Incident Response Playbook Template

Incident Type

GuardDuty Finding: PrivilegeEscalation-Kubernetes:PrivilegedContainer

Introduction

This playbook is provided as a template to customers using AWS products and who are building their incident response capability. You should customize this template to suit your particular needs, risks, available tools and work processes.

Security and Compliance is a shared responsibility between you and AWS. AWS is responsible for “Security of the Cloud”, while you are responsible for “Security in the Cloud”. For more information on the shared responsibility model, please review our documentation (https://aws.amazon.com/compliance/shared-responsibility-model/).

Summary

This Playbook

This playbook outlines response steps for incidents involving deployment of a privileged container. These steps are based on the NIST Computer Security Incident Handling Guide (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf) (Special Publication 800-61 Revision 2) that can be used to:

- Gather evidence
- Contain and then eradicate the incident
- Recover from the incident
- Conduct post-incident activities, including post-mortem and feedback processes

Interested readers may also refer to the AWS Security Incident Response Guide (https://docs.aws.amazon.com/whitepapers/latest/aws-security-incident-response-guide/welcome.html) which contains additional resources.

Once you have customized this playbook to meet your needs, it is important that you test the playbook (e.g., Game Days) and any automation (functional tests), update as necessary to achieve the desired results, and then publish to your knowledge management system and train all responders.

Note that some of the incident response steps noted in each scenario may incur costs in your AWS account(s) for services used in either preparing for, or responding to incidents. Customizing these scenarios and testing them will help you to determine if additional costs will be incurred. You can use AWS Cost Explorer (https://aws.amazon.com/aws-cost-management/aws-cost-explorer/) and look at costs incurred over a particular time frame (such as when running Game Days) to establish what the possible impact might be.

In reviewing this playbook, you will find steps that involve processes that you may not have in place today. Proactively preparing for incidents means you need the right resource configurations, tools and services in place that allow you to respond to an incident. The next section will provide a summary of this incident type, and then cover the five steps (parts 1 - 5) for handling privileged containers.

This Incident Type

An Amazon GuardDuty finding represents a potential security issue detected within your network. GuardDuty generates a finding whenever it detects unexpected and potentially malicious activity in your AWS environment. All GuardDuty finding references in this playbook will be related to the GuardDuty finding JSON that can be seen in the GuardDuty console, downloaded from the GuardDuty or Security Hub console, or exported to S3.

PrivilegeEscalation:Kubernetes/PrivilegedContainer (https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-kubernetes.html#privilegeescalation-kubernetes-privilegedcontainer)

A privileged container with root level access was launched on your Kubernetes cluster.

This finding informs you that a privileged container was launched on your Kubernetes cluster using an image has never before been used to launch privileged containers in your cluster. A privileged container has root level access to the host. Adversaries can launch privileged containers as a privilege escalation tactic to gain access to and then compromise the host.

Details on what resources are involved with this activity can be found in the finding details (https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_findings-summary.html#findings-resource-affected).

Incident Response Process

Part 1: Acquire, Preserve, Document Evidence

For Any Incident

1. You become aware of potential indicators of compromise (IoCs). These could come in various forms, but the original source is a GuardDuty finding:
    - An internal ticketing system (the sources of the ticket are varied and could include any of the means below)
    - From an alert in one of your monitoring systems either inside or external to AWS (that are ingesting GuardDuty Findings, in AWS, this could include AWS Security Hub)
    - Alarms or observations that resources have been created or deleted in your account that cannot be accounted for in your CMDB, exist in regions that you do not operate infrastructure in, or themselves have generated alerts (Amazon Detective (https://aws.amazon.com/detective/getting-started/) is a useful tool for understanding these relationships)
2. Confirm a ticket/case has been raised for the incident. If not, manually raise one
3. Determine and begin to document any end-user impact/experience of the issue. Findings should be documented in the ticket/case related to the incident
4. Open an incident war room using your standard communication tools, and page in the relevant stakeholders
5. In the case of automatically created tickets/cases, verify the finding in GuardDuty (what caused the ticket to be created?)

For This Incident Type

1. Identify the specific EKS cluster impacted. In GuardDuty, this will be in the Resource.EksClusterDetails.Name section of the finding.
2. Identify Pod, Node, and User information to be used in **Part 2**.
    - Pod information can be found in the Resource.KubernetesWorkloadDetails section of the GuardDuty finding.
    - Node information can be found after determing the pod name. Once you have the pod name you can run the command provided below to determine the node name.
        - kubectl get pods --namespace -o=jsonpath='{"\n"}'
    - User information can be found in the Resource.KubernetesDetails.KubernetesUserDetails section of the GuardDuty Finding.
3. Identify the origination information from which the Kubernetes APIs were called. In GuardDuty, Look at the Resource.Service.Action section of the finding
    - Verify that the IP address is valid for your enterprise users
    - Verify that the Location and/or ASN/Organization is known and valid for this request
4. If the principal that launched the privileged container is associated with a person in the organization, contact them to verify the launched privilege container is valid and was intended.
    - If the person states that they did launch the privilege container and the configuration change was intended, verify that there is a valid Change Management (CM) request or other authorization:
        - Once verified, move on to Part 5 to review the incident, and propose improvements that would stop or automatically archive findings for valid configuration changes within your organization.
        - If authorization for the change cannot be verified, communicate to stakeholders your intent to remediate or rollback the change and proceed to Part 2.
    - If the person states they did not launch the privileged container, proceed to **Part 2**.

Part 2: Contain the Incident

If you determine that the activity is unauthorized, or decide it is prudent to assume so, the first priority is to prevent further compromise without impact to production workloads.

1. Verify that disabling the Kubernetes user or isolating this pod will not result in a service outage.

2. If required, retain the user account for further forensic analysis by removing permissions from the Kubernetes User resource responsible for the activity using the appropriate steps below.

3. Built-in Kubernetes admin – The default user assigned by Amazon EKS to the IAM identity that created the cluster. This user type is identified by the user name kubernetes-admin.

    - To revoke access of a built-in Kubernetes admin:
        - Identify the userType from the Access Key details section.
            - If the userType is Role and the role belongs to an EC2 instance role:
                - Identify that instance then follow the instructions in Remediating a compromised EC2 instance (https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_remediate.html#compromised-ec2 (https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_remediate.html#compromised-ec2)).
            - If the userType is User, or is a Role that was assumed by a user:
                1. Rotate the access key of that user with the steps listed below: To rotate access keys for an IAM user without interrupting your applications (console)
                    1. While the first access key is still active, create a second access key.
                        - Sign in to the AWS Management Console and open the IAM console at (https://console.aws.amazon.com/iam/ (https://console.aws.amazon.com/iam/)).
                        - In the navigation pane, choose Users.
                        - Choose the name of the intended user, and then choose the Security credentials tab.
                        - Choose Create access key and then choose Download .csv file to save the access key ID and secret access key to a .csv file on your computer. Store the file in a secure location. You will not have access to the secret access key again after this closes. After you have downloaded the .csv file, choose Close. The new access key is active by default. At this point, the user has two active access keys.
                    2. Update all applications and tools to use the new access key.
                    3. Determine whether the first access key is still in use by reviewing the Last used column for the oldest access key. One approach is to wait several days and then check the old access key for any use before proceeding.
                    4. Even if the Last used column value indicates that the old key has never been used, we recommend that you do not immediately delete the first access key. Instead, choose Make inactive to deactivate the first access key.
                    5. Use only the new access key to confirm that your applications are working. Any applications and tools that still use the original access key will stop working at this point because they no longer have access to AWS resources. If you find such an application or tool, you can choose Make active to reenable the first access key. Then return to Step 3 and update this application to use the new key.
                    6. After you wait some period of time to ensure that all applications and tools have been updated, you can delete the first access key:
                        - Sign in to the AWS Management Console and open the IAM console at (https://console.aws.amazon.com/iam/ (https://console.aws.amazon.com/iam/)).
                        - In the navigation pane, choose Users.
                        - Choose the name of the intended user, and then choose the Security credentials tab.
                        - Locate the access key to delete and choose its X button at the far right of the row. Enter the access key ID to confirm the deletion and then choose Delete.
                2. Rotate any secrets that user had access to. Depending on where you stored your secrets will dictate what is the best process for rotation. If you are storing your secrets in the native secrets management capabilities of EKS use the kubernetes documentation to rotate your secrets (https://kubernetes.io/docs/concepts/configuration/secret/ (https://kubernetes.io/docs/concepts/configuration/secret/)). If you are using an external secrets store follow the product specific directions for secrets rotation, for example if you are using AWS Secrets Manager you can follow this documentation to rotate secrets (https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html (https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html)).
                3. Review the information in My AWS account may be compromised for further details (https://aws.amazon.com//premiumsupport/knowledge-center/potential-account-compromise/ (https://aws.amazon.com//premiumsupport/knowledge-center/potential-account-compromise/)).

4. To revoke access of an OIDC authenticated user, which is typically a user has an email address as a user name follow the steps below:

    - To check if your cluster uses OIDC use the following AWS CLI command: "aws eks list-identity-provider-configs --cluster-name **your cluster name**"
    - Rotate the credentials of that user in the OIDC provider.
    - Rotate any secrets that user had access to. Depending on where you stored your secrets will dictate what is the best process for rotation. If you are storing your secrets in the native secrets management capabilities of EKS use the kubernetes documentation to rotate your secrets (https://kubernetes.io/docs/concepts/configuration/secret/ (https://kubernetes.io/docs/concepts/configuration/secret/)). If you are using an external secrets store follow the product specific directions for secrets rotation, for example if you are using AWS Secrets Manager you can follow this documentation to rotate secrets (https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html (https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html)).

5. To revoke access of an AWS ConfigMap user:

    - Use the following command to open the ConfigMap.

```
kubectl edit configmaps aws-auth --namespace kube-system
```

6. Identify the role or user entry under the mapRoles or mapUsers section with the same user name as the one reported in the Kubernetes user details section of your GuardDuty finding. See the following example, where the admin user has been identified in a finding.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::444455556666:role/eksctl-my-cluster-nodegroup-standard-wo-NodeInstanceRole-1WP3NUE3O6UCF
      user name: system:node:EC2_PrivateDNSName
      groups:
        - system:bootstrappers
        - system:nodes
  mapUsers: |
    - userarn: arn:aws:iam::123456789012:user/admin
      username: admin
      groups:
        - system:masters
    - userarn: arn:aws:iam::111122223333:user/ops-user
      username: ops-user
      groups:
        - system:masters
```

1. Remove that user from the ConfigMap. See the following example where the admin user has been removed.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/eksctl-my-cluster-nodegroup-standard-wo-NodeInstanceRole-1WP3NUE3O6UCF
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
  mapUsers: |
    - userarn: arn:aws:iam::111122223333:user/ops-user
      username: ops-user
      groups:
        - system:masters
```

1. If the finding does not have a "accessKeyDetails" section, the user is a Kubernetes service account. The service account provides an identity for pods and can be identified by a user name with the following format: system:serviceaccount:namespace:service_account_name. To revoke access to a service account:

- Rotate the service account credentials to do this you will need to rotate the IAM role credentials that were assigned to the service account with the steps below.
    1. While the first access key is still active, create a second access key.
        - Sign in to the AWS Management Console and open the IAM console at (https://console.aws.amazon.com/iam/ (https://console.aws.amazon.com/iam/)).
        - In the navigation pane, choose Users.
        - Choose the name of the intended user, and then choose the Security credentials tab.
        - Choose Create access key and then choose Download .csv file to save the access key ID and secret access key to a .csv file on your computer. Store the file in a secure location. You will not have access to the secret access key again after this closes. After you have downloaded the .csv file, choose Close. The new access key is active by default. At this point, the user has two active access keys.
    2. Update all applications and tools to use the new access key.
    3. Determine whether the first access key is still in use by reviewing the Last used column for the oldest access key. One approach is to wait several days and then check the old access key for any use before proceeding.
    4. Even if the Last used column value indicates that the old key has never been used, we recommend that you do not immediately delete the first access key. Instead, choose Make inactive to deactivate the first access key.
    5. Use only the new access key to confirm that your applications are working. Any applications and tools that still use the original access key will stop working at this point because they no longer have access to AWS resources. If you find such an application or tool, you can choose Make active to reenable the first access key. Then return to Step 3 and update this application to use the new key.
    6. After you wait some period of time to ensure that all applications and tools have been updated, you can delete the first access key:
        - Sign in to the AWS Management Console and open the IAM console at (https://console.aws.amazon.com/iam/ (https://console.aws.amazon.com/iam/)).
        - In the navigation pane, choose Users.
        - Choose the name of the intended user, and then choose the Security credentials tab.
        - Locate the access key to delete and choose its X button at the far right of the row. Enter the access key ID to confirm the deletion and then choose Delete.

2. If the user has write privileges, it is recommended to audit all changes made by the user in question. This can be accomplished by querying the EKS audit logs in CloudWatch Logs (https://console.aws.amazon.com/cloudwatch/home#logsV2:log-groups).

    - Locate the Log group for the relevant cluster (you noted this in Part 1) under `Logs >> Log groups` in the CloudWatch console. They will start with /aws/eks/cluster_name/audit
    - Look for unusual entries in Audit logs
    - You can also use CloudWatch Log Insights to query CloudWatch logs. An example can be found below that looks for create, update, and delete operation to ClusterRoleBindings. You can find more examples at (https://aws.github.io/aws-eks-best-practices/security/docs/detective/#analyze-logs-with-log-insights (https://aws.github.io/aws-eks-best-practices/security/docs/detective/#analyze-logs-with-log-insights))

3. fields @timestamp, @message | sort @timestamp desc | limit 100 | filter objectRef.resource="clusterrolebindings" and verb in ["create", "update", "patch", "delete"]

4. Continue to provide updates to incident stakeholders on the current state of the incident response.

5. Now that we have contained the user specified in the GuardDuty finding next we will want to contain the pod associated with the finding. Use the sections below to isolate the Pods.

    - A deny all traffic rule may help stop an attack that is already underway by severing all connections to the pod. The following Network Policy will apply to a pod with the label app=web.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  - Egress
Attention
```

1. A Network Policy may prove ineffective if an attacker has gained access to underlying EC2 worker node. If you suspect that has happened, you can use AWS Security Groups to isolate a compromised worker node from other worker nodes. When changing a worker nodes security group, be aware that it will impact all containers running on that worker node.

2. Next we need to cordon the worker node. By deploying a privileged container we have given a container root level permissions to the worker node potentially allowing that container to manipulate the underlying system. By cordoning the impacted worker node, you're informing the scheduler to avoid scheduling pods onto the affected node. This will allow you to remove the node for forensic study without disrupting other workloads.

3. Enable termination protection on impacted worker node - An attacker may attempt to erase their misdeeds by terminating an affected node. Enabling termination protection can prevent this from happening. Instance scale-in protection will protect the node from a scale-in event. To do this follow the steps below.

    - Open the Amazon EC2 console at (https://console.aws.amazon.com/ec2/ (https://console.aws.amazon.com/ec2/)).
    - Select the instance, and choose Actions, Instance Settings, Change Termination Protection.
    - Choose Yes, Enable.

4. Label the offending Node with a label indicating that it is part of an active investigation - This will serve as a warning to cluster administrators not to tamper with the affected Pods/Nodes until the investigation is complete. To apply a label to a node use the command below.

    - kubectl label nodes <your-node-name> <key>=<value>

5. Capture volatile artifacts on the worker node¶

    - Capture the operating system memory. This will capture the Docker daemon and its subprocess per container. MargaritaShotgun (https://github.com/ThreatResponse/margaritashotgun (https://github.com/ThreatResponse/margaritashotgun)), a remote memory acquisition tool, can aid in this effort.
    - Perform a netstat tree dump of the processes running and the open ports. This will capture the docker daemon and its subprocess per container.
    - Run docker commands before evidence is altered on the worker node.
        - `docker container top CONTAINER` for processes running.
        - `docker container logs CONTAINER` for daemon level held logs.
        - `docker container port CONTAINER` for list of open ports.
        - `docker container diff CONTAINER` to capture changes to files and directories to container's filesystem since its initial launch.
    - Pause the container for forensic capture.
    - Snapshot the instance's EBS volumes. Steps to do this below.
        - Open the Amazon EC2 console at (https://console.aws.amazon.com/ec2/ (https://console.aws.amazon.com/ec2/))
        - In the navigation pane, choose Snapshots, Create snapshot.
        - For Resource type, choose Volume.
        - For Volume ID, select the volume from which to create the snapshot.The Encryption field indicates the selected volume's encryption status. If the selected volume is encrypted, the snapshot is automatically encrypted using the same KMS key. If the selected volume is unencrypted, the snapshot is not encrypted.
        - (Optional) For Description, enter a brief description for the snapshot.
        - (Optional) To assign custom tags to the snapshot, in the Tags section, choose Add tag, and then enter the key-value pair. You can add up to 50 tags.
        - Choose Create snapshot.

Part 3: Eradicate the Incident

This is the stage for taking remedial action to minimize the impact of the unintended activities.

1. Using your preferred monitoring tool, access CloudTrail and search for all API actions performed by the AWS EKS resource in the last 90 days:
    1. If this tool is a third-party tool such as Splunk, Sumo Logic or others, follow the normal procedure for obtaining log information from that tool
    2. If you do not ingest CloudTrail logs into a third-party tool, but do send those logs to Amazon Simple Storage Service (Amazon S3), you will be able to use Amazon Athena to query the logs. The remaining steps will focus on AWS tools to retrieve the necessary information
2. Create an Athena table referencing the bucket containing your CloudTrail logs (https://aws.amazon.com/premiumsupport/knowledge-center/athena-tables-search-cloudtrail-logs/) that link also includes example queries that can be run in Athena
3. In the Athena console, run a query that shows all API actions taken by the compromised resources post-compromise date/time. From the resulting list, determine which API calls:
    - Accessed sensitive data (such as S3 GetObject)

    NOTE to get S3 data events CloudTrail must be configured to collect S3 data events prior to the incident. More information about CloudTrail data events can be found here (https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-data-events-with-cloudtrail.html)

    - Created new AWS resources, such as databases, Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS Lambda functions or S3 buckets, etc.
    - Services that create resources should also be carefully checked; for example, CloudFormation Stacks/StackSets, AWS Firewall Manager Security Policies, AWS Elastic Beanstalk resources, etc.
    - Created or modified AWS identity resources that could be used to extend a foothold into the account (or other accounts, for example with AWS Security Token Service (AWS STS) API methods such as AssumeRole). Within an account, API methods including (but not limited to) the following should also be investigated:
        - CreateUser
        - CreateRole
        - AssumeRole*
        - Get*Token
        - Attach*Policy
        - RunInstances (especially with PassRole)
        - *Image*
        - *Provider
        - Tag*
        - Untag*
        - Create*
        - Delete*
        - Update*
        - etc.
    - Deleted existing AWS resources
    - Modified existing AWS resources
4. Based on the results of the previous step, determine if any applications are potentially impacted:
    - Obtain the ARN and/or tag information for each resource impacted (from step 4, above)
    - Go back to your CMDB and determine which application that resource belongs to
    - Notify the application owner based on the results of the above steps
5. At this point you will need to follow the appropriate playbook based on the type of application impact or further compromised resources to determine full impact and contain any further compromise.

Part 4: Recover from the Incident

1. After idenityfing, containing, and eradicating any involved pods, nodes, users, and any other AWS resources follow the steps below to complete recovery (note that some steps below could also be considered as eradication).
2. For resources that were modified during the compromise:
    1. If the resource can be destroyed and replaced, do so. For example, Identify the vulnerability that compromised the pods, implement the fix for that vulnerability and start new replacement pods, and then delete the vulnerable pods.
    2. If the resource cannot be replaced, either:

    1. Restore the resource from a known good backup, or;
    2. Prepare a new resource and configure it into the application's infrastructure, while isolating the compromised resource and removing it from the application's infrastructure. Update your CMDB accordingly
    3. Either destroy the compromised resource, or continue to leave it isolated for post-incident forensics

3. For resources that were deleted during the compromise:

    1. Determine what (if any) application the resource belonged to, by checking your CMDB, or confirming the resource's tag(s) (Check AWS Config if the tags aren't listed in the CloudTrail entry and the resource is supported by AWS Config (https://docs.aws.amazon.com/config/latest/developerguide/resource-config-reference.html))
    2. If the deleted resource can be restored from backup, commence the restore procedure
    3. If the deleted resource cannot be restored from backup, consult your CMDB to obtain the resource's configuration, and recreate the resource and configure it into the application's infrastructure

4. For resources that were created during the compromise:

    1. Confirm these have been deleted or isolated for further analysis, per the steps in **Part 2**.

Part 5: Post-Incident Activity

This activity contains two parts. Firstly, some compromised resources may require forensic analysis, either to fulfil regulatory obligations or improve incident handling, both taking input from the root cause analysis that will result from forensic investigation. The second part is a "sharpen the saw" activity which helps teams to assess their response to the actual incident, determine what worked and what didn't, update the process based on that information and record these findings.

Firstly, perform any required forensic investigation to determine (for compromised resources) what methods the actors may have used and to determine if additional risks and risk mitigations are required for the resources and/or applications in question.

1. For any compromised resources that have been isolated for further analysis, perform the forensic activity on those resources and incorporate the findings into the post-incident report.
2. Ensure that the CMDB is correctly updated to reflect the current status of all resources and applications impacted
3. Once the analysis is completed, ensure relevent resources are terminated and the resulting data and any artifacts from the analysis maintained and/or entered into the relevant system for record-keeping

Secondly, review the incident itself and the response to it, to determine if anything needs to be changed for handling any similar incidents in the future.

1. Review the incident handling and the incident handling process with key stakeholders identified in Part 1, Step 8
2. Document lessons learned, including attack vector(s) mitigation(s), misconfiguration, etc.
3. Store the artifacts from this process with the application information in the CMDB entry for the application and also in the CMDB entry for the credential compromise response process.

Additional Resources

Incident Response

https://aws.amazon.com/blogs/security/how-get-started-security-response-automation-aws/ (https://aws.amazon.com/blogs/security/how-get-started-security-response-automation-aws/)
https://docs.aws.amazon.com/whitepapers/latest/https://docs.aws.amazon.com/whitepapers/latest/aws-security-incident-response-guide/welcome.html (https://docs.aws.amazon.com/whitepapers/latest/https://docs.aws.amazon.com/whitepapers/latest/aws-security-incident-response-guide/welcome.html)
https://aws.amazon.com/blogs/security/how-to-automate-incident-response-in-aws-cloud-for-ec2-instances/ (https://aws.amazon.com/blogs/security/how-to-automate-incident-response-in-aws-cloud-for-ec2-instances/)
https://aws.amazon.com/blogs/security/forensic-investigation-environment-strategies-in-the-aws-cloud/ (https://aws.amazon.com/blogs/security/forensic-investigation-environment-strategies-in-the-aws-cloud/)
https://aws.amazon.com/blogs/security/how-to-automate-forensic-disk-collection-in-aws/ (https://aws.amazon.com/blogs/security/how-to-automate-forensic-disk-collection-in-aws/)

https://aws.github.io/aws-eks-best-practices/security/docs/incidents/ (https://aws.github.io/aws-eks-best-practices/security/docs/incidents/)
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty-remediate-kubernetes.html (https://docs.aws.amazon.com/guardduty/latest/ug/guardduty-remediate-kubernetes.html)

GuardDuty

https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_findings-summary.html (https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_findings-summary.html)

Other

https://docs.aws.amazon.com/quicksight/latest/APIReference/API_Reference.html (https://docs.aws.amazon.com/quicksight/latest/APIReference/API_Reference.html) https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html (https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html)