVE-2025-30066
로그로 토큰 유출

직접적인 권한 상승 발생

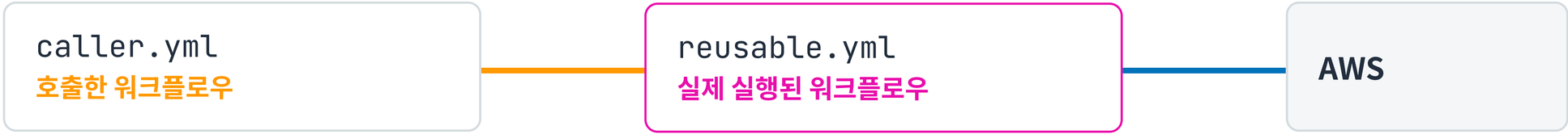
리스크 현실화

Reusable Workflow 에서의 Claim 검증 오류

Reusable Workflow 란?

일반 — 워크플로우가 직접 실행

reusable — 호출한 것과 실제 실행된 것이 다르다



일반 — OIDC 토큰

```
"workflow_ref":     "…/deploy.yml",
"job_workflow_ref": "…/deploy.yml"
```

reusable — OIDC 토큰

```
"workflow_ref":     "…/caller.yml",
"job_workflow_ref": "…/reusable.yml"
```

Reusable Workflow 에서의 Claim 검증 오류

같은 sub, 갈리는 검증

direct — assume-direct.yml

```
{
  "sub": "repo:...oidc-cases:ref:refs/heads/main",
  "aud": "sts.amazonaws.com",
  "workflow_ref":
     ".../assume-direct.yml@...main",
  "job_workflow_ref":
     ".../assume-direct.yml@...main"
}
```

via-reusable

```
{
  "sub": "repo:...oidc-cases:ref:refs/heads/main",
  "aud": "sts.amazonaws.com",
  "workflow_ref":
     ".../assume-via-reusable.yml@...main",
  "job_workflow_ref":
     ".../reusable-assume.yml@...main"
}
```

Reusable Workflow 에서의 Claim 검증 오류

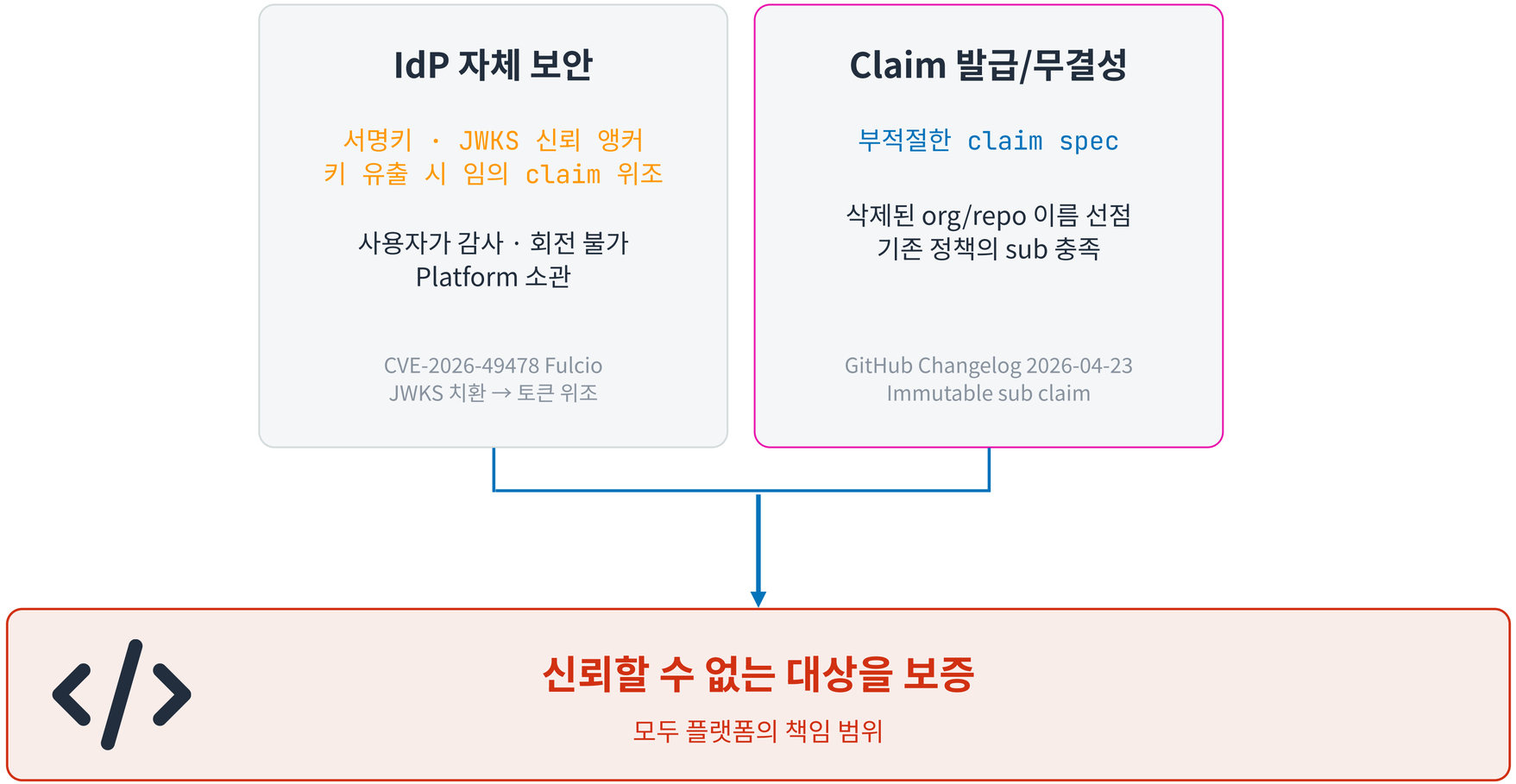
같은 sub, 갈리는 검증

```
▶ Run TOKEN=$(curl -sSf \
----- OIDC claims (audience=sts.amazonaws.com) -----
{
  "sub": "repo:3iuy-prog/oidc-cases:ref:refs/heads/main",
  "aud": "sts.amazonaws.com",
  "repository": "3iuy-prog/oidc-cases",
  "ref": "refs/heads/main",
  "workflow_ref": "3iuy-prog/oidc-cases/.github/workflows/assume-via-reusable.yml@refs/heads/main",
  "job_workflow_ref": "3iuy-prog/oidc-cases/.github/workflows/reusable-assume.yml@refs/heads/main"
}

▶ Run aws sts get-caller-identity
{
    "UserId": "AROAZROFFXQGV4FMYZDVP:GitHubActions",
    "Account": "655932898317",
    "Arn": "arn:aws:sts::655932898317:assumed-role/krug-oidc-a-safe/GitHubActions"
}
----- reading s3://krug-oidc-a-target-20260704085259827100000001/loot/fake-secret.txt -----
KRUG-LAB DUMMY SECRET — if you can read this via web identity, the pivot worked.
```

IdP 의 발행 과정에서의 위협 표면

잘못된 검증으로 인한 권한 상승 지점



ARC + EKS IRSA 환경에서의 OIDC Claim

IdP 의 Token Spec 으로 인한 공격 표면

Deploying Self-Hosted GitHub Runners on Kubernetes (EKS) with a Custom Docker Image

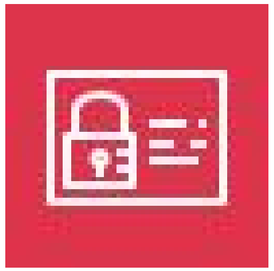
Daniel Kotev

07.05.2026

Step 8: Create an IAM Role and Service Account

Create an IAM role and attach it to a Kubernetes service account:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: runner-serviceaccount
  namespace: NAMESPACE
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::${accountid}:role/role-name
```



Github Action 에서 사용하던
Trust Policy 가 과연 동작할까?

ARC + EKS IRSA 환경에서의 OIDC Claim

Github Action 과 EKS IRSA 간 OIDC Token claim 스펙 차이

GitHub Actions

EKS self-hosted 러너

GitHub OIDC 토큰

```
{
  "iss": "token.actions.githubusercontent.com",
  "sub": "repo:3iuy-prog/oidc-cases:ref:…/main",
  "aud": "sts.amazonaws.com",
  "workflow_ref":      "…/assume-via-reusable.yml",
  "job_workflow_ref":  "…/reusable-assume.yml"
}
```

EKS IRSA 토큰

```
{
  "iss": "oidc.eks.ap-northeast-2…/id/…",
  "sub": "system:serviceaccount:arc-runners:arc-runner-sa",
  "aud": "sts.amazonaws.com",
  "kubernetes.io": { namespace · pod · sa }

  ✗  "workflow_ref"       →  없음
  ✗  "job_workflow_ref"   →  없음
}
```

ARC + EKS IRSA 환경에서의 OIDC Claim

Github Action 과 EKS IRSA 간 OIDC Token claim 스펙 차이

Github Action OIDC

```
----- (1) GitHub OIDC token — carries job_workflow_ref (Case A SAFE axis) -----
GITHUB_SUB: repo:3iuy-prog/oidc-cases:ref:refs/heads/main
GITHUB_WORKFLOW_REF: 3iuy-prog/oidc-cases/.github/workflows/irsa-transfer-attacker.yml@refs/heads/main
GITHUB_JOB_WORKFLOW_REF: 3iuy-prog/oidc-cases/.github/workflows/irsa-transfer-attacker.yml@refs/heads/main
```

EKS IRSA OIDC

```
----- (2) EKS IRSA (SA) token — NO job_workflow_ref, only sub=SA -----
IRSA_ISS: https://oidc.eks.ap-northeast-2.amazonaws.com/id/FD93E63E23D634F771A13A5807CFC80A
IRSA_SUB: system:serviceaccount:arc-runners:arc-runner-sa
IRSA_HAS_JOB_WORKFLOW_REF: NO
AWS_ROLE_ARN: arn:aws:iam::655932898317:role/krug-oidc-b-runner-irsa
```

ARC + EKS IRSA 환경에서의 OIDC Claim

“sub” 필드에서 식별되는 SA - Trust Policy 에서 식별되는 SA

IRSA OIDC Token

```
----- (2) EKS IRSA (SA) token — NO job_workflow_ref, only sub=SA -----
IRSA_ISS: https://oidc.eks.ap-northeast-2.amazonaws.com/id/FD93E63E23D634F771A13A5807CFC80A
IRSA_SUB: system:serviceaccount:arc-runners:arc-runner-sa
IRSA_HAS_JOB_WORKFLOW_REF: NO
AWS_ROLE_ARN: arn:aws:iam::655932898317:role/krug-oidc-b-runner-irsa
```

AWS Trust Policy

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
```

ARC + EKS IRSA 환경에서의 OIDC Claim

SA 단위로 “sub” 필드가 지정됨 이용해 SA 별 Workflow runner 운영

deploy-workflow-sa — A Workflow

```
{
  "aud": ["sts.amazonaws.com"],
  "iss": "https://oidc.eks.
    ap-northeast-2.amazonaws.com
    /id/AA52B6...",
  "sub": "system:serviceaccount:
    arc-runners:deploy-workflow-sa",
  "kubernetes.io": {
    "namespace": "arc-runners",
    "serviceaccount": {
      "name": "deploy-workflow-sa"
    }
  }
}
```

prod-deploy-sa — B Workflow

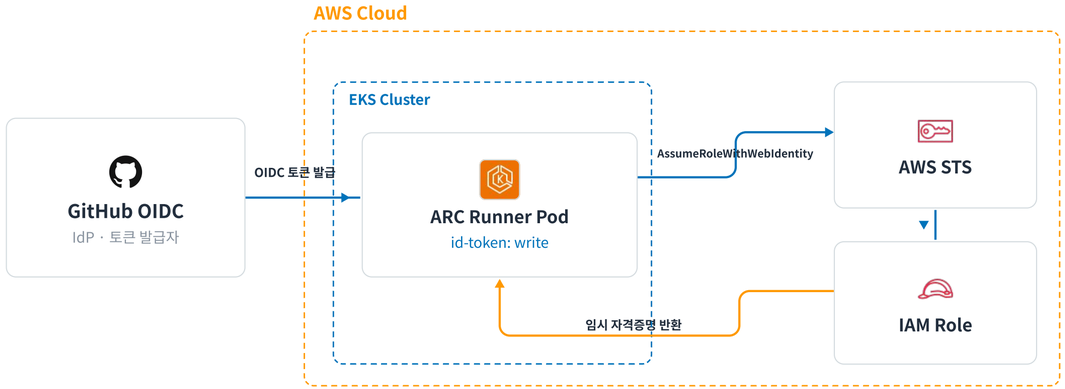
```
{
  "aud": ["sts.amazonaws.com"],
  "iss": "https://oidc.eks.
    ap-northeast-2.amazonaws.com
    /id/AA52B6...",
  "sub": "system:serviceaccount:
    arc-runners:prod-deploy-sa",
  "kubernetes.io": {
    "namespace": "arc-runners",
    "serviceaccount": {
      "name": "prod-deploy-sa"
    }
  }
}
```

readonly-sa — C Workflow

```
{
  "aud": ["sts.amazonaws.com"],
  "iss": "https://oidc.eks.
    ap-northeast-2.amazonaws.com
    /id/AA52B6...",
  "sub": "system:serviceaccount:
    arc-runners:readonly-sa",
  "kubernetes.io": {
    "namespace": "arc-runners",
    "serviceaccount": {
      "name": "readonly-sa"
    }
  }
}
```

ARC + EKS IRSA 환경에서의 OIDC Claim

처음부터 IRSA 자체가 ARC 에게는 적합한 IdP 가 아님



Deploying Self-Hosted GitHub Runners on Kubernetes (EKS) with a Custom Docker Image

Daniel Kotev

07.05.2026

사실은 ARC 또한 Github OIDC를 사용할 수 있으므로 Github OIDC Token 쓰면 해결되는 문제

OIDC flow 가 외부에서는 잘 보이지 않으므로 신뢰 체인을 고려하여 적절한 구조 선정 필요

SA 별로 Role을 나누면 Role 관리는 어떻게 하나요?

Role 관리는 어떻게 해야 해요?

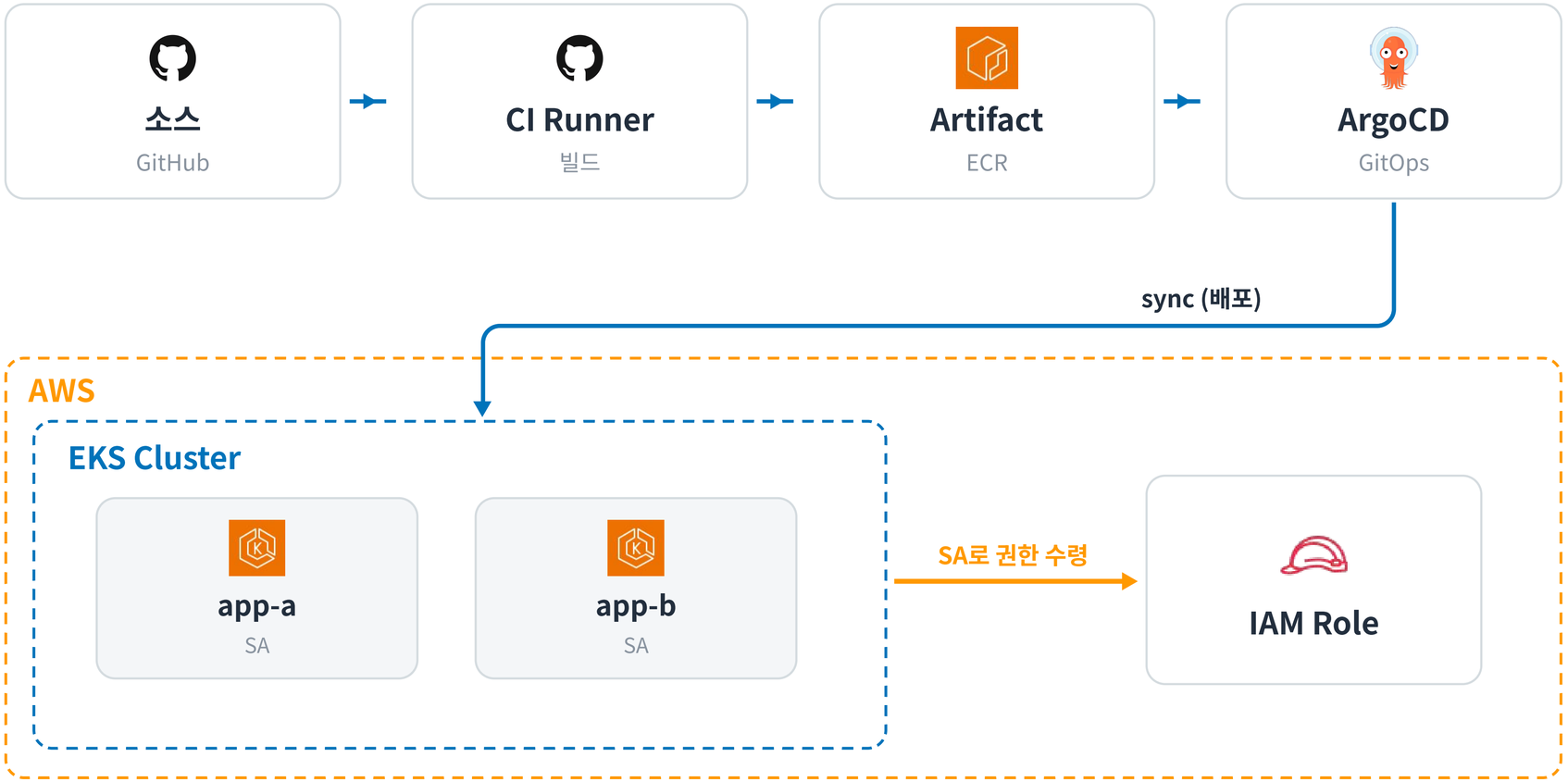
4

EKS IRSA 환경의 ArgoCD에서의 권한 관리

4부 · 여기까진 Best Practice

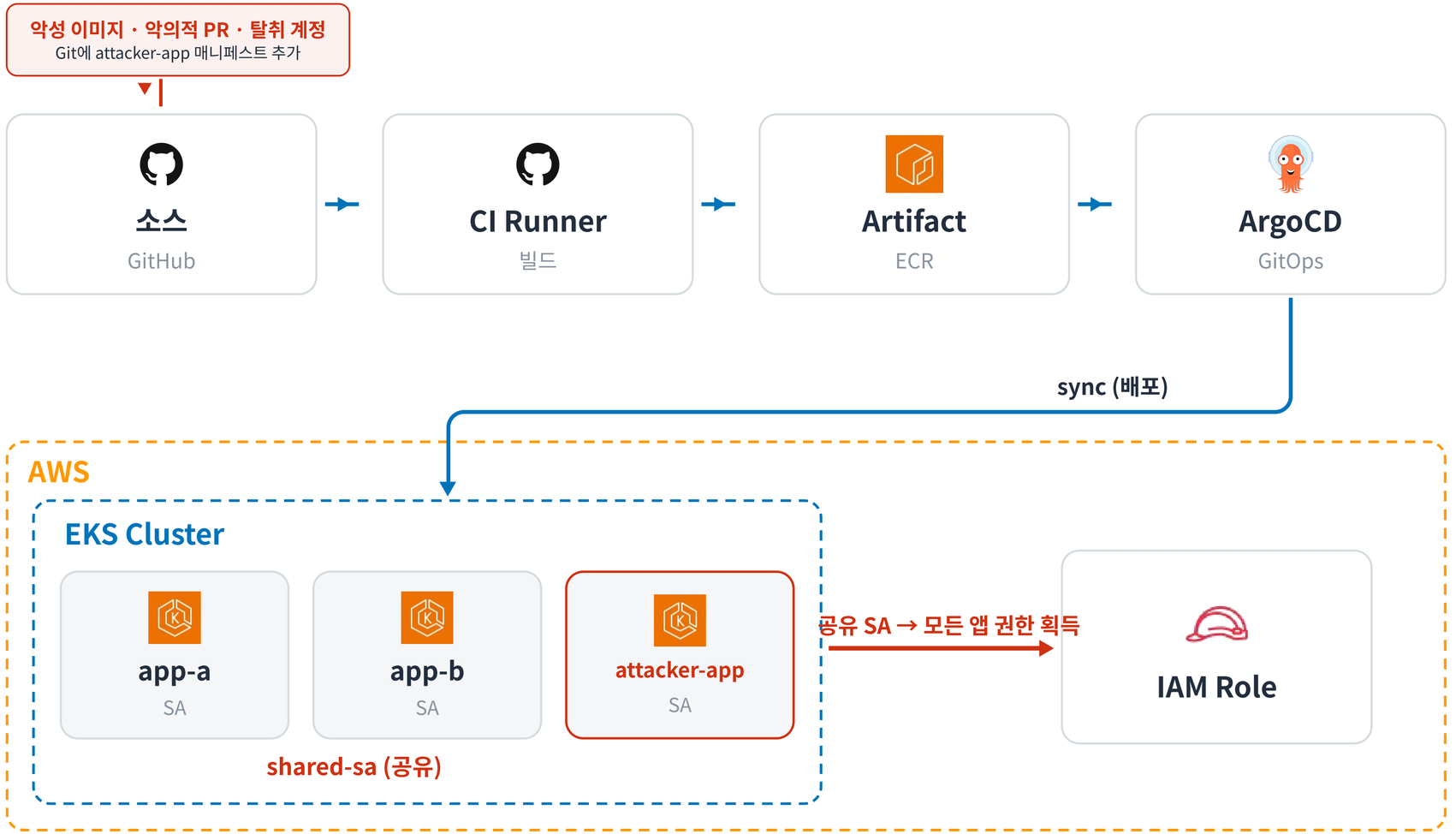
ArgoCD는 하나의 클러스터 위에서 여러 App들을 배포한다

앱 마다 배포하기 위한 권한이 달라지는데, 기본적인 IRSA 구성으로는 이를 대응할 수 없다



배포된 워크로드 하나가 오염되면

ArgoCD는 Git에 선언된 걸 그대로 배포 — 침해 워크로드도 공유 SA면 다른 앱 권한을 그대로 얻는다



공유 SA — 토큰도 정책도 같은 신원

attacker-app의 OIDC sub과 Trust Policy가 승인 워크로드와 동일

OIDC 토큰 (attacker-app)

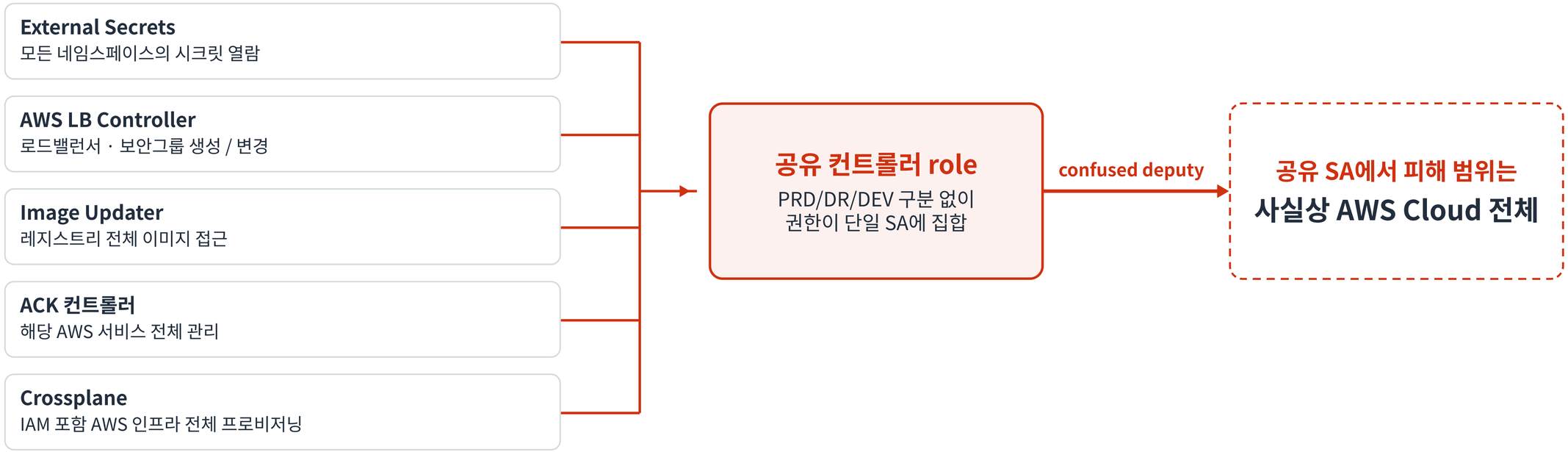
```
// attacker-app pod · OIDC token
{
  "aud": ["sts.amazonaws.com"],
  "iss": "https://oidc.eks..../id/<ID>",
  "kubernetes.io": {
    "namespace": "apps",
    "pod": { "name": "attacker-app-..." },
    "serviceaccount": {
      "name": "shared-sa"
    }
  },
  "sub": "system:serviceaccount:apps:shared-sa"
}
```

Trust Policy (공유 sub만 pin)

```
// role: krug-abac-s1-shared · Trust Policy
{
  "Effect": "Allow",
  "Principal": { "Federated":".../oidc-provider/
oidc.eks..../<ID>" },
  "Action": "sts:AssumeRoleWithWebIdentity",
  "Condition": {
    "StringEquals": {
      "...:aud": "sts.amazonaws.com",
      "...:sub": "system:serviceaccount:apps:shared-sa"
    }
  }
}
```

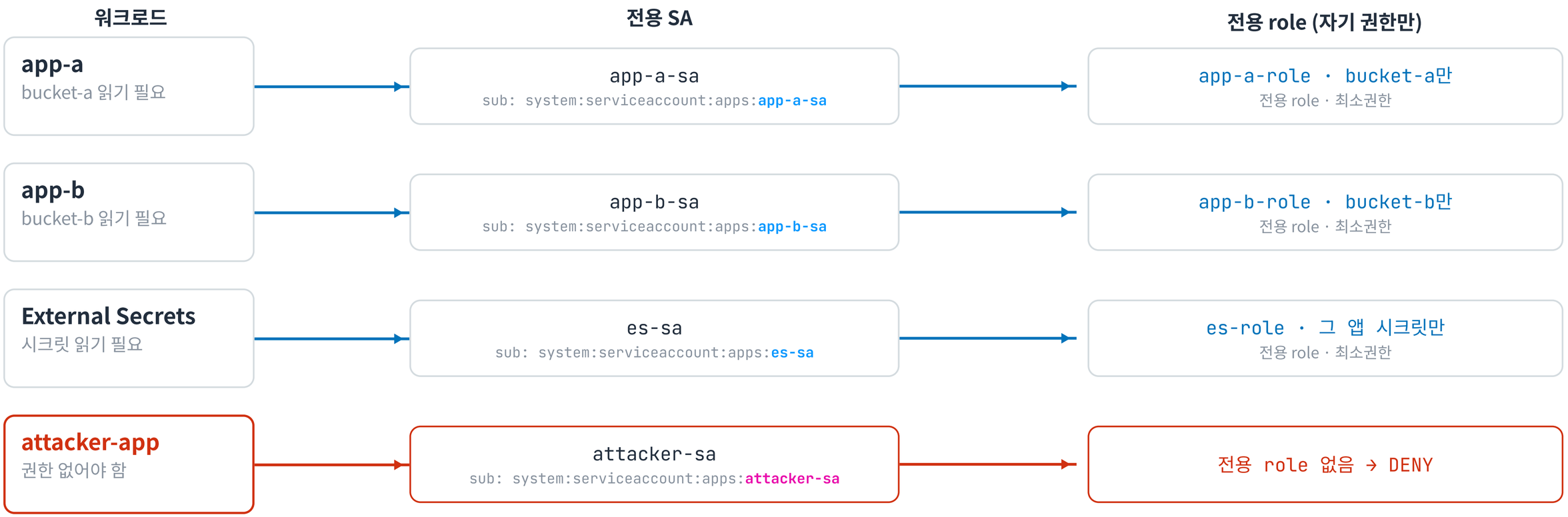
권한은 소수 컨트롤러에 쌓인다

ArgoCD 코어·애드온은 클러스터 전역을 컨트롤 하기 위한 모든 Role의 집합체가 된다



해결 — SA별로 나누면 피해가 격리된다

각 워크로드에 전용 SA · 전용 role · 자기 권한만 (1:1, 최소권한) — 하나 뚫려도 그 하나만



그런데 Role이 기하급수로 늘어난다

전용 SA = SA마다 전용 role. 앱당 3 role × 환경(DEV/STG/PRD) 3 = 앱 하나에 9 role

Application 수	Role 수 (×9)	상태
10	90	여유
50	450	여유
100	900	기본 한도 근접
120	1,080	기본 1,000 초과
500	4,500	위험
1,000	9,000	최대 10,000 근접

AWS 계정 한도

IAM role: 기본 1,000 / 최대 10,000

앱 ~120개면 기본 한도, ~1,100개면 상한

진짜 벽은 role 수가 아니다

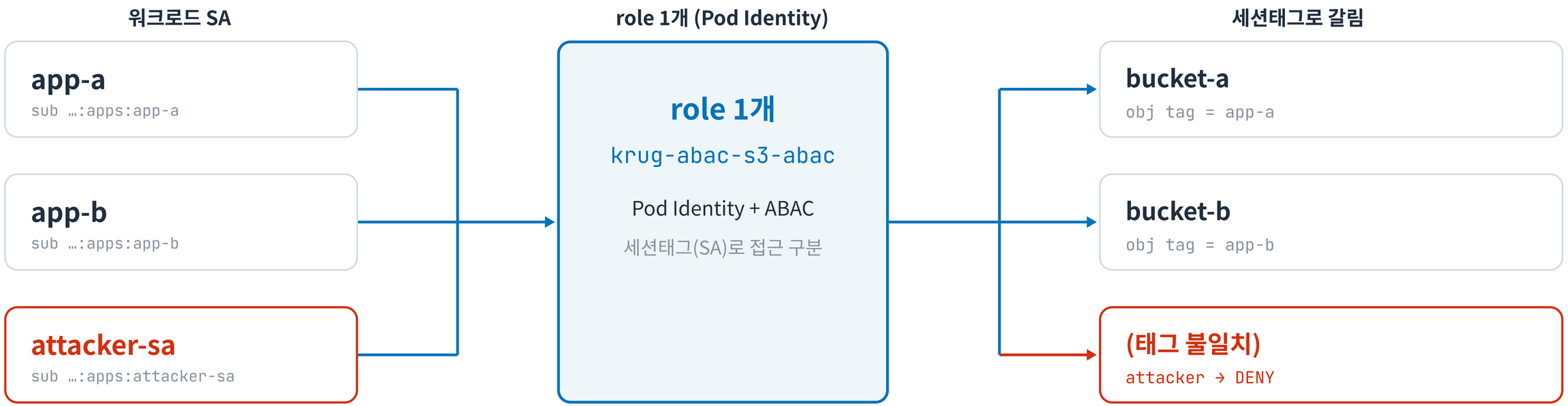
Trust policy 크기(8,192자) · 계정당 OIDC provider(700개)가 먼저 터진다

가정

앱당 전용 role 3개 (앱 · External Secrets · 잡)
× 환경 3개 (DEV / STG / PRD) = 앱 하나에 9 role

해소 — role 하나 + ABAC 세션태그로 흡수

세 SA를 같은 role에 Pod Identity association. 접근은 세션태그(SA 이름)로만 구분 → role 폭증 해소



SA가 “sub” 값이므로, SA로 ABAC 정책 부여

SA 값으로 접근 가능한 리소스가 달라지도록 ABAC 정책 설계

ABAC 권한 정책

```
// role: krug-abac-s3-abac · GetObject
"Condition": { "StringEquals": {
  // ① SA 태그 = 객체 태그  (매칭 선언)
  "s3:ExistingObjectTag/kubernetes-service-account":
      "${aws:PrincipalTag/kubernetes-service-account}",

  // ② 동명 SA 스푸핑 방지  (반드시 함께 pin)
  "aws:PrincipalTag/kubernetes-namespace": "apps",
  "aws:PrincipalTag/eks-cluster-arn":
      "arn:aws:eks:...:cluster/krug-oidc-b"
}}
```

판정 (role 1개 · 세션태그로 갈림)

워크로드	bucket-a	bucket-b
app-a	ALLOW	DENY
app-b	DENY	ALLOW
attacker-sa	DENY	DENY

무엇이 자동이고, 무엇을 선언하나

자동 (EKS가 함)

- SA·NS·cluster 태그 주입
- pod 위조 불가
- claim→태그 매핑

선언 (직접 작성)

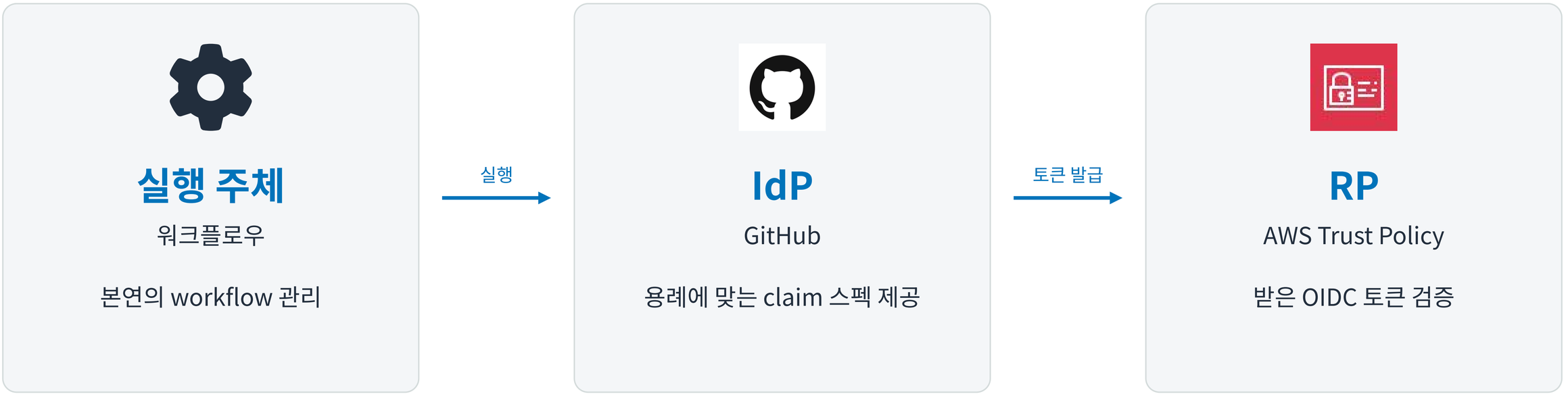
- trust: sts:TagSession
- 매칭 Condition
- NS + cluster-arn pin

5

정리 및 대응 방안

결국 핵심 참여자 모두가 함께 잘해야 유지될 수 있는 신뢰 체인

완전 방어가 불가능하다는 현실을 여전



내가 관리하는 Repository 부터 관리 철저

실행 주체 Level 에서의 신뢰

Workflow 보호

`Poisoned Pipeline`

신뢰 컨텍스트에서
공격자 코드 실행

Unit42 OH-MY-DC
DEF CON 32

PR 트리거 규칙 강화

`pull_request_target`

fork PR가
base 권한으로 실행

Valsorda Compromise
Survey · 18건 중 5+

Hash Pinning

`@v1 → 악성 재지정`

reusable · action을
SHA로 미고정

tj-actions CVE-2025-30066
23,000+ 리포

최소 권한 원칙

`id-token: write`

permissions
광범위 노출

러너 토큰 탈취면 확대

Trust Policy 강화

RP Level 에서의 신뢰


```
{
  "typ": "JWT",
  "alg": "RS256",
  "x5t": "example-thumbprint",
  "kid": "example-key-id"
}
{
  "jti": "example-id",
  "sub": "repo:octo-org/octo-repo:environment:prod",
  "environment": "prod",
  "aud": "https://github.com/octo-org",
  "ref": "refs/heads/main",
  "sha": "example-sha",
  "repository": "octo-org/octo-repo",
  "repository_owner": "octo-org",
  "actor_id": "12",
  "repository_visibility": "private",
  "repository_id": "74",
  "repository_owner_id": "65",
  "run_id": "example-run-id",
  "run_number": "10",
  "run_attempt": "2",
  "runner_environment": "github-hosted",
  "actor": "octocat",
  "workflow": "example-workflow",
  "head_ref": "",
  "base_ref": "",
  "event_name": "workflow_dispatch",
  "repo_property_workspace_id": "ws-abc123",
  "ref_type": "branch",
  "job_workflow_ref": "octo-org/octo-automation/.github/workflows/oidc.yml@refs/
heads/main",
  "iss": "https://token.actions.githubusercontent.com",
  "nbf": 1632492967,
  "exp": 1632493867,
  "iat": 1632493567
}
```


```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Federated": "arn:aws:iam::<ACCOUNT_ID>:oidc-provider/
token.actions.githubusercontent.com"
        },
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {
          "StringEquals": {
            "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
            "token.actions.githubusercontent.com:repository": "my-org/my-repo",
            "token.actions.githubusercontent.com:repository_owner_id": "123456789",
            "token.actions.githubusercontent.com:job_workflow_ref":"my-org/my-
repo/.github/workflows/deploy.yml@refs/heads/main",
            "token.actions.githubusercontent.com:job_workflow_sha":
"e3b0c44298fc1c149afbf4c8996fb92427ae41e4",
            "token.actions.githubusercontent.com:runner_environment": "github-hosted",
            "token.actions.githubusercontent.com:environment": "production"
          },
          "StringLike": {
            "token.actions.githubusercontent.com:sub": "repo:my-org/my-repo:*"
          }
        }
      }
    ]
  }
```

최신 OIDC 관련 GA에 주목하기

IdP Level 는 인증 공급자의 최신 정책 변경 사항 참조

GitHub — IdP claim 강화

Immutable sub claim

2026-04-23 발표 · 07-15↑ 신규 레포 자동

sub에 불변 owner·repo ID 삽입 — 조직명 재사용 탈취 차단. 15일 시점 신규 레포부터 켜짐.

repo_property_* claim

PREVIEW

2026-03-12 public preview

레포 custom property를 OIDC claim으로 자동 주입 — 속성 기반 크로스클라우드 접근.

Immutable Actions / Releases

Releases GA · Actions PREVIEW

Releases GA 2025-10 · Actions preview

Release 자산·태그 잠금은 GA. action을 OCI로 불변 게시하는 Immutable Actions는 아직 preview.

AWS — RP 검증 강화

STS: GitHub claim native 검증

GA

2026-02-02 GA (전 상용 리전)

job_workflow_ref 등 13개 claim을 trust policy condition key로 직접 검증. sub 접기 불필요.

Pod Identity: Session Policies

GA

2026-03-24 GA

association마다 권한 축소(role∩session). 단 session tag와 상호배타 — ABAC 쓰면 못 씀.

Pod Identity: 크로스계정 · ABAC

GA

2025-06 크로스계정 · ABAC 6태그

IRSA엔 없는 native session tag(ns·sa·pod 6종)로 ABAC. targetRoleArn 크로스계정 체이닝.

출처: AWS What's New · IAM condition-keys-wif · github.blog changelog · github/roadmap (2026-07 기준)

감사합니다.

Q&A