RSAConference2020

San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: CSV-T11

Continuous Security and Governance in the Cloud Using a Graph-Based CMDB

Sean Catlett

CISO, Reddit

Erkang Zheng

CISO, LifeOmic
Founder, JupiterOne

reddit

JUPITERONE

#RSAC

Overview

- CMDBs – *been there, done that?*
- Reddit's Vulnerability Management approach with a graph-based CMDB foundation
- Why this matters

How well do you know your own environments?

You

Bad guy

"know yourself and know your enemy –
a hundred battles, a hundred victories."

You need a **good CMDB** for this.

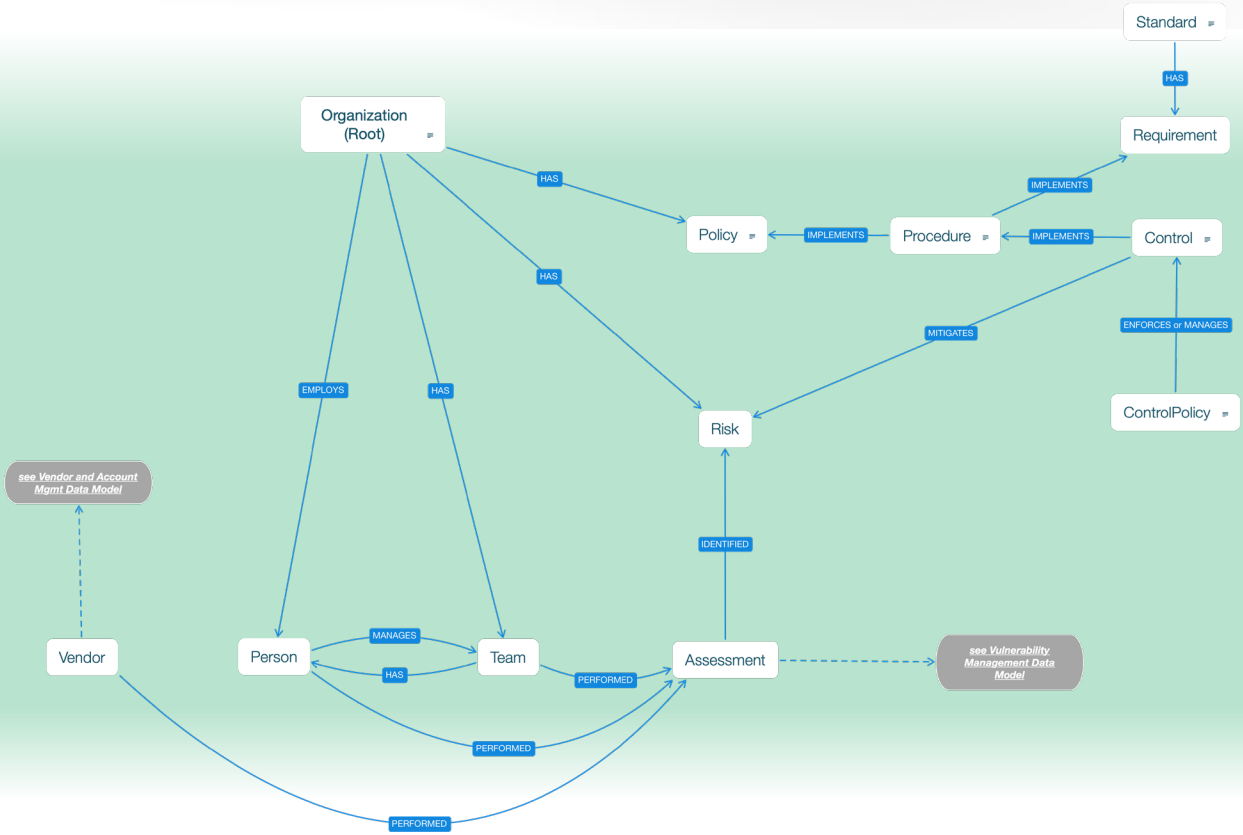
Context comes from relationships.
Relationships are represented in graphs.
It's time we think in graphs, not lists.

Aggregate data from all sources into the graph

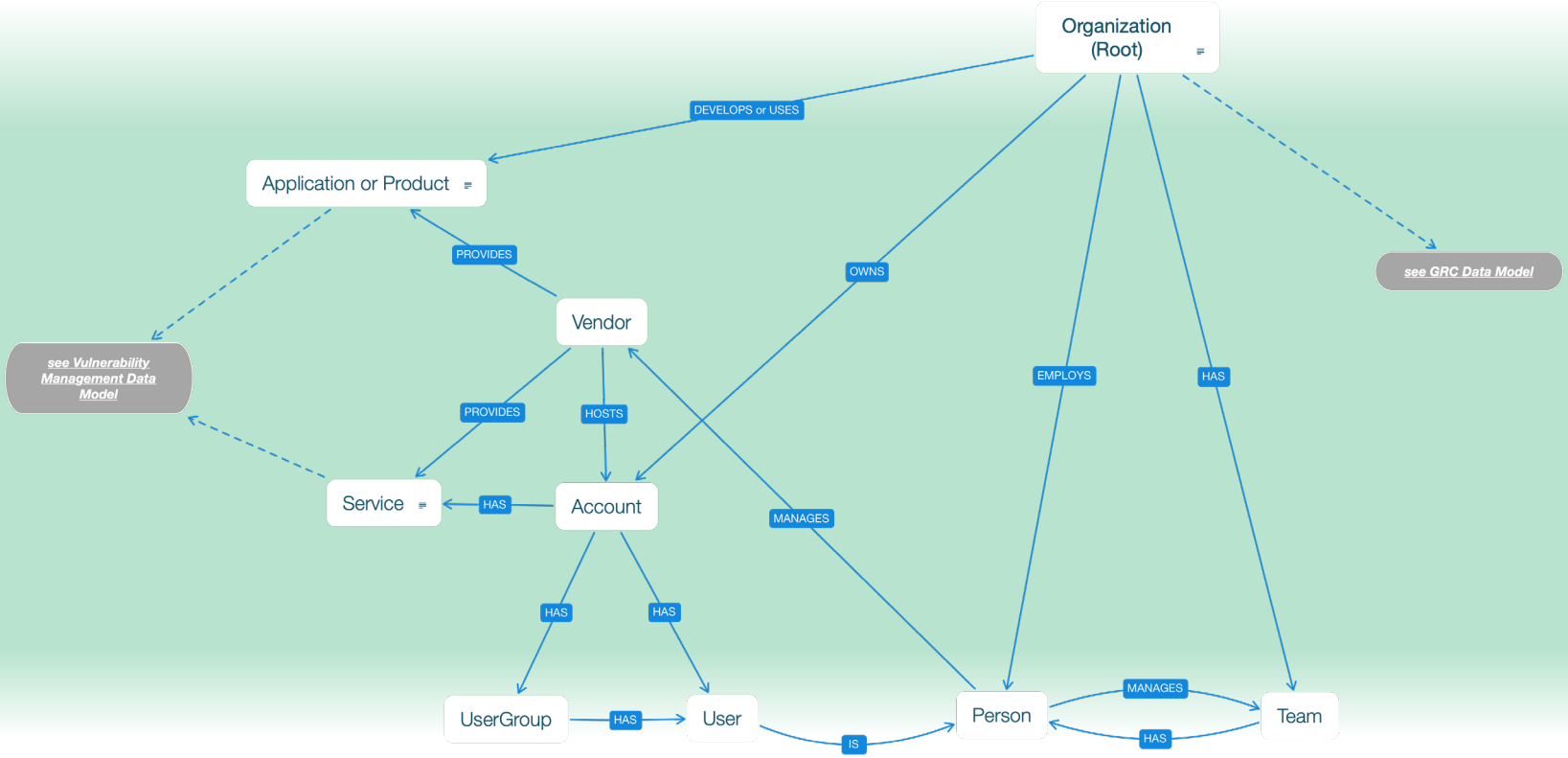
You can use a graph to represent any and all entities + relationships for your security operations and compliance. For example:

- **Policies, Controls, Risk, Compliance**
- **Organization Users, Accounts, Vendors**
- **Vulnerability Management**
- **Network and Endpoint Infrastructure**
- **AWS Resources and IAM Permissions**
- etc.

Policies, Controls, Risk, Compliance Graph



Organization Users, Accounts, Vendors Graph



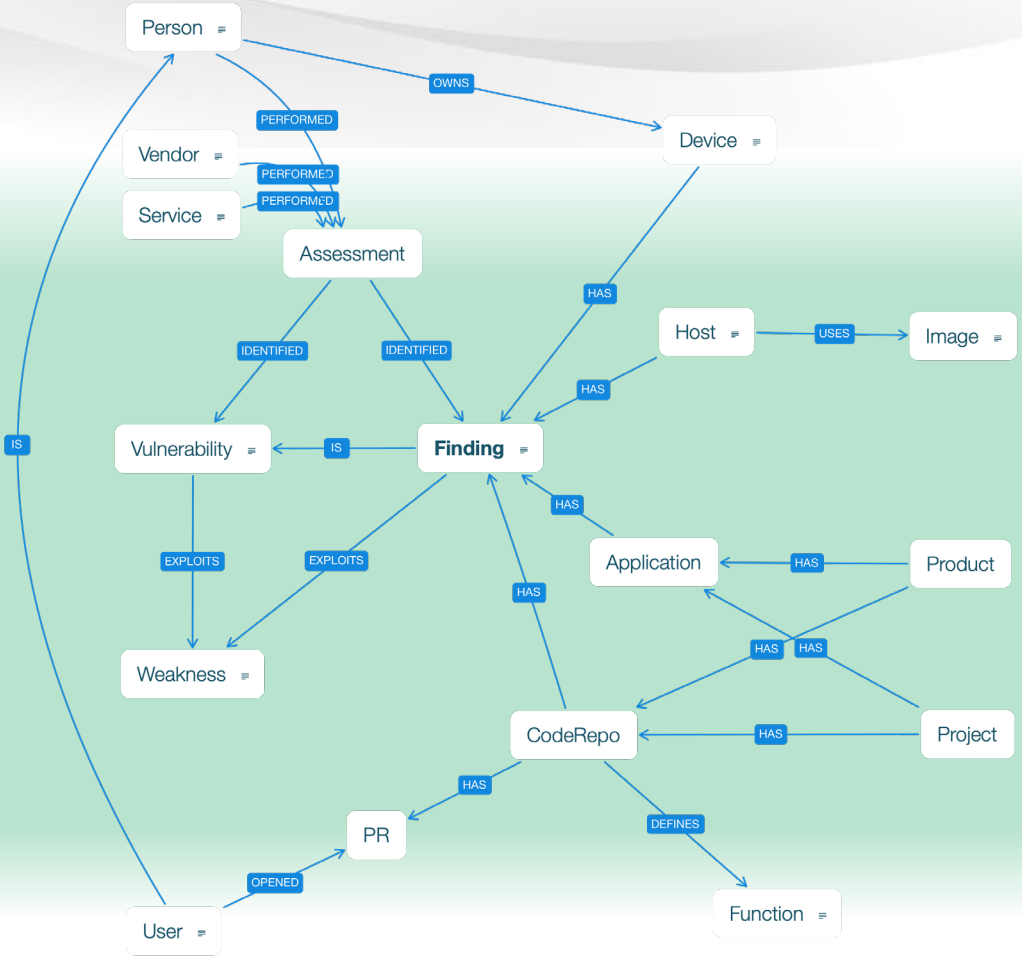
Vulnerability Management Graph

Ask questions like:

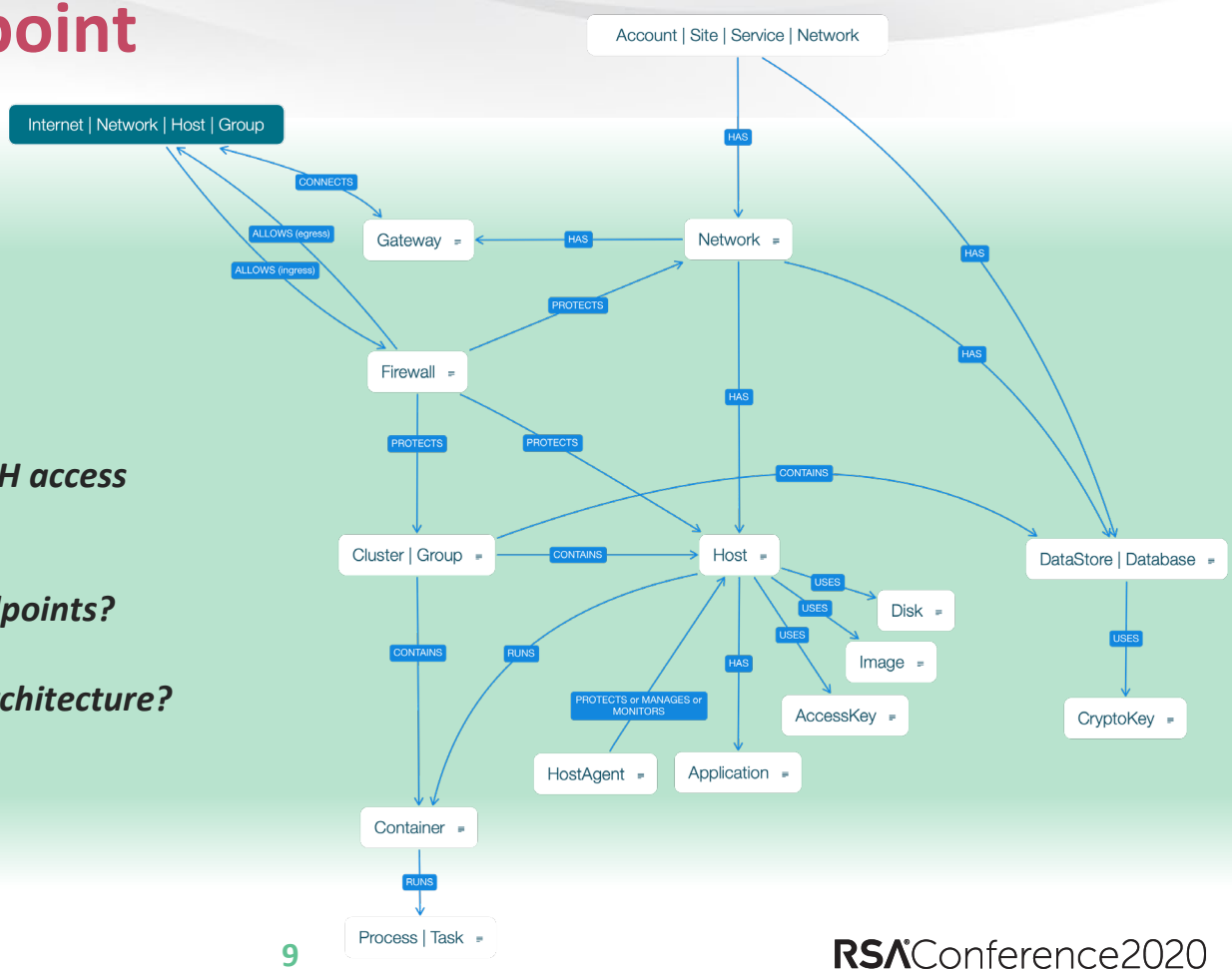
Which PRs / developer introduced new vulnerability findings this past week?

What patterns exist in the findings?

What weakness are most prevalent?



Network and Endpoint Infrastructure Graph



Ask questions like:

Which hosts are allowed inbound SSH access from the Internet?

Is disk encryption enabled on all endpoints?

What is the container / serverless architecture?

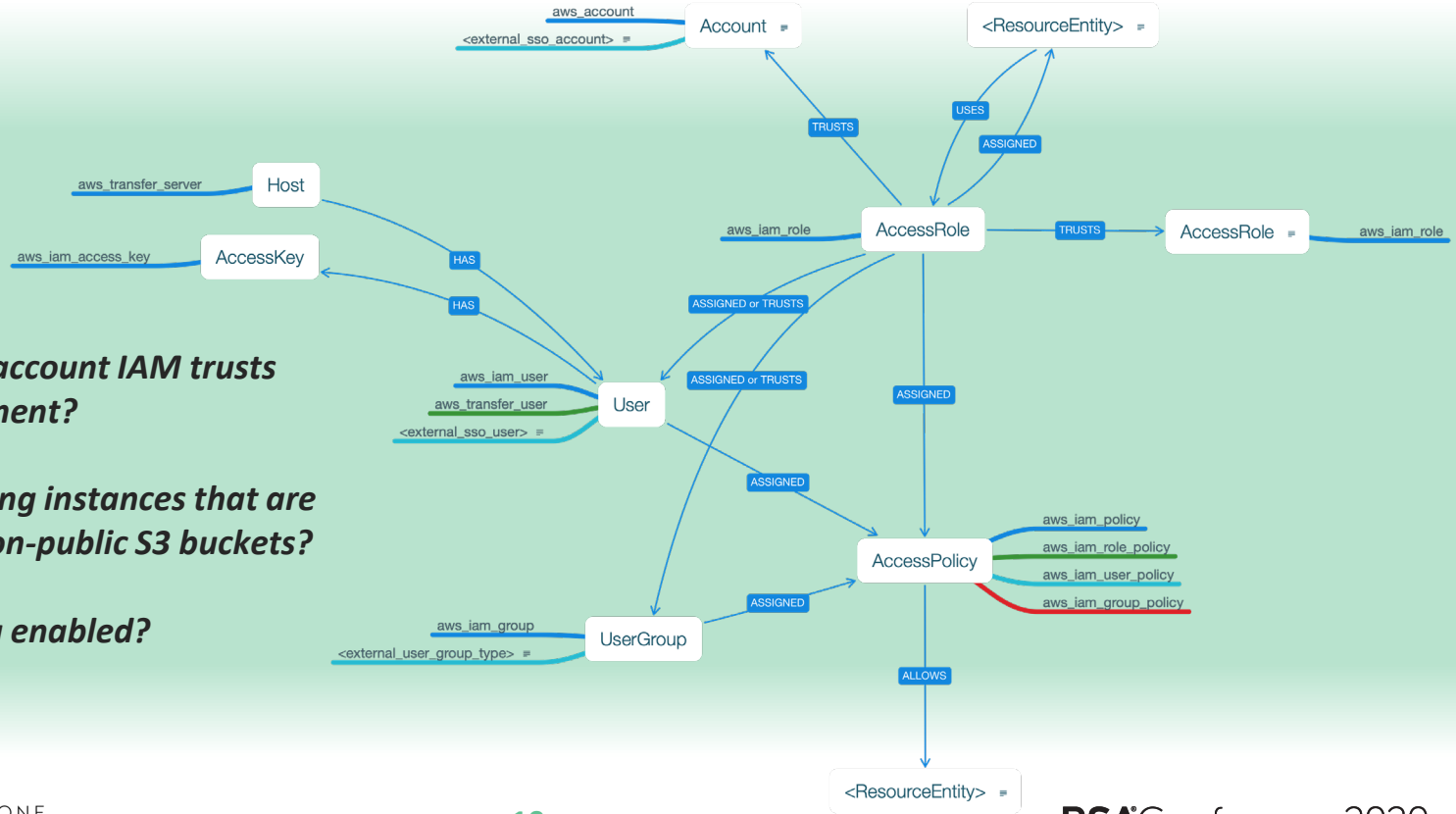
AWS Resources and IAM Permissions Graph

Ask questions like:

What are the cross-account IAM trusts in my AWS environment?

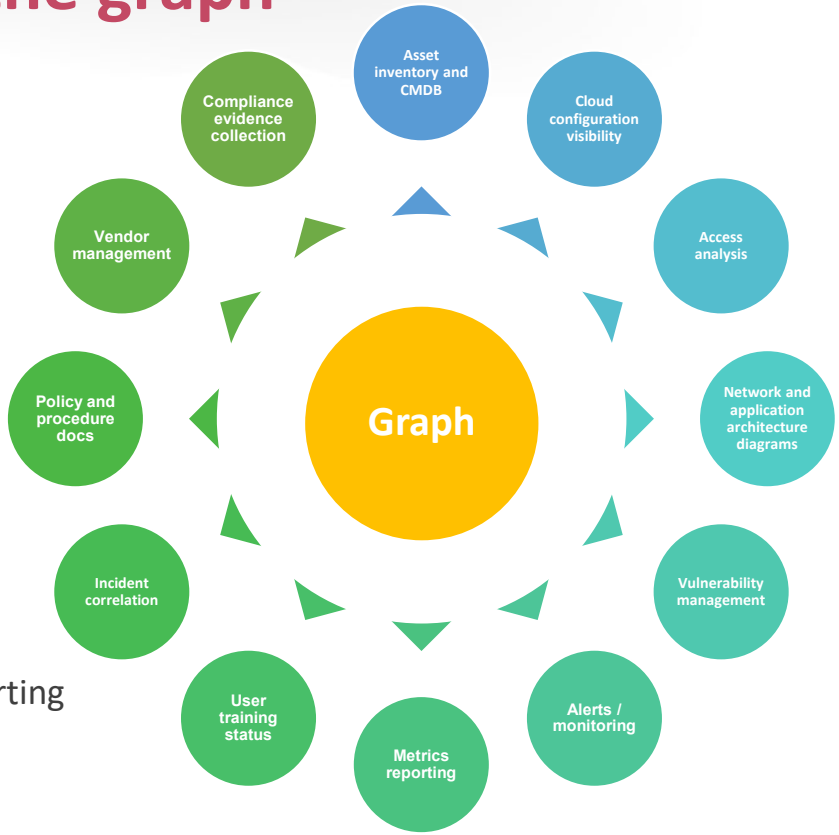
Are there public facing instances that are allowed to access non-public S3 buckets?

Is CloudTrail logging enabled?



Do all of these by querying the graph

- Asset inventory and CMDB
- Vulnerability management
- Cloud configuration visibility
- Incident analysis assistance
- Access review and analysis
- Network and application architecture diagrams
- Alerts / monitoring
- Metrics reporting
- User training status
- Policies and procedures documentation
- Vendor management
- Data-driven compliance evidence collection and reporting
- etc.



Answer complex questions with graph queries

For example:

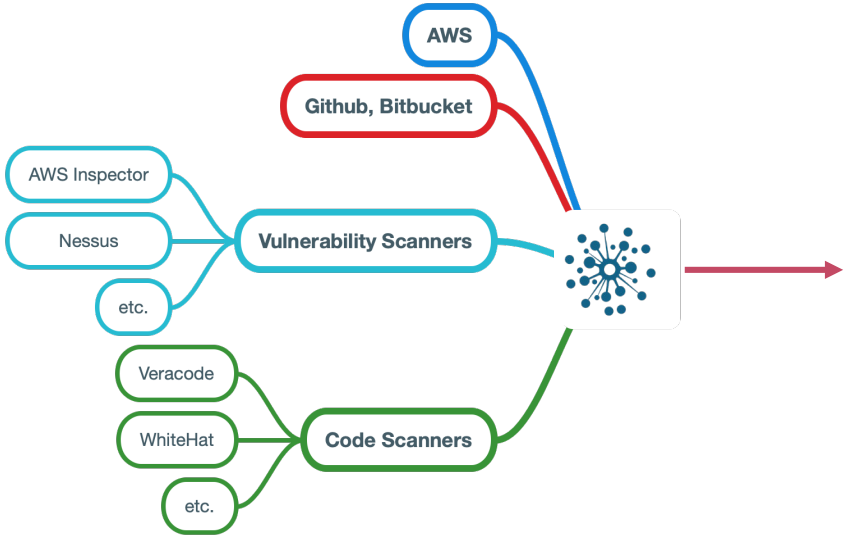
Are there Internet-facing EC2 instances that are allowed access to non-public S3 buckets?

This question involves analysis of the following conditions:

- **EC2 instances that are active**
- **The security groups associated with these instances allow access to/from the Internet**
- **The instances are in a publicly routable network/VPC**
- **The network/VPC has ACLs allowing Internet access**
- **EC2 instances have IAM Roles assigned to them**
- **The IAM Policies associated with the roles give them access to one or more S3 buckets**
- **The S3 buckets are not tagged / classified as Public**

This can be done via graph query, across multiple AWS accounts, covering thousands of instances, roles, policies, and buckets, instead of hours or days of manual effort.

Use a graph DB for vulnerability mgmt.



- Ingest data
- Run query to correlate
  Finding → Code Repo → PR → Developer
- Build dashboards
- Set up alerts

RSAConference2020

How Reddit uses a Graph-Based CMDB for Vulnerability Management

Vulnerability Management in a nutshell

Source Image from https://github.com/OWASP/www-project-vulnerability-management-guide

Vulnerability Management in a nutshell

- TL;DR: Divided up into 3 Cycles
  1. **Detection**
     - Infrastructure (Packages)
     - Applications (Libraries)
  2. **Reporting**
     - Scan Coverage (Scanned everything?)
     - Metrics (Measure progress)
  3. **Remediation**
     - Owners (Who can patch?)
     - Scheduling (Agreed SLAs to patch)

Source Image from https://github.com/OWASP/www-project-vulnerability-management-guide

What a Snoo wants, what a Snoo needs

What we wanted

- An inventory of all **infrastructure-**vulnerable packages
- Ability to search our infrastructure for affected packages/libraries
- Identify who are the owners of the servers

What we needed

- Configuration Management Database (CMDB)
- Agentless non-invasive script to collect vulnerability data
- A platform that consolidates data, analysis & reporting

How Reddit manages infrastructure

Deployment

- AWS EC2 deployments via Terraform – Infrastructure treated as code
- All deployments include 'tags' for the service name

Configuration

- Configuration management via Puppet – agent used to/for
  - install/update packages
  - setup application requirements
  - services

```
Verify Package
package { 'openssh':
  ensure => present,
}

Create a File
file { 'motd':
  path => '/etc/motd',
}

Start a Service
service { 'httpd':
  ensure => running,
  enable => true,
}
```

Implementing a CMDB

- **Reddit uses a graph-based CMDB** and Terraform deploys this solution with little to no effort

1. Create one AWS service account (IAM role) with **Security Auditor Permission** on all AWS accounts
2. Add integrations – then schedule to auto-collect data
   - AWS accounts
   - Github, Okta, etc.
3. **Profit!**
   - Asset Management ☑

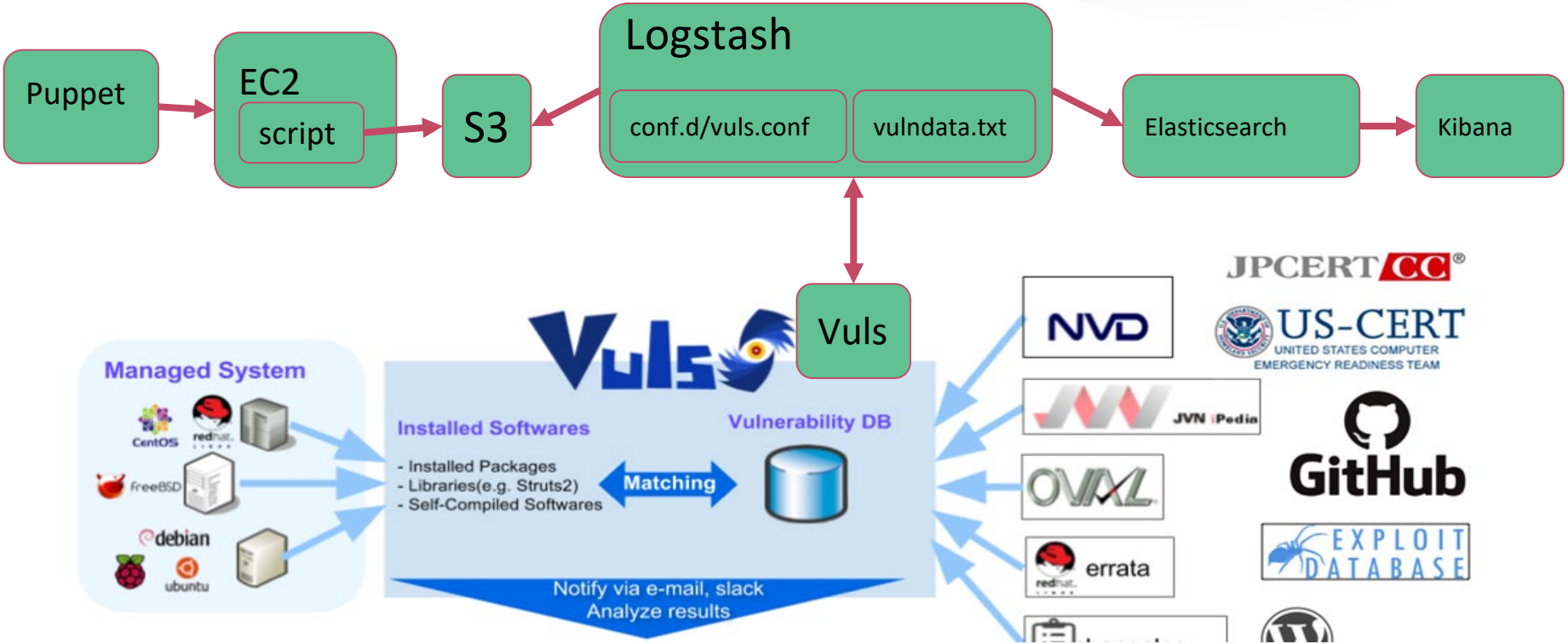
How Reddit leverages a graph database

- All resources are defined as an “entity” on the graph
- Use queries to answer questions about the environment
  - e.g. Find all public security_groups (CIDR 0.0.0.0/0) attached to instances in Production that talks to a Database port.
  - e.g. Find all aws_instances related to a VPC/Tag/Account
- Update Entities with custom fields
- Create dashboards of aggregated resources, not just siloed ones

Using a graph-based cloud CMDB has allowed us to achieve a streamlined vulnerability management process.

Collect Vulns

(We use Vuls @Reddit. You may choose to use a commercial product instead.)



TIL... (gotchas)

- **When scheduling a cron on several servers**
  - Schedule script to execute at random intervals (Puppet supports this)
- **Save time**
  - Talk to your Infrastructure team to understand the environment
  - They will guide you in the right direction and avoid over-engineering
- **Don't individually update CMDB entities 1-to-1**
  - CMDBs may apply rate-limiting, use bulk entity updates
- **Airflow**
  - When doing concurrent requests use pools! Duplication can occur
- **DefectDojo**
  - Deploy using Kubernetes to support threading with multiple cores.

Manage Vulns – the more difficult part

(ElasticSearch – CMDB – Github – Airflow – DefectDojo)

- **Service and Owner Correlation**
  - Identify owners via Github or graph-based CMDB
  - Can iterate over different pieces of information to correlate to product or service ownership
- **Risk Analysis**
  - Prioritize: Public Facing Services, entities with connections to sensitive data stores
  - Create Jira tickets per CVE
  - Use ELK to compare month-over-month vulnerability findings
  - Track number of Jira Tickets Created/In-Progress/Done
- **Remediation**
  - How many vulnerabilities exist per product
  - How many tickets closed/opened, Unassigned vs Assigned
- **Reporting**
  - Track specific risks by environment, product or (future) connection to class of data

How a Digital CMDB Helps Reddit with Vulnerability management

- We prove the % of Servers in Production are Scanned daily
- We know how many services/products exist in our organization
  - Create a break down EC2 Servers missing Service Tag, List of all Services, # of Servers scanned per AWS Account
  - # of Servers exposed to Internet per AWS account/service name
  - List of all Security Groups that interact with Database
  - List of all Wide Open Security Groups (src CIDR 0.0.0.0/0) to internet
- We can immediately verify our % coverage of our environment that has not been vulnerability scanned
- We can enforce users to tag their terraform deployments to be categorized as a product/service
- We can distinguish autoscaled vs non-autoscaled instances
- We have a snapshot of history when instances were scanned
  - In our CMDB and ELK

Future development

Real-time results using Kafka stream processing

- Trigger Vuls Scan script when EC2 Servers deployed by Terraform
- Updates CMDB
- Uploads data to DefectDojo

Here is why all of this matters

Whether it is vulnerability management or another use case...

Continuous Security and Governance must start with

understanding the **entities and relationships** in your environment

Application

- *Challenges in building a graph-based CMDB*
  - Maintain data accuracy and data integrity every time, all the time
  - Provider limitations – rate limit, auth flow, bugs
  - Querying – performance, flexibility, scale
  - Relationship mapping, especially cross environment
- *Next steps*
  - Decide if a CMDB is a tool that fits your infrastructure and objectives
  - Define your vulnerability management SLAs
  - Implement a vulnerability management automation pipeline
  - Build your first report / dashboard

Resources

- **Asset Attack Vectors by Morey J. Harbor, Brad Hibbert**
  - https://www.apress.com/gp/book/9781484236260
- **Open source resources for graph CMDB**
  - https://github.com/jupiterone
  - https://github.com/lyft/cartography
- **OWASP Vulnerability Management Guide**
  - https://github.com/OWASP/www-project-vulnerability-management-guide
- **DefectDojo**
  - https://github.com/DefectDojo/django-DefectDojo
- **Learn Airflow from “Apply Data Science” YouTube Tutorial**
  - https://www.youtube.com/playlist?list=PLYizQ5FvN6pvIOcOd6dFZu3lQqc6zBGp2

Thank You!

Questions?