cloudsecops

PacBot – Open Source Compliance Automation Tool

Published by **Setu** on February 24, 2019

What is PacBot?

PacBot is Policy as Code Bot which does continuous compliance monitoring, compliance reporting and security automation for AWS(as of the date I am writing this post) from T-Mobile. In PacBot, security and compliance policies are implemented as a code. All resources discovered by PacBot are evaluated against a set of policies to gauge policy conformance.

How to Install PacBot?

Installation instructions for PacBot are located here on their official GitHub repo. However, there are some prerequisites that needs to be installed and you can use this EC2 User Data script to automate the prerequisites installation process as well. I spent a couple of days deploying PacBot and finally was able to deploy with a t2.medium instance.

```
#!/bin/bash
#EC2 user script to install Pacbot p

#Setting up Prereqs

cd /opt
yum update -y
yum install git -y
yum install wget -y

```

```
#Install and setup Python3.6

yum install epel-release -y
yum install python36-pip -y
echo alias python3=python3.6 >> ~/.bashrc
echo alias pip3=pip3.6 >> ~/.bashrc
source ~/.bashrc

#Clone Repo

git clone https://github.com/tmobile/pacbot.git

#Other Prereqs
yum -y install java-1.8.0-openjdk docker maven unzip mysql
systemctl start docker
wget https://releases.hashicorp.com/terraform/0.11.8/terraform_0.11.8_linux_amd64.zip
unzip terraform_0.11.8_linux_amd64.zip
mv terraform /usr/bin/
pip3.6 install -r /opt/pacbot/installer/requirements.txt

#Setup UI components
sudo yum install nodejs npm -y
cd /opt/pacbot/webapp
npm install -g @angular/cli@1.6.8
sudo npm install -g bower
sudo npm install
bower install --allow-root

#Copy the default Settings file and create a local.py
cp /opt/pacbot/installer/settings/default.local.py
/opt/pacbot/installer/settings/local.py
```

Once the EC2 server is ready to use, you need to update the local.py settings file with the Access Keys, VPC ID, Subnets(different regions), etc and other requested information. The final steps are to laugh the UI and kick off the terraform build script.

```
vim /opt/pacbot/installer/settings/local.py <<EOF

AWS_ACCESS_KEY = "<>"
AWS_SECRET_KEY = "<>"
AWS_REGION = "<>"
VPC ID: "<>",
CIDR_BLOCKS: ["<>"],
 SUBNETS": ["<>", "<>"]
}
# Launch UI
ng serve &>/dev/null
python3.6 /opt/pacbot/installer/manager.py install
```


It will take up to 20mins to complete the terraform build. Once the build is completed you sh
able to access the Internal ELB URL using a windows server launched within the same VPC or using a

VPN tunnel between the VPC and your on-prem network. Don't forget to update the security group rules to use the Pacbot Internal ELB from you on-prem network. Note the summary of the build success message to access Kibana and Elastisearch cluster URLs that were generated as a part of the build process.

What are the services that the PacBot Installer script deployed?

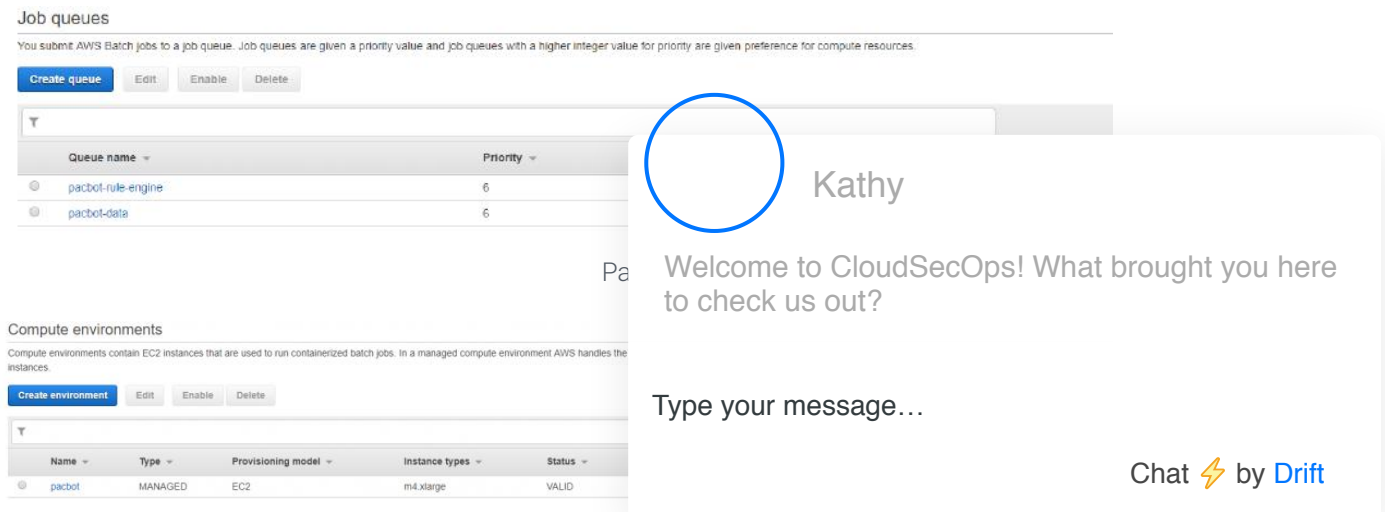
IAM Roles, IAM Policies, S3 Bucket, RDS, MySQL 5.6.X, Elasticsearch Service, Elasticsearch version 5.5, Redshift, Single Node, Batch, Compute environments, Job Definitions and Job Queues, Elastic Container Registry, Repositories – for batch job, API and UI, Elastic Container Service – AWS Fargate, Clusters – for APIs, UI and Batch, Task Definitions – for APIs and UI, Lambda Functions, SubmitBatchJob and SubmitRuleJob, CloudWatch Rules

How does PacBot does Compliance Automation?

Pacbot discovers resources using AWS Batch Jobs and these assets are evaluated against predefined policies(~60) to gauge policy conformance. We can also create and write custom policies as per our organizational compliance needs. There are two main batch jobs that are responsible to achieve compliance monitoring

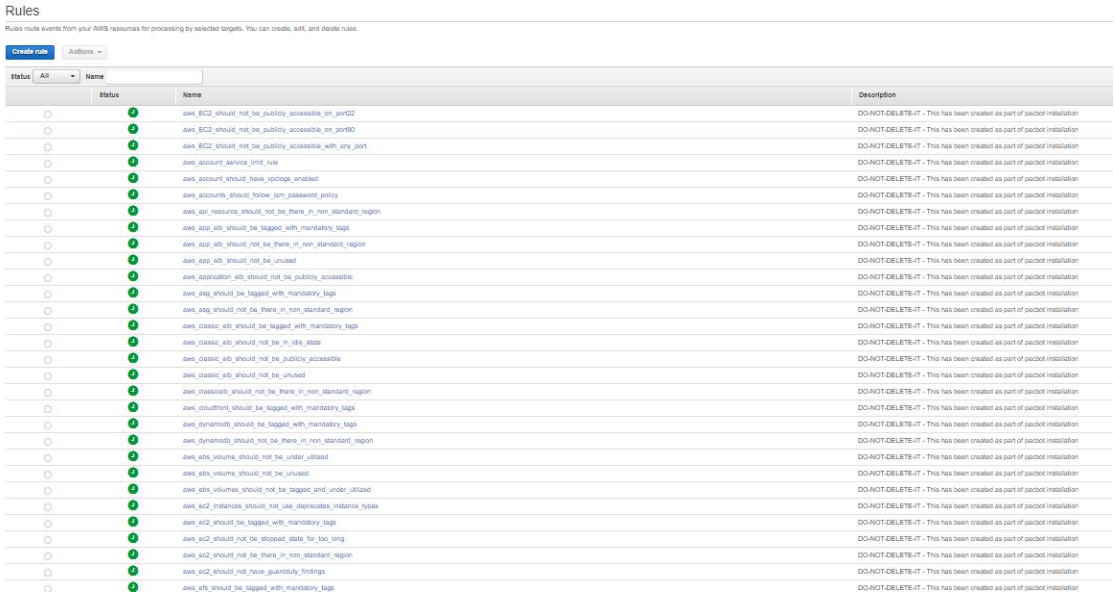
1. PacBot Rule Engine – Runs a predefined set of rules aganist the assets discovered by PacBot data collector
2. PacBot Data Collector – Set of batch jobs that run and discover resources in an AWS Account and stores the information

Screenshots from my PacBot Deployment in a sandbox environment:

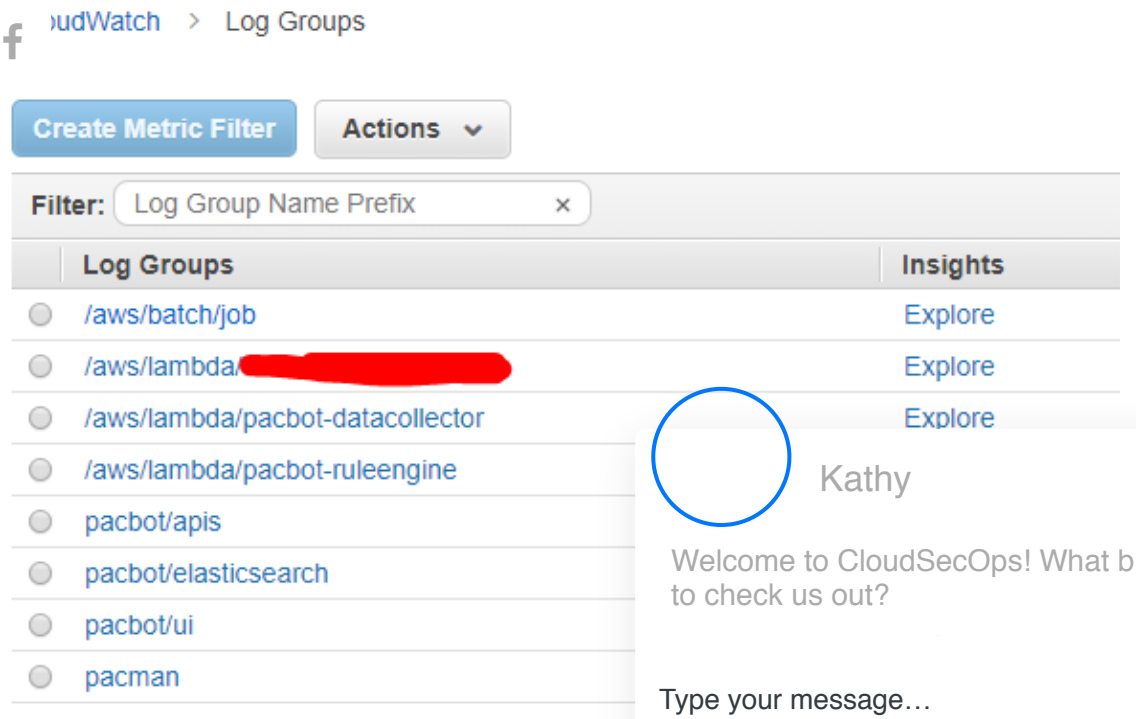


PacBot Compute Environment for running the Batch Jobs

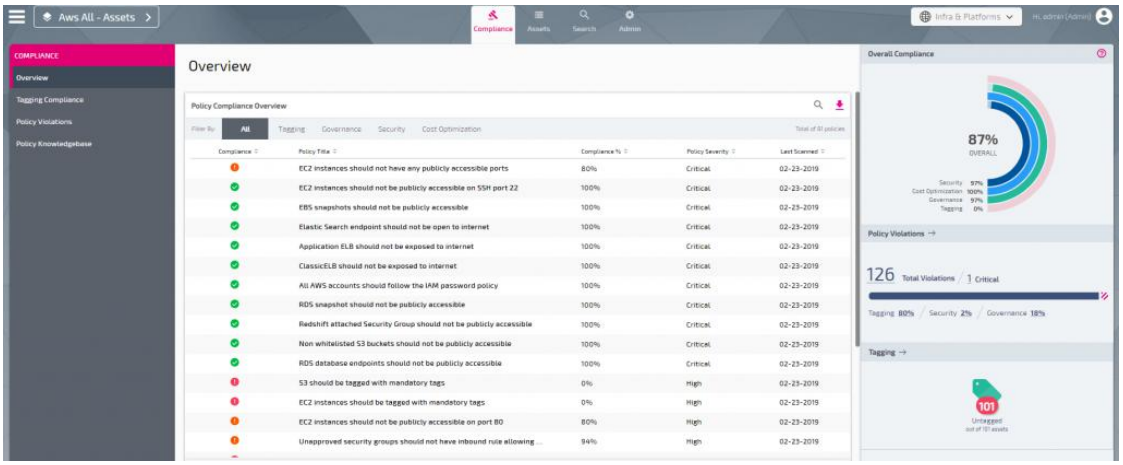
batch Jobs Dashboard



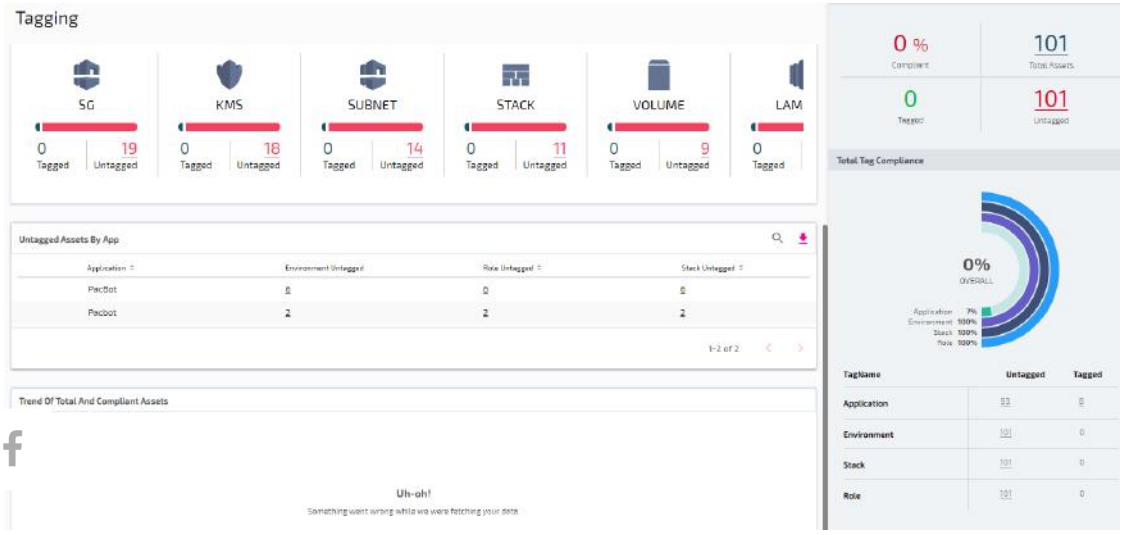
Snippet of a few PacBot CloudWatch Rules



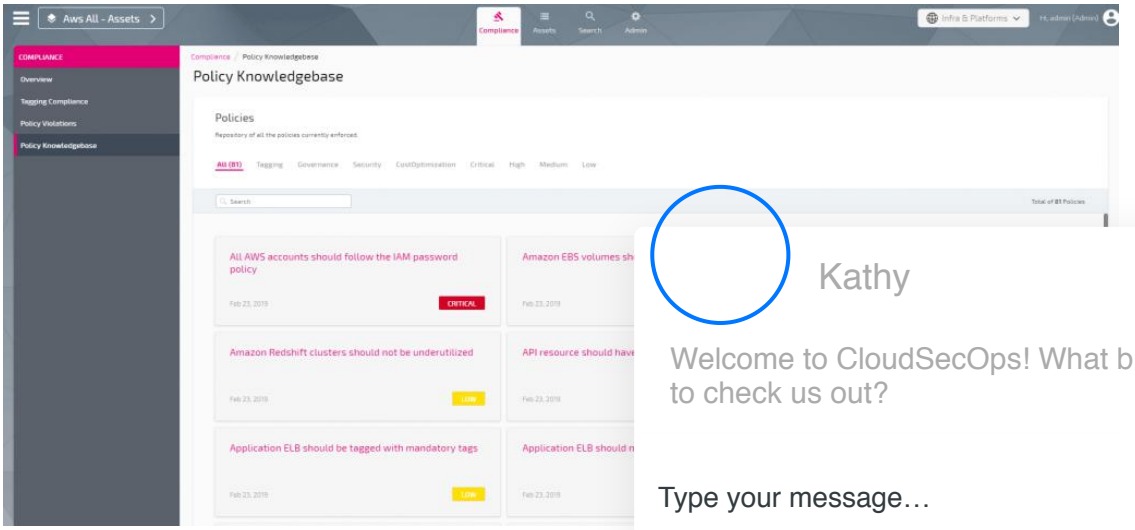
Logs f



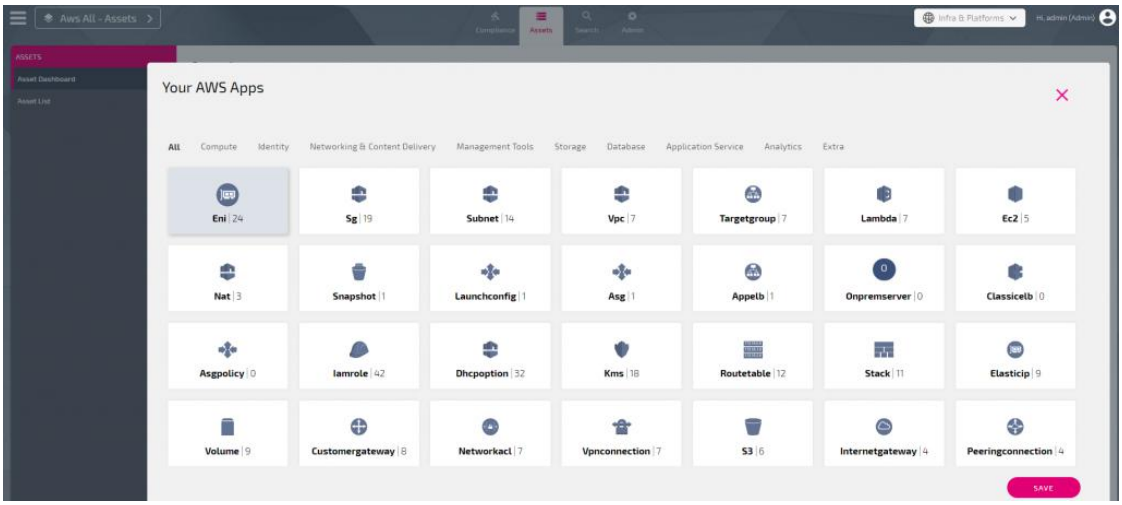
Asset Overview Board



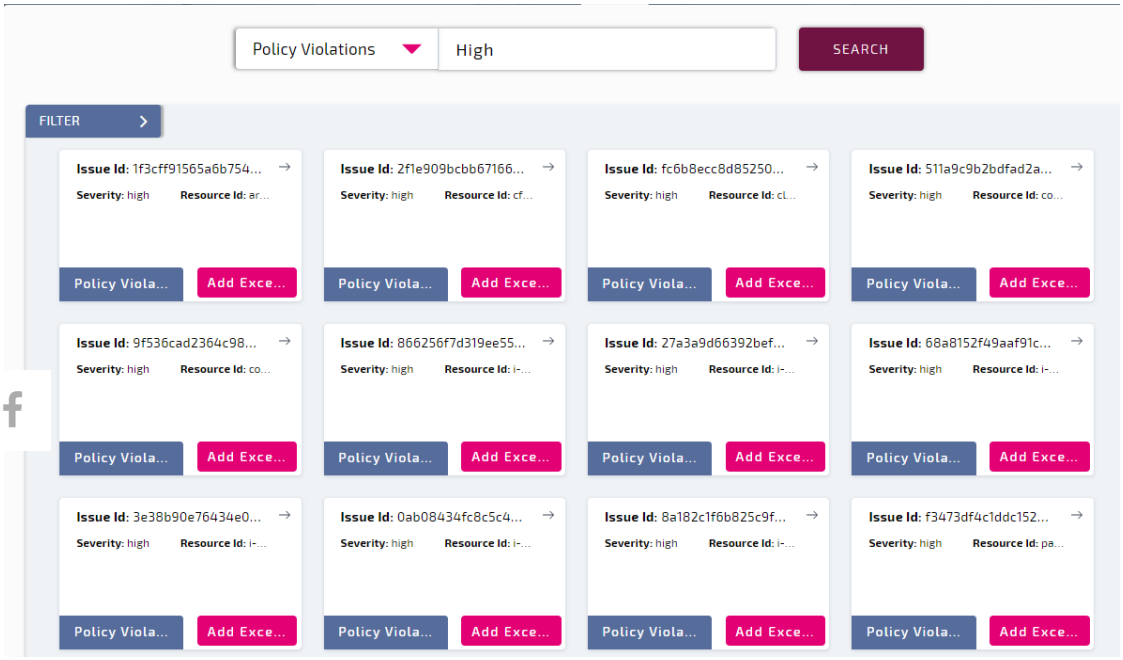
Tagging Compliance Dashboard(Facing Issues here, working on troubleshooting)



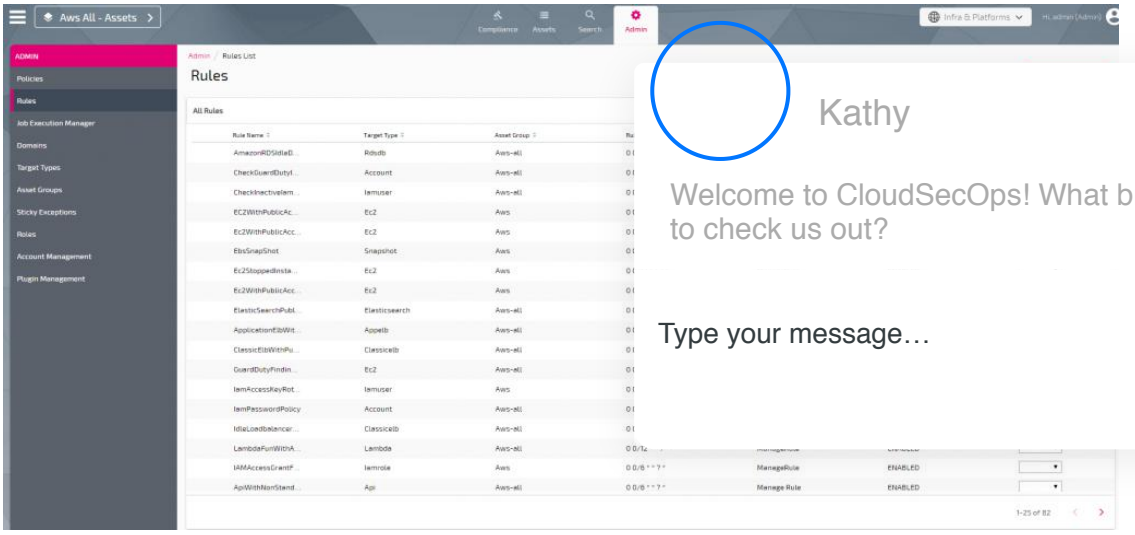
Polic



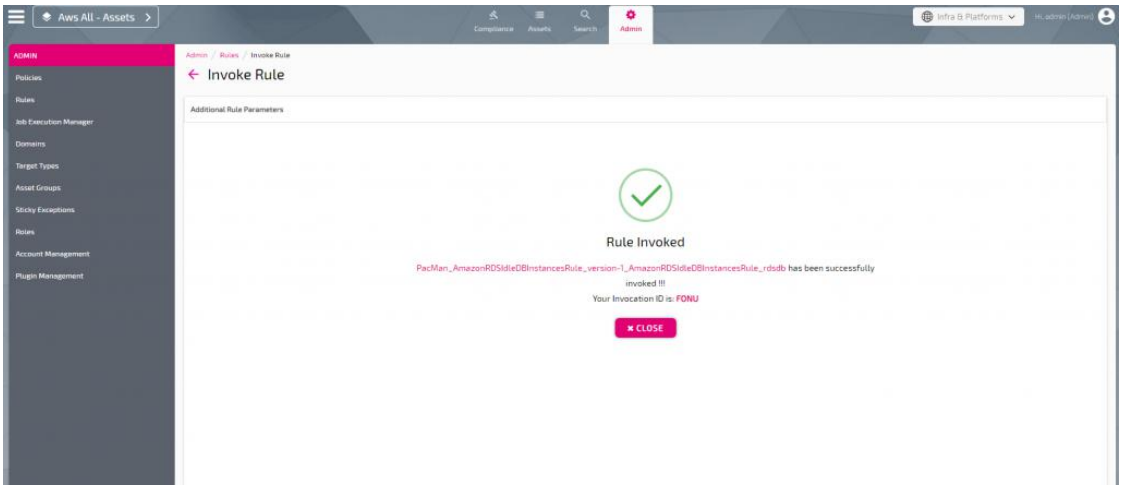
Assets Dashboard



Policy Violation Search Wizard



Admin – Panel – Compliance Rules



Invoking Ad-Hoc Rules

What are the currently available admin features?

- Create Asset Group
- Update Asset Group
- Delete Asset Group
- Rule\Policy Configuration
- Rule Troubleshooting

w to add multiple AWS Accounts to be monitored PacBot?

1. **IAM Role Changes** The account where PacBot is installed is called base account. The accounts that are monitored by PacBot is called client account.
    - Client Account Change: Create an IAM role named pacbot_ro and attach ReadOnlyAccess, AmazonGuardDutyReadOnlyAccess & AWSSupportAccess policies. Allow pacbot_ro from the base account to assume this role. Sample trust configuration for pacbot_ro role is here

```
1.  {   "Version":"2012-10-17",   "Statem
    "AWS":["arn:aws:iam::Base_Account_ID
    "Action":"sts:AssumeRole"       }]
```

1. Base Account Change: Fetch client acc
   policy which is associated with pacbot_

```
1.  {   "Version":"2012-10-17",   "State
    "Action":"sts:AssumeRole",     "Reso
    ["arn:aws:iam::Client_Account_ID_1:role/pacbot_ro","arn:aws:iam::Client_Account_ID_2:role/
    pacbot_ro"]     }] }
```

1. **Cloudwatch Rule Changes**

- Update "**accountinfo**" value (in *Constant (JSON text)* of cloudwatch rule) with new client account ids in cloudwatch rule named "**AWS-Data-Collector**". Sample configuration is `{"encrypt":false,"value":"Base_Account_ID,Client_Account_ID_1,Client_Account_ID_2","key":"accountinfo"}`

References:

https://github.com/tmobile/pacbot/wiki

My Personal Experiences:

- Batch Jobs fail several times while running policy engine rule, and asset collection. The fix would be going ahead and manually invoking the PacBot asset collector lambda function with the using the cloudwatch rule JSON payload
- Create a private hosted zone in AWS and map the internal ELB URL as a CNAME
- Add SSL certificate to Internal ELB and Map the targets with appropriate PacBot Services using Target Rules in ELB
- There will be a need for reverse DNS resolver that needs to be added to AWS Route 53 to access PacBot private hosted zone CNAME from you internal on-prem network.
- Change the default password of PacBot admin by performing a simple CRUD to PacBot RDS database, instructions are available in PacBot Wiki
- The instance which was used to build PacBot setup can be converted to a t2.micro for cost optimization. Rules can be configured based to run as per your organization time requirements to save the cost related to AWS Fargate/ECS/Batch Jobs.
- Several issues can occur during the deployment and during the bild destroy process, you can reach PacBot team and they are "VERY" good at providing some valuable suggestions to resolve the issues. Their Gitter chat URL: https://gitter.im/TMO-OSS/PacBot

Thank you for reading! – Setu Parimi & Steve

Sign up for the blog directly here.

Check out our professional services here.

Feedback is welcome! For professional servic
contact setu@cloudsecops.com

Share this:

Like this:

Loading...

Related

Security audit using Cloud Custodian for compliance in AWS
June 15, 2018
In "Amazon Web Services"

Post Exploitation in AWS using Nimbostratus
June 23, 2018
In "Cloud Penetration Testing"

Auditing AWS Environments for HIPAA Compliance
July 16, 2018
In "Cloud Security"

Categories: CLOUD SECURITY TOOLS

Tags: #compliance awssecurity monitoring

Leave a Reply

Name *

Email *

Website

What's on your mind?

Notify me of follow-up comments by email.

Notify me of new posts by email.

I'm not a spammer.

POST COMMENT

Search …

Subscribe

Always be the first to know news related to Cloud Security. We will never spam you.

Email Address

SUBSCRIBE

Categories

- Amazon Web Services
- Cloud Penetration Testing
- Cloud Security
- Cloud Security Tools
- Cloud Security Tutorial
- Uncategorized

Recent Posts

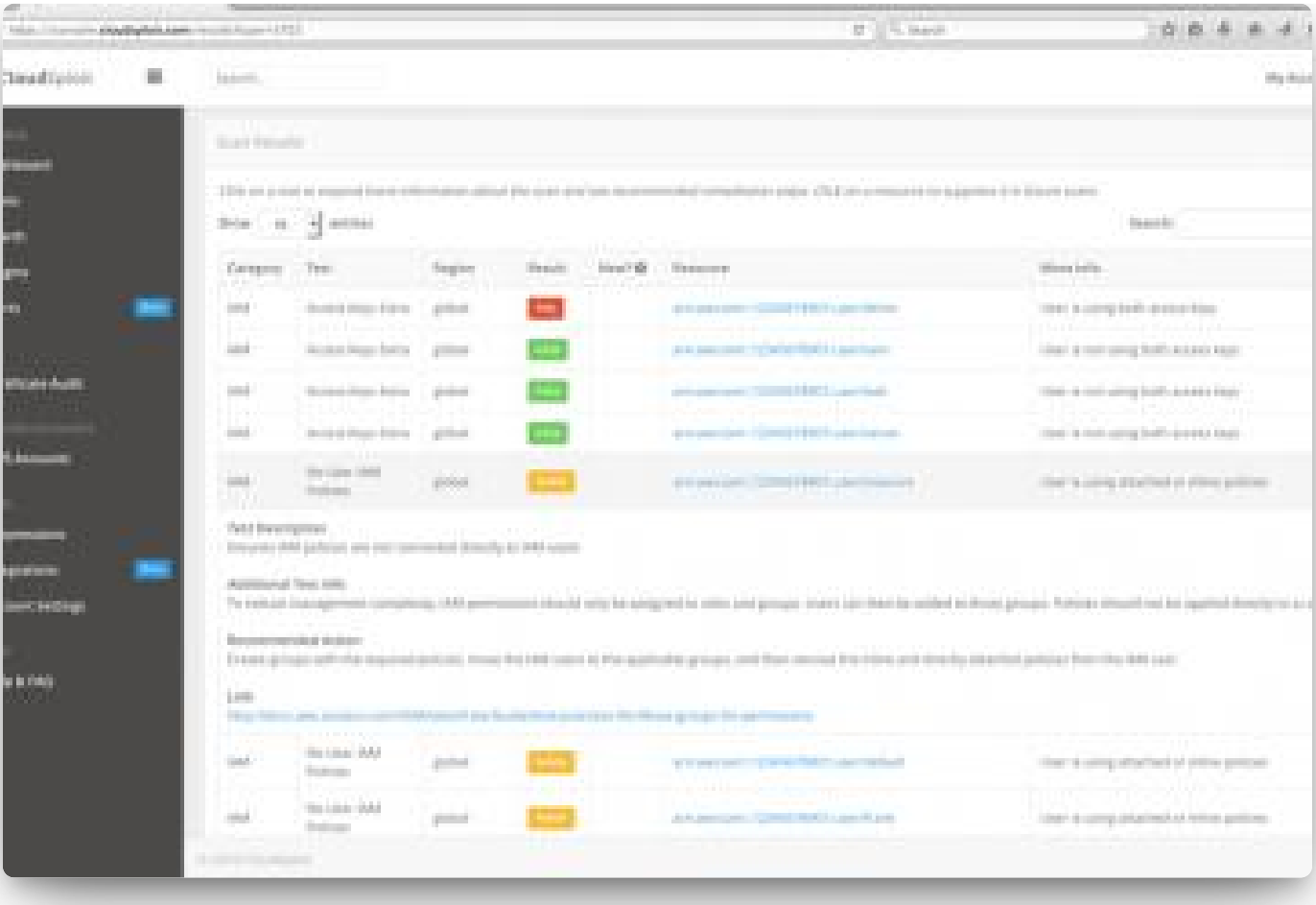
- PacBot – Open Source Compliance Automation Tool
- Automated Remediation for CloudTrail Disruption

Shifting strategy from DevOps to DevSecOps

Auditing AWS Environments for HIPAA Compliance

AWS Post Exploitation – Part 1

Related Posts



CLOUD SECURITY

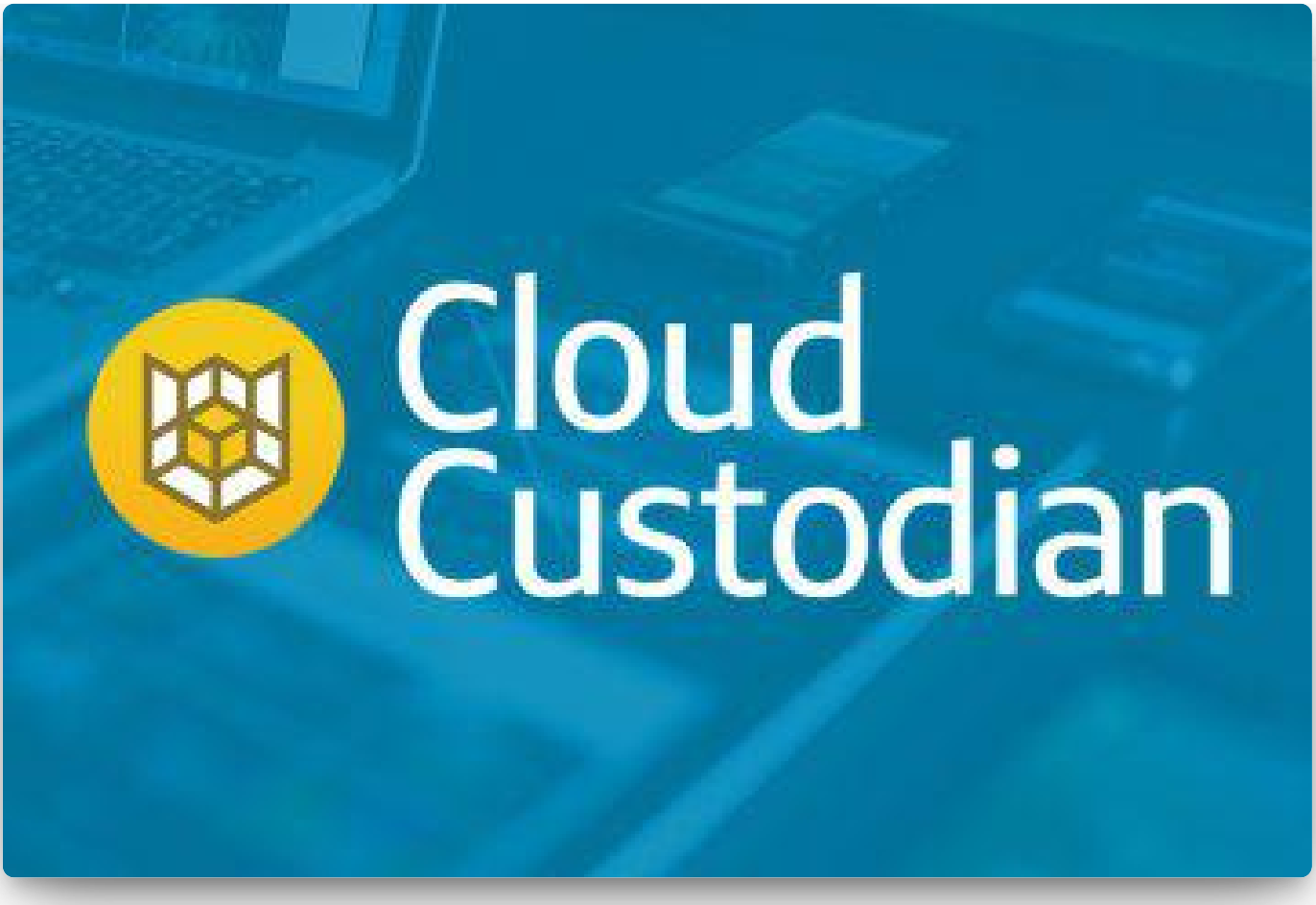
Auditing AWS Environments for HIPAA Compliance

Introduction CloudSploit is an AWS compliance, security and configuration monitoring scanner which is the first of its kind. It is an open source project designed to detect security risks in AWS. The CloudSploit Scans is built Read more…

CLOUD PENETRATION TESTING

Post Exploitation in AWS using Nimbostratus

Introduction Nimbostratus is a tool developed by Andres Riancho for fingerprinting and exploiting Amazon cloud infrastructures. Nimbostratus uses any application level HTTP proxy vulnerability to enumerate the instance and credentials from the metadata service which Read more...



AMAZON WEB SERVICES

Security audit using Cloud Custodian for compliance in AWS

Introduction In this article, we will be talking about Cloud Custodian, an open source rules engine for fleet management in AWS. The simple YAML DSL allows you to easily define rules to enable a well-managed

Read more…