

We Have CSPM at Home

Using DuckDB, SQL, and Open Source Tools to Discover Your
Cloud Security Posture

Press Space for next page →

Everyone always asks **whoami**, nobody asks **howami**

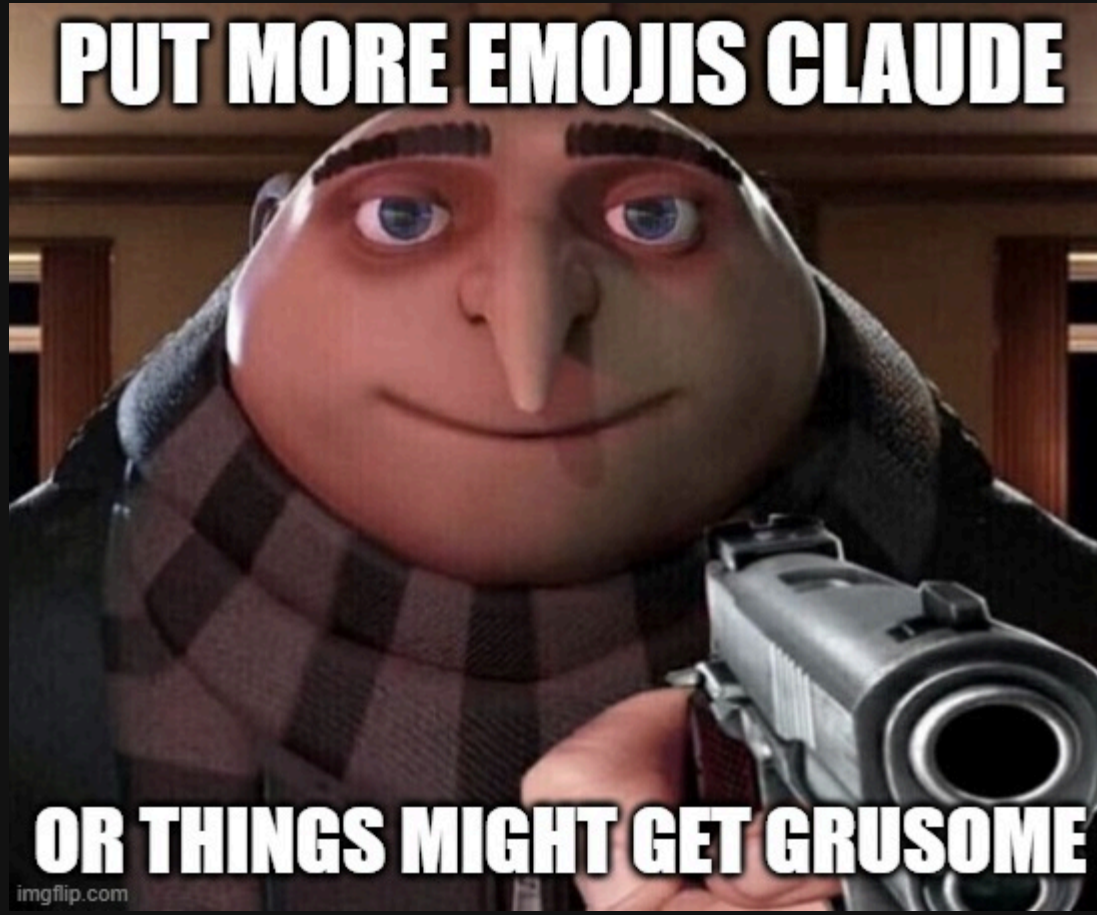
- ☐ Jared Gore
- ☁ Cloud Security Engineer working in ☐ Azure, ☐ AWS, ☐ GCP, and ☐ Oracle ☐
- ☐ 5x GIAC, AWS Security, CISSP
- ♥ helping people transition to tech oriented careers
- ☐ Not doing to great lately tbh

☐ One of my toxic traits is I believe whatever I eat at a restaurant I could make better at home.

<https://jaredgore.com>



Hope You Like Emojis on Slides



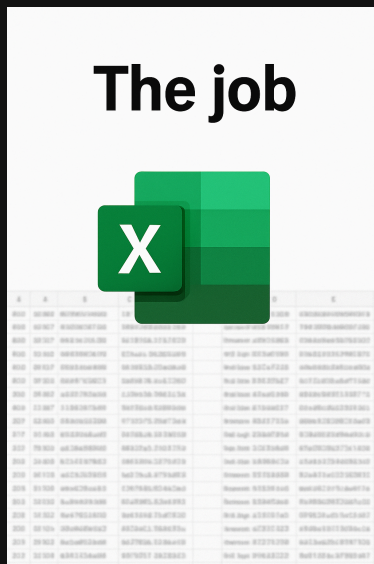
The Problem | "The Meme"

- □ Newcomers can struggle with what "work" looks like
- □ Job descriptions are littered with 100s of open source and commercial tools
- □ Vendors may not make software available to students or career transitioners
- □ How can we bridge the gap between the job description's requirements?



Getting the Job | "The Reality"

- ☐ Folks overthink difficulty of commercial tools for day-to-day work
- ☐ Tools are documented and are easy to use
- ☐ Commercial + OSS tools spit out common formats (JSON, CSV)
- ☐ Logs can be in silos (different Clouds, different workloads)
- ☐ The Final Boss of many Security Jobs is an Excel Doc



How Did We Get Here?

- I Spend a lot of time playing with myself (Homelab, Personal Cloud)
- I didn't always "Have the job" in Security
- At work I get to use a lot of fancy stuff (Wiz)
- At home I do not, neither do students

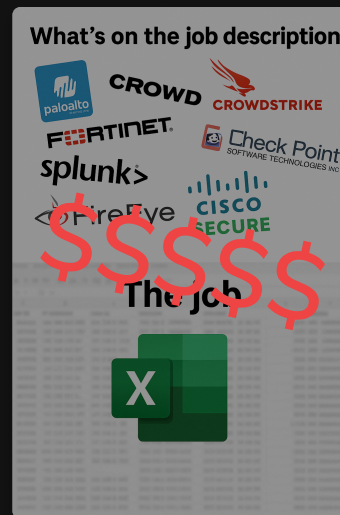
The Learning Gap

□ Students and Career Transitioners Struggle to Get Experience

- Commercial security tools cost \$\$\$\$
- CSV, JSON, Parquet - it's all just standard formats
- Even the fanciest tools return these formats
- Cross-referencing can be painful manual work

The Concepts They Need to be Exposed To

- ☐ Search Event and Audit Logs
- ☐ Working with Output from Any Tool
- ☐ Monitor Netflow + WAF + Load Balancer Logs
- ☐ Querying Cloud Resources for Misconfig








Let's Teach These Concepts using Open Source
Software

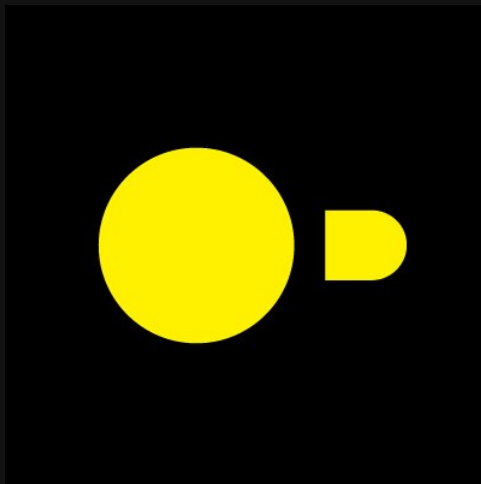
"We have CSPM at home!"

Prior Art

Tool	Highlights
Steampipe	"Infrastructure as SQL", PostgreSQL FDWs, Real-time cloud API queries
CloudCustodian	YAML for cloud compliance, Rules engine for security/governance
Prowler	CIS Benchmark scans, GDPR/HIPPA/SOC2/FedRAMP, Multi-framework support

DuckDB: In-Process SQL Magic

-  Embeddable, in-process SQL OLAP database
- ⚡ Zero setup (download and go)
-  Works with JSON, CSV, Parquet
-  Supports Cloud Storage platforms like S3, GCS, Azure Blob
-  Works with ANY security tool's output
-  Perfect for security data analysis



☐☐ Select Quite Literally What you Want ☐☐

- ☐☐ SQL = structured query language
- ☐ Your queries are like spells or incantations:
 - Filter data based on record location and value
 - Find unique entities in DNS or Web logs
 - Gather findings across multiple security tool outputs
- ☐ Works across virtually all data

```
SELECT what_you_want  -- ☐  
FROM where_it_lives   -- ☐  
WHERE conditions_you_care_about  -- ☐
```



A Disclaimer!



DEMOS

Let's Pretend

- Come to work with me! Let's imagine you are riding alongside at work.
- Cover some common tasks

Let's Start with Wiz Findings

We are going to use our newly found □

□ `Witch|Wizard|Warlock|Goblin` □

powers to ask a CSV file some questions. Let's perform some □ Alchemy to turn what we want into a query:

□ Our First Question: Show me ALL YOUR FIELDS/Give me all the data



```
-- First, let's examine the structure
SELECT * FROM read_csv('open_findings.csv', auto_detect=true);
```


The Result

Let's get specific:

Now that we know the shape of the data

- □ We only desire important fields
- ⚠ We only want to see Critical issues
- □ We only want to see Open issues
- □ That are Publicly Exposed.

```
-- Find critical severity publicly exposed resources
SELECT
    "Title",
    "Resource Name",
    "Resource Type",
    "Resource Region",
    "Resource Platform"
FROM read_csv('open_findings.csv')
WHERE "Severity" = 'CRITICAL'
    AND "Status" = 'OPEN'
    AND "Title" LIKE '%publicly exposed%';
```

The Outcome

Actions we take:

- □ Track Resource Owners Down
- ✕ ~~Fight Them Until It's Fixed~~
- □ Partner closely to remediate or resolve

How can we assist?

- □ Provide example code
- □ Submit PR
- ⚠ Demo the threat

Results

```
> #
```

Security Spell: Find Jenkins Servers at Risk

- CI/CD is a Goldmine for Attackers ☐☐
- Cloud Pipelines have High Privilege ☐
- Jenkins is a PITA to ☐
- Protect Critical Resources ↑

```
SELECT
    "Resource Name", "Title",
    "Severity", "Resource Platform"
FROM read_csv('open_findings.csv')
WHERE
    "Resource Name" LIKE '%jenkins%'
    AND "Status" = 'OPEN'
ORDER BY
    CASE "Severity"
        WHEN 'CRITICAL' THEN 1
        WHEN 'HIGH' THEN 2
        WHEN 'MEDIUM' THEN 3
        ELSE 4
    END;
```


Result:

> #

Outcome:

- ☐ These resources need to be patched
- ☐ Developers are underwater
- ☐ SRE and Dev Services have long backlogs
- ☐ Nobody makes progress being mean
 - How can Security Teams help?
 - How are your Leaders helping?

Tailpipe: DuckDB-Powered SIEM for the Cloud



What is Tailpipe?

- ☐ Open-source SIEM built on DuckDB
- ☐ Developed by Turbot (creators of Steampipe)
- ☐ Collects logs from various sources:
 - ☐ Robust plugin system
 - ☐ Flow logs
 - ☐ Control plane audit logs
 - ☐ Kubernetes logs
- ⚡ Analyzes millions of events in seconds
- ☐ Works completely offline with your logs

Setting up Tailpipe

Step 1: Install the Go Binary

- Simple one-liner script to setup Tailpipe locally.

```
`sudo /bin/sh -c "$(curl -fsSL https://tailpipe.io/install/tailpipe.sh)"`
```

Step 2: Create your Configuration File

- Tailpipe config is `HCL2`` and looks similar to Terraform

```
# Install AWS plugin
tailpipe plugin install aws

# Configure AWS connection (save to ~/.tailpipe/config/aws.tpc)
connection "aws" "prod" {
  profile = "your-aws-profile-name"
}

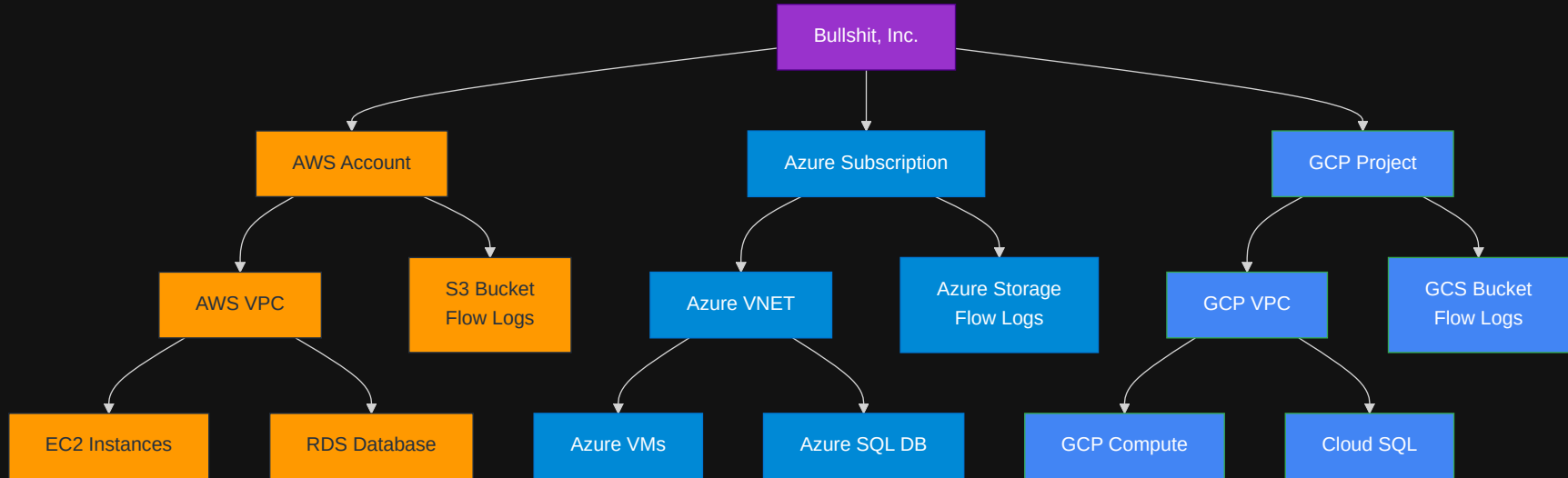
partition "aws_cloudtrail_log" "my_s3_bucket" {
  source "aws_s3_bucket" {
    connection = connection.aws.prod
    bucket = "your-cloudtrail-bucket-name"
```


Step 3: Collect Them Logs!

```
# Collect logs
export AWS_PROFILE=log-archive
$ tailpipe collect aws_cloudtrail_logs.my_s3_buckets
```

BSI, Inc. - Fake It Till You Make it

- When I don't have it, I make it all up
- Spend as little \$\$\$ as possible
- Set it up as close to real as possible



⚠️ Tailpipe to detect AWS Root Logins

- Bullshit, Inc. has setup AWS Organizations and SSO for admins to perform daily tasks.
- To further security operations, BSI built a query to detect any "Root" level login to an AWS Account in their org:

```
-- Root account usage (high-risk activity)
SELECT
    event_time, event_name,
    source_ip_address, user_agent
FROM aws_cloudtrail_log
WHERE user_identity.type = 'Root'
    AND user_identity.session_context.session_issuer IS NULL
ORDER BY event_time DESC;

-- Ideally, this query should return few or no results
-- Root account should rarely be used in a secure environment
```

□ We Have a Hit! □

Unfortunately we found root account activity.

What Happened?

Admin gets an email "AWS Estimated Charges Exceeding Threshold"...

- panic
- fear
- these are unfortunately paths to the dark side

Our illustrious admin tries to login via his FEDERATED secure SSO account but it fails! The admin is stunned.
A poorly configured Billing policy results in not being able to see the bill.

The Consequences of Fake Business ☐☐

Left with a possible bankrupting cloud bill our admin breaks best practices and logs in to see the heartbreak.

The Outcome

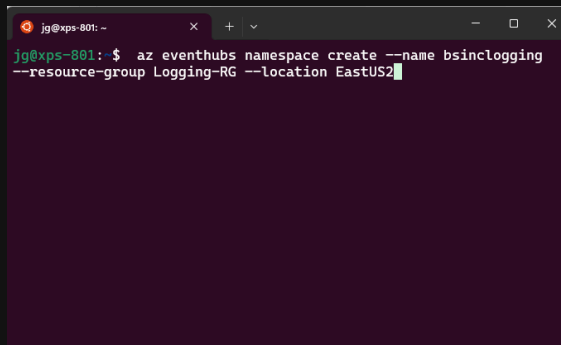
- ⚠ Not his first offense
- ☐ Overbudget and out of control
- ☐ Policy to restrict Root Account activity
- ☐ Performance Improvement Plan

Capturing Cloud Configuration

- □ CSPM tools save the state of cloud resources via JSON
- □ They make it easy to search for configuration mishaps
- □ How can we save cloud resource configuration into DuckDB?
- □ How can we query for insecure cloud resources?

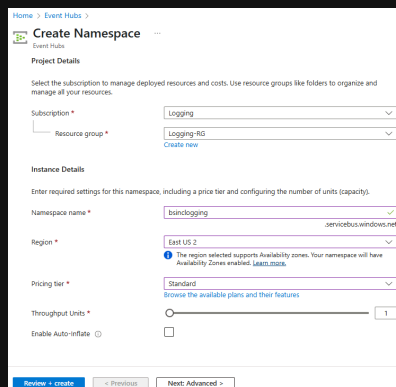
Demystifying the Cloud

What is Configured??




```
jg@xps-801: ~  
jg@xps-801: $ az eventhubs namespace create --name bsinclogging  
--resource-group Logging-RG --location EastUS2
```

Managing cloud resources via
Command Line



The screenshot shows the 'Create Namespace' page in the Azure Portal. It is divided into two main sections: 'Project Details' and 'Instance Details'. In the 'Project Details' section, the 'Subscription' is set to 'Logging' and the 'Resource group' is 'Logging-RG'. In the 'Instance Details' section, the 'Namespace name' is 'bsinclogging', the 'Region' is 'East US 2', and the 'Pricing tier' is 'Standard'. There is a 'Throughput Units' slider set to 1 and an 'Enable Auto-inflate' checkbox that is unchecked. At the bottom, there are buttons for 'Review & Create', 'Previous', and 'Next: Advanced'.

Working with resources in the
Cloud Portal



```
resource "azurerm_resource_group" "example" {  
  name     = "example-resources"  
  location = "West Europe"  
}  
  
resource "azurerm_eventhub_namespace" "example" {  
  name                = "example-namespace"  
  location             = azurerm_resource_group.example.location  
  resource_group_name = azurerm_resource_group.example.name  
  sku                 = "Standard"  
  capacity            = 2  
  
  tags = {  
    environment = "Production"  
  }  
}
```

Deploying infrastructure with
Infrastructure as Code

Regardless of how you manage your cloud, the inputs stay consistent.

DuckDB can load outputs from any CSP CLI tool!

Level 1: EZ Mode in Azure

- Let's import some Azure NSG Rules into DuckDB:

```
# Get NSG rules for a specific NSG including default ones
az network nsg rule list --include-default --resource-group your-resource-group --nsg-name your-nsg-name > azure_nsg_rules.
```

Prepare yourself this query is a big boy:

```
-- Azure NSG Rules with correct structure and empty string handling
CREATE OR REPLACE TABLE azure_nsg_rules AS
SELECT
  'Azure' as cloud_provider,
  id as rule_id,
  'your-nsg' as group_name, -- Replace with actual NSG name
  name as rule_name,
  protocol,
  sourceAddressPrefix as source_cidr,
  destinationPortRange as port_range,
  CASE
    WHEN destinationPortRange = '*' THEN NULL
    WHEN destinationPortRange = '' THEN NULL
    WHEN POSITION('-' IN destinationPortRange) > 0 THEN
      CAST(SPLIT_PART(destinationPortRange, '-', 1) AS INTEGER)
    ELSE CAST(destinationPortRange AS INTEGER)
```

Results

Tried for my crimes, placed into the contraption

EZ Mode in AWS

```
# Get security groups
```

```
aws ec2 describe-security-groups > aws_security_groups.json
```

```
-- Create a table for AWS security group ingress rules with CIDR sources
```

```
CREATE OR REPLACE TABLE aws_security_group_ingress_cidr_rules AS
```

```
SELECT
```

```
    sg.GroupId AS group_id,  
    sg.GroupName AS group_name,  
    sg.VpcId AS vpc_id,  
    sg.Description AS group_description,  
    'Ingress' AS direction,  
    perm.IpProtocol AS ip_protocol,  
    perm.FromPort AS from_port,  
    perm.ToPort AS to_port,  
    cidr.CidrIp AS source_cidr,  
    NULL AS source_cidr_description
```

```
FROM
```

```
    read_json_auto('aws_security_groups.json') AS root_data  
    CROSS JOIN UNNEST(root_data.SecurityGroups) AS sg_list(sg)  
    CROSS JOIN UNNEST(sg.IpPermissions) AS perm_list(perm)  
    CROSS JOIN UNNEST(perm.IpRanges) AS cidr_list(cidr);
```

```
SELECT * FROM aws_security_group_ingress_cidr_rules;
```

Result

EZ Mode in GCP

```
# Get all firewall rules in the project
```

```
gcloud compute firewall-rules list --format=json > gcp_firewall_rules.json
```

```
# Optionally, filter by network
```

```
gcloud compute firewall-rules list --filter="network=your-network" --format=json > gcp_network_firewall_rules.json
```

```
-- Create a table of GCP firewall rules with security focus
CREATE OR REPLACE TABLE gcp_firewall_rules AS
SELECT
  rules.id,
  rules.name,
  rules.direction,
  array_to_string(rules.sourceRanges, ',') AS source_cidr,
  rules.priority,
  rules.description,
  CASE
    WHEN rules.sourceRanges IS NOT NULL
      AND array_contains(rules.sourceRanges, '0.0.0.0/0') THEN TRUE
    ELSE FALSE
  END AS internet_exposed,
  CASE
    WHEN rules.name LIKE '%ssh%' THEN 'SSH'
    WHEN rules.name LIKE '%rdp%' THEN 'RDP'
    WHEN rules.name LIKE '%all%' THEN 'ALL TRAFFIC'
    ELSE 'OTHER'
  END AS service,
  CASE
    WHEN rules.name LIKE '%all%' THEN 'CRITICAL'
    WHEN rules.name LIKE '%ssh%' OR rules.name LIKE '%rdp%' THEN 'HIGH'
```


name varchar	direction varchar	source_cidr varchar	service varchar	priority int64	risk_level varchar	description varchar
allow-ssh	INGRESS	0.0.0.0/0	SSH	1000	CRITICAL	

```

SELECT
  name,
  direction,
  source_cidr,
  service,
  priority,
  risk_level,
  description
FROM gcp_firewall_rules
WHERE
  internet_exposed = TRUE
  AND direction = 'INGRESS'
ORDER BY
  CASE
    WHEN risk_level = 'CRITICAL' THEN 1
    WHEN risk_level = 'HIGH' THEN 2
    ELSE 3
  END,
  name;

```

name varchar	direction varchar	source_cidr varchar	service varchar	priority int64	risk_level varchar	description varchar
allow-ssh	INGRESS	0.0.0.0/0	SSH	1000	CRITICAL	

)

Multi-Cloud Firewall Findings - The query that never ends ☐☐

```
-- Comprehensive cross-cloud security findings
WITH aws_findings AS (
  SELECT
    'AWS' AS cloud,
    group_name AS resource_name,
    'Ingress' AS direction,
    ip_protocol AS protocol,
    CAST(from_port AS VARCHAR) || '-' || CAST(to_port AS VARCHAR) AS ports,
    source_cidr AS cidr,
    CASE
      WHEN ip_protocol = '-1' THEN 'CRITICAL'
      WHEN from_port IN (22, 3389, 3306) THEN 'HIGH'
      ELSE 'MEDIUM'
    END AS risk_level,
    CASE
      WHEN ip_protocol = '-1' THEN 'ALL TRAFFIC'
      WHEN from_port = 22 THEN 'SSH'
      WHEN from_port = 3389 THEN 'RDP'
      WHEN from_port = 3306 THEN 'MySQL'
      WHEN from_port = 80 THEN 'HTTP'
      WHEN from_port = 443 THEN 'HTTPS'
      ELSE 'PORT ' || from_port
```

```
SELECT * FROM azure_findings
UNION ALL
SELECT * FROM gcp_findings
) AS all_findings
ORDER BY
CASE
  WHEN risk_level = 'CRITICAL' THEN 1
  WHEN risk_level = 'HIGH' THEN 2
  ELSE 3
END,
cloud,
resource_name;
```

cloud varchar	resource_name varchar	direction varchar	service varchar	protocol varchar	ports varchar	cidr varchar	risk_level varchar
AWS	cspmlab014235-all-open	Ingress	ALL TRAFFIC	-1	NULL	0.0.0.0/0	CRITICAL
Azure	your-nsg/AllowInternetOutBound	Outbound	ALL TRAFFIC	*	*	*	CRITICAL
Azure	your-nsg/AllowOutboundInternet	Outbound	ALL TRAFFIC	*	*	*	CRITICAL
GCP	allow-ssh	INGRESS	SSH	SSH	SSH	0.0.0.0/0	CRITICAL
AWS	cspmlab014235-mysql-open	Ingress	MySQL	tcp	3306-3306	0.0.0.0/0	HIGH
AWS	cspmlab014235-rdp-open	Ingress	RDP	tcp	3389-3389	0.0.0.0/0	HIGH
AWS	cspmlab014235-ssh-open	Ingress	SSH	tcp	22-22	0.0.0.0/0	HIGH
Azure	your-nsg/SSH	Inbound	SSH	Tcp	22	*	HIGH
AWS	ha-lab-alb-sg	Ingress	HTTP	tcp	80-80	0.0.0.0/0	MEDIUM
Azure	your-nsg/Allow-SSH-Egress	Outbound	NULL	Tcp		*	MEDIUM
Azure	your-nsg/AllowOutboundSSH	Outbound	NULL	Tcp		*	MEDIUM
Azure	your-nsg/SSH-Alt	Inbound	PORT 2222	Tcp	2222	*	MEDIUM
12 rows							8 columns

Automation Opportunities

- CSP CLI scripts can be Scheduled with Cron Jobs [Example Here](#)

```
0 1 * * * /path/to/cloud_firewall_collector.sh
```

This will collect your cloud firewall data daily at 1:00 AM, analyze it for security risks, and send you a report with the findings.

- Python Scripts could also be a great solution:

```
import boto3
import duckdb
import json

def aws_sg_to_duckdb(db_file='security.db', regions=['us-east-1']):
    """Collect AWS security group rules and save to DuckDB."""
    # Connect to DuckDB
    con = duckdb.connect(db_file)

    # Collect security groups from all specified regions
    all_sgs = []
    for region in regions:
        ec2 = boto3.client('ec2', region_name=region)
        response = ec2.describe_security_groups()
```

Where do we go from here? ☐☐

- ☐ To say this solution has rough edges
- ☐☐ DuckDB needs developers with a security focus to harness it's potential in Security
- ☐ Tailpipe is a great example, but there is more work to be done
- ☐ I have been building a more scalable solution in Golang to smooth out some of the details
- Unfortunately I am ☐ Developer

Corkscrew Demo



MY PROJECT'S HOPES AND DREAMS

Questions?