Security Posture Analysis

Kostyantyn Vorobyov
Principal Researcher

ORACLE

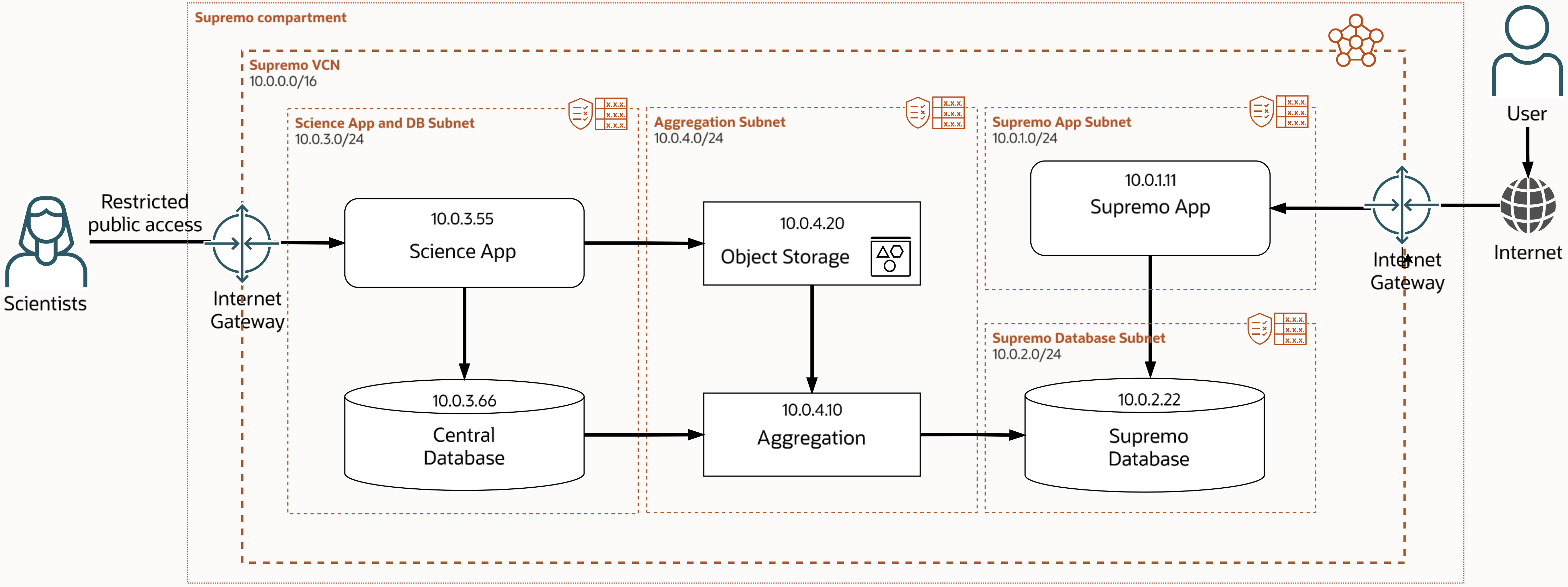
Cloud

Oracle Cloud Infrastructure (OCI)

ORACLE
Cloud

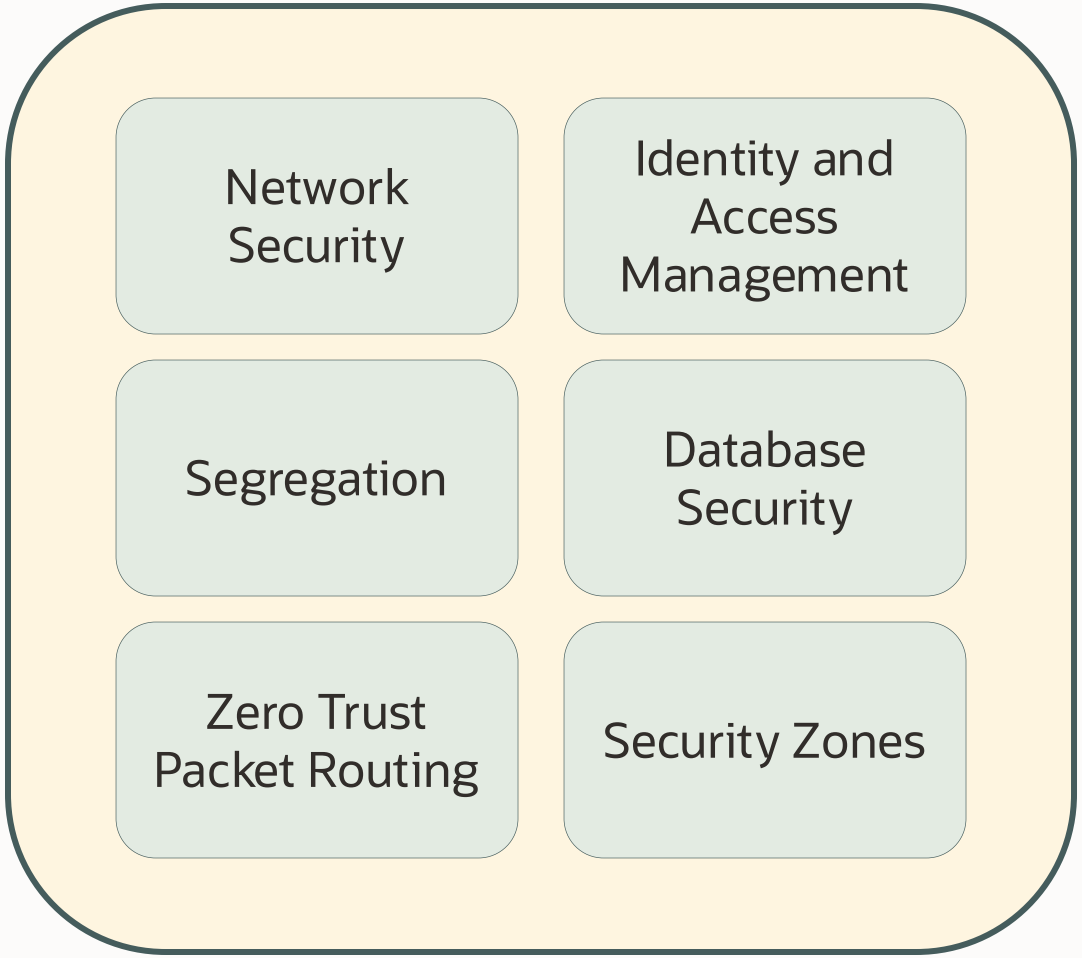
- Compute
- Analytics
- Networking and Connectivity
- Storage
- Autonomous database
- AI and Machine Learning
- Observability and Management
- … and more

Application Cloud Deployment



https://blogs.oracle.com/cloud-infrastructure/post/first-principles-zero-trust-packet-routing

Security Posture in OCI



- Security in the cloud is fragmented
- No universal security mechanism
- Security posture is the union of *all* policies
- An application's security posture rely on different mechanisms enforcing orthogonal policies

OCI Security Mechanisms

Network Security (NSG, SL)

```
ROUTE
    DESTINATION: "0.0.0.0/0"
    TARGET TYPE: "Internet Gateway"

NSG INGRESS
    SOURCE-TYPE: "CIDR"
    SOURCE: "160.39.115.23/24, ..."
    PROTOCOL: "TCP"
    SOURCE PORT: "ALL"
    DESTINATION PORT: "888"

NSG EGRESS
    SOURCE-TYPE: "CIDR"
    SOURCE: "10.0.3.66/32, ..."
    PROTOCOL: "TCP"
    SOURCE PORT: "ALL"
    DESTINATION PORT: "1521"
```

Identity and Access Management (IAM)

```
allow
    group Scientist
    to use autonomous-database-family
    in compartment Supremo

allow
    dynamic-group App
    to use objects
    in compartment Supremo
```

OCI Security Mechanisms

Zero Trust Packet Routing (ZPR)

```
VCN "Supremo"
    CIDR: "10.0.0.0/16"
    ZPR-security attribute: "Prod"

RESOURCE HOSTVNICs "ScienceApp"
    CIDR: "10.0.3.55"
    ZPR-security attribute: "Science"

RESOURCE HOSTVNICs "Central Database"
    CIDR: "10.0.3.66"
    ZPR-security attribute: "Sensitive"

In network:prod VCN
    allow app:science endpoints
    to connect to database:sensitive endpoints
```

Security Zones

```
deny public_bucket
```

Security Posture Analysis

Tooling for checking the overall security posture of an application does not exist

Goal

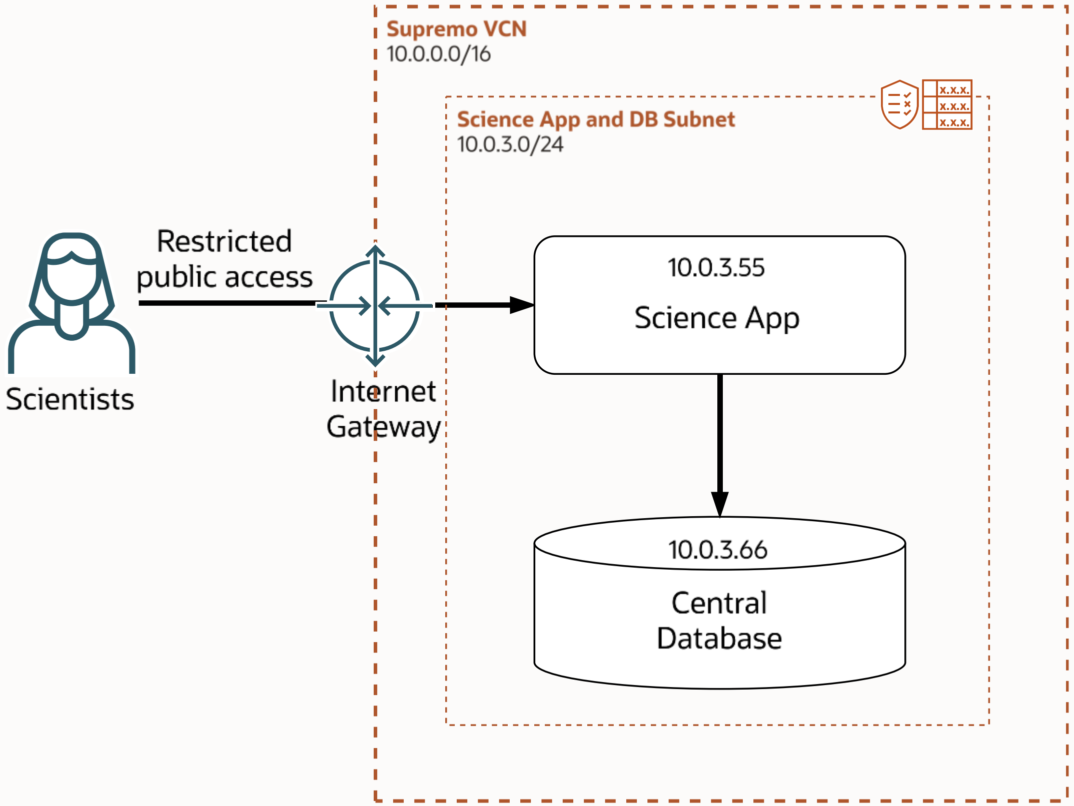
- Represent a system and all its security policies using a unified intermediate representation
- Statically check security policies
- Leave enforcement to existing mechanisms

Intermediate Representation for Cloud Security Posture Analysis

- Low-level
- Intent-oriented (*not* a policy language)
- System modeling
- Specification of policies (express *all* relevant policy languages)
- Built-in mechanism to check whether policies hold

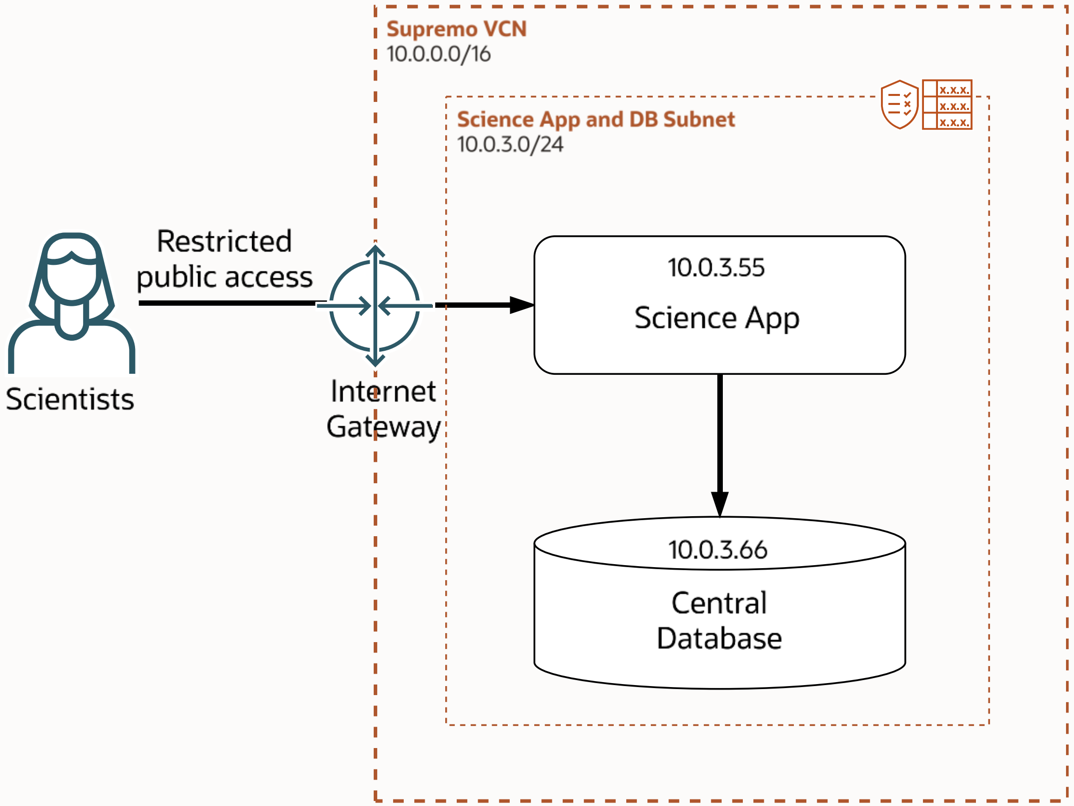
Intermediate Representation for Security Posture Analysis

Intermediate Representation (IR): System State



- System state is described using *entities* and *relations* (e.g., *nouns* and *verbs)*
- An *entity* (noun) is an actor, service, location, entry point, process or even a concept
- A *relation* (verb) is an action or an association denoting interaction between entities

Intermediate Representation (IR): Entities

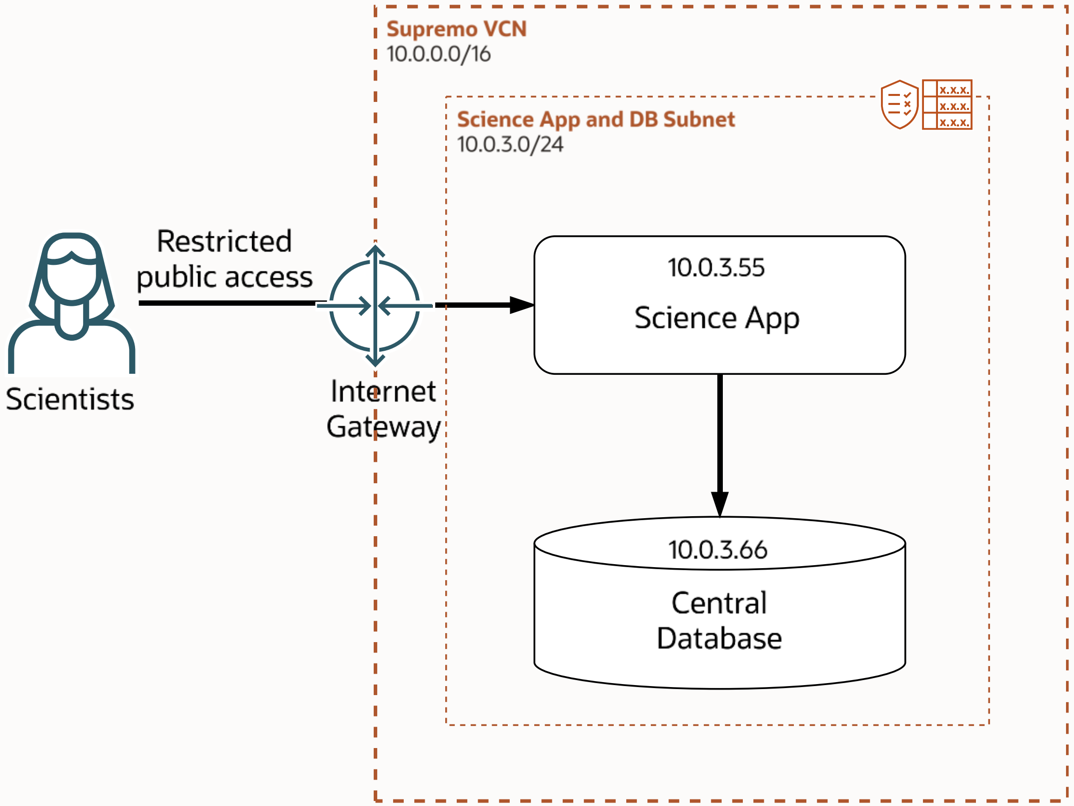



```
# Science App
entity App {
    type: [ application, datasource ]
    tag: [ science ]
}

# Central Database
entity CD {
    type: [ database, service ]
    tag: [ sensitive ]
}

# Scientists
entity Scientists {
     type: [user]
}
```

Intermediate Representation (IR): Relations

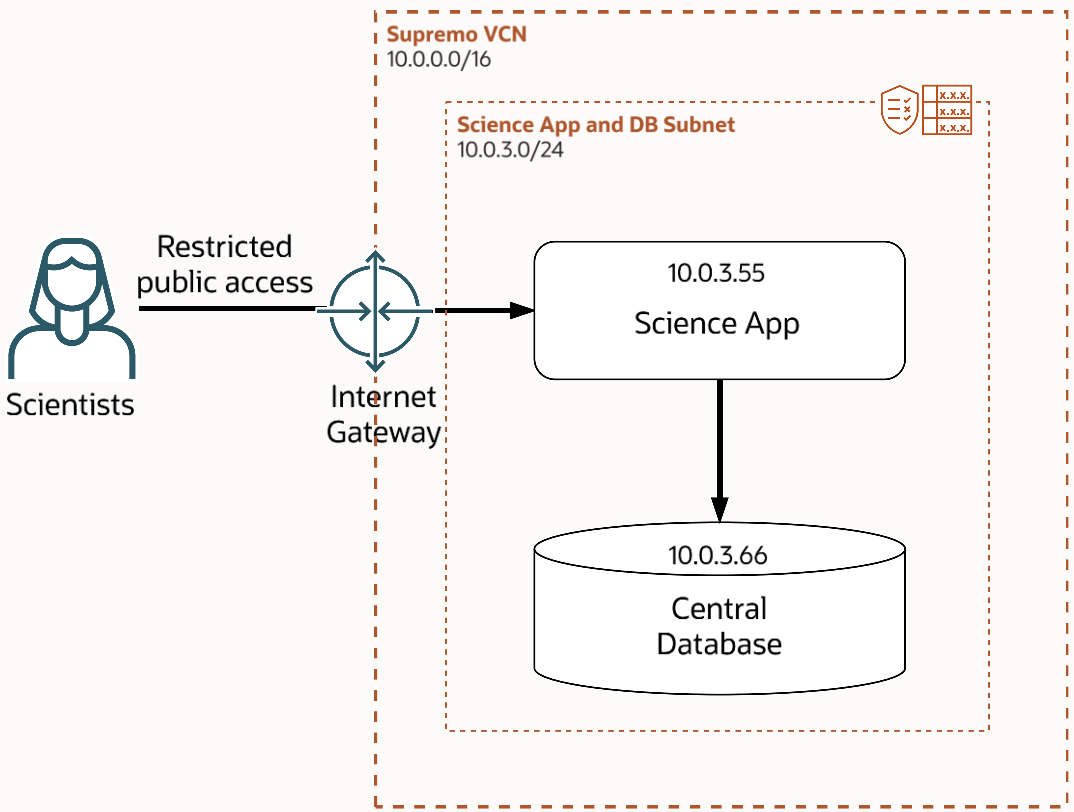



```
# Literal relations
nsg(App, CD);
nsg(CD, Aggregation);
sl(App, CD);
sl(CD, Aggregation);

# Defined relations
let connect(a, b) :=
    nsg(a, b) && sl(a, b);
```

Intermediate Representation (IR): Assertions

First order logic with Transitive Closure




```
# Transitive closure notation
assert connect^(App, CD)

# Quantification
assert connect(a, b) && !(a = b)
    for all a, b;

# Operations on properties
assert connect(a, b) &&
    application in a.type
    for some a, b;
```

Assertion Checking

PIR statements translated to a finite model finder

- Z3
- CVC5
- Kodkod
- Own engine based on DPLL algorithm

Fixedpoint tasks

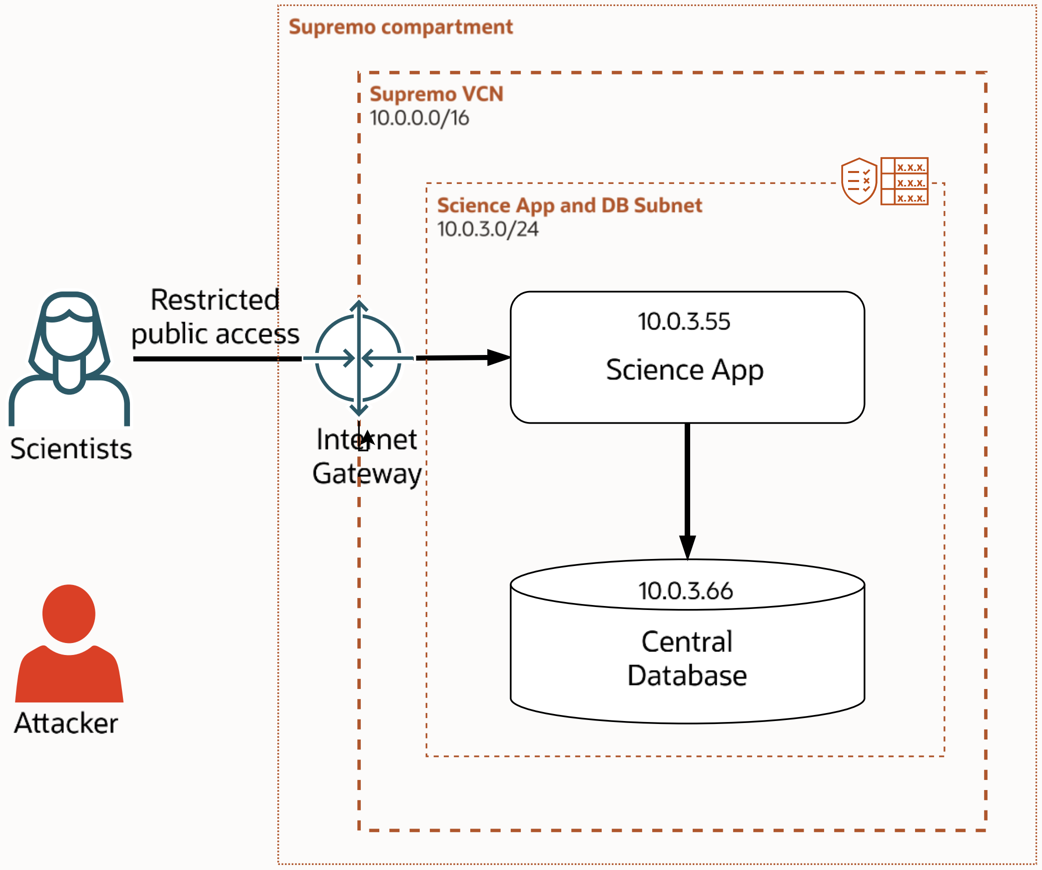
- Built-in analyses using pre-defined configurations of entity properties generating relations
- Z3 datalog engine

Preliminary Evaluation

- Small examples
- Randomly generated programs

Use Cases

Modelling Networking Policies



ROUTE DESTINATION: "0.0.0.0/0" TARGET TYPE: "Internet Gateway"

NSG INGRESS SOURCE-TYPE: "CIDR" SOURCE: "160.39.115.23/24, 103.112.2.0/32, 172.234.10.0/26, 116.67.3.0/24, 136.45.33.0/30, 148.167.51.8/32"

PROTOCOL: "TCP" SOURCE PORT: "ALL" DESTINATION PORT: "888"

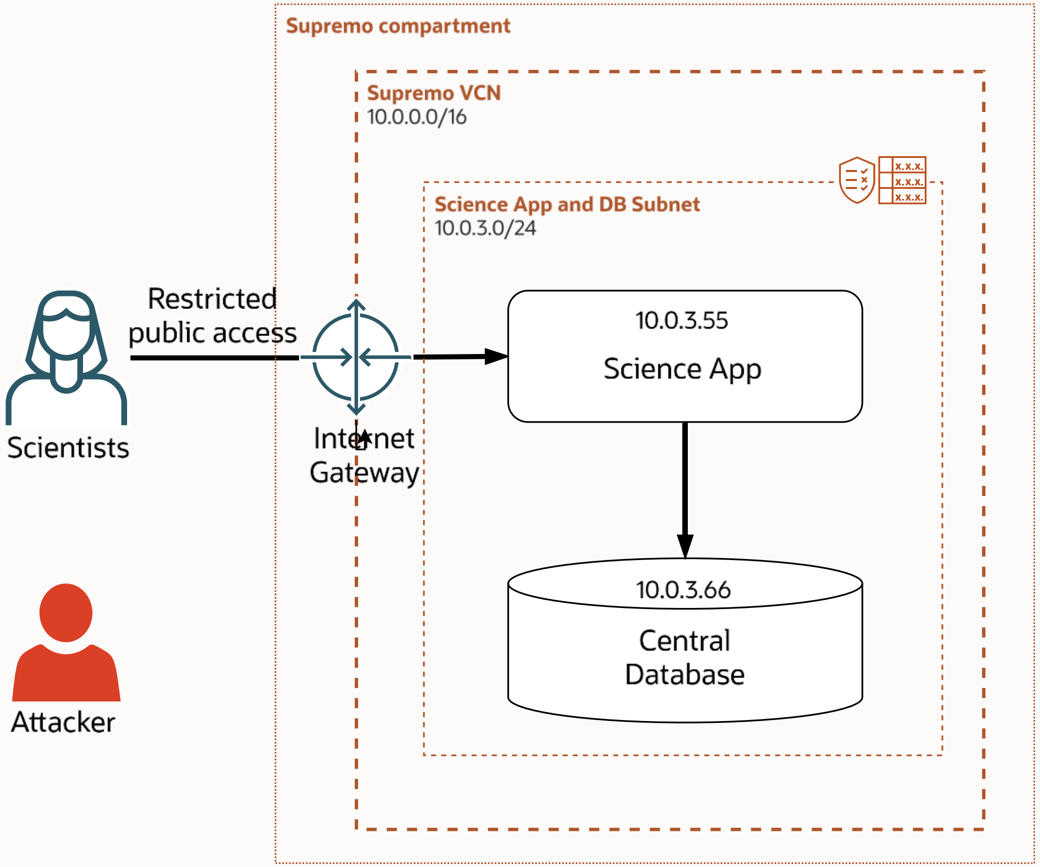
NSG EGRESS SOURCE-TYPE: "CIDR" SOURCE: "10.0.3.66/32"

PROTOCOL: "TCP" SOURCE PORT" "ALL" DESTINATION PORT: "1521"

NSG INGRESS SOURCE TYPE: "CIDR" SOURCE: "10.0.3.55/32"

PROTOCOL: "TCP" SOURCE PORT: "ALL" DESTINATION PORT: "1521"

Checking Networking Security Policies



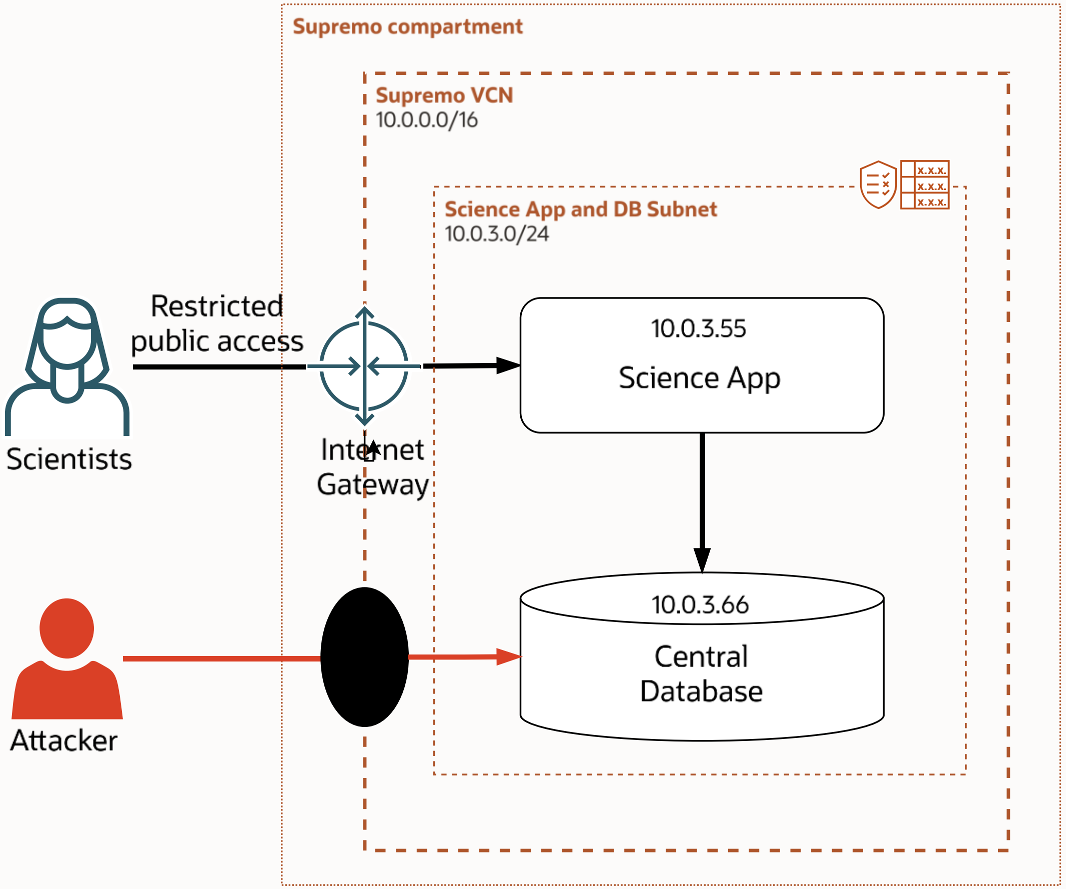

```
# Network security groups (NSG) policies
nsg(Scientists, App);
nsg(App, CD);

# Network security lists (SL) policies
sl(Scientists, App);
sl(App, CD);

# Compound relations
let connect(a,b) := nsg(a,b) && sl(a,b);

# Check that all connections to CD come from App
assert !(connect(node, CD) && !(node = App))
    for all node;
Assertion passed
```

Detecting Networking Policy Misconfigurations



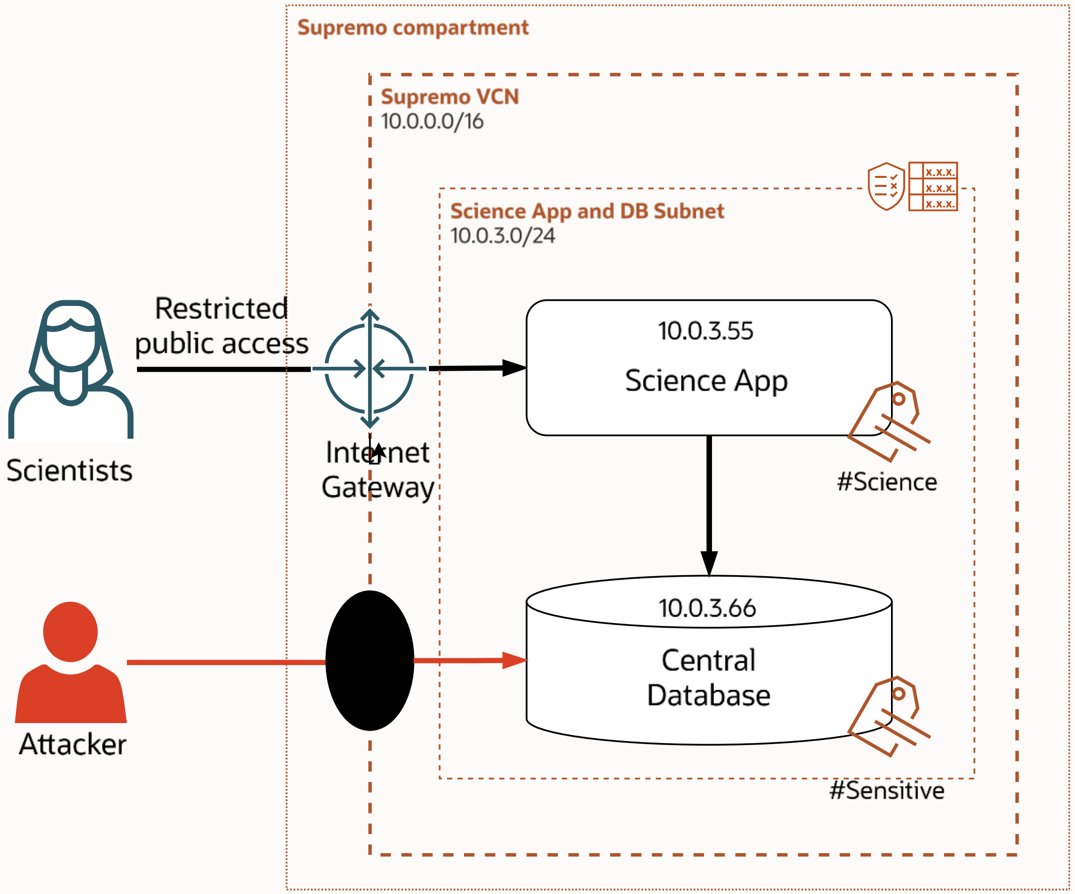

```
# Vulnerable NSG rule added
NSG INGRESS SOURCE
    TYPE: "CIDR"
    SOURCE: "160.39.115.0/24"

# New entity added to the IR
entity Attacker { }

# New NSG connection is added to the IR
nsg(Attacker, CD);

# Assertion fails
assert !(connect(node, CD) && !(node = App))
  for all node;
Assertion failed with: node = Attacker
```

Detecting Networking Policy Misconfigurations



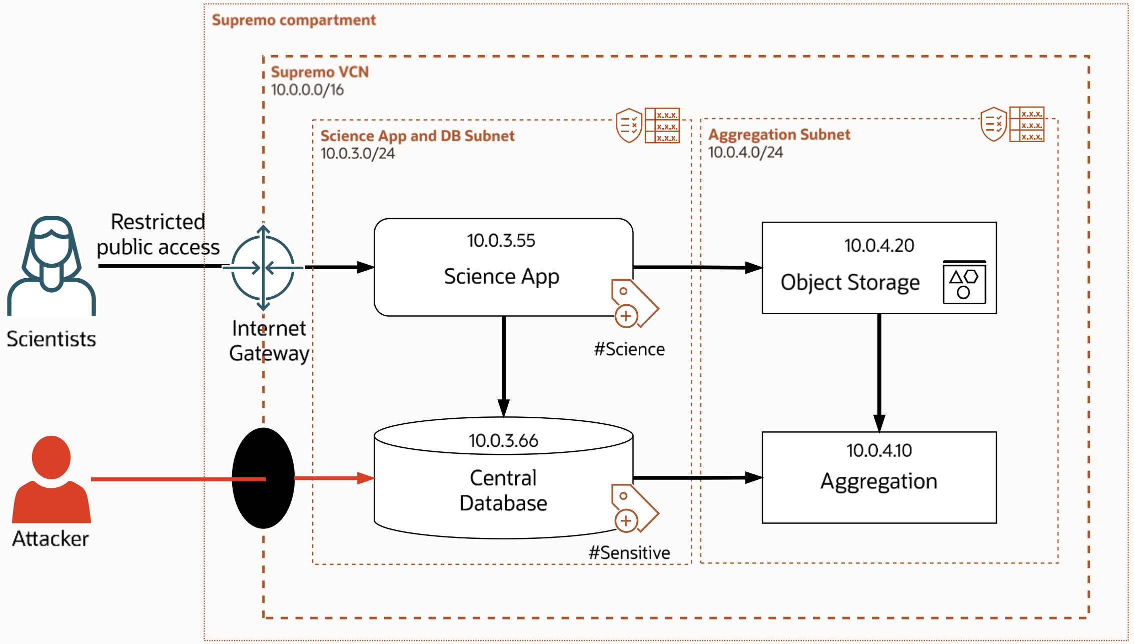

```
# ZPR policy added
In network:prod VCN
   allow app:science endpoints to connect to
   database:sensitive endpoints

# New ZPR connection is added to the IR
zpr(App, CD);

# The connect relation is updated with ZPR
let connect(a,b) :=
     nsg(a,b) && sl(a,b) && zpr(a,b);

# Assertion now passes
assert !(connect(node, CD) && !(node = App))
  for all node;
Assertion passed
```

Checking Identity and Access Management Policies



IAM policy for Object Storage

```
Allow dynamic-group App to manage objects in
compartment Supremo
```

Manage privileges are added to App

```
iam_obj_storage_object_delete(App, OS);
iam_obj_storage_object_version_delete(App, OS);
...
```

IAM verbs for Object Storage:

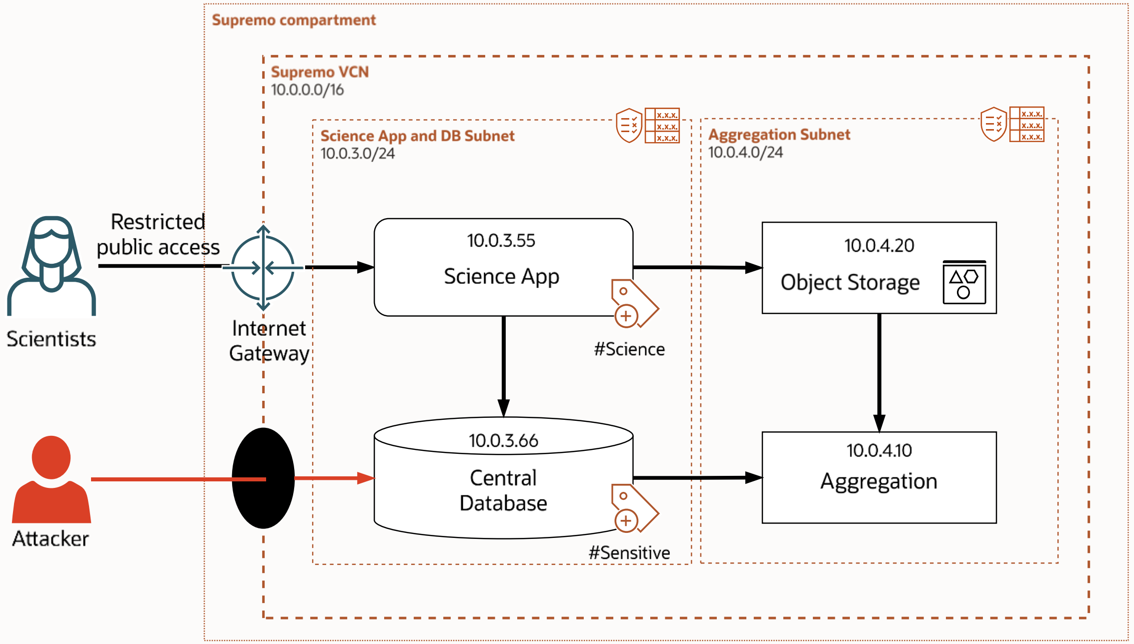
```
let iam_obj_storage_obj_inspect(a,b) :=
 iam_obj_storage_obj_object_inspect(a,b)
...
```

Check that App can only *write* to Object Storage

```
assert iam_obj_storage_obj_use(App, OS) &&
       !(iam_obj_storage_obj_manage(App, OS));
```

Assertion failed

Checking Security Zone Policies




```
# Existing public bucket
entity ObjectStorage {
    visibility: [ public ]
}

# Compartment of the bucket
compartment(Supremo, ObjectStorage)

# Security Zone policy is added for Object Storage
deny public_bucket

# IR is updated accordingly
sz_no_public_bucket(Supremo)

# Check that no bucket violates the security zone policy
assert !(compartment(comp, os)
    && sz_no_public_bucket(comp) && os in os.type
    && public in os.visibility) for all comp, os;
Assertion failed
```

Conclusions and Future Work

The journey so far

- Design and implementation of intermediate representation for reasoning about security policies
- Works well for example applications and use-case scenarios

Next steps

- Evaluation for larger examples and realistic applications
- Explore expressiveness of the intermediate representation
    - especially for the problems requiring fixedpoint computations
- Automatic extraction of application configurations and policies
- Evaluating scalability of assertion checking