AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

SEC310-R2

Automated reasoning: Protect data through provable security

John Backes (he/him)
Senior Applied Scientist
AWS – EC2 Networking

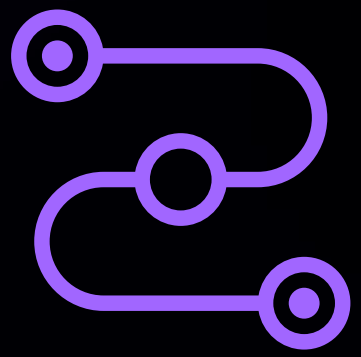
Dan Peebles (he/him)
Senior Software Development Engineer
AWS – Identity

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

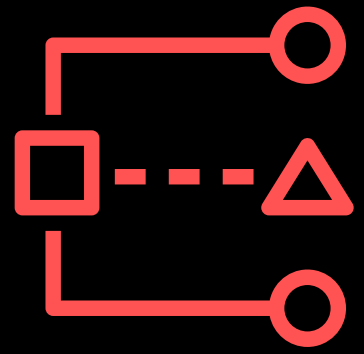
What am I going to talk about?



Amazon Inspector



Amazon VPC
Reachability Analyzer

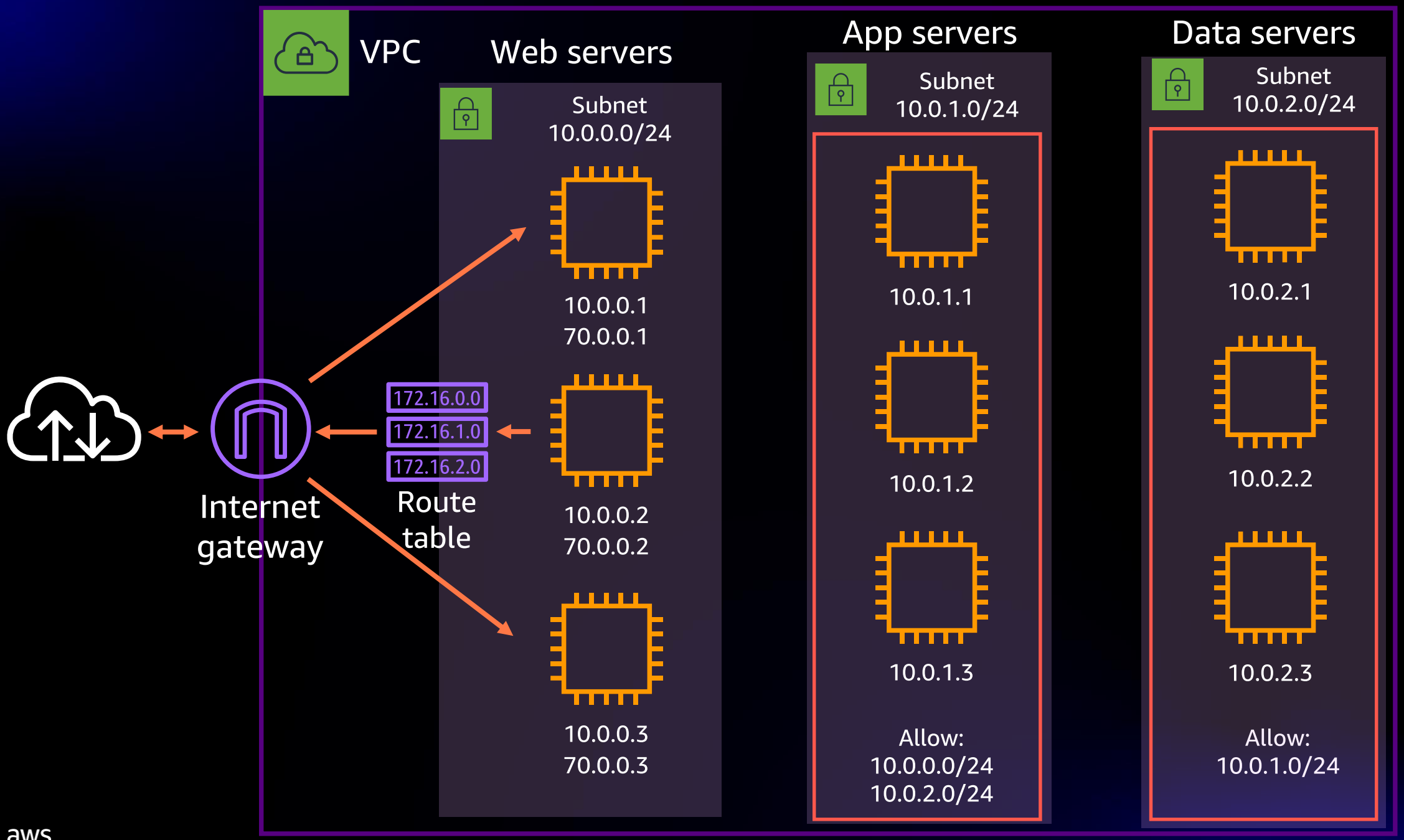


AWS IAM Access
Analyzer

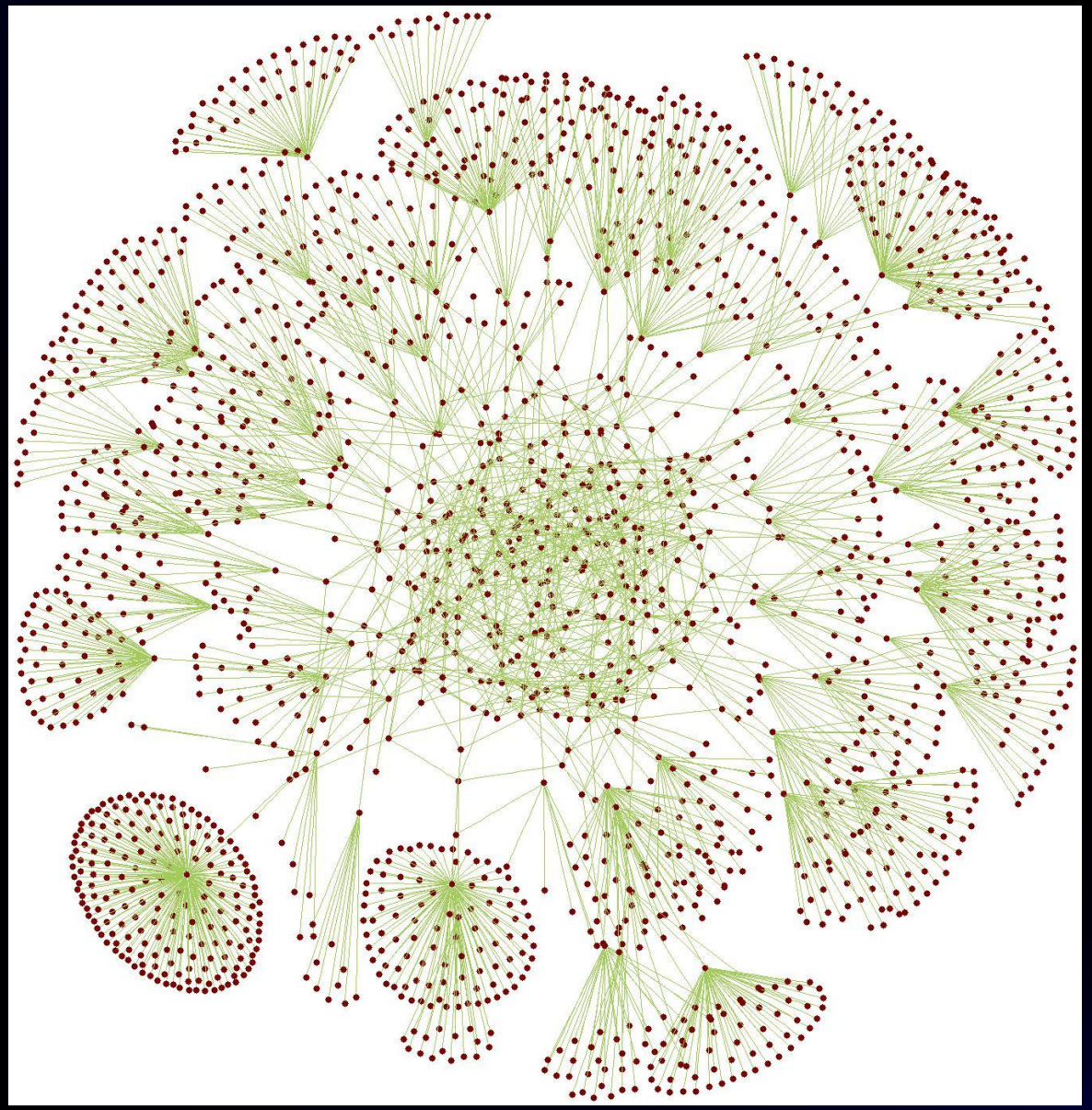
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Who has access to what?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

IAM policies

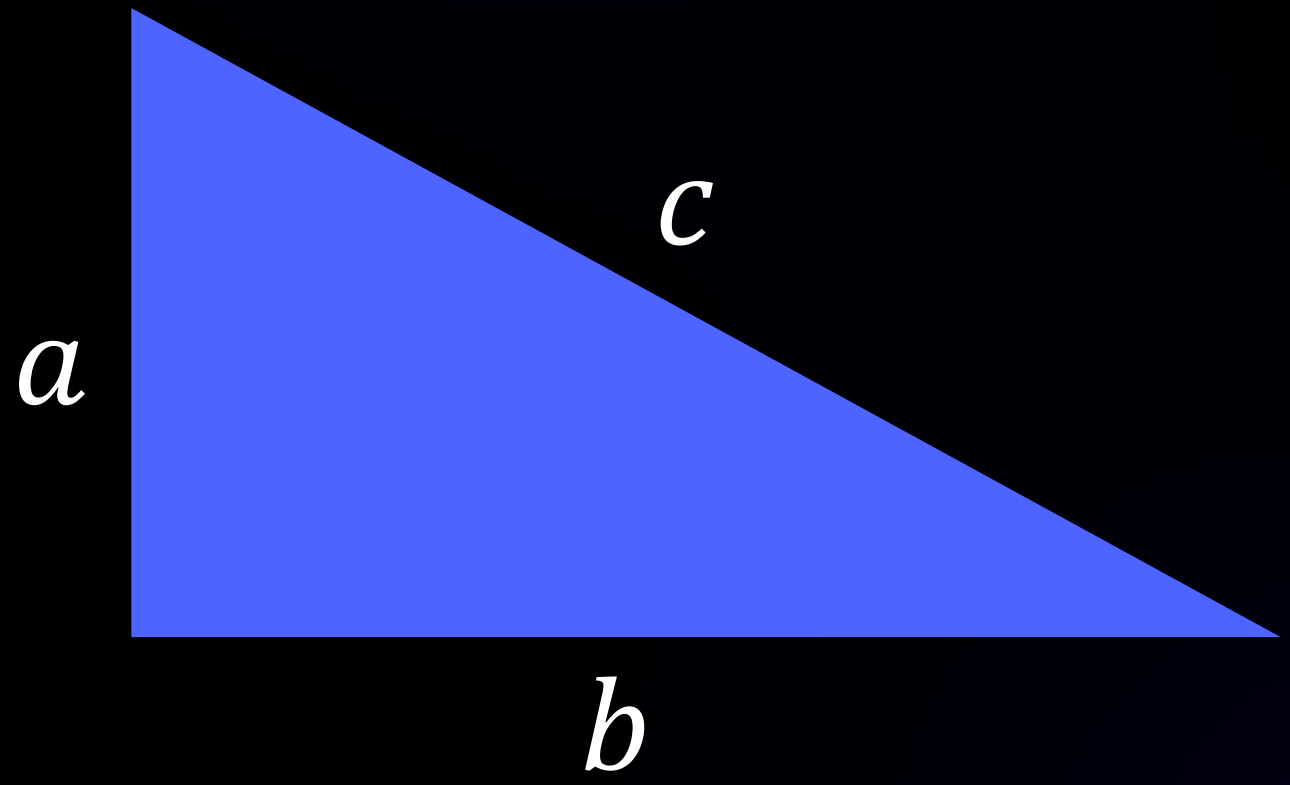
```
{
  "Version" : "2012-10-17"
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "AWS": "111122223333" },
      "Action": [ "s3:GetObject" ],
      "Resource": "arn:aws:s3:::BUCKET/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::BUCKET/secret/*",
      "Condition": { "NotIpAddress": { "aws:SourceIp": "54.240.143.0/24" } }
    }
  ]
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How do we check everything?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How do we check everything?



$$a^2 + b^2 \stackrel{?}{=} c^2$$

$$3^2 + 4^2 = 5^2$$

$$5^2 + 12^2 = 13^2$$

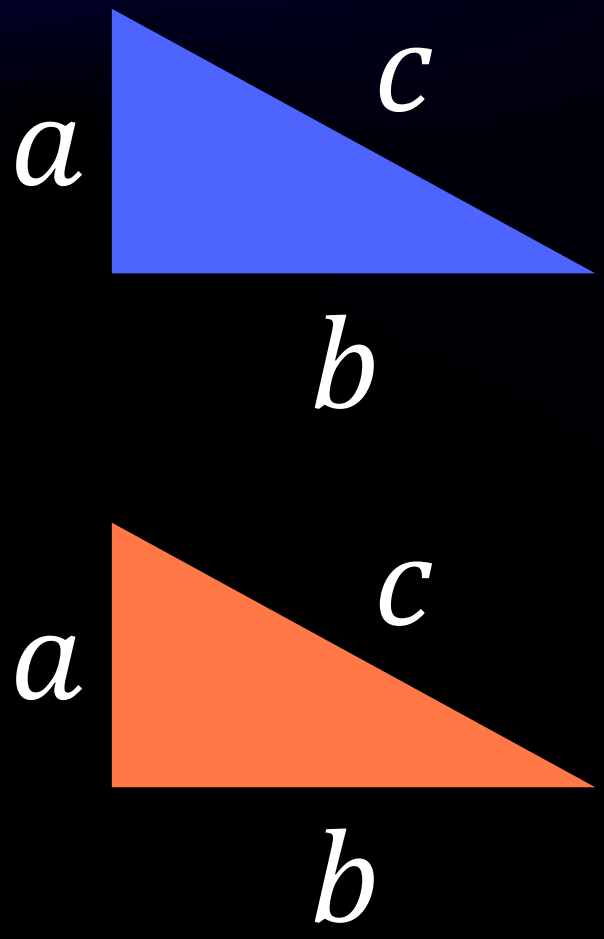
$$8^2 + 15^2 = 17^2$$

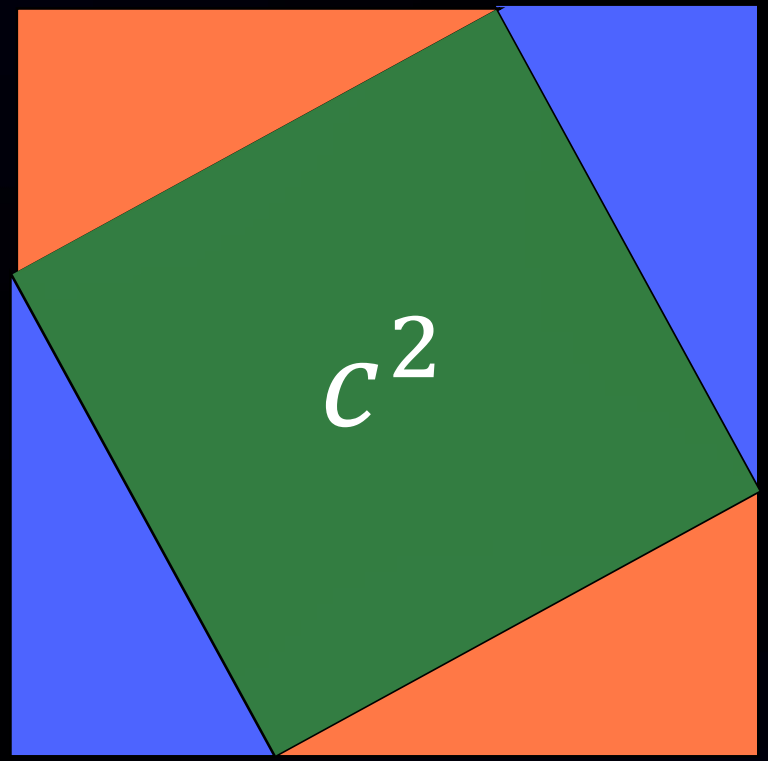
$$\ldots$$

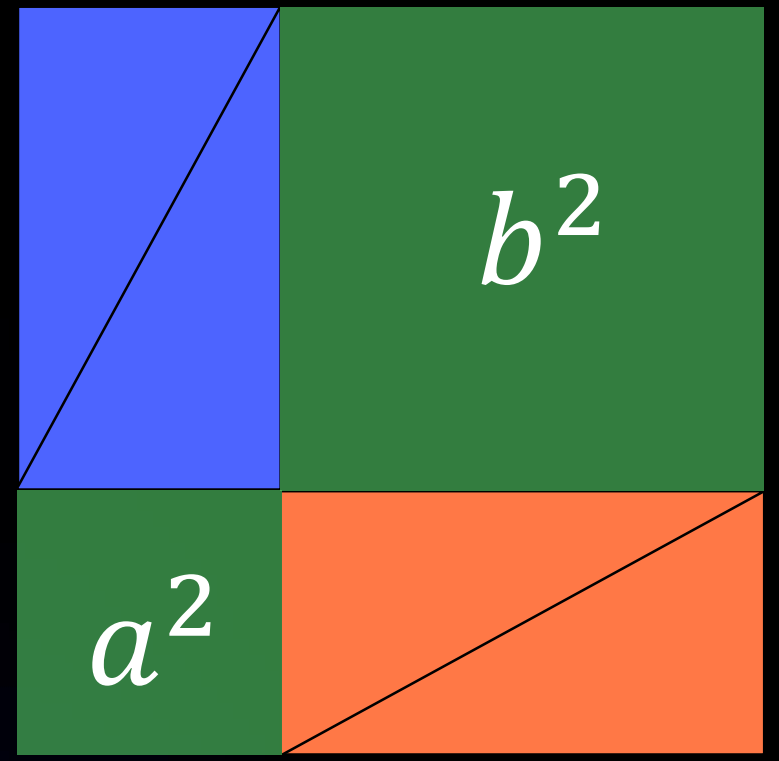
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How do we check everything?







$$a^2 + b^2 = c^2$$

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

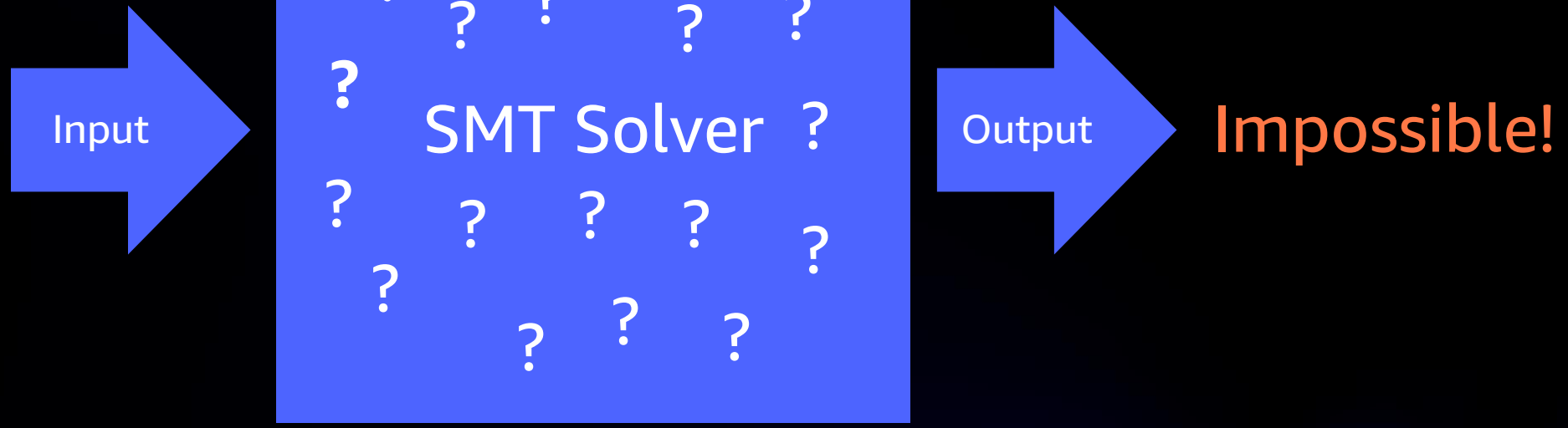
How to automate proof: SMT Solvers!



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How to automate proof: SMT Solvers!

$0 < x < 4$
and
$2y < x$
and
$z < y$
and
$z > 0$



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

IAM policies → Logic

```
{
  "Version" : "2012-10-17"
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "AWS": "111122223333" },
      "Action": [ "s3:GetObject" ],
      "Resource": "arn:aws:s3:::BUCKET/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::BUCKET/secret/*",
      "Condition": { "NotIpAddress": { "aws:SourceIp": "54.240.143.0/24" } }
    }
  ]
}
```

```
(
  principal == "111122223333" &&
  action == "s3:GetObject" &&
  resource ~= "arn:aws:s3:::BUCKET/*”
) &&
!(
  action ~= "s3:*" &&
  resource ~= "arn:aws:s3:::BUCKET/secret/*" &&
  !cidr_match(aws:SourceIp,"54.240.143.0/24")
)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SMT – Satisfiability Modulo Theories

Some SMT Solvers: CVC5, MonoSAT, Yices, Z3

Sample program:

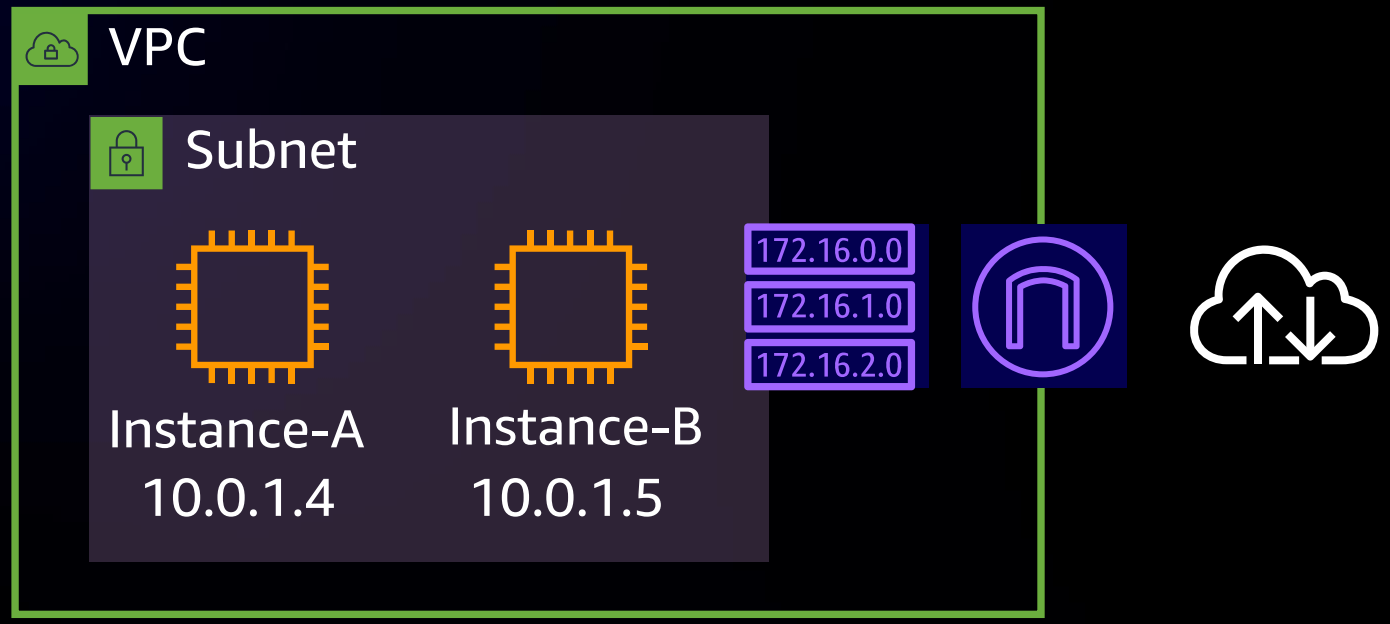
```
(assert (= principal "11122223333"))
(assert (= action "s3:PutObject"))

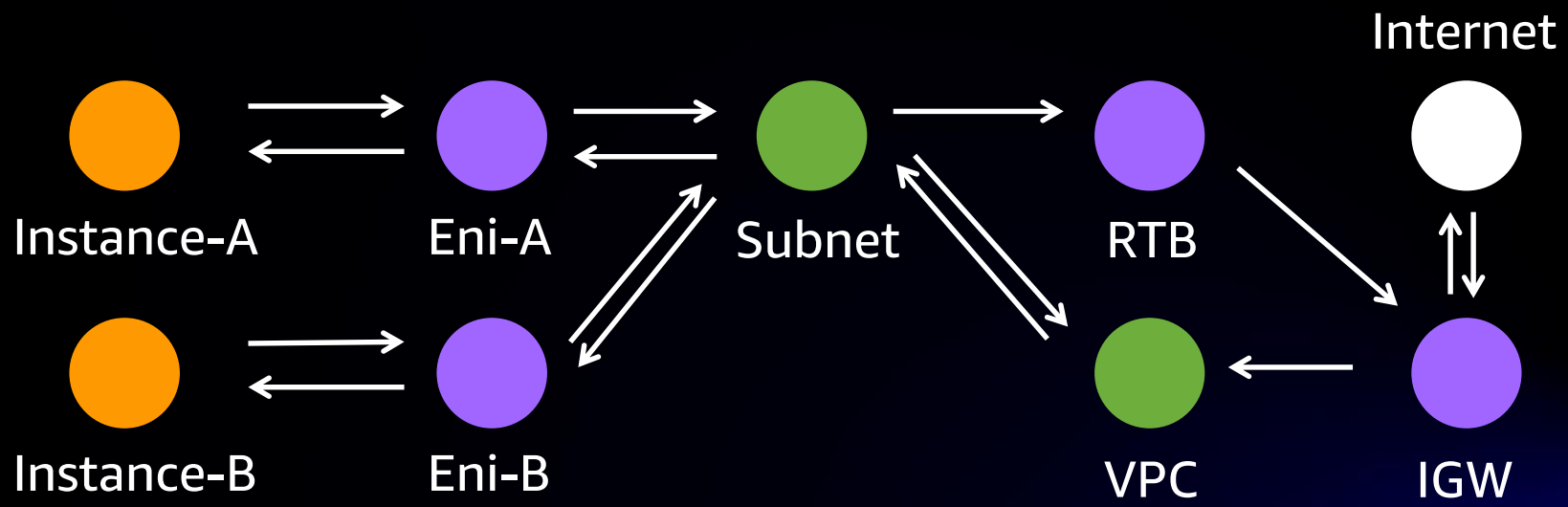
; Action must not match s3:Get*
(assert
  (not
    (str.in.re action
      (re.++ (str.to.re "s3:Get") (re.* re.allchar)))

(check-sat)
(get-model)
```

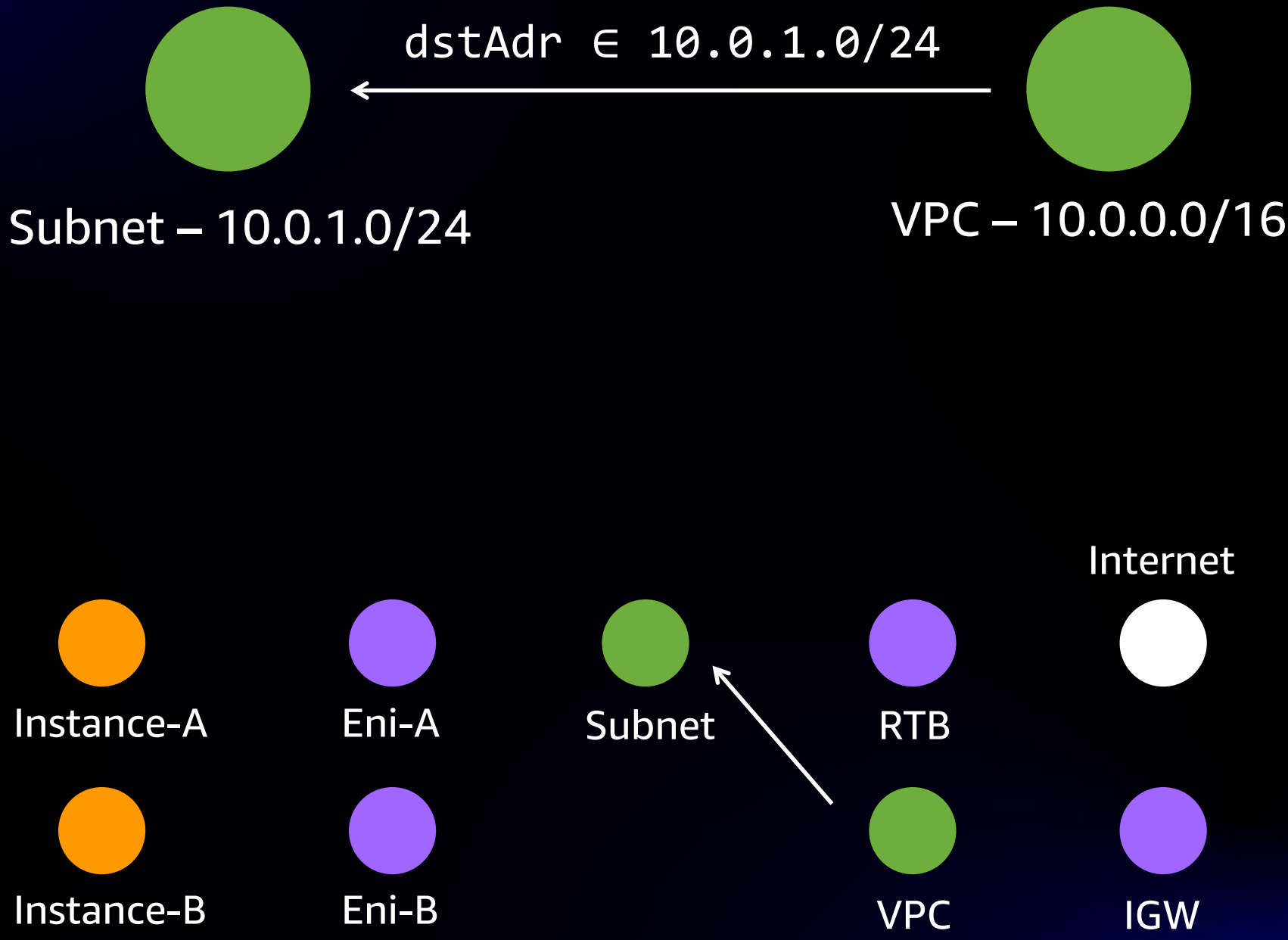
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

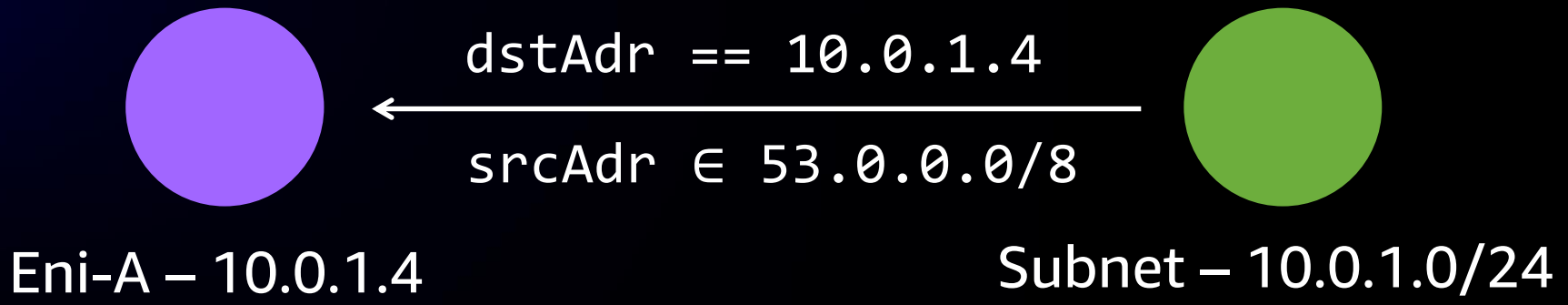


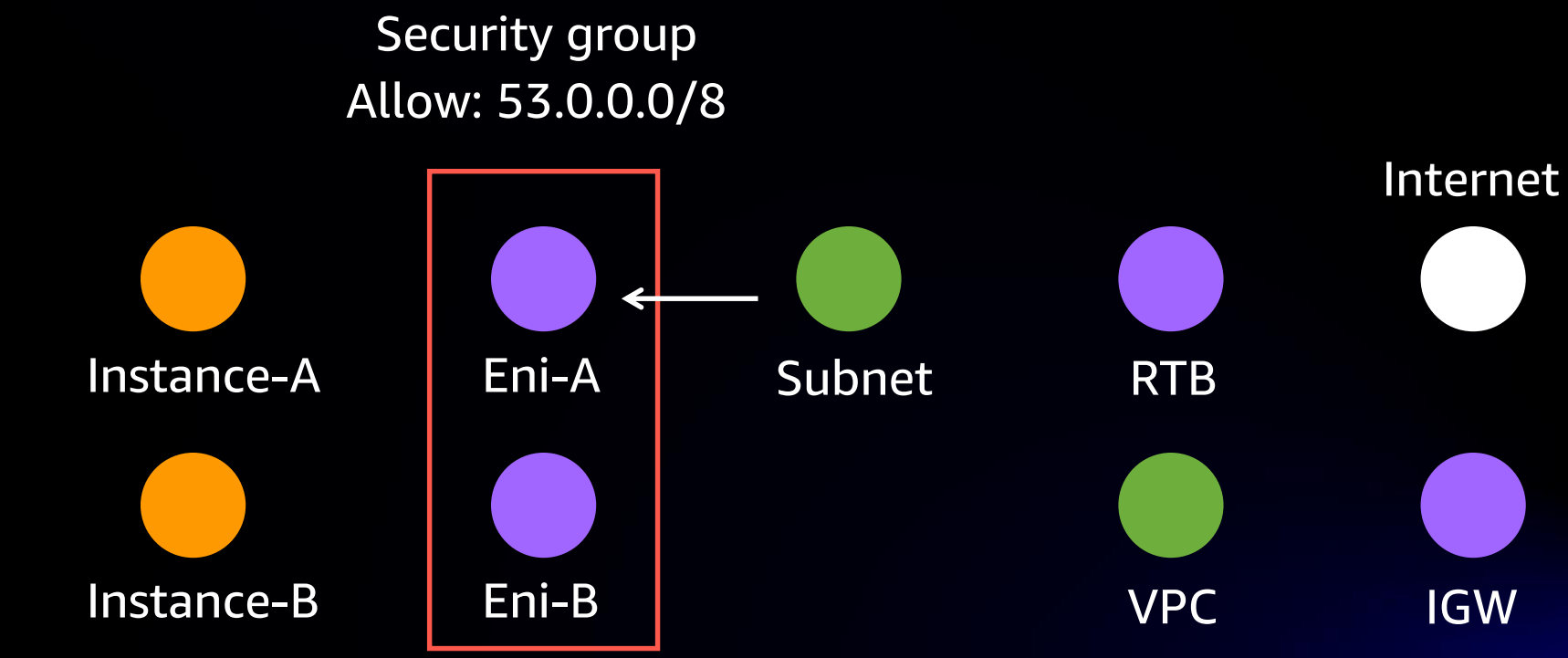


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.





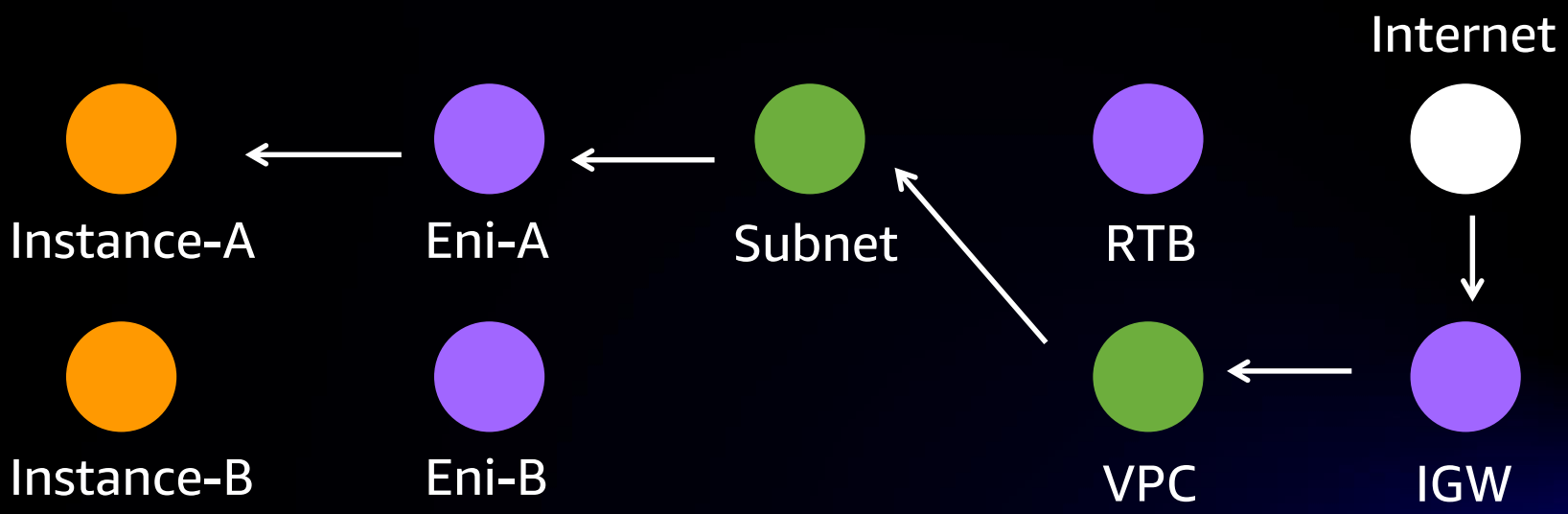
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Can I ssh into Instance-A from the internet?

```
dstPort == 22 and
protocol == TCP and
reaches(Internet, Instance-A)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

```
protocol = TCP
srcAdr = 53.0.0.2,
dstAdr = 10.0.1.4,
dstPrt = 22
```

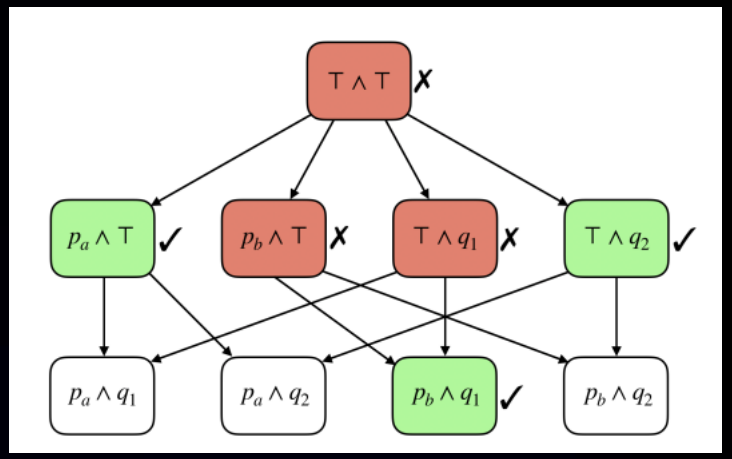


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Enumerating policy access

Intuition: game of 20 questions

```
- Effect: Allow
  Condition:
    StringEquals:
      SrcVpc:
      - vpc-a
      - vpc-b
- Effect: Allow
  Condition:
    StringEquals:
      OrgID: o-2
- Effect: Deny
  Condition:
    StringEquals:
      SrcVpc: vpc-b
    StringNotEquals:
      OrgID: o-1
```



Decide which questions to ask Zelkova

Ask efficiently and track answers

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Where can I learn more about how this works?

Stratified Abstraction of Access Control Policies

John Backes, Ulises Berrueco, Tyler Bray, Daniel Brim, Byron Cook, Andrew Gacek(✉), Ranjit Jhala, Kasper Luckow, Sean McLaughlin, Madhav Menon, Daniel Peebles, Ujjwal Pugalia, Neha Rungta, Cole Schlesinger, Adam Schodde, Anvesh Tanuku, Carsten Varming, and Deepa Viswanathan

Amazon Web Services, Seattle, USA
gacek@amazon.com

Abstract. The shift to cloud-based APIs has made application security critically depend on understanding and reasoning about *policies* that regulate access to cloud resources. We present *stratified predicate abstraction*, a new approach that summarizes complex security policies into a compact set of positive and declarative statements that precisely state *who* has access to a resource. We have implemented stratified abstraction and deployed it as the engine powering AWS's IAM Access Analyzer service, and hence, demonstrate how formal methods and SMT can be used for security policy *explanation*.

1 Introduction

A growing number of developers are using cloud-based implementations of basic resources like associative arrays, encryption, storage, queuing, and event-driven execution, to engineer client applications. For example, millions of Amazon Web Services (AWS) customers use cloud APIs like Amazon SQS for queues, Amazon

Reachability Analysis for AWS-based Networks

J. Backes[1], S. Bayless[14], B. Cook[12], C. Dodge[1], A. Gacek[1], A.J. Hu[4], T. Kahsai[1], B. Kocik[1], E. Kotelnikov[13], J. Kukovec[15], S. McLaughlin[1], J. Reed[6], N. Rungta[1], J. Sizemore[1], M. Stalzer[1], P. Srinivasan[1], P. Subotić[12], C. Varming[1], B. Whaley[1], Y. Wu[1]

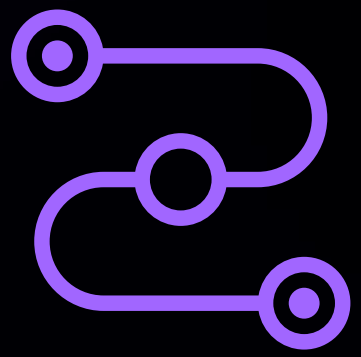
[1]Amazon Web Services, [2]University College London, [3]Chalmers University of Technology, [4]University British Columbia, [5]Vienna University of Technology, [6]Semmle Inc

Abstract. Cloud services provide the ability to provision virtual networked infrastructure on demand over the internet. The rapid growth of these virtually provisioned cloud networks has increased the demand for automated reasoning tools capable of identifying misconfigurations or security vulnerabilities. This type of automation gives customers the assurance they need to deploy sensitive workloads. It can also dramatically reduce the cost and time-to-market for regulated customers looking to establish compliance certification for cloud-based applications. In this industrial case-study, we describe a new network reachability reasoning tool, called TIROS, that uses off-the-shelf automated theorem proving tools to fill this need. TIROS is the foundation of a recently introduced network security analysis feature in the *Amazon Inspector* service now available to millions of customers building applications in the cloud. TIROS is also used within Amazon Web Services (AWS) to automate the checking of compliance certification and adherence to security invariants for many AWS services that build on existing AWS networking features.

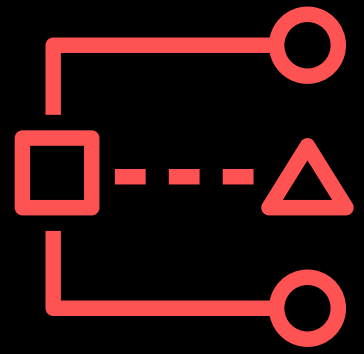
Where can I use this?



Amazon Inspector



Amazon VPC Reachability Analyzer



AWS IAM Access Analyzer

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.