
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SEC310

Security alchemy: How AWS uses math to prove security

Brigid Johnson (she/her)
GM of IAM Access Analyzer
AWS

Neha Rungta (she/her)
Director in AWS Identity
AWS

Let’s start with a story

Once upon a time in AWS…

we were in search for something magical

something that could tell us with certainty

if security configurations were right.

Security alchemy: The rubric

Higher security

The apothecary

Automated reasoning & comprehensive analysis

Transfigurations

Brewing potions

Security charms in AWS

Higher security

Growing security controls

Network

Control access to inbound and outbound traffic

Access

Control which identities can access actions and resources under specific conditions

Data encryption

Layer of security to encrypt data at rest and in transit

Security control examples

Service control policies (SCPs)

Identity-based policies

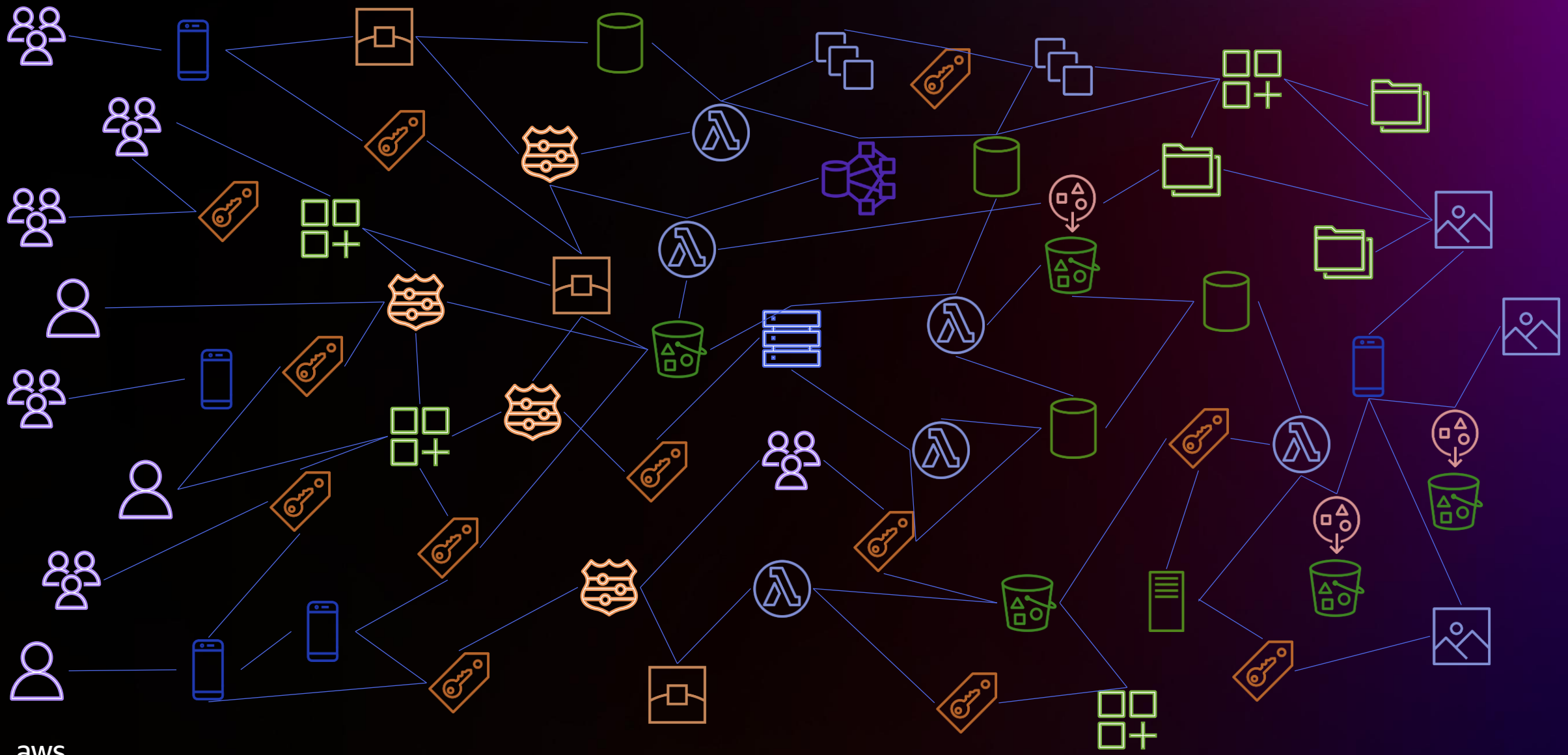
Resource-based policies

Endpoint policies

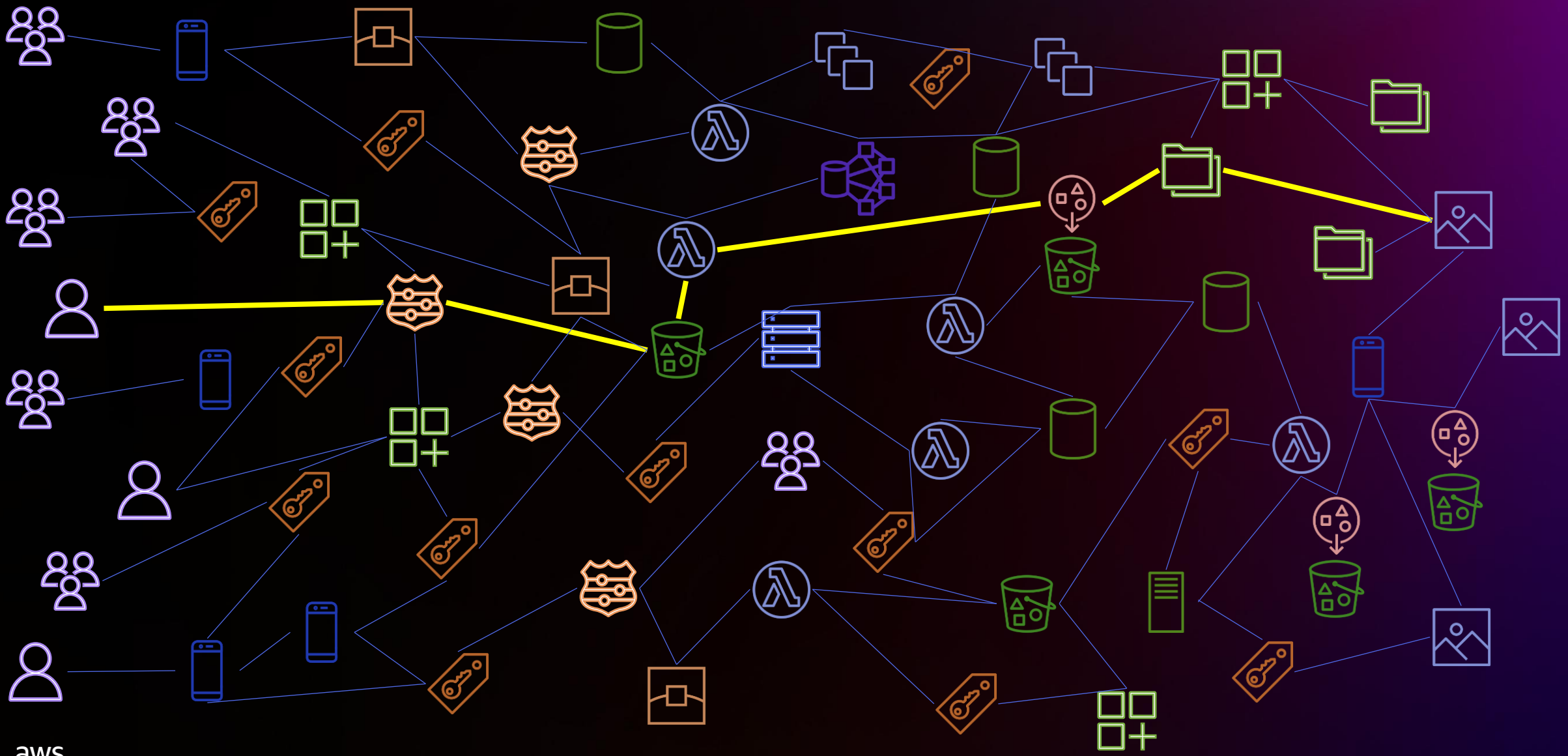
Block public access (BPA) with Amazon S3

Service-specific controls such as AWS Resource Access Manager

Access paths of systems



Determine if there is a path



The apothecary

Your shopping list

Universal statements in comprehensive analysis section

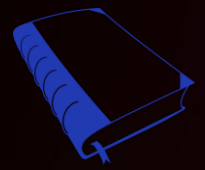
AWS security controls

Security configurations and resource metadata

Transformer potion

Solver potion

Compute

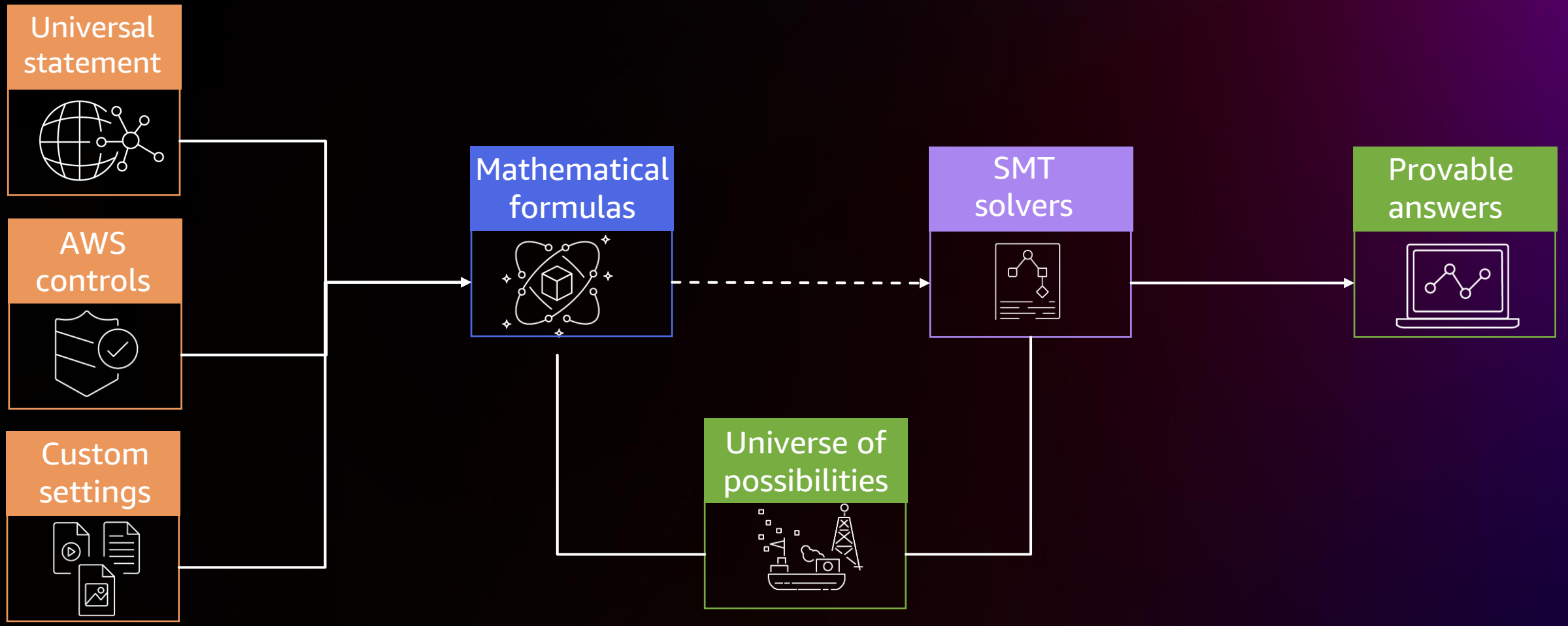


Automated reasoning potions textbook

Automated reasoning and comprehensive analysis

Introduction to automated reasoning

Collect → Transform → Solve → Comprehensive



Comprehensive as a science

Muggle	Question	Configurations	Comprehensive
Scientist	Property	All possible paths	Proof of one path
Security alchemist	Universal statement 	Universe of possibilities 	Provable answers 

Universal statements

A statement that always holds

A statement you can prove

If it doesn’t hold, there is a reason

Universal statement examples

"All my data at rest is encrypted"

"There is no public access to my S3 buckets"

"No credentials are logged"

"There are no deadlocks in my code"

AWS controls (the knobs)

Fine-grained security controls that have a defined behavior

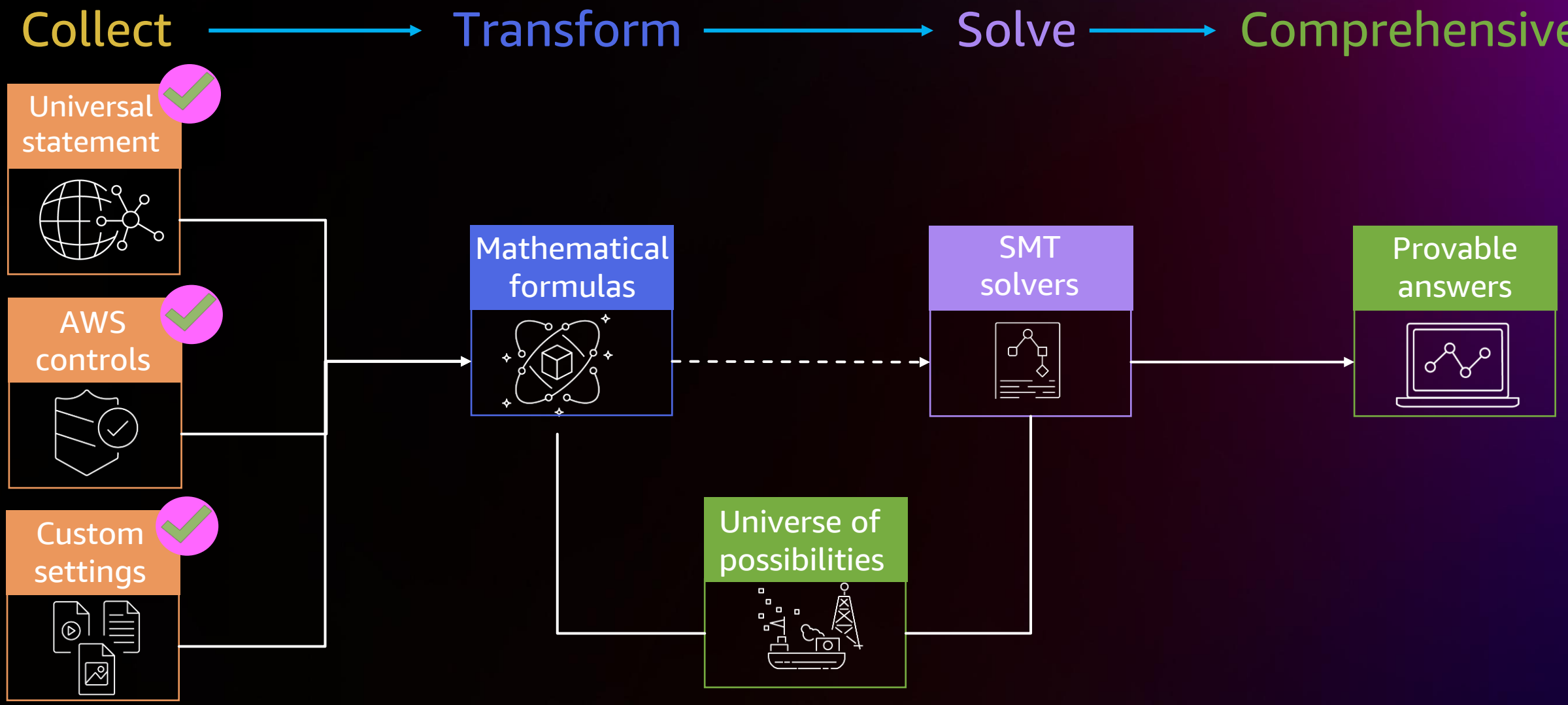
`Deny` statement in policies	Access restricted for the action, resource-specified, and conditions met
Assume role policy	Cross-account access allowed to set of accounts specified in **`Principal`** element—and no others
Block Public Access (BPA)	Restricts public access for all new buckets

Customer security configurations

Established security controls based on customer intent

Resource policy on an S3 Access Point		Allow access to only `Principal` elements within `organization-witches`—and no others
IAM policies with conditions		Deny access to `Principal` elements outside of `organization-witches`
Block Public Access (BPA)		Restricts public access for all buckets in the account `broom-dance`, regardless of the policy attached to the bucket

Ready to transform



Transfigurations

Encoding AWS security controls into math

Semantic-based Automated Reasoning for AWS Access Policies using SMT

John Backes, Pauline Bolignano, Byron Cook, Catherine Dodge, Andrew Gacek,
Kasper Luckow, Neha Rungta, Oksana Tkachuk, Carsten Varming
Amazon Web Services

***Abstract*—Cloud computing provides on-demand access to IT resources via the Internet. Permissions for these resources are defined by expressive access control policies. This paper presents a formalization of the Amazon Web Services (AWS) policy language and a corresponding analysis tool, called ZELKOVA, for verifying policy properties. ZELKOVA encodes the semantics of policies into SMT, compares behaviors, and verifies properties. It provides users a sound mechanism to detect misconfigurations of their policies. ZELKOVA solves a PSPACE-complete problem and is invoked many millions of times daily.**

I. INTRODUCTION

Cloud computing provides on-demand access to IT resources via the Internet. The convenience of accessing resources in the cloud is made secure by user-specified *access control policies*. An access control policy is an expressive specification of what resources can be accessed, by whom,

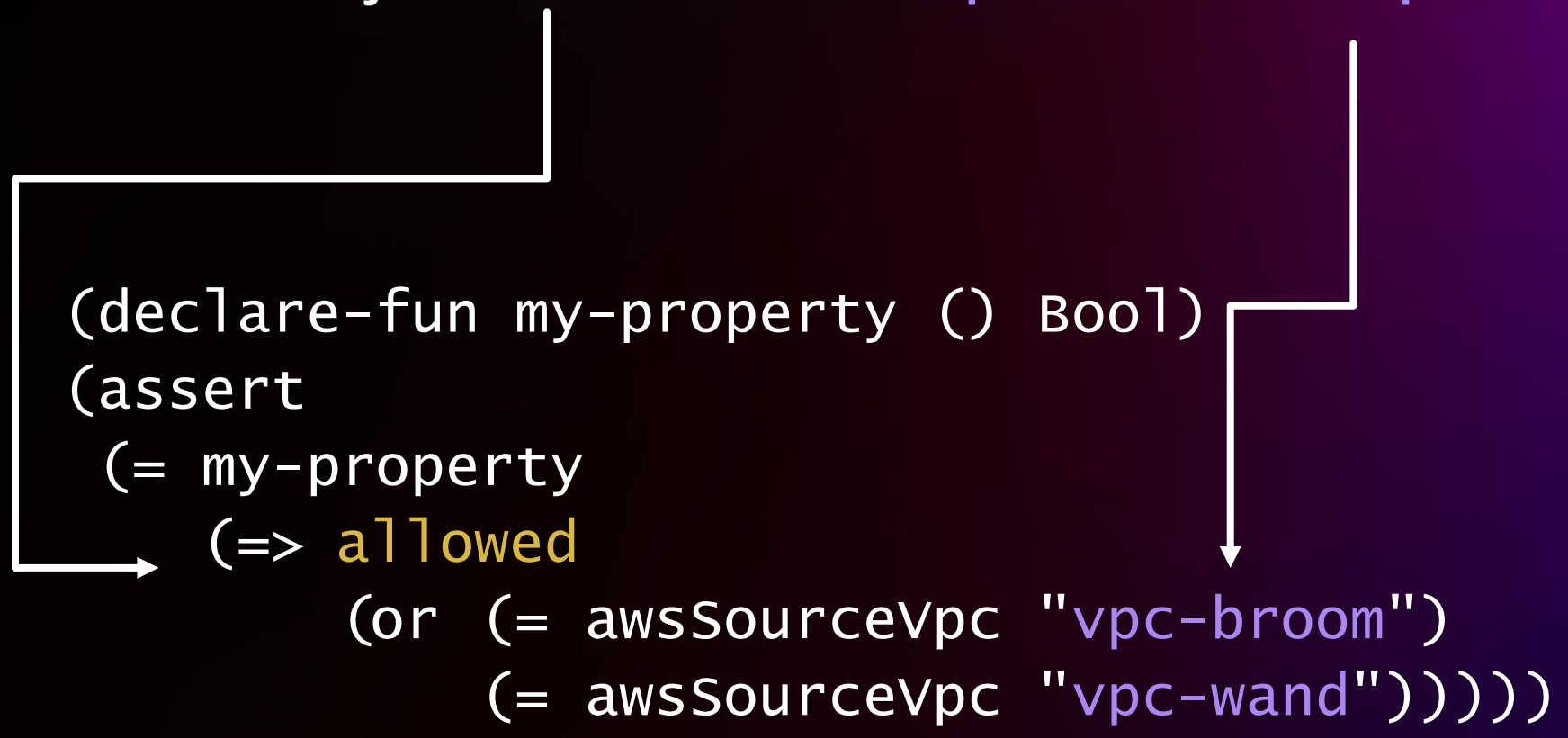
In this paper, we present the development and application of ZELKOVA, a policy analysis tool designed to reason about the semantics of AWS access control policies. ZELKOVA translates policies and properties into Satisfiability Modulo Theories (SMT) formulas and uses SMT solvers to check the validity of the properties. We use off-the-shelf solvers and an in-house extension of Z3 called Z3AUTOMATA.

ZELKOVA reasons about all possible permissions allowed by a policy in order to verify properties. For example, ZELKOVA can answer the questions "Is this resource accessible by a particular user?" and "Can an arbitrary user write to this resource?". The property to be verified is specified in the policy language itself, eliminating the need for a different specification or formalism for properties. In addition, ZELKOVA provides many built-in checks for common properties.

The SMT encoding uses the theory of strings, regular

From muggle to math

My bucket magic-potions is only accessible inside vpc-broom or vpc-wand



```
(declare-fun my-property () Bool)
(assert
 (= my-property
    (=> allowed
        (or (= awsSourceVpc "vpc-broom")
            (= awsSourceVpc "vpc-wand")))))
```

From policy to math

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "11112222333"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::magic-potions/*"
},
{
  "Effect": "Deny",
  "Principal": "*",
  "Action": "*",
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "aws:SourceVpc": ["vpc-broom", "vpc-wand"]
    }
  }
}
```

AND

NOT

OR

```
(declare-fun principal () String)
(declare-fun action () String)
(declare-fun resource () String)
(declare-fun awsSourceVpc () String)

(declare-fun s1-match () Bool)
(assert
 (= s1-match
  ( and
   (= principal "111122223333")
   (= action "s3:GetObject")
   (str.in_re resource
              (re.++
               (str.to_re "arn:aws:s3:::magic-potions/")
               re.all)))))

(declare-fun s2-match () Bool)
(assert
 (= s2-match
  ( not( or(= awsSourceVpc "vpc-broom")
           (= awsSourceVpc "vpc-wand")))))

(declare-fun allowed () Bool)
(assert
 (= allowed
    (and s1-match
         (not s2-match))))
```

At least one allow matches, and no denies match

Universe of possibilities

Model of all potential scenarios

Coverage is complete

Includes future scenarios

Universe of possibilities example

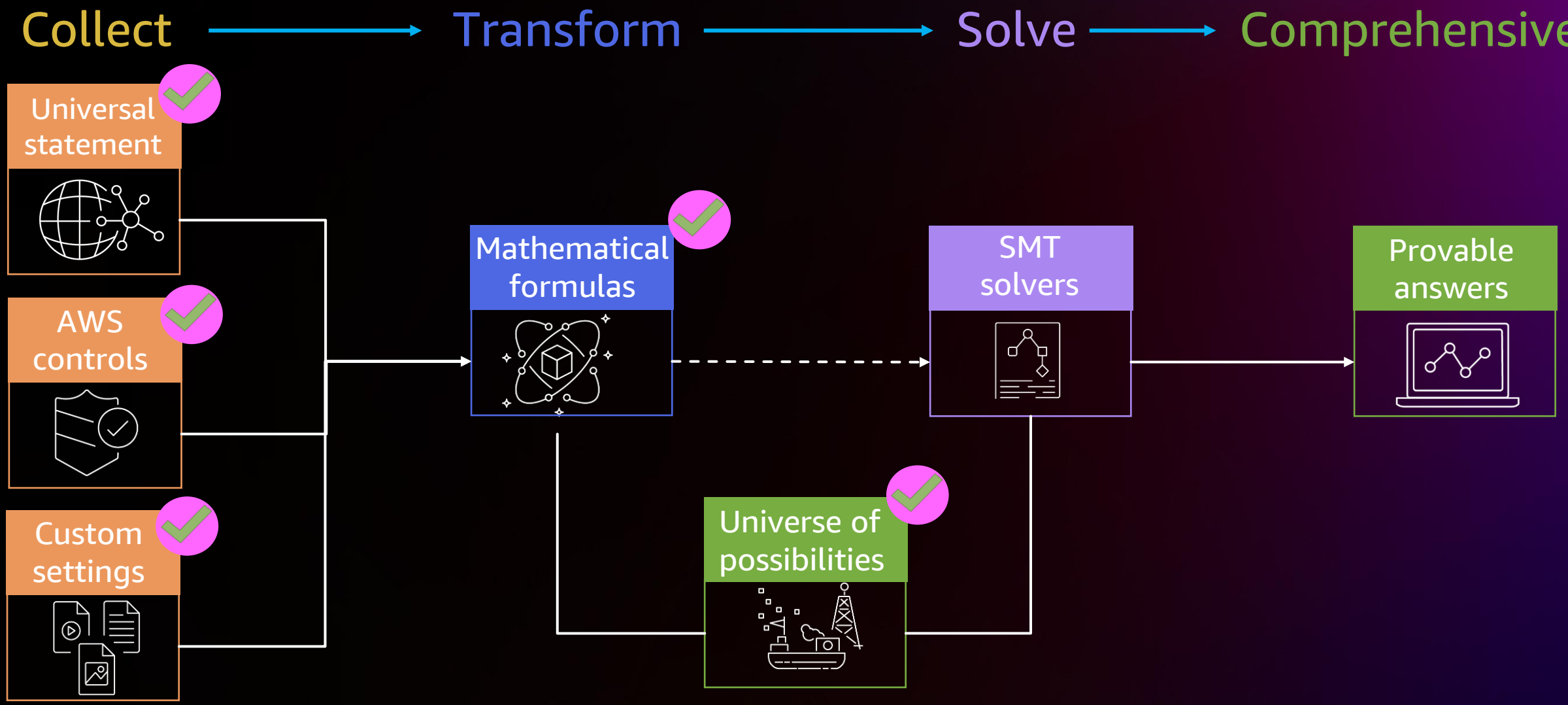
```
(declare-fun principal () String)
(declare-fun action () String)
(declare-fun resource () String)
(declare-fun awsSourceVpc () String)
(declare-fun s1-match () Bool)
(assert
 (= s1-match
    (and
     (= principal "111122223333")
     (= action "s3:GetObject")
     (str.in_re resource
       (re.++
        (str.to_re "arn:aws:s3:::magic-potions/")
        re.all)))))

(declare-fun s2-match () Bool)
(assert
 (= s2-match
    (not (or (= awsSourceVpc "vpc-broom")
             (= awsSourceVpc "vpc-wand")))))

(declare-fun allowed () Bool)
(assert
 (= allowed
    (and s1-match
         (not s2-match))))
```

000000000000 calling s3:AbortMultiPartUpload from vpc-0000000?
000000000000 calling s3:AbortMultiPartUpload from vpc-0000001?
…
111122223333 calling s3:GetObject from vpc-broom?
…
111122223333 calling s3:GetObject from vpc-wand?
…
999999999999 calling s3:UpdateJobStatus from vpc-ffffffe?
999999999999 calling s3:UpdateJobStatus from vpc-fffffff?

Ready to solve



SMT solvers

Theorem prover for satisfiability modulo theories

Solves against constraints

Combines proven algorithms

Optimized for real-world scenarios

Examples:

- Z3
- CVC4
- CVC5

Solve for a path

Configurations

```
(assert
 (= s1-match
    (and
     (= principal"111122223333")
...
```

Universal statement

```
(declare-fun my-property () Bool)
(assert
 (= my-property
    (=> allowed
        (or (= awsSourceVpc "vpc-broom")
            (= awsSourceVpc "vpc-wand")))))
```

Solver potion

```
(assert
  (not my-property))
```

Provable answer

```
False
“No path found”
```

Demo: Solve

Demo!

Question

My bucket **`magic-potions`** is accessible only inside **`vpc-broom`** or **`vpc-wand`**

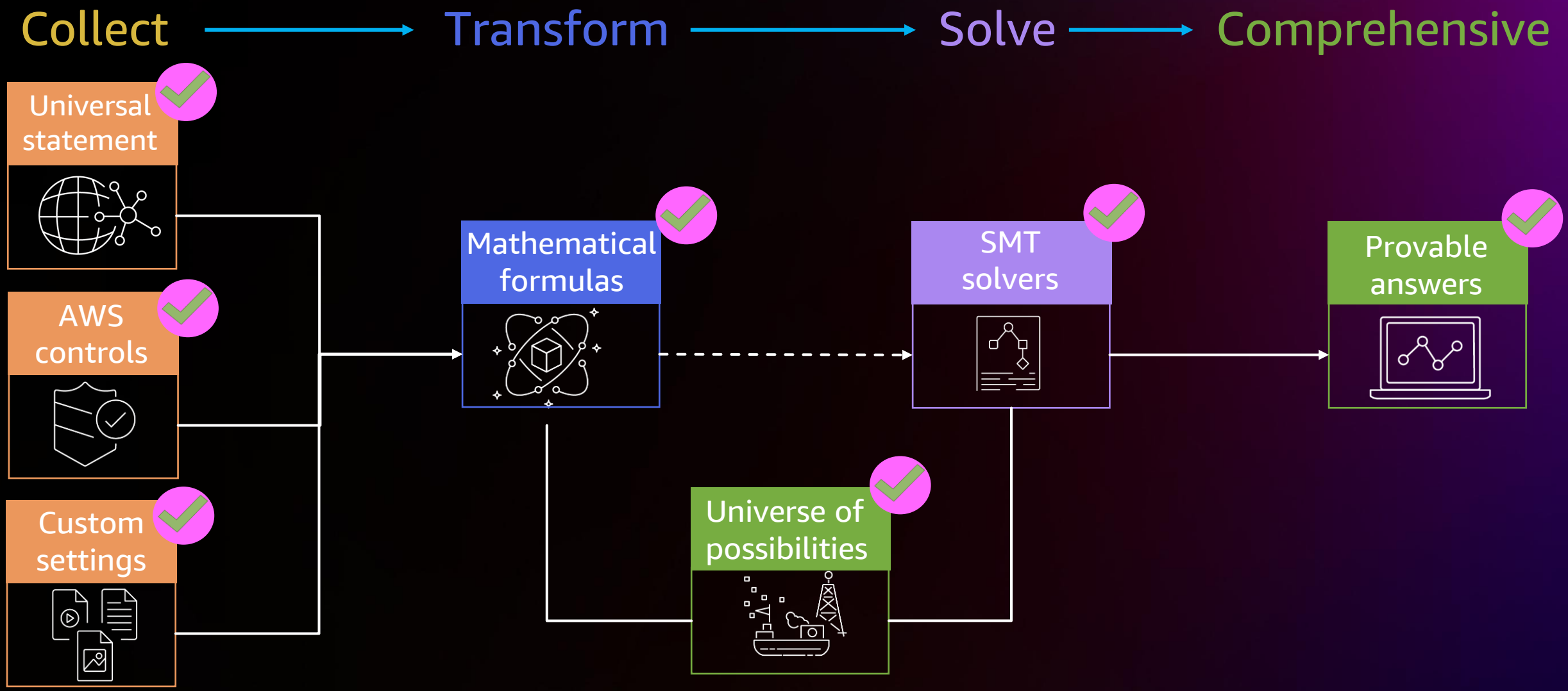
Path

Is there an access path that is not in **`vpc-broom`** or **`vpc-wand`**?

Demo steps

1. Input the universal statement (in math)
2. Input the policy (in math)
3. Use an SMT solver

Automated reasoning



Brewing comprehensive potions

Advanced automated reasoning

More complex questions

Solving with speed

Solving at scale

More complex questions

"My KMS keys are only accessed from inside my AWS organization"

"My bucket is not shared beyond my account"

Multiple universal statements for one question

"Does Pickles have cross-account access to **`apple-bucket`**?"

Universal statement

The resource policy grants public access

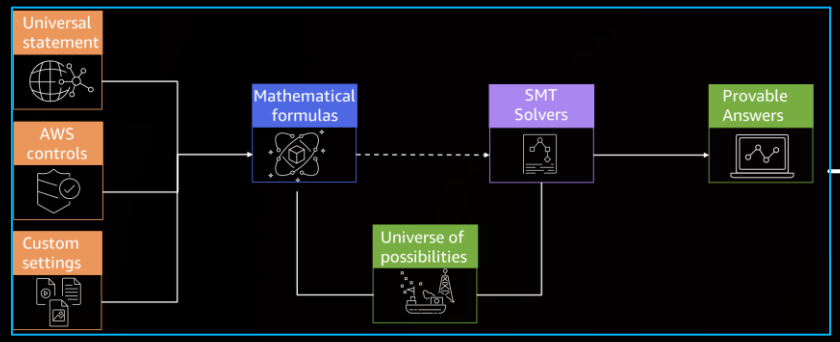
The access point policy grants public access

Pickles has admin access to **`apple-bucket`**

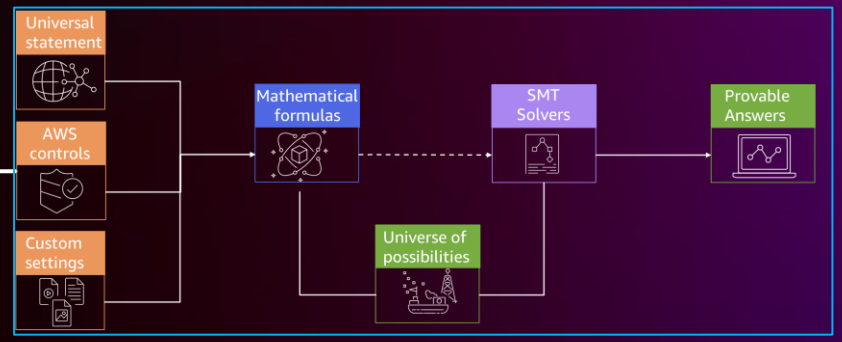
Pickles has eat access to **`apple-bucket`**

Combining provable answers

Resource policy grants public access

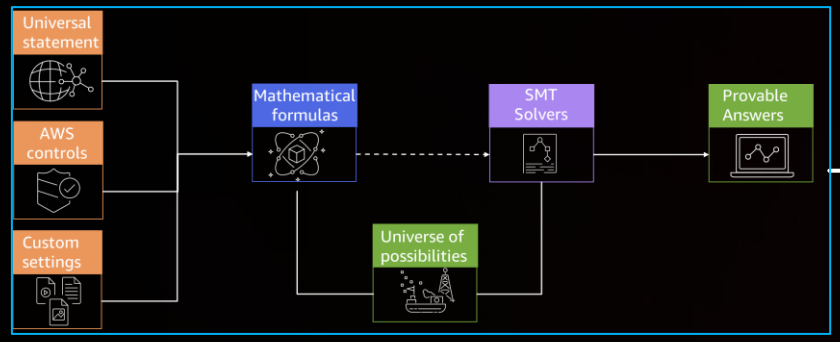


Pickles has admin access to `apple-bucket`

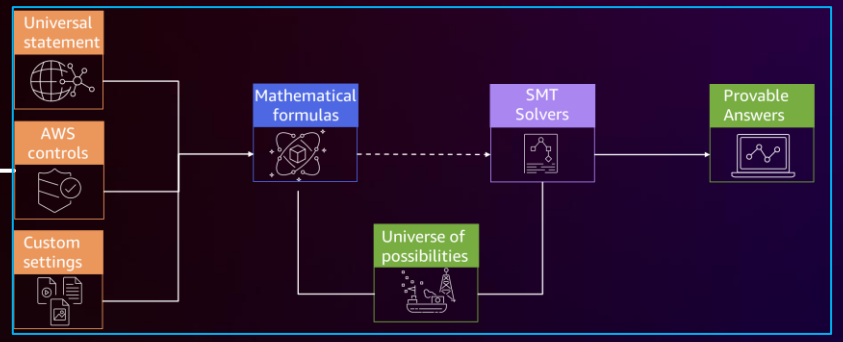


Provable answers

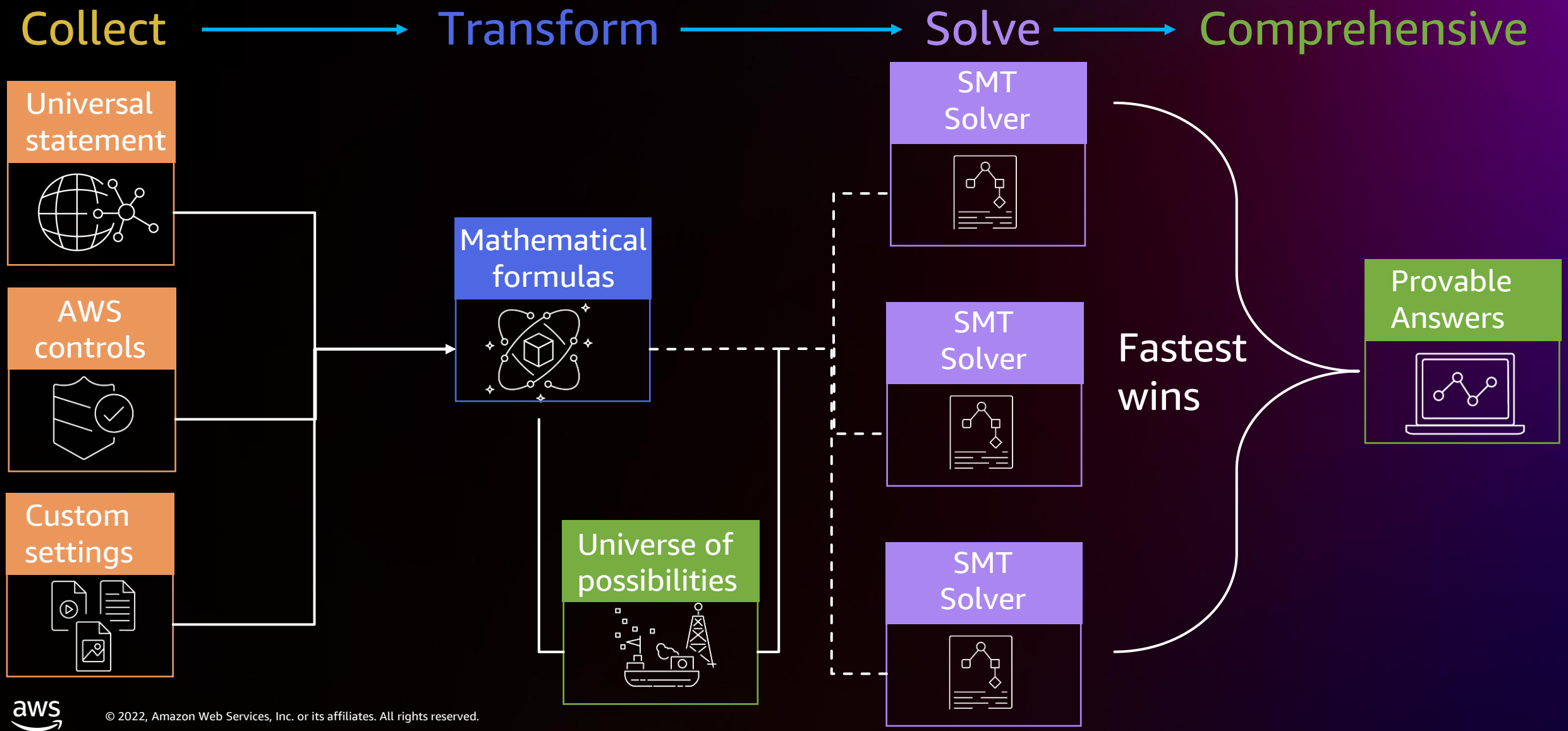
Access point policy grants public access



Pickles has eat access to `apple-bucket`



Solving with speed

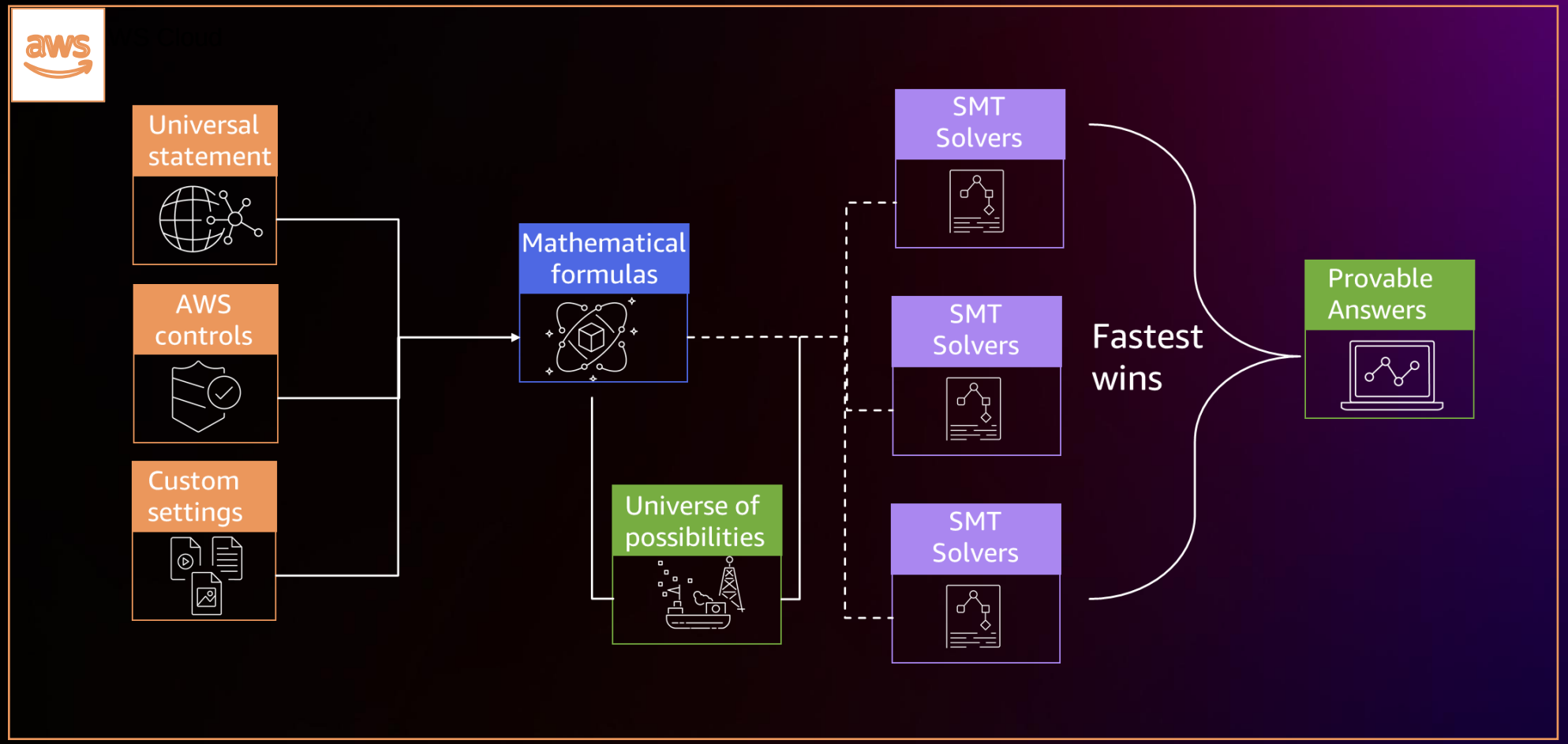


Solving at scale

AWS runs over a billion SMT queries a day and growing

Compute

Blog link



Security charms in AWS

Managed automated reasoning

IAM Access Analyzer

VPC Network Access Analyzer

Block Pubic Access

“Show me public access”

Use IAM Access Analyzer to review public findings for 14 resource types

Enable IAM Access Analyzer in your account or organization

Continuously **monitors and reviews** access controls across 14 resource types

Uses **automated reasoning** to determine public or cross-account access

Generates findings for you to review and determine if they match your intent

Verify public and cross-account access by previewing access as you modify resource policies

Review public access findings

Demo!

Question

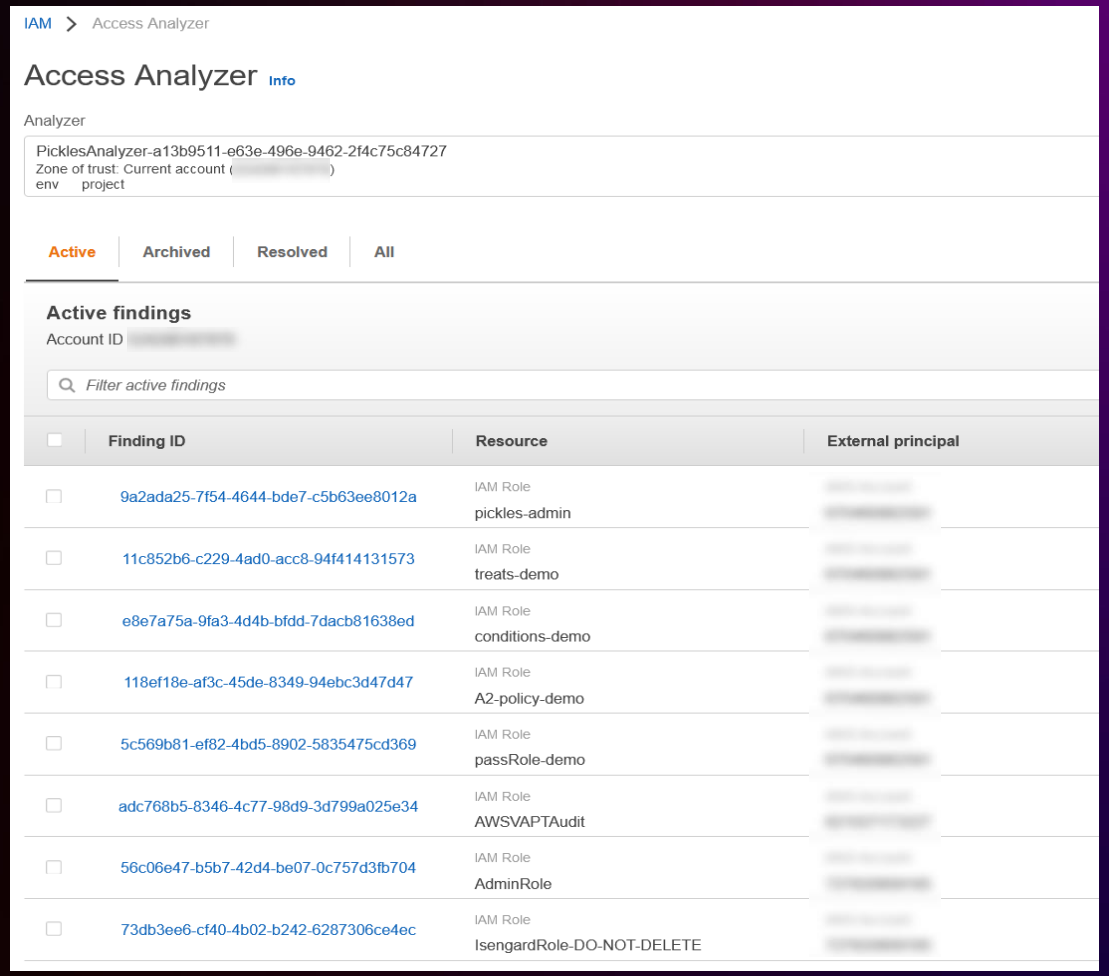
Which of my buckets are public?

Path

For each bucket, is there a path to public?

Demo steps

1. Prep: Enable IAM Access Analyzer
2. Review public findings
3. Investigate if public is intended



“Show me the future”

Demo!

Use IAM Access Analyzer to see how policy updates change the future

Verify public and cross-account access by previewing access as you modify resource policies

Question

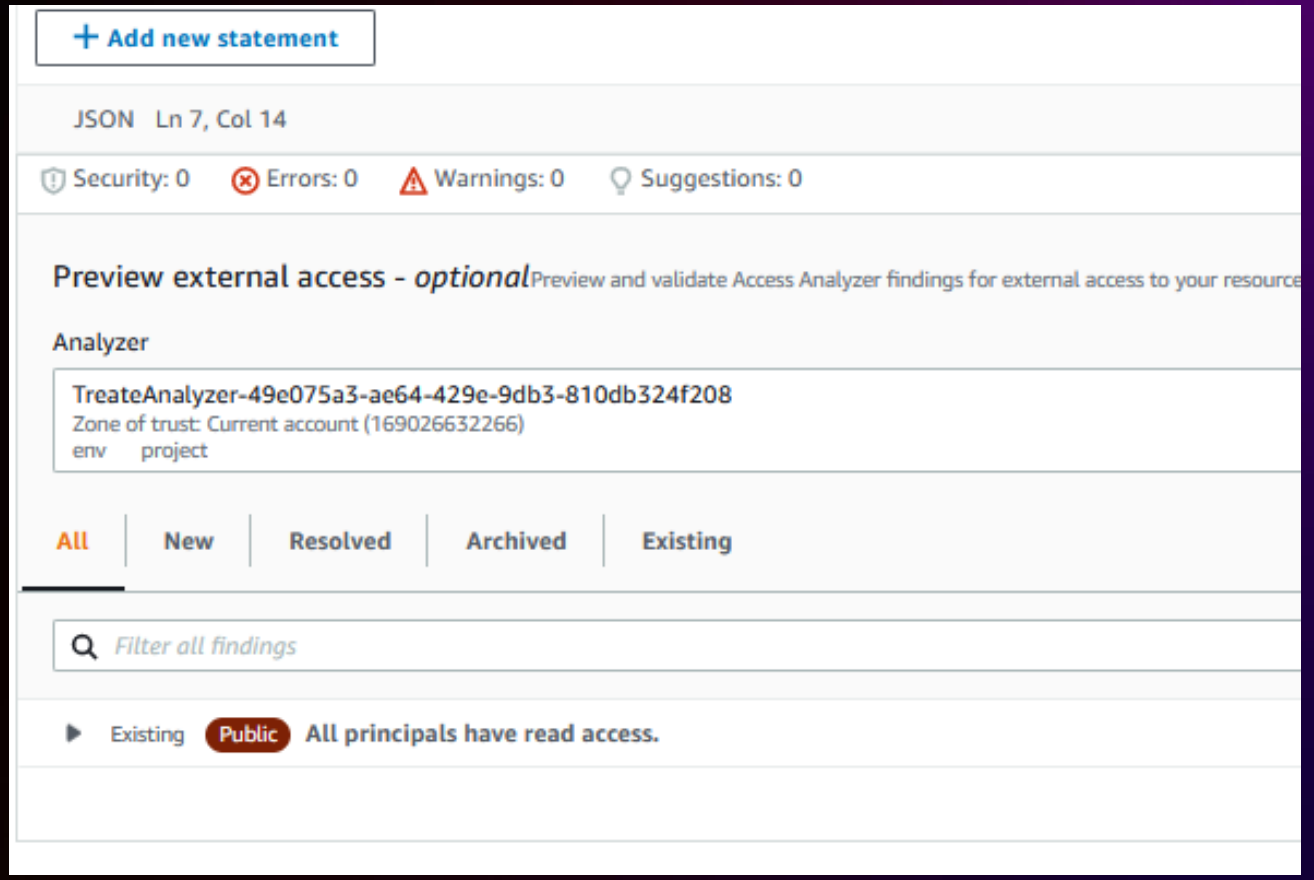
Does this policy update resolve or grant public access?

Path

Is there a path to public for this policy?

Demo steps

1. Update bucket policy to remediate public access
2. Review resolved and new findings



“Block public access”

Enable Block Public Access for S3 buckets

Enable BPA for Amazon S3 in your account on your bucket

Restricts access to an access point or bucket policy that grants public access

Rejects calls to attach access point or bucket policy if the specified policy allows public access

Block Public Access demo

Demo!

Universal statement

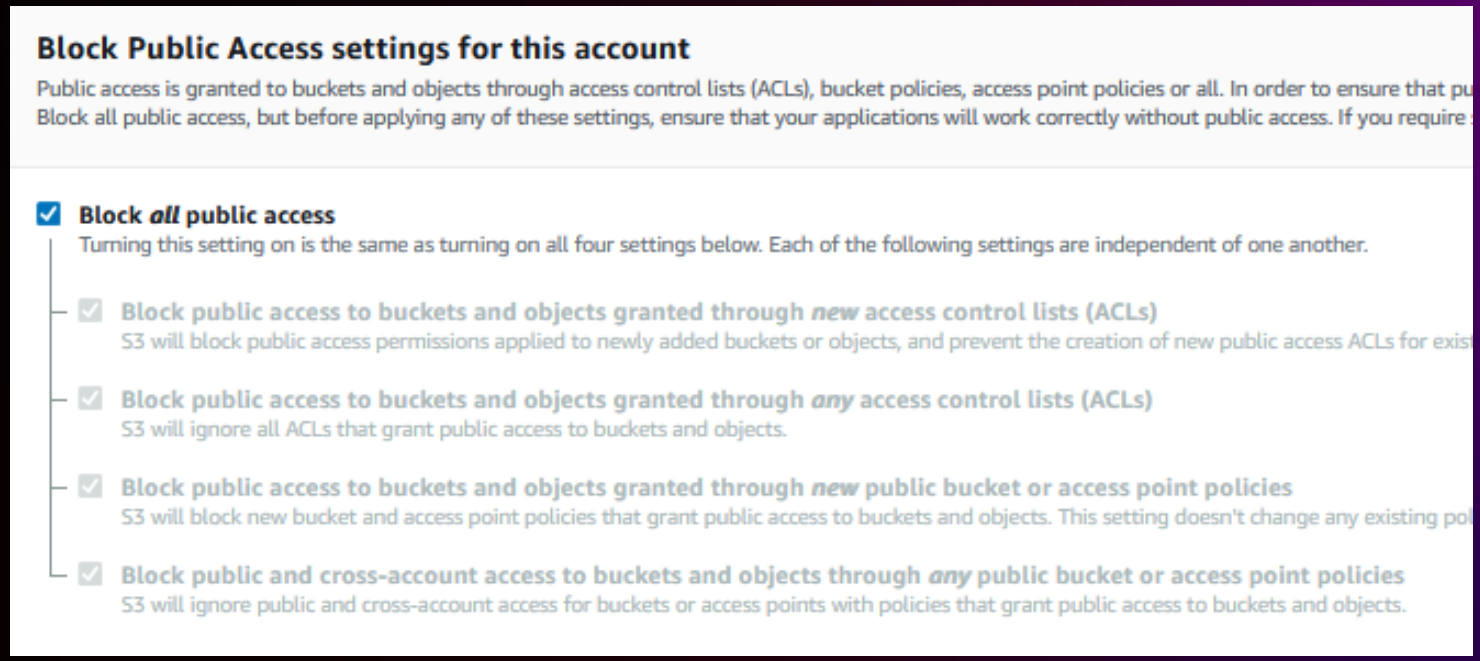
Block public access to all my buckets

Path

Is there a path to public?

Demo steps

1. Prep: Enable BPA for an account
2. Edit a bucket policy to grant public access
3. Watch the rejection



“Tell me who can talk to the internet”

Use VPC Network Access Analyzer to review network configurations

VPC Network Access Analyzer helps you **identify unintended network access** relative to your security and compliance requirements

Helps you **verify** network segmentation, internet accessibility, trusted network paths, and trusted network access

Start with four **purpose-built network scopes** or create your own

Uses **automated reasoning** to help you understand network access

Use VPC Network Access Analyzer in the **console or API**

Internet accessibility demo

Demo!

Question

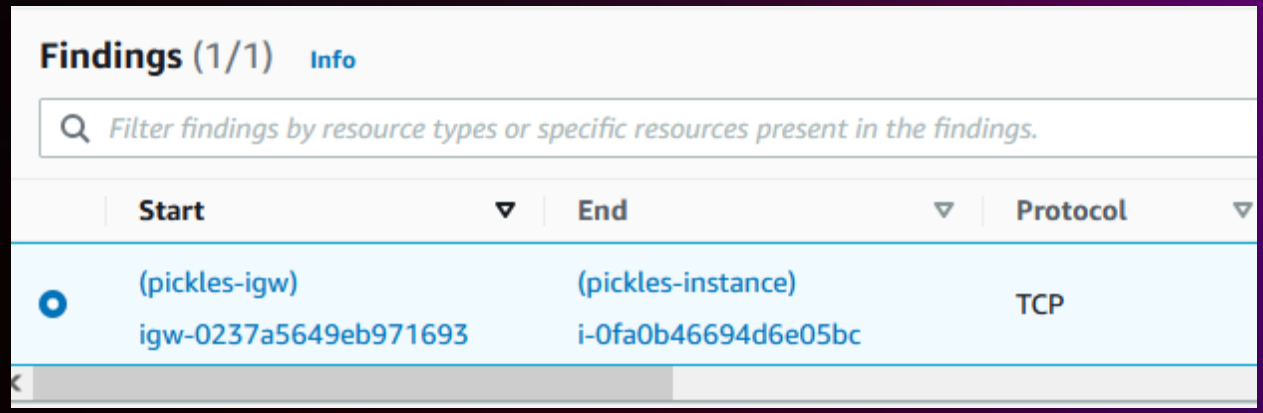
Which resources can be accessed from internet gateways?

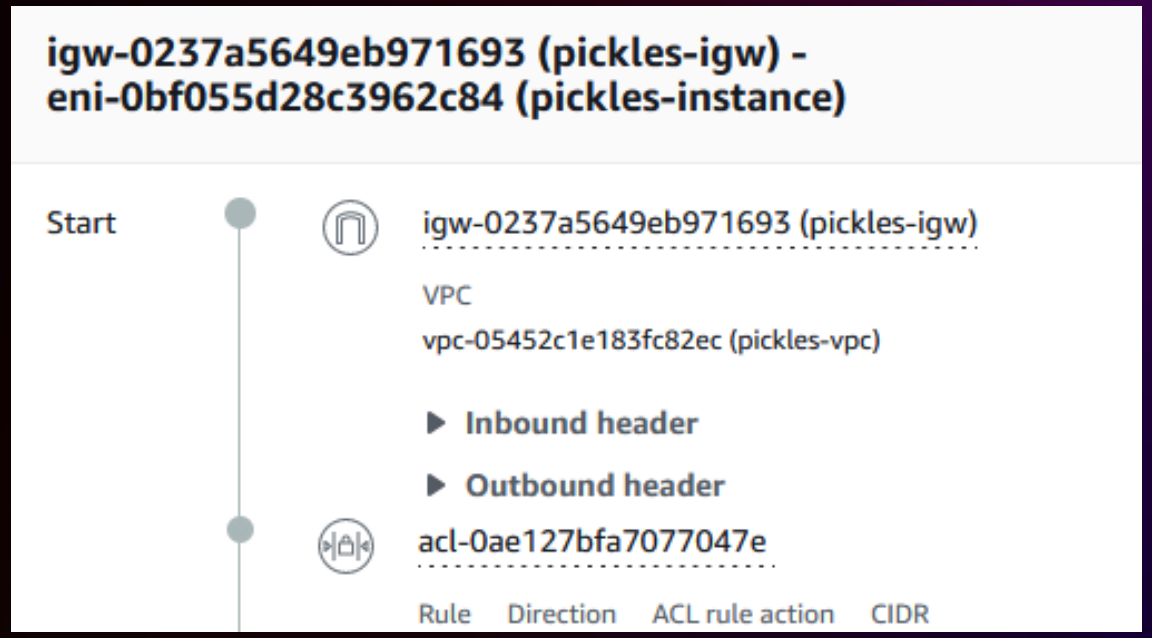
Path

For each network control, prove if there is a path to the internet

Demo steps

1. Navigate to VPC Network Access Analyzer
2. Review internet accessible findings
3. Verify they match your intent





AWS security charms

Show me public access

Show me cross-account access

Block Public Access

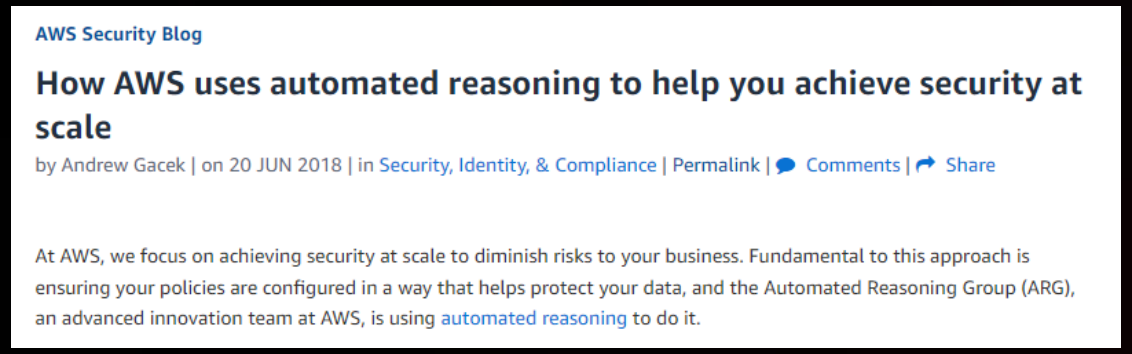
Tell me who can talk to the internet

Tell me who the internet can talk to

What will come next?

Our favorite security alchemy resources

How AWS uses automated reasoning to help you achieve security at scale


AWS Security Blog

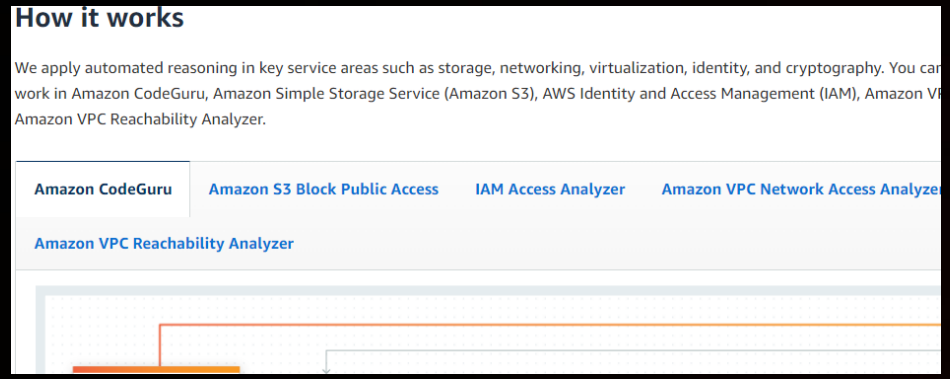
How AWS uses automated reasoning to help you achieve security at scale

by Andrew Gacek | on 20 JUN 2018 | in Security, Identity, & Compliance | Permalink | Comments | Share

At AWS, we focus on achieving security at scale to diminish risks to your business. Fundamental to this approach is ensuring your policies are configured in a way that helps protect your data, and the Automated Reasoning Group (ARG), an advanced innovation team at AWS, is using automated reasoning to do it.

Dive Deep into IAM Access Analyzer (SEC309)

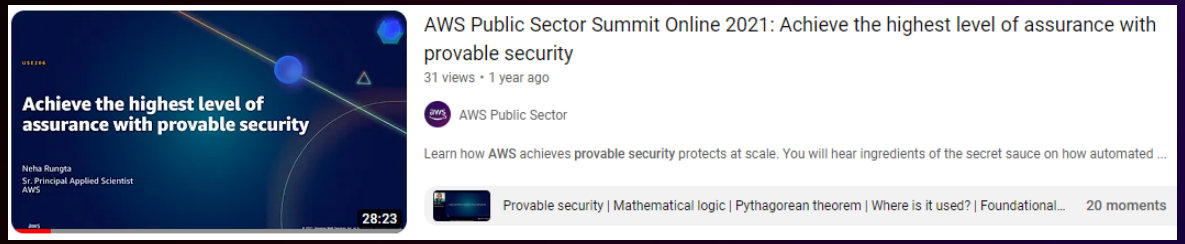
Provable Security


How it works

We apply automated reasoning in key service areas such as storage, networking, virtualization, identity, and cryptography. You can work in Amazon CodeGuru, Amazon Simple Storage Service (Amazon S3), AWS Identity and Access Management (IAM), Amazon VPC, Amazon VPC Reachability Analyzer.

Amazon CodeGuru | Amazon S3 Block Public Access | IAM Access Analyzer | Amazon VPC Network Access Analyzer | Amazon VPC Reachability Analyzer

Achieve the highest level of assurance with provable security



Thank you!

Brigid Johnson
@bjohnso5y

Neha Rungta
@neharungta

Please complete the session survey in the **mobile app**