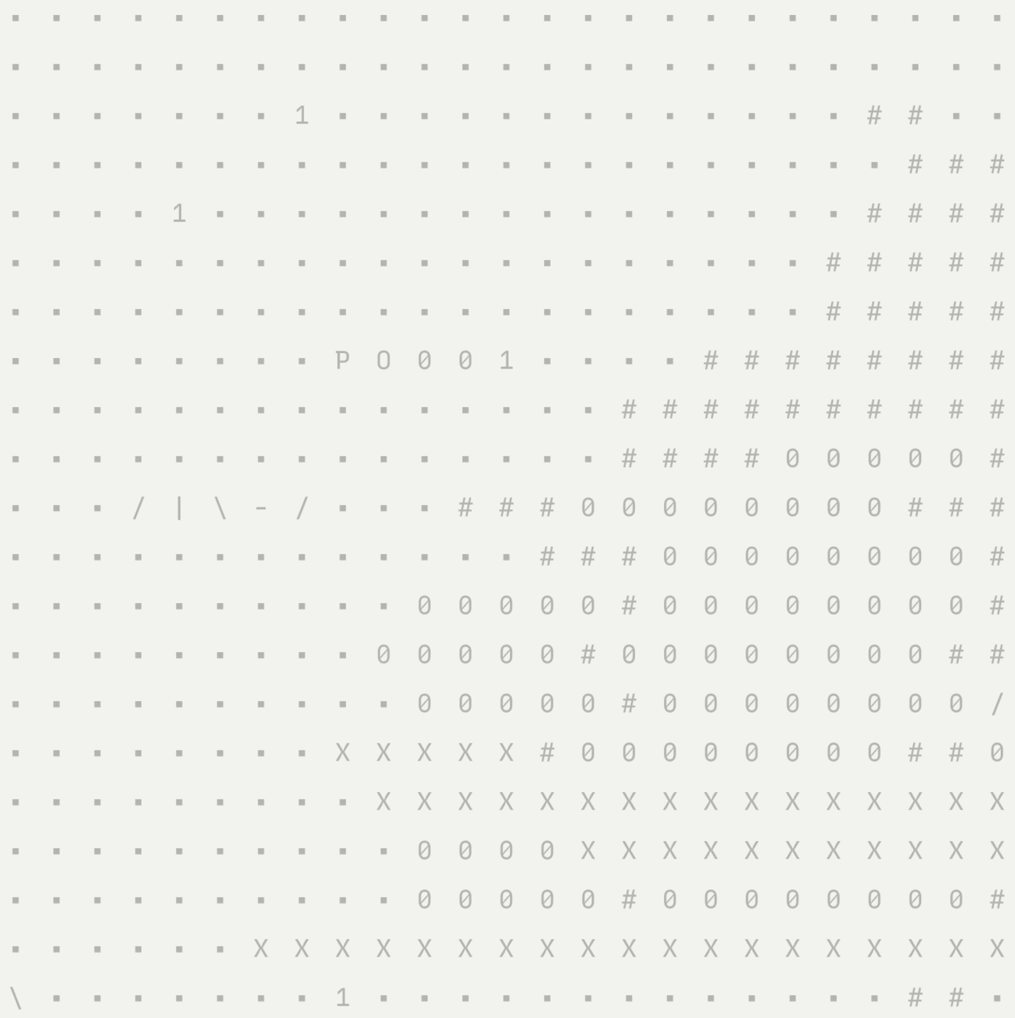
stedi

Why Security Pros Should Embrace the AWS CDK!

Dakota Riley



About Stedi

- SaaS startup
- Fully distributed team
- Fully serverless

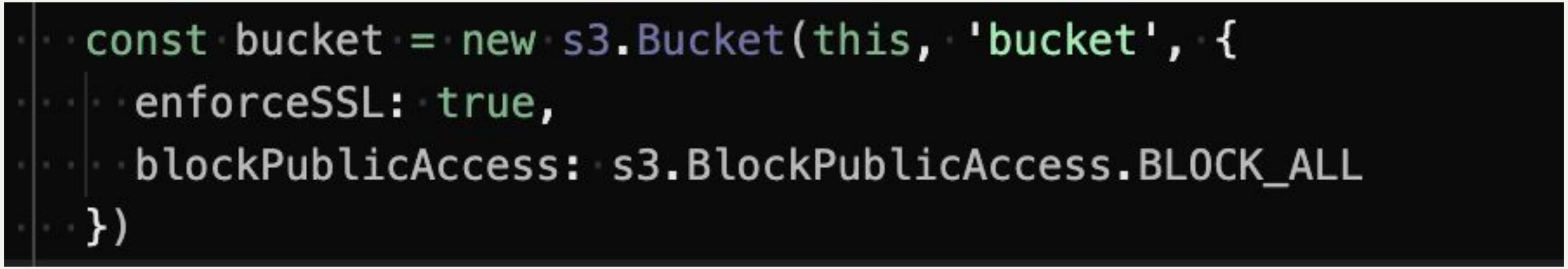
Why security teams should love the AWS CDK

- Makes it easier to develop least privileged IAM policies
- Can create flexible constructs for Dev teams (and yourselves!)
- Can use your existing static analysis tooling

L2 Constructs

- High Level Abstractions of AWS Resources
- Remove the need to worry about “boilerplate”
- Expose various methods for manipulating the resource

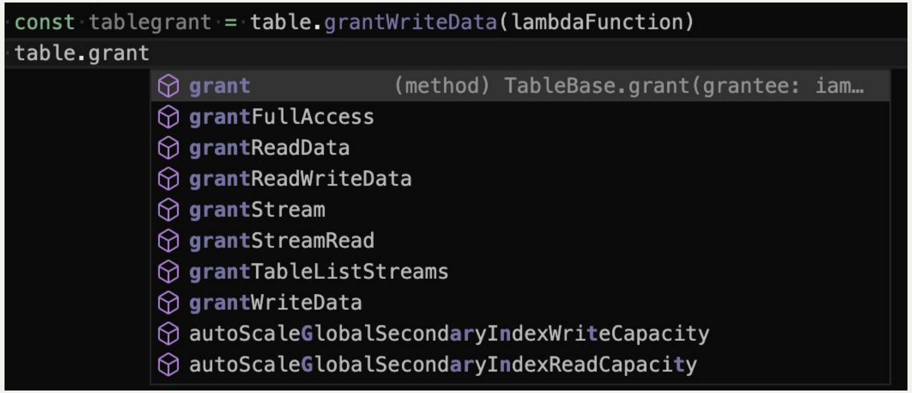
Examples: s3.Bucket(), lambda.function(), secretsmanager.secret()



```
  const bucket = new s3.Bucket(this, 'bucket', {
    enforceSSL: true,
    blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL
  })
```

L2 Constructs and IAM

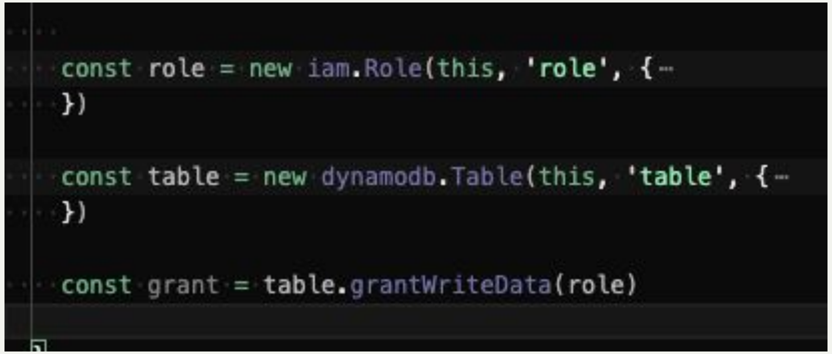
- Most L2 Constructs provide an elegant abstraction for IAM via methods
- Dozens of Access Patterns Captured
- At your fingertips - docstrings and intellisense



```
const tablegrant = table.grantWriteData(lambdaFunction)
table.grant
        grant                  (method) TableBase.grant(grantee: iam…
        grantFullAccess
        grantReadData
        grantReadWriteData
        grantStream
        grantStreamRead
        grantTableListStreams
        grantWriteData
        autoScaleGlobalSecondaryIndexWriteCapacity
        autoScaleGlobalSecondaryIndexReadCapacity
```

L2 Constructs and IAM

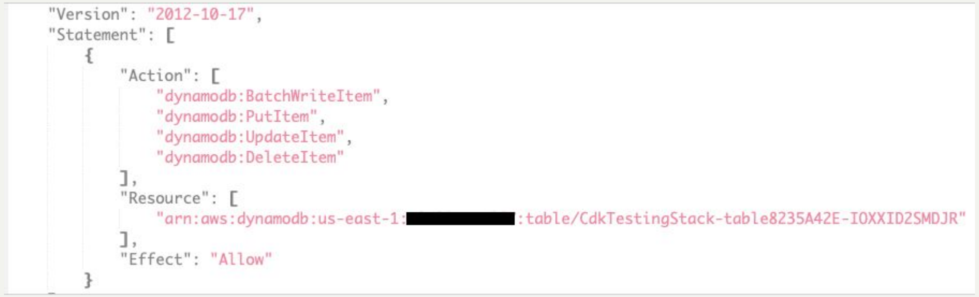
Before vs After



```
const role = new iam.Role(this, 'role', {⋯
})

const table = new dynamodb.Table(this, 'table', {⋯
})

const grant = table.grantWriteData(role)
```



```
"Version": "2012-10-17",
"Statement": [
    {
        "Action": [
            "dynamodb:BatchWriteItem",
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb:DeleteItem"
        ],
        "Resource": [
            "arn:aws:dynamodb:us-east-1:[redacted]:table/CdkTestingStack-table8235A42E-IOXXID2SMDJR"
        ],
        "Effect": "Allow"
    }
```

L2 Constructs and IAM

Out of 467 Lambda Function Roles created by the CDK in our Production Environment (specifically looking at services that store/access data - s3, dynamodb, secretsmanager)

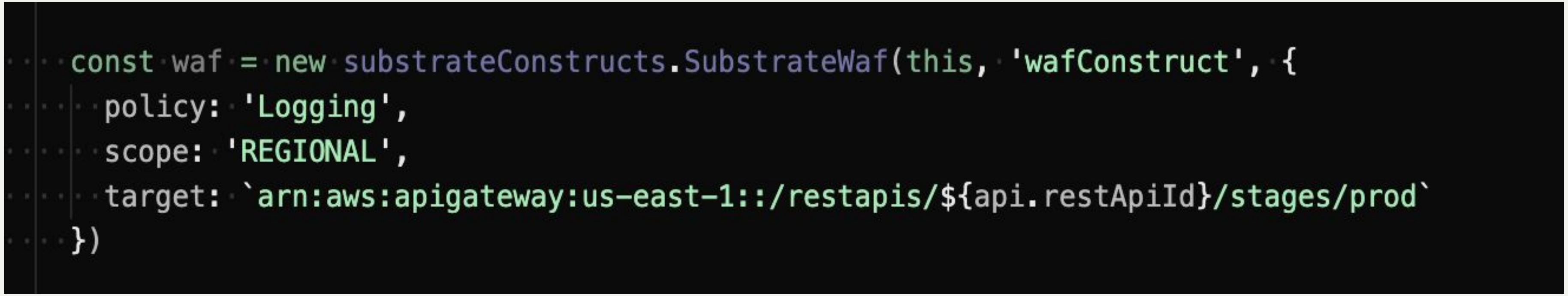
- Only 12 roles weren’t using specific s3 actions (s3:*)
- Only 3 roles weren’t using specific dynamodb actions (dynamodb:*)
- 0 roles were using secretsmanager:*
- Overwhelming majority were using both specific actions and resource level permissions
- All Roles had at least resource level permissioning

Everyone Gets A WAF!

- Asked Service Teams to protect public facing API Gateway endpoints
- Ran into two problems:
    - Complexity
    - No L2 Construct

Everyone Gets A WAF!

- Developed a WAF Construct using https://asecure.cloud/a/AWS_WAF_Common as inspiration
- baseline rulesets
- Escape hatches/overrides
- Docstrings!



```
    const waf = new substrateConstructs.SubstrateWaf(this, 'wafConstruct', {
      policy: 'Logging',
      scope: 'REGIONAL',
      target: `arn:aws:apigateway:us-east-1::/restapis/${api.restApiId}/stages/prod`
    })
```

Static Analysis - Use your current tools!

- Tools that support cloudformation also still supported for the AWS CDK
- Open Source Tools
  - CfnGuard (https://github.com/aws-cloudformation/cloudformation-guard)
    - Implemented in Rust - lightning fast
    - Rules written in a shorthanded DSL
    - Ability to generate rules from a given cfn template
  - Checkov (https://github.com/bridgecrewio/checkov)
    - Rules written in Python
    - ~80 Cloudformation checks included from the start
- Great as a local tool or a deploy time check

Conclusion

- L2 Constructs make it easy to “Do the right thing” for IAM and general security configurations
- We meet the Developer Teams where they are by embracing tooling they like, and writing constructs that make security requirements easy to meet

Thanks for watching

Feel free to reach out with questions!

Email: Dakota@Stedi.com
LinkedIn: https://www.linkedin.com/in/dakota-riley-b48401b7
GitHub: https://www.github.com/rileydakota